Decomposition approaches for parallel machine scheduling of step-deteriorating jobs to minimize total tardiness and energy consumption

Xiao Wu^{1,2} · Peng Guo^{1,2} · Yi Wang³ · Yakun Wang¹

Received: 25 June 2021 / Accepted: 26 November 2021 / Published online: 13 December 2021
© The Author(s) 2021

Abstract

In this paper, an identical parallel machine scheduling problem with step-deteriorating jobs is considered to minimize the weighted sum of tardiness cost and extra energy consumption cost. In particular, the actual processing time of a job is assumed to be a step function of its starting time and its deteriorating threshold. When the starting time of a job is later than its deteriorating threshold, the job faces two choices: (1) maintaining its status in holding equipment and being processed with a base processing time and (2) consuming an extra penalty time to finish its processing. The two work patterns need different amounts of energy consumption. To implement energy-efficient scheduling, the selection of the pre-processing patterns must be carefully considered. In this paper, a mixed integer linear programming (MILP) model is proposed to minimize the total tardiness cost and the extra energy cost. Decomposition approaches based on logic-based Benders decomposition (LBBD) are developed by reformulating the studied problem into a master problem and some independent sub-problems. The master problem is relaxed by only making assignment decisions. The sub-problems are to find optimal schedules in the job-to-machine assignments given by the master problem. Moreover, MILP and heuristic based on Tabu search are used to solve the sub-problems. To evaluate the performance of our methods, three groups of test instances were generated inspired by both real-world applications and benchmarks from the literature. The computational results demonstrate that the proposed decomposition approaches can compute competitive schedules for medium- and large-size problems in terms of solution quality. In particular, the LBBD with Tabu search performs the best among the suggested four methods.

Keywords Scheduling · Deteriorating jobs · Parallel machine · Energy consumption · Mixed integer programming · Benders decomposition

Introduction

In traditional scheduling theory, it is assumed that the processing time of a job is known in advance and kept fixed throughout the whole operation process [41]. However, there are many practical situations where any delay or waiting to process a job may increase its processing time. Take a prilling process of urea for example. In a urea prilling process, the concentrated urea melt is fed to a prilling device located at the top of the prilling tower. Liquid droplets are formed, solidified, and cooled on free fall through the tower against a forced or natural up-draft of the ambient air [42]. However, the concentrated urea melt may be delayed because previous processes are not synchronized before feeding into the prilling tower. During the waiting process, its temperature may drop below a certain level, and it has to be reheated for prilling. The extra reheating time can be regarded as a

✉ Peng Guo
pengguo318@swjtu.edu.cn

Xiao Wu
wuxiao@swjtu.edu.cn

Yi Wang
ywang2@aum.edu

Yakun Wang
wangyakun1125@hotmail.com

¹ School of Mechanical Engineering, Southwest Jiaotong University, Chengdu, China

² Technology and Equipment of Rail Transit Operation and Maintenance Key Laboratory of Sichuan Province, Chengdu 610031, China

³ Department of Mathematics, Auburn University at Montgomery, Montgomery, AL 36117, USA

penalty for its delayed processing. The processing time of the prilling operation depends on its starting time and its temperature cooling threshold. Alternatively, its temperature is monitored, and heat preservation holding equipment is needed to maintain its status. In this case, the temperature of the urea melt is always suitable for processing. In either case, the extra operations have an effect on the product delivery time and the production energy consumption.

In the above example, the processing time of the concentrated urea melt depends on its starting time, and the corresponding problem is called the time-dependent scheduling problem [2,20,21]. Among various time-dependent scheduling models, a piece-wise defined function (piece-wise linear deterioration or step-wise deterioration) can be used to describe a job's processing time. Other application scenarios can be found in financial management, steel production, equipment maintenance, medicine treatment et al. More details on applications of time-dependent scheduling can be found in Gawiejnowicz [20] (Section 6.6). Besides, the urea prilling processing example involves whether the heat preservation operation is adopted. If heat preservation is adopted, the extra energy consumption should be considered in a scheduling optimization decision.

With the shortage of fuel resources and large amounts of CO₂ emission, the manufacturing industry faces intense pressure in energy saving and emission reduction. Specifically, manufacturing accounts for up to 55% of the total energy consumption in China [35]. Moreover, China was by far the biggest driver of energy consumption, accounting for more than three-quarters of net global growth [9]. Due to public consciousness on environmental effects, manufacturing companies have to adopt efficient approaches to reduce energy consumption and carbon emissions while fulfilling their orders. For the example mentioned above, it is meaningful to consider the energy consumption in determining the schedule of a prilling process. From the view of energy consumption, it is necessary to find a trade-off between the two pre-processing patterns: heat preservation and reheating. A suitable selection of a pre-processing pattern can reduce energy consumption. Furthermore, minimizing the total tardiness is also very important for the proposed scheduling problem. As scheduling with fixed delivery dates are often found when a factory relies on a timetable to deliver its products to customers [33].

Motivated by the above application scenarios, we consider an identical parallel machine scheduling problem in which the processing time of a job is described via a step-wise deterioration. We aim to propose a suitable solution approach capable of finding an optimal schedule that minimizes the objectives of total tardiness and total energy consumption simultaneously in a reasonable amount of runtime. *The studied problem has not been considered in the literature to the best of our knowledge.* Although, as it is reviewed in the next

section, developing solution approaches based on heuristics, including dispatching rules and meta-heuristics, is one of the main research directions in solving parallel machine scheduling problems, this paper aims to design a logic-based Benders decomposition framework, which is shown to reliably generate feasible solutions with good quality in a reasonable time, decisively outperforming a commercial optimization solver.

To improve the solving speed, we tried to integrate a heuristic based on Tabu search into the framework when solving a sub-problem within the framework of LBBD. Our numerical test results show that the studied parallel machine scheduling problem with step-deteriorating jobs can be solved with the proposed decomposition framework in an acceptable computational time. In addition, a sensitivity analysis on the change of unit energy cost is presented for exploring the trend of the mentioned costs.

The main contributions of this paper can be concluded as follows:

1. A novel parallel machine scheduling problem with step-deteriorating jobs and energy consumption is studied. Taking both step-deteriorating jobs and energy consumption into account, each job's actual processing time depends on its starting time and a pre-processing pattern on a machine. A new mathematical formulation for the studied problem is presented based on a liner ordering paradigm.
2. Decomposition approaches based on logic-based Benders decomposition are developed, in which job-to-machine assignments and job sequencing on a machine are separately optimized in two levels. The generated sub-problems are solved via mixed integer programming and Tabu search. To communicate between the master problem and sub-problems, an optimality cut and a greedy assignment cut are designed for our decomposition framework.
3. The performances of the proposed approaches are verified and the sensitivity of unit energy cost is analyzed. Furthermore, the change trends of the total tardiness cost and the energy cost are discussed.

The remainder of this paper is organized as follows. "Literature review" reviews some germane literature. "Problem description and formulation" contains a detailed description of the problem and presents its mixed integer linear programming formulation. Due to its intractability, "Logic-based Benders decomposition approach" presents a logic-based Benders decomposition framework for solving the problem. "Computational study" presents computational experiments before final remarks are given in "Conclusions".

Literature review

For a general orientation in the field, there are in-depth survey papