U.PORTO
FEP FACULDADE DE ECONOMIA
UNIVERSIDADE DO PORTO

Backward Scheduling Constructive Heuristics for the Single Machine Weighted Tardiness Scheduling Problem

Ana Sofia Madama Martins

Dissertation

Master in Modelling, Data Analytics and Decision Support Systems

Supervised by

Professor Jorge Valente

Professor Jeffrey Schaller

2023

Abstract

This work considers the single machine scheduling problem, with the objective of minimizing the total weighted tardiness. The main literature for this problem is reviewed with its focus on several constructive heuristics suited to this problem. These include more straightforward and general forward rules (EDD, WSPT, MDD, SLACK, and WSLKP) and some more complex and better-performing heuristics for the weighted tardiness problem (ATC, AR, WMDD, WSLK_SPT, H2, H3, PAR1, PAR2, G1_B, G2_F, and G2_B). Additionally, six backward dispatching rules are proposed for the weighted tardiness problem, adapted from previous works. Three versions are considered for each of these backward rules. Computational results show that B5 and B6 outperform the other procedures but may need excessive time for extremely large instances. The forward heuristics WMDD, WSLK_SPT, AR, and H3 achieve quality solutions within a reasonable amount of time, making these heuristics both effective and efficient. The non-dominated procedures are identified when considering both solution quality and runtime.

Keywords: Scheduling, Single Machine, Weighted Tardiness, Heuristics, Dispatching Rules

Resumo

Este trabalho considera o problema de sequenciamento numa única máquina, no qual o objetivo é minimizar o atraso ponderado. A principal literatura para este problema é revista, com o foco em heurísticas construtivas adequadas. Estas incluem regras mais simples e gerais (EDD, WSPT, MDD, SLACK e WSLKP), bem como algumas heurísticas mais complexas e de melhor desempenho para o problema de atraso ponderado (ATC, AR, WMDD, WSLK_SPT, H2, H3, PAR1, PAR2, G1_B, G2_F, e G2_B). Adicionalmente, são propostas seis regras regressivas para o problema, adaptadas de trabalhos anteriores. Três versões são consideradas para cada uma destas regras. Os resultados computacionais mostram que B5 e B6 superam os outros procedimentos, mas podem necessitar de tempo excessivo para instâncias extremamente grandes. As heurísticas WMDD, WSLK_SPT, AR, e H3 alcançam soluções de qualidade dentro de um intervalo de tempo razoável, tornando-as eficazes e eficientes. As heurísticas não dominadas são identificadas, considerando tanto a qualidade da solução quanto o tempo de execução.

Palavras-Chave: Sequenciamento, Máquina Única, Atraso Ponderado, Heurísticas, Regras de Despacho

Table of Contents

List of Tables

List of Figures

1. Introduction

This dissertation covers the single machine scheduling problem, with the objective of minimizing the total weighted tardiness. Due to the single machine environment, jobs must be processed on a single machine. It is assumed that this machine can handle only one job at a time and is continuously available for processing. Each job has a specific processing time and weight that reflects the relative importance of the job or the respective customer. Additionally, the due date corresponds to when the job should be completed. A job is classified as tardy if it is finished after its scheduled due date, and its tardiness is just the amount of time by which the job is completed late. Thus, the goal is to find an optimal schedule that minimizes the total weighted tardiness costs.

The single machine environment is shared in several real-world production systems that effectively schedule jobs on a single machine. This problem is also essential when dealing with multiple machines but with a single bottleneck machine. Moreover, single machine models may provide valuable insights into other production settings, such as parallel machines, job shops, or flow shops.

Within the scope of this dissertation, it is considered a linear tardiness objective function. The tardiness objective is one of the most used criteria when due dates are involved. It reflects the necessity of meeting the delivery dates required by clients since late deliveries can result in contractual penalties, lost sales, and loss of customer goodwill. However, other measures can be used depending on the preferences or priorities of the decision-maker, including tardiness and earliness objectives. Additionally, different settings of the weighted tardiness problem are frequently studied, involving release dates and setup times.

This problem has been studied for many years, in light of being one of the main problems in operations research and computer science. It is NP-hard (Lawler, 1977; Lenstra et al., 1977), meaning no polynomial time optimal algorithms are known. Nevertheless, several works have extensively explored methods to solve it efficiently. Among the methods studied in the past years, including exact methods, heuristics, and metaheuristics, those considering backward scheduling address a gap in the existing literature. In backward scheduling, the schedule is built from the end, and at each iteration, a job is added to the beginning of the current partial sequence. Indeed, many studies combine several approaches to dispatching

rules, from the simplest to the most complex versions. Still, nearly all consider forward rules, where the schedule is built from the beginning. A similar approach to the one held in this dissertation was conducted by Valente and Schaller (2012), where the authors compare forward and backward heuristics yet consider weighted quadratic tardiness costs.

This work is focused on dispatching rules, which are relatively simple but quite efficient procedures. These rules are widely used in practice, and most real scheduling systems are either based on them or use them to some extent since they are often the only approach capable of delivering solutions for large instances within a reasonable time. Furthermore, they are frequently used with other procedures; for instance, metaheuristics and exact methods generally resort to dispatching rules to generate an initial solution.

With this in mind, several dispatching rules are presented in this research, categorized into four types of procedures. First, general forward rules are considered, which have been used in multiple settings involving problems with due dates. Secondly, forward rules developed for the weighted tardiness objective are presented, which are the best-performing ones found in the current literature. Next are acknowledged greedy heuristics, which make locally optimal choices at each algorithm step and are also expected to perform well. Finally, backward dispatching rules are described, inspired, and adapted from the work developed by Valente and Schaller (2012). For each backward rule, three versions are considered, and an additional variant that chooses the version according to the characteristics of each specific instance. In addition, exhaustive experiments are conducted to determine the most adequate value for various parameters required by some heuristic procedures.

All these dispatching rules are applied to a dataset with greater variety than existing ones: larger instances (up to 2000 jobs), a higher number of combinations of the tardiness factor and the due date range factor, and a more considerable number of instances. Furthermore, the various heuristic methods are assessed to determine the non-dominated ones regarding solution quality and computational time. As a result, the outcomes of this work provide decision-makers valuable insights into selecting an appropriate method based on the available time for generating a solution.

This dissertation will include an in-depth analysis and exhaustive comparison of the best-performing forward dispatching rules known and novel backward heuristics suited for this problem, in contrast to alternative approaches that concentrate on a limited subset of specific

heuristics. Additionally, the heuristics undergo rigorous parameter adjustment tests, have all been implemented using a uniform programming language by the same people, and are subject to testing on diverse and exhaustive datasets with a large number of instances, ensuring fairness and rigor through the analysis.

The remainder of this work is organized as follows. Section 2 covers the problem description, and the primary literature is reviewed. All the heuristic procedures considered in this work are detailed in section 3. The experimental design is outlined in section 4, describing the problem set, performance measures, and preliminary parameter adjustment tests. Section 5 contains the computational results, including comparing the heuristic procedures and statistical tests that complement the analysis. Finally, section 6 concludes the paper.

2. Problem Formulation and Literature Review

This dissertation will consider the single machine scheduling problem with the objective of minimizing the total weighted tardiness costs. The problem can be formulated in the following way. A set of n independent jobs $\{1,2,\dots,n\}$ has to be processed on a single machine that can handle only one job at a time. The machine is considered continuously available for processing from time zero onwards, and no preemptions are allowed. Each job $j, j \in N,$ has a known processing time, p_j, and a due date, d_j, which is the date the job should be delivered. This date is measured from time zero, which is the time when all the jobs are available for processing. Additionally, each job has a weight, w_j, that reflects the relative importance of the job.

Additionally, the completion time of job j is denoted by C_j, and for a given schedule, the tardiness of the job is defined as $T_j = \max\{0, C_j - d_j\}$, and reflects the amount of time by which a job is finished late, that is, after its due date. The goal is to find a schedule for n jobs such that the sum of the weighted tardiness, $\sum_{j=1}^{n} w_j T_j$, is minimized.

Prior research has implemented diverse approaches to efficiently solve the single machine weighted tardiness problem. In recent years, the main focus has been on metaheuristics and exact methods applied in various problem settings.

Different types of metaheuristics are included in the following approaches to solve the weighted tardiness problem. Valente et al. (2011) proposed a genetic approach based on a random key alphabet and presented several algorithms based on this approach, considering both quadratic earliness and tardiness costs. Kirlik and Oguz (2012) proposed a general variable neighborhood heuristic, which proved to be effective and efficient for minimizing tardiness in the presence of setup times. More recently, Mexicano et al. (2023) made a comparative study of two metaheuristics, namely simulated annealing and tabu search, the former being the most efficient method.

Iterated local search procedures have proven to obtain outstanding results for the weighted tardiness single machine problem (Congram et al., 2002; Grosso et al., 2004; Subramanian et al., 2014). A neighborhood search technique called dynasearch, which uses dynamic programming to search an exponential-size neighborhood in polynomial time, was

introduced by Congram et al. (2002), and these results were significantly improved by Grosso et al. (2004). Subramanian et al. (2014) proposed an iterated local search heuristic, which obtained high-quality solutions within reasonable computation times. Moreover, Avci et al. (2003) proposed a problem space genetic algorithm that can improve both solution quality and robustness over other local search algorithms. Anghinolfi and Paolucci (2009) presented a Discrete Particle Swarm Optimization (DPSO) approach, which outperformed the best-known results for a specific benchmark.

Additionally, an evaluation and comparison of different metaheuristics, specifically iterated local search, variable greedy, and steady-state genetic algorithm, was made by Gonçalves et al. (2016), where the proposed metaheuristics provide an optimal solution for all the smaller problem sizes, considering weighted quadratic tardiness costs.

Equivalently, great importance has been given to exact methods. Abdul-Razaq et al. (1990) surveyed and compared exact methods. Schaller and Valente (2012) developed a branch-and-bound algorithm for optimally solving the weighted quadratic tardiness problem, and Tanaka and Araki (2013) proposed an exact algorithm for the problem with sequence-dependent setup times.

Hence, the literature on backward heuristic procedures is scarce, which clarifies the choice for this research problem. The development of new backward heuristic procedures requires forward heuristic methods to further compare and evaluate the backward methods. Several heuristic procedures were analyzed by Potts and Van Wassenhove (1991). More recently, Sen et al. (2003) provided an updated literature review of exact and heuristic methods for this linear problem.

This work addresses several general dispatching rules that work as benchmarks for several other constructive heuristics. These include the rules EDD (Jackson, 1955), WSPT (Smith, 1956), MDD (Baker & Bertrand, 1982; Vepsalainen & Morton, 1987), SLACK, and WSLKP (Panwalkar & Iskander, 1977; Vepsalainen & Morton, 1987).

Rules for the weighted tardiness objective arise as more effective when compared to the general rules; however, they require additional computational effort. Such rules are WMDD (Kanet & Li, 2004), ATC (Vepsalainen & Morton, 1987), AR (Alidaee & Ramakrishnan, 1996), and WSLK_SPT (Osman et al., 2009). More recently, new approaches have emerged,

mainly the H2 and H3 rules (Yoon & Lee, 2011) and PAR1 and PAR2 (Yin et al., 2016; Yin & Wang, 2013), that have outperformed the previously mentioned rules, when considering small problem sizes. Additionally, this research will consider two greedy procedures: G1_B (Volgenant & Teerhuis, 1999) and both forward and backward versions of the heuristic developed by Chou et al. (2005), G2_F and G2_B.

The backward dispatching rules targeted in this research are adapted from previous work developed by Valente and Schaller (2012). While Valente and Schaller focus on minimizing the squared tardiness costs, the backward heuristics in this dissertation are adapted to the linear version of the problem.

All the previously described rules will be explained in the following section, along with the priority index for each one.

3. Heuristic Procedures

3.1. General Forward Heuristics

Several heuristics, including dispatching rules, have been proposed to solve the single machine scheduling problem. This section will detail the most general and straightforward procedures, considering their use in multiple problem settings. These rules are expected to perform poorly. However, they still need to be addressed since they are often used in different environments and work as benchmarks for the specific rules presented later.

Some additional essential notations to be considered are as follows. When using forward scheduling, the current time in the scheduling procedure is denoted as t^F, and for a given job j, the slack is defined as $s_j^F = d_j - t^F - p_j$. Additionally, the average processing time of the current unscheduled jobs is denoted by p, and k is a parameter that will be addressed further in this research.

The earliest due date (EDD) rule (Jackson, 1955) is one of the first sequencing rules developed and is widely applied in scheduling problems with due dates. It schedules jobs in non-decreasing order of their due dates d_j, corresponding to using a priority index of $-d_j$. The weighted shortest processing time (WSPT) rule selects the job with the largest value of the priority index w_j/p_j at each iteration. This rule has proven to be optimal for the weighted completion time problem but also for the weighted tardiness objective when all the jobs are necessarily tardy ($p_j > d_j$ for all j) (Smith, 1956). The modified due date (MDD) (Baker & Bertrand, 1982; Vepsalainen & Morton, 1987) considers the priority index presented below, selecting, at each iteration, the job with the highest value.

$$MDD_j = \begin{cases} 1/p_j, & if\ s_j^F \leq 0 \\ 1/(d_j - t^F), & otherwise \end{cases}$$

The minimum slack (SLACK) and weighted minimum slack per required time (WSLKP) rules (Panwalkar & Iskander, 1977; Vepsalainen & Morton, 1987) also introduce priority indexes considered benchmarks to the more specific heuristics. The SLACK rule selects, at each iteration, the job with the minimum slack, which corresponds to choosing the job with

the largest value of the priority index of $-s_j^F$. The WSLKP selects, at each iteration, the job with the highest value of the following index.

$$WSLKP_j = \begin{cases} -(w_j/p_j) * s_j^F, & if\ s_j^F \leq 0 \\ -(p_j/w_j) * s_j^F, & otherwise \end{cases}$$

3.2. Forward Heuristics for the Weighted Tardiness Objective

The following dispatching rules include the best-performing procedures for the weighted tardiness problem. These rules perform better than the general rules presented in the previous section.

The weighted modified due date (WMDD) was introduced by Kanet and Li (2004), and the apparent tardiness costs rule (ATC) by Vepsalainen and Morton (1987). The AR rule denotes the heuristic that provided the best performance among the dispatching rules analyzed by Alidaee and Ramakrishnan (1996). Prior research has thoroughly investigated efficient dispatching rules for the weighted tardiness problem, and these three heuristics have proven to give a good performance.

The distinction of these heuristics relies on the choice of the priority index when a job is early. The conclusion mentioned above by Smith (1956) that the WSPT rule provides optimal schedules when all the jobs are necessarily tardy is reflected in these three procedures. Thus, they select the job with the highest value of the priority index w_j/p_j when a job is completed after its due date. Alternatively, the WMDD, ATC, and AR rules differ only in the priority index for when a job is still early, reducing the job's priority in those cases. Indeed, the earlier the job is, the higher the reduction in the priority. Additionally, the ATC and the AR rule are provided with a lookahead capability, represented by the parameter k. It is related to the number of competing critical jobs close to becoming tardy (Vepsalainen & Morton, 1987).

Briefly, these heuristics select, at each iteration, the job with the highest value of the following priority indexes:

$$WMDD_j = \begin{cases} w_j/p_j, & if\ s_j^F \leq 0 \\ w_j/(d_j - t^F), & otherwise \end{cases}$$

$$ATC_j = \begin{cases} w_j/p_j, & if\ s_j^F \leq 0 \\ (w_j/p_j)\ \exp\left(-s_j^F/kp\right), & otherwise \end{cases}$$

$$AR_j = \begin{cases} w_j/p_j, & if\ s_j^F \leq 0 \\ (w_j/p_j)\left[kp/(kp + s_j^F)\right], & otherwise \end{cases}$$

The weighted minimum slack/shortest processing time (WSLK_SPT) heuristic (Osman et al., 2009) selects, at each iteration, the job with the minimum value of the weighted slack or weighted processing time, which corresponds to a priority index of:

$$WSLK\_SPT_j = \begin{cases} w_j/p_j, & if\ s_j^F \leq p_j \\ w_j/s_j, & otherwise \end{cases}$$

Yoon and Lee (2011) proposed three new heuristics, denoted by H1, H2, and H3. The results concluded that H2 and H3 outperformed H1, so this report will solely focus on H2 and H3. These two heuristics acknowledge the case where a job has a large due date and a substantial weight, which can result in scheduling first jobs that are not urgent.

H2 can be described in the following way. At each time t in the scheduling procedure, this rule computes a time limit denoted by T. If there is at least one unscheduled job with $d_j \leq T$, the job with the largest value of $max\ \{p_j, d_j - t\}/w_j$ is selected. When there is no scheduled job with $d_j \leq T$, we select the job with the lowest value of $(d_j - t)/w_j$. The time limit T is determined with the following expression. $T = \frac{(\alpha+\beta)}{2}$, where $\alpha = max\{t + p_j, d_j\}$ and $\beta = min\{t + p_j, d_j\}$, of the unscheduled job set.

H3 works similarly, but from the unscheduled jobs with $d_j \leq T$, it selects the one with the lowest value of p_j/w_j. Additionally, this heuristic uses the expression $T = max\{t + p_{j^*}, d_{j^*}\}$ to calculate the time limit, where j^* represents the job selected by WMDD rule.

More recently, a new pair of constructive heuristics was introduced to handle the weighted tardiness problem for a single machine (Yin et al., 2016; Yin & Wang, 2013). These two heuristics, denoted by PAR1 and PAR2, are described as follows. At each point of the scheduling procedure, the unscheduled jobs are analyzed. If there are no tardy jobs in this set, i.e., all jobs are early or on time, the job with the earliest due date (EDD) is selected to

be processed. If all the unscheduled jobs are tardy, the job with the smallest p_j/w_j is selected. Finally, if there are both tardy and non-tardy jobs, job i is selected from the non-tardy jobs, and job j is selected from the tardy jobs, using the previous two rules, respectively. The one with the smaller incremental objective value will be processed first. This works by calculating $w_j * (t + p_i + p_j - d_j)$ and $w_j \times (t + p_j - d_j) + w_i * max\{t + p_j + p_i - dj, 0\}$. Then, if $w_i * (t + p_i + p_j - dj) > w_j \times (t + p_j - dj) + w_i * max\{t + p_j + p_i - dj, 0\}$ we select job j. Otherwise, job i is chosen. The PAR2 heuristic goes accordingly yet uses a different rule when the unscheduled jobs are early. Here, the job with the lowest value of $(d_j - t)/w_j$ is selected.

