U.PORTO

FEP FACULDADE DE ECONOMIA
UNIVERSIDADE DO PORTO

HEURISTICS FOR THE SINGLE MACHINE WEIGHTED QUADRATIC TARDINESS SCHEDULING PROBLEM WITH UNEQUAL RELEASE DATE

James Teddy Limousin Sambuchetti

Dissertation

Master in Modeling, Data Analysis and Decision Support System in Optimization

Supervised by

Professor Jorge Miguel Silva Valente

Professor Jeffrey Schaller

2018

Biographic Note

James Teddy Limousin Sambuchetti was born on January 7th, 1986, in Asuncion, Paraguay. He finished his studies in Computer Science Engineering in 2012. In 2016, he obtained the possibility to join the Data Analytics Master at the University of Porto, thanks to the Erasmus Mundus Scholarship in the context of the Euroinka Project supported by the Manchester Metropolitan University.

Resumo

Hoje em dia, a capacidade de uma empresa entregar produtos no momento certo ou acordado é algo que se traduz numa vantagem competitiva. Em ambientes de negócios altamente competitivos, esta capacidade traduz-se numa mais valia para a empresa, na medida em que assegura atempadamente as necessidades dos clientes, daí resultando uma imagem de qualidade e profissionalismo.

Em cenários reais, a eficaz execução de planos é fundamental para o nível de sucesso. Seja no contexto da indústria fabril, de companhias farmacêuticas, de empresas de telecomunições ou mesmo de administração de aeroportos, a eficácia do processo de sequenciamento tem um impacto considerável no serviço final prestado.

Este trabalho considera um problema de sequenciamento com uma única máquina, no qual o objetivo é otimizar o sequenciamento de operações, nomeadamente ao nível do atraso das entregas. De facto, a função objetivo consiste em minimizar a soma ponderada do quadrado do atraso das entregas. Os trabalhos ou operações têm datas de disponibilidade distintas, e serão utilizadas diversas regras de despacho para efetuar o seu sequenciamento. Estas heurísticas incluem não apenas versões lineares (ou seja, desenvolvidas para o problema no qual o atraso não é elevado ao quadrado), mas também versões que têm em consideração o facto da função objetivo ser quadrática.

Os resultados mostram que as regras que consideram de forma explícita o caráter quadrático da função objetivo têm um desempenho significativamente superior ao das regras lineares, independentemente da dimensão do problema. Adicionalmente, a comparação entre as estratégias de sequenciamento ativo e sem atrasos (non-delay) mostra que as regras que utilizam a estratégia sem atrasos permitem a obtenção de um melhor desempenho, tanto ao nível da função objetivo, como no que se refere ao tempo total necessário para processar todos os trabalhos.

.

Abstract

Nowadays the ability of a company to deliver products at the right moment gives them competitive advantage in the market. When dealing with crowded environments it allows the possibility to make room and position the brand by surprising clients with the new approach; and when trying to innovate with a new idea, it gives the image of professionalism and quality in the product.

In real world situations it is the execution of the plan that determines the level of success. Whether in the context of goods production industrial facilities, pharmaceutical companies, telecommunication scenarios or even airport management, the effectiveness of the scheduling process makes a great impact on the final provided service.

The focus of this work is the Single Machine Scheduling Problem where the main goal is to optimize the scheduling of jobs in a single machine by reducing the cost of late delivery. The objective function takes in consideration the Weighted Squared Tardiness of jobs with unequal release dates, using different dispatching rules heuristics. The linear version of these procedures will be taken in consideration as well as the adapted versions suitable for the quadratic objective function.

The final results indicate that the rules that take in consideration the quadratic objective function greatly outperforms the linear versions regardless of the size of the problem. Also, the comparison between active and non-delay scheduling strategies is presented, where non-delay strategies outperforms the active, showing better solutions in terms of quality, that is the value of the objective function, and in terms of total time required to process the job set instances, also named makespan.

Table of Contents

List of Tables

Chapter 1 - Introduction

1.1 Motivation

The concept of scheduling jobs is a global idea that can be applied to almost any business process, whether it is setting up the way an assembling machine has to work in a factory or the different stages the development of an application has to go through. The organization of the work in order to achieve the efficient and effective use of resources is a never ending task that professionals deal with every day.

When working on modeling reality we often find that the complexity of real world situations has many moving parts, so when trying to apply exact methods to find the optimal solution we came to the conclusion that the complexity makes it unpractical when considering the cost in time and resources to find the best outcome.

This is one of the reasons why using heuristic approaches to solve problems is so important: perfect solutions are not always found in a reasonable amount of time, and usually they do not apply to real life situations, since we must take into consideration the human factor all the time.

This work tackles the use of heuristic algorithms to schedule jobs on a single machine which is an approach that can be applied to generalize many different concepts. In this scenario a job is a single operation performed only by one machine without any precedence constraint between jobs, which means no job requires a previous job to be finished before it can start. Another point to mention is that idle time may or may not be considered between two consecutives jobs, this will be depending on the specific case and dispatching rule.

With the proposed objective function, we reinforce the impact on the cost that each unit of work has on the overall performance of the system. This means that some level of delay can be acceptable but considerable deviation is heavily penalized, since in a commercial scenario, the client relation may be affected, with a severe impact on the business.

1.2 Problem Proposition

In this dissertation, we study the single machine scheduling problem with total weighted quadratic tardiness cost of jobs with different release dates.

The problem can be stated as a set of *n* jobs (1, 2, …, n) that have to be scheduled on a single machine, which can handle only one job at a time and is available continuously. Jobs have a release date r_j and can not be started before it, they require a processing time p_j, have a weight w_j and should ideally be completed before their due date d_j. The setup time of jobs is not considered as well as any dependency between them.

The tardiness for a job *j* is defined as $T_j = \max\{0, C_j - d_j\}$, where C_j is the completion time of job *j*. The objective is to find a feasible schedule with minimum total weighted squared tardiness which can be denoted as $1|r_j|\sum w_j T_j^2$. The expression combines the weight and the tardiness of jobs that have a release date.

The use of a single machine in a practical scenario may not seem very common; there are works in literature with real scenarios as is the case of (Wagner, Davis, & Kher, 2002) where the author presents a case in the chemical industry. Additionally, the are many production systems that sometimes depend on a single critical bottleneck component which can impact in the overall performance.

Besides practical cases, insights and results for single machine models usually prove useful when deriving procedures for more sophisticated production environments as shown recently in areas such as data streaming (Yang et al., 2016), sequencing arrival schedule for aircrafts (Zhang, Zheng, & Ge, 2017) and video streaming in maritime scenarios (Tingting et al., 2017).

The problem with the identical release dates and a linear objective function, denoted as $1||\sum w_j T_j$, is known to be NP-hard in the strong sense as shown in (Kan, 1976). Based on the previous statement, the objective function that we propose is still open, but it seems likely that it may also be NP-hard.

The objective function for the proposed problem is based on the tardiness of a job, which in practical scenarios is a common measure for scheduling activities due to the importance of delivering goods and services on time, thus preventing penalties in contracts or any other type of inconvenience with the end customer that may appear. In the literature there

are many works that tackle the single machine scheduling problem using total linear tardiness or the maximum tardiness which makes them usual measures.

When considering total linear tardiness, $1||\sum T_j$, the objective function does not take into account the possible concentration of the tardiness in only a few jobs as opposed to a more balanced solution where the tardiness is shared evenly among a larger number of jobs. When comparing these scenarios with a squared tardiness objective function, the resulting schedules will prevent the occurrence of completing jobs with high values of tardiness since the impact on the cost will be severe, as it is mentioned in (Sun, Noble, & Klein, 1999).

As per the use of the maximum tardiness objective function $1||T_{max}$ and also mentioned in (Thomalla, 2001), the solution focuses on preventing schedules with a large delay, without taking into consideration the possible existence of an overall tardiness in most or all jobs, which gets disregarded. Each of these tardiness measures have its own application scenarios, and it cannot be stated that any one is always better than the others. Indeed, the choice of measure is to be made by the decision maker according to the situation and preferences at hand.

Another consideration regarding the impact of using squared tardiness can be compared to the work of (Hoitomt, Luh, Max, & Pattipati, 1990) (Hoitomt et al., 1990; Thomalla, 2001), where the authors state that when using linear tardiness, the penalty for the increase in a job's tardiness does not change the final result, since having two jobs with ten units of lateness is the same as having one with twenty. When using the quadratic measure, however, having ten jobs with only one unit of tardiness has a lower penalty than having one job with a tardiness of ten.

This is a valid observation to be considered when dealing with client relations in any company. Indeed, (Taguchi, 1986) in his work proposes that the dissatisfaction of clients increases quadratically and not linearly with the tardiness, indicating that this objective function has real world applications.

The solutions will be generated using two types of scheduling strategies, which are active and non-delay, together with multiple dispatching rules that will consider the linear and quadratic version of the objective function in order to perform the analysis.

1.3 Contents

This work tackles the analysis of the use of different dispatching rules applied to the single machine scheduling problem using a particular objective function, weighted squared tardiness with jobs that have release dates. The different stages of the investigation will be described in the following chapters together with results and final conclusions from the study. The order and content of each chapter will be described below.

In Chapter 2 we present the literature review. Since to the best of our knowledge there is only one previous work that considers the scheduling with a weighted quadratic tardiness function with release dates, we will be including the ones that do not consider jobs with release dates since most of them can be adapted and they also provide insights and heuristics that can be used in the version that we are proposing. In addition, we will consider the works that use linear tardiness with release dates due to the fact that they can provide insights and on how to develop heuristics for our problem.

Next, in Chapter 3 we detail the two types of scheduling strategies that we will be using to generate the solutions using the dispatching rules, along with this, the pseudo code that is used to solve the instances will be presented in order to provide a clear view of the process.

Chapter 4 will introduce the description and characteristics of the dispatching rules that will be used in the analysis. Also the necessary adaptation of each one to consider the objective function will be added.

Chapter 5 describes how we generated instance sets that will be used for parameter adjustments and for the computational evaluation of the procedures. The technology environment were we will run the algorithms and the special parameters that need to be considered in the dispatching rules are described. Finally, the preliminary tests to get the values of the parameters that are required for some of the procedures are indicated.

In Chapter 6 the computational results of the dispatching rules in the different schedule types will be shown. The rules are compared using the performance measure for different parameter combinations in order to provide the level of effect of each component of the schedule types. As a final analysis the best rules of each type of schedule will be reviewed to determine the best overall heuristics and the difference in performance that was obtained in the experiment.

Finally, in Chapter 7 we do a recapitulation of the work made, together with the main conclusions that were obtained from the results of the study and the future work that can be foreseen following the steps of the experiment as a way to improve the solutions.

Chapter 2 - Literature Review

The literature review for this work includes the analysis of different academic works and books focusing on the use of the single machine scheduling problem as the main approach for the solutions. This is due to the fact that several works include the scenario of having different release dates, and others with similar objectives functions that are aligned with our interests.

Between the objective functions considered are: the one proper of this study which is the weighted squared tardiness; we also added versions of the problem that do not consider release dates; we consider the relevance of the problems with squared earliness and tardiness, on jobs without release dates, as a way to add a special approach to the problem and the linear version of the objective function, which is weighted tardiness with jobs that have release dates.

2.1 Weighted Squared Tardiness

In the literature regarding the scheduling problem denoted as $1|r_j|\sum w_j T_j^2$, and as mentioned before to the best of our knowledge there is only one work that considers the same objective function (Sun et al., 1999), however in this paper the jobs have the constraint of sequence dependent setup times. They present a solution using a Lagrangian relaxation method by decomposing the problem and then applying a sub-gradient technique. The results were then compared using the EDD (Earliest Due Date) and ATC (Apparent Tardiness Cost) dispatching heuristics, suitably adapted for the problem at hand, and meta-heuristics like the four way swap Local Search, Tabu Search and Simulated Annealing (SA). Their results indicate that the Lagrangian relaxation method outperforms most heuristics and generates almost identical solutions as the SA with a better efficiency.

Other similar examples of using the same objective function, but considering only jobs with identical release dates, denoted as $1||\sum w_j T_j^2$, are (Schaller & Valente, 2012; Valente & Schaller, 2012; Gonçalves, Valente, & Schaller, 2016). (Schaller & Valente, 2012) present a comparison of two Branch and Bound (B&B) solutions where in one of them, the authors introduce new Dominance Rules to improve the computational analysis. Both solutions take into consideration a new Lower Bound that is part of their work and the tests were performed taking into consideration jobs with various due dates ranges and tightness. The conclusions indicate that the procedure with dominance rules outperforms the simple version of the

algorithm significantly and can solve small size problems optimally and medium size problems can be solved in reasonable time depending on the due date tightness.

In (Valente & Schaller, 2012) the authors introduce several dispatching rules, that include heuristics for the problem that do not use the quadratic tardiness objective function and the adapted versions that do take the mentioned objective function into consideration. Also the list of procedures considers backward and forward types of algorithms. The results indicate that the proposed quadratic versions of the dispatching rules perform better than the linear versions, and the best results were obtained using backward procedures when solving instances with a medium or low tardiness factor.

(Gonçalves et al., 2016) introduces three meta-heuristics which are the Steady-State Genetic Algorithm, Iterated Local Search (ILS) and variable greedy algorithm as well as a Local Search procedure, the results show an increased performance when compared with the previous work made by (Valente & Schaller, 2012).

Another approach is the problems that consider the weighted squared tardiness objective function with multiple machines working in parallel. We can mention (Hoitomt et al., 1990; Thomalla, 2001).

(Hoitomt et al., 1990) presents a near-optimal method using the Lagrangian relaxation technique for jobs that have multiple operations of different size, precedence constraint and timeouts between operations; their analysis involve up to 40 machines working at the same time.

(Thomalla, 2001) presents a similar solution using the Lagrangian relaxation method and to solve jobs that have multiple operations and precedent constraints. In this scenario, they consider that the machines have different performance and also introduce the possibility that one machine becomes unavailable during the process. The final results are compared with solutions made with heuristics and priority rules presented in the literature showing competitive results.

2.2 Quadratic Earliness and Tardiness

Another type of objective function present in literature that is worth mentioning is the one that considers earliness and tardiness together as the cost function $1||\sum h_j E_j^2 + w_j T_j^2$, along with these two variables the quadratic consideration helps the resulting schedules to be balanced in regards of not having only one job to influence the whole evaluation.

On this subject we can mention the recent work made by (Valente & Alves, 2008; Valente & Moreira, 2009; Valente, 2010; Valente, Moreira, Singh, & A. F. S. Alves, 2011; Singh, Valente, & Moreira, 2012). In (Valente & Alves, 2008) the authors present new heuristics based on the combination of previous dispatching rules adapted to consider the objective function, they also introduce improvement procedures that are applied to the final results generated by the heuristics. Comparisons of the adapted heuristics and the linear ones are presented as well as the results of the new procedures; the computational results take in consideration all types of problem sizes including very large ones.

The work by (Valente & Moreira, 2009) introduces several greedy randomized dispatching heuristics that differ from each other on multiple elements like the strategies in the construction of the greedy randomized schedules, the use of a final improving step and the use of a Local Search after each greedy randomized construction. The results are compared using the work made by (Valente & Alves, 2008; Valente, 2010).

In (Valente, 2010) the scenario also considers schedules with no idle time between jobs. The work introduces several beam search heuristics like the classic, recovering and filtered procedures. Together with the Beam Search heuristics, dispatching rules were introduced specifically EDD (Earliest Due Date), ECTL (Early-Critical-Tardy) and ETP (Early/Tardy Priority). The final results indicate that the recovering procedure of the Beam Search heuristic is the selected one for large instances of the problem and the detailed beam search is selected for small and medium size versions.

(Valente et al., 2011) takes into consideration the proposed objective function along with the constraint that no idle time is allowed between jobs. The solution that they present is based on a genetic algorithm that includes additional procedures for the initial population generation and the migration step by the definition of the individuals added to the new generation. They present different versions of the heuristic and variations of the mentioned procedures and the best results are obtained with the integrated version that include all of them in addition to a local search step. The results of the heuristics are compared with the work on (Valente & Moreira, 2009; Valente, 2010), showing better results for small and medium size problems.

In (Singh et al., 2012) the authors present a hybrid heuristic where they use a steady-state genetic algorithm together with three improvement procedures that are integrated into the heuristic. The first two procedures are used to improve the solution after the genetic

operators and the final one is used to improve the optimal solution at the end of the heuristic. The results indicate that, compared to the work previously made by (Valente, 2010), the hybrid heuristic has better results with less processing time.

2.3 Quadratic Lateness

Lateness is another measure of performance that takes in consideration the due date of the jobs, it is stated as $L_j = C_j - d_j$, the main difference with tardiness is that it can have negative values if a job is early. The use of this measure together with the quadratic factor helps to penalize jobs that differ significantly from the scheduling plan. The presented papers tackle the problem using techniques that can be adapted to our version, making them worth to mention.

In the work by (Gupta & Sen, 1983) a Branch and Bound and a heuristic procedure are introduced based on a lower bound present in literature, their results are tested using parameters that determine the difficulty of the problems. The conclusions indicate that the branch and bound performs well with small and medium size problems finding optimal solutions, while the heuristic finds good solutions reducing the computational effort for large problems.

The work by (Sen, Dileepan, & Lind, 1996) presents a Branch and Bound solution for the problem with jobs that do not have a release date, but taking into consideration the addition of initial idle time on the schedule in order to improve its quality, the analysis was made using small size instances up to 10 jobs. This last work was corrected and extended in (H. M. Soroush, 2009; H.M. Soroush, 2010), where the author presents conditions where the analysis of the previous work does not apply correctly.

In (Su & Chang, 1998) the authors introduce a new heuristic based on a decision index obtained by a rearrangement of the objective function. Their analysis is made with middle sized job sets and scheduling them with and without idle time between jobs, this work is compared with the work by (Gupta & Sen, 1983).

The work by (Schaller, 2002) presents a solution with a new procedure that introduces an idle time into the schedule, which is incorporated in a B&B algorithm to solve small and middle size instances in a reasonable amount of time and with better quality when compared with previous work. Also a simple descent heuristic procedure was proposed to solve large problems.

In (H.M. Soroush, 2010) the goals include the optimal amount of idle time to be inserted before the first job and a lower bound in order to reduce the size of the search spectrum in their branch and bound solution. Their work also allows them to tackle large versions of the problem that were solve quickly.

2.4 Linear weighted tardiness with release dates

When considering the linear version of the objective function $1|r_j|\sum w_j T_j$, the work made by (Selim Akturk & Ozdemir, 2000) introduces dominance rules for a branch and bound algorithm that is capable of solving optimally small size problems. Their results are compared with commercial scheduling software showing better results and also with similar works that do not consider release dates.

(Akturk & Ozdemir, 2001) presents a new algorithm that gives a sufficient condition for local optimality; they test the solution on many dispatching heuristics like KZRM (Kanet and Zhou approach to ATC), X-RM (modify the ATC rule to allow heuristic algorithms to insert idle times) and GRASP showing better results on all cases.

(Chang, Chou, & Lee, 2004) introduces a new Look Ahead one step heuristic for the total weighted tardiness problem by also taking in account jobs with setup times, they compare their results with exact methods on job sets with medium and large size problems.

(Jouglet, Savourey, Carlier, & Baptiste, 2008) consider total tardiness, total completion time and introduces dominance rules to improve the result of some common greedy algorithms and also a new Tabu Search inspired method to apply the rules presented.

(van den Akker, Diepen, & Hoogeveen, 2010) uses Integer Linear Programming (ILP) and a Lagrangian relaxation method to present better results for jobs in special conditions like equal due dates or equal release dates; the study can also be applied to identical parallel machines.

(Çakar, 2011) uses a dominance rule generated with trained Neural Networks algorithms in order to improve results generated with many dispatching rules like EDD, COVERT, ATC, WSPT (Weighted Shortest Processing Time) and others. The results show a better performance in terms of computational time for medium sized problem. In (Cakar & Köker, 2015) the main idea is to introduce the integration of different meta-heuristics techniques like PSO (Particle Swarm Optimization), GA and SA together with a Neuro-Dominance rule to improve the result of an initial solution.

