HEURISTICS FOR A PERMUTATION FLOWSHOP SCHEDULING PROBLEM WITH WEIGHTED SQUARED TARDINESS

by

Maria Raquel Carvalho Costa

Dissertation for Master in Modelling, Data Analysis and Decision Support Systems

in Optimization

Advised by

Jorge Valente

Jeffrey Schaller

2015

Biographical introduction of the candidate

Raquel Costa was born in Viana do Castelo in 1989.

In 2010, she concluded her bachelor in Economics in Universidade do Minho and, in the same year, she started her career as a customer's care specialist in a well-known Portuguese telecommunication company.

Seeking for more knowledge, the candidate applied for the Master in Modelling, Data Analysis and Decision Support Systems in order to expand her capabilities in the job market.

Raquel Costa is an avid reader and enthusiastic traveller and currently works as a bankruptcy manager.

Abstract

One of the main issues that industries should try to prevent is customers' dissatisfaction caused by tardy jobs, since it means loss of sales and other financial inconvenients, as well as loss of goodwill. It is also proven that the weight of each job should be taken into account due to the fact that every job has its own associated priority. This investigation considers these two factors and presents algorithms for a permutation flowshop scheduling problem with a weighted squared tardiness objective which includes a comparison between linear and quadratic constructive heuristics and the application of three improvement methods in the QATC and the QWMDD rules, after concluding that these rules have the best performances. Overall, this study proves that the quadratic heuristics outperform their linear counterpart and that the application of the improvement methods results in improvements of over 40%.

Keywords: permutation flowshop, weighted squared tardiness, dispatching rules, scheduling

Table of Contents

List of tables

Chapter 1 - Introduction

1.1 Motivation

In various sectors, but especially in industries, a quite large variety of jobs has to go through different operations on a number of different machines. Scheduling problems have been studied for several years due their major importance in various industries; among its most important goals are meeting due dates and avoiding delay penalties.

A flowshop can be defined as a conventional manufacturing system in which machines are arranged in the order in which operations are performed on jobs and the operation sequence is the same for all jobs (Parthasarathy and Rajendran 1998). The machines are set up in a series, and whenever a job completes its processing on one machine, it joins the queue at the next. A particular case of this sequencing type is a permutation flowshop which does not allow bypassing, i.e. the order of the jobs has to be the same on all machines.

There are two very important factors that this study takes into account: job priority (weights) and squared tardiness. On one hand (Valente and Schaller 2012) proved that heuristics which take in consideration the quadratic tardiness have a better performance than the heuristics developed for the linear problem, since it gives more importance to the jobs that are more tardy, avoiding large delays and situations where most of the tardiness occurs in a small group of jobs . On the other hand, (Vepsalainen and Morton 1987) showed that a strategic weight should be considered in order to reflect different job priorities. These two factors represent customer's dissatisfaction, loss of future sales, and rush shipping costs, among others.

In what regards the type of procedure, this work focuses on dispatching rules. First, this is often the only method that can find solutions for large instances in adequate computational times. Also, dispatching rules and their priority indexes are often used in real life scheduling systems. Furthermore, the solution provided by these rules is also used as the starting point for some improvement methods, such as local search procedures or metaheuristics.

1.2 The Problem

This study considers a permutation flowshop scheduling problem with weighted quadratic tardiness costs. Formally, the problem can be stated as follows.

A set $N = \{1,2,\dots,n\}$ of n independent jobs have to be processed on a set $M = \{1,2,\dots,m\}$ of m machines. All jobs follow the same route through the machines, and it is assumed that the processing order of the jobs is the same for all machines, so the production environment is the so-called permutation flowshop. The machines are continuously available from time zero onwards, and preemptions are not allowed.

Job $j, j \in N$, requires a processing time p_{ij} on machine $i, i \in M$, and has a weight w_j and a due date d_j. Let C_{ij} denote the completion time of job $j, j \in N$ on machine $i, i \in M$. Furthermore, let the job sequenced in position j be denoted by $[j]$ and recall that $C_{1[0]} = 0$, since all machines are available at time zero. Then, $C_{1[j]} = C_{1[j-1]} + p_{1j}$ and $C_{k[j]} = max\{C_{k-1[j]}, C_{k[j-1]}\} + p_{1j}$, for $k = \{2,3,\dots,m\}$. Finally, and for convenience, let the completion time of job j, that is, the time at which job j finishes processing on the last machine, also be denoted by C_j, so $C_j = C_{mj}$.

For a given schedule, the tardiness of job j is defined as $T_j = \max\{C_j - d_j; 0\}$. The objective is then to find a schedule that minimizes the sum of the weighted squared tardiness values $\sum_{j=1}^{n} w_j T_j^2$.

Let S be the current partial schedule, that is, the sequence of jobs that are scheduled so far. Also, let $C_j(S)$ be the completion time of job $j \notin S$ if j is scheduled at the end of sequence S. Let $s_j(S)$ be the slack of job $j \notin S$ if j is scheduled at the end of sequence S, where $s_j(S) = d_j - C_j(S)$. Additionally, let $t_i(S)$ be the current availability time of machine i under schedule S. For convenience, the current availability time on the first machine will also be denoted by t, so $t = t_1(S)$.

Let $P_j(S) = C_j(S) - t$ be the total time (total processing time plus any eventual forced idle time) between the start and finish of job $j \notin S$ if j is scheduled at the end of sequence S. Also, let $\bar{P}(S)$ be the average, over all jobs $j \notin S$, of the $P_j(S)$ values.

Let $T_j(S) = max\{C_j(S) - d_j; 0\}$ be the tardiness of job $j \notin S$ if j is scheduled at the end of sequence S.

1.3 Structure

As previously explained, this dissertation aims to study different heuristics for a permutation flowshop scheduling problem with weighted squared tardiness. The remainder of this work is organized as follows.

Chapter 2 summarizes the scientific articles that had been presented over the last decades. Mainly, they introduce solutions for a single machine scheduling and linear or squared earliness and tardiness, and were extremely useful for this particular investigation, since they can be easily adapted for it.

Chapter 3 presents all the heuristic procedures considered in the study, such as the constructive dispatching rules and the improvement methods applied to improve the solutions generated by these dispatching heuristics.

Chapter 4 presents the computational results. First, the quadratic heuristics will be compared with their linear counterparts; the ones with an inferior performance will not be considered further. Secondly, the remaining heuristics will be compared with those that do not have both a linear and a quadratic version. Additionally, some improvement methods (multiple sequence version, NEH procedure and local search) will be taken in consideration in order to find an upgraded version of the best procedures found previously. The best procedures and its upgrade will also be compared.

Finally, Chapter 5 will summarize all the work done, comparing the various heuristics examined and concluding about the best procedure that should be use in real life situations, highlighting the most important conclusions. It will also provide some suggestions for future research.

Chapter 2 – Literature review

The authors know of no other works about weighted quadratic tardiness in a permutation flowshop scheduling problem. However, there are some studies considering these measures for a single machine (which can be easily applied in more complex problems) as well as the sum of linear earliness and tardiness in single machines and permutation flowshop scheduling problems.

In order to find a schedule that minimizes the sum of the weighted squared tardiness, (Valente and Schaller 2012) proposed some heuristics for scheduling in a single machine regarding forward and backward scheduling. In the case of forward dispatching rules, their study considered the best performing rules in the literature (WMDD, ATC and AR) for the linear problem and their quadratic version (QWMDD, QATC and QAR). The backwards dispatching rules were take into account for comparison purposes.

This study shows that the quadratic version of the rules outperform better than their linear counterparts. Also, it proved that the backward scheduling heuristics perform much better compared to the forward scheduling heuristics. This makes sense if we take into account that the backward procedure chooses between the tardy jobs first, contrarily to the forward scheduling, in which jobs can be quite early in the first iterations.

(Valente and Schaller 2012) concluded that backwards scheduling heuristics are efficient and effective and that can give a quick result for large instances with good results close to the optimum.

(Schaller and Valente 2012) presented a branch-and-bound algorithm to apply in a single machine sequencing problem with the objective of minimizing the sum of weighted squared tardiness. Also, in this study methods to increase the efficiency of an optimal branch-and-bound algorithm are developed.

When branching occurs and new nodes are created, a lower bound on the sum of weighted squared tardiness that would be obtained by the completion of the partial

sequence corresponding to those nodes is calculated. If the lower bound is less than the lowest sum of weighted squared tardiness found so far for complete sequences and the node does not represent a complete sequence, the node is retained for additional branching. If the lower bound is less that the incumbent value and all the jobs have been sequenced in the branch ending with the node (the node represents a complete sequence), then the incumbent value is updated to equal the lower bound, the sequence is recorded and the node is eliminated (Schaller and Valente 2012). If the lower bound is greater that the incumbent value, the node is eliminated. The algorithm proposed uses a depth first strategy, i.e. the algorithm chooses for branching the node at the lowest level of the tree, breaking the connection choosing the node with the least lower bound.

This study, which took into account different number of jobs, degrees of tightness and ranges of due dates, proved that the proposed dominance conditions significantly improved the efficiency of the branch-and-bound algorithm. It was also shown that problems with up to 40 jobs can be solved in a practical amount of time; the same occurred for larger problems if due dates were not very tight.