3.3. Greedy Heuristics

The greedy heuristics that will be considered are presented here. A greedy algorithm makes choices based on what looks best at the moment: at each stage, it selects the local optimum to try to find a global optimum.

The first heuristic to be considered is the Greedy_V1 (Volgenant & Teerhuis, 1999). This greedy procedure is designed to work backward and goes as follows. All pairs (i, j) of yet unscheduled jobs are considered at each iteration. Initially, the priority of each unscheduled job is set to 0. Then, for each pair of jobs (i, j), the jobs are scheduled in the two possible orders (i, then j, and j, then i), and the cost of these two jobs in each order is calculated. The priority of the job that comes last in the ordering that provides a lower cost is increased by 1. Finally, the job with the highest priority is chosen. Further in this report, this heuristic will be addressed as G1_B.

The second heuristic is identified as Greedy_V2 (Chou et al., 2005). There are two phases in this algorithm. The second phase applies a local search procedure that is not of interest to this research. Thus, the first phase is considered here and can be described as such. At each iteration, one unscheduled job is chosen as the currently chosen job. Then, the other unscheduled jobs are checked, one at a time. The currently chosen job and the other unscheduled job being considered are scheduled in the two possible orders. If the order in which the currently chosen job is scheduled first has a lower (or equal) cost, then the currently

chosen job remains unchanged. If the order in which the other unscheduled job comes first has a lower cost, then that job becomes the new currently chosen job. After going through all the unscheduled jobs, the currently chosen job is selected. This heuristic was designed to work forward and will be addressed in this report as G2_F. Nevertheless, a backward version of this was created that works equivalently. At each iteration, one unscheduled job is chosen as the currently chosen job, and the other unscheduled jobs are checked one at a time. The currently chosen job and the other unscheduled job being considered are scheduled in the two possible orders. If the order in which the currently chosen job is scheduled last has a lower (or equal) cost, then the currently chosen job remains unchanged. If the order in which the other unscheduled job comes last has a lower cost, then that job becomes the new currently chosen job. After going through all the unscheduled jobs, the currently chosen job is selected and placed at the beginning of the partial sequence. This backward version will be referred to as G2_B.

3.4. Backward Heuristics

Apart from the greedy heuristics described before, another six backward rules will be considered in this work. The notation described in the forward heuristics is applied here similarly. As for the notation not described yet, in the backward rules, we use s_j^B, which represents the slack of job j when using backward scheduling and is defined as $s_j^B = t^B - d_j$. t^B is the current time in the backward scheduling, i.e., the time at which the job to be scheduled will be completed. Finally, p_{max} is the maximum processing time of the current unscheduled jobs.

For each backward heuristic, three versions will be presented, the difference being in the p_j^{mod}. This represents the processing time of job j, which may differ in the three versions. In the first and basic version, p_j^{mod} is simply p_j, the processing time of the job. The other two versions modify p_j^{mod} to consider the job's tardiness. Let us consider T^{min}, the minimum positive tardiness of all the jobs to be scheduled, and T_j^{min}, the minimum positive tardiness of all jobs other than j. Accordingly, in the second and third versions (Mod1 and Mod2), p_j^{mod} equals $min\{p_j, T^{min}\}$, and $min\{p_j, T_j^{min}\}$, respectively. Finally, the value v

represents a parameter that measures the slack type weight, which will be explained in detail further in this section. The rules for the six backward heuristics considered in this work are presented below.

$$Back\_V1 = \begin{cases} p_j, & if\ s_j^B \leq 0 \\ -(w_j/p_j^{mod}), & otherwise \end{cases}$$

$$Back\_V2 = \begin{cases} p_j, & if\ s_j^B \leq 0 \\ -(w_j/p_j^{mod})s_j^B, & otherwise \end{cases}$$

$$Back\_V3 = \begin{cases} p_j, & if\ s_j^B \leq 0 \\ -(w_j/p_j^{mod})[s_j^B - \max(t^B - p - d_j; 0)], & otherwise \end{cases}$$

$$Back\_V4 = \begin{cases} p_j, & if\ s_j^B \leq 0 \\ -(w_j/p_j^{mod})[s_j^B - \max(t^B - p_{max} - d_j; 0)], & otherwise \end{cases}$$

$$Back\_V5 = \begin{cases} p_j, & if\ s_j^B \leq 0 \\ -(w_j/p_j^{mod})[s_j^B - v(\max(t^B - p - d_j; 0)], & otherwise \end{cases}$$

$$Back\_V6 = \begin{cases} p_j, & if\ s_j^B \leq 0 \\ -(w_j/p_j^{mod})[s_j^B - v(\max(t^B - p_{max} - d_j; 0)], & otherwise \end{cases}$$

The back versions presented in this report are adjusted from the work done by Valente and Schaller (2012), where the authors explore and compare forward and backward heuristics to minimize the weighted quadratic tardiness. Thus, in this paper, the backward heuristics they presented will be equivalent but applied to the linear problem.

All these dispatching rules are designed to schedule an early job whenever one is available, using a positive priority index, the processing time of that job. Thus, this job will have an objective function value of zero and decrease the tardiness of all the currently tardy jobs. On the other hand, a negative priority index is set to tardy jobs. The backward heuristics differ only in this case, i.e., in the choice of the priority index when a job is tardy.

As Valente and Schaller explained, the two alternative versions arose from the following reasons. Suppose we consider tardy jobs: the higher the processing time, the higher the job's

priority. When a job has a more significant processing time, it will decrease the tardiness of the other jobs because it makes the current time go further back than if the p_j was small. This might even cause one of those other jobs to become early.

Nonetheless, the large processing time of a job may increase the priority index more than an adequate value. Let us consider two tardy jobs with two processing times, p_j and p_k, where $p_j < p_k$, where the scheduling of either leads to another job becoming early in the next iteration. This means that selecting either job j or k to be scheduled at the current iteration will allow another job to become early in the next iteration. Hence, the two jobs have different processing times, which does not translate into any advantage for scheduling j first and then k. To overcome this effect, the second and third versions, Mod1 and Mod2, are built to reduce the p_j of a job when the processing time is larger than the time required for some other job to become early.

The Back_V1 is the simpler heuristic of the backward rules considered. All the others are derived from Back_V1 and take into account different aspects that hopefully increase the performance of this heuristic. It only considers the weight and the processing time of the job to be scheduled, which corresponds to using the WSPT rule when all unscheduled jobs are tardy. Back_V2 is very similar to Back_V1 but acknowledges the slack of the job, increasing the priority index if the slack is smaller. This results in scheduling first jobs less tardy than others, all else unchanged.

The priority expression for the dispatching rules Back_V3 and Back_V4 combines the Back_V2 with an additional term. This term follows the same line of thought of the modified versions as it captures the reduction of a job j tardiness if another job is scheduled in the current iteration. Indeed, when there is a larger reduction in the objective function value of j for some other job being selected for processing, this favors not scheduling job j at the current iteration. Furthermore, this term captures the average processing time of the current unscheduled jobs, p, in Back_V3, and the maximum of those processing times, p_{max}, in Back_V4.

Finally, the Back_V5 and Back_V6 merge the priority indices of Back_V2 with Back_V3 (and Back_V2 with Back_V4, respectively). These rules were included because, as explained by Valente and Schaller (2012), preliminary tests proved that the corresponding heuristic to

the one denoted here as Back_V3 outperformed the other backward rules. The exception was Back_V2 that for specific settings provided better results, specifically when several jobs were about to become early. Thus, Back_V5 introduces the parameter v, which allows the priority index to change between those of Back_V2 and Back_V3, depending on the characteristics of the currently unscheduled jobs. Accordingly, Back_V6 follows the same logic, changing between Back_V2 and Back_V4 priority indexes.

The parameter v, denominated in this work as the slack type weight, is calculated at each iteration as follows. Let us consider s^B the average of s_j^B values, this is, the average of the slack of the currently unscheduled jobs. When $p \geq s^B$, v is set at 0. In fact, in this situation, the slacks are small, and with this setting, it is ensured that the Back_V5 (and Back_V6) priority index turns into the index of the Back_V2 rule since the parameter is zero. On the other hand, when $p < s^B$, there are two possible outcomes. If $s^B/t^B > w$, where w is a user-defined value that works as the critical slack ratio, v is set at 1. Indeed, in this context, the slacks are large, so setting the parameter at 1 will convert in using the index of Back_V3 (and Back_V4, equivalently). Finally, if $s^B/t^B \leq w$, v is equal to $(s^B - p)/s^B$. This represents a situation where the slacks are neither large nor small, so a fair parameter v is achieved by calculating an intermediate value. Thus, the priority index of Back_V5 (and Back_V6) is set somewhere between the indexes of Back_V2 and Back_V3 (and Back_V4) heuristics, according to the relative size of the slack.