(Yang et al., 2016) propose an optimized solution to the transmission of video as data packets using genetic algorithms and proposing a gene representation for the information to be transferred.

(Davari, Demeulemeester, Leus, & Talla Nobibon, 2016) present a branch and bound algorithm for handling jobs with precedence constraints, they also present many dominance rules to improve the performance of the solution which is then compared to other B&B solutions. The results indicate that the proposed method solves medium sized problems optimally and has better performance when considering special cases of the problem.

Chapter 3 - Scheduling Strategies

3.1 Schedule Types

There are different schedule types present in literature, they mainly differ on the unscheduled jobs that are considered in each iteration for the calculation of the objective functions. We considered the following two types: non-delay and active. In the Non-delay Schedule, only the jobs that are ready to be processed are considered among the unscheduled ones, their priority depends mainly on the objective function which will indicate the next job to be scheduled. In this scenario the machine is not kept idle at any time when a job is available for processing.

The other type considered is the Active Schedule where there cannot be a job that is completed earlier without delaying another one. In this type, the unscheduled jobs that are analyzed can include some that might not yet be ready to be processed. With this concept, we have that Non-delay Schedules are Active Schedules, however, Active Schedules may not be Non-delay.

Another relation between these two types of schedules is that Active Schedules have been proved to be dominant for regular objective functions (Miller, Conway, & Maxwell, 1967; Baker, 1974), i.e. they are better or equal than non-active schedules. An objective function is regular when it is non-decreasing with respect to all completion times, so when the completion time increases the objective function never decreases (it increases or stays the same). The proposed objective function of this work which is weighted squared tardiness can be considered regular.

3.2 Dispatching Rules Process

To improve the concepts of the schedules types we will introduce the pseudo-code where the dispatching rules will be used. The elements of the procedure include having S as the current scheduled jobs in sequence, U as the set of unscheduled jobs. The current time is t and $r_j^{min}(U)$ denotes the minimum release date among the currently unscheduled jobs.

Let EU be the set of eligible unscheduled jobs, that is, the set of currently unscheduled jobs that can, at the current iteration, be chosen as the next scheduled job. In our case those will be the Non-delay or Active Schedules mentioned previously.

Dispatching procedure pseudo-code:

1. Set $\mathrm{t} = 0$, $\mathrm{S} = \emptyset$ and $\mathrm{U} = \{1,2, \dots, \mathrm{n}\}$.
2. While $\mathrm{U} \neq \emptyset$
   2.1. Set $\mathrm{t} = \max\{\mathrm{t}, \mathrm{r}_\mathrm{j}^{\min}(\mathrm{U})\}$.
   2.2. Calculate a priority index for all jobs $\mathrm{j} \in \mathrm{EU}$.
   2.3. Let j^* be the job with the largest value of the priority index; break ties by selecting the job with the smallest due date d_j.
   2.4. Add j^* at the end of S and remove it from U.
   2.5. Set $\mathrm{t} = \mathrm{t} + \mathrm{p}_{\mathrm{j}^*}$.
3. Return S and its corresponding objective function value.

When using the non-delay strategy, the jobs in the EU set, which are the eligible unscheduled jobs, will be the ones that are available for processing at the current time t, meaning that $j \in U$ with $r_j \leq t$. In the active schedule strategy, the set of unscheduled jobs UE will include the jobs that in the current iteration, will lead to an active schedule. Let $C_j^{min}(U)$ be the minimum possible completion time among the currently unscheduled jobs. Therefore, we have $C_j^{min}(U) = min_{j \in U}\{max\{t, r_j\} + p_j\}$. In active schedules, a job cannot be completed without delaying another job and the set EU will include the unscheduled jobs that are available for processing before this minimum possible completion time (Jouglet et al., 2008), i.e. jobs $j \in U$ with $r_j < C_j^{min}(U)$. Naturally, this may include jobs that are not available at the current time (but become available before $C_j^{min}(U)$.

Chapter 4 - Heuristics

4.1 Definition

A heuristic is a set of procedures used to find a suitable solution to a given problem in reasonable amount of time and with limited resources. In optimization problems it is common to use exact methods to find the best possible solution regardless of the time and resources, but when the problems have a considerable complexity or size it is not feasible to achieve the optimal solution, this is where the use of heuristics comes along.

The use of heuristics does not ensure that the best solution is always achieved but together with different techniques it is possible to ensure some level of quality in the solutions which makes them an excellent alternative. On some types of heuristics, the definition of the problem is a key factor that is why not every heuristic can be applied to any situation. They may depend on the problem definition, the resources available for the current situation or even the scope of search for feasible solutions.

4.2 Priority Rules

Using the two different schedule types presented previously we will introduce the dispatching rules that will handle the generation of the priority values to schedule the job instances. A list of the considered dispatching rules, along with their priority indexes is given in Table 1.

Table 1 - Dispatching rules

Heuristic	Priority Index
EDD	$1/d_j$
MDD	$\begin{cases} 1/(C_j(S)-t) & if\ s_j(S) \le 0 \\ 1/(d_j-t) & otherwise \end{cases}$
SLK	$\begin{cases} C_j(S)-d_j & if\ s_j(S) \le -1 \\ 1/(s_j(S)+2) & otherwise \end{cases}$
WSLK/P	$\begin{cases} (w_j/p_j)*(C_j(S)-d_j) & if\ s_j(S) \le -1 \\ (w_j/p_j)*[1/(s_j(S)+2)] & otherwise \end{cases}$
WSPT	w_j/p_j
WSLK_SPT	$\begin{cases} w_j/p_j & if\ s_j(S) \le p_j \\ w_j/s_j(S) & otherwise \end{cases}$
WMDD	$\begin{cases} w_j/p_j & if\ s_j(S) \le 0 \\ w_j/(d_j-t) & otherwise \end{cases}$
AR	$\begin{cases} w_j/p_j & if\ s_j(S) \le 0 \\ (w_j/p_j)*[k\bar{p}/(k\bar{p}+s_j(S))] & otherwise \end{cases}$
ATC	$\begin{cases} w_j/p_j & if\ s_j(S) \le 0 \\ (w_j/p_j)*\exp(-s_j(S)/k\bar{p}) & otherwise \end{cases}$
QWSPT	$(w_j/p_j)*(\bar{p}+2T_j(S))$
QWSLK_SPT	$\begin{cases} (w_j/p_j)*(\bar{p}+2T_j(S)) & if\ s_j(S) \le p_j \\ (w_j/s_j(S))*\bar{p} & otherwise \end{cases}$
QWMDD	$\begin{cases} (w_j/p_j)*(\bar{p}+2T_j(S)) & if\ s_j(S) \le 0 \\ (w_j/(d_j-t))*\bar{p} & otherwise \end{cases}$
QAR	$\begin{cases} (w_j/p_j)*(\bar{p}+2T_j(S)) & if\ s_j(S) \le 0 \\ (w_j/p_j)*\bar{p}*[k\bar{p}/(k\bar{p}+s_j(S))] & otherwise \end{cases}$
QATC	$\begin{cases} (w_j/p_j)*(\bar{p}+2T_j(S)) & if\ s_j(S) \le 0 \\ (w_j/p_j)*\bar{p}*\exp(-s_j(S)/k\bar{p}) & otherwise \end{cases}$

Between the rules we can mention, Earliest Due Date (EDD), Modified Due Date (MDD), Slack (SLK) and Weighted Slack per processing time (WSLK/P) which are general dispatching rules that can be applied to any problem with due dates.

The next set of rules are the linear ones, since they do not take in consideration the quadratic value of the tardiness, which are WSPT, WSLK_SPT, WMDD, AR and ATC. Finally the quadratic version of the linear rules, that are present in literature, will be applied using the proposed objective function of this work which is $1|r_j|\sum w_j T_j^2$, therefore we have the rules QWSPT, QWSLK_SPT, QWMDD, QAR and QATC.

Before moving to the details of each dispatching technique, the following concepts will be introduced: the completion time of an unscheduled job j that is available to be sequenced after the current partial sequence S, will be denoted by $C_j(S)$, with $C_j(S) = \max\{t, r_j\} + p_j$. The slack of an eligible unscheduled job j, given a current partial sequence S, is given by $s_j(S) = d_j - C_j(S)$. The tardiness of an eligible unscheduled job j, when j is sequenced after the current partial sequence S, will be denoted by $T_j(S)$, with $T_j(S) = \max\{C_j(S) - d_j; 0\}$. Let $\bar{p}$ denote the average processing time among all eligible unscheduled jobs. Finally, k is a user-defined lookahead parameter required in some of the heuristics. We will now describe the rules and their basic functionality in order to provide details on their functionality.

4.2.1 Earliest Due Date (EDD)

Based on the work made by (Jackson, 1955), this rule uses the due date of the job for the priority index. The schedule of jobs is generated by sorting the instances by due date in an increasing order, this means the jobs with late due dates will be at the end of the list.

$$index = 1/d_j$$

4.2.2 Modified Due Date (MDD)

In the modified due date (MDD) rule (Baker & Bertrand, 1982; Vepsalainen & Morton, 1987), at each iteration we select the job with the minimum value of the modified due date, where in each iteration it is calculated as $max\{d_j, C_j(S)\} = max\{d_j - t, C_j(S) - t\}$. This rule uses the completion time for the priority, when the job finishes after the due date, that is when we have a positive slack; otherwise, the due date is used.

$$\begin{cases} 1/(C_j(S) - t) & \text{if } s_j(S) \leq 0 \\ 1/(d_j - t) & \text{otherwise} \end{cases}$$

4.2.3 Slack (SLK)

The original minimum slack rule (Panwalkar & Iskander, 1977; Vepsalainen & Morton, 1987) chooses, at each iteration, the job with the minimum slack $s_j(S) = d_j - C_j(S)$. For the application of this rule and to maintain the relation that jobs with higher priority will be

processed before, we will use the following index: When the slack is negative, $C_j(S) - d_j$ is an adequate priority value, the priority should be higher when this value is higher. On the other hand, when the slack is positive, $1/\left(d_j - C_j(S)\right)$ is an adequate priority value: jobs with higher slacks will indeed have a lower priority.

