Flexible Services and Manufacturing Journal (2020) 32:487–522
https://doi.org/10.1007/s10696-019-09363-6

Efficient procedures for the weighted squared tardiness permutation flowshop scheduling problem

Maria Raquel C. Costa[1] · Jorge M. S. Valente[2] · Jeffrey E. Schaller[3]

Published online: 2 July 2019

Abstract

This paper addresses a permutation flowshop scheduling problem, with the objective of minimizing total weighted squared tardiness. The focus is on providing efficient procedures that can quickly solve medium or even large instances. Within this context, we first present multiple dispatching heuristics. These include general rules suited to various due date-related environments, heuristics developed for the problem with a linear objective function, and procedures that are suitably adapted to take the squared objective into account. Then, we describe several improvement procedures, which use one or more of three techniques. These procedures are used to improve the solution obtained by the best dispatching rule. Computational results show that the quadratic rules greatly outperform the linear counterparts, and that one of the quadratic rules is the overall best performing dispatching heuristic. The computational tests also show that all procedures significantly improve upon the initial solution. The non-dominated procedures, when considering both solution quality and runtime, are identified. The best dispatching rule, and two of the non-dominated improvement procedures, are quite efficient, and can be applied to even very large-sized problems. The remaining non-dominated improvement method can provide somewhat higher quality solutions, but it may need excessive time for extremely large instances.

Keywords Scheduling · Permutation flowshop · Weighted squared tardiness · Dispatching rules · Improvement procedures

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10696-019-09363-6) contains supplementary material, which is available to authorized users.

Jorge M. S. Valente
jvalente@fep.up.pt

Extended author information available on the last page of the article

1 Introduction

In this paper, we consider a permutation flowshop scheduling problem. In a flowshop production environment, jobs are processed on a set of machines, and all jobs follow the same route through the machines. A permutation flowshop is considered, meaning that the processing order of the jobs is the same for all machines.

Each job has a weight, which reflects the importance of the job, or of the associated customer. The date by which the job should be completed is called the due date. A job is considered tardy if completed after its due date, and its tardiness is simply the amount of time by which it is completed late. The objective is to find a sequence of the jobs that minimizes the sum of the weighted squared tardiness values.

The flowshop production environment is quite common in practice, and has been often studied. The permutation assumption is usually made, since it not only reduces the computational effort, but is also realistic, since in practice it is often difficult, or even impossible, to change the order of the jobs between machines. When dealing with customer due dates, the tardiness measure is also widely used, since delays can lead to contractual penalties and/or loss of customers or goodwill.

In this paper, we consider a squared tardiness objective function. A large number of studies, however, have considered a linear tardiness performance measure. Additionally, the minimization of the maximum tardiness among all jobs is also a common tardiness-related objective. All of these tardiness measures are relevant, and none is inherently superior to the others. The choice of which one to use in a practical setting depends on how tardiness affects customers, as well as on the preferences or priorities of the decision maker.

When maximum tardiness is used, the focus is on the largest amount of tardiness that may result for a customer. However, any costs incurred by other tardy jobs are not considered. If linear tardiness is chosen, all jobs are taken into account. However, the distribution of the tardiness among jobs is not considered, so having two jobs that are both four time units tardy is equivalent to having one job seven time units tardy and another job one time unit tardy. With squared tardiness, large values of tardiness are more harshly penalized. Therefore, schedules in which one or only a few jobs contribute the majority of the cost are avoided, as described in more detail in Sun et al. (1999).

Also, while in linear tardiness the incremental penalty of a job remains constant as tardiness increases, in squared tardiness the incremental penalty increases with tardiness, as pointed out by Hoitomt et al. (1990) and Thomalla (2001). This is in agreement with the loss function proposed by Taguchi (1986), in which the dissatisfaction of a customer grows quadratically with tardiness. Therefore, an objective based on squared tardiness is appropriate for real settings. Indeed, scheduling methodologies developed by Hoitomt et al. (1990) and Luh and Hoitomt (1993) considered a squared tardiness objective, and were actually put in practice at a Pratt and Whitney plant.

In this paper, we focus on efficient procedures, capable of quickly solving even medium or large instances. In this context, several dispatching heuristics are first presented. Dispatching rules are widely used in practice, and most real scheduling systems are either based on them, or at least use them to some degree. They are also quite fast, and are often the only approach capable of providing solutions for large instances in reasonable time. Furthermore, dispatching rules are frequently used by other procedures; for instance, they are often used in metaheuristics, to generate an initial solution.

We include three types of dispatching rules. First, we consider general dispatching rules, that is, rules that have been used for multiple problems with due dates, in various production settings. Second, we also consider rules that were developed for the linear tardiness problem. Finally, we additionally include quadratic rules, that essentially modify the linear rules in order to take into account the squared tardiness objective.

Most of the dispatching rules were previously used in single machine or parallel machines settings. Thus, they are suitably adapted in order to be applied to a flowshop environment. Also, some procedures require a user-defined parameter. Experiments were performed to determine an adequate value for that parameter, under the permutation flowshop setting.

Then, we present several improvement procedures, which are applied to the best of the dispatching rules. These improvement methods use one or more of three techniques, namely: multiple sequence dispatching, the well-known NEH procedure (Nawaz et al. 1983), and local search with an insertions neighborhood. To the best of our knowledge, multiple sequence dispatching was previously used only in the context of the EDD (earliest due date) rule. In this paper, we describe its application to a different, and more complex, dispatching heuristic.

The various improvement procedures are analyzed, in order to determine the ones which are non-dominated, when taking into account both solution quality and computation time. Therefore, the computational results in this paper provide a guide to decision makers on the method of choice (dispatching rule and/or different improvement procedures), given the time available to generate a solution.

The remainder of the paper is organized as follows. In Sect. 2, the problem is formally described, and the relevant literature is reviewed. The dispatching rules are presented in Sect. 3. First, some notation is introduced. Then, the general, linear and quadratic rules are presented. A lower bound on the makespan, required by some of the heuristics, is also described.

The improvement procedures are addressed in Sect. 4. The multiple sequence version of the best dispatching rules is presented first. Then, the NEH and insertions local search procedures are described. Additional methods, which combine two or more of the multiple sequence, NEH and insertions procedures, are then presented.

Section 5 contains the computational results. First, we describe the problem set, performance measures and preliminary parameter adjustment tests. The dispatching rules are then compared, followed by an analysis of the improvement procedures. The non-dominated procedures are also compared with optimal solutions for small instances. Finally, Sect. 6 concludes the paper.

2 Problem formulation and literature review

Formally, the problem considered in this paper can be stated as follows. A set $N = \{1, 2, \ldots, n\}$ of n independent jobs have to be processed on a set $M = \{1, 2, \ldots, m\}$ 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 (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, $C_{1[j]} = C_{1[j-1]} + p_{1[j]}$ and $C_{k[j]} = max\left\{ C_{k-1[j]}, C_{k[j-1]} \right\} + p_{k[j]}$, for $k = \{2, 3, \ldots, m\}$. Finally, 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 $C_j = C_{mj}$.

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

The squared tardiness objective function has been previously studied in various production settings. Several papers addressed the single machine problem. Approaches include: dominance rules and branch-and-bound procedures incorporating these rules (Schaller and Valente 2012), efficient dispatching heuristics (Valente and Schaller 2012), and metaheuristics (Gonçalves et al. 2016). Various heuristics have also been proposed for the single machine problem, but with release dates and sequence-dependent setups (Sun et al. 1999).

A parallel machines environment has also been addressed. In this context, a Lagrangian relaxation procedure was developed and applied to some examples from a Pratt and Whitney plant (Hoitomt et al. 1990; Luh and Hoitomt 1993). Furthermore, several efficient heuristics and an improvement procedure were proposed by Schaller and Valente (2018).

Multiple stage problems were addressed by Luh and Hoitomt (1993), Thomalla (2001) and Dalfard et al. (2011). In both Luh and Hoitomt (1993) and Thomalla (2001), a Lagrangian relaxation procedure is developed for a job shop environment. Dalfard et al. (2011) consider a three-stage problem, with parallel machines with sequence-dependent setup times at the first stage, transportation times in a second stage, and assembly of components into a final product in the third stage. A hybrid genetic algorithm is proposed to minimize the weighted sum of four objectives, one of which is weighted squared tardiness.

Therefore, the literature on multiple stage problems with a squared tardiness objective is limited and, to the best of our knowledge, has never addressed the specific problem we consider. Furthermore, the literature on multiple stage environments considers job shops, and multi-stage scenarios with an assembly stage, which are quite different in nature from a permutation flowshop. Thus, this paper addresses a gap in the existing literature.

The flowshop is a common manufacturing environment, and as such has been studied in a quite large number of papers, as illustrated by the multiple reviews that have been conducted (Framinan et al. 2004; Neufeld et al. 2016; Reza Hejazi and Saghafian 2005; Ruiz and Maroto 2005; Sun et al. 2011). The minimization of the makespan is likely the most addressed objective function. A comprehensive review and evaluation of both constructive procedures and metaheuristics for the makespan objective is provided by Fernandez-Viagas et al. (2017). The total completion time objective has also been considered in many papers. In what regards this objective, an overview and comparison of multiple algorithms is given in Fernandez-Viagas and Framinan (2015b), while Fernandez-Viagas et al. (2016) perform a computational evaluation of constructive and improvement procedures for a flowshop with the blocking constraint.

Multiple papers have considered the permutation flowshop with a total linear unweighted tardiness objective. A review and evaluation of heuristics and metaheuristics is given in Vallada et al. (2008). Genetic algorithms with path relinking are proposed by Vallada and Ruiz (2010), and compared with state-of-the-art methods. A comparison of procedures based on the NEH heuristic (Nawaz et al. 1983) is conducted in Fernandez-Viagas and Framinan (2015a). A hybrid iterated greedy procedure is proposed in Karabulut (2016). Iterated-greedy-based algorithms with a beam search initialization were developed by Fernandez-Viagas et al. (2018), and a comprehensive computational comparison was performed against existing procedures.

3 Dispatching rules

3.1 Notation

In the following, let S be the current partial schedule, that is, the sequence of jobs that are scheduled so far. The completion time of job $j \notin S$, if j is scheduled at the end of sequence S, is denoted by $C_j(S)$. Also, let $s_j(S)$ be the slack of job $j \notin S$ if j is scheduled at the end of S, where $s_j(S) = d_j - C_j(S)$.

