Decision Analytics Journal 10 (2024) 100432

Contents lists available at ScienceDirect

Decision Analytics Journal

journal homepage: www.elsevier.com/locate/dajour

A new dispatching mechanism for parallel-machine scheduling with different efficiencies and sequence-dependent setup times

Gen-Han Wu [a], Pourya Pourhejazy [b,*], Wang-Xian Li [c], Tai-Hsi Wu [d]

[a] Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan 32003, Taiwan
[b] Department of Industrial Engineering, UiT- The Arctic University of Norway, Lodve Langesgate 2, Narvik 8514, Norway
[c] Graduate Institute of Logistics Management, National Dong Hwa University, Hualien 97401, Taiwan
[d] Department of Business Administration, National Taipei University, Taipei 237303, Taiwan

ARTICLE INFO

Keywords:
Production planning
Apparent tardiness cost
Parallel machines
Dispatching
Optimization
Sequence-dependent setup

ABSTRACT

The Apparent Tardiness Cost (ATC) dispatching rule was initially developed to minimize tardiness in single-machine scheduling problems. ATC extensions have been frequently applied in other production settings, relying heavily on blocking idle machine capacity with a single-machine outlook; this approach may not result in the best outcomes, considering that machines have different efficiencies. This study develops a new dispatching rule for parallel-machine scheduling, considering different machine efficiencies, ready times, and sequence-dependent setup times to minimize the total weighted tardiness. The proposed method reduces the time interference factor of the denominator item in the dispatching rule and uses more effective methods for selecting the best processing machine for the jobs. The grid approach is used to evaluate the method against the state-of-the-art. The experimental results confirm that the developed method is superior regardless of the type of parallel machines, the problem scale, and other operational parameters. It is also shown that other ATC dispatching rules can be improved by applying the proposed approach. The proposed method could be incorporated into soft computing techniques for more effective and efficient scheduling.

1. Introduction

With the proliferation of supply alternatives and rising customer expectations, corporations ought to alleviate delays and fulfill orders in the shortest time possible to survive the competition. Production scheduling ensures that orders are processed timely and that the desired performance indicators are at their best. While enumerative and exact optimization approaches provide guaranteed best solutions, they are computationally intensive and are not practical for solving industry-scale problems. Dispatching rules are applicable alternatives for scheduling jobs on the shop floor with high workloads [1] and dynamic situations [2], which are also efficient construction heuristics [3]; the Earliest Due Date (EDD), Shortest Processing Time (SPT), Weighted Shortest Processing Time (WSPT), Least Slack (LS), and Critical Ratio (CR) are some of the most prominent dispatching methods [4]. Despite their merits, these methods are myopic; composite methods are developed to integrate the advantages of various dispatching rules and adjust the dispatching outlook based on operational requirements [5]. From the existing composite dispatching tools for minimizing tardiness in production scheduling, the Apparent Tardiness Cost (ATC) has received wide recognition [6].

Developed by Vepsalainen and Morton [7], ATC differs from its earlier counterparts in that it accounts for the ready time and due date characteristics, and can adjust its features considering the operational situation. As a prime example, a conversion to the weighted shortest WSPT or LS rules would be beneficial if scheduling jobs with a tight due date is sought. ATC has found its way into different production environments while still applying the dispatching logic applied for a single-machine configuration; that is, selecting the earliest ready machine for processing the pending job. Although the method can still prevent machine idleness, it does not guarantee an overall best performance, particularly because machines differ in their efficiency. Moreover, the existing studies have made unrealistic assumptions to reduce the complexity of the method. For example, it is often assumed that all jobs and machines are ready, the process can begin at any moment, and that different jobs can be processed without machine preparations.

From the most relevant developments in the ATC literature, setup and idle times, as the prime examples of non-value-adding activities [8], have been incorporated into the formulation. Ow and Morton [9] for the first time included the Sequence-Dependent Setup Times (SDST) into the ATC dispatching rule to find the Minimized Apparent

* Corresponding author.
E-mail addresses: genhanwu@saturn.yzu.edu.tw (G.-H. Wu), pourya.pourhejazy@uit.no (P. Pourhejazy), lmes882092000@gmail.com (W.-X. Li), taiwu@mail.ntpu.edu.tw (T.-H. Wu).

https://doi.org/10.1016/j.dajour.2024.100432
Received 14 November 2023; Received in revised form 7 February 2024; Accepted 19 February 2024
Available online 22 February 2024
2772-6622/© 2024 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Early/Tardy Costs (MATC). The solution quality of the MATC dispatching rule was proven to be better than the original ATC dispatching rule. Later, Lee et al. [10] introduced the Apparent Tardiness Cost with Setups (ATCS) dispatching rule by defining the SDSTs as the exponential function; they showed that ATCS performed better than MATC. These formulators did not account for the possibility of jobs arriving at the production system at different times. Logendran and Subur [11] proposed the Apparent Tardiness Cost with Ready times (ATCR) dispatching rule in combination with tabu search to solve unrelated parallel machine scheduling problems. Considering that jobs are subject to SDSTs and have different ready times, Pfund et al. [12] proposed the Apparent Tardiness Cost with Setups and Ready Times (ATCSR) dispatching rule and demonstrated that their rule is more effective in solving identical parallel machine scheduling problems relative to the ATCS, the Apparent Tardiness Cost with Setups considering single Batch (BATCS; [13]), and Multiple Batch (MBATCS) variants. Xi and Jang [14] considered the situation where jobs have SDSTs and different ready times. Suggesting that setups can be done (1) with job processing (implemented in advance) or (2) after jobs are ready at the production system, Xi and Jang [14] extended the ATCSR models considering Continuous and Separable Setups and named them MATCSR and ATCSSR dispatching rules, respectively; they showed that both rules outperform the ATCSR dispatching rule in solving identical parallel machine scheduling problems. In a more recent development, Xi and Jang [15] analyzed their ATC under the conditions of continuous and separable job setup time and proposed the ATCRCS and the ATCRSS dispatching rules, respectively. They compared the new dispatching rules with ATCSR for single-machine scheduling and demonstrated that their method further reduced the total tardiness. Most recent studies extended the basic ATC considering other real-world requirements, like machine eligibility restrictions and the use of a common server [16].

The existing studies extended and/or applied ATC based on the idea of minimizing delays with a single-machine scheduling $1|r_j|\sum w_jT_j$ outlook which has been proven to be an *NP*-hard problem [17]. Dispatching jobs in the parallel-machine production $P|r_j|\sum w_jT_j$ or $R|r_j|\sum w_jT_j$ environment, however, requires a collective outlook considering that the machines have different efficiencies. To the best of the authors' knowledge, this feature has not been addressed in the scheduling literature. To address this research gap, the present study formulates a novel dispatching method that simultaneously accounts for the SDSTs and different ready times while accounting for machines with different efficiencies. We expect our method to perform better than the state-of-the-art in increasing profit and preventing delays in job delivery.

The remainder of this manuscript begins with an exhaustive review of the literature in Section 2. Next, the developed method is described in detail in Section 3. Section 4 presents the numerical experiments and provides insights into the dispatching procedure. Finally, the study is concluded in Section 5, where the major findings and suggestions for future research are provided.

2. Literature review

This section reviews the published literature on the ATC dispatching rule, where the method is either extended or used as a benchmark algorithm. We first provide an overview of the literature considering the production setting, technical features, objective functions, and the sort of ATC developed/applied in each study investigated. The section is then concluded by a critical review of the relevant literature and highlighting the research gap.

2.1. Early studies

The early version of the ATC dispatching rule was developed using a combination of the WSPT and LS dispatching rules to integrate their features. The WSPT rule is generally used to deal with tight due dates or with delivery delays applying simple dispatching rules. To reduce tardiness penalties in such conditions, priority jobs can be scheduled by measuring the weights and processing time of all jobs. The LS rule is mostly employed for relatively loose due dates. In this method, the jobs with longer slack time should be postponed, whereas those with a short slack time can be processed as a greater priority for reducing the possibility of delivery delays. In so doing, the ATC dispatching rule can determine the optimal job in both tight and loose schedules, which results in minimizing total tardiness and maximizing revenue while considering job cost and profitability.

ATC was later evolved based on the Cost OVER Time (COVERT) rule, which was formulated by $I_j\ (t) = 1/P_j\ * I_j$ to demonstrate the priority of jobs, where greater values show a higher processing priority. In this formulation, $C_j = ((\overline{p} - max(d_j - p_j - t, 0))/\overline{p})$ refers to the job weight, where $\overline{p}$ denotes the mean expected processing time for awaiting jobs, and $max(d_j - p_j - t, 0)$ is the remaining slack time with d_j, p_j, and t being the job due date, required job processing time, and current available processing time, respectively. When the job has no remaining slack time, i.e., $C_j = 1$, the COVERT rule changes to $I_j(t) = 1/p_j$ and the jobs with a shorter processing time are scheduled at a higher priority. When the job has slack time, both the mean estimated job completion time for awaiting jobs and the remaining slack time should be calculated. In this situation, a longer remaining slack time results in a smaller weighted value, C_j. Considering $I_j(t) = C_j/p_j$, jobs with looser due dates are postponed, whereas those with tighter dues are scheduled at higher priorities. The COVERT rule performed well in measuring indicators associated with tardiness. Vepsalainen and Morton [7] modified the COVERT model and proposed the ATC dispatching rule; they showed that ATC performs better than COVERT, WSPT, SPT, and EDD in situations with a greater number of orders and where temporary order insertions are allowed.