A genetic algorithm to minimize total earliness and tardiness in permutation flowshops was proposed by (Schaller and Valente 2013), where the chromosomes were randomly created using EDD for one chromosome and NEH for another chromosome. This algorithm applied a selection operator (n-tournament) in which the individual with the lowest total earliness and tardiness, between a percentage of individuals, is chosen for the mating process. This procedure takes in consideration that the best parents have more jobs copied to their progenitors, and consequently, might lead to better children.

The heuristic presented by (Schaller and Valente 2013) was tested in different problem sizes and it generated better solutions, when compared with other algorithms.

There are other important investigations that should be mentioned due to their relevance for the scheduling problem here in study, especially the studies which take into account weighted versions of some well-known dispatching rules.

(Vepsalainen and Morton 1987) made a weighted version of the COVERT rule, first introduced by Carroll (1965), and defined a new rule called Apparent Tardiness Cost (ATC), which has been mentioned in the literature countless times. The COVERT

priority rule represents the expected tardiness cost per unit of imminent processing time, or cost over time; the ATC trades off job's urgency against machine utilization.

With this study, (Vepsalainen and Morton 1987) concluded that weighted COVERT is often much better than simple rules such as EDD (Earliest Due Date), FCFS (First Come-First Served) and WSPT (Weighted Shortest Processing Time), and the ATC rule outperforms COVERT consistently. They also considered that the weights criterion sums the economic performance of the rules, i.e. inventory holdings, rush shipping, customer badwill, among others.

On one hand, this study shows that the ATC rule is robust since it ranks first in all load conditions, and, on the other hand, that the COVERT rule is reliable in congested shops and with tight due dates.

(Ow and Morton 1989) introduced introduced two new rules (LIN-ET and EXP-ET) for a new search method called Filtered Beam Search and compared it with some already existing rules.

Beam Search is a heuristic search method that, without backtracking, searches β candidate solution paths in parallel (the beamwidth), and saves only β paths at each stage. In the filtered method, the evaluation of the best at each stage is done with a heuristic that passes some number of nodes, up to αβ (α is the filterwidht), to another that selects up to β nodes from them.

Adjacency is a necessary condition for an optimal schedule and is based on the assumption that a globally optimal schedule must also be locally optimal so that no improvement can be gained by a pairwise interchange of adjacent jobs.

Accordingly to (Morton et al 1984), a locally optimal sequence is defined as one that cannot be improved by interchanging the positions of adjacent pairs of jobs. Local optimality is a necessary but not sufficient condition for global optimality.

LIN-ET is a linear priority rule derived after (Morton et al 1984)'s rule; it estimates the priority of a job by taking into account its impact on the next k jobs.

In this study, they also considered EXP-ET, which is a combination of two other functions and substitutes an intermediate function of LIN-ET: one reflects a priority that focuses on the tardiness cost of a job as its slack becomes smaller; and other illustrates the situation when the slack is large and the early cost dominates.

This paper examined both heuristics and search methods for the single machine early/tardy problem. Overall, EXP-ET showed a better performance than LIN-ET. The priority function EXP-ET appears to be quite accurate in that when used to schedule jobs using the dispatch method, relatively good schedules are obtained.

(Parthasarathy and Rajendran 1998) proposed heuristics based on simulated annealing (SA) with the objective of minimizing weighted mean tardiness of jobs in flowshops and cells; their procedure involves two phases: the first one is the determination of a seed sequence and the second is the improvements of this seed sequence using a simulated annealing algorithm. The novelty in their algorithm is two new perturbation schemes: Random Insertion Perturbation Scheme (RIPS) and Curtailed Random Insertion Perturbation Scheme (CRIPS).

This study, based on computational evaluation, shows that the proposed simulated annealing heuristics have a superior performance than the existing ones for problems of scheduling in flowshops and manufacturing cells. Also, the proposed heuristics have good results when job with different relative weights for tardiness are considered.

(Kanet and Li 2004) compared the performance of several rules in a weighted tardiness scheduling problem. They also developed a weighted version of the Modified Due Date (MDD), developed by Baker and Bertrand (1982), and modified some well-known rules for this problem. WMDD is a combination of WSPT and a weighted remaining allowance (WRA), i.e. it minimizes weighted tardiness when all jobs are tardy and gives preference to jobs with larger tardiness weight and less slack.

This study proved WMDD and ATC (once again) to be statistically superior when compared with WCOVERT, WRA and WSPT, taking in account unrestricted, proportional and agreeable weights; another advantage of the WMDD rule is its simplicity over ATC and WCOVERT (which ranked third place).

(Hasija and Rajendran 2004) developed a simulated annealing algorithm to minimize the total tardiness in flowshops, considering a JIBIS (job-index-based-insertion-scheme) to improve the sequences, which consists in inserting each job in all the possible positions; this scheme always finds a solution that is better or equal than the seed sequence.

In this study two perturbation schemes are proposed: JSB (job-shift-based) and PSS (probabilistic-step-swap). On the first scheme, a job in the seed sequence is chosen based on a probabilistic function and is inserted either to the right or to the left of its original position; on the second one, a new sequence is generated first from the seed sequence by probabilistically swapping jobs next to each other and then by probabilistically swapping jobs farther away.

At the finishing point, the ten best sequences are subjected to the JIBIS with the purpose of reaching the global minimum around those points of local minima.

This proposed proposed heuristic when compared with the Armentano and Ronconi (1999)'s tabu search and the Parthasarthy and Rajendran (1998)'s simulated annealing heuristic showed superior performance.

(Ruiz and Stützle 2008) considered the minimization of the makespan and the minimization of the total weighted tardiness and presented two Iterated Greedy algorithms for flowshop problems. One is called IG_RS and it interacts over a greedy construction heuristic (NEH); this procedure has a destruction phase in which some jobs are removed from the current sequence, and it has also a subsequent construction phase in which the heuristic is applied to reconstruct the sequence, reinserting the jobs that were removed. The other one also incorporates an extension phase that applies local search and is called IG_RS(LS). The results showed that these two algorithms have a very good performance and were simple to apply in real-world environments.

A WMSPT (Weighted Minimum Slack Shortest Processing Time) rule was proposed by (Osman et al 2009); it is a parameter-free heuristic that combines WSPT and WMS (Weighted Minimum Slack) rules. This dispatching rule schedules jobs one at a time taking into account a priority ranking index calculated for the remaining unscheduled jobs.

The authors showed that WMSPT is a very competitive rule proving to be the most effective and efficient procedure among those that were tested. Furthermore, this rule captures the best characteristics of WMS and WSPT and is very easy to implement, since it does not have any parameter to estimate.

The most recent paper found, (M'Hallah 2014), takes into account the minimization of earliness and tardiness for a permutation flowshop that uses a Variable Neighbourhood Search and a Variable Neighbourhood Descent to find a good sequence of jobs.

The Variable Neighbourhood Search moves from its current local sequence when it discovers a better solution or when it stagnates; this algorithm thoroughly changes the neighbourhood it is exploring to search for a near-global minimum or to escape from local minima. It consists of two loops: the inner loop returns a local optimal at each iteration; the outer loop is a multiple restart of the inner loop.

The Variable Neighbourhood Descent searches a neighbourhood to find a local optimum and every time it does not find a better solution it enlarge the neighbourhood until it does not find a better local optimum.

Both procedures prove to be efficient and effective.

Chapter 3 – Heuristic Procedures

This section presents all the heuristic procedures considered in the current study. First, there is a description of the dispatching rules and then three improvement methods will also be described: multiple sequence heuristics, NEH and, finally, local search

3.1 Dispatching rules

A dispatching rule is a rule that prioritizes all the jobs that are waiting for processing on a machine. The prioritization scheme may take into account the job's attributes and the machine's attributes, as well as the current time. Whenever a machine has been freed, a dispatching rule inspects the waiting jobs and selects the job with the highest priority.

The heuristics were chosen according to the existent literature, which considers those dispatching rules as the best performing procedures for the linear problem, as well as modified versions of these procedures, suitably adapted to the quadratic tardiness objective. All these heuristics are forward scheduling procedures, that is, the selected job at a given iteration is added to the end of the current partial sequence.

The priority indexes of the heuristics are given in Table 1.

The earliest due date (EDD) rule is one of the earliest sequencing rules, and is commonly used for scheduling problems with due dates; it schedules the jobs in non-decreasing order of their due dates d_j. Equivalently, the EDD rule select, at each iteration, the job with the largest value of the priority index $EDD_j(S) = -d_j$.

The earliest weighted due date (EWDD) rule schedules the jobs in non-decreasing order of their weighted due dates d_j/w_j. Identically, the EWDD rule selects, at each iteration, the job with the largest value of the priority index $EWDD_j(S) = w_j/d_j$.

In the modified due date (MDD) heuristic, at each iteration we select the job with the minimum value of the modified due date $max\{d_j, C_j(S)\} = max\{d_j, t + P_j(S)\} =$

$max\{d_j - t, P_j(S)\}$. This rule selects, at each iteration, the job with the largest value of the priority index $MDD_j(S)$: $MDD_j(S) = \begin{cases} \frac{1}{P_j(S)} & \text{if } s_j(S) \leq 0 \\ \frac{1}{d_j - t} & \text{otherwise} \end{cases}$.