4. Experimental Design

This section presents the experimental design, including a description of the problem set used to obtain the computational results, which will be presented in the next section. Further, the performance measures and the preliminary tests performed to find the optimal values for the parameters required by some heuristics will be explained.

4.1. Problem Set

The computational experiments were performed on a randomly generated set of problems with different settings regarding the number of jobs and processing times, with multiple combinations of due date tightness and range. To test the different heuristics procedures, a new dataset was created for the need for larger problem sizes and a higher number of instances. Two different datasets were created: a test set to test the heuristic procedures and a separate train set to perform the parameter adjustment tests to avoid possible overfitting. Both sets were randomly generated in the following way. The sets have 50, 100, 250, 500, 1000, 2000 jobs. For each job, the processing time, p_j, was randomly generated from a uniform distribution [1, 100], and the weight, w_j, was obtained from a uniform distribution [1, 10].

For each job j, the due dates were generated from a uniform distribution $P[1 - T - R/2; 1 - T + R/2]$, where P is the sum of processing times of all jobs, T is the tardiness factor, and R is the range factor. The latter two parameters considered values of 0.2, 0.4, 0.6, 0.8, and 1.0.

For each combination of n, T, and R, were generated 10 instances for the train set, resulting in a total of 250 instances for each n. As for the test set, 100 instances were generated for each combination of n, T, and R, totaling 2500 instances for each problem size.

The procedures were coded in Python and executed on a personal computer with a Windows 10 64–bit operating system, an Intel Core i7 4770 3.4G processor, and 16 GB RAM. For the statistical tests, it was used SPSS and the autorank Python package (Herbold, 2020).

4.2. Performance Measures

The comparison of the heuristic procedures will mostly rely on two different performance measures that will be used together to provide a more accurate analysis of all the rules.

The first measure of performance, previously used by Valente and Schaller (2012), is denoted by relative improvement versus the worst result (ivw). The ivw for heuristic H_i, when examined with heuristics $H_1, H_2, \ldots, H_z$, is computed in the following way. Let ofv_{worst} be the objective function value of the worst heuristic, and similarly, ofv_{best} be the best value of all the z heuristics considered. When $ofv_{worst} = ofv_{best}$, the relative improvement versus the worst result is set at 0. Otherwise, the ivw is calculated as $(ofv_{worst} - ofv_{H_i})/ofv_{worst} * 100$, with ofv_{H_i} being the objective function value of the heuristic H_i. Thus, higher ivw values indicate superior performances, as they quantify the improvement offered by a specific heuristic over the worst result provided among all the heuristic methods.

The second measure to analyze the different rules is the relative deviation index (rdi). The notation described for ivw works accordingly in this performance measure. Thus, and as happens with ivw, when $ofv_{worst} = ofv_{best}$, the relative deviation index is set at 0. Otherwise, the relative deviation index is computed using the expression $(ofv_{H_i} - ofv_{best})/(ofv_{worst} - ofv_{best})$. This performance measure yields values ranging from 0 to 1, with those approaching 0 indicating superior performance compared to values close to 1.

The reason for the choice of the ivw measure was described in Valente and Schaller (2012). When due dates are relatively loose, or there is a wide range of due dates, identifying a schedule with no tardy jobs becomes straightforward, yielding an objective function value of 0. However, when one or more heuristics achieve an optimal solution with an objective function value of 0, using metrics such as the deviation from the best heuristic result becomes problematic due to the potential for division by 0, resulting in errors in the performance measure. To mitigate this concern, the approach of the relative improvement versus the worst result is set at 0 for specific scenarios so that division by 0 does not occur. The only situation where the denominator would be 0 is if all dispatching rules find an optimal solution with an objective function value equal to 0. In this case, all procedures would be optimal, and we would have $ofv_{worst} = 0$. In such a manner, and as mentioned previously, the

possible scenario where division by 0 would occur, i.e., when $ofv_{worst} = ofv_{best}$, is overcome by setting ivw to zero in those cases.

Equivalently, the relative deviation index procedure encounters problems for scenarios that cause a division by zero, precisely when the objective function attains a value of 0 for both the best and worst heuristic methods or when these two have the same result. To this extent, when $ofv_{worst} = ofv_{best}$, a value of 0 is attributed to the relative deviation index.

The logic behind using both performance methods instead of just one was the following. Let us consider a problem where $ofv_{worst} = 300$, $ofv_{best} = 0$, and $ofv_{H_i} = 150$. With this setting, the rdi measure will have a value of 0.5. Conversely, a different problem with $ofv_{worst} = 300$, $ofv_{best} = 280$, and $ofv_{H_j} = 290$ will also measure a rdi of 0.5. Indeed, the heuristics H_i and H_j, although they have strongly different objective function values, achieve the same rdi. Naturally, in these scenarios, the rdi would not be the most accurate measure to compare the performance of these two heuristics. At the same time, ivw would be more suitable for these cases, computing two distinct values in the different scenarios.

Likewise, let us consider again two different problem scenarios. The first has $ofv_{worst} = 300$, and $ofv_{H_i} = 200$. As anticipated, the ivw for this scenario would be 50%. Alternatively, another problem with $ofv_{worst} = 8$ and $ofv_{H_j} = 4$ would have an equal ivw of 50%, although in the first case, the improvement is much larger in absolute value. Thus, when the objective function values are small, ivw can still give large values that correspond to small absolute values. Neither one of these measurements (ivw and rdi) is perfect, so using both measures together always provides a more in-depth analysis than using just one.

Finally, the computational time (in seconds) required by the different heuristic procedures is also analyzed to compare the efficiency of all the methods.

4.3. Parameter Adjustment Tests

As mentioned in section 3.2., the ATC and AR heuristic procedures require a parameter k, denoted as the lookahead capability, and is related to the number of jobs close to becoming tardy. Several approaches have been previously tested to determine a value for parameter k.

A possible approach is to set k at a fixed value, but the one used in this work consists of a variable k parameter. This approach allows the lookahead parameter to change at each iteration, increasing k with the rising count of critical competing jobs. The implementation of this approach is outlined below.

A critical slack value, s_{crit}^F, is initially computed at each iteration. This value is established as $s_{crit}^F = s_{prop}^F P_{unsch}^F$, with s_{prop}^F being the critical slack proportion, that ranges between 0 and 1 $(0 < s_{prop}^F < 1)$, and P_{unsch}^F corresponding to the sum of the processing times of the unscheduled jobs. A particular job j is categorized as a critical job if its slack factor, s_j^F, satisfies the condition $0 < s_j^F \leq s_{crit}^F$, which is indicative of the job being not yet tardy $\left(0 < s_j^F\right)$ but on the verge of becoming so $(s_j^F \leq s_{crit}^F)$. Consequently, k equals the number of critical jobs or 0.5 if no job is critical. This value represents the smallest fixed value for k considered in prior studies.

Parameter adjustment tests were conducted in the above-mentioned train set to find the best values for the critical slack proportion used in ATC and AR rules. The values {0.05, 0.1, 0.15, …, 0.9, 0.95} were considered for s_{prop}^F. In the AR heuristic procedure, the s_{prop}^F that provided the best results was consistent across all problem sizes and T and R parameter combinations. Thus, s_{prop}^F was set at 0.05.

For ATC, the results from the train set were not cohesive across different parameter combinations, and it was possible to conclude that choosing a single value for s_{prop}^F would not lead to results up to the standards of the ATC procedure. Examining the ivw and rdi results, it was explicit that a critical slack proportion of 0.5 worked best when R<0.2, and a s_{prop}^F of 0.05 was better for the remaining cases. With this in mind, a new rule denoted as ATC_Formula was developed. This new heuristic combined both cases: for R<0.2, s_{prop}^F was set at 0.5, and for R>0.4, it was set at 0.05. For measurements falling within the range of 0.2 to 0.4, a value was determined through interpolation. Two approaches were considered to implement the ATC_Formula: the first by calculating the value of R and choosing s_{prop}^F only at the beginning of the instance, and the second by applying this procedure at each iteration. The results of this preliminary test will be addressed later in this section.

Back_V5 and Back_V6 rules, as noted earlier in section 3.4., require using a parameter w, the critical slack ratio. The values {0.05, 0.1, 0.15, …, 0.9, 0.95} were considered for this parameter. For Back_V5 and its alternative versions, w was set at 0.5, while for Back_V6 and its variants, w was placed at 0.55.

Each backward dispatching rule (from Back_V1 to Back_V6) was considered in three versions: the basic version, Mod1, and Mod2. An extensive analysis of the ivw and rdi values for all these heuristic procedures concluded that the best version for each rule depended on T, making the selection of a single version infeasible. Thus, a novel heuristic was formulated for each backward rule, comprising a fusion of distinct versions. The appropriate version to be employed depended upon the value of T. The subsequent table presents the optimal version for each rule, corresponding to different values of T. For T between 0.6 and 0.8 (and alternatively between 0.4 and 0.6), a value was determined via interpolation.