The current availability time of machine i under schedule S will be represented by $t_i(S)$. 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. The average, over all jobs $j \notin S$, of the $P_j(S)$ values will be denoted by $\overline{P}(S)$. Finally, let $T_j(S) = max\left\{C_j(S) - d_j;0\right\}$ be the tardiness of job $j \notin S$ if j is scheduled at the end of sequence S.

3.2 General rules

Several general rules are considered because they have been previously used for multiple problems with due date related criteria, under multiple production settings. We do not expect these rules to match the performance of the other, more sophisticated, dispatching procedures. Nevertheless, their inclusion is still warranted, since they are widely used in a variety of environments, and usually considered for comparison purposes.

The earliest due date (EDD) (Jackson 1955) is one of the first and most well-known sequencing rules, and has been extensively applied to scheduling models with due dates. This rule schedules the jobs in non-decreasing order of their due dates d_j. Equivalently, the EDD rule selects, 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 (Ruiz and Stützle 2008) schedules the jobs in non-decreasing order of their weighted due dates d_j/w_j, thereby expanding the EDD heuristic to take into account job-specific weights. Equivalently, 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 (Baker and Bertrand 1982; Vepsalainen and Morton 1987), at each iteration we select the job with the minimum value of the modified due date $max\{d_j, C_j(S)\} = max\{d_j, t + P_j(S)\} = max\{d_j - t, P_j(S)\}$. Alternatively, this rule selects, at each iteration, the job with the largest value of the priority index $MDD_j(S)$:

$$MDD_j(S) = \begin{cases} 1/P_j(S) & if\ s_j(S) \leq 0 \\ 1/\left(d_j - t\right) & otherwise \end{cases}.$$

The minimum slack (SLK) rule (Panwalkar and Iskander 1977; Vepsalainen and Morton 1987) chooses, at each iteration, the job with the minimum slack $s_j(S)$ or, equivalently, the job with the largest value of the priority index $SLK_j(S) = -s_j(S)$. The minimum slack per required time (SLK/P) (Panwalkar and Iskander 1977; Vepsalainen and Morton 1987), on the other hand, selects, at each iteration, the job with the minimum value of the ratio between the slack and the total required time $s_j(S)/P_j(S)$. Alternatively, it chooses the job with the largest value of the priority index $SLK/P_j(S) = -\left(s_j(S)/P_j(S)\right)$.

3.3 Rules for the linear objective function

We consider a simple but commonly used procedure, as well as more sophisticated rules, including those shown to have performed best in the single machine linear weighted tardiness problem. Again, it is expected that these rules will be outperformed by the quadratic procedures, which specifically take the squared objective function into account. However, the inclusion of the linear rules makes it possible to evaluate how much of an improvement is made possible by taking the quadratic nature of the problem into consideration, instead of simply using procedures developed for the linear setting.

Most of these procedures were originally developed for a single machine environment. Some of them, however, have also been applied to other production settings. In this subsection, we explicitly show how the priority indexes of these procedures are adjusted in order to take the flowshop environment into account.

The weighted shortest processing time (WSPT) rule (Smith 1956) schedules the jobs in non-increasing order of the ratio $w_j/P_j(S)$ or, equivalently, chooses, at each iteration, the job with the highest priority index $WSPT_j(S) = w_j/P_j(S)$. This rule provides an optimal sequence for the single machine linear problem if all jobs are necessarily tardy.

The weighted minimum slack/shortest processing time (WSLK_SPT) rule (Osman et al. 2009) 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 $WSLK\_SPT_j(S)$:

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

The weighted modified due date (WMDD) (Kanet and Li 2004), Alidaee–Ramakrishnan (AR) (Alidaee and Ramakrishnan 1996) and Apparent Tardiness Cost (ATC) (Vepsalainen and Morton 1987) heuristics were developed for the single machine linear problem, and several computational studies show that they provide the best performance among the efficient dispatching rules available for that problem (Alidaee and Ramakrishnan 1996; Kanet and Li 2004; Volgenant and Teerhuis 1999). These heuristics select, at each iteration, the job with the largest value of the following priority indexes:

$$WMDD_j(S) = \begin{cases} w_j/P_j(S) & if\ s_j(S) \le 0 \\ w_j/\left(d_j - t\right) & otherwise \end{cases},$$

$$AR_j(S) = \begin{cases} w_j/P_j(S) & if\ s_j(S) \le 0 \\ \left(w_j/P_j(S)\right) * \left[k\overline{P}(S)/\left(k\overline{P}(S) + s_j(S)\right)\right] & otherwise \end{cases} \quad \text{and}$$

$$ATC_j(S) = \begin{cases} w_j/P_j(S) & if\ s_j(S) \le 0 \\ \left(w_j/P_j(S)\right) * \exp\left(-s_j(S)/k\overline{P}(S)\right) & otherwise \end{cases}.$$

The parameter k provides the ATC and AR heuristics with a look ahead capability. Indeed, and as described by Vepsalainen and Morton (1987), the parameter k is related with the number of competing critical jobs. i.e. jobs which are close to becoming tardy. In this paper, we consider a job to be critical if its slack is positive, but less than or equal to a value slk_thr, which stands for "slack threshold". Thus, a job is considered critical if $0 < s_j(S) \leq slk\_thr$.

At each iteration, k is then set equal to the number of critical jobs. If, at a given iteration, no job is critical according to our criterion, k is then set equal to 0.5, since this is the lowest value that has been usually considered for this parameter (Alidaee and Ramakrishnan 1996; Holsenback et al. 1999).

The slack threshold is meant to represent a value such that slacks which are greater are considered large, so the job is not close to becoming tardy, and is therefore not critical. As such, we calculate the slk_thr parameter as follows. At each iteration, the slack threshold is set equal to $slk\_thr = v \times \left(C_{max}^{LB}(S) - t\right)$, where $C_{max}^{LB}(S)$ is a lower bound on the completion time of the last job on the final machine (makespan), given the current schedule S, and $0 \leq v \leq 1$ is a user–defined parameter. The calculation of the lower bound on the makespan will be described at the end of this section.

3.4 Rules for the quadratic objective function

The linear rules presented in the previous subsection have been adapted to a weighted quadratic tardiness objective, and tested under a single machine environment. We will show how their priority indexes are adjusted in order to take the flowshop setting into account. Since the quadratic rules specifically consider the squared nature of the objective function, it is expected that they will outperform the previous procedures.

The quadratic weighted shortest processing time (QWSPT) rule (Valente and Alves 2008) is an adaptation of the WSPT heuristic to a quadratic setting. At each iteration, the QWSPT rule selects the job with the largest value of the priority index $QWSPT_j(S) = \left(w_j/P_j(S)\right) * \left(\overline{P}(S) + 2T_j(S)\right)$.

The WSLK_SPT rule can be adapted to a quadratic setting by essentially replacing, in its priority index, the WSPT component by a QWSPT expression. The resulting quadratic weighted minimum slack/shortest processing time (QWSLK_SPT) rule chooses, at each iteration, the job with the largest value of the priority index $QWSLK\_SPT_j(S)$:

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

The WMDD, AR and ATC rules have been adapted to a quadratic setting (Valente and Schaller 2012) by, once more, replacing WSPT by QWSPT in their priority indexes. The resulting QWMDD, QAR and QATC heuristics select, at each iteration, the job with the largest value of the priority indexes:

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

$$QAR_j(S) = \begin{cases} \left(w_j/P_j(S)\right) * \left(\overline{P}(S) + 2T_j(S)\right) & \textit{if } s_j(S) \le 0 \\ \left(w_j/P_j(S)\right) * \overline{P}(S) * \left[k\overline{P}(S)/\left(k\overline{P}(S) + s_j(S)\right)\right] & \textit{otherwise} \end{cases} \quad \text{and}$$

$$QATC_j(S) = \begin{cases} \left(w_j/P_j(S)\right) * \left(\overline{P}(S) + 2T_j(S)\right) & \textit{if } s_j(S) \le 0 \\ \left(w_j/P_j(S)\right) * \overline{P}(S) * \exp\left(-s_j(S)/k\overline{P}(S)\right) & \textit{otherwise} \end{cases}.$$

A list of all the considered procedures (general, linear and quadratic), along with their corresponding priority index and references, is provided in Table 1.

3.5 Lower bound on the makespan

The AR, ATC, QAR and QATC heuristics use the look ahead parameter k in their priority indexes. In order to set the value of k at each iteration we require $C_{max}^{LB}(S)$, a lower bound on the completion time of the last job on the final machine, given the current schedule S.

This lower bound on the makespan is calculated using an adaptation of the procedure proposed by Taillard (1993). Indeed, the lower bound given in Taillard (1993) assumes that all machines are available at time zero. Since a lower bound must here be calculated at each iteration, the procedure was adapted to deal with non-zero machine availability times, which will necessarily occur as jobs are scheduled.

The lower bound $C_{max}^{LB}(S)$ is then calculated as follows. Let $Bef\_M_i(S) = min_{j \notin S}\left(t + \sum_{k=1}^{i-1} p_{kj}\right)$. Then, $Bef\_M_i(S)$ is a lower bound on the time needed before reaching machine i, since it considers the availability time on the first machine, plus the minimum, over all unscheduled jobs, of the sum of the processing times on the machines that precede i. Also let $TPT\_M_i(S) = \sum_{j \notin S} p_{ij}$; thus, $TPT\_M_i(S)$ is simply the total processing time required by all unscheduled jobs on machine i. Furthermore, let $Aft\_M_i(S) = min_{j \notin S}\left(\sum_{k=i+1}^{m} p_{kj}\right)$. Then, $Aft\_M_i(S)$ is a lower bound on the time required after machine i, since it considers the minimum, over all unscheduled jobs, of the sum of the processing times on the machines that follow i.

For each machine i, a lower bound on the makespan can then be calculated as $C_{max}^{LB}\_M_i(S) = max\left(Bef\_M_i(S), t_i\right) + TPT\_M_i(S) + Aft\_M_i(S)$. In the original lower bound presented in Taillard (1993), all the machine availability times were assumed to be zero, since the lower bound was being calculated for all jobs.