In the weighted weighted modified due date (WMDD) heuristic, at each iteration we select the job with the minimum value of the weighted modified due date $max\{d_j, C_j(S)\}/w_j = max\{d_j, t + P_j(S)\}/w_j = max\{d_j - t, P_j(S)\}/w_j$. Equivalently, the WMDD rule selects, at each iteration, the job with the largest value of the priority index $WMDD_j(S)$:

$$WMDD_j(S) = \begin{cases} \frac{w_j}{P_j(S)} & \text{if } s_j(S) \leq 0 \\ \frac{w_j}{d_j - t} & \text{otherwise} \end{cases}.$$

The weighted shortest processing time (WSPT) rule schedules the jobs in non-increasing order of the ratio $w_j/P_j(S)$.

The minimum slack (SLK) rule chooses, at each iteration, the job with the minimum slack $s_j(S) = d_j - C_j(S)$, and the minimum slack per required time (SLK/P) selects, at each iteration, the job with the minimum value of the ratio between the slack and the total required time, that is, the job with the minimum $SLK/P_j(S) = s_j(S)/P_j(S)$.

The weighted minimum slack / shortest processing time (WSLK_SPT) rule selects, at each iteration, the job with the minimum value of the weighted slack or weighted processing time, as appropriate, that is, it selects the job with the minimum ratio $max\{s_j(S), P_j(S)\}/w_j$. Equivalently, the WSLK_SPT rule selects, at each iteration, the job with the largest value of the priority index $WSLK\_SPT_j(S)$:

$$WSLK\_SPT_j(S) = \begin{cases} \frac{w_j}{P_j(S)} & \text{if } s_j(S) \leq P_j(S) \\ \frac{w_j}{s_j(S)} & \text{otherwise} \end{cases}.$$

The apparent tardiness cost (ATC) dispatching rule chooses, at each iteration, the job with the largest value of the priority index $ATC_j(S)$:

$$ATC_j(S) = \begin{cases} \frac{w_j}{P_j(S)} & \text{if } s_j(S) \leq 0 \\ \frac{w_j}{P_j(S)} * exp\left(-\frac{s_j(S)}{k\bar{P}(S)}\right) & \text{otherwise} \end{cases}.$$

The AR dispatching rule provides the best results for the weighted objective; it chooses, at each iteration, the job with the largest value of the priority index $AR_j(S)$:

$$AR_j(S) = \begin{cases} \frac{w_j}{P_j(S)} & \text{if } s_j(S) \leq 0 \\ \frac{w_j}{P_j(S)} * \frac{k\bar{P}(S)}{k\bar{P}(S)+s_j(S)} & \text{otherwise} \end{cases}.$$

The parameter k provides the ATC and the AR heuristics with a look ahead capability and it is related with the number of competing critical jobs, that is, it takes into account the number of jobs which will become tardy in the next few iterations.

The previous heuristics are suited for the linear problem. However, several of these procedures can be modified, in order to adapt them to a quadratic setting, as it follows.

The quadratic weighted shortest processing time (QWSPT) rule schedules the jobs in non-increasing order of $\left(w_j/P_j(S)\right) * \left(\bar{P}(S) + 2T_j(S)\right)$. The QWSPT is equivalent, in a quadratic setting, to the WSPT, which means that the quadratic heuristics, normally, substitute the WSPT for the QWSPT in their priority indexes.

In the quadratic weighted modified due date (QWMDD) heuristic, at each iteration we select the job with the largest value of the priority index $QWMDD_j(S)$:

$$QWMDD_j(S) = \begin{cases} \frac{w_j}{P_j(S)} * \left(\bar{P}(S) + 2T_j(S)\right) & \text{if } s_j(S) \leq 0 \\ \frac{w_j}{d_j - t} * \bar{P}(S) & \text{otherwise} \end{cases}.$$

The quadratic quadratic weighted minimum slack / shortest processing time (QWSLK_SPT) rule selects, at each iteration, the job with the of the priority index $QWSLK\_SPT_j(S)$:

$$QWSLK\_SPT_j(S) = \begin{cases} \frac{w_j}{P_j(S)} * \left(\bar{P}(S) + 2T_j(S)\right) & \text{if } s_j(S) \leq P_j(S) \\ \frac{w_j}{s_j(S)} * \bar{P}(S) & \text{otherwise} \end{cases}.$$

The quadratic apparent tardiness cost (QATC) dispatching rule chooses, at each iteration, the job with the largest value of the priority index $QATC_j(S)$:

$$QATC_j(S) = \begin{cases} \frac{w_j}{P_j(S)} * \left(\bar{P}(S) + 2T_j(S)\right) & \text{if } s_j(S) \leq 0 \\ \frac{w_j}{P_j(S)} * \bar{P}(S) * exp\left(-\frac{s_j(S)}{k\bar{P}(S)}\right) & \text{otherwise} \end{cases}.$$

The QAR dispatching rule chooses, at each iteration, the job with the largest value of the priority index $QAR_j(S)$:

$$QAR_j(S) = \begin{cases} \frac{w_j}{P_j(S)} * \left(\bar{P}(S) + 2T_j(S)\right) & \text{if } s_j(S) \leq 0 \\ \frac{w_j}{P_j(S)} * \bar{P}(S) * \frac{k\bar{P}(S)}{k\bar{P}(S)+s_j(S)} & \text{otherwise} \end{cases}.$$

The k parameter, as previously mentioned, represents the number of critical jobs, that is, jobs that are in danger of becoming tardy. In this work, a job is considered critical if its slack is positive, but less or equal than $slk\_thr = w \times (C_{\max}^{LB}(S) - t)$, where w is a parameter specifically chosen by the user. If, at any iteration, no job is critical according to this definition, k is then set equal to 0.5, since this value proved to provide good results in previous studies.

3.2 Improvement Methods

Based on the computational results presented on Chapter 4, two heuristics were chosen to be improved: QATC and QWMDD. As previously mentioned, the improvement methods applied on both these dispatching rules were multiple sequence heuristics, NEH and local search.

Instead of finding the best sequence regarding all n jobs in all m machines, the multiple sequence heuristics generates m sequences, one for each machine, and selects the best of them. The sequence generated during the iteration related to machine i uses data that is specific to that machine.

The earliest apportioned due date (EADD) heuristic (Hasija and Rajendran 2004) obtains a sequence for each machine i by scheduling the jobs in non-decreasing order of their apportioned due dates d_{ij}. The best of those m sequences is then selected.

It calculates a due date for each job on each machine. Let d_{ij} be the apportioned due date of job j on machine i. The due dates d_{ij} are calculated as:

$$d_{1j} = \left(d_j \times p_{1j}\right) / \sum_{i=1}^{m} p_{ij}$$

and

$$d_{ij} = d_{i-1,j} + \left(d_j \times p_{ij}\right) / \sum_{k=1}^{m} p_{kj}\,, i = 2,3, \dots, m.$$

The due date of job j on machine i is then obtained by allocating the original due date according to the accumulated sum of the processing times on the various machines. That is, d_{ij} is calculated by multiplying d_j by the ratio between the sum of the processing times of job j up to and including machine i and the sum of the processing times of job j on all machines. Thus, on the final machine the apportioned due date will be equal to the original due date, that is $d_{mj} = d_j$.

The changes required in order to adapt the priority indexes of QATC and QWMDD to a multiple sequence setting are described in the next paragraphs.

Let $C_{ij}(S)$ be the completion time of job $j \notin S$, on machine i, if j is scheduled at the end of sequence S. Also, let $s_{ij}(S) = d_{ij} - C_{ij}(S)$ be the slack of job $j \notin S$, on machine i, if j is scheduled at the end of sequence S. Therefore, the slack of a certain job on a given machine is obtained by using the corresponding apportioned due date and completion time. In the multiple sequence versions, the general slack $s_j(S)$ is then replaced, in the priority index, by the machine–dependent slack $s_{ij}(S)$.

Let $C_{\max}^{LB}\_M_i(S)$ be a lower bound on the completion time of the last job on machine i (that is, a lower bound on the makespan of machine i), given the current schedule S. The machine makespan lower bound $C_{\max}^{LB}\_M_i(S)$ is calculated as previously described for the final machine lower bound $C_{max}^{LB}(S)$, with the difference that, naturally, only the processing times on the machines up to and including machine i are considered. Therefore, the lower bound is calculated as if only the first i machines existed. The slack threshold parameter in the multiple sequence procedures is calculated as before, with the difference that the machine lower bound $C_{\max}^{LB}\_M_i(S)$ replaces the final machine lower bound $C_{max}^{LB}(S)$, that is $slk\_thr = w \times (C_{\max}^{LB}\_M_i(S) - t)$.

Let $P_{ij}(S) = C_{ij}(S) - t$ be the total time (total processing time plus any eventual forced idle time) between the start of job $j \notin S$ and its finish on machine i, if j is scheduled at the end of sequence S. In this version of the multiple sequence heuristics (QATC_M and QWMDD_M), the total time between the start and finish of a job $P_j(S)$ is then replaced, in the priority index, by the total time up to and including the current machine $P_{ij}(S)$.