Heuristic	T	Best Version
Back_V1	0.2, 0.4 and 0.6	Back_V1_Mod2
	0.8 and 1	Back_V1
Back_V2	0.2 and 0.4	Back_V2_Mod1
	0.6, 0.8 and 1	Back_V2
Back_V3	0.2, 0.4 and 0.6	Back_V3_Mod2
	0.8 and 1	Back_V3
Back_V4	0.2, 0.4 and 0.6	Back_V4_Mod2
	0.8 and 1	Back_V4
Back_V5	0.2 and 0.4	Back_V5_Mod2
	0.6, 0.8 and 1	Back_V5
Back_V6	0.2 and 0.4	Back_V6_Mod2
	0.6, 0.8 and 1	Back_V6

Table 1 - Best version for each backward rule, depending on T.

As it happened in the ATC_Formula, two approaches were considered: one by calculating the T value at the beginning of the instance and the other at each iteration. The new combined rules are labeled as Back_V1_Combo and Back_V1_Combo_Iter, depending on the approach being applied at the beginning or applied at each iteration, respectively. The same

applies to Back_V2_Combo and Back_V2_Combo_Iter (and to the rest of the backward rules) and ATC_Formula and ATC_Formula_Iter.

The outcomes of the different methodologies applied to the aforementioned rules are presented in Table 2. Runtimes for both approaches exhibited high similarities, thus prompting the focus on the ivw and rdi measures. The best value for each problem size and heuristic procedure is in bold.

	ivw						rdi					
	50	100	250	500	1000	2000	50	100	250	500	1000	2000
ATC_Formula	0.64	0.66	0.25	0.23	0.33	0.34	0.18	0.20	0.21	0.23	0.22	0.22
ATC_Formula_Iter	**1.40**	**1.20**	**1.40**	**1.32**	**1.05**	**0.90**	**0.07**	**0.06**	**0.04**	**0.03**	**0.03**	**0.04**
Back_V1_Combo	**0.12**	**0.08**	**0.06**	**0.05**	**0.04**	**0.04**	**0.11**	**0.12**	**0.10**	**0.09**	**0.09**	**0.09**
Back_V1_Combo_Iter	0.12	0.07	0.04	0.04	0.04	0.04	0.11	0.13	0.15	0.15	0.16	0.16
Back_V2_Combo	**0.11**	**0.05**	**0.02**	**0.00**	**0.00**	**0.00**	**0.06**	**0.06**	**0.07**	**0.05**	**0.07**	**0.06**
Back_V2_Combo_Iter	0.03	0.02	0.01	0.00	0.00	0.00	0.12	0.15	0.17	0.18	0.17	0.17
Back_V3_Combo	0.07	0.05	0.03	0.02	0.01	0.01	**0.10**	0.12	**0.10**	**0.09**	**0.09**	**0.08**
Back_V3_Combo_Iter	**0.09**	**0.06**	**0.03**	**0.02**	**0.02**	**0.02**	0.11	**0.11**	0.14	0.15	0.15	0.16
Back_V4_Combo	**0.05**	**0.04**	0.02	0.01	0.01	0.00	**0.09**	**0.11**	**0.11**	**0.11**	**0.09**	**0.08**
Back_V4_Combo_Iter	0.04	0.03	**0.02**	**0.01**	**0.01**	**0.01**	0.10	0.11	0.12	0.13	0.14	0.15
Back_V5_Combo	**0.17**	**0.16**	**0.12**	**0.07**	**0.06**	**0.05**	0.13	**0.13**	**0.11**	**0.09**	**0.06**	**0.05**
Back_V5_Combo_Iter	0.17	0.11	0.04	0.02	0.01	0.00	**0.12**	0.16	0.19	0.20	0.24	0.25
Back_V6_Combo	**0.13**	**0.09**	**0.07**	**0.06**	**0.04**	**0.03**	**0.10**	**0.11**	**0.11**	**0.10**	**0.09**	**0.09**
Back_V6_Combo_Iter	0.09	0.09	0.04	0.04	0.01	0.00	0.11	0.13	0.15	0.17	0.19	0.19

Table 2 - Comparison of the two approaches for ATC and the backward rules, by n.

Regarding the ATC_Formula, it is evident that the strategy applied at each iteration consistently provides better results. Concerning the backward rules, although there is not a uniform pattern across all values of n, it is notable that the version implemented at the beginning yields superior average results compared to the approach executed at each iteration. In fact, the ivw values are substantially small, considering they are presented in percentage. Thus, in

those cases where ivw and rdi are not consistent, the choice was made focusing primarily on the rdi metric.

Statistical tests were conducted to determine the statistical significance of differences between each rule and its corresponding iteration approach. Given the uniform application of two methods to the same instances, a paired-sample test can be conducted. The non-parametric Wilcoxon Signed-Rank test was selected since the paired-samples t-test normality assumption was not entirely fulfilled.

The test was applied to each pair of heuristics, and the significance level was set at 0.05. The outcomes of these tests revealed that the differences between each pair are statistically significant. In fact, the hypothesis that the beginning and iteration approaches have similar performance was consistently rejected across all pairs of procedures. Thus, and under the analysis of the ivw and rdi measures, the heuristics applied at each iteration performed significantly better than the beginning version for the ATC. In contrast, the opposite was verified for the backward rules.

Given careful consideration of all these factors, from this point onward, this study will employ the ATC_Formula_Iter version along with the six backward procedures using the beginning approach. The versions with the opposite approaches, as well as the individual backward versions and their modified counterparts, will no longer be considered in the remainder of this work. In order to streamline the denomination of the heuristic procedures, the term ATC_Formula_Iter will henceforth be simplified to ATC. Similarly, the designation Back_V1_Combo will be modified to B1. This naming convention will also extend to the remaining backward heuristics, which will be named B2, B3, B4, B5, and B6.

5. Computational Results

The results outlined in Table 3 describe the runtime allocated to each heuristic procedure, expressed in seconds. Although problem sizes of 50 and 100 jobs were also considered, Table 3 solely focuses on instances with n greater than 250, resulting from the considerably negligible runtimes associated with the smaller problem sizes, which often approach zero and make the comparison with the larger problem sizes impractical. The heuristic procedures are organized in ascending order of the runtime for n=2000.

As expected by its intrinsic simplicity, EDD emerges as the fastest heuristic method accounted for in this paper, regardless of the problem size considered. Conversely, the G1_B stands out as the most time-intensive heuristic. As a matter of fact, the intention was initially to apply this heuristic across all problem sizes; however, its complexity made it unfeasible to apply to problems with 500 or more jobs. Indeed, the runtime for n=250 is already notably excessive. Consequently, only this latter value is presented within the results.

Besides the aforementioned heuristic, G2_B and G2_F exhibit the highest computational time within comparable values of the remaining heuristics' runtimes.

Upon comprehensive analysis of Table 3, it becomes evident that the simplest heuristic methods, such as EDD, SLACK, WSLKP, MDD, and WMDD, demonstrate the shortest execution times, aligning with their inherently straightforward nature. The PAR1 and PAR2 rules, while retaining an element of simplicity compared to the backward dispatching rules, occupy the uppermost positions within the table.

Contrarily, the backward dispatching rules assume the lowermost ranks, barring the greedy heuristics. This observation prompts the conclusion that these backward dispatching rules are slower than the forward heuristics. Finally, rules such as ATC, AR, WSLK_SPT, and H2 and H3 exhibit greater complexity than the general forward methods, positioning them within the midrange of the table.

	Runtime			
Heuristic	250	500	1000	2000
EDD	0.00	0.00	0.00	0.00
PAR1	0.02	0.09	0.38	1.53
PAR2	0.03	0.11	0.47	1.83
SLACK	0.03	0.13	0.54	2.13
WSLKP	0.05	0.20	0.81	3.23
MDD	0.05	0.20	0.82	3.25
WMDD	0.05	0.21	0.85	3.40
WSLK_SPT	0.05	0.21	0.85	3.42
H2	0.06	0.24	0.96	3.92
AR	0.07	0.28	1.10	4.50
H3	0.07	0.28	1.17	4.75
ATC	0.08	0.32	1.29	5.16
B1	0.09	0.35	1.40	5.70
B2	0.10	0.39	1.57	6.28
B3	0.11	0.43	1.68	6.83
B4	0.11	0.44	1.82	7.29
B5	0.12	0.46	1.87	7.59
B6	0.12	0.50	2.01	8.05
G2_F	0.15	0.65	2.54	10.14
G2_B	0.17	0.67	2.67	10.72
G1_B	13.96			

Table 3 - Runtime of the heuristic procedures, in ascending order of n=2000.

Regarding the evaluation of solution quality, Table 4 furnishes a detailed overview of the top-performing rules, considering both ivw and rdi. The arrangement of the heuristic procedures in this table follows a descending order based on the mean of the ivw values for all problem sizes. While the order of the heuristic methods differs slightly for rdi, it maintains a substantial similarity, reinforcing both performance measures' consistency.

Starting the analysis from the lowermost part of the table reveals that the worst heuristic methods are positioned toward the bottom of the table. These include EDD, SLACK, MDD, and WSLKP. Similarly, both PAR heuristics, along with the simplest backward rules, namely B1 and B2, are situated within the lower half of the table. Among the remaining backward dispatching rules, B5, B6, and B4 are positioned in the table's upper half, notably excelling in terms of ivw and rdi performance.

Directing attention to WMDD and WSLK_SPT, which have achieved favorable positions in terms of runtime, their notable performance is further reflected in their favorable ivw and rdi metrics. These confirm their effectiveness as heuristics that excel in both solution quality and runtime.

	ivw						rdi					
Heuristic	50	100	250	500	1000	2000	50	100	250	500	1000	2000
B5	67.61	68.75	69.41	69.57	69.76	69.82	0.010	0.008	0.004	0.003	0.002	0.002
B6	67.65	68.63	69.12	69.22	69.41	69.44	0.009	0.009	0.008	0.008	0.007	0.007
B4	66.25	67.18	67.59	67.67	67.99	68.02	0.029	0.028	0.028	0.028	0.026	0.026
WMDD	64.52	67.62	68.12	68.07	68.11	68.05	0.043	0.021	0.022	0.024	0.026	0.027
H3	64.27	67.61	68.10	68.01	68.00	67.91	0.046	0.021	0.022	0.025	0.027	0.029
WSLK_SPT	63.57	67.30	68.12	68.09	68.14	68.06	0.056	0.025	0.022	0.024	0.025	0.027
B3	65.63	66.76	67.33	67.53	67.85	67.94	0.037	0.034	0.031	0.030	0.028	0.027
G2_B	65.35	66.56	67.25	67.44	67.82	67.92	0.040	0.036	0.032	0.031	0.028	0.027
H2	65.52	67.38	67.61	67.37	67.23	67.14	0.035	0.025	0.029	0.034	0.038	0.039
AR	64.11	66.47	67.62	67.88	68.07	68.06	0.054	0.039	0.031	0.029	0.029	0.029
ATC	65.50	66.73	67.38	67.51	67.60	67.48	0.040	0.037	0.034	0.034	0.035	0.037
PAR2	64.39	66.64	67.51	67.70	67.91	67.95	0.049	0.036	0.031	0.029	0.029	0.029
B1	64.49	66.00	66.89	67.26	67.67	67.80	0.051	0.043	0.036	0.033	0.030	0.029
G1_B	65.38	66.54	67.25				0.040	0.036	0.032			
PAR1	63.37	64.85	65.63	65.99	66.18	66.32	0.068	0.061	0.057	0.053	0.053	0.051
B2	64.52	64.79	64.48	64.23	64.10	63.99	0.069	0.079	0.089	0.094	0.097	0.099
WSLKP	56.04	58.45	59.92	60.41	60.77	60.91	0.188	0.168	0.153	0.147	0.144	0.142
MDD	51.85	52.67	52.72	52.61	52.63	52.69	0.267	0.268	0.272	0.275	0.277	0.276
G2_F	38.09	37.40	38.04	38.44	39.05	39.39	0.342	0.354	0.348	0.343	0.336	0.332
EDD	27.81	26.33	25.14	24.60	24.30	24.17	0.682	0.717	0.740	0.749	0.755	0.757
SLACK	20.40	22.52	23.50	23.75	23.87	23.96	0.788	0.771	0.764	0.762	0.761	0.760

Table 4 - Comparison of all heuristic procedures.

In order to provide additional support for the preceding analysis, statistical tests were carried out to determine if the differences between each pair of rules were statistically significant. These statistical tests were conducted for both ivw and rdi values associated with all heuristic procedures. Initially, an analysis involving normality tests and Q-Q Plots for each heuristic method was undertaken, which rejected the null hypothesis that the population was normal for all heuristic procedures. Therefore, we assume that not all populations were normal, revealing that the assumptions of the repeated measures ANOVA test were not all met.

In light of the preceding, the non-parametric Friedman test was applied to determine if there were any significant differences between the median values of the populations. The significance level was set at 0.05. Additionally, it was applied the post-hoc Nemenyi test to infer which differences were significant. The following was observed for both ivw and rdi values.

The Friedman test's null hypothesis (p=0.000), suggesting the absence of differences in the central tendency among all populations, was rejected. Consequently, it was assumed that there was a statistically significant difference between the median values of the populations. Based on the post-hoc Nemenyi test, it was assumed that there are no significant differences within the following groups: ATC, AR, B1, and B2; PAR2 and B4; B4 and H2; H2 and WMDD. All other differences are significant.

The critical distance (CD) of the post-hoc Nemenyi test for ivw is revealed in Figure 1, whereas for rdi is presented in Figure 2. The groups of heuristics with non-significant differences are represented with a black horizontal line. The arrangement of heuristics is displayed in ascending order, positioning the worst-performing one at the far-left end and the best one at the far-right. These findings align with Table 4, particularly placing B5 and B6 as the top-performing heuristics. Notice that the ranking achieved by ivw and rdi is identical.

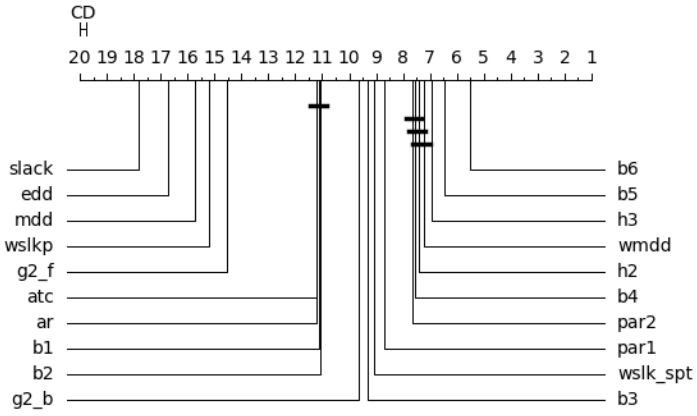

Figure 1 – Critical distance for ivw values

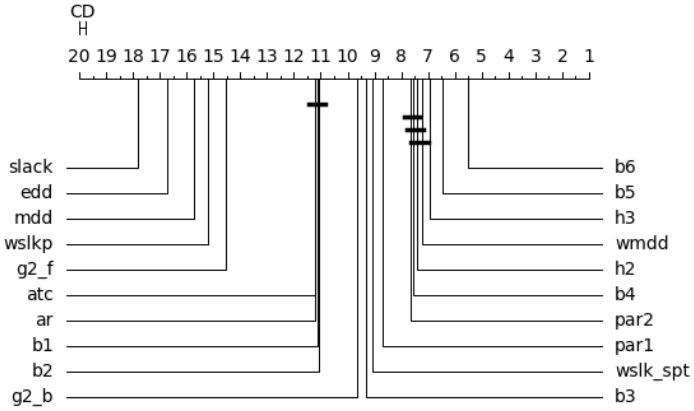

Figure 2 – Critical distance for rdi values

Furthermore, the results regarding solution quality and computational times can be combined to achieve the non-dominated heuristics, i.e., heuristics that are not overcome by any other in terms of ivw and runtime, simultaneously. These performance measures were compared for n=2000, and the results are displayed in Figures 3 through 5.

Figure 3 displays all the heuristic procedures mentioned in this work, except for EDD. This heuristic has shallow runtime values compared to the rest of the methods, making its inclusion in the graphs unfeasible. Figure 4 excludes the heuristics with the most extreme values to simplify the analysis, making the differences in the heuristics with higher performances more explicit. Additionally, Figure 5 considers exclusively the non-dominated heuristics.

PAR1 emerges as the fastest of the non-dominated procedures, apart from EDD. It takes shallow computational time, although it does not obtain solutions with equivalent quality. On the contrary, PAR2, with a runtime reasonably close to PAR1, achieves much better performances, reaching an average improvement of 68% versus the worst heuristic for problem sizes of 2000 jobs. On the same note, WMDD and WSLK_SPT have even more remarkable performances but are slightly slower. The heuristic with the best solution quality is, undoubtedly, B5, although it is also the slower method, taking more than double the time required by WMDD or WSLK_SPT. These results guide decision-makers in the choice of the heuristic procedure. Indeed, as the time available to generate a solution increases, the decision maker can switch from PAR2 to WMDD and WSKK_SPT and finally to B5.