The normal expression to separate the two rules is the condition $s_j(S) \leq 0$. However, in this case a value of 0 will cause a division by 0 in the second expression which does not work here (both rules need to have the same priority value for the split condition value).

The proposed value for the split condition is -1, along with a modification in the priority index calculation.

$$\begin{cases} C_j(S) - d_j & if\ s_j(S) \leq -1 \\ 1/\left(s_j(S) + 2\right) & otherwise \end{cases}$$

This way both branches have the same priority value for the split condition and we maintain the relation that if the slack of a job is negative, that is the job is tardy already, it will have a higher priority than a job that have a slight negative value or positive, that is more than -1.

4.2.4 Weighted Slack (WSLK/P)

The weighted slack per processing time has the same concept as the Slack rule in (Panwalkar & Iskander, 1977; Vepsalainen & Morton, 1987), with the addition of the relation between the weight and the processing given by w_j/p_j of the job j that is being considered. The split value problem is also the same so the solution for the previous rule is applied, which is to prevent from having negative values on the priorities, this way the priority index is given by:

$$\begin{cases} (w_j/p_j) * \left(C_j(S) - d_j\right) & if\ s_j(S) \leq -1 \\ (w_j/p_j) * \left[1/\left(s_j(S) + 2\right)\right] & otherwise \end{cases}$$

4.2.5 Weighted Shortest Processing Time (WSPT)

The rule presented in (E., 1956), takes the relation between the weight of the job j and its processing time as the priority index value, this way the resulting index is always positive and jobs are scheduled by putting first the ones where the importance is close to the processing

time, so longer jobs will be pushed to the end unless they are quite important. The priority index has the expression:

$$w_j/p_j$$

4.2.6 Weighted Slack and Shortest Processing Time (WSLK_SPT)

This rule proposed by (Osman, Belouadah, Fleszar, & Saffar, 2009),takes the base of the WSPT priority index using the relation between the weight of the job w_j and the processing time p_j, together with the SLACK rule using the concept $s_j(S) = d_j - C_j(S)$, the rule maintains the idea that jobs with higher values of priorities will be scheduled first. The split value condition for the rule is modified to compare the slack value with the processing time $s_j(S) \leq p_j$, this means that jobs will be prioritized using the processing time as long as the slack is maintained with a positive value, otherwise the value of the slack will be use as the factor in the calculation.

$$\begin{cases} w_j/p_j & if\ s_j(S) \leq p_j \\ w_j/s_j(S) & otherwise \end{cases}$$

4.2.7 Weighted Modified Due Date (WMDD)

The WMDD rule (Kanet & Li, 2004), maintains the expression used for the split value considering the slack of the schedule in time t and the priority index for the jobs with a negative slack, which is when the the schedule is not tardy, the rule uses the relation between the weight of the job w_j and the processing time p_j. For the cases when the slack is positive, the same expression as the original MDD index is used and only the weight is added w_j.

$$\begin{cases} w_j/p_j & if\ s_j(S) \leq 0 \\ w_j/(d_j - t) & otherwise \end{cases}$$

4.2.8 Alidaee and Ramakrishnan (AR)

The rule proposed by (Alidaee & Ramakrishnan, 1996) uses the concepts mentioned in the previous chapter, the look ahead parameter k and the average processing time $\bar{p}$ of the unscheduled jobs, in order to calculate the priority value when the slack is positive $s_j(S) > 0$. The benefit of the look ahead parameter, is to take in consideration the jobs that are close to

become tardy in the following iterations, also called critical jobs. Therefore, the rule improves the quality of the analysis by using the look ahead parameter which in this work has a dynamic value that depend on the number of critical jobs. The parameter k is then applied to the relation of the weight w_j and the processing time p_j in order to maintain the importance of the jobs as a factor in the calculation of the priority index. In the case that the slack is negative, the rule only considers the weight and processing time.

$$\begin{cases} w_j/p_j & if\ s_j(S) \leq 0 \\ (w_j/p_j) * \left[k\bar{p}/(k\bar{p} + s_j(S))\right] & otherwise \end{cases}$$

4.2.9 Apparent Tardiness Cost (ATC)

In this rule proposed by (Vepsalainen & Morton, 1987), the advantage of having the dynamic look ahead parameter is also present, when the slack value is positive the priority index calculation uses exponential function together with the value of the slack $s_j(S)$ and the multiplication of the k parameter and the average processing time $\bar{p}$ of the unscheduled jobs. The rule focuses in maintaining the importance of critical jobs that help to keep the value of the slack in a reasonable value.

$$\begin{cases} w_j/p_j & if\ s_j(S) \leq 0 \\ \left(w_j/p_j\right) * \exp\left(-s_j(S)/k\bar{p}\right) & otherwise \end{cases}$$

4.2.10 Quadratic Weighted Shortest Processing Time (QWSPT)

This rule is an adaptation of the WSPT to consider the quadratic objective function when generating the priority index value. Proposed by (Valente & Alves, 2008), the rule takes in consideration the basic concept of the relation between the weight w_j and the processing time p_j of the job. The second part of the priority index uses the average processing time of the eligible unscheduled jobs $\bar{p}$ and the expression $C_j(S) - d_j$, which can be described as the tardiness of the job in the current schedule, to allow the rule to consider the delay of the jobs as a critical component to prioritize.

$$\left(w_j/p_j\right) * \left(\bar{p} + 2T_j(S)\right)$$

4.2.11 Quadratic AR, ATC, WMDD and WSLK_SPT

The quadratic version of these rules, proposed by (Valente & Schaller, 2012), is an adaptation of the original rules to consider the quadratic objective function. In the mentioned work the difference with the linear version, in both rules, is that the expression of the priority index is modified when the job is tardy. The relation between the weight w_j and the processing time p_j of the job, which is the WSPT rule, is replaced by the QWSPT index. Additionally, a correction is made to the expression that is used when the slack is positive, the average processing time is added to maintain the equality of the rule in the split value condition which is when the slack is zero, except for the case of QWSLK_SPT where the split value is equal to the processing time p_j of the job.

QAR
$$\begin{cases} (w_j/p_j) * \left(\bar{p} + 2T_j(S)\right) & if\ s_j(S) \leq 0 \\ (w_j/p_j) * \bar{p} * \left[k\bar{p}/(k\bar{p} + s_j(S))\right] & otherwise \end{cases}$$

QATC
$$\begin{cases} (w_j/p_j) * \left(\bar{p} + 2T_j(S)\right) & if\ s_j(S) \leq 0 \\ (w_j/p_j) * \bar{p} * \exp\left(-s_j(S)/k\bar{p}\right) & otherwise \end{cases}$$

QWMDD
$$\begin{cases} (w_j/p_j) * \left(\bar{p} + 2T_j(S)\right) & if\ s_j(S) \leq 0 \\ \left(w_j/(d_j - t)\right) * \bar{p} & otherwise \end{cases}$$

QWSLK_SPT
$$\begin{cases} (w_j/p_j) * \left(\bar{p} + 2T_j(S)\right) & if\ s_j(S) \leq p_j \\ \left(w_j/s_j(S)\right) * \bar{p} & otherwise \end{cases}$$

Chapter 5 - Experimental Design

In the following chapter we will present the different aspects regarding the computational analysis used to study the particular problem presented. Starting with the description of the test scenarios, then moving to the different parameters for the instance generation, the tune up of the dispatching rules to consider the objective function, the technology to code and run the algorithms and the performance measures that will be used to compare how well or bad the different dispatching rules resulted in.

5.1 Instance Sets

In order to test the proposed heuristics, we generate the instances based on previous work present in literature to have a variety set of instances with different conditions and workloads, as a way to validate the scenarios in which the proposed solution is studied.

Since the generation of instances require the setup of different variables like numbers of jobs per set, processing time, weight, release date and due date; we will describe how we generated them in accordance with previous work made by (Chu, 1992; Selim Akturk & Ozdemir, 2000; Akturk & Ozdemir, 2001; Baptiste, Carlier, & Jouglet, 2004; Chang et al., 2004; Jouglet et al., 2008; Su & Chen, 2008).

The number of jobs per set that will be considered are $\mathrm{n} = 10, 15, 20, 25, 50, 100, 250, 500, 1000, 1500 \text{ and } 2000$. These values assure that the experiment will be done starting with small size problem to reasonable large size ones.

The processing p_j of the jobs uses a uniform distribution of 1 to 100 to be generated. The weight w_j property uses a uniform distribution of 1 to 10.

For the release date, r_j uses a uniform distribution of $[0, \alpha \sum_{j=1}^{n} p_j]$. The α parameter is based on the work of (Chu, 1992) and it uses 4 different values to control the values of the release date $\alpha = 0.25, 0.5, 1.0, 1.5$.

The calculation of due date does not have a single defined calculation, instead we use the earliest completion time which is defined as $(r_j + p_j)$, together with an integer slack time with a uniform distribution $[0, \beta \sum_{j=1}^{n} p_j]$. So the value of the due dates is generated with the expression $d_j = (r_j + p_j) + slack_{time}$. The β parameter is also based on the work of (Chu,

1992) and it uses 3 different values to control the values of the release date $\beta = 0.05, 0.25, 0.5$.

For our study we will use two types of jobs sets that are generated separately. The first type of job set, used for heuristic comparisons, uses 100 instances per parameter combination, that is, for each combination of number of jobs n, release date parameter α and due date parameter β.

The second type, which has no relation with the first type of job set, is used for parameter adjustment and uses 10 instances per parameter combination. For each instance size n, we then have 12 combinations of release date range and due date range, thus providing a wide variety of conditions.

5.2 Technology Environment

The coding of the heuristic algorithms and the test scenarios are developed using the C++ programming language using the Microsoft Visual Studio development tool. The execution of the tests and the analysis of the results were made using a Windows 10, 64–bit operating system, on an Intel Core i7 4770, with 3.4G processor and 16GB RAM.