To summarize, the procedures QATC_M and QWMDD_M choose, at each iteration being performed while generating the sequence for machine i, the job with the largest value of the priority index $QATC\_M_{ij}(S)$ and $QWMDD\_M_{ij}(S)$, respectively. These priority indexes are equal to

$$QATC\_M_{ij}(S) = \begin{cases} \dfrac{w_j}{P_{ij}(S)} * \left(\bar{P}(S) + 2T_{ij}(S)\right) & \text{if } s_{ij}(S) \leq 0 \\ \dfrac{w_j}{P_{ij}(S)} * \bar{P}(S) * exp\left(-\dfrac{s_{ij}(S)}{k\bar{P}(S)}\right) & \text{otherwise} \end{cases}$$

and

$$QWMDD\_M_{ij}(S) = \begin{cases} \frac{w_j}{P_{ij}(S)} * \left(\bar{P}(S) + 2T_{ij}(S)\right) & \text{if } s_{ij}(S) \leq 0 \\ \frac{w_j}{d_{ij}-t} * \bar{P}(S) & \text{otherwise} \end{cases},$$

We remark that this version will return a sequence that is at least as good as the one generated by the corresponding single sequence heuristic. This is due to the fact that the solution obtained for the last machine is the same as the one generated by the single sequence procedure. Indeed, and for the last machine m, we have $s_{mj}(S) = s_j(S)$, $C^{LB}_{\max}\_M_m(S) = C^{LB}_{\max}(S)$ and $P_{ij}(S) = P_j(S)$.

The second improvement method applied was the well-known NEH procedure developed in (Nawaz, Enscore Jr et al. 1983); this method is an insertion procedure which requires an initial sequence of jobs (in this case, the solution provided by the initial heuristic improved by multiple sequence method) to create another sequence, hopefully better than the original one.

During the insertion phase, the jobs are considered in the order in which they appear in the initial sequence or list. At each step, the currently considered job is tentatively inserted in each possible position of the currently partial sequence. The job is then inserted in the position which provides the best objective function value.

In our implementation, the sequence resulting from the NEH procedure is kept if it is not worse than the initial sequence. Otherwise, the (better) initial sequence is retained.

Since it can happen that some jobs finish early, and the sequences representing this cases have a cost equal to zero, it was important to choose a tie-breaking method. (Fernandez-Viagas and Framinan 2015) presented several tie-breaking methods for an unweighted tardiness setting which can also be applied in an weighted quadratic tardiness scenario. The procedure chosen for this study is called Total Idle Time (IT1) since it was the one with the better performance, improving the NEH method in more than 25% while requiring similar computational time. This tie-breaking method is

extensively presented in (Fernandez-Viagas and Framinan 2015), but it can be briefly explain as follows: when there are two, or more, equal objective function values, it calculates the total idle time as a sum of the idle time for each machine i and chooses the minimum IT1; in this method the definition of idle time includes front delays (the idle time before the first job starts on a machine) and excludes back delays (the time between the finish time on a machine and the overall finish time).

Finally, the last improvement method applied was a local search procedure that included both interchange and the insertion neighborhoods, with a first-improve strategy. First, all possible interchanges between pairs of jobs are first considered; an improving exchange is performed whenever it is detected, and this is repeated until no improving interchange is found. Then, all possible insertions (removing one job from its current position and inserting it in another position) are considered; again, an improving move is immediately performed, and this is repeated until no insertion can lead to a better objective function value. The process of performing interchanges followed by insertions is repeated until no further improvement is made.

Chapter 4 – Computational Results

In this section, the computational experiments and results are presented. First, the set of test problems used to obtain the computational results is described and the preliminary tests that were performed in order to determine adequate values for the parameters required by some of the heuristics are presented. A comparison of the dispatching rules is then performed. Finally, the results of the best heuristics are compared with the results after the improvement methods are applied.

4.1 Experimental design and parameter adjustment tests

The computational tests were performed on a set of randomly generated problems, with various sizes in terms of both the number of jobs and the number of machines, and for multiple combinations of due date tightness and range. More specifically, the problems were generated as follows.

In what regards the number of jobs, the following sizes were considered: 8, 10, 12, 15, 17, 20, 25, 30, 40, 50, 75, 100, 200, 300, 400 and 500. For the machines, we considered problems with 5, 10 and 20 machines. For each job j, the processing times on the various machines p_{ij} were generated from a uniform distribution over the integers 1 to 100, while an integer weight w_j was obtained from a uniform distribution [1, 10].

Finally, for each job j, an integer due date d_j was generated from the uniform distribution $[MS(1-T-R/2), MS(1-T+R/2)]$, where MS is an estimate of the makespan calculated using the lower bound proposed in (Taillard 1993), T is the tardiness factor and R is the range of due dates. Both the tardiness factor and the range of due dates parameters were set at 0.2, 0.4, 0.6, 0.8 and 1.0.

For each combination of n, M, T and R, 50 instances were randomly generated. Therefore, a total of 1250 instances were generated for each problem size, where the size is given by both the number of jobs and the number of machines.

The procedures were coded in C++, compiled for 64–bit Windows, and executed on a personal computer with a Windows 7 64–bit operating system, an Intel Core i7 4770 3.4G processor and 16GB RAM.

The ATC, AR, QATC and QAR dispatching rules require a value for the parameter w, $0 \leq w \leq 1$. Extensive preliminary tests were performed in order to determine an adequate value of w for the single sequence heuristics. These tests were performed on a separate problem set that included instances with 15, 25, 50, 75, 100, 200, 300, 400 and 500 jobs, and contained 5 instances for each combination of n, M, T and R.

The values {0.00, 0.05, 0.10, 0.15, 0.20, …, 0.90, 0.95, 1.00} were considered for the parameter w. The ATC, AR, QATC and QAR dispatching rules were then applied to the instances on the smaller test set, and the objective function value was calculated for each considered value. These results were then analysed, and we selected a value that provided good performance across all instance types. The value of w was then set at 0.0, since it improved the objective function values between 17.88% and 27.27% among the other values analyzed. The same value was used for the improvement methods.

4.2 Comparisons of the heuristic procedures

A comparison of the dispatching rules that specifically consider the quadratic objective with their linear tardiness counterparts is provided in Table 2. For each pair of quadratic heuristic and its linear counterpart, this table gives the mean relative improvement versus the worst result (%ivw) of the quadratic (%ivw_q) and the linear (%ivw_l) procedures, as well as the number of times each quadratic rule provides a solution that is better (btr), equal (eql) or worse (wrs) than the one provided by the corresponding linear procedure. The global line provides the mean for the variables %ivw_q and %ivw_l and the sum for btr, eql and wrs of the appropriate performance measure over all the instances for all problem sizes.

The particular nature of the squared weighted tardiness problem motivated the use of the relative improvement versus the worst result performance measure, instead of the more usual relative improvement a procedure provides over another heuristic. Indeed, and particularly for instances with a low tardiness factor *T* and a high range of due dates *R*, the objective function value given by the heuristic procedures can be equal to 0, meaning that all jobs are completed on time. This is troublesome when the relative improvement is used, since division by 0 is undefined, and motivated the use of a different performance measure.

More specifically, and for a given instance, the relative improvement versus the worst result of heuristic H_i, when compared with heuristics H_1, H_2, … , H_z, is calculated as follows. Let ofv_{best} and ofv_{worst} be the best and worst objective function values obtained by all the z heuristic procedures, respectively. When $ofv_{best} = ofv_{worst}$, the relative improvement versus the worst result of heuristic H_i is set at 0. Otherwise, the relative improvement versus the worst result is calculated as $\left(ofv_{worst} - ofv_{H_i}\right)/ofv_{worst} * 100$, where ofv_{H_i} is the objective function value of heuristic H_i. This performance indicator thereby measures the relative improvement a given heuristic provides over the worst result obtained among all procedures being compared, and circumvents the division by 0 issue.

Since the purpose of Table 2 is to analyse the performance of each quadratic procedure versus the corresponding linear heuristic, the %ivw_q and %ivw_l values given in this

table were calculated separately for each pair of quadratic heuristic and its linear counterpart. More particularly, and using the QAR and AR procedures as an example, these values were calculated as follows. The %ivw_q is the relative improvement QAR provides over the worst result among the QAR and AR heuristics. Similarly, %ivw_l is the relative improvement given by AR versus the worst result between QAR and AR. The same reasoning applies to the other pairs of quadratic heuristic and its linear counterpart.

The results presented in Table 2 show that the heuristics that have been suitably adapted to the quadratic objective outperform their linear tardiness counterparts, which is to be expected. This table shows that the difference in performance is quite significant; indeed, the quadratic dispatching rules not only provide a much larger relative improvement versus the worst result, but they also obtain better (or equal) results for most, or in some cases actually all, of the test instances. Note that, although this table just illustrates some instances, the global line considers the values of all instances in each situation; the same happens with the other tables.

Therefore, it is most certainly recommended to use heuristics that have been designed in order to take into account the quadratic tardiness objective, instead of simply relying on procedures originally developed for the linear problem. Thus, in the remainder of this section, the linear rules analyzed in Table 2 will no longer be considered.

Similarly, Table 3 provides a comparison between the quadratic dispatching rules and the rules with no quadratic version available, using, once again, the mean relative improvement versus the worst result (%ivw). The avg line provides the mean of the %ivw of the appropriate performance measure over all the instances for all problem sizes.

On average, Table 3 shows that the best performing heuristic is QATC, which provides better results for all number of machines consistently over all the other rules, followed by the QAR and the QWMDD heuristics. Since the results are very close between these three rules, this study focuses on two of them: QATC, for obvious reasons, and QWMDD, due to its simplicity and efficiency when compared to both QATC and QAR.

In order to identify how many times the QWMDD and the QATC heuristics provide a solution that is better (btr), equal (eql) or worst (wrs) than the one provided by other heuristic, Table 4 and Table 5 are presented, respectively. The sum line provides the total of cases in which each heuristic is better, equal or worse than the rule in analysis, over all the instances for all problem sizes.

These two tables illustrate, in accordance with Table 3, that both rules outperform the others, and even though the QATC performs better than the QWMDD in more instances, they both perform equally for more than 50% of the tested instances.

All the heuristics proved to be efficient and can be applied in an extensive dataset; even though the QATC, QAR, QWMDD and QWPT_WSLK_SPT rules require more computational effort, they still allow results in less than 0.05 seconds for large instances.

4.3 Comparison with improvement methods

A comparison of the three improvement methods applied in the two best performing heuristics, the QATC and the QWMDD rules, is provided in Table 6. For both dispatching rules, this table gives the number of times each improvement method results in a better (btr) solution than the original one, as well as the mean relative improvement (imp_%) and the percentage in which it contributed to the total improvement (%_tot_imp). The global line provides the sum for the variable btr and the mean for all the others.

Due to time restrictions, it was only possible to run the improvement procedures on instances with up to $n = 300$.

All three improvement methods gave good results, improving the inicial heuristic in more than 40%, being the NEH the one that contributed the most for the total improvement in both rules. The mean relative improvement of the multiple sequence heuristic decreases when the number of jobs increase, and the opposite happens with NEH and LS.

The effect of the T and R parameters on the performance of these improvement methods is illustrated in Table 7 and Table 8 for the QATC and the QWMDD rules, respectively, for $n = 300$ and $M = 10$.

Even though, in both cases, the multiple sequence heuristic has an inconsistent performance, the NEH and the local search procedures have an excellent performance when $T \geq 0.4$, and a near optimal result when $T = 0.2$ which decreases when the range is high, especially for the QATC rule. The multiple sequence heuristic is affected by the range: its performance is better when the range is low and decreases as the range increases; this effect is common to both rules.

In order to compare both rules after the three methods were applied, Table 9 is presented. Once again, the %ivw was calculated for each heuristic, as well as the sum of instances where one rule was better, equal or worse than the other.

Overall, Table 9 shows that the dispatching rule with the best improvement rates is the QATC_M_NEH_LS, but they both have similar results and, quite often, the same

improvement. Both heuristics perform better with a high number of jobs, in particular the QATC_M_NEH_LS when $M = 5$ or $M = 20$.

The effect of the T and R parameters on the performance of the two improved heuristics is illustrated in Table 10, for $n = 300$ and $M = 10$. Table 10 shows that when $T \geq 0.8$ these heuristics have a quite similar performance, in accordance with Table 7; the same happens for high levels of the range parameter. Thus, the two improved heuristics basically provide the same level of performance, in what regards solution quality.

The runtimes presented in Table 11 prove that the QATC_M_NEH_LS and the QWMDD_M_NEH_LS require more computational effort than the first heuristics mentioned in this study, especially for large instances, but the time required is still reasonable. In particular, the multiple sequence and NEH improvements require very little additional computational time; indeed, it is the local search phase that is responsible for most of the extra runtime. Therefore, applying only the first two improvements is certainly a possibility for quite large instances, on which the local search procedure may require impractical times.

Finally, the runtimes are similar for both the QATC_M_NEH_LS and the QWMDD_M_NEH_LS procedures, being the second heuristic faster when only the first two improvement methods are applied. Given that they also provided similar results in terms of solution quality, either of these heuristics can be selected, since they are quite close in terms of both effectiveness and efficiency.

Chapter 5 - Conclusions

This investigation focused on heuristics for a permutation flowshop scheduling problem with weighted quadratic tardiness, due its relevance in many industries when trying to prevent customers' dissatisfaction as well as financial inconvenients.

On a set of randomly generated problems, with various sizes of number of jobs and machines, as well as multiple combinations of due date tightness and range, we compared some of the most well-known performing heuristics in the literature and applied improvement methods in two of the best performing rules among the ones that were analysed.

For each pair of quadratic heuristic and its linear counterpart, the calculation of the mean relative improvement versus the worst result showed that the quadratic versions consistently outperform their linear counterpart; indeed, the heuristics suitably adapted to the quadratic objective provide better results in most of the test instances. Therefore, it is recommended to use heuristics that have been designed to take into account the quadratic tardiness objective.

A comparison between the quadratic dispatching rules and the rules with no quadratic version available showed that the QATC, QAR and QWMDD are the best performing rules when compared with more simple heuristics; these three rules are around 50% better than the others and frequently have equal results between them.

The improvement methods applied, namely, the multiple sequence heuristic, the NEH and the local search procedures, improved the QATC and the QWMDD by more than 40%, with NEH contributing the most and the local search the least, despite its larger computational effort. However, and naturally, the local search could possibly have provided a larger improvement if either or both of the two previous improvement methods had not been applied. Both improved heuristics, QATC_M_NEH_LS and QWMDD_M_NEH_LS, have similar results regarding solution quality. Also, their runtimes are similar, so it is basically indifferent to use one rule or the other, since both are effective and efficient.

Regarding future research, one possibility consists in the application of metaheuristics to the considered problem. Indeed, metaheuristics usually outperform dispatching rules, and can usually be applied, within adequate runtimes, to medium size instances. The dispatching rules and improvement procedures given in this work can provide an initial solution, as well as improvements steps, to these metaheuristics. Another alternative is to add to the considered problem some features that are present in certain real life situations, such as different release dates or setups. Finally, the quadratic tardiness objective function could be applied to other production settings, such as open shops or job shops.

Appendix

Tables

Table 1 – Dispatching rules

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

Table 2 – Comparison between quadratic heuristics and their linear counterpart

		QAR vs AR					QATC vs ATC					QWMDD vs WMDD					QWSLK_SPT vs WSLK_SPT					QWSPT vs WSPT				
m	n	%ivw_q	%ivw_l	btr	eql	wrs	%ivw_q	%ivw_l	btr	eql	wrs	%ivw_q	%ivw_l	btr	eql	wrs	%ivw_q	%ivw_l	btr	eql	wrs	%ivw_q	%ivw_l	btr	eql	wrs
5	10	11.73	3.99	770	45	435	5.04	91.78	71	0	1179	13.00	3.82	785	38	427	14.04	3.49	805	46	399	13.99	3.50	801	47	402
	25	20.14	1.57	1000	0	250	99.84	0.00	1248	2	0	21.69	1.38	1017	0	233	23.15	1.45	1034	0	216	23.78	1.35	1035	0	215
	50	25.85	0.70	1087	0	163	99.60	0.00	1245	5	0	28.38	0.52	1123	0	127	29.35	0.61	1128	0	122	30.55	0.55	1130	0	120
	75	29.67	0.36	1154	0	96	97.76	0.00	1222	28	0	32.05	0.30	1173	0	77	33.05	0.29	1166	0	84	34.68	0.28	1175	0	75
	100	31.53	0.29	1154	0	96	96.64	0.00	1208	42	0	34.20	0.20	1179	0	71	34.95	0.27	1183	0	67	36.89	0.26	1178	0	72
	300	33.71	0.20	1144	28	78	91.76	0.00	1147	103	0	39.78	0.11	1213	0	37	40.90	0.07	1222	0	28	44.34	0.04	1225	0	25
	500	31.33	0.20	1099	73	78	89.12	0.00	1114	136	0	40.68	0.12	1197	2	51	42.05	0.05	1222	2	26	46.51	0.01	1241	0	9
10	10	8.53	3.78	718	35	497	25.27	72.67	327	0	923	9.10	3.75	726	32	492	10.03	3.51	732	37	481	10.09	3.54	730	35	485
	25	14.52	1.41	958	0	292	100.00	0.00	1250	0	0	15.82	1.43	964	0	286	17.20	1.45	976	0	274	17.37	1.39	986	0	264
	50	20.40	0.67	1086	0	164	100.00	0.00	1250	0	0	21.88	0.61	1100	0	150	23.42	0.64	1101	0	149	23.60	0.60	1095	0	155
	75	23.05	0.46	1125	0	125	100.00	0.00	1250	0	0	25.28	0.38	1142	0	108	26.62	0.37	1143	0	107	27.15	0.37	1143	0	107
	100	26.20	0.34	1137	0	113	99.84	0.00	1248	2	0	28.19	0.27	1155	0	95	29.35	0.29	1159	0	91	30.35	0.31	1152	0	98
	300	33.58	0.10	1199	0	51	95.76	0.00	1197	53	0	36.28	0.05	1222	0	28	37.02	0.04	1226	0	24	39.42	0.06	1216	0	34
	500	35.94	0.07	1206	2	42	93.20	0.00	1165	85	0	39.30	0.03	1232	0	18	40.11	0.00	1241	0	9	42.79	0.01	1241	0	9
20	10	5.92	3.12	657	42	551	100.00	0.00	1250	0	0	6.17	3.12	659	41	550	6.57	3.10	673	37	540	6.57	3.10	673	37	540
	25	9.94	1.54	892	0	358	100.00	0.00	1250	0	0	10.62	1.55	910	0	340	11.34	1.57	913	0	337	11.41	1.57	909	0	341
	50	14.60	0.77	1034	0	216	100.00	0.00	1250	0	0	15.53	0.77	1029	0	221	16.97	0.77	1029	0	221	17.01	0.77	1025	0	225
	75	17.60	0.39	1106	0	144	100.00	0.00	1250	0	0	19.03	0.35	1127	0	123	20.23	0.39	1113	0	137	20.31	0.42	1109	0	141
	100	19.93	0.38	1119	0	131	100.00	0.00	1250	0	0	21.74	0.33	1128	0	122	22.84	0.36	1129	0	121	22.96	0.39	1119	0	131
	300	28.46	0.09	1189	0	61	99.92	0.00	1249	1	0	31.01	0.04	1214	0	36	31.96	0.04	1218	0	32	33.26	0.03	1219	0	31
	500	32.42	0.03	1225	0	25	98.56	0.00	1232	18	0	34.84	0.02	1226	0	24	35.78	0.02	1233	0	17	37.69	0.02	1232	0	18
global		19.31	1.44	15727	230	4043	89.62	7.29	18039	265	1696	21.28	1.39	16026	158	3816	22.31	1.33	16181	157	3662	23.15	1.32	16176	156	3668