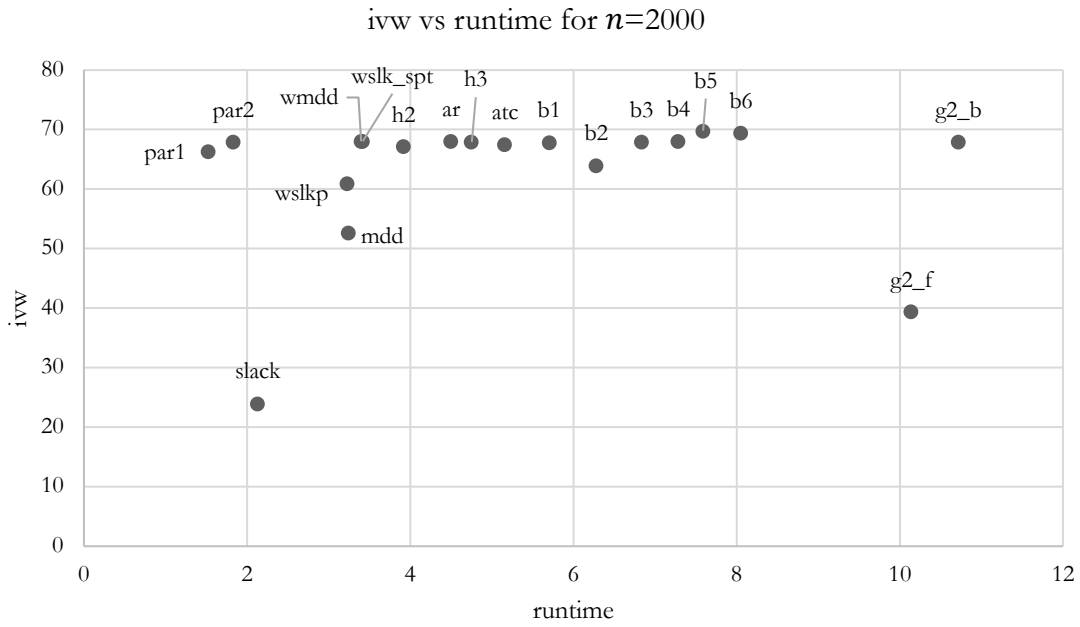

Figure 3 - ivw vs runtime for all heuristic procedures, with n=2000

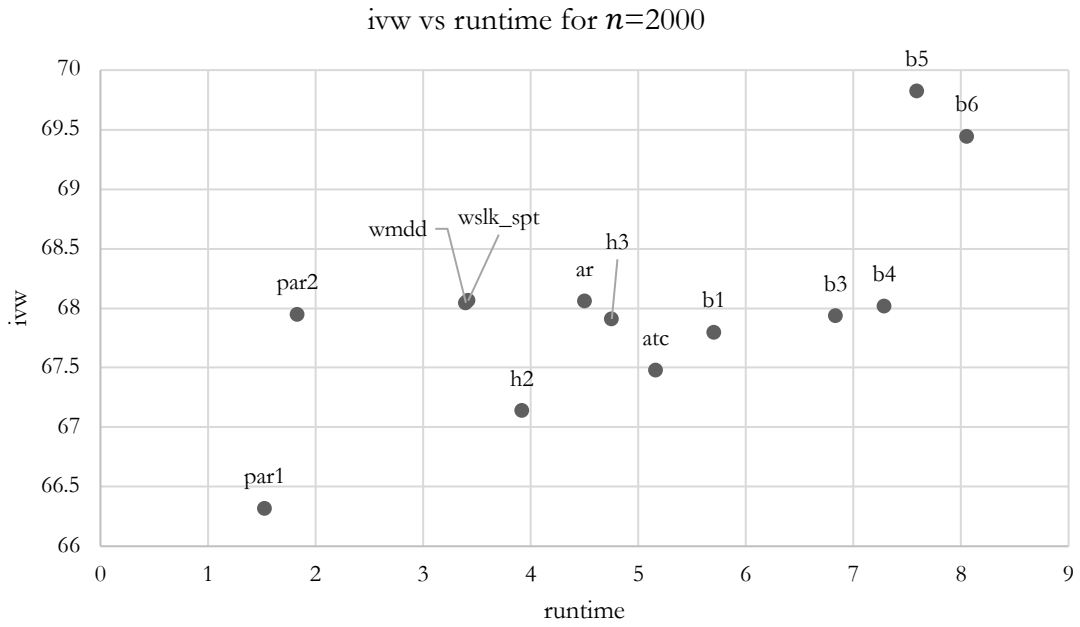

Figure 4 - ivw vs runtime for heuristics procedures, excluding the most extreme ones, with n=2000

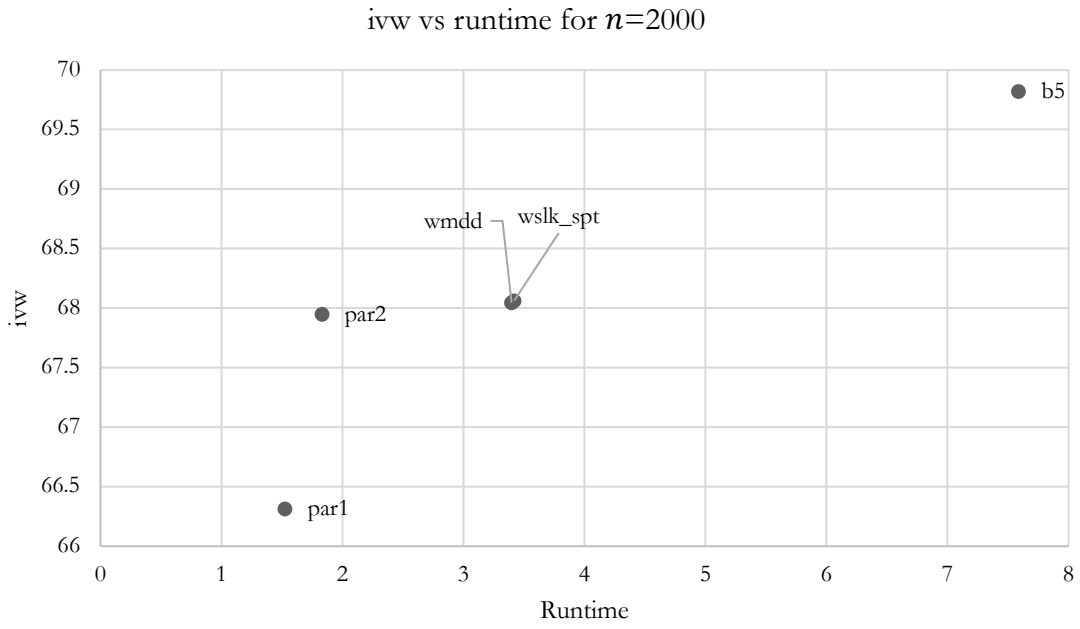

Figure 5 - ivw vs runtime for non-dominated heuristics, with n=2000

Finally, Table 6 provides the improvement versus the worst result (ivw) and the relative deviation index (rdi) for each combination of the tardiness factor and the range of due dates parameters. This table focuses solely on the non-dominated heuristic procedures.

The improvement versus worst result tends to exhibit a decrease as the tardiness factor T increases. It is worth noting that the ivw can achieve remarkable values, occasionally almost reaching the uppermost limit of 100% when the tardiness factor is set to a lower value. These large ivw values frequently correspond to relatively minor absolute improvements. Indeed, when T assumes a low value, only some jobs will be tardy, subsequently leading to relatively small objective function values. However, the effect of the range of due dates R on the ivw metric lacks a straightforward interpretation. Precisely, the ivw frequently displays an ascending trend with increasing R when the tardiness factor equals 0.2, 0.4, or 0.6. However, this trend reverses, leading to a descending pattern when T assumes values of 0.8 or 1.0. These conclusions are consistent with the rdi measure, which has the inverse behavior; for instance, the relative deviation index tends to decrease as the tardiness factor increases.

		ivw						rdi					
T	*R*	B5	WMDD	WSLK_SPT	PAR2	PAR1	EDD	B5	WMDD	WSLK_SPT	PAR2	PAR1	EDD
0.2	0.2	76.87	73.94	73.99	72.53	56.36	14.09	0.018	0.056	0.055	0.074	0.281	0.823
	0.4	97.99	92.97	92.20	94.54	97.65	96.72	0.006	0.059	0.067	0.042	0.010	0.020
	0.6	99.67	98.24	98.23	98.67	99.67	99.67	0.000	0.014	0.014	0.010	0.000	0.000
	0.8	99.50	98.78	99.20	98.82	99.50	99.50	0.000	0.007	0.003	0.007	0.000	0.000
	1.0	99.67	99.30	99.42	99.32	99.67	99.67	0.000	0.004	0.003	0.003	0.000	0.000
0.4	0.2	72.91	68.33	68.42	67.22	53.94	5.60	0.002	0.065	0.064	0.080	0.262	0.925
	0.4	72.25	66.19	66.41	63.36	59.39	4.32	0.005	0.090	0.087	0.129	0.182	0.941
	0.6	72.17	70.77	70.63	66.94	65.39	6.98	0.024	0.043	0.044	0.094	0.115	0.908
	0.8	94.96	91.98	90.38	93.74	94.42	88.79	0.014	0.047	0.064	0.028	0.021	0.087
	1.0	98.35	93.55	91.59	96.61	97.79	95.74	0.002	0.053	0.074	0.021	0.009	0.035
0.6	0.2	66.10	61.64	61.70	60.94	53.49	3.63	0.001	0.069	0.068	0.079	0.191	0.945
	0.4	68.20	62.75	62.96	60.90	59.06	2.38	0.001	0.081	0.078	0.109	0.135	0.965
	0.6	72.13	69.04	69.15	66.88	66.19	2.25	0.002	0.044	0.043	0.075	0.085	0.968
	0.8	75.58	75.88	75.52	74.07	73.87	2.23	0.010	0.005	0.010	0.030	0.032	0.970
	1.0	68.77	69.15	68.40	68.44	68.36	1.98	0.011	0.005	0.017	0.015	0.016	0.970
0.8	0.2	57.94	55.98	56.01	55.60	53.28	2.73	0.001	0.035	0.034	0.041	0.081	0.953
	0.4	59.05	59.29	59.28	59.18	59.11	1.59	0.006	0.002	0.002	0.004	0.005	0.973
	0.6	57.80	58.13	57.98	58.12	58.10	1.10	0.006	0.000	0.003	0.001	0.001	0.981
	0.8	54.73	54.95	54.77	54.95	54.93	0.81	0.005	0.001	0.004	0.000	0.001	0.985
	1.0	51.21	51.39	51.17	51.39	51.38	0.63	0.004	0.001	0.005	0.001	0.001	0.987
1.0	0.2	45.45	45.46	45.44	45.46	45.46	1.85	0.000	0.000	0.000	0.000	0.000	0.959
	0.4	44.27	44.28	44.26	44.29	44.29	0.99	0.000	0.000	0.001	0.000	0.000	0.978
	0.6	42.59	42.62	42.58	42.62	42.62	0.64	0.001	0.000	0.001	0.000	0.000	0.985
	0.8	41.28	41.33	41.27	41.33	41.33	0.52	0.001	0.000	0.002	0.000	0.000	0.987
	1.0	39.45	39.49	39.42	39.49	39.49	0.40	0.001	0.000	0.002	0.000	0.000	0.990