5.3 Look Ahead Parameter k

The following dispatching rules AR, ATC, QAR and QATC, make use of a parameter referred to as k, that as mentioned in (Vepsalainen & Morton, 1987), represents a look ahead parameter related to the number of unscheduled jobs that will become tardy in the near future, also called critical jobs. The priority index in the mentioned dispatching rules involve the multiplication of the average processing time of the eligible unscheduled jobs with the parameter k, this will help the rules to consider the critical jobs.

Since the number of critical jobs will change in time, we will not use a particular value for k as it is suggested in previous works in literature. Instead we modify the value of the parameter in each step of the algorithm, so it will increase with the proportion of the number of jobs that will become tardy.

In each step we use the component s_{prop} which is a user-defined slack proportion value parameter with values $0 < s_{prop} < 1$, and the component P_{elig_unsch} as the sum of the

processing times of the eligible unscheduled jobs at time t. With these values we calculate the critical slack value s_{crit} as $s_{crit} = s_{prop} * P_{elig\_unsch}$, this value will be the condition for the expression $0 < s_j \leq s_{crit}$ indicating if a particular job j is about to become tardy or critical, where the slack of the job is greater than 0 but less than s_{crit}; finally k is set to the number of critical jobs, or 0.5 if no job is critical. The values {0.05, 0.10, 0.15, 0.20, … , 0.85, 0.90, 0.95} were considered for s_{prop}.

5.4 Parameter Adjustment Test

Using the tests instances for the dispatching rules AR, ATC, QAT and QATC we worked on the adjustment of the parameter k which is the look ahead value that depends on the critical jobs at each iteration. After the tests, the conclusion was that using a single value for the s_{prop} parameter would no suffice all conditions when considering the different variables that define a set of job instances, which are n, α and β as described previously.

Therefore, the following rules will be applied for the value of s_{prop} when considering non-delay schedule types, AR and ATC rules will use the value of 0.05 on all instances. QATC rule will use 0.2 when the size of instances is 50 or less, or having α = 1.0, 1.5 or for instances where β = 0.5. For the remaining configurations we will proceed with a 0.85 value for s_{prop}.

In the case of QAR, the value of 0.05 will be used in the same scenarios as QATC, for instances of size 50 or less, or when having α = 1.0, 1.5 or β = 0.5 with any size instances. The same case of using the value 0.85 for s_{prop} for the other configurations.

For the active schedule versions, the value of 0.05 was selected to be used in all instances for AR and ATC on the s_{prop} parameter. On the QATC rule, the value of 0.05 when the size of instances is 100 or less, or having α = 1.0, 1.5 or for instances where β = 0.5. The rest of the configurations will use a value of 0.4. For the rule QAR, the value of 0.05 will be used for instances of size 500 or less, or when having α = 1.0, 1.5 or β = 0.5 with any size instances. For the remaining instances, 0.5 was selected for s_{prop}.

5.5 Performance Measure

The performance measure that will be used to analyze the relation between the dispatching rules is the Relative Deviation Index (RDI) against the worst result. This method was proposed by (Zemel, 1981) and is a common measure used in problems that include an objective that is a function of tardiness, as an example we have the work from (Kim, 1993; Kim, Lim, & Park, 1996; Vallada, Ruiz, & Minella, 2008).

$$\frac{ofv_{rule} - ofv_{best}}{ofv_{best} - ofv_{worst}}$$

In the expression, ofv_{best} and ofv_{worst} represent the best and worst objective functions values of the analyzed results, that is the list of rules that are being compared against each other, and ofv_{rule} denotes the objective function of the rule for which the index is being generated.

There is a note to be made regarding the implementation of this method. There will be problems where the value of the objective function weighted squared tardiness will be 0 for the best and worst, thus the presented formula will have a division by zero error because of the denominator. For these cases and also when the best and the worst are the same we will be controlling that scenario and assign a 0 value to the RDI measure. The performance measure will return values between 0 and 1, where values closer to 0 indicate a better performance than the ones close to 1.

Chapter 6 - Computational Results

The computational results of the different dispatching rules heuristics will be presented considering the two types of schedules involved in the work, that is non-delay and active schedules, in order to present the results in the proper context of the application of the solutions.

6.1 Non-delay Schedule Heuristics

For the non-delay schedules we will start by comparing the linear dispatching rules with their quadratic versions, the results will indicate the behavior of adding the quadratic tardiness factor in the objective function. The second analysis involves the general rules comparing the quadratic dispatching rules, here the performance will indicate the different of using a general approach to a scheduling problem versus having a more specific solution which considers multiple properties in the scenario.

6.1.1 Linear versus Quadratic Dispatching Rules

For the non-delay schedules we will start by comparing the linear dispatching rules with their quadratic versions using the average value of the performance measure grouped by the jobs instances size n. Table 2, compares the dispatching rules in pairs, where smaller values indicate a better performance, although there is no relation of comparison between different types of dispatching rules.

Table 2 – Non-Delay Linear versus Quadratic

	Average RDI									
N	AR	QAR	ATC	QATC	WMDD	QWMDD	WSPT	QWSPT	WSLK_SPT	QWSLK_SPT
10	0,508	0,061	0,451	0,101	0,533	0,076	0,584	0,034	0,587	0,043
15	0,620	0,064	0,558	0,118	0,662	0,064	0,730	0,025	0,702	0,037
20	0,679	0,055	0,613	0,098	0,701	0,066	0,790	0,021	0,753	0,032
25	0,698	0,046	0,648	0,077	0,723	0,053	0,822	0,013	0,758	0,030
50	0,736	0,023	0,682	0,050	0,742	0,043	0,897	0,004	0,777	0,016
100	0,713	0,008	0,658	0,032	0,717	0,029	0,933	0,000	0,731	0,015
250	0,668	0,002	0,615	0,021	0,677	0,006	0,947	0,000	0,678	0,003
500	0,616	0,002	0,588	0,017	0,633	0,002	0,946	0,000	0,639	0,001
1000	0,598	0,002	0,578	0,014	0,600	0,002	0,955	0,000	0,604	0,001
1500	0,586	0,002	0,558	0,020	0,588	0,000	0,958	0,000	0,593	0,000
2000	0,579	0,001	0,549	0,018	0,578	0,003	0,958	0,000	0,582	0,002
General	**0,636**	**0,024**	**0,591**	**0,051**	**0,650**	**0,031**	**0,865**	**0,009**	**0,673**	**0,016**

The results in the table indicate the clear difference in performance between the linear and the quadratic versions of the dispatching rules, moreover the final values are at least 10 times smaller in the case of the quadratic rules.

A detail that can be mentioned is that the average values of the performance measure for the quadratic rules decreases as the size of n increases, meaning that the rules had better results as the problems were getting larger. This scenario gives a clear view that considering the quadratic tardiness objective function indeed provides better results.

The rule with the best performance comparing the linear version was QWSPT where the average difference of 0.009 against 0.865 shows that the linear version was in the majority of the instances, the one with the worst result. The rule with the second largest improvement versus the linear version was QWSLK_SPT with a 0.016 general average, in this case the difference was not as big as the best rule but still noticeable. In both cases they greatly outperformed not only in the average but also in the total number of cases where the linear versions had a worst result.

6.1.2 General versus Quadratic Dispatching Rules

Table 3, shows the comparison between the general purpose rules, namely EDD, MDD, SLK, WSLK/P, against the quadratic rules. In this case the comparison will be all rules against each

other using the average of the performance measure, this way the rule with the lowest value will be the best among the entire group, so we will also be comparing quadratic rules among themselves.

Table 3 – Non-Delay General versus Quadratic

Average RDI									
N	EDD	MDD	SLK	WSLK/P	QAR	QATC	QWMDD	QWSLK_SPT	QWSPT
10	0,286	0,316	0,645	0,128	0,062	0,037	0,081	0,106	0,177
15	0,354	0,370	0,686	0,119	0,058	0,035	0,072	0,092	0,244
20	0,394	0,423	0,674	0,105	0,057	0,036	0,067	0,087	0,262
25	0,425	0,458	0,662	0,091	0,045	0,029	0,059	0,073	0,286
50	0,477	0,509	0,625	0,070	0,034	0,023	0,038	0,045	0,322
100	0,499	0,521	0,588	0,050	0,029	0,022	0,032	0,038	0,373
250	0,489	0,510	0,519	0,024	0,017	0,018	0,020	0,024	0,405
500	0,481	0,501	0,496	0,026	0,022	0,016	0,027	0,027	0,413
1000	0,479	0,491	0,487	0,020	0,017	0,015	0,016	0,016	0,441
1500	0,473	0,474	0,477	0,022	0,019	0,016	0,020	0,020	0,447
2000	0,473	0,467	0,475	0,022	0,020	0,018	0,025	0,025	0,452
Global	**0,439**	**0,458**	**0,576**	**0,062**	**0,035**	**0,024**	**0,042**	**0,050**	**0,347**

This table shows how most of the general rules are outperformed by the quadratic ones, although there are two noticeable scenarios that must be mentioned. The first one is the result obtained by the WSLK/P rule that came in fifth place with a very close value comparing with the quadratic rules. The second scenario is the QWSPT rule where the performance measure is closer to the worst results, this can be explained by the fact that this rule is appropriate when the jobs are already tardy, , which in several times will not be the case, unless due dates are very tight. However, when comparing with its linear version, it had the best increase and average result, as shown in the previous analysis.

Regarding the best rules in the table, the quadratic ones show again an increase in performance as the size of the instances get larger, this is a sustained behavior that shows the quality of results when compared to more simple rules.

The QATC rule is the one with the best overall performance followed closely by the QAR rule. The main differences were obtained in the small size instances where QATC did almost two times better, but this behavior got reduced as the instances got larger.

6.2 Active Schedule Heuristics

On the results for the dispatching rules considering active schedules we will also compare the linear against the quadratic versions of the rules as a first step, where we will have the variation in performance for the quadratic tardiness consideration. The analysis of the quadratic versus general rules will be presented as well, having the idea of showing the difference of considering the second type of schedules in the results.

6.2.1 Linear versus Quadratic Dispatching Rules

The comparison of the linear versus the quadratic versions for the active schedules will follow the same approach as the non-delay. We will use the average value of the performance measure and the table will indicate the results in pairs. Again there is no relation in the values between different types of rules.