Table 3 – Comparison between quadratic heuristics and heuristics with no quadratic versions

		%ivw									
m	n	EDD	EWDD	MDD	SLK	SLKP	QAR	QATC	QWMDD	QWSLK_SPT	QWSPT
5	10	25.79	34.26	23.92	16.57	22.56	44.41	44.80	43.09	40.72	40.41
	25	29.76	32.38	19.82	24.71	30.30	50.42	51.53	48.51	43.06	40.55
	50	34.98	30.85	20.07	32.57	36.55	56.58	58.54	54.64	49.34	43.23
	75	38.71	30.84	21.49	37.36	40.79	60.50	62.32	59.04	54.43	46.72
	100	40.83	31.84	21.69	39.65	42.29	63.13	64.52	61.91	57.86	49.57
	300	49.28	33.95	24.93	48.81	50.44	69.80	70.46	69.87	68.08	58.78
	500	51.71	34.83	25.63	51.40	52.57	71.54	72.18	71.80	70.59	61.48
10	10	19.26	30.27	18.34	11.31	16.01	36.53	36.61	36.12	35.33	35.25
	25	23.23	33.48	11.72	19.67	22.78	44.55	44.95	43.33	40.36	39.50
	50	30.33	34.42	11.68	28.00	30.04	51.10	52.41	49.46	45.05	43.62
	75	34.73	34.60	13.15	33.43	35.30	54.68	56.24	52.87	47.60	45.43
	100	38.39	35.66	14.06	37.14	38.76	58.60	59.78	56.58	51.30	48.42
	300	48.43	36.81	19.86	48.04	48.93	68.06	68.99	66.77	62.95	57.40
	500	51.86	37.96	22.14	51.61	52.37	71.59	71.98	70.68	68.01	61.80
20	10	13.83	24.66	14.31	8.20	11.63	29.46	29.47	29.26	28.95	28.95
	25	17.55	30.66	6.44	14.26	16.73	38.05	38.15	37.71	36.47	36.46
	50	24.56	35.30	3.87	22.83	24.01	45.81	46.19	44.82	42.52	42.38
	75	29.44	36.43	6.22	28.12	29.14	49.84	50.53	48.50	44.89	44.51
	100	33.33	37.95	7.19	32.33	33.37	53.16	53.94	51.76	47.75	46.82
	300	44.61	39.53	12.72	44.10	44.73	63.48	64.88	61.73	57.09	54.47
	500	48.96	39.45	15.71	48.63	49.15	67.75	68.71	65.99	61.84	57.97
avg		30.74	33.14	15.93	27.27	30.23	50.46	51.18	49.43	46.39	43.85

Table 4 – Comparison between the QWMDD rule and all the other rules

		QWMDD vs																									
		EDD			EWDD			MDD			SLK			SLKP			QATC			QAR			QWSLK_SPT			QWSPT		
m	n	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs
5	10	998	2	250	824	148	278	978	0	272	1087	0	163	1056	0	194	125	878	247	81	959	210	285	790	175	312	736	202
	25	1021	0	229	1055	2	193	1068	0	182	1055	0	195	1035	0	215	211	646	393	208	654	388	530	561	159	539	527	184
	50	1014	0	236	1142	0	108	1036	0	214	1024	0	226	1017	0	233	202	555	493	213	566	471	567	505	178	629	483	138
	75	996	0	254	1179	0	71	1022	0	228	990	0	260	989	0	261	206	533	511	236	545	469	594	503	153	677	479	94
	100	991	0	259	1193	0	57	1018	0	232	996	0	254	993	0	257	210	536	504	225	538	487	588	517	145	669	489	92
	300	979	0	271	1229	0	21	997	0	253	980	0	270	978	0	272	261	534	455	273	534	443	562	522	166	729	483	38
	500	973	1	276	1231	0	19	994	1	255	975	1	274	972	1	277	252	540	458	289	540	421	547	527	176	733	470	47
10	10	1063	0	187	746	155	349	1008	0	242	1132	0	118	1092	0	158	88	1040	122	57	1095	98	174	964	112	180	959	111
	25	1104	0	146	975	2	273	1149	0	101	1118	0	132	1103	0	147	162	774	314	161	794	295	386	688	176	392	671	187
	50	1078	0	172	1087	0	163	1121	0	129	1083	0	167	1085	0	165	179	659	412	197	665	388	477	610	163	517	585	148
	75	1028	0	222	1144	0	106	1088	0	162	1034	0	216	1035	0	215	177	610	463	207	615	428	541	547	162	568	537	145
	100	1021	0	229	1160	0	90	1082	0	168	1024	0	226	1029	0	221	231	559	460	224	577	449	557	525	168	589	522	139
	300	983	0	267	1229	0	21	1030	0	220	982	0	268	982	0	268	195	548	507	201	548	501	582	542	126	654	533	63
	500	983	0	267	1236	0	14	1020	0	230	984	0	266	986	0	264	203	547	500	177	547	526	592	541	117	687	535	28
20	10	1090	0	160	706	168	376	996	0	254	1162	0	88	1112	0	138	27	1177	46	24	1187	39	62	1153	35	62	1153	35
	25	1176	0	74	938	3	309	1219	0	31	1189	0	61	1182	0	68	118	951	181	108	960	182	266	853	131	266	851	133
	50	1135	0	115	1039	0	211	1208	0	42	1146	0	104	1140	0	110	160	769	321	168	779	303	397	703	150	389	699	162
	75	1104	0	146	1083	0	167	1168	0	82	1106	0	144	1108	0	142	191	680	379	193	687	370	457	646	147	464	642	144
	100	1097	0	153	1111	0	139	1158	0	92	1105	0	145	1099	0	151	171	651	428	183	653	414	485	636	129	510	628	112
	300	1015	0	235	1207	0	43	1094	0	156	1018	0	232	1016	0	234	168	548	534	216	548	486	570	546	134	613	546	91
	500	984	0	266	1227	0	23	1076	0	174	988	0	262	990	0	260	169	550	531	167	550	533	606	550	94	650	550	50
sum		50425	6	9569	49022	1867	9111	51597	2	8401	51740	3	8257	51002	4	8994	7605	36063	16332	7609	36919	15472	19923	33434	6643	21544	32595	5861