The ATC dispatching rule has been used and extended for nearly 30 years building on these seminal works. WSPT, LS, the Shortest Setup Time (SST), and the Shortest Ready Time (SRT) are among the major modules considered in these studies, which can be represented in Eq. (1). An exhaustive list of these methods is summarized in Table 1 followed by an elaboration on their developments.

$$I_j(t) = (WSPT\ term) \times (LS\ term) \times (SST\ term) \times (SRT\ term) \quad (1)$$

The ATC rules have evolved in three major directions. In the first direction, learning-based approaches were developed for determining the ATC's lookahead parameters. Kim et al. [18] used neural networks to predict the best values for the look-ahead parameters of ATC. Valente [19] introduced generalized approaches based on mapping instance statistics into a unique value and using the characteristics of the current workload for adjusting the lookahead parameter of the ATCs. For the same purpose, Min and Kim [20] developed a reinforcement learning-based approach based on the job data; they considered unequal job arrival times in a single-machine production environment. Chen et al. [21] introduced a unified multi-phase approach for robust scaling of such ATC parameters along with other dispatching rules.

The second stream of studies employed ATC as a complimentary tool. As a prime example, ATC was used as a solution initialization module in soft computing techniques, for example, single-machine scheduling with SDSTs [22]. Shin et al. [23] incorporated the ATC with setup consideration into the Tabu Search algorithm to solve a single machine while accounting for release times and due dates in addition to SDSTs. ATC was also used for scheduling in other contexts. The problem of scheduling in a flow-line-based manufacturing cell was approached using ATC in the simulation-based optimization approach developed by Bengu [24]. Kang et al. [25] investigated the problem of reentrant flowshop with SDSTs using revised ATC with setups (RATCS). ATC is also used for rescheduling problems considering machine breakdowns and total completion time as the optimization objective [26]. Several studies used the basic ATC in application areas

Table 1
Summary of the ATC dispatching rules in the literature.

Method	Priority index			
	WSPT	LS	SST	SRT
ATC (1987)	$\frac{w_j}{p_j}$	$exp(-\frac{max(d_j-p_j-t,0)}{k_1\bar{p}})$	NA	NA
MATC (1989)	$\frac{w_j}{p_j+s_{ij}}$	$exp(-\frac{max(d_j-p_j-s_{ij}-t,0)}{k_1\bar{p}})$	NA	NA
ATCS (1997)	$\frac{w_j}{p_j}$	$exp(-\frac{max(d_j-p_j-t,0)}{k_1\bar{p}})$	$exp(-\frac{s_{ij}}{k_2\bar{s}})$	NA
ATCR (2004)	$\frac{w_j}{p_j^m}$	$exp(-\frac{max(d_j-p_j^m-t_m,0)}{k_1\bar{p}^m})$	NA	$exp(-\frac{max(r_j-t,0)}{k_2\bar{r}})$
BATCS (2008)	$\frac{w_j}{p_j}$	$exp(-\frac{max(d_j-p_j+r_j-t,0)}{k_1\bar{p}})$	$exp(-\frac{s_{ij}}{k_2\bar{s}})$	NA
MBATCS (2008)	$\frac{w_j}{p_j}$	$exp(-\frac{max(d_j-p_j+max(r_j-t,0),0)}{k_1\bar{p}})$	$exp(-\frac{s_{ij}}{k_2\bar{s}})$	NA
ATCSR (2008)	$\frac{w_j}{p_j}$	$exp\left(-\frac{max(d_j-p_j-max(r_j,t),0)}{k_1\bar{p}}\right)$	$exp\left(-\frac{s_{ij}}{k_2\bar{s}}\right)$	$exp(-\frac{max(r_j-t,0)}{k_3\bar{p}})$
MATCSR (2012)	$\frac{w_j}{p_j+s_{ij}+max(r_j-t,0)}$	$exp\left(-\frac{max(d_j-p_j-s_{ij}-max(r_j,t),0)}{k_1\bar{p}}\right)$	$exp\left(-\frac{s_{ij}}{k_2\bar{s}}\right)$	$exp\left(-\frac{max(r_j-t,0)}{k_3\bar{p}}\right)$
ATCSSR (2012)	$\frac{w_j}{p_j+max(s_{ij},r_j-t)}$	$exp\left(-\frac{max(d_j-p_j-max(r_j,t+s_{ij}),0)}{k_1\bar{p}}\right)$	$exp\left(-\frac{s_{ij}}{k_2\bar{s}}\right)$	$exp\left(-\frac{max(r_j-t-s_{ij},0)}{k_3\bar{p}}\right)$
ATCRCS (2013)	$\frac{w_j}{p_j+s_{ij}+max(r_j-t,0)}$	$exp\left(-\frac{max(d_j-p_j-t,0)}{k_1(\bar{p}+\bar{s})}\right)$	$exp\left(-\frac{s_{ij}}{k_2\bar{s}}\right)$	$exp\left(-\frac{max(r_j-t,0)}{k_3\bar{p}}\right)$
ATCRSS (2013)	$\frac{w_j}{p_j+max(s_{ij},r_j-t)}$	$exp\left(-\frac{max(d_j-p_j-t,0)}{k_1(\bar{p}+\bar{s})}\right)$	$exp\left(-\frac{s_{ij}}{k_2\bar{s}}\right)$	$exp\left(-\frac{max(r_j-t,0)}{k_3\bar{p}}\right)$

$I_j(t)$ represents the priority of a job j, with a greater value indicating a higher processing priority; t, current available processing time, w_j, job weight; d_j, due date; p_j, job processing time; s_{ij}, sequence-dependent setup time; r_j, time required for jobs to reach the production system; $\bar{p}$, mean processing time for awaiting jobs; $\bar{s}$, mean setup time for awaiting jobs; $\bar{r}$, mean ready time for awaiting jobs; k_1, k_2, and k_3, adjustable parameters of the dispatching rule.

other than production, like improving the quality of service on web-based systems [27] and grid computing systems [28]. ATC is also used as a benchmark to evaluate the effectiveness of other dispatching rules, like in single-machine [29–32], flowshop [33,34], job shop [35], flexible job shop [36], open job shop [37], dynamic workshops [38], and general scheduling situations [39].

The third stream of the ATC-related research works extended the ATC dispatching rule's formulation to address real-world and practical features; the present study belongs to this category. From the early works, Ow and Morton [9] incorporated SDSTs into the ATC dispatching rule and named the modified ATC dispatching rule the MATC dispatching rule. The solution quality of the MATC dispatching rule, in which s_{lj} is included in the WSPT and the LS rules, was proven to be better than that obtained by the original ATC dispatching rule that dismissed SDSTs. Lee et al. [10] later introduced the ATCS dispatching rule by defining the SDST as the exponential function in the ATC dispatching rule. The SDST (s_{lj}) was also considered in the MATC and ATCS dispatching rules, but ATCS performed better than MATC. Abdallah and Jang [40] extended the ATC rule for addressing sequence-dependent family setup times.

As the seminal ATC-based dispatching rules, MATC and ATCS include SDSTs but do not account for the possibility of jobs arriving at different times. To address this limitation, Logendran and Subur [11] proposed the ATCR dispatching rule in combination with tabu search to solve unrelated parallel machine scheduling problems. Considering that jobs require setups and may have different ready times, Pfund et al. [12] proposed the ATCSR dispatching rule, and demonstrated that it is more effective than ATCS, BATCS, and MBATCS dispatching rules for solving identical parallel machine scheduling problems. Xi and Jang [14] considered SDST and different ready times for every job. They suggested SDSTs should be divided into (1) situations where setups can be separated with job processing and implemented in advance and (2) situations where setups have to proceed after jobs are ready at the production stage. They developed the MATCSR and ATCSSR dispatching rules to address these two types of setup times and demonstrated that their method performed better than the ATCSR for solving identical parallel machine scheduling problems. In their subsequent work, Xi and Jang [15] developed two variants of the basic ATC that consider ready time and continuous/separable setups for solving single-machine scheduling problems; introducing the ATCRCS and ATCRSS dispatching rules for solving single-machine scheduling problems, they demonstrated a reduction of more than 5.1 percent of total tardiness compared to ATCSR. The most recent developments are reviewed next.

2.2. Most recent developments

Salama et al. [41] proposed genetic programming for generating dispatching rules for job shop scheduling problems; the objective was to reduce the computation time by using a Support Vector Machine classifier. A classifier module, which is trained with data from the first generation of rules, using a new representation, was used to filter out the least-performing dispatching rules in the next generations. Kasper et al. [42] proposed a novel method for production control in high-variety manufacturing. The method selects the next job based on its global influence in terms of multiple objectives. This differs from the traditional dispatching rules that consider hierarchical methods for order release and simple priority rules for order dispatching. Wang et al. [43] presented a new approach to tackle dynamic job shop scheduling problems with minimal assumptions and high uncertainty; they combined data-driven simulation, machine learning, and evolutionary algorithm to create new dispatching rules. The authors also tested the use of XGBoost, instead of the Total Work Content (TWK) rule, to predict job due dates online.

Dispatching rules are frequently used for dynamic scheduling focusing for single-objective optimization. Multiple objective problems are time- and resource-intensive. Đurasević et al. [44] proposed a new way of combining multiple dispatching rules into groups to handle multiple objective problems more efficiently. They use a simple ensemble construction method to make these groups and test them on different multiple objective problems. They compare them with dispatching rules made by non-dominated sorting genetic algorithms and showed that their approach outperformed dispatching rules with standard multiple-objective algorithms. Đurasević et al. [45] proposed to use ensembles of dispatching rules to improve the performance in complex cases where a solo dispatching rule may not be suffice for well-informed production planning decisions. They introduced a novel way of ensembling different dispatching methods for dynamic scheduling, showing that the combination method improves scheduling outcomes. Gui et al. [46] developed a deep reinforcement learning approach for dynamic flexible job-shop scheduling (DFJSP) with a composite scheduling action; they proposed a composite approach that allows for combining several dispatching rules, a continuous and flexible rule space, and weight adjustment. A reward function based on mean tardiness was defined to guide the learning process.