Table 4 – Active Linear versus Quadratic

	Average RDI									
N	AR	QAR	ATC	QATC	WMDD	QWMDD	WSPT	QWSPT	WSLK_SPT	QWSLK_SPT
10	0,741	0,043	0,632	0,074	0,731	0,063	0,798	0,028	0,849	0,022
15	0,824	0,037	0,708	0,067	0,819	0,053	0,878	0,022	0,932	0,019
20	0,852	0,032	0,73	0,057	0,843	0,041	0,898	0,012	0,969	0,008
25	0,863	0,021	0,743	0,032	0,859	0,028	0,879	0,013	0,973	0,008
50	0,832	0,007	0,733	0,014	0,810	0,034	0,832	0,013	0,987	0,003
100	0,788	0,003	0,692	0,009	0,761	0,023	0,765	0,011	0,998	0
250	0,723	0,001	0,629	0,01	0,691	0,005	0,688	0,003	0,998	0
500	0,668	0	0,59	0	0,654	0	0,653	0	0,999	0
1000	0,605	0	0,584	0,0008	0,608	0,001	0,607	0	0,999	0
1500	0,590	0	0,578	0,0008	0,590	0,001	0,59	0	1	0
2000	0,587	0	0,565	0,0008	0,583	0	0,581	0	0,998	0
Global	**0,734**	**0,013**	**0,653**	**0,024**	**0,723**	**0,023**	**0,742**	**0,009**	**0,973**	**0,005**

Table 4, shows again that the quadratic ones performed quite better. In the case of active schedules, the average values of the performance measure for these rules are even smaller indicating that the difference in performance is larger. The same situation as the non-delay schedule occurs, that is, the value of the average RDI decreases as the size of the instances gets larger.

The dispatching rule with the best result was QWSLK_SPT with a 0.009 general average against the linear version with a 0.973, in the case of the active schedules, the quadratic had a much better performance than the non-delay scenario. The second best rule was QWSPT with similar results and the comparison is still with a large gap between the versions.

When comparing the linear and the quadratic rules, in both types of schedules we have that the best results were from the quadratic and also that those dispatching rules were QWSLK-SPT and QWSPT. This indicates that the use of the slack clearly increases the overall results in the heuristics.

6.2.2 General versus Quadratic Dispatching Rules

The same analysis is done in this case using the results of the active schedule, the general purpose rules EDD, MDD, SLK, WSLK/P, are compared altogether with the quadratic rules. Using this approach, the rule with the lowest value will have the best performance of the entire group, meaning that the table indicates the relation between all the heuristics.

Table 5 – Active General versus Quadratic

Average RDI									
N	EDD	MDD	SLK	WSLK/P	QAR	QATC	QWMDD	QWSLK_SPT	QWSPT
10	0,281	0,372	0,606	0,260	0,330	0,245	0,148	0,309	0,549
15	0,327	0,402	0,614	0,213	0,290	0,181	0,105	0,225	0,560
20	0,361	0,439	0,589	0,161	0,232	0,133	0,069	0,167	0,556
25	0,391	0,452	0,595	0,140	0,200	0,106	0,060	0,143	0,552
50	0,442	0,481	0,558	0,111	0,138	0,054	0,031	0,070	0,539
100	0,470	0,496	0,535	0,084	0,096	0,026	0,021	0,035	0,515
250	0,470	0,492	0,492	0,046	0,051	0,010	0,011	0,013	0,495
500	0,465	0,483	0,474	0,038	0,040	0,005	0,008	0,008	0,489
1000	0,461	0,466	0,466	0,039	0,039	0,003	0,006	0,006	0,495
1500	0,458	0,454	0,461	0,042	0,041	0,003	0,006	0,005	0,499
2000	0,455	0,444	0,457	0,042	0,042	0,003	0,005	0,005	0,503
Global	**0,416**	**0,453**	**0,531**	**0,107**	**0,136**	**0,070**	**0,043**	**0,090**	**0,523**

The results in Table 5, come from the comparison using the RDI measure indicate once again that the quadratic dispatching rules have better results over the general ones, although the QWSPT was the exception having almost the worst performance just like the scenario with the non-delay schedules.

The best overall performance was obtained by QWMDD rule followed by QATC, which was the best in the non-delay results. For the larger instances QATC had the best values in this case followed very close by QWMDD and QWSLK_SPT, because of this, we will select them for the next analysis when comparing active against non-delay dispatching rules results.

6.3 Active and Non-delay Schedule Analysis

In this section we will provide an analysis of the best dispatching rules from the non-delay schedules, namely QATC, QAT and QMDD, and the best rules from the active schedule section, which were QWMDD, QATC and QWSLK_SPT. The first analysis will be using the performance measure RDI, which means that the value of the objective function, weighted quadratic tardiness, will indicate the ranking of the dispatching rules.

The second comparison will use a new measure, the makespan of each job set instance, which is the difference between the completion time of the last job and the earliest possible start time, in the case of our work it is the smallest release date. For the non-delay schedules, since there is no unforced idle time added because we only consider jobs that are already available, the dispatching rules will have the same makespan. In the case of the active schedule rules, different rules may insert different amount of unforced idle time, therefore, they may have different makespans.

In order to calculate the performance measure of the active dispatching rules, we will use the relative increase over the non-delay makespan, which will be calculated as:

$$\frac{makespan_{active} - makespan_{non-delay}}{makespan_{non-delay} * 100}$$

In the expression, $makespan_{active}$ corresponds to the total time spent by the active schedule rule and $makespan_{non-delay}$, the makespan of the non-delay schedule rules. This way the results will show the amount of time that is introduced when considering jobs that are not immediately available at the current time.

Table 6 – Non-Delay versus Active Average RDI

	Average RDI					
N	NON_QATC	NON_QAR	NON_QWMDD	ACT_QWMDD	ACT_QATC	ACT_QWSLK_SPT
10	0,138	0,151	0,167	0,491	0,661	0,749
15	0,123	0,128	0,134	0,500	0,671	0,739
20	0,122	0,132	0,140	0,529	0,656	0,738
25	0,136	0,146	0,154	0,525	0,634	0,721
50	0,152	0,181	0,177	0,551	0,602	0,635
100	0,185	0,215	0,241	0,593	0,592	0,609
250	0,248	0,249	0,310	0,573	0,550	0,562
500	0,250	0,287	0,428	0,595	0,499	0,590
1000	0,274	0,305	0,501	0,603	0,443	0,595
1500	0,256	0,327	0,563	0,602	0,423	0,591
2000	0,269	0,338	0,605	0,603	0,395	0,606
Global	**0,196**	**0,224**	**0,311**	**0,560**	**0,557**	**0,649**

Table 6 will be the starting point of the analysis, the columns where the name start with the prefix "NON_", correspond to the dispatching rules of the non-delay schedules and the ones with "ACT_" to the active schedules rules.

The results clearly indicate that the best performance were obtained by the non-delay rules where the global average RDI of the rules are almost half of the values from the active schedules values. These results can be explained by the use of the quadratic tardiness objective function, since when using active schedules, the introduction of idle time in order to provide better results when considering critical jobs in the horizon is a common behavior. Therefore, when having a non-delay case with particular job that has a tardiness of 20, the objective function would be 400, but then the same job in the active schedule version for some reason ends up with a tardiness of 21, the objective function goes to 441, the difference in terms of performance may seem very different when in reality it only was modified in one unit. When considering a quadratic objective function, the use of active schedules can come with great cost, even if the idea behind is to prioritize critical jobs and make better decisions.

Regarding the values for the RDI on the non-delay rules, we can mention that the average slowly grows as the size of the instances gets larger, but in the case of the active rules, the values maintain a certain range for all instances, this could become a factor if the size of the considered instances gets a significant increase for a particular problem.

The best performance was obtained by the NON_QATC rule followed by NON_QAR, indicating that the QWSPT part of the rules impacts considerably when we have job set instances that do not have delay. In the case of the ACT_QATC a point that can be mentioned is that the performance in the larger instances is better than the NON_QWMDD and very close to NON_QAR, which confirms even more that the best overall rule is the QATC.

For the NON_QWMDD we can mention that when compared to the active counterpart, the values of the RDI for the larger instances are almost the same, however, the main difference is when dealing with small size problems.

Table 7 - Non-Delay versus Active Average Makespan

Average Makespan			
N	ACT_QWMDD	ACT_QATC	ACT_QWSLK_SPT
10	4,393	6,332	6,268
15	3,389	4,804	4,615
20	2,832	3,734	3,679
25	2,342	3,096	3,012
50	1,467	1,776	1,633
100	1,014	1,038	1,021
250	0,512	0,528	0,507
500	0,301	0,314	0,298
1000	0,189	0,181	0,187
1500	0,135	0,129	0,132
2000	0,105	0,104	0,103
Global	**1,516**	**2,003**	**1,951**

The analysis using the makespan comparison is presented in Table 7, here we have the increase introduced by the active schedule rules compared to the non-delay rules, where they all have the same makespan value. In all cases the overall average increase barely gets to 2%, the most effect is noticed on the small size instances where 6,3% is the maximum value, on the large size instances the average does not even get to 1%. This is the same behavior presented in the RDI comparison scenario, for large values of N the performance of the rules in both types of schedules is similar.

The rule with the best relation in terms of makespan was again QWMDD which outperformed the other active schedule rules in almost all scenarios, although for large instances that percentage of difference can barely be noticed.

In the following parts of the section we will introduce the effect of the parameters alpha α and beta β in both performance measure as a way to analyse if there is a particular trend with specific values of these parameters.