Table 5 – Comparison between the QATC rule and all the other rules

		QATC vs																										
		EDD			EWDD			MDD			SLK			SLKP			QAR			QWMDD			QWSLK_SPT			QWSPT		
m	n	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs
5	10	1027	2	221	883	60	307	999	0	251	1110	0	140	1083	0	167	157	978	115	247	878	125	345	716	189	361	686	203
	25	1053	2	195	1051	0	199	1095	2	153	1089	2	159	1081	2	167	335	700	215	393	646	211	534	556	160	552	526	172
	50	1060	5	185	1124	0	126	1080	5	165	1070	5	175	1063	5	182	423	590	237	493	555	202	580	505	165	645	483	122
	75	1034	28	188	1179	0	71	1062	28	160	1037	28	185	1030	28	192	457	555	238	511	533	206	605	503	142	684	479	87
	100	1037	42	171	1178	0	72	1061	42	147	1041	42	167	1037	42	171	462	548	240	504	536	210	584	516	150	657	489	104
	300	1009	103	138	1201	0	49	1026	103	121	1009	103	138	1006	103	141	449	565	236	455	534	261	514	522	214	710	483	57
	500	1010	136	104	1205	0	45	1026	136	88	1009	136	105	999	136	115	409	613	228	458	540	252	499	527	224	710	470	70
10	10	1072	0	178	785	95	370	1021	0	229	1144	0	106	1101	0	149	55	1139	56	122	1040	88	202	909	139	207	907	136
	25	1129	0	121	983	0	267	1170	0	80	1145	0	105	1131	0	119	226	869	155	314	774	162	414	676	160	405	663	182
	50	1104	0	146	1079	0	171	1143	0	107	1101	0	149	1107	0	143	349	700	201	412	659	179	481	609	160	520	584	146
	75	1067	0	183	1142	0	108	1112	0	138	1066	0	184	1073	0	177	389	642	219	463	610	177	563	546	141	579	537	134
	100	1055	2	193	1144	0	106	1105	2	143	1057	2	191	1050	2	198	396	602	252	460	559	231	560	525	165	589	522	139
	300	1021	53	176	1221	0	29	1059	53	138	1015	53	182	1017	53	180	469	549	232	507	548	195	571	542	137	658	533	59
	500	1016	85	149	1226	0	24	1049	85	116	1023	85	142	1016	85	149	448	549	253	500	547	203	554	541	155	670	535	45
20	10	1096	0	154	727	140	383	997	0	253	1166	0	84	1119	0	131	9	1237	4	46	1177	27	84	1117	49	84	1117	49
	25	1178	0	72	942	1	307	1227	0	23	1199	0	51	1190	0	60	104	1059	87	181	951	118	277	847	126	278	845	127
	50	1153	0	97	1045	0	205	1221	0	29	1164	0	86	1164	0	86	241	846	163	321	769	160	416	700	134	411	696	143
	75	1125	0	125	1082	0	168	1189	0	61	1123	0	127	1126	0	124	305	735	210	379	680	191	468	646	136	468	642	140
	100	1116	0	134	1119	0	131	1172	0	78	1118	0	132	1114	0	136	341	672	237	428	651	171	507	636	107	526	628	96
	300	1054	1	195	1208	0	42	1111	1	138	1059	1	190	1059	1	190	484	553	213	534	548	168	608	546	96	631	546	73
	500	1036	18	196	1225	0	25	1098	18	134	1042	18	190	1038	18	194	473	550	227	531	550	169	610	550	90	657	550	43
sum		51682	798	7520	49145	1170	9685	52582	794	6624	52964	797	6239	52254	797	6949	13175	39059	7766	16332	36063	7605	20441	32776	6783	22192	32066	5742

Table 6 – Comparison between each improvement method for both the QATC rule and the QWMDD rule

		QATC										QWMDD									
		M			NEH			LS			tot_imp_%	M			NEH			LS			tot_imp_%
m	n	btr	imp_%	%_tot_imp	btr	imp_%	%_tot_imp	btr	imp_%	%_tot_imp		btr	imp_%	%_tot_imp	btr	imp_%	%_tot_imp	btr	imp_%	%_tot_imp	
5	10	1031	17.90	42.95	1153	20.89	44.33	913	8.61	12.48	38.46	1042	18.52	43.39	1154	21.77	44.17	914	8.91	12.21	39.18
	15	992	15.68	33.00	1226	29.02	53.35	1140	13.86	13.57	44.85	1020	16.42	33.64	1230	29.90	52.99	1149	15.09	13.37	45.69
	25	909	13.34	24.79	1248	37.73	60.97	1223	22.27	14.08	51.41	961	14.23	25.87	1249	38.43	60.21	1234	23.71	13.92	52.14
	30	882	12.23	22.42	1247	40.61	64.72	1209	21.73	12.70	53.17	954	13.52	23.51	1249	40.98	63.59	1231	24.19	12.90	53.73
	50	727	7.90	14.97	1244	44.32	72.09	1180	23.08	12.54	54.69	785	9.77	16.76	1250	44.57	70.20	1237	28.73	13.04	55.45
	100	535	3.79	7.92	1207	44.35	77.98	1143	21.23	10.74	51.65	595	5.09	9.20	1250	47.56	79.53	1217	28.46	11.26	55.40
	300	209	0.62	1.87	1147	39.18	81.06	1095	16.46	8.83	43.30	227	1.23	2.47	1250	46.92	88.56	1160	22.42	8.96	51.32
10	10	1076	14.42	44.29	1149	14.78	43.65	887	4.44	11.90	29.67	1107	14.70	44.96	1145	14.90	43.13	892	4.66	11.75	29.97
	15	1126	14.27	37.48	1215	21.32	48.72	1126	8.64	13.79	36.80	1151	14.84	38.15	1216	21.37	47.92	1128	9.15	13.93	37.19
	25	1086	13.13	28.46	1249	30.95	56.69	1239	16.54	14.85	45.82	1121	13.64	28.84	1249	31.15	56.10	1240	17.21	15.06	46.16
	30	1069	12.37	25.89	1248	34.42	59.68	1245	19.85	14.44	48.64	1101	12.75	26.20	1248	34.47	59.27	1246	20.51	14.53	48.79
	50	973	9.37	18.26	1250	40.91	67.49	1247	25.19	14.26	52.97	1035	10.25	19.06	1250	41.09	66.61	1250	26.07	14.33	53.34
	100	807	6.74	12.13	1248	46.15	74.41	1228	28.67	13.29	55.92	860	7.08	12.29	1250	46.24	74.01	1250	31.27	13.70	56.30
	300	555	2.55	5.40	1197	44.32	79.64	1141	23.20	10.73	50.66	639	4.39	7.26	1250	48.16	81.43	1238	31.89	11.31	55.12
20	10	1146	10.31	43.92	1158	10.65	45.44	852	2.47	10.56	21.53	1145	10.58	44.32	1159	10.63	45.17	851	2.48	10.43	21.73
	15	1172	11.21	39.40	1213	14.38	48.06	1103	4.58	12.54	27.03	1178	11.65	39.98	1210	14.46	47.68	1103	4.65	12.34	27.40
	25	1167	10.54	30.22	1249	21.95	55.34	1238	8.20	14.44	34.64	1185	10.71	30.39	1249	21.99	55.03	1240	8.69	14.57	34.92
	30	1176	10.12	27.40	1248	24.47	57.92	1242	10.00	14.68	37.13	1178	10.27	27.58	1248	24.74	57.86	1242	10.18	14.55	37.36
	50	1104	8.47	20.03	1250	32.57	64.47	1250	16.40	15.49	44.55	1138	9.09	20.72	1250	32.54	63.72	1250	17.09	15.56	44.82
	100	958	6.34	13.06	1250	40.57	71.93	1250	24.23	15.01	50.85	1029	6.76	13.41	1250	40.43	71.21	1250	24.84	15.38	51.11
	300	761	3.42	6.83	1249	46.70	79.90	1220	27.86	13.19	54.33	849	3.97	7.39	1250	46.50	78.93	1250	31.01	13.67	54.63
global		39571	10.18	25.67	51004	30.83	60.63	47218	15.58	12.95	42.94	41221	10.85	26.34	51380	31.64	60.57	47942	17.45	13.04	43.94

Table 7 – Comparison between the improvement methods for $n = 300$ and $M = 10$, for the QATC rule

		M			NEH			LS			tot_imp_%
T	R	btr	imp_%	%_tot_imp	btr	imp_%	%_tot_imp	btr	imp_%	%_tot_imp	
0.2	0.2	41	15.50	17.60	50	69.32	68.50	50	46.19	13.90	86.42
	0.4	10	2.42	2.43	50	93.74	91.85	50	93.87	5.72	99.61
	0.6	8	2.78	2.78	50	99.66	96.91	41	82.00	0.31	100.00
	0.8	0	0.00	0.00	40	80.00	80.00	0	0.00	0.00	80.00
	1	0	0.00	0.00	7	14.00	14.00	0	0.00	0.00	14.00
0.4	0.2	39	8.59	14.53	50	43.39	71.03	50	15.89	14.44	56.65
	0.4	25	3.98	5.66	50	56.57	81.38	50	20.94	12.96	67.01
	0.6	3	0.19	0.23	50	71.08	86.81	50	37.05	12.96	81.70
	0.8	1	0.02	0.02	50	88.22	89.83	50	86.27	10.15	98.16
	1	0	0.00	0.00	50	96.40	96.41	50	99.90	3.59	99.99
0.6	0.2	49	7.68	18.23	50	30.65	69.57	50	7.80	12.20	41.06
	0.4	32	3.94	8.33	50	36.95	79.57	50	8.90	12.10	44.84
	0.6	18	1.36	2.64	50	40.95	81.93	50	12.96	15.43	49.31
	0.8	4	0.46	0.84	50	40.77	80.06	50	16.24	19.10	50.62
	1	1	0.04	0.08	50	38.96	81.76	50	14.09	18.16	47.55
0.8	0.2	49	5.76	18.58	50	22.24	70.62	50	4.35	10.80	29.92
	0.4	41	2.66	8.65	50	24.68	79.48	50	4.86	11.87	30.26
	0.6	27	0.93	3.04	50	24.87	84.49	50	4.85	12.47	29.17
	0.8	23	1.02	3.29	50	24.46	82.84	50	5.39	13.87	29.26
	1	13	0.31	1.05	50	22.36	84.09	50	5.01	14.86	26.47
1	0.2	44	2.08	8.89	50	18.29	81.36	50	2.68	9.76	22.14
	0.4	40	1.80	7.63	50	18.90	82.81	50	2.66	9.56	22.48
	0.6	31	0.95	4.47	50	17.77	84.68	50	2.74	10.85	20.79
	0.8	27	0.66	3.17	50	17.44	86.20	50	2.56	10.63	20.09
	1	29	0.55	2.74	50	16.27	84.75	50	2.83	12.50	19.08
global		555	2.55	5.40	1197	44.32	79.64	1141	23.20	10.73	50.66