Table 1 Heuristic priority indexes

Heuristic	Reference (s)	Priority index	Remarks
EDD	(Jackson 1955)	$-d_j$	*General rules*
EWDD	(Ruiz and Stützle 2008)	w_j/d_j	These rules have been used for multiple problems with due date related criteria, under various production settings
MDD	(Baker and Bertrand 1982; Vepsalainen and Morton 1987)	$\begin{cases} 1/P_j(S) & \textit{if } s_j(S) \le 0 \\ 1/\left(d_j - t\right) & \textit{otherwise} \end{cases}$	
SLK	(Panwalkar and Iskander 1977; Vepsalainen and Morton 1987)	$-s_j(S)$	
SLK/P	(Panwalkar and Iskander 1977; Vepsalainen and Morton 1987)	$-\left(s_j(S)/P_j(S)\right)$	
WSPT	(Smith 1956)	$w_j/P_j(S)$	*Linear rules*
WSLK_SPT	(Osman et al. 2009)	$\begin{cases} w_j/P_j(S) & \textit{if } s_j(S) \le P_j(S) \\ w_j/s_j(S) & \textit{otherwise} \end{cases}$	These rules have been applied to problems with a linear weighted tardiness objective function
WMDD	(Kanet and Li 2004)	$\begin{cases} w_j/P_j(S) & \textit{if } s_j(S) \le 0 \\ w_j/\left(d_j - t\right) & \textit{otherwise} \end{cases}$	
AR	(Alidaee and Ramakrishnan 1996)	$\begin{cases} w_j/P_j(S) & \textit{if } s_j(S) \le 0 \\ \left(w_j/P_j(S)\right) * \left[k\overline{P}(S)/\left(k\overline{P}(S) + s_j(S)\right)\right] & \textit{otherwise} \end{cases}$	
ATC	(Vepsalainen and Morton 1987)	$\begin{cases} w_j/P_j(S) & \textit{if } s_j(S) \le 0 \\ \left(w_j/P_j(S)\right) * \exp\left(-s_j(S)/k\overline{P}(S)\right) & \textit{otherwise} \end{cases}$	

Table 1 (continued)

Heuristic	Reference (s)	Priority index	Remarks
QWSPT	(Valente and Alves 2008)	$(w_j/P_j(S)) * \left(\overline{P}(S) + 2T_j(S)\right)$	*Quadratic rules* These are adaptations of the linear rules to a weighted quadratic tardiness objective function
QWSLK_SPT		$\begin{cases} (w_j/P_j(S)) * \left(\overline{P}(S) + 2T_j(S)\right) & \textit{if } s_j(S) \le P_j(S) \\ (w_j/s_j(S)) * \overline{P}(S) & \textit{otherwise} \end{cases}$	
QWMDD	(Valente and Schaller 2012)	$\begin{cases} (w_j/P_j(S)) * \left(\overline{P}(S) + 2T_j(S)\right) & \textit{if } s_j(S) \le 0 \\ (w_j/(d_j - t)) * \overline{P}(S) & \textit{otherwise} \end{cases}$	
QAR	(Valente and Schaller 2012)	$\begin{cases} (w_j/P_j(S)) * \left(\overline{P}(S) + 2T_j(S)\right) & \textit{if } s_j(S) \le 0 \\ (w_j/P_j(S)) * \overline{P}(S) * \left[k\overline{P}(S)/\left(k\overline{P}(S) + s_j(S)\right)\right] & \textit{otherwise} \end{cases}$	
QATC	(Valente and Schaller 2012)	$\begin{cases} (w_j/P_j(S)) * \left(\overline{P}(S) + 2T_j(S)\right) & \textit{if } s_j(S) \le 0 \\ (w_j/P_j(S)) * \overline{P}(S) * \exp\left(-s_j(S)/k\overline{P}(S)\right) & \textit{otherwise} \end{cases}$	

Given a partial schedule, and/or machine availability times which differ from zero, two adaptations were then required.

The first is to include the availability time of the first machine in $Bef\_M_i(S)$. The second is in using the maximum between the lower bound on the time needed before reaching machine i and the availability time of this machine: $max\left(Bef\_M_i(S), t_i\right)$. The lower bound on the makespan $C_{max}^{LB}(S)$ is then simply equal to the maximum of all machine lower bounds, that is, $C_{max}^{LB}(S) = max_{i\in M}\left(C_{max}^{LB}\_M_i(S)\right)$.

4 Improvement procedures

4.1 Multiple sequence dispatching rule

The first improvement procedure is a method that modifies a dispatching heuristic so that it generates multiple sequences, instead of a single one. This method, as well as those described in the next subsections, was only applied to the QATC rule, since this was the best performing of the heuristic procedures. We will denote this method by MS, standing for multiple sequences. The multiple sequence version of the QATC rule builds m sequences, one for each machine and using data related to that machine, and selects the best of those sequences.

The MS version is inspired by the earliest apportioned due date (EADD) heuristic developed by Hasija and Rajendran (2004). Indeed, this heuristic first calculates a due date for each job on each machine. Then, a sequence is obtained for each machine i by scheduling the jobs in non-decreasing order of their due dates on that machine. Finally, the best of those m sequences is then selected.

The multiple sequence version of the QATC heuristic also uses the apportioned due dates of the EADD procedure. Additional modifications of the QATC rule are, however, required in order to develop a multiple sequence version. Indeed, the EADD heuristic is a multiple sequence version of the EDD rule, which relies solely on a job's due date. The QATC priority index, on other hand, uses additional information. Thus, more extensive changes are needed in order to achieve a procedure that adequately generates multiple sequences, one for each machine and using data related to that machine.

The apportioned due date of job j on machine i, denoted by d_{ij}, is obtained precisely by apportioning 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 $d_{mj} = d_j$. Formally, the due dates d_{ij} are then calculated as:

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

and

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

In order to present the MS version of QATC, some notation must first be defined. 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 version of QATC, the general slack $s_j(S)$ is then replaced, in the priority index, by the machine–dependent slack $s_{ij}(S)$.

Similarly, let $T_{ij}(S) = max\left\{C_{ij}(S) - d_{ij};0\right\}$ be the tardiness of job $j \notin S$, on machine i, if j is scheduled at the end of sequence S. Again, the machine–dependent tardiness $T_{ij}(S)$ is used in the priority index of the multiple sequence QATC, instead of the general tardiness $T_j(S)$.

Let $C^{LB}_{\max}\_M_i(S)$ 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 $C^{LB}_{\max}\_M_i(S)$ is calculated as previously described for the final machine lower bound $C^{LB}_{max}(S)$, 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 is calculated as before, with the difference that the machine lower bound $C^{LB}_{\max}\_M_i(S)$ replaces the final machine lower bound $C^{LB}_{max}(S)$, that is $slk\_thr = v \times \left(C^{LB}_{\max}\_M_i(S) - t\right)$.

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 the multiple sequence version of QATC, 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 $P_{ij}(S)$. In the same way, let $\overline{P}_i(S)$ be the average, over all jobs $j \notin S$, of the $P_{ij}(S)$ values. Again, $\overline{P}_i(S)$ takes the place of $\overline{P}(S)$ in the multiple sequence version.

In short, and when considering machine i, the priority index of the multiple sequence procedure is then obtained by replacing $s_j(S)$, $T_j(S)$, $P_j(S)$ and $\overline{P}(S)$ by their machine–specific counterparts $s_{ij}(S)$, $T_{ij}(S)$, $P_{ij}(S)$ and $\overline{P}_i(S)$, respectively. The priority index of the multiple sequence version of QATC is then equal to:

$$QATC_{ij}(S) = \begin{cases} \left(w_j/P_{ij}(S)\right) * \left(\overline{P}_i(S) + 2T_{ij}(S)\right) & if\ s_{ij}(S) \leq 0 \\ \left(w_j/P_{ij}(S)\right) * \overline{P}_i(S) * \exp\left(-s_{ij}(S)/k\overline{P}_i(S)\right) & otherwise \end{cases}$$

We remark that procedure MS will generate a sequence that is at least as good as the one obtained by the original (single sequence) QATC heuristic. This is due to the fact that the solution obtained for the last machine is the same as the one generated by the original QATC rule. Indeed, and for the last machine m, we have $s_{mj}(S) = s_j(S)$, $T_{mj}(S) = T_j(S)$, $P_{mj}(S) = P_j(S)$, $\overline{P}_m(S) = \overline{P}(S)$ and $C^{LB}_{\max\_}M_m(S) = C^{LB}_{\max}(S)$.

4.2 NEH insertion procedure

The well-known NEH method, developed by Nawaz et al. (1983), is an insertion procedure that requires an initial sequence or list of the jobs. If the NEH method is applied alone, or when it comes first in a combined improvement procedure, this initial sequence or list is simply the solution provided by the QATC rule. When NEH is applied after another method in a combined procedure, the initial sequence is the one generated by the previous improvement method.

Given the initial sequence, an insertion procedure is then used to create another sequence. 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 current partial sequence. The job is then inserted in the position which provides the best objective function value.

The importance of using a tie–breaking method in the NEH procedure in the context of the total tardiness objective has been analyzed by Fernandez-Viagas and Framinan (2015a). This work showed that an appropriate tie–breaking method could lead to substantially better results. Indeed, and particularly in the first iterations and/or when the tardiness factor of a problem is low, multiple insertions positions may lead to the same lowest objective function value of 0. When this is the case, a good tie–breaking method is essential in enhancing the performance of the NEH procedure.

The tie–breaking methods proposed in Fernandez-Viagas and Framinan (2015a), though developed for the total tardiness problem, are still applicable to our weighted squared problem, in which the issue of adequately dealing with multiple objective function values of 0 is even more pressing, given our objective function includes squared tardiness. Therefore, our implementation of the NEH procedure uses the tie–breaking method which performed best among those proposed in Fernandez-Viagas and Framinan (2015a), and which was denoted by Total Idle Time IT1.

In short, when multiple insertion positions lead to the same lowest objective function value, the tie is broken by selecting the position with the minimum value of the total idle time over all machines. In the Total Idle Time IT1 method, the definition of idle time includes front delays (idle time before the first job starts on a machine) but excludes back delays (the time between the finish time on a machine and the overall finish time). For further details concerning IT1, please see Fernandez-Viagas and Framinan (2015a).

Also, 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. This choice was motivated by some preliminary tests which showed that the NEH method could lead to a final schedule that was worse than the initial sequence. Though this behavior was infrequent, it was nevertheless decided to keep the best of the two sequences.

4.3 Insertions local search

A third improvement method consists in a local search procedure, using the insertions neighborhood, and a first-improve strategy. Given that an insertions neighborhood is used, this procedure will be denoted by INS.