Table 8 - Non-Delay versus Active Average RDI by Alpha

		Average RDI					
N	alpha	NON_QATC	NON_QAR	NON_QWMDD	ACT_QWMDD	ACT_QATC	ACT_QWSLK_SPT
20	0,25	0,137	0,117	0,120	0,559	0,810	0,657
	0,5	0,071	0,083	0,093	0,592	0,765	0,763
	1	0,093	0,125	0,138	0,517	0,586	0,820
	1,5	0,186	0,204	0,210	0,448	0,465	0,712
25	0,25	0,172	0,142	0,138	0,533	0,767	0,634
	0,5	0,085	0,090	0,084	0,591	0,791	0,760
	1	0,091	0,126	0,148	0,549	0,579	0,789
	1,5	0,195	0,227	0,246	0,427	0,398	0,701
1500	0,25	0,077	0,095	0,807	0,952	0,542	0,940
	0,5	0,067	0,240	0,513	0,890	0,570	0,867
	1	0,272	0,310	0,267	0,458	0,456	0,456
	1,5	0,609	0,665	0,665	0,109	0,123	0,101
2000	0,25	0,081	0,110	0,857	0,967	0,493	0,961
	0,5	0,066	0,248	0,564	0,890	0,581	0,881
	1	0,287	0,352	0,347	0,454	0,400	0,468
	1,5	0,641	0,642	0,651	0,100	0,107	0,115

Table 8, presents the results of the average RDI using only two types of job size instances, small ones (20, 25) and large (1500, 2000). The new added column is the one that corresponds to the alpha α parameter, which handles the spread of the release dates of the jobs, the idea is to show the effect of the parameter in the performance of the rules. Here can be noticed that the value of 1,5 in the alpha variable has a very distinctive effect in the performance of the non-delay dispatching rules, the rules performed almost as bad as the active rules for small job instances and significantly worse for large instances. This can be explained because of the consideration of non-available jobs in the generation of the schedule that the active rules provide, using the look ahead parameter, it helps them have better results than the non-delay rules. Besides that, another point can be made, that as the size of alpha gets larger, the release dates are more spread out so the number of jobs that will be tardy tends to decrease, therefore,

the extra time introduced by active rules will be reduced and the results will be closer to the non-delay versions.

Table 9 - Non-Delay versus Active Average RDI by beta

		Average RDI					
N	beta	NON_QATC	NON_QAR	NON_QWMDD	ACT_QWMDD	ACT_QATC	ACT_QWSLK_SPT
20	0,05	0,058	0,059	0,068	0,390	0,821	0,829
	0,25	0,121	0,120	0,134	0,569	0,674	0,725
	0,5	0,187	0,217	0,219	0,628	0,474	0,661
25	0,05	0,047	0,049	0,056	0,402	0,790	0,842
	0,25	0,132	0,127	0,140	0,558	0,660	0,695
	0,5	0,228	0,263	0,266	0,615	0,452	0,625
1500	0,05	0,240	0,224	0,472	0,685	0,548	0,653
	0,25	0,319	0,282	0,694	0,606	0,342	0,611
	0,5	0,210	0,476	0,523	0,516	0,378	0,509
2000	0,05	0,276	0,252	0,540	0,662	0,519	0,649
	0,25	0,315	0,287	0,725	0,619	0,296	0,646
	0,5	0,214	0,476	0,550	0,527	0,371	0,524

In Table 9, again the two types of job instance size are presented using the RDI as the values for the performance measure, but in this case the results indicate the effect of the beta β parameter, which handles the relation between the release dates and the due dates of the jobs, that is, larger values of beta make the relation to be more spread out. The observation that can be made with these results is that in the case of the QWMDD rule, when we have larger job instances there is no relation with the values that the parameter beta has and the performance is very similar on both versions, non-delay and active. Besides this, there is no clear and consistent effect of the value of the parameter beta in the results of the performance measure.

Table 10 - Non-Delay versus Active Average Makespan by Alpha

Average Makespan				
N	alpha	ACT_QWMDD	ACT_QATC	ACT_QWSLK_SPT
20	0,25	2,024	3,148	2,576
	0,5	2,979	4,113	4,024
	1	4,710	5,755	6,033
	1,5	1,616	1,921	2,085
25	0,25	1,584	2,389	2,021
	0,5	2,610	3,641	3,453
	1	3,788	4,791	4,852
	1,5	1,387	1,564	1,723
1500	0,25	0,034	0,044	0,034
	0,5	0,074	0,071	0,073
	1	0,381	0,352	0,372
	1,5	0,049	0,050	0,048
2000	0,25	0,026	0,031	0,025
	0,5	0,056	0,058	0,056
	1	0,307	0,294	0,300
	1,5	0,032	0,031	0,032

Regarding the effect of the parameters in the makespan comparison, Table 10 shows how the alpha α parameter impacts on the values obtained. The best value for the parameter is 0,25, which in all cases regardless of the size of the job instances, the smaller amount of percentage increase is observed. When looking at the worst possible value to use in the parameter, the value of 1 clearly stands out, in all cases the percentage increased is significantly greater than when using other values, if we remove the specific value from the comparison the overall difference would be considerably lower.

Table 11 - Non-Delay versus Active Average Makespan by Beta

Average Makespan				
N	beta	ACT_QWMDD	ACT_QATC	ACT_QWSLK_SPT
20	0,05	1,987	4,075	4,040
	0,25	3,017	3,591	3,447
	0,5	3,492	3,537	3,551
25	0,05	1,604	3,258	3,264
	0,25	2,376	2,851	2,717
	0,5	3,048	3,180	3,055
1500	0,05	0,121	0,115	0,118
	0,25	0,136	0,134	0,133
	0,5	0,147	0,138	0,145
2000	0,05	0,090	0,090	0,086
	0,25	0,113	0,108	0,112
	0,5	0,113	0,114	0,112

For the effect of the beta β parameter, the Table 11 indicates that the best value to be used is 0.05 where in most cases the increase in makespan is less than on the other values, having this behavior in all three dispatching rules. Therefore, we can state that when the relation of the release and due dates is small, the results are better in terms of the makespan added by the active schedule rules.

Chapter 7 - Conclusion

The main subject of this work was to study the single machine schedule problem, by proposing solutions using dispatching rules heuristics considering the weighted squared tardiness as the objective function and also taking in account jobs with unequal release dates. In literature there are very few works that tackle the same problem with the mentioned objective function and most of them only consider jobs without release dates.

The dispatching rules considered include general rules that can be applied to any problem with due date like EDD, MDD. SLK and WSLK/P. Also the linear rules which do not consider the quadratic version of the objective function WSPT, WSLK_SPT, WMDD, AR and ATC. Finally, the last set of rules are the ones, which are the linear ones modified to consider the weighted quadratic tardiness, QWSPT, QWSKL_SPT, QWMDD, QAR and QATC.

The linear and quadratic rules considered, are present in literature and they all require different setup variables that need to be adjusted in order to provide the best results, so the first step was to determine the best combination of parameters for each rule, which as shown in the parameter adjustment section, required the use of different values according to the size of the job set instances that needed to be solved. In the case of the look ahead parameter required by the AR, ATC, QAR and QATC rules, a different setup for the use of the parameter was presented since there was no definitive value for the variable that could fit all the scenarios.

In order to provide context to the obtained results, we presented solutions based on two type of schedules, which are non-delay and active. In each scenario all the dispatching rules have been tested and compared between each other and finally the overall comparison of the best rules of each group were analyzed. In all cases we have provided clear results that the rules that considered the quadratic version of the objective function considerably outperformed their linear version, and when compared to the general rules, the WSLK/P has shown to be quite effective having results which are very similar to the best quadratic rules.

When considering non-delay schedules the rule with the best overall performance was QATC followed closely by QAR, and when looking closer to the large instances of the problem the same scenario is maintained although the difference is minimum. In the case of

the active schedules, the QWMDD rule had the best overall results followed by QATC, but in terms of handling large instances, we see that QATC has better results than QWMDD and this is a factor that must be taken under consideration.

After the analysis of the best rules in each type of schedule, we proceed to compare the result of the best rules from the non-delay versus the active. The first comparison was made using the average RDI performance measure just like the previous steps, the results obtained show how the non-delay rules outperformed considerably the active ones. Although the difference in terms of objective functions is noticeable, the reason might be behind the use of a quadratic objective function in a scheduling strategy that inserts unforced idle time. This factor together with having each unit of delay considered quadratically, makes that even the smallest delay to have a great impact in the performance of the dispatching rules.

The second comparison made involved the study of the makespan time, using the value for the non-delay rules, we calculated the percentage of increase that the best active rules had against those values. The overall average of percentage increased was 2% and in the case of large size instances it was less than 0.5. This provides a better view of the impact of the active scheduling since it shows that the performance difference wasn't that much as to what the previous comparison showed.

As subjects for future research on this work, we can mention the use of the X-RM correction (Morton & Ramnath, 1995; Jouglet et al., 2008; Pfund, Fowler, Gadkari, & Chen, 2008), in the active schedule rules which may reduce the impact of considering jobs that are not yet available. Also the use of metaheuristics and branch and bound methods can be considered in order to see if the solution of small and medium sized instances can be solved optimally within reasonable computational time. Another possible addition would be the use of local search procedures as an improvement step applied to the solutions after the dispatching rules.

Chapter 8 - References

Jackson, J. R. (1955). *Scheduling a production line to minimize maximum tardiness*: Office of Technical Services.

E., S. W. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly, 3*(1-2), 59-66. doi:doi:10.1002/nav.3800030106

Miller, L. W., Conway, R. W., & Maxwell, W. L. (1967). Theory of Scheduling.

Baker, K. R. (1974). Introduction to Sequencing and Scheduling.

Kan, A. H. G. R. (1976). Machine Sequencing Problem: Classification, Complexity and Computation.

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

Zemel, E. (1981). Measuring the Quality of Approximate Solutions to Zero-One Programming Problems. *Mathematics of Operations Research, 6*(3), 319-332. doi:10.1287/moor.6.3.319

Baker, K. R., & Bertrand, J. W. M. (1982). A dynamic priority rule for scheduling against due-dates. *Journal of Operations Management, 3*(1), 37-42. doi:https://doi.org/10.1016/0272-6963(82)90020-1

Gupta, S. K., & Sen, T. (1983). Minimizing a quadratic function of job lateness on a single machine. *Engineering Costs and Production Economics, 7*(3), 187-194. doi:https://doi.org/10.1016/0167-188X(83)90012-5

Taguchi, G. (1986). Introduction to Quality Engineering: Designing Quality into Products and Processes.