Most recently, Xiong et al. [47] studied a dynamic single-machine scheduling problem with predicted arriving jobs where some jobs have

known or predictable release times. They proposed six weighted dispatching rules and two improved heuristics based on the so-called Decision Theory-Tactically Delayed approach. Ghaedy-Heidary et al. [48] proposed a mathematical model for a Stochastic Flexible Job-Shop Scheduling Problem and linked it with a simulation tool that mimic the photolithography process in the semiconductor industry context. The simulation tool generates an initial schedule using the LWR rule, which is to be improved using a customized genetic algorithm.

These studies are predominantly concerned with improving the performance of ATCs using advanced computational techniques, like machine learning. We think that there still is room for improving ATCs from a theoretical perspective. The next subsection reviews ATCs for identical and unrelated parallel machine scheduling, which consitute the most relevant research works.

2.3. Relevant studies

The identical parallel machine scheduling is *NP*-hard in the strong sense [49]. Park et al. [50] were the first to use APC with setups as a heuristic rule for minimizing total weighted tardiness. Pfund et al. [12] investigated identical parallel-machine scheduling considering sequence-dependent setup and ready times using ATC with setups. Eom et al. [51] applied the same variant of the ATC rule for a general parallel processing problem with sequence-dependent family setup times. Parallel batch processing problem considering incompatible job families and unequal ready times was studied by Mönch et al. [52]. Li et al. [53] extended the parallel batch processing by considering dynamic job arrivals and SDSTs in addition to incompatible job families and developed the batched variant of ATC to minimize makespan and total weighted tardiness. Tseng et al. [54] adjusted the ATC dispatching rule with setups considering minimum completion time for optimizing parallel machines. Driessel and Mönch [55] considered the basic ATC as a benchmark for evaluating variable neighborhood search approaches for scheduling identical parallel machines with sequence-dependent setups, precedence, and ready time constraints. Lamothe et al. [56] introduced the calendar constraints in parallel machine scheduling and used ATC as a benchmark for evaluating their simulated annealing algorithm. Xi and Jang [14] extended the ATC dispatching rule to simultaneously account for setup and ready times, testing it for identical parallel-machine scheduling. Anzanello et al. [57] introduced the ATC rule with ergonomics factors for minimizing total weighted tardiness and the allocation of batches with similar complexities to the same operators in parallel processing settings. Su et al. [58] extended the basic ATC to allow for flexibility considerations for solving parallel machine scheduling with eligibility constraints. More recently, Vimala Rani and Mathirajan [59] used the batched ATC approach for evaluating dynamic scheduling problems in identical parallel machine settings considering various optimization objectives.

In unrelated parallel machine studies, independent jobs must be scheduled in parallel, but not necessarily identical machines; polynomial time approximation algorithms have been developed for special cases of makespan minimization [60], ordered and nested machine sets [61], among other examples, where the problem becomes tractable by restricting some of their parameters [62]. Logendran and Subur [11] were the first to use the basic ATC for scheduling considering the job-splitting feature. Chen [63] used the ATC with setups for scheduling unrelated parallel machines with SDSTs and due date constraints. ATC was also considered a benchmark for comparing various dispatching rules for scheduling unrelated parallel machines with due and release dates [64]. Bektur and Saraç [16] employed ATC with setups for scheduling unrelated parallel machines with SDSTs, machine eligibility restrictions, and the use of a common server. Most recently, Jaklinović et al. [65] proposed a genetic programming approach for constructing adjustable and problem-specific dispatching rules and showed that the automatically generated rules perform better than the general dispatching rules like ATC.

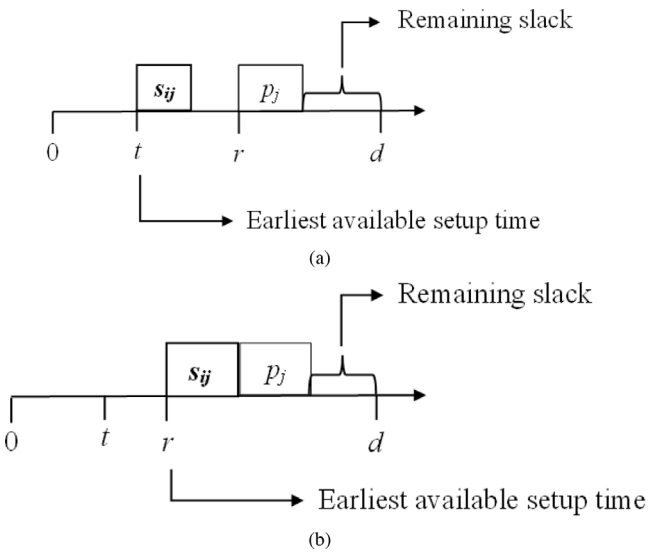

Fig. 1. An illustration of the separable (a) and continuous (b) setup formations.

Overall, the ATC dispatching rule has been extended considering different features and methods. All these studies applied the idea of minimizing delays with a single-machine scheduling outlook. We believe that dispatching jobs on parallel machines requires a more comprehensive and system-oriented outlook; something that is dismissed in the scheduling literature. The next section elaborates on the proposed dispatching method.

3. Proposed method

3.1. Problem definition

To formalize the problem, let us assume that m parallel machines are to process n independent jobs. The parallel machines are either identical or unrelated. The machines differ with respect to machine ready time t_m, job ready time r_j, job processing time p_j, due date d_j, and weight of job tardiness penalty w_j. The objective is to dispatch the jobs such that a schedule with the minimal weighted total tardiness (i.e., *Min* $\sum_{j=1}^{n} w_j T_j$) is achieved. In this definition, T_j is the job tardiness and can be obtained by $Max\left(0,\ C_j - d_j\right)$ and C_j specifies the completion time. It is assumed that machines can only process one job at a time, and thus have no preemption. Besides, jobs are inseparable and the whole system is stable with no machine malfunction.

Since jobs with different attributes are processed consecutively, a setup time must be considered for changing/feeding materials and/or cleaning/maintaining the machine. That is, after the completion of the previous job (j), preparations are required to be able to process the following job (i); this is denoted by s_{ij}. Xi and Jang [14] suggested that setup time can be divided into separable and continuous variants where the operation takes place before and after the arrival of jobs at the production system, respectively, as shown in Fig. 1. Setup can be predominantly activated before the arrival of the following job; hence, separable SDST is generally adopted. The application of continuous SDST is limited to a few application areas, such as metal processing, where metals must be preprocessed using hot stamping that must be implemented only after the arrival of the job to the respective production stage. Considering the limited application of continuous setup time, separable SDST is considered in the present study.

3.2. Modeling components

The proposed dispatching rule is composed of the components shown in the dispatching aggregate function, Eq. (2). Terms A and B

refer to the WSPT rule and the LS rule, respectively. These two rules are the foundational components of the ATC dispatching rule. Terms C and D refer to the SST and SRT rules. These two rules can be added depending on job characteristics. Like other ATC-based dispatching methods, our method is developed around analyzing and modifying the dispatching elements and parameters (i.e., k_1, k_2, and k_3).

$$\begin{aligned}\text{Index} = \text{A(WSPT)} \times \text{B}((LS\ numerator)/(LS\ denominiator)) \times \text{C}\\ ((SST\ numerator)/(SST\ denominiator)) \times \text{D}((SRT\ numerator)/\\ (SRT\ denominiator))\end{aligned} \quad (2)$$

This composition operates as follows. In cases of having a tight due date, k_1 should be set at its maximum value. In this situation, and considering the dispatching aggregate function, the natural exponential of B(LS) becomes 1, hence, the dispatching rule depends on A(WSPT). Consequently, the rule selects jobs with higher weights and shorter processing times at a higher priority to reduce the loss incurred from late deliveries. Alternatively, when the due date is relatively loose, with no overdue jobs, k_1 can be set to a minimum value. Therefore, the natural exponential of B(LS) becomes very small, and the value of the aggregation equation decreases accordingly. In this case, the dispatching system discourages the priority processing of jobs with longer slack time to lower the possibility of scheduling delays using the LS rule. In situations with long setup times, i.e., having highly diverse products/orders, k_2 should be set to a smaller value. A smaller exponential of C(SST) results in a smaller value of the aggregation equation, meaning that the dispatching system discourages the priority processing of jobs with an overly long setup time. In circumstances when jobs are not yet ready while the machines are already available, the exponential of the relation between the job ready time and the machine ready time becomes smaller when k_3 in D(SRT) is set to a smaller value. In this situation, the overall dispatching value decreases, indicating that the dispatching system tends to postpone the processing time of later-arriving jobs to avoid unnecessary machine idleness.

This study employs the grid approach to analyze the impact of dispatching parameters. The method has been widely applied in the optimization context. For example, Pfund et al. [12] applied the grid approach using regression analysis based on the *k*-value range proposed by Lee et al. [10]. On this basis, they considered a total of 22 $k_1 \times$ 11 $k_2 \times$ 13 $k_3 = 3146$ combinations to account for the minimal required partial *k*-value settings and solved the instances to find the best alternative. Xi and Jang [14] analyzed the same experimental factors by generating 5103 random instances and solved them. The results revealed that on average, 75% of the k values used for the optimal solution fell within the sets generated by the grid approach, confirming that the grid approach is comprehensive and quite effective for determining the dispatching parameters.

In addition to the k values, the effectiveness of the dispatching rule is influenced by the combinations of the WSPT, LS, SST, and SRT functions considering the inherent differences. These functions are now discussed to determine the best configuration based on experimental analysis presented in the next section.

3.2.1. Analysis of WSPT

The basic WSPT is encapsulated in w_j/p_j, which becomes $w_j/(p_j + S_{ij})$ in the presence of setup term. Under the condition of differing job ready times and a separable SDST, preparations can be made prior to the arrival of jobs, although the machines may still be in an idle state after preparations are completed. Hence, the shortest completion time of the current job, j, can be calculated by $p_j + max(s_{ij},\ r_j - t)$, where $r_j - t$ specifies the machine idle time before the arrival of jobs at the production system. Therefore, it is reasonable to regard the machine idle time as part of the job processing time. A similar argument is by Xi and Jang [15], where $w_j/(p_j + max(S_{ij},\ r_j - t))$ is considered for constituting the WSPT.

3.2.2. Analysis of the LS numerator

The numerator of the LS exponential function can be in the following forms:

i. $-max\left(d_j - p_j - t, 0\right)$. This is the most basic form for calculating the least slack time. In this form, jobs with looser due dates are postponed and jobs with tighter due dates are assigned a greater value and are thus processed at greater priority. Notably, the setup time s_{ij} is not incorporated in this formation. Several studies have applied this form in calculations, including the recent ATC dispatching study conducted by Xi and Jang [15].

ii. $-max\left(d_j - p_j - s_{ij} - t, 0\right)$. This form incorporates SDSTs into the calculation of the least slack time. Compared with the preceding form, this numerator term yields a more precise estimate of the slack time, which enables factory managers to utilize production capacity more effectively and reduce unnecessary machine idleness.

iii. $-max\left(d_j - p_j - \max(r_j, t), 0\right)$. This form is formulated to calculate the least slack time when jobs have differing ready times and when the machine's idleness prior to the arrival of jobs must be considered. For this purpose, the earliest available job processing time can be obtained using $max(r_j,\ t)$. In terms of job ready time, the least slack times are $-max(d_j - p_j - t, 0)$ and $-max(d_j - p_j - r_j, 0)$ when jobs arrive at the production system before and after the machine ready time, respectively.

iv. $-max\left(d_j - p_j - \max(r_j, t + s_{ij}), 0\right)$. When jobs have a separable SDST in between and have differing ready times, the earliest machine ready time is $max(r_j,\ t + s_{ij})$. In this situation, the minimal slack times are $-max(d_j - p_j - r_j, 0)$ and $-max(d_j - p_j - t - s_{ij}, 0)$ when preparations are completed before and after the jobs arrive, respectively.

3.2.3. Analysis of LS denominator

To make a balance between the components of the LS exponential function, the existing studies divided the denominator by the mean processing time $\overline{p}$ of the current pending jobs. However, when the value of the setup time s_{ij} is too large, restricting $\overline{p}$ solely to the LS items is likely to weaken the LS rule. To address this issue, the denominator is set to $k_1(\overline{p}+\overline{s})$. The use of variable mean processing time $\overline{p}$ and variable mean setup time $\overline{s}$ helps ensure that every time a job is scheduled, recalculations are required due to the change in the remaining number of pending jobs.

Applying this setting, the current machine ready time t constantly changes, and variations in $\overline{p}$ and $\overline{s}$ also affect dispatch calculations. Considering that denominator restrictions may impact the preciseness of the calculations of the numerators, we propose a method based on the mean processing time $\overline{p}_{fixed}$ and the mean setup time $\overline{s}_{fixed}$ of all initial jobs to reduce the interference of denominator parameters. The developed dispatching rule, therefore, only accounts for the current machine ready time t, and the rule schedules priority jobs based on the numerator. Furthermore, we multiplied $\overline{p}_{fixed}$ by α_1 to estimate the influence of the mean processing time $\overline{p}_{fixed}$ on the solution quality. The LS denominator is, therefore, written as $k_1(\alpha_1 * \overline{p}_{fixed} + \overline{s}_{fixed})$.

Concerning the C(SST) term in the dispatching rule, although SDST is already considered in the WSPT and LS functions, it is suggested that treating the setup time factor as a separate item can significantly reduce total tardiness during production. The numerator of the SST exponential function is $max(s_{ij})$, hence, jobs with a greater negative value, i.e., a longer setup time, are postponed. The denominator $k_2(\overline{s})$ regulates the level of influence that the SST term has on the dispatching rule. For a similar reason explained in function (B) and to reduce the interference of time parameter variability, we propose to apply the initial setup time $\overline{s}_{fixed}$ for calculation and $k_2(\overline{s}_{fixed})$ for examination.

3.2.4. Analysis of SRT

The basic SRT function in the dispatching rule is mostly used for static scheduling in production environments with sufficient raw materials. In actual situations, jobs have different ready times due to inconsistent logistics schedules for a variety of raw material providers or due to the implementation of quality inspections prior to material feeding. Actual production capacity and job scheduling accuracy may decrease if these considerations are not included in the analysis. To address this point, recent studies have expressed machine idle time and the SRT term as $r_j - t$ and $max(r_j - t, 0)$, respectively; this represents the case when machines are available, but jobs are not yet ready. Jobs with later ready times result in an increase in the machine's idle time and should thus be postponed to avoid unnecessary idleness. The denominator $k_3(\overline{p})$ is used to regulate the level of influence that the SRT term has on the dispatching rule.

Considering that setup operations can proceed before jobs are ready, machine idle time can be updated to $r_j - s_{ij} - t$, hence, the SRT function can thereby verify whether jobs are yet to arrive at the production system when setups are complete. For this purpose, $max(r_j - s_{ij} - t,\ 0)$ is used to measure machine idleness more precisely. When the setup time is too long, restricting $k_3\ (\overline{p})$ to SRT may be ineffective. Hence, we propose to incorporate $k_1(\overline{p}+\overline{s})$. Given that denominator ranges may impact the preciseness of calculating the numerators, we applied the mean processing time obtained by all initial jobs $\overline{p}_{fixed}$ and the mean setup time $\overline{s}_{fixed}$ as the basis to reduce the interference of the denominator parameter. By doing so, one can only consider the current available processing time t and schedule the current priority jobs using the numerator. The SRT function uses different denominator restrictions than that of LS to test for possible solutions while developing additional solution combinations. Finally, $\overline{p}_{fixed}$ is multiplied by α_2 to evaluate the level of influence that $\overline{p}_{fixed}$ has on the dispatching rule; this item is formulated as $k_3(\alpha_2 * \overline{p}_{fixed} + \overline{s}_{fixed})$.

3.3. Computational steps of the dispatching rule

Two approaches for machine selection in unrelated parallel machines are proposed.

The ***first*** approach is ATC considering full utilization of the machine capacity and its optimal completion time. Chen [63] proposed a dispatching rule for a full utilization of the machine's capacity. Our study builds on this method and extends it to solving unrelated parallel-machine scheduling. In the mechanism used in the earlier parallel-machine studies, the job does not necessarily have to be processed by the earliest available machine. In this definition, if the earliest machine is not the best choice for a current job, it is best to leave the current machine idle and dispatch the job to the machine with the highest efficiency to achieve the shortest completion time. The process is shown in the following steps.

Step 1. Set the initial problem parameters, i.e., the number of machines, the number of jobs, the time point at which the machines can be processed, the processing time of jobs, the SDSTs, job arrival time, the due date, and the tardiness penalties.

Step 2. Identify the earliest machine that can be activated at present. For this purpose, the remaining jobs are first evaluated using the proposed ATC dispatching rule. The job with the largest indicator value is then re-evaluated on every machine. Finally, the job is scheduled on the machine with the shortest completion time; that is, $Min\left(Max\left(r_j,\ t_m + s_{mij}\right) + p_{mj}\right),\ m = 1, \ldots, M$.

Step 3. Update the average remaining time, setup time, and the machine ready time. Repeat steps 2∼3 for the remaining unscheduled jobs until all jobs are scheduled.

Step 4. Calculate the total weighted tardiness time and store the target value.

Step 5. Repeat steps 1∼5 and apply the Grid Approach as a basis for the stopping condition. Report the best target value.

The ***second*** approach is ATC considering different machine ready times and its optimal completion time. The existing literature considers the full usage of the machine capacity and the time of the earliest available machine, t, as the time reference point of the ATC dispatching rule for solving unrelated parallel machines. For example, in the ATCS dispatching rule below,

$$I_j(t,l) = \frac{w_j}{p_j} exp\left(-\frac{(max\left(d_j - p_j - t\right),\ 0)}{\left(k_1\right)\overline{p}}\right) exp\left(-\frac{S_{ij}}{\left(k_2\right)\overline{s}}\right)$$

the approach is limited in that the job should only be dispatched to the earliest available machine. Considering other machines' ready time as the time reference point in the ATC dispatch calculation may enable a more effective job processing. We suggest that the ATC dispatching rules should be modified considering different machine ready times.

Taking ATCS as the example, the new dispatching rule should be updated by including the time point at which each of the machines can process the job; that is, the ready time of each machine (t^m, $m = 1, \ldots, M$). The alternative formulation is

$$I_j(t^m,l) = \frac{w_j}{p_{mj}} exp\left(-\frac{(max\left(d_j - p_{mj} - t^m\right),\ 0)}{\left(k_1\right)\overline{p}}\right) exp\left(-\frac{S_{mij}}{\left(k_2\right)\overline{s}}\right)$$

This rule determines the processing priority of the job more accurately and calculates the weighted tardiness of the same job in different machines. The main idea is that the job with the largest indicator value calculated by the ATC dispatching rule is re-measured on every machine, and the machine with the shortest completion time is selected. The computational procedure is different in Step 2, which is: The remaining jobs are evaluated considering time reference points and different machine ready times. Next, the job with the largest indicator value is re-measured on every machine. Finally, the job is scheduled on the machine with the shortest completion time. That is, $Min\left(max\left(r_j,\ t_m + s_{mij}\right) + p_{mj}\right), m = 1, \ldots, M$.

4. Numerical experiments

This section analyzes the performance of the developed dispatching rule comparing it with the ATC with Ready-time and Continuous Setups (ATCRCS) developed by Xi and Jang [15]. Factor analysis is first presented to determine the modeling parameters. A combination analysis of the developed dispatching rule is provided next, followed by a comparative analysis against the benchmark dispatching method. The algorithms are coded and compiled using Microsoft Visual Studio C++ on a PC with an Intel Core i7-6700 (3.4 GHz) processor and 32 GB of RAM.

4.1. Parameter setting

This study considers the factors suggested by Pfund et al. [12] to adjust the model for different parallel machine instances considering the performance of the dispatching rule. For this purpose, the experimental factors, order quantity factor (μ), lead time factor (η), delivery time factor (τ), delivery range factor (R), initial order arrival scale factor (Ja), order arrival time factor (r_τ) are used to simulate changes in the job order considering three levels for each factor, i.e., low, medium, and high. Table 2 lists the experimental factors and their values. Three levels for each of the six parameters result in a total of $3^6 = 729$ configurations. Given seven distinct and randomly generated instances under each configuration, 5103 experiments would be required. To solve the problem of setting the k value, we adopted the Grid Approach proposed by Pfund et al. [12]. This method is based on the range of k values proposed by Lee and Pinedo [66]: $k1 = 4.5 + R,\ for\ R \le 0.5, k1 = 6 - 2R,\ for\ R \ge 0.5$ and $k2 = \tau/(2\sqrt{\eta})$, from which 22 $k1$ values, 11 $k2$ values, and 13 $k3$ values were selected, a total of $22 * 11 * 13 = 3146$ combinations, increasing the possibility of solving the ATC scheduling rule. They formed a combination of scaling parameters by selecting one

Table 2
Factor analysis for determination of the model parameters.

Parameter	Levels		
	Low	Medium	High
Order quantity (μ)	11	19	27
Lead time (η)	0.020	1.010	2.000
Delivery time (τ)	0.300	0.600	0.900
Delivery range (R)	0.250	0.630	1.000
Order arrival scale (Ja)	0.200	0.500	0.800
Order arrival time (r_τ)	1.000	5.500	10.000

value each of $k1$, $k2$, and $k3$ from the list; the findings confirmed that the Grid Approach improves ATC's efficiency.

Considering the above experimental factors, the operational parameters are defined as follows. Given m representing the number of machines, the number of jobs is equal to $n = m * \mu$. The job processing times, p_j, are generated randomly using uniform distribution within the interval (50, 150) s for both identical and unrelated parallel machine instances. On this basis, the processing time parameter ($\overline{p}$) for the identical and unrelated parallel machine settings are, $\overline{p} = \sum_{j=1:n} p_j/n$ and $\overline{p} = \sum_{i=1:m}\sum_{j=1:n} p_{ij}/(m*n)$, respectively. The coefficients of weighted delays are generated randomly using $U(1, 10)$. The setup time factor amounts to $\overline{s} = \eta * \overline{p}$ based on which the SDSTs are generated randomly from $(0, 2\overline{s})$. The maximum completion time is calculated following Lee et al. [10], that is $C_{max} = (\beta.\overline{s}+\overline{p})\mu$, where $\beta = 0.4+{}^{10}/_{\mu^2}-{}^{\eta}/_{7}$. The delivery time parameter is calculated using $\overline{d} = C_{max}(1-\tau)$, based on which the delivery time values are generated randomly with uniform distribution of $U[(1-R)\overline{d}, \overline{d}]$ with the probability of τ, and $U[\overline{d}, \overline{d} + \left(C_{max} - \overline{d}\right)R]$ with probability of $1-\tau$. Finally, the arrival time parameter for the job orders is generated as follows. For identical parallel machines, with probability of Ja, $r_\tau = 0$ or generated randomly within $[\max\left(d_j - r_\tau * p_j, 0\right), d_j]$ when $1 - Ja$. For unrelated parallel machines, the above values are $r_\tau = 0$ and $[\max\left(d_j - r_\tau * \overline{p_j}, 0\right), d_j]$. Finally, the arrival times are generated randomly, using $U(1, 100)$.

4.2. Analysis of the dispatching combinations and parameters

Operational characteristics such as machine selection, SDST, and differing job ready times, among other terms listed in Table 3, are used for numerical analysis. This table shows that when, for example, the WSPT and SST terms are fixed, the LS numerator gets four situations for identical parallel machines ($B1 - B4$) and unrelated parallel machines ($\acute{B}1 - \acute{B}4$). Likewise, the SRT numerator account for the identical parallel machines ($D1 - D2$) and the unrelated parallel machines ($\acute{D}1 - \acute{D}2$). Finally, $E1 - E5$ denote the respective combinations considering the MS, SST, and SRT denominators. These terms and values are aggregated using Eq. (2) to generate an index.

Considering four/two values for the LS/SRT numerators while WSPT and SST are fixed, six parameters at three different levels make 729 scenarios; with seven distinct problems under each scenario, a total of 5103 instances has resulted. In the proposed configuration, E5, the alpha parameters should be calibrated to determine the best dispatching terms. For this purpose, α_1 and α_2 values are set from low to high (i.e., 0.1, 0.3, 0.5, and 0.7) to estimate the influence of the mean processing time changes on the solution quality. Different combinations of α_1 and α_2 in the denominators of the LS and SRT are then considered. The alternatives for $k_1\left(\alpha_1.\overline{p_{fixed}} + \overline{s_{fixed}}\right)$, $k_2\left(\overline{s_{fixed}}\right)$, and $k_3\left(\alpha_2.\overline{p_{fixed}} + \overline{s_{fixed}}\right)$, are then compared with respect to the number of improved solutions; the objective is to reduce the interference of initial time variability, which is considered as the comparison criterion. Table 4 summarizes the results.

The analysis of alpha parameters in Table 4 shows that better solutions were obtained with smaller α_2 and regardless of the changes to α_1. Overall, the number of best and similar solutions was the greatest when α_1 and α_2 were equal to 0.3. Finally, α_1 and α_2 are set at 0.3 and three additional levels below and above the selected thresholds are considered to verify the solution quality, as shown in Table 5.

The analysis shows that when the values of α_1 and α_2 are both 0, the denominators of LS and SRT excluding the mean processing time $\overline{p}_{fixed}$ obtained solutions for all random problems that performed 45 percent worse relative to the comparison criterion. This finding implies that the $\overline{p}_{fixed}$ parameter must be included in the analysis. However, when α_1 and α_2 are 0.1 and the LS and SRT denominators had a $\overline{p}_{fixed}$ that was 0.1 times the mean processing time, the number of improved solutions obtained was greater than those obtained using the E5 terms. When α_1 and α_2 ranged from 0.2 to 0.4, the number of improved solutions gradually increased and when alphas were set to 0.4, the most improved and equal-quality solutions were obtained (i.e., about 77 percent of all problems). Finally, with alphas ranging between 0.5 and 0.6, the number of improved solutions gradually decreased. As a result, $k_1\left(0.4 * \overline{p_{fixed}} + \overline{s_{fixed}}\right)$, $k_2\left(\overline{s_{fixed}}\right)$, and $k_3\left(0.4 * \overline{p_{fixed}} + \overline{s_{fixed}}\right)$ will be used for the next step of the numerical experiments. Next, the dispatching combinations are obtained and analyzed separately for solving identical and unrelated parallel machines considering the same instances. The results are summarized in Tables 6–7.

The results show that the proposed combination, $k_1\left(0.4 * \overline{p_{fixed}} + \overline{s_{fixed}}\right)$, $k_2\left(\overline{s_{fixed}}\right)$, and $k_3\left(0.4 * \overline{p_{fixed}} + \overline{s_{fixed}}\right)$, yielded an average of approximately 92 and 80 percentages of improved solutions for identical and unrelated parallel machines, respectively, regardless of the changes in SRT/LS terms. This finding implies that reducing the interference of time variability significantly improves the performance of the dispatching rule. On this basis, the optimal dispatching combination for the identical and unrelated parallel machine scheduling problems are presented in Eqs. (3), and (4), respectively.

$$I_j(t,l) = \frac{w_j}{p_j + max\left(s_{ij},\ r_j - t\right)} exp\left(-\frac{\max(d_j - p_j - s_{ij} - t,\ 0)}{\left(k_1\right)\ \left(0.4 * \overline{p_{fixed}} + \overline{s_{fixed}}\right)}\right) exp\left(-\frac{\max(s_{ij})}{\left(k_2\right)\overline{s_{fixed}}}\right) exp\left(-\frac{max\left(r_i - t, 0\right)}{\left(k_3\right)\left(0.4 * \overline{p_{fixed}} + \overline{s_{fixed}}\right)}\right) \quad (3)$$

Table 3
Terms and combinations in the dispatching rule.