Table 8 – Comparison between the improvement methods for $n = 300$ and $M = 10$, for the QWMDD rule

		M			NEH			LS			tot_imp_%
T	R	btr	imp_%	%_tot_imp	btr	imp_%	%_tot_imp	btr	imp_%	%_tot_imp	
0.2	0.2	40	17.26	19.40	50	69.36	66.38	50	48.86	14.22	87.14
	0.4	19	4.46	4.47	50	91.73	87.91	50	97.04	7.62	99.76
	0.6	18	5.47	5.47	50	98.76	93.35	50	100.00	1.18	100.00
	0.8	21	11.89	11.89	50	99.72	87.85	49	98.00	0.26	100.00
	1	34	23.07	23.07	50	99.92	76.87	39	78.00	0.06	100.00
0.4	0.2	46	11.10	18.54	50	43.29	66.44	50	17.24	15.02	58.30
	0.4	27	5.02	7.03	50	55.67	77.69	50	24.90	15.28	68.27
	0.6	12	1.88	2.16	50	69.96	82.12	50	45.10	15.73	83.59
	0.8	1	0.01	0.01	50	86.20	87.62	50	89.61	12.37	98.34
	1	1	0.03	0.03	50	94.10	94.08	50	99.95	5.89	100.00
0.6	0.2	47	7.66	18.19	50	31.05	69.73	50	7.78	12.08	41.35
	0.4	32	3.78	8.12	50	36.35	79.21	50	9.08	12.67	44.34
	0.6	13	1.53	2.91	50	40.31	80.61	50	13.65	16.48	49.26
	0.8	4	0.46	0.84	50	40.77	80.06	50	16.24	19.10	50.62
	1	1	0.04	0.08	50	38.98	81.79	50	14.08	18.14	47.56
0.8	0.2	48	5.06	16.48	50	22.85	73.33	50	4.13	10.19	29.78
	0.4	41	2.66	8.62	50	24.68	79.39	50	4.93	11.98	30.30
	0.6	27	0.93	3.04	50	24.87	84.49	50	4.85	12.47	29.17
	0.8	23	1.02	3.29	50	24.46	82.84	50	5.39	13.87	29.26
	1	13	0.31	1.05	50	22.36	84.09	50	5.01	14.86	26.47
1	0.2	44	2.08	8.89	50	18.29	81.36	50	2.68	9.76	22.14
	0.4	40	1.80	7.63	50	18.90	82.81	50	2.66	9.56	22.48
	0.6	31	0.95	4.47	50	17.77	84.68	50	2.74	10.85	20.79
	0.8	27	0.66	3.17	50	17.44	86.20	50	2.56	10.63	20.09
	1	29	0.55	2.74	50	16.27	84.75	50	2.83	12.50	19.08
global		639	4.39	7.26	1250	48.16	81.43	1238	31.89	11.31	55.12

Table 9 – Comparison between the QATC and the QWMDD after the improvement methods are applied

m	n	%ivw_QATC_M_NEH_LS	%ivw_QWMDD_M_NEH_LS	QATC_M_NEH_LS btr QWMDD_M_NEH_LS	QATC_M_NEH_LS eql QWMDD_M_NEH_LS	QATC_M_NEH_LS wrs QWMDD_M_NEH_LS
5	10	0.43	0.35	28	1185	37
	15	1.23	1.03	114	1029	107
	25	1.78	1.65	198	858	194
	30	1.56	1.60	222	821	207
	50	1.66	1.40	236	758	256
	100	1.17	1.08	243	739	268
	300	0.46	0.78	256	760	234
10	10	0.07	0.14	16	1220	14
	15	0.46	0.73	55	1113	82
	25	1.74	1.93	171	904	175
	30	2.37	2.64	190	862	198
	50	2.79	2.81	229	771	250
	100	1.36	1.67	245	745	260
	300	1.24	1.12	252	740	258
20	10	0.01	0.01	3	1244	3
	15	0.09	0.14	23	1198	29
	25	0.43	0.53	95	1054	101
	30	0.72	0.66	137	997	116
	50	1.66	1.42	226	820	204
	100	2.60	2.63	254	733	263
	300	1.91	1.21	318	707	225
global		1.16	1.14	6443	39687	6370

Table 10 – Comparison between the QATC and the QWMDD after the improvement methods are applied, for $n = 300$ and $M = 10$

T	R	%ivw_QATC_M_NEH_LS	%ivw_QWMDD_M_NEH_LS	QATC_M_NEH_LS btr QWMDD_M_NEH_LS	QATC_M_NEH_LS eql QWMDD_M_NEH_LS	QATC_M_NEH_LS wrs QWMDD_M_NEH_LS
0.2	0.2	1.42	1.81	23	0	27
	0.4	8.77	8.63	27	1	22
	0.6	0.00	0.00	0	50	0
	0.8	0.00	0.00	0	50	0
	1	0.00	0.00	0	50	0
0.4	0.2	0.71	0.61	25	0	25
	0.4	1.17	1.09	25	0	25
	0.6	2.00	1.53	25	0	25
	0.8	7.57	7.29	26	0	24
	1	7.69	4.62	5	41	4
0.6	0.2	0.34	0.59	22	0	28
	0.4	0.40	0.57	23	0	27
	0.6	0.49	0.83	21	0	29
	0.8	0.00	0.00	0	50	0
	1	0.00	0.00	1	49	0
0.8	0.2	0.38	0.39	29	0	21
	0.4	0.00	0.07	0	49	1
	0.6	0.00	0.00	0	50	0
	0.8	0.00	0.00	0	50	0
	1	0.00	0.00	0	50	0
1	0.2	0.00	0.00	0	50	0
	0.4	0.00	0.00	0	50	0
	0.6	0.00	0.00	0	50	0
	0.8	0.00	0.00	0	50	0
	1	0.00	0.00	0	50	0
global		1.24	1.12	252	740	258

Table 11 – Runtimes for each improvement method's step for both the QATC and the QWMDD rules

m	n	QATC	QATC_M	QATC_M_NEH	QATC_M_NEH_LS	QWMDD	QWMDD_M	QWMDD_M_NEH	QWMDD_M_NEH_LS
5	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	50	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.02
	100	0.00	0.00	0.00	0.14	0.00	0.00	0.00	0.15
	300	0.00	0.01	0.08	5.71	0.00	0.01	0.07	6.12
10	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	30	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01
	50	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.03
	100	0.00	0.01	0.01	0.27	0.00	0.00	0.01	0.28
	300	0.01	0.05	0.16	10.54	0.00	0.03	0.15	11.24
20	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	25	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01
	30	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.01
	50	0.00	0.01	0.01	0.06	0.00	0.00	0.01	0.06
	100	0.00	0.03	0.04	0.55	0.00	0.02	0.03	0.57
	300	0.02	0.21	0.44	21.45	0.01	0.14	0.38	22.19
avg		0.00	0.01	0.03	1.16	0.00	0.01	0.02	1.22

References

Fernandez-Viagas, V. and J. M. Framinan (2015). "NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness." Computers & Operations Research **60**(0): 27-36.

Hasija, S. and C. Rajendran (2004). "Scheduling in flowshops to minimize total tardiness of jobs." International Journal of Production Research **42**(11): 2289-2301.

Kanet, J. J. and X. M. Li (2004). "A weighted modified due date rule for sequencing to minimize weighted tardiness." Journal of Scheduling **7**(4): 261-276.

M'Hallah, R. (2014). "An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop." International Journal of Production Research **52**(13): 3802-3819.

Nawaz, M., E. E. Enscore Jr and I. Ham (1983). "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem." Omega **11**(1): 91-95.

Ow, P. S. and T. E. Morton (1989). "The Single-Machine Early Tardy Problem." Management Science **35**(2): 177-191.

Parthasarathy, S. and C. Rajendran (1998). "Scheduling to minimize mean tardiness and weighted mean tardiness in flowshop and flowline-based manufacturing cell." Computers & Industrial Engineering **34**(2): 531-546.

Ruiz, R. and T. Stützle (2008). "An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives." European Journal of Operational Research **187**(3): 1143-1159.

Schaller, J. and J. M. S. Valente (2012). "Minimizing the weighted sum of squared tardiness on a single machine." Computers & Operations Research **39**(5): 919-928.

Schaller, J. and J. M. S. Valente (2013). "A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness." International Journal of Production Research **51**(3): 772-779.

Taillard, E. (1993). "Benchmarks for basic scheduling problems." European Journal of Operational Research **64**(2): 278-285.

Valente, J. M. S. and J. E. Schaller (2012). "Dispatching heuristics for the single machine weighted quadratic tardiness scheduling problem." Computers & Operations Research **39**(9): 2223-2231.

Vepsalainen, A. P. J. and T. E. Morton (1987). "Priority Rules for Job Shops with Weighted Tardiness Costs." Management Science **33**(8): 1035-1047.