Vepsalainen, A. P. J., & Morton, T. E. (1987). Priority Rules for Job Shops with Weighted Tardiness Costs. *Management Science, 33*(8), 1035-1047. doi:10.1287/mnsc.33.8.1035

Hoitomt, D. J., Luh, P. B., Max, E., & Pattipati, K. R. (1990). Scheduling jobs with simple precedence constraints on parallel machines. *IEEE Control Systems Magazine, 10*(2), 34-40. doi:10.1109/37.45792

Chu, C. (1992). A branch-and-bound algorithm to minimize total tardiness with different release dates. *Naval Research Logistics (NRL), 39*(2), 265-283. doi:10.1002/1520-6750(199203)39:2<265::AID-NAV3220390209>3.0.CO;2-L

Kim, Y.-D. (1993). Heuristics for Flowshop Scheduling Problems Minimizing Mean Tardiness. *Journal of the Operational Research Society, 44*(1), 19-28. doi:10.1057/jors.1993.3

Morton, T. E., & Ramnath, P. (1995). Guided Forward Search in Tardiness Scheduling of Large One Machine Problems. In D. E. Brown & W. T. Scherer (Eds.), *Intelligent Scheduling Systems* (pp. 79-100). Boston, MA: Springer US.

Alidaee, B., & Ramakrishnan, K. R. (1996). A computational experiment of covert-AU class of rules for single machine tardiness scheduling problem. *Computers & Industrial Engineering, 30*(2), 201-209. doi:https://doi.org/10.1016/0360-8352(95)00166-2

Kim, Y.-D., Lim, H.-G., & Park, M.-W. (1996). Search heuristics for a flowshop scheduling problem in a printed circuit board assembly process. *European Journal of Operational Research, 91*(1), 124-143. doi:https://doi.org/10.1016/0377-2217(95)00119-0

Sen, T., Dileepan, P., & Lind, M. R. (1996). Minimizing a weighted quadratic function of job lateness in the single machine system. *International Journal of Production Economics, 42*(3), 237-243. doi:https://doi.org/10.1016/0925-5273(95)00201-4

Su, L.-H., & Chang, P.-C. (1998). A heuristic to minimize a quadratic function of job lateness on a single machine. *International Journal of Production Economics, 55*(2), 169-175. doi:https://doi.org/10.1016/S0925-5273(98)00041-3

Sun, X., Noble, J. S., & Klein, C. M. (1999). Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. *IIE Transactions, 31*(2), 113-124. doi:10.1080/07408179908969810

Selim Akturk, M., & Ozdemir, D. (2000). An exact approach to minimizing total weighted tardiness with release dates. *IIE Transactions, 32*(11), 1091-1101. doi:10.1080/07408170008967464

Akturk, M. S., & Ozdemir, D. (2001). A new dominance rule to minimize total weighted tardiness with unequal release dates. *European Journal of Operational Research, 135*(2), 394-412. doi:https://doi.org/10.1016/S0377-2217(00)00319-2

Thomalla, C. S. (2001). Job shop scheduling with alternative process plans. *International Journal of Production Economics, 74*(1), 125-134. doi:https://doi.org/10.1016/S0925-5273(01)00119-0

Schaller, J. (2002). Minimizing the sum of squares lateness on a single machine. *European Journal of Operational Research, 143*(1), 64-79. doi:https://doi.org/10.1016/S0377-2217(01)00322-8

Wagner, B. J., Davis, D. J., & Kher, H. V. (2002). The Production of Several Items in a Single Facility with Linearly Changing Demand Rates. *Decision Sciences, 33*(3), 317-346. doi:10.1111/j.1540-5915.2002.tb01647.x

Baptiste, P., Carlier, J., & Jouglet, A. (2004). A Branch-and-Bound procedure to minimize total tardiness on one machine with arbitrary release dates. *European Journal of Operational Research, 158*(3), 595-608. doi:https://doi.org/10.1016/S0377-2217(03)00378-3

Chang, T.-Y., Chou, F.-D., & Lee, C.-E. (2004). A HEURISTIC ALGORITHM TO MINIMIZE TOTAL WEIGHTED TARDINESS ON A SINGLE MACHINE WITH RELEASE DATES AND SEQUENCE-DEPENDENT SETUP TIMES. *Journal of the Chinese Institute of Industrial Engineers, 21*(3), 289-300. doi:10.1080/10170660409509410

Kanet, J. J., & Li, X. (2004). A Weighted Modified Due Date Rule for Sequencing to Minimize Weighted Tardiness. *Journal of Scheduling, 7*(4), 261-276. doi:10.1023/b:josh.0000031421.64487.95

Jouglet, A., Savourey, D., Carlier, J., & Baptiste, P. (2008). Dominance-based heuristics for one-machine total cost scheduling problems. *European Journal of Operational Research, 184*(3), 879-899. doi:https://doi.org/10.1016/j.ejor.2006.11.036

Pfund, M., Fowler, J. W., Gadkari, A., & Chen, Y. (2008). Scheduling jobs on parallel machines with setup times and ready times. *Computers & Industrial Engineering, 54*(4), 764-782. doi:https://doi.org/10.1016/j.cie.2007.08.011

Su, L.-H., & Chen, C.-J. (2008). Minimizing total tardiness on a single machine with unequal release dates. *European Journal of Operational Research, 186*(2), 496-503. doi:https://doi.org/10.1016/j.ejor.2006.07.051

Valente, J. M. S., & Alves, R. A. F. S. (2008). Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. *Computers & Operations Research, 35*(11), 3696-3713. doi:https://doi.org/10.1016/j.cor.2007.04.006

Vallada, E., Ruiz, R., & Minella, G. (2008). Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research, 35*(4), 1350-1373. doi:https://doi.org/10.1016/j.cor.2006.08.016

Osman, I. H., Belouadah, H., Fleszar, K., & Saffar, M. (2009). Hybrid of the weighted minimum slack and shortest processing time dispatching rules for the total weighted tardiness single machine scheduling problem with availability constraints. *MISTA 2009 - Multidisciplinary International Conference on Scheduling: Theory and Applications, Dublin, Ireland.*

Soroush, H. M. (2009). A note on "minimizing a weighted quadratic function of job lateness in the single machine system". *International Journal of Production Economics, 121*(1), 296-297. doi:https://doi.org/10.1016/j.ijpe.2009.05.022

Valente, J. M. S., & Moreira, M. R. A. (2009). Greedy randomised dispatching heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. *The International Journal of Advanced Manufacturing Technology, 44*(9), 995-1009. doi:10.1007/s00170-008-1906-6

Soroush, H. M. (2010). Single-machine scheduling with inserted idle time to minimise a weighted quadratic function of job lateness. *European Journal of Industrial Engineering, 4*(2), 131-166. doi:10.1504/ejie.2010.031075

Valente, J. M. S. (2010). Beam search heuristics for quadratic earliness and tardiness scheduling. *Journal of the Operational Research Society, 61*(4), 620-631. doi:10.1057/jors.2008.191

van den Akker, J. M., Diepen, G., & Hoogeveen, J. A. (2010). Minimizing total weighted tardiness on a single machine with release dates and equal-length jobs. *Journal of Scheduling, 13*(6), 561-576. doi:10.1007/s10951-010-0181-1

Çakar, T. (2011). Single machine scheduling with unequal release date using neuro-dominance rule. *Journal of Intelligent Manufacturing, 22*(4), 481-490. doi:10.1007/s10845-009-0309-3

Valente, J. M. S., Moreira, M., Singh, A., & A. F. S. Alves, R. (2011). *Genetic algorithms for single machine scheduling with quadratic earliness and tardiness costs* (Vol. 54).

Schaller, J., & Valente, J. M. S. (2012). Minimizing the weighted sum of squared tardiness on a single machine. *Computers & Operations Research, 39*(5), 919-928. doi:https://doi.org/10.1016/j.cor.2011.07.018

Singh, A., Valente, J. M. S., & Moreira, M. R. A. (2012). Hybrid heuristics for the single machine scheduling problem with quadratic earliness and tardiness costs. *International Journal of Machine Learning and Cybernetics, 3*(4), 327-333. doi:10.1007/s13042-011-0067-3

Valente, J. M. S., & Schaller, J. E. (2012). Dispatching heuristics for the single machine weighted quadratic tardiness scheduling problem. *Computers & Operations Research, 39*(9), 2223-2231. doi:https://doi.org/10.1016/j.cor.2011.11.005

Cakar, T., & Köker, R. (2015). *Solving Single Machine Total Weighted Tardiness Problem with Unequal Release Date Using Neurohybrid Particle Swarm Optimization Approach* (Vol. 2015).

Davari, M., Demeulemeester, E., Leus, R., & Talla Nobibon, F. (2016). Exact algorithms for single-machine scheduling with time windows and precedence constraints. *Journal of Scheduling, 19*(3), 309-334. doi:10.1007/s10951-015-0428-y

Gonçalves, T. C., Valente, J. M. S., & Schaller, J. E. (2016). Metaheuristics for the single machine weighted quadratic tardiness scheduling problem. *Computers & Operations Research, 70*(Supplement C), 115-126. doi:https://doi.org/10.1016/j.cor.2016.01.004

Yang, T., Feng, H., Zhang, G., Zhang, W., Yang, C., Deng, R., & Su, Z. (2016). Towards Scheduling to Minimize the Total Penalties of Tardiness of Delivered Data in Maritime CPSs (Invited Paper). In Q. Yang, W. Yu, & Y. Challal (Eds.), *Wireless Algorithms, Systems, and Applications: 11th International Conference, WASA 2016, Bozeman, MT, USA, August 8-10, 2016. Proceedings* (pp. 427-439). Cham: Springer International Publishing.

Tingting, Y., Hailong, F., Jian, Z., Ruilong, D., Ying, W., & Zhou, S. (2017). Genetic optimization–based scheduling in maritime cyber physical systems. *International Journal of Distributed Sensor Networks, 13*(7), 1550147717717163. doi:10.1177/1550147717717163

Zhang, J. F., Zheng, Z. X., & Ge, T. T. (2017). Sequencing approach of arrival aircrafts based on composite dispatching rules. *Jiaotong Yunshu Gongcheng Xuebao/Journal of Traffic and Transportation Engineering, 17*(3), 141-150.