WSPT			Identical	$A1$												$\frac{w_j}{p_j+max(s_{ij},r_j-t)}$
			Unrelated	$\acute{A}1$												$\frac{w_j}{p_{mj}+max(s_{mij},r_j-t)}$
LS	Numerator	Identical	$B1$	$\max\ (d_j - p_j - t, 0)$	$B2$	$\max\ (d_j - p_j - s_{ij} - t, 0)$	$B3$	$\max\ (d_j - p_j - max(r_j, t), 0)$	$B4$	$\max\ (d_j - p_j - max(r_j, t + s_{ij}), 0)$						
		Unrelated	$\acute{B}1$	$\max\ (d_j - p_{mj} - t, 0)$	$\acute{B}2$	$\max\ (d_j - p_{mj} - s_{mij} - t, 0)$	$\acute{B}3$	$\max\ (d_j - p_{mj} - max(r_j, t), 0)$	$\acute{B}4$	$\max\ (d_j - p_{mj} - max(r_j, t + s_{mij}), 0)$						
	Denominator	Identical	E1	$k_1\left(\overline{p}\right)$	E2	$k_1\left(\overline{p}\right)$	E3	$k_1\left(\overline{p} + \overline{s}\right)$	E4	$k_1\left(\overline{p} + \overline{s}\right)$	E5	$k_1\left(\alpha_1.\overline{p_{fixed}} + \overline{s_{fixed}}\right)$				
		Unrelated														
SST	Numerator	Identical	$C1$	$\max\ (s_{ij})$												
		Unrelated	$\acute{C}1$	$\max\ (s_{mij})$												
	Denominator	Identical	E1	$k_2\left(\overline{s}\right)$	E2	$k_2\left(\overline{s}\right)$	E3	$k_2\left(\overline{s}\right)$	E4	$k_2\left(\overline{s}\right)$	E5	$k_2\left(\overline{s_{fixed}}\right)$				
		Unrelated														
SRT	Numerator	Identical	D1	$\max\ (r_j - t, 0)$	D2	$\max\ (r_j - s_{ij} - t, 0)$										
		Unrelated	$\acute{D}1$	$\max\ (r_j - t, 0)$	$\acute{D}2$	$\max\ (r_j - s_{mij} - t, 0)$										
	Denominator	Identical	E1	$k_3\left(\overline{p}\right)$	E2	$k_3\left(\overline{p} + \overline{s}\right)$	E3	$k_3\left(\overline{p}\right)$	E4	$k_3\left(\overline{p} + \overline{s}\right)$	E5	$k_3\left(\alpha_2.\overline{p_{fixed}} + \overline{s_{fixed}}\right)$				
		Unrelated														

Table 4
Analyses of E5 parameters $\alpha 1$ and $\alpha 2$.

α_1	α_2	Improved solutions		Similar solutions		Inferior solutions	
		Frequency	Percentage	Frequency	Percentage	Frequency	Percentage
0.1	0.1	1856	36.3	1499	29.3	1748	34.2
0.1	0.3	1828	35.8	1563	30.6	1712	33.5
0.1	0.5	1777	34.8	1556	30.4	1770	34.6
0.1	0.7	1727	33.8	1630	31.9	1746	34.2
0.3	0.1	2010	39.3	1722	33.7	1371	26.8
0.3	0.3	2016	39.5	1809	35.4	1278	25.0
0.3	0.5	1975	38.7	1828	35.8	1300	25.4
0.3	0.7	1885	36.9	1900	37.2	1318	25.8
0.5	0.1	1974	38.6	1859	36.4	1278	25.0
0.5	0.3	1946	38.1	1969	38.5	1188	23.2
0.5	0.5	1875	36.7	1966	38.5	1262	24.7
0.5	0.7	1779	34.8	2092	40.9	1232	24.1
0.7	0.1	1951	38.2	1832	35.9	1320	25.8
0.7	0.3	1915	37.5	1952	38.2	1236	24.2
0.7	0.5	1844	36.1	1951	38.2	1308	25.6
0.7	0.7	1744	34.1	2054	40.2	1305	25.5

Table 5
Initial calibration of alpha parameters.

α_1	α_2	Improved solutions		Similar solutions		Inferior solutions	
		Frequency	Percentage	Frequency	Percentage	Frequency	Percentage
0.0	0.0	1275	25	1545	30	2283	45
0.1	0.1	1856	36	1499	29	1748	35
0.2	0.2	2016	40	1809	35	1278	25
0.3	0.3	2016	40	1809	35	1278	25
0.4	0.4	2016	40	1907	37	1212	23
0.5	0.5	1875	37	1966	39	1262	24
0.6	0.6	1851	36	1967	39	1285	25

Table 6
Analysis of the dispatching combinations for identical parallel machines.

Combination	Solutions (Percentage)	$r_j - t$				$r_j - s_{ij} - t$			
		$d_j - p_j - max(r_j, t)$	$d_j - p_j - max(r_j, t + s_{ij})$	$d_j - p_j - t$	$d_j - p_j - s_{ij} - t$	$d_j - p_j - max(r_j, t)$	$d_j - p_j - max(r_j, t + s_{ij})$	$d_j - p_j - t$	$d_j - p_j - s_{ij} - t$
E1	Improved	0	29.6	7.7	31.1	24.0	30.2	25.2	30.7
	Similar	100	34.8	85.4	32.7	31.1	17.1	31.3	16.4
	Inferior	0	35.7	6.9	36.2	44.9	52.7	43.5	52.9
E2	Improved	17.5	32.7	18.6	33.3	24.8	31.3	25.5	31.6
	Similar	65.5	29.4	64.5	28.7	29.9	16.5	30.8	16.1
	Inferior	17.0	38.0	17.0	38.0	45.3	52.1	43.7	52.3
E3	Improved	32.2	41.5	34.6	42.1	37.9	40.4	39.0	40.5
	Similar	46.1	24.7	41.7	23.7	20.4	13.6	20.5	13.2
	Inferior	21.7	33.8	23.7	34.2	41.7	46.0	40.5	46.3
E4	Improved	37.4	43.5	38.5	44.2	38.5	40.7	39.5	41.0
	Similar	36.3	21.9	35.5	21.3	19.9	13.3	20.1	13.0
	Inferior	26.3	34.6	26.0	34.5	41.6	46.0	40.3	46.1
E5	Improved	91.8	91.7	91.8	91.8	91.8	91.7	91.8	91.8
	Similar	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Inferior	8.2	8.3	8.2	8.2	8.2	8.3	8.2	8.2

$$I_j(t^m, l) = \frac{w_j}{p_{mj} + max\left(s_{mij},\ r_j - t^m\right)} \times exp\left(-\frac{\max(d_j - p_{mj} - s_{ij} - t^m, 0)}{(k_1)\ (0.4 * \overline{p_{fixed}} + \overline{s_{fixed}})}\right) exp\left(-\frac{s_{mij}}{(k_2)\,\overline{s_{fixed}}}\right) exp\left(-\frac{max\left(r_i - t^m, 0\right)}{(k_3)\,(0.4 * \overline{p_{fixed}} + \overline{s_{fixed}})}\right) \quad (4)$$

4.3. Comparative analysis

The Proposed ATC vs. ATCRSS. The dispatching rule developed for identical parallel machines (Eq. (3)) is first compared with the ATCRSS method developed by Xi and Jang [15]. In the comparative analysis, the job number ($\mu = n/M$), setup ($\eta = \overline{s}/\overline{p}$), due date ($\tau = 1 - \overline{d}/C\_max$), due date range ($R = ((d\_max - d\_min))/C\_max$), the ratio of initial ready jobs (J_a = initial number of ready jobs at time 0/ total number of jobs),

Table 7
Analysis of the dispatching combinations for unrelated parallel machines.

Combination	Solutions (Percentage)	$r_j - t$				$r_j - s_{ij} - t$			
		$d_j - p_j - max(r_j, t)$	$d_j - p_j - max(r_j, t + s_{ij})$	$d_j - p_j - t$	$d_j - p_j - s_{ij} - t$	$d_j - p_j - max(r_j, t)$	$d_j - p_j - max(r_j, t + s_{ij})$	$d_j - p_j - t$	$d_j - p_j - s_{ij} - t$
E1	Improved	0	31.4	6.4	31.8	21.9	29.0	20.9	28.7
	Similar	100	36.3	35.9	34.9	30.3	17.7	31.8	17.8
	Inferior	0	32.3	7.7	33.0	47.9	53.3	47.3	53.6
E2	Improved	17.2	34.2	16.3	34.2	22.6	30.1	22.1	29.8
	Similar	69.1	33.0	69.8	32.5	35.6	21.2	36.9	21.1
	Inferior	13.7	32.8	13.9	33.3	41.7	48.6	41.1	49.0
E3	Improved	35.4	44.5	37.0	45.0	39.6	42.3	39.5	42.0
	Similar	44.2	24.4	40.7	23.8	23.6	17.3	24.1	17.0
	Inferior	20.4	31.1	22.3	31.2	36.7	40.4	36.0	41.0
E4	Improved	40.5	46.5	40.0	46.7	40.5	43.1	40.0	42.8
	Similar	36.4	22.8	36.5	22.6	23.1	17.1	23.8	17.0
	Inferior	23.1	30.7	23.5	30.7	36.4	39.8	36.2	40.3
E5	Improved	80.7	80.6	80.5	80.3	80.2	80.1	80.1	79.9
	Similar	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Inferior	19.3	19.4	19.5	19.7	19.8	19.9	19.9	20.1

Table 8
Comparative analysis of the scheduling of identical parallel machines.