Table 5 - ivw and rdi for the non-dominated heuristics, by *T* and *R*

6. Conclusion

This work considers the single machine scheduling problem with a weighted tardiness objective function. The main goal of this research was to identify the best heuristic methods to solve this problem efficiently. Thus, a broad group of procedures was acknowledged in this dissertation. The focus shifted between the simpler forward dispatching rules, forward rules suited to the weighted tardiness objective, greedy heuristics, and backward dispatching rules. These latter were previously used in a different setting but were suitably adapted to take the linear objective into account.

Meticulous preliminary tests were conducted to determine the adequate values of the parameters required by each heuristic procedure. In fact, for the ATC heuristic, the appropriate value for the parameter was dependent upon the value of R, which encouraged the use of an adapted rule that would take this relation into account. Additionally, each of the backward rules and its two alternative counterparts were also a target of this approach. Thus, based on the best-performing version of the backward rules for each T value, the various alternatives were combined in novel methods. This approach was implemented at the beginning of the instance and alternatively at each iteration.

Results showed that greedy heuristics are not very efficient, and most are neither effective. Rules B5 and B6 significantly outperform the other backward procedures, as well as the forward methods, despite taking more time to generate solutions. As expected, the forward heuristics suited for the weighted tardiness objective are substantially more effective than the general rules considered. In particular, WMDD, WSLK_SPT, AR, and H3 achieve quality solutions within a reasonable amount of time, especially for larger problem sizes, making these heuristics both effective and efficient.

The computational results also showed that EDD, PAR1, PAR2, WMDD, WSKP_SPT, and B5 procedures emerge as non-dominated when considering both solution quality and runtime. An extensive look at the performance measures for different T values concluded that ivw has an inverse relation with the tardiness factor, with the opposite happening to rdi. However, the performance measures do not display an explicit trend with the range of due dates R.

The highlight of his work resides in the fact that all heuristics methods, from forward to backward ones, were applied and tested on the same problem set, including medium and large problem sizes and with a large number of instances. This approach ensures transparency in the comparison of all procedures, allied with a uniform programming language used by the same team of developers to exhaustively adjust parameters and test all heuristic procedures.

A possibility for future work would be considering these heuristics procedures, particularly the new approaches of the backward rules, under other machine settings, such as parallel machines or flow shops. Another possibility would be to include additional problem settings, such as release dates and setup times.

7. References

Abdul-Razaq, T. S., Potts, C. N., & Van Wassenhove, L. N. (1990). A survey of algorithms for the single machine total weighted tardiness scheduling problem [Article]. *Discrete Applied Mathematics*, *26*(2-3), 235-253. https://doi.org/10.1016/0166-218X(90)90103-J

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

Anghinolfi, D., & Paolucci, M. (2009). A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times [Article]. *European Journal of Operational Research*, *193*(1), 73-85. https://doi.org/10.1016/j.ejor.2007.10.044

Avci, S., Akturk, M. S., & Storer, R. H. (2003). A problem space algorithm for single machine weighted tardiness problems [Article]. *IIE Transactions (Institute of Industrial Engineers)*, *35*(5), 479-486. https://doi.org/10.1080/07408170304390

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

Chou, F. D., Chang, T. Y., & Lee, C. E. (2005). A heuristic algorithm to minimize total weighted tardiness on a single machine with release times [Article]. *International Transactions in Operational Research*, *12*(2), 215-233. https://doi.org/10.1111/j.1475-3995.2005.00499.x

Congram, R., Potts, C., & Van De Velde, S. (2002). An Iterated Dynasearch Algorithm for the Single-Machine Total Weighted Tardiness Scheduling Problem. *INFORMS Journal on Computing*, *14*, 52-67. https://doi.org/10.1287/ijoc.14.1.52.7712

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

Grosso, A., Della Croce, F., & Tadei, R. (2004). An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem [Article]. *Operations Research Letters*, *32*(1), 68-72. https://doi.org/10.1016/S0167-6377(03)00064-6

Herbold, S. (2020). Autorank: A Python package for automated ranking of classifiers. *Journal of Open Source Software*, *5*, 2173. https://doi.org/10.21105/joss.02173

Jackson, J. R. (1955). *Scheduling a production line to minimize maximum tardiness*. Office of Technical Services. https://books.google.de/books?id=4jnPJgAACAAJ

Kanet, J. J., & Li, X. (2004). A weighted modified due date rule for sequencing to minimize weighted tardiness [Article]. *Journal of Scheduling*, *7*(4), 261-276. https://doi.org/10.1023/B:JOSH.0000031421.64487.95

Kirlik, G., & Oguz, C. (2012). A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine [Article]. *Computers and Operations Research*, *39*(7), 1506-1520. https://doi.org/10.1016/j.cor.2011.08.022

Lawler, E. L. (1977). A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness [Article]. *Annals of Discrete Mathematics*, *1*(C), 331-342. https://doi.org/10.1016/S0167-5060(08)70742-8

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of Machine Scheduling Problems. In P. L. Hammer, E. L. Johnson, B. H. Korte, & G. L. Nemhauser (Eds.), *Annals of Discrete Mathematics* (Vol. 1, pp. 343-362). Elsevier. https://doi.org/https://doi.org/10.1016/S0167-5060(08)70743-X

Mexicano, A., Carmona-Frausto, J. C., Montes-Dorantes, P. N., Cervantes, S., Cervantes, J. A., & Rodríguez, R. (2023). Simulated Annealing and Tabu Search for Solving the Single Machine Scheduling Problem [Conference paper]. *Lecture Notes in Networks and Systems*, *571 LNNS*, 86-95. https://doi.org/10.1007/978-3-031-19945-5_8

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.

Panwalkar, S. S., & Iskander, W. (1977). A Survey of Scheduling Rules. *Operations Research*, *25*(1), 45-61. http://www.jstor.org/stable/169546

Potts, C. N., & Van Wassenhove, L. N. (1991). Single Machine Tardiness Sequencing Heuristics. *IIE Transactions*, *23*(4), 346-354. https://doi.org/10.1080/07408179108963868

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

Sen, T., Sulek, J. M., & Dileepan, P. (2003). Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey [Article]. *International Journal of Production Economics*, *83*(1), 1-12. https://doi.org/10.1016/S0925-5273(02)00265-7

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

Subramanian, A., Battarra, M., & Potts, C. N. (2014). An Iterated Local Search heuristic for the single machine total weighted tardiness scheduling problem with sequence-

dependent setup times [Article]. *International Journal of Production Research*, *52*(9), 2729-2742. https://doi.org/10.1080/00207543.2014.883472

Tanaka, S., & Araki, M. (2013). An exact algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times [Article]. *Computers and Operations Research*, *40*(1), 344-352. https://doi.org/10.1016/j.cor.2012.07.004

Valente, J. M. S., Moreira, M. R. A., Singh, A., & Alves, R. A. F. S. (2011). Genetic algorithms for single machine scheduling with quadratic earliness and tardiness costs [Article]. *International Journal of Advanced Manufacturing Technology*, *54*(1-4), 251-265. https://doi.org/10.1007/s00170-010-2921-y

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

Vepsalainen, A. P. J., & Morton, T. E. (1987). PRIORITY RULES FOR JOB SHOPS WITH WEIGHTED TARDINESS COSTS [Article]. *Management Science*, *33*(8), 1035-1047. https://doi.org/10.1287/mnsc.33.8.1035

Volgenant, A., & Teerhuis, E. (1999). Improved heuristics for the n-job single-machine weighted tardiness problem [Article]. *Computers and Operations Research*, *26*(1), 35-44. https://doi.org/10.1016/S0305-0548(98)00048-3

Yin, A., Punnen, A. P., & Hu, D. (2016). Priority allocation rules for single machine total weighted linear and square tardiness problems [Article]. *Production Engineering*, *10*(4-5), 471-476. https://doi.org/10.1007/s11740-016-0680-9

Yin, A., & Wang, J. (2013). Mixed dispatch rule for single machine total weighted tardiness problem [Article]. *Journal of Applied Sciences*, *13*(21), 4616-4619. https://doi.org/10.3923/jas.2013.4616.4619

Yoon, S. H., & Lee, I. S. (2011). New constructive heuristics for the total weighted tardiness problem [Article]. *Journal of the Operational Research Society*, *62*(1), 232-237. https://doi.org/10.1057/jors.2009.186