In the insertions neighborhood, a move consists in removing one job from its current position, and inserting it in another position. In the INS procedure, all possible insertions are considered. An improving insertion is performed whenever it is detected (first-improve). This is repeated until no improving insertion is found.

4.4 Combined improvement procedures

We have considered not only the standalone application of each of the MS, NEH and INS methods, but also several other procedures which combine two of more of them, in order to see if a better performance could be obtained. In total, we considered four combined improvement procedures, denoted by MS+NEH, MS+INS, NEH+INS and MS+NEH+INS. In these four combined procedures, the improvement methods are applied in succession. So, and for instance, NEH+INS consists in applying NEH, followed by INS.

The four proposed combined procedures essentially correspond to the various combinations of the three standalone methods, when they are used in increasing order of their search space. Indeed, INS can be more disruptive, and generate more alternatives, than NEH, which is itself more general than MS.

5 Computational results

5.1 Problem set

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. The method chosen to generate the test problems is quite common, and in line with both initial tardiness papers (Ow and Morton 1988; Potts and van Wassenhove 1991), and recent works on permutation flowshop with a tardiness criterion (Vallada and Ruiz 2010; Vallada et al. 2008). More specifically, the problems were generated as follows.

In what regards the number of jobs, the following sizes were considered: 25, 50, 75, 100, 300 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, and the weight w_j was obtained using a uniform distribution [1, 10].

Finally, for each job j, the due date d_j was obtained using a uniform distribution $\left[MS(1-T-R/2), MS(1-T+R/2)\right]$, 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 dataset is available from the corresponding author, on reasonable request. 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 16 GB RAM.

5.2 Performance measures

The analysis and comparison of the dispatching rules will mostly rely on a measure of performance denoted by relative improvement versus the worst result (ivw). This measure was previously used by Valente and Schaller (2012) for the single machine scheduling problem with a weighted squared tardiness objective.

The relative improvement versus the worst result (ivw), for heuristic H_i, when evaluated with heuristics H_1, H_2,…,H_z, on a given instance, is calculated as follows. Let ofv_{worst} be the worst objective function value obtained by all the z heuristic procedures. If $ofv_{worst} = 0$, then the ivw for each heuristic H_i is set to 0; otherwise, ivw 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 measurement quantifies the improvement provided by a certain heuristic over the worst result provided by all of the considered procedures. As such, higher ivw values are indicative of a better performance.

The particular nature of the squared weighted tardiness problem motivated the use of the relative improvement versus the worst result performance measure, instead of more usual measures, such as the relative improvement over another heuristic, or the deviation from the best heuristic result. Indeed, as and also described in Valente and Schaller (2012), when due dates are relatively loose, or there is a wide range of due dates, a schedule with no tardy jobs is easy to find, with a resulting objective function value of 0.

When one or more heuristics find an optimal solution with an objective function value of 0, measures such as the deviation from the best heuristic result cannot be used, since they would lead to a division by 0. The relative improvement versus the worst result avoids this problem. Indeed, the only situation in which 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 were optimal and we have $ofv_{worst} = 0$. As mentioned above, when this occurs the relative improvement versus the worst is set at 0 for all heuristics, and division by 0 does not occur.

The number of times a dispatching rule provides a result that is better (btr), equal (eql) or worse (wrs) than another procedure will also be used as a performance measure. The computational time (in seconds) required by the dispatching rules is also considered.

The comparison of the improvement procedures will rely on three performance measures. The first is the relative improvement a procedure provides over the QATC dispatching rule, denoted by imp.

The relative improvement over the QATC rule (imp), for improvement procedure IP, on a given instance, is calculated as follows. Let ofv_{QATC} and ofv_{IP} be the objective function values of the schedules generated by the QATC rule and the improvement procedure, respectively. If $ofv_{QATC} = 0$, then the relative improvement imp is set to 0; otherwise, imp is calculated as $\left(ofv_{QATC} - ofv_{IP}\right)/ofv_{QATC} * 100$.

The number of times an improvement procedure provides a result that is better (btr) than that of the QATC rule is also used as a performance measure. The computational time (in seconds) required to run the improvement procedures, including the initial application of the QATC rule when appropriate, is also considered.

The comparison with the optimal results will involve two performance measures. One is simply the number of times a procedure provides the optimal solution, denoted by n_opt. The other measure is the relative improvement provided by the optimum objective function value over a heuristic procedure, previously used in Valente and Schaller (2012) for the single machine problem, denoted by ivh. This measure was chosen over the relative deviation from the optimum due to the same reason that motivated the use of the ivw measure.

The relative improvement provided by the optimum objective function value over heuristic procedure H, on a given instance, is calculated as follows. Let ofv_{OPT} be the optimum objective function value, while ofv_H is the objective function value of the schedule generated by heuristic procedure H, respectively. When $ofv_H = 0$, the relative improvement versus the heuristic procedure is set at 0. Otherwise, the relative improvement provided by the optimum is calculated as $\left(ofv_H - ofv_{OPT}\right)/ofv_H * 100$.

5.3 Parameter adjustment tests

The AR, ATC, QAR and QATC heuristics require a value for the parameter v, $0 \leq v \leq 1$. Preliminary tests were then performed in order to find a good value for v. In order to avoid possible overfitting, these tests were performed on a separate, and smaller, test set. This test set was generated in the same way as described for the full problem set. However, only five instances were generated for each combination of n, m, T and R.

The values $\{0.00, 0.05, 0.10, 0.15, 0.20, \ldots, 0.90, 0.95, 1.00\}$ were considered for the parameter v. The AR, ATC, QAR and QATC dispatching rules were then applied to the instances on the test set, and the objective function value was calculated for each considered value. These results were then analyzed, and we selected a value that provided good performance across all instance types. The value of v was then set at 0.00 for all four dispatching rules.

Setting v equal to 0.00 means that the slack threshold slk_thr will also be equal to 0, since at each iteration the slack threshold is set equal to $slk\_thr = v \times \left(C_{max}^{LB}(S) - t\right)$. Therefore, no job will ever be considered critical, since our criterion states that a job is critical if $0 < s_j(S) \leq slk\_thr$, and we will always have slk_thr equal to 0. As previously described, if no job is critical according to our criterion, the parameter k in the AR, ATC, QAR and QATC rules is set at 0.5. Therefore, a value of 0.00 for v means that the parameter k will always be equal to 0.5.

This result is quite different from those obtained in previous experiments in the single machine environment. Indeed, Valente and Schaller (2012) showed that, in the single machine problem, the most adequate value of v was 0.1 and 0.3, for the QAR and QATC rules, respectively. This highlights the importance of performing parameter adjustment tests for each production environment, instead of relying on values obtained for other settings. Indeed, if the single machine values for v were used, the performance of the QAR and QATC rules would have suffered.

5.4 Comparison of the dispatching rules

We first compare the linear dispatching rules (i.e. the rules developed for the linear weighted tardiness problem) with their quadratic counterparts (that is, their adaptations to the quadratic objective). Table 2 provides, for each pair of heuristics (quadratic and its linear equivalent), the mean ivw of each heuristic. This measure is denoted as ivw_q for the quadratic heuristic and ivw_l for the associated linear heuristic. The table also provides the number of instances in which the quadratic rule was better (btr), equal (eql) or worse (wrs) than its corresponding linear rule. The overall average (average) across all instances is also given.

Since Table 2 aims at directly comparing each quadratic rule with its associated linear rule, the values given in this table are calculated separately for each pair of quadratic procedure and corresponding linear counterpart. Thus, and as an example, in the comparison of QWSPT with WSPT, ofv_{worst} is the worst result among these two procedures.

The results in Table 2 clearly show that the quadratic dispatching rules significantly outperform their linear counterparts. Indeed, for the larger instances the mean ivw is between about 20 and 40% for the quadratic procedures, and close to 0 for the linear heuristics. Also, the quadratic rules provide better results for a quite large number of instances and, again for the larger problem sizes, are rarely worse.

We performed tests to determine if the differences between each quadratic rule and its linear counterpart are statistically significant. Since the heuristics were applied to the same instances, a paired-samples test can be conducted. The non-parametric Wilcoxon signed-rank test was selected, since the assumptions of the paired-samples t test were not all met.

The test was applied to each pair of heuristics, and for each combination of the number of jobs n and the number of machines m, and the significance level was set at 0.05. To take into account, and correct for, the multiple tests that were performed, we applied Holm's procedure (also known as Holm's sequential Bonferroni) to adjust the significance level.

These tests showed that the differences between the quadratic rules and their linear counterparts are statistically significant. Indeed, the hypothesis that quadratic and linear rules have similar performance was always rejected, for each pair of procedures.

Table 2 Comparison of quadratic and linear dispatching rules

m	n	QWSPT versus WSPT					QWSLK_SPT versus WSLK_SPT				
		ivw_q	ivw_l	btr	eql	wrs	ivw_q	ivw_l	btr	eql	wrs
5	25	23.78	1.35	1035	0	215	23.15	1.45	1034	0	216
	50	30.55	0.55	1130	0	120	29.35	0.61	1128	0	122
	75	34.68	0.28	1175	0	75	33.05	0.29	1166	0	84
	100	36.89	0.26	1178	0	72	34.95	0.27	1183	0	67
	300	44.34	0.04	1225	0	25	40.90	0.07	1222	0	28
	500	46.51	0.01	1241	0	9	42.05	0.05	1222	2	26
10	25	17.37	1.39	986	0	264	17.20	1.45	976	0	274
	50	23.60	0.60	1095	0	155	23.42	0.64	1101	0	149
	75	27.15	0.37	1143	0	107	26.62	0.37	1143	0	107
	100	30.35	0.31	1152	0	98	29.35	0.29	1159	0	91
	300	39.42	0.06	1216	0	34	37.02	0.04	1226	0	24
	500	42.79	0.01	1241	0	9	40.11	0.00	1241	0	9
20	25	11.41	1.57	909	0	341	11.34	1.57	913	0	337
	50	17.01	0.77	1025	0	225	16.97	0.77	1029	0	221
	75	20.31	0.42	1109	0	141	20.23	0.39	1113	0	137
	100	22.96	0.39	1119	0	131	22.84	0.36	1129	0	121
	300	33.26	0.03	1219	0	31	31.96	0.04	1218	0	32
	500	37.69	0.02	1232	0	18	35.78	0.02	1233	0	17
Average		30.00	0.47	1135	0	115	28.68	0.48	1135	0	115

Table 2 (continued)

m	*n*	QWMDD versus WMDD					QAR versus AR					QATC versus ATC				
		ivw_q	ivw_l	btr	eql	wrs	ivw_q	ivw_l	btr	eql	wrs	ivw_q	ivw_l	btr	eql	wrs
5	25	21.69	1.38	1017	0	233	20.14	1.57	1000	0	250	16.85	1.65	963	12	275
	50	28.38	0.52	1123	0	127	25.85	0.70	1087	0	163	20.17	0.73	1058	18	174
	75	32.05	0.30	1173	0	77	29.67	0.36	1154	0	96	21.17	0.48	1068	42	140
	100	34.20	0.20	1179	0	71	31.53	0.29	1154	0	96	21.51	0.49	1066	54	130
	300	39.78	0.11	1213	0	37	33.71	0.20	1144	28	78	21.37	0.27	1053	110	87
	500	40.68	0.12	1197	2	51	31.33	0.20	1099	73	78	21.30	0.18	1024	138	88
10	25	15.82	1.43	964	0	286	14.52	1.41	958	0	292	13.45	1.50	955	0	295
	50	21.88	0.61	1100	0	150	20.40	0.67	1086	0	164	17.94	0.75	1065	0	185
	75	25.28	0.38	1142	0	108	23.05	0.46	1125	0	125	19.90	0.48	1111	0	139
	100	28.19	0.27	1155	0	95	26.20	0.34	1137	0	113	22.64	0.45	1115	2	133
	300	36.28	0.05	1222	0	28	33.58	0.10	1199	0	51	23.64	0.17	1111	56	83
	500	39.30	0.03	1232	0	18	35.94	0.07	1206	2	42	23.17	0.14	1083	87	80
20	25	10.62	1.55	910	0	340	9.94	1.54	892	0	358	9.59	1.63	886	0	364
	50	15.53	0.77	1029	0	221	14.60	0.77	1034	0	216	13.84	0.79	1014	0	236
	75	19.03	0.35	1127	0	123	17.60	0.39	1106	0	144	16.43	0.42	1102	0	148
	100	21.74	0.33	1128	0	122	19.93	0.38	1119	0	131	17.95	0.40	1111	0	139
	300	31.01	0.04	1214	0	36	28.46	0.09	1189	0	61	24.66	0.13	1179	1	70
	500	34.84	0.02	1226	0	24	32.42	0.03	1225	0	25	25.22	0.07	1181	20	49
Average		27.57	0.47	1131	0	119	24.94	0.53	1106	6	138	19.49	0.60	1064	30	156

ivw_q and ivw_l: mean improvement versus the worst result, for the quadratic and the linear version, respectively

btr, eql, wrs: number of instances in which the quadratic rule performed better, equal or worse, respectively, than its corresponding linear rule

Table 3 Dispatching rules comparison (mean improvement versus the worst result)

m	n	EDD	EWDD	MDD	SLK	SLK/P	QWSPT	QWSLK_SPT	QWMDD	QAR	QATC
5	25	29.76	32.38	19.82	24.71	30.30	40.55	43.06	48.51	50.42	51.53
	50	34.98	30.85	20.07	32.57	36.55	43.23	49.34	54.64	56.58	58.54
	75	38.71	30.84	21.49	37.36	40.79	46.72	54.43	59.04	60.50	62.32
	100	40.83	31.84	21.69	39.65	42.29	49.57	57.86	61.91	63.13	64.52
	300	49.28	33.95	24.93	48.81	50.44	58.78	68.08	69.87	69.80	70.46
	500	51.71	34.83	25.63	51.40	52.57	61.48	70.59	71.80	71.54	72.18
10	25	23.23	33.48	11.72	19.67	22.78	39.50	40.36	43.33	44.55	44.95
	50	30.33	34.42	11.68	28.00	30.04	43.62	45.05	49.46	51.10	52.41
	75	34.73	34.60	13.15	33.43	35.30	45.43	47.60	52.87	54.68	56.24
	100	38.39	35.66	14.06	37.14	38.76	48.42	51.30	56.58	58.60	59.78
	300	48.43	36.81	19.86	48.04	48.93	57.40	62.95	66.77	68.06	68.99
	500	51.86	37.96	22.14	51.61	52.37	61.80	68.01	70.68	71.59	71.98
20	25	17.55	30.66	6.44	14.26	16.73	36.46	36.47	37.71	38.05	38.15
	50	24.56	35.30	3.87	22.83	24.01	42.38	42.52	44.82	45.81	46.19
	75	29.44	36.43	6.22	28.12	29.14	44.51	44.89	48.50	49.84	50.53
	100	33.33	37.95	7.19	32.33	33.37	46.82	47.75	51.76	53.16	53.94
	300	44.61	39.53	12.72	44.10	44.73	54.47	57.09	61.73	63.48	64.88
	500	48.96	39.45	15.71	48.63	49.15	57.97	61.84	65.99	67.75	68.71
Average		37.26	34.83	15.47	35.70	37.68	48.84	52.73	56.44	57.70	58.68

Table 4 Comparison between QATC and the other dispatching rules

m	*n*	EDD			EWDD			MDD			SLK			SLK/P		
		btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs
5	25	1053	2	195	1051	0	199	1095	2	153	1089	2	159	1081	2	167
	50	1060	5	185	1124	0	126	1080	5	165	1070	5	175	1063	5	182
	75	1034	28	188	1179	0	71	1062	28	160	1037	28	185	1030	28	192
	100	1037	42	171	1178	0	72	1061	42	147	1041	42	167	1037	42	171
	300	1009	103	138	1201	0	49	1026	103	121	1009	103	138	1006	103	141
	500	1010	136	104	1205	0	45	1026	136	88	1009	136	105	999	136	115
10	25	1129	0	121	983	0	267	1170	0	80	1145	0	105	1131	0	119
	50	1104	0	146	1079	0	171	1143	0	107	1101	0	149	1107	0	143
	75	1067	0	183	1142	0	108	1112	0	138	1066	0	184	1073	0	177
	100	1055	2	193	1144	0	106	1105	2	143	1057	2	191	1050	2	198
	300	1021	53	176	1221	0	29	1059	53	138	1015	53	182	1017	53	180
	500	1016	85	149	1226	0	24	1049	85	116	1023	85	142	1016	85	149
20	25	1178	0	72	942	1	307	1227	0	23	1199	0	51	1190	0	60
	50	1153	0	97	1045	0	205	1221	0	29	1164	0	86	1164	0	86
	75	1125	0	125	1082	0	168	1189	0	61	1123	0	127	1126	0	124
	100	1116	0	134	1119	0	131	1172	0	78	1118	0	132	1114	0	136
	300	1054	1	195	1208	0	42	1111	1	138	1059	1	190	1059	1	190
	500	1036	18	196	1225	0	25	1098	18	134	1042	18	190	1038	18	194
Average		1070	26	154	1131	0	119	1111	26	112	1076	26	148	1072	26	151

Table 4 (continued)

m	n	QWSPT			QWSLK_SPT			QWMDD			QAR		
		btr	eql	wrs	btr	eql	wrs	btr	eql	wrs	btr	eql	wrs
5	25	552	526	172	534	556	160	393	646	211	335	700	215
	50	645	483	122	580	505	165	493	555	202	423	590	237
	75	684	479	87	605	503	142	511	533	206	457	555	238
	100	657	489	104	584	516	150	504	536	210	462	548	240
	300	710	483	57	514	522	214	455	534	261	449	565	236
	500	710	470	70	499	527	224	458	540	252	409	613	228
10	25	405	663	182	414	676	160	314	774	162	226	869	155
	50	520	584	146	481	609	160	412	659	179	349	700	201
	75	579	537	134	563	546	141	463	610	177	389	642	219
	100	589	522	139	560	525	165	460	559	231	396	602	252
	300	658	533	59	571	542	137	507	548	195	469	549	232
	500	670	535	45	554	541	155	500	547	203	448	549	253
20	25	278	845	127	277	847	126	181	951	118	104	1059	87
	50	411	696	143	416	700	134	321	769	160	241	846	163
	75	468	642	140	468	646	136	379	680	191	305	735	210
	100	526	628	96	507	636	107	428	651	171	341	672	237
	300	631	546	73	608	546	96	534	548	168	484	553	213
	500	657	550	43	610	550	90	531	550	169	473	550	227
Average		575	567	108	519	583	148	436	622	193	376	661	214

btr, eql, wrs: number of times the QATC rule performed better, equal or worse, respectively, than each of the other procedures

Thus, heuristics that specifically take into account the quadratic nature of the objective function perform significantly better than their counterparts that were designed for a linear problem. As such, the linear dispatching rules will not be considered again in the remainder of this paper.

As previously mentioned, it was to be expected that the quadratic rules, which specifically take the squared nature of the objective function into account, would perform better than the linear rules. The results in Table 2, and the statistical tests, show that the difference in performance is quite large, and statistically significant. This is most relevant from a managerial point of view. Indeed, the results clearly show that, when dealing with a quadratic tardiness measure, managers should use a rule that is suitably adapted to such a measure, instead of simply relying on procedures developed for a linear problem.

We now compare the general dispatching heuristics and the quadratic rules. Table 3 gives the mean ivw of each procedure; ofv_{worst} is now the worst result among all the heuristics included in this table. In Table 4, we provide the number of times the QATC rule performed better (btr), equal (eql) or worse (wrs) than each of the other procedures. In both tables, the overall average (average) across all instances is also given.

The results in Table 3 show that, among the general rules, EDD, SLK and SLK/P perform better than EWDD and MDD. However, and as expected, the general rules are considerably outperformed by the quadratic dispatching heuristics. Once more, this result is useful for managers. Though simple and general rules may seem attractive, their performance is quite inferior to that of the specialized and more sophisticated quadratic procedures.

The QWSPT is inferior to the remaining quadratic rules. Again, this is to be expected, since the other quadratic procedures essentially use the QWSPT priority when a job is late (or on time), but then adjust this priority value as the slack of the job increases.

The overall best performance is provided by the QATC rule, closely followed by the QAR procedure. Indeed, the QATC heuristic not only provides the largest mean ivw, but also provides better results for a large number of instances.