Factor	Level	Frequency		
		Improved	Similar	Inferior
μ	Low (11)	1655	0	46
	Moderate (19)	1689	0	12
	High (27)	1698	0	3
η	Low (0.2)	1672	0	29
	Moderate (1.0)	1688	0	13
	High (2.0)	1677	0	24
τ	Low (0.3)	1701	0	0
	Moderate (0.6)	1698	0	3
	High (0.9)	1650	0	51
R	Low (0.25)	1686	0	15
	Moderate (0.63)	1685	0	16
	High (1.0)	1688	0	13
J_a	Low (0.2)	1687	0	14
	Moderate (0.5)	1688	0	13
	High (0.8)	1684	0	17
r_τ	Low (1.0)	1687	0	14
	Moderate (5.5)	1680	0	21
	High (10.0)	1684	0	17

and the ready time factor (r_τ) are considered separately. We controlled no more than one parameter at a time, and a total of 1701 problems were randomly generated by the seven sets of random problems in each of the $3^5 = 243$ scenarios for the comparative analysis. The results are presented in Table 8.

Results show that the vast majority of solutions for small-, medium, and large-scale instances are improved, and the superiority becomes even more evident for larger problems. The same conclusion is true for the rest of the parameters, that is, the developed dispatching rule performs better than the baseline regardless of the parameters and the associated levels. Analyzing the solutions considering the tightness between mean job due dates and total completion time, we observed that when τ approached 1, the due date was very tight because the completion time was greater than the due date while with τ approaching 0, the due date became relatively loose. Considering the relation between the due date range and total completion time, when R approached 0, the due dates of jobs were close to each other; otherwise, they were relatively scattered. Besides, in instances with Ja closer to 0, machines had a higher probability of being idle because most jobs were not ready at the production system when the machine was ready. Finally, with a decrease in r_τ, the probabilities of machine idleness and scheduling tardiness increased because job ready times tended to approach the corresponding due dates. As a final step, the overall effectiveness of the proposed dispatching rule over the 5103 random test instances is calculated. Considering the tardiness deviation value calculated by $Proposed\ ATC - ATCRSS / Proposed\ ATC \times 100\%$, the total weighted tardiness is reduced by approximately 14 percent (on average).

The Proposed ATC in Unrelated Parallel-Machine Scheduling. As a final step to the numerical analysis, the developed dispatching rule (method 2), which uses different machine ready times as a criterion for sorting orders, is compared with method 1 (where the earliest machine ready time is considered as the indicator) for unrelated parallel machines. It can be seen in Table 9 that method 2 proposed in this study performs significantly better than method 1. Among the 5103 random questions generated in this study, nearly 83% of the questions are better than or equal to method 1.

The last test is conducted considering the weighted tardiness time deviation value formula; that is, (method 1 − method 2)/method 2 × 100%. We found that method 2 reduces the total weighted tardiness time by 3.4 percent on average compared with method 1, which confirms the superiority of the proposed method in this study. Overall, the second method incorporated different machine ready times and their optimal completion time into the ATC dispatching rule, which is evidently more effective than the ATC dispatching method that considers the earliest machine ready time by the existing methods.

5. Conclusions

Dispatching rules are used for prioritizing the jobs assigned for processing by a machine as well as solution initialization for metaheuristics. The well-known ATC dispatching rule, which was initially developed for solving single-machine scheduling problems, is widely applied in other production settings for job prioritization while it may not be as effective when more machines are involved, like in parallel-machine environments. This study developed a novel dispatching rule for optimizing identical and unrelated parallel machine scheduling problems more effectively.

The existing ATC-based dispatching rules consider the earliest machine completion time as the time reference point, which prevents unnecessary machine idleness and helps maximize the machine's production capacity. Applying this logic means that only one job at a time (i.e., the job with the highest priority value) will be assigned

Table 9
Comparative analysis of ATCs in unrelated parallel machine scheduling.

Method	Improved	Similar	Inferior
Considering full utilization of the machine capacity and its optimal completion time	0	5103 (100%)	0
Considering different machine ready times and its optimal completion time	2035 (40%)	2205 (43%)	863 (17%)

to the earliest ready machine. However, the job priority might be different if the time reference point is calculated considering the other machines. In parallel-machine scheduling, jobs may have their optimal machines due to differing machine efficiencies; hence, all machines ought to be considered during job assignment from which, the machine with the shortest completion time can be selected for processing. The dispatching rule developed in this study considers all current ready jobs on all current ready machines for scheduling. The developed approach prevents the dispatching rule from selecting a local solution, which may be the best for a single-machine situation, but not for a parallel-machine; this guarantees more precise job priorities.

Four ATC-based dispatching rules, namely the WSPT, LS, SST, and SRT rules, were classified and analyzed; on this basis, a new method for reducing the interference of time variability in the denominator is introduced to ensure that the numerators of the LS and SRT rules compile with the same scheduling criterion considering different operational dynamics. Further, new dispatching combinations were explored. The experiments demonstrated that the developed dispatching rule obtained solutions with better total weighted tardiness when compared to the most widely applied approach in the literature; the results were improved by approximately 14 percent.

This study is limited in that static k values were considered for all operational situations; that is, the selection of the best value for parameter k impacts the effectiveness of the developed dispatching rule. As the first suggestion for future research, one can develop an optimization model for k-value approximation considering the job information and processing time for dynamic adjustment. Second, the proposed dispatching rule should be employed in various soft computing techniques to obtain better solutions in a shorter computational time when solving different classes of scheduling problems. Besides, a dynamic dispatching mechanism comprising more methods can be even more effective for hybrid scheduling when factories have temporary job insertions. Finally, the developed dispatching rule can be improved by incorporating the Pareto-front and domination concepts for optimizing other objective functions along with the weighted tardiness.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] S. Jain, W.J. Foley, Dispatching strategies for managing uncertainties in automated manufacturing systems, Eur. J. Oper. Res. 248 (1) (2016) 328–341.

[2] C. Ferreira, G. Figueira, P. Amorim, Effective and interpretable dispatching rules for dynamic job shops via guided empirical learning, Omega (Westport) 111 (2022) 102643.

[3] S. Panda, Y. Mei, M. Zhang, Simplifying dispatching rules in genetic programming for dynamic job shop scheduling, 2022, pp. 95–110.

[4] M. Đurasević, D. Jakobović, A survey of dispatching rules for the dynamic unrelated machines environment, Expert Syst. Appl. 113 (2018) 555–569.

[5] S. Wang, H. Su, G. Wan, L. Zhong, Surgery scheduling in the presence of operating room eligibility and dedicated surgeon: an adaptive composite dispatching method, Int. J. Prod. Res. (2022) 1–16.

[6] J. Heger, T. Voss, Dynamically adjusting the k -values of the ATCS rule in a flexible flow shop scenario with reinforcement learning, Int. J. Prod. Res. (2021) 1–15.

[7] A.P.J. Vepsalainen, T.E. Morton, Priority rules for job shops with weighted tardiness costs, Manage. Sci. 33 (8) (1987) 1035–1047.

[8] C.-Y. Cheng, P. Pourhejazy, K.-C. Ying, C.-F. Lin, Unsupervised learning-based artificial bee colony for minimizing non-value-adding operations, Appl. Soft Comput. [Internet] 105 (2021) 107280, Available from: https://linkinghub.elsevier.com/retrieve/pii/S1568494621002039.

[9] P.S. Ow, T.E. Morton, The single machine early/tardy problem, Manage. Sci. 35 (2) (1989) 177–191.

[10] Y.H. Lee, K. Bhaskaran, M. Pinedo, A heuristic to minimize the total weighted tardiness with sequence-dependent setups, IIE Trans. 29 (1) (1997) 45–52.

[11] R. Logendran, F. Subur, Unrelated parallel machine scheduling with job splitting, IIE Trans. [Internet] 36 (4) (2004) 359–372, Available from: http://www.tandfonline.com/doi/abs/10.1080/07408170490279598.

[12] M. Pfund, J.W. Fowler, A. Gadkari, Y. Chen, Scheduling jobs on parallel machines with setup times and ready times, Comput. Ind. Eng. [Internet] 54 (4) (2008) 764–782, Available from: https://linkinghub.elsevier.com/retrieve/pii/S036083520700229X.

[13] S.J. Mason, J.W. Fowler, W. Matthew Carlyle, A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops, J. Sched. 5 (3) (2002) 247–262.

[14] Y. Xi, J. Jang, Scheduling jobs on identical parallel machines with unequal future ready time and sequence dependent setup: An experimental study, Int. J. Prod. Econ. 137 (1) (2012) 1–10.

[15] Y. Xi, J. Jang, Minimizing total weighted tardiness on a single machine with sequence-dependent setup and future ready time, Int. J. Adv. Manuf. Technol. 67 (1–4) (2013) 281–294.

[16] G. Bektur, T. Saraç, A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server, Comput. Oper. Res. 103 (2019) 46–63.

[17] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G.R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, in: Annals of discrete mathematics, Vol. 5, Elsevier, 1979, pp. 287–326, http://dx.doi.org/10.1016/S0167-5060(08)70356-X.

[18] S.-Y. Kim, Y.-H. Lee, D. Agnihotri, A hybrid approach to sequencing jobs using heuristic rules and neural networks, Prod. Plan. Control 6 (5) (1995) 445–454.

[19] JMS. Valente, Improving the performance of the ATC dispatch rule by using workload data to determine the lookahead parameter value, Int. J. Prod. Econ. 106 (2) (2007) 563–573.

[20] B. Min, C.O. Kim, State-dependent parameter tuning of the apparent tardiness cost dispatching rule using deep reinforcement learning, IEEE Access 10 (2022) 20187–20198.

[21] J.Y. Chen, M.E. Pfund, J.W. Fowler, D.C. Montgomery, T.E. Callarman, Robust scaling parameters for composite dispatching rules, IIE Trans. 42 (11) (2010) 842–853.