A test was performed to determine if the differences between the QATC rule and each of the other procedures are statistically significant. As before, the non-parametric Wilcoxon signed-rank test was selected, and the significance level was set at 0.05. Again, this test was applied to each pair of heuristics tested (QATC versus each of the other procedures), and for each combination of the number of jobs n and the number of machines m. Holm's procedure was once more used to take into account the multiple comparisons.

The tests showed that the differences between the QATC heuristic and the general rules (EDD, EWDD, MDD, SLK and SLK/P) were always statistically significant. The identical performance hypothesis was also always rejected when comparing with the QWSPT and QWSLK_SPT heuristics. In what regards the QWMDD and QAR heuristics, the differences were not statistically significant in only about 11% and 22% of the cases, respectively. More detailed information about the statistical tests, and their results, are available in the electronic supplementary material.

The results in Tables 3 and 4, and the statistical tests, show that the QATC is the best performing dispatching heuristic, with the QAR rule not too far behind. Indeed, the results show that the QATC significantly outperforms all the other heuristics, with the exception of QAR, for most or all of the instance sizes. The QAR rule is not statistically different from QATC in about 22% of the problem sizes, but is still significantly outperformed in the remaining instances.

The results concerning the relative performance of the QATC and QAR rules are the opposite of those previously obtained for the single machine problem (Valente and Schaller 2012). Indeed, in the single machine environment the QAR rule was somewhat superior to QATC. This shows the importance of comparing procedures for each production environment, instead of simply assuming that the heuristic that performed best for another setting will also perform best in a different environment.

The heuristic procedures were all extremely efficient, and therefore a table with computational times is omitted, since most would be quite small. For the largest problem size (500 jobs and 20 machines), and on average, the general rules required less than 0.01 s to solve a problem. The QWSPT, QWSLK_SPT and QWMDD rules needed less than 0.02 s, while the QATC and QAR took about 0.04 s.

5.5 Comparison of the improvement procedures

Table 5 provides, for each improvement procedure, the mean relative improvement over the QATC rule (imp), as well as the number of times a better result is achieved (btr). The overall average (average) across all instances is also given.

The MS method is outperformed by the other improvement procedures. Indeed, not only does it provide the lowest relative improvement, but it also finds a better result a much lower number of times. The remaining procedures, on the other hand, manage to improve the QATC result on nearly all, and in some cases actually all, of the test instances.

In what regards the mean relative improvement imp, procedures with and without MS are quite close. That is, NEH and MS+NEH provide quite similar imp values, as do the pairs INS and MS+INS, and NEH+INS and MS+NEH+INS. The application of both NEH and INS provides a slightly higher imp than just using INS. Also, the mean relative improvement given by solely applying INS is slightly higher than that given by NEH alone.

For the MS procedure, the performance decreases clearly as the number of jobs increase. The effect of the number of machines is not as clear. The number of times a better solution is reached increases with the number of machines. However, the mean relative improvement decreases with the number of machines for the smallest number of jobs (25), but increases with m for the two largest job sizes (300 and 400), and increases then decreases for the remaining values of the number of jobs.

In what regards the other improvement procedures, the number of times a better solution is found increases with the number of machines. The mean relative improvement is usually first increasing, then decreasing, with the number of jobs. As the number of machines increases, the switching point seems to increase. That is,

Table 5 Improvement procedures comparison

m	n	MM		NEH		MM+NEH		INS		MM+INS		NEH+INS		MM+NEH+INS	
		imp	btr	imp	btr	imp	btr	imp	btr	imp	btr	imp	btr	imp	btr
5	25	13.34	909	45.58	1248	45.00	1248	50.54	1248	50.67	1248	50.90	1248	50.85	1248
	50	7.90	727	49.21	1245	48.61	1245	53.76	1245	53.73	1245	54.20	1245	54.11	1245
	75	5.31	603	48.14	1222	47.46	1222	51.88	1222	51.81	1222	52.41	1222	52.35	1222
	100	3.79	535	47.12	1208	46.47	1208	50.46	1208	50.40	1208	51.09	1208	50.98	1208
	300	0.62	209	39.79	1147	39.63	1147	41.58	1147	41.56	1147	42.63	1147	42.59	1147
	500	0.17	100	35.37	1114	35.34	1114	36.57	1114	36.55	1114	37.81	1114	37.79	1114
10	25	13.13	1086	39.71	1250	39.39	1250	45.06	1250	45.04	1250	45.47	1250	45.41	1250
	50	9.37	973	46.34	1250	46.00	1250	52.02	1250	52.00	1250	52.55	1250	52.50	1250
	75	7.67	858	48.75	1250	48.49	1250	53.91	1250	53.82	1250	54.71	1250	54.77	1250
	100	6.74	807	49.68	1248	49.36	1248	54.53	1248	54.46	1248	55.41	1248	55.37	1248
	300	2.55	555	46.13	1197	45.69	1197	48.86	1197	48.72	1197	50.16	1197	50.08	1197
	500	1.57	462	42.20	1165	41.81	1165	44.00	1165	43.90	1165	45.49	1165	45.43	1165
20	25	10.54	1167	29.92	1250	29.84	1250	33.95	1250	34.02	1250	34.34	1250	34.46	1250
	50	8.47	1104	37.96	1250	37.96	1250	43.45	1250	43.46	1250	44.20	1250	44.25	1250
	75	7.20	1045	41.91	1250	41.89	1250	47.47	1250	47.45	1250	48.43	1250	48.45	1250
	100	6.34	958	44.02	1250	44.01	1250	49.41	1250	49.35	1250	50.54	1250	50.50	1250
	300	3.42	761	48.44	1249	48.33	1249	52.31	1249	52.18	1249	54.03	1249	53.98	1249
	500	2.47	665	47.24	1232	47.06	1232	50.14	1232	50.00	1232	52.03	1232	51.98	1232
Average		6.15	751	43.75	1224	43.46	1224	47.77	1224	47.73	1224	48.69	1224	48.66	1224

imp: mean relative improvement over the QATC rule

btr: number of times an improvement procedure provides a result better than that of the QATC rule the number of jobs at which the mean relative improvement changes from increasing to decreasing is higher when there are more machines.

The effect of the number of machines on the mean relative improvement depends on the number of jobs. As n increases, the mean relative improvement switches from decreasing, to increasing then decreasing, to finally increasing, as the number of machines increases. It should be pointed out, however, that the mean relative improvement is always quite high. For instance, it is usually higher than 40% for the NEH and MS+NEH procedures, and quite often higher than 50% for the procedures that include INS.

A statistical test was performed to determine if the improvement provided by each of the improvement procedures, when compared with the QATC rule, is statistically significant. The Wilcoxon signed-rank test was selected, with a significance level of 0.05.

This test was applied to all pairs consisting of one improvement procedure and the QATC rule (e.g., MS vs. QATC, NEH+INS vs. QATC), and for each combination of number of jobs n and the number of machines m. Holm's procedure was once more used to take into account the multiple comparisons. All these comparisons were statistically significant. Thus, all the improvement procedures provide results that are significantly better than the QATC rule.

A second statistical test was performed in order to compare the improvement procedures among themselves. As usual, we conducted a Wilcoxon signed-rank test, with a significance level of 0.05. In this second test, we compared the following eight pairs of procedures: NEH versus MS; INS versus MS; INS versus NEH; MS+NEH versus NEH; MS+INS versus INS; NEH+INS versus NEH, NEH+INS versus INS and MS+NEH+INS versus NEH+INS. As before, the tests are applied for each combination of number of jobs n and the number of machines m, and Holm's procedure is used to take into account the multiple comparisons.

The tests showed that the differences in pairs NEH versus MS, INS versus MS, INS versus NEH and NEH+INS versus NEH are, with a single exception, statistically significant. Therefore, in what concerns the standalone methods, both NEH and INS are superior to MS, and INS outperforms NEH. The superiority of INS when compared with NEH is again stressed by the fact that NEH+INS is statistically better than NEH.

Procedure NEH+INS was also always statistically different from INS, so there are benefits in applying NEH in addition to INS. On the contrary, there seem to be little to no benefits in adding MS to the other methods. Indeed, MS+NEH was not statistically different from NEH in about 40% of the cases. Also, only about 22% and 17% of the cases yielded significant differences in the pairs MS+INS versus INS and MS+NEH+INS versus NEH+INS, respectively.

The results of the statistical tests therefore show that, in terms of solution quality, the procedures essentially can be divided in the following four ranks (in descending order of performance): (1) (MS+)NEH+INS; (2) (MS+)INS; (3) (MS+)NEH; (4) MS. Detailed information about these statistical tests, and their results, are available in the electronic supplementary material.

Table 6 provides the mean relative improvement over the QATC rule (imp), as well as the number of times a better result is achieved (btr), for each combination

Table 6 Improvement procedures comparison, for $n = 300$ and $m = 10$

T	*R*	MM		NEH		MM+NEH		INS	
		imp	btr	imp	btr	imp	btr	imp	btr
0.2	0.2	15.50	41	78.29	50	74.43	50	85.63	50
	0.4	2.42	10	94.13	50	93.92	50	99.59	50
	0.6	2.78	8	99.79	50	99.69	50	100.00	50
	0.8	0.00	0	80.00	40	80.00	40	80.00	40
	1.0	0.00	0	14.00	7	14.00	7	14.00	7
0.4	0.2	8.59	39	51.46	50	48.46	50	54.89	50
	0.4	3.98	25	59.84	50	58.31	50	64.86	50
	0.6	0.19	3	71.32	50	71.14	50	80.16	50
	0.8	0.02	1	88.32	50	88.23	50	97.70	50
	1.0	0.00	0	96.40	50	96.40	50	99.99	50
0.6	0.2	7.68	49	37.40	50	36.06	50	38.85	50
	0.4	3.94	32	39.43	50	39.45	50	41.95	50
	0.6	1.36	18	41.72	50	41.76	50	45.13	50
	0.8	0.46	4	41.17	50	41.04	50	46.42	50
	1.0	0.04	1	38.99	50	38.98	50	44.17	50
0.8	0.2	5.76	49	26.97	50	26.72	50	27.65	50
	0.4	2.66	41	26.81	50	26.69	50	27.71	50
	0.6	0.93	27	25.54	50	25.56	50	26.56	50
	0.8	1.02	23	25.36	50	25.23	50	26.50	50
	1.0	0.31	13	22.51	50	22.59	50	24.09	50
1.0	0.2	2.08	44	20.23	50	19.99	50	20.45	50
	0.4	1.80	40	20.38	50	20.36	50	20.59	50
	0.6	0.95	31	18.56	50	18.56	50	19.06	50
	0.8	0.66	27	17.92	50	17.98	50	18.32	50
	1.0	0.55	29	16.73	50	16.72	50	17.27	50
Average		2.55	22	46.13	48	45.69	48	48.86	48

T	*R*	MM+INS		NEH+INS		MM+NEH+INS	
		imp	btr	imp	btr	imp	btr
0.2	0.2	85.69	50	85.85	50	85.69	50
	0.4	99.59	50	99.58	50	99.58	50
	0.6	100.00	50	100.00	50	100.00	50
	0.8	80.00	40	80.00	40	80.00	40
	1.0	14.00	7	14.00	7	14.00	7
0.4	0.2	54.57	50	56.20	50	55.62	50
	0.4	64.62	50	66.37	50	65.98	50
	0.6	80.20	50	80.91	50	80.82	50
	0.8	97.70	50	97.68	50	97.68	50
	1.0	99.99	50	99.99	50	99.99	50

Table 6 (continued)

T	R	MM+INS		NEH+INS		MM+NEH+INS	
		imp	btr	imp	btr	imp	btr
0.6	0.2	38.39	50	40.64	50	40.28	50
	0.4	41.64	50	44.22	50	44.04	50
	0.6	44.83	50	48.20	50	48.07	50
	0.8	46.31	50	49.18	50	49.14	50
	1.0	44.18	50	46.28	50	46.28	50
0.8	0.2	27.49	50	29.34	50	29.35	50
	0.4	27.26	50	29.64	50	29.62	50
	0.6	26.17	50	28.63	50	28.68	50
	0.8	26.33	50	28.62	50	28.64	50
	1.0	24.01	50	26.03	50	25.94	50
1.0	0.2	20.26	50	21.82	50	21.71	50
	0.4	20.33	50	22.03	50	22.12	50
	0.6	19.04	50	20.40	50	20.40	50
	0.8	18.21	50	19.76	50	19.75	50
	1.0	17.23	50	18.69	50	18.69	50
Average		48.72	48	50.16	48	50.08	48

imp: mean relative improvement over the QATC rule

btr: number of times an improvement procedure provides a result better than that of the QATC rule

of the tardiness factor and the range of due dates parameters. The overall average (average) across all instances is also given. This table refers to instances with 300 jobs and 10 machines; nonetheless, results are similar for other instance sizes.

The mean relative improvement is usually decreasing with the tardiness factor T. We remark that the mean relative improvement can be quite large, and sometimes even equal to the maximum possible value of 100%, when the tardiness factor is low. These large relative improvements often correspond to relatively small absolute improvements. Indeed, few jobs will be tardy when T is low, resulting in relatively small objective function values. The improvement procedures can greatly reduce even further these small values, and often they even generate an optimal solution with no tardy jobs. The effect of the range of due dates R on the mean relative improvement, however, is not so clear-cut. Indeed, the mean relative improvement is often increasing in R when the tardiness factor is equal to 0.4 or 0.6, but decreasing when T is 0.8 or 1.0.

Table 7 provides the computational time (in seconds) required to run the improvement procedures, including the initial application of the QATC rule, when appropriate (i.e. when the procedure does not include MS). The overall average (average) across all instances is also given. Procedure MS is extremely efficient, requiring only a little over half a second for the largest instances in the problem set. The NEH and MS+NEH methods are also quite fast. In this regard,

Table 7 Computational times (in seconds)

m	*n*	MM	NEH	MM+NEH	INS	MM+INS	NEH+INS	MM+NEH+INS
5	100	0.0017	0.0034	0.0046	0.1135	0.1216	0.0844	0.0879
	300	0.0147	0.0650	0.0767	4.2868	4.2941	3.0392	3.1034
	500	0.0399	0.2878	0.3293	23.6987	23.5554	16.3991	16.1620
10	100	0.0064	0.0062	0.0118	0.2312	0.2197	0.1821	0.1800
	300	0.0500	0.1199	0.1734	7.6485	7.6100	5.9519	6.0529
	500	0.1355	0.5217	0.6811	43.1620	42.9692	31.7147	33.1299
20	100	0.0269	0.0157	0.0343	0.4086	0.4271	0.3266	0.3577
	300	0.2101	0.2538	0.4538	14.1708	14.3808	12.2796	12.3444
	500	0.5664	1.0774	1.6752	85.0410	86.6824	74.7569	74.1412
Average		0.1168	0.2612	0.3822	19.8624	20.0289	16.0816	16.1733

NEH is more computationally efficient than MS+NEH, particularly for instances with a larger number of machines.

The four improvement procedures that include the insertions local search require a much higher computational time. In the context of these four procedures, it should first be remarked that the presence of MS changes runtimes only very slightly. Indeed, the times required by INS and MS+INS, or by NEH+INS and MS+NEH+INS, are quite similar.

However, and secondly, the inclusion of NEH reduces the computational effort. In fact, NEH+INS is faster than INS alone, and the same is true when comparing MS+NEH+INS with MS+INS. Therefore, the time required by the initial application of NEH is more than offset by the computational savings it allows when performing the insertion local search.

The results regarding the mean relative improvement and the computation times can be combined to derive the non-dominated methods. Indeed, MS emerges as the fastest procedure, but also the one which provides the smallest improvement. Next, NEH is the second fastest, and its improvement is superior to that of MS. Finally, NEH+INS (and also MS+NEH+INS) provides the largest improvement, but is also slower than NEH or MS. These results provide a guide to decision makers on the method of choice. Indeed, as the time available to generate a solution increases, the decision maker can switch from QATC, to MS, then NEH, and finally NEH+INS.

Improvement procedures MS+NEH (slower than NEH) and INS and MS+INS (inferior to NEH+INS and MS+NEH+INS in both solution quality and runtime), on the other hand, are dominated. As such, in the next subsection, we will only consider the improvement procedures MS, NEH and NEH+INS, as well as the QATC rule without any improvement.

5.6 Comparison with optimal results

The comparison with optimal results is conducted on another problem set, containing instances with only 8, 10 and 12 jobs. These instances were otherwise generated as previously described in Subsection 4.1, and 50 problems were created for each

Table 8 Comparison with optimum results

m	n	ivh				n_opt			
		QATC	MM	NEH	NEH+INS	QATC	MM	NEH	NEH+INS
5	8	34.05	20.93	6.67	1.68	13	41	453	935
	10	39.22	27.49	10.24	2.87	2	8	225	732
	12	42.38	32.57	13.33	4.54	1	1	124	552
10	8	25.82	14.68	3.69	0.75	11	41	473	967
	10	30.37	19.24	5.62	1.48	2	4	252	762
	12	33.62	23.30	8.17	2.20	0	0	115	533
20	8	19.89	10.84	2.01	0.50	4	38	501	950
	10	21.96	13.31	3.03	0.80	1	4	255	750
	12	25.35	16.29	4.15	1.26	0	0	143	529
Average		30.29	19.85	6.32	1.79	4	15	282	746

ivh: mean relative improvement provided by the optimum over a heuristic procedure

n_opt: number of times a heuristic finds the optimal solution

combination of n, m, T and R. The optimal solutions were calculated using complete enumeration. As previously mentioned, only the QATC dispatching rule, as well as the non-dominated improvement procedures MS, NEH and NEH+INS, are compared with optimal solutions.

Table 8 gives the mean relative improvement provided by the optimum over each heuristic approach (ivh), as well as the number of times each procedure finds an optimal solution (n_opt). The overall average (average) across all instances is also given.

The performance of the QATC rule is somewhat poor. Indeed, this rule rarely finds the optimal solution, and its results are usually 20% to 30% above the optimum for instances with 10 and 20 machines. For instances with 5 machines, this deviation increases to close to 40%. The MS procedure provides better results, but it is still around 10% to 30% above the optimum.

The performance of the NEH method is much better. Indeed, for instances with 20 and 10 machines, the mean relative improvement given by the optimum is under about 4% and 8%, respectively. For instances with 5 machines, however, that mean relative improvement can exceed 10%.

The NEH+INS procedure provides very good results. Indeed, it achieves an optimal solution in about 40% to 75% of the instances. Also, the mean relative improvement given by the optimum is below about 4.5%, and for some instance sizes below 1%.

The performance of the procedures, when compared with the optimal results, somewhat degrades as the number of jobs increases. However, and particularly in what regards the mean relative improvement provided by the optimum, the performance greatly improves as the number of machines increases. Indeed, all the procedures, including the QATC rule without any improvement method, are closer to the optimal solution when the number of machines is higher.

Table 9 Comparison with optimum results, for $n = 10$ and $m = 10$

T	*R*	ivh				n_opt			
		QATC	MM	NEH	NEH+INS	QATC	MM	NEH	NEH+INS
0.2	0.2	57.69	36.33	15.87	5.07	0	0	5	22
	0.4	58.25	38.57	13.81	2.95	0	0	4	31
	0.6	58.80	45.55	14.28	5.49	0	0	12	29
	0.8	56.67	49.53	15.76	2.77	0	0	8	35
	1.0	46.60	35.69	9.12	0.93	0	0	13	38
0.4	0.2	40.01	22.18	6.84	1.85	0	0	9	30
	0.4	40.53	23.00	5.87	2.09	0	0	9	28
	0.6	30.44	19.52	6.83	2.15	0	0	9	27
	0.8	30.62	19.86	5.42	1.56	0	0	13	29
	1.0	30.49	21.52	4.72	0.93	0	0	12	35
0.6	0.2	31.39	14.85	3.55	1.07	0	0	11	31
	0.4	30.16	14.73	3.92	1.23	0	1	6	29
	0.6	24.20	14.29	2.95	0.44	0	0	15	38
	0.8	22.54	13.53	4.30	1.27	0	0	8	27
	1.0	19.93	13.08	3.32	0.90	0	0	7	24
0.8	0.2	20.96	10.13	3.56	0.71	0	0	10	33
	0.4	20.84	10.78	2.57	0.67	0	0	8	29
	0.6	19.54	11.77	2.96	0.63	0	0	10	34
	0.8	18.47	9.83	1.86	0.74	0	0	16	32
	1.0	16.73	11.16	1.93	0.61	0	0	12	31
1.0	0.2	18.06	8.62	2.80	0.69	0	1	15	33
	0.4	17.61	9.46	2.20	0.37	0	0	8	28
	0.6	17.42	9.40	2.17	0.44	2	2	10	33
	0.8	16.30	8.50	1.91	0.80	0	0	11	28
	1.0	15.11	9.06	1.97	0.57	0	0	11	28
Average		30.37	19.24	5.62	1.48	0	0	10	30

ivh: mean relative improvement provided by the optimum over a heuristic procedure

n_opt: number of times a heuristic finds the optimal solution

The results obtained through the comparison with optimal solutions, and though it must be remarked that the instance sizes are quite distinct, are somewhat in line with those described when analyzing the improvement procedures. For instance, the QATC rule is sometimes up to 40% above the optimum results, while the NEH+INS improvement procedure is quite close to optimality. This is in line with the close to 50% relative improvement that the NEH+INS procedure provides for many of the instance sizes on which the improvement methods were analyzed. Again, however, we stress that the sizes of the instances used in the comparison with optimal results, and in the analysis of the improvement procedures, are very different.