[22] M.F. Tasgetiren, Q.-K. Pan, Y.-C. Liang, A discrete differential evolution algorithm for the single machine total weighted tardiness problem with sequence dependent setup times, Comput. Oper. Res. 36 (6) (2009) 1900–1915.

[23] H.J. Shin, C.-O. Kim, S.S. Kim, A Tabu search algorithm for single machine scheduling with release times, due dates, and sequence-dependent set-up times, Int. J. Adv. Manuf. Technol. 19 (11) (2002) 859–866.

[24] G. Bengu, A simulation-based scheduler for flexible flowlines, Int. J. Prod. Res. 32 (2) (1994) 321–344.

[25] Y.-H. Kang, S.-S. Kim, H.J. Shin, A scheduling algorithm for the reentrant shop: an application in semiconductor manufacture, Int. J. Adv. Manuf. Technol. 35 (5–6) (2007) 566–574.

[26] J. Chen, F.F. Chen, Adaptive scheduling in random flexible manufacturing systems subject to machine breakdowns, Int. J. Prod. Res. 41 (9) (2003) 1927–1951.

[27] N. Ye, E.S. Gel, X. Li, T. Farley, Y.-C. Lai, Web server QoS models: applying scheduling rules from production planning, Comput. Oper. Res. 32 (5) (2005) 1147–1164.

[28] L.Y. Tseng, Y.H. Chin, S.C. Wang, The anatomy study of high performance task scheduling algorithm for grid computing system, Comput. Stand. Interfaces 31 (4) (2009a) 713–722.

[29] F.J. Gil-Gala, C. Mencía, M.R. Sierra, R. Varela, Evolving priority rules for on-line scheduling of jobs on a single machine with variable capacity over time, Appl. Soft Comput. 85 (2019) 105782.
[30] F. Mallor, I.G. Guardiola, The Weibull scheduling index for client driven manufacturing processes, Int. J. Prod. Econ. 150 (2014) 225–238.
[31] X. Sun, J.S. Noble, C.M. Klein, Single-machine scheduling with sequence dependent setup to minimize total weighted squared tardiness, IIE Trans. 31 (2) (1999) 113–124.
[32] S.H. Yoon, I.S. Lee, New constructive heuristics for the total weighted tardiness problem, J. Oper. Res. Soc. 62 (1) (2011) 232–237.
[33] K. Caskey, R.L. Storch, Heterogeneous dispatching rules in job and flow shops, Prod. Plan. Control 7 (4) (1996) 351–361.
[34] P. Valledor, A. Gomez, P. Priore, J. Puente, Solving multi-objective rescheduling problems in dynamic permutation flow shop environments with disruptions, Int. J. Prod. Res. 56 (19) (2018) 6363–6377.
[35] X. Sun, J.S. Noble, An approach to job shop scheduling with sequence-dependent setups, J. Manuf. Syst. 18 (6) (1999) 416–430.
[36] J. Chen, F.F. Chen, Adaptive scheduling and tool flow control in flexible job shops, Int. J. Prod. Res. 46 (15) (2008) 4035–4059.
[37] M.S. Jayamohan, C. Rajendran, New dispatching rules for shop scheduling: A step forward, Int. J. Prod. Res. 38 (3) (2000) 563–586.
[38] P.C. Luo, H.Q. Xiong, B.W. Zhang, J.Y. Peng, Z.F. Xiong, Multi-resource constrained dynamic workshop scheduling based on proximal policy optimisation, Int. J. Prod. Res. (2021) 1–19.
[39] T.F. Ho, *.RK. Li, Heuristic dispatching rule to maximize TDD and IDD performance, Int. J. Prod. Res. 42 (24) (2004) 5133–5147.
[40] K.S. Abdallah, J. Jang, Family splitting algorithm for a single machine total tardiness scheduling problem with job family setup times, Int. J. Ind. Eng.-Theory Appl. Pract. 26 (4) (2019) 452–470.
[41] S. Salama, T. Kaihara, N. Fujii, D. Kokuryo, Dispatching rules selection mechanism using support vector machine for genetic programming in job shop scheduling, IFAC-PapersOnLine 56 (2) (2023) 7814–7819.
[42] T.A.A. Kasper, M.J. Land, R.H. Teunter, Towards system state dispatching in high-variety manufacturing, Omega (Westport) 114 (2023) 102726.
[43] H. Wang, T. Peng, A. Nassehi, R. Tang, A data-driven simulation–optimization framework for generating priority dispatching rules in dynamic job shop scheduling with uncertainties, J. Manuf. Syst. 70 (2023) 288–308.
[44] M. Đurasević, F.J. Gil-Gala, D. Jakobović, CA. Coello Coello, Combining single objective dispatching rules into multi-objective ensembles for the dynamic unrelated machines environment, Swarm Evol. Comput. 80 (2023) 101318.
[45] M. Đurasević, F.J. Gil-Gala, L. Planinić, D. Jakobović, Collaboration methods for ensembles of dispatching rules for the dynamic unrelated machines environment, Eng. Appl. Artif. Intell. 122 (2023) 106096.
[46] Y. Gui, D. Tang, H. Zhu, Y. Zhang, Z. Zhang, Dynamic scheduling for flexible job shop using a deep reinforcement learning approach, Comput. Ind. Eng. 180 (2023) 109255.
[47] H. Xiong, H. Wang, S. Shi, K. Chen, Comparison study of dispatching rules and heuristics for online scheduling of single machine scheduling problem with predicted release time jobs, Expert Syst. Appl. 243 (2024) 122752.
[48] E. Ghaedy-Heidary, E. Nejati, A. Ghasemi, S.A. Torabi, A simulation optimization framework to solve stochastic flexible job-shop scheduling problems—Case: Semiconductor manufacturing, Comput. Oper. Res. 163 (2024) 106508.
[49] M. Skutella, G.J. Woeginger, A PTAS for minimizing the total weighted completion time on identical parallel machines, Math. Oper. Res. 25 (1) (2000) 63–75.
[50] Y. Park, S. Kim, Y.-H. Lee, Scheduling jobs on parallel machines applying neural network and heuristic rules, Comput. Ind. Eng. 38 (1) (2000) 189–202.
[51] D.-H. Eom, H.-J. Shin, I.-H. Kwun, J.-K. Shim, S.-S. Kim, Scheduling jobs on parallel machines with sequence-dependent family set-up times, Int. J. Adv. Manuf. Technol. 19 (12) (2002) 926–932.
[52] L. Mönch, J. Zimmermann, P. Otto, Machine learning techniques for scheduling jobs with incompatible families and unequal ready times on parallel batch machines, Eng. Appl. Artif. Intell. 19 (3) (2006) 235–245.
[53] L. Li, F. Qiao, Q.D. Wu, ACO-based multi-objective scheduling of parallel batch processing machines with advanced process control constraints, Int. J. Adv. Manuf. Technol. 44 (9–10) (2009) 985–994.
[54] L.-Y. Tseng, Y.-H. Chin, S.-C. Wang, A minimized makespan scheduler with multiple factors for grid computing systems, Expert Syst. Appl. 36 (8) (2009b) 11118–11130.
[55] R. Driessel, L. Mönch, Variable neighborhood search approaches for scheduling jobs on parallel machines with sequence-dependent setup times, precedence constraints, and ready times, Comput. Ind. Eng. 61 (2) (2011) 336–345.
[56] J. Lamothe, F. Marmier, M. Dupuy, P. Gaborit, L. Dupont, Scheduling rules to minimize total tardiness in a parallel machine problem with setup and calendar constraints, Comput. Oper. Res. 39 (6) (2012) 1236–1244.
[57] M.J. Anzanello, F.S. Fogliatto, L. Santos, Learning dependent job scheduling in mass customized scenarios considering ergonomic factors, Int. J. Prod. Econ. 154 (2014) 136–145.
[58] H. Su, M. Pinedo, G. Wan, Parallel machine scheduling with eligibility constraints: A composite dispatching rule to minimize total weighted tardiness, Nav. Res. Logist. 64 (3) (2017) 249–267.
[59] M. Vimala Rani, M. Mathirajan, Performance evaluation of due-date based dispatching rules in dynamic scheduling of diffusion furnace, OPSEARCH 57 (2) (2020) 462–512.
[60] G. Muratore, U.M. Schwarz, G.J. Woeginger, Parallel machine scheduling with nested job assignment restrictions, Oper. Res. Lett. 38 (1) (2010) 47–50.
[61] H.-C. Hwang, S.Y. Chang, K. Lee, Parallel machine scheduling under a grade of service provision, Comput. Oper. Res. 31 (12) (2004) 2055–2061.
[62] D. Hermelin, J.-M. Kubitza, D. Shabtay, N. Talmon, G.J. Woeginger, Scheduling two agents on a single machine: A parameterized analysis of NP-hard problems, Omega (Westport) 83 (2019) 275–286.
[63] J.-F. Chen, Scheduling on unrelated parallel machines with sequence- and machine-dependent setup times and due-date constraints, Int. J. Adv. Manuf. Technol. 44 (11–12) (2009) 1204–1212.
[64] A. Bilyk, L. Mönch, A variable neighborhood search approach for planning and scheduling of jobs on unrelated parallel machines, J. Intell. Manuf. 23 (5) (2012) 1621–1635.
[65] K. Jaklinović, M. Đurasević, D. Jakobović, Designing dispatching rules with genetic programming for the unrelated machines environment with constraints, Expert Syst. Appl. 172 (2021) 114548.
[66] Y.H. Lee, M. Pinedo, Scheduling jobs on parallel machines with sequence-dependent setup times, Eur. J. Oper. Res. 100 (3) (1997) 464–474, http://dx.doi.org/10.1016/S0377-2217(95)00376-2.