The NEH+INS method can provide very good results in relatively short computational times. For quite large instances, in which this procedure may take excessive time, the NEH improvement method can still provide reasonable quality solutions in an adequate computational time.

Table 9 provides the mean relative improvement provided by the optimum (ivh), as well as the number of times an optimal solution is found (n_opt), for each combination of the tardiness factor and the range of due dates parameters. The overall average (average) across all instances is also given. This table refers to instances with 10 jobs and 10 machines; however, results are similar for other instance sizes.

The performance of the procedures nearly always improves as the tardiness factor T increases. The effect of the range of due dates R on the mean relative improvement given by the optimum, however, is not so clear. The performance of the MS method tends to improve as the range of due dates increases. For the other procedures, however, this effect is not always present.

6 Conclusion

In this paper, we considered the permutation flowshop scheduling problem with a weighted squared tardiness objective function. Our focus was on identifying efficient procedures, capable of quickly solving even medium or large instances.

In this context, we presented several dispatching heuristics. These included general rules, heuristics developed for the linear problem, and heuristics suitably adapted to take the squared objective into account. Most of these rules were previously used in other production settings, and they were suitably adapted to the flowshop environment. Also, preliminary experiments were performed to determine an adequate value for a parameter required by some of the heuristics.

Then, we presented several improvement procedures, which use one or more of three techniques: multiple sequence dispatching (MS), the widely used NEH procedure, and insertions-based local search (INS). These procedures were applied to the best performing dispatching rule (QATC).

The results showed that the quadratic heuristics clearly outperform their linear counterparts, as well as the general rules. Therefore, when dealing with a squared tardiness environment, it is important to use rules that specifically address the quadratic nature of the objective function, instead of relying on simple and general rules, or on procedures developed for the linear problem.

The computational results additionally showed that all improvement procedures significantly improve upon the QATC solution. The MS, NEH and NEH+INS procedures emerge as non-dominated, when taking both solution quality and runtime into account. The QATC, MS and NEH procedures are quite efficient, and can be applied to even quite large instances. The NEH+INS method can provide somewhat higher quality solutions, but its computational requirements may be excessive for very large instances. The results thereby provide a guide to decision makers on the method of choice (dispatching rule and/different improvement procedures), given the time available to obtain a solution.

Future research may involve the development of lower bounds and branch-and-bound algorithms, able to provide optimum solutions more efficiently than complete enumeration. Another possibility is to consider metaheuristics, incorporating the QATC rule and the NEH or NEH+INS improvement procedure, in order to try to achieve higher quality solutions. Finally, the addition of features such as release dates or setup times is yet another option for further work.

Acknowledgements The authors would like to thank the two anonymous referees, and the associate editor, for their helpful comments and suggestions, which have improved this paper.

References

Alidaee B, Ramakrishnan KR (1996) A computational experiment of COVERT-AU class of rules for single machine tardiness scheduling problem. Comput Ind Eng 30:201–209

Baker KR, Bertrand JWM (1982) A dynamic priority rule for scheduling against due-dates. J Op Manage 3:37–42

Dalfard VM, Ardakani A, Banihashemi TN (2011) Hybrid genetic algorithm for assembly flow-shop scheduling problem with sequence-dependent setup and transportation times. Tehnicki Vjesnik 18:467–504

Fernandez-Viagas V, Framinan JM (2015a) NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness. Comput Op Res 60:27–36

Fernandez-Viagas V, Framinan JM (2015b) A new set of high-performing heuristics to minimise flowtime in permutation flowshops. Comput Op Res 53:68–80

Fernandez-Viagas V, Leisten R, Framinan JM (2016) A computational evaluation of constructive and improvement heuristics for the blocking flow shop to minimise total flowtime. Expert Syst Appl 61:290–301

Fernandez-Viagas V, Ruiz R, Framinan JM (2017) A new vision of approximate methods for the permutation flowshop to minimise makespan: state-of-the-art and computational evaluation. Eur J Op Res 257:707–721

Fernandez-Viagas V, Valente JMS, Framinan JM (2018) Iterated-greedy-based algorithms with beam search initialization for the permutation flowshop to minimise total tardiness. Expert Syst Appl 94:58–69

Framinan JM, Gupta JND, Leisten R (2004) A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. J Op Res Soc 55:1243–1255

Gonçalves TC, Valente JMS, Schaller JE (2016) Metaheuristics for the single machine weighted quadratic tardiness scheduling problem. Comput Op Res 70:115–126

Hasija S, Rajendran C (2004) Scheduling in flowshops to minimize total tardiness of jobs. Int J Prod Res 42:2289–2301

Hoitomt DJ, Luh PB, Max E, Pattipati KR (1990) Scheduling jobs with simple precedence constraints on parallel machines. IEEE Control Syst Mag 10:34–40

Holsenback JE, Russell RM, Markland RE, Philipoom PR (1999) An improved heuristic for the single-machine, weighted-tardiness problem. Omega 27:485–495

Jackson JR (1955) Scheduling a production line to minimize maximum tardiness, Management Science Research Project, Research Report 43. University of California, Los Angeles

Kanet JJ, Li XM (2004) A weighted modified due date rule for sequencing to minimize weighted tardiness. J Sched 7:261–276

Karabulut K (2016) A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops. Comput Ind Eng 98:300–307

Luh PB, Hoitomt DJ (1993) Scheduling of manufacturing systems using the Lagrangian relaxation technique. IEEE Trans Autom Control 38:1066–1079

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

Neufeld JS, Gupta JND, Buscher U (2016) A comprehensive review of flowshop group scheduling literature. Comput Op Res 70:56–74

Osman IH, 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. In: Paper presented at the MISTA 2009—multidisciplinary international conference on scheduling: theory and applications, Dublin, Ireland
Ow PS, Morton TE (1988) Filtered beam search in scheduling. Int J Prod Res 26:35–62
Panwalkar SS, Iskander W (1977) Survey of scheduling rules. Op Res 25:45–61
Potts CN, van Wassenhove LN (1991) Single-machine tardiness sequencing heuristics. IIE Trans 23:346–354
Reza Hejazi S, Saghafian S (2005) Flowshop-scheduling problems with makespan criterion: a review. Int J Prod Res 43:2895–2929
Ruiz R, Maroto C (2005) A comprehensive review and evaluation of permutation flowshop heuristics. Eur J Op Res 165:479–494
Ruiz R, Stützle T (2008) An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. Eur J Op Res 187:1143–1159
Schaller J, Valente JMS (2012) Minimizing the weighted sum of squared tardiness on a single machine. Comput Op Res 39:919–928
Schaller J, Valente JMS (2018) Efficient heuristics for minimizing weighted sum of squared tardiness on identical parallel machines. Comput Ind Eng 119:146–156
Smith WE (1956) Various optimizers for single-stage production. Naval Res Logist Q 3:59–66
Sun XQ, Noble JS, Klein CM (1999) Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness. IIE Trans 31:113–124
Sun Y, Zhang C, Gao L, Wang X (2011) Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects. Int J Adv Manuf Technol 55:723–739
Taguchi G (1986) Introduction to quality engineering: designing quality into products and processes. Asian Productivity Organization, Tokio
Taillard E (1993) Benchmarks for basic scheduling problems. Eur J Op Res 64:278–285
Thomalla CS (2001) Job shop scheduling with alternative process plans. Int J Prod Econ 74:125–134
Valente JMS, Alves RAFS (2008) Heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. Comput Op Res 35:3696–3713
Valente JMS, Schaller JE (2012) Dispatching heuristics for the single machine weighted quadratic tardiness scheduling problem. Comput Op Res 39:2223–2231
Vallada E, Ruiz R (2010) Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. Omega 38:57–67
Vallada E, Ruiz R, Minella G (2008) Minimising total tardiness in the m-machine flowshop problem: a review and evaluation of heuristics and metaheuristics. Comput Op Res 35:1350–1373
Vepsalainen APJ, Morton TE (1987) Priority rules for job shops with weighted tardiness costs. Manage Sci 33:1035–1047
Volgenant A, Teerhuis E (1999) Improved heuristics for the n-job single-machine weighted tardiness problem. Comput Op Res 26:35–44

Maria Raquel C. Costa received her MS in data analytics from Faculdade de Economia, Universidade do Porto, Portugal, in 2015. She currently works at the Information Systems Department of NORS.

Jorge M. S. Valente received his PhD in management science from the Faculdade de Economia, Universidade do Porto, Portugal, in 2005. He is an associate professor at Faculdade de Economia, Universidade do Porto, Portugal. His main research interests consist in the application of exact and heuristic algorithms to scheduling problems.

Jeffrey E. Schaller received his PhD in operations management from the College of Business Administration, University of Florida, USA, in 1996. He held various operations management positions in several companies, and is currently a full professor at the Department of Business Administration, Eastern Connecticut State University, USA. His main research interests include scheduling, lean production and cellular manufacturing.

Affiliations

Maria Raquel C. Costa[1] · Jorge M. S. Valente[2] · Jeffrey E. Schaller[3]

Maria Raquel C. Costa
mcosta@nors.com

Jeffrey E. Schaller
schallerj@easternct.edu

[1] Information Systems Department, NORS, Rua Manuel Pinto de Azevedo 711, 1º, 4149-010 Porto, Portugal

[2] LIAAD – INESC TEC, Faculdade de Economia da Universidade do Porto, Rua Dr. Roberto Frias, s/n, 4200-464 Porto, Portugal

[3] Department of Business Administration, Eastern Connecticut State University, 83 Windham St., Willimantic, CT 06226-2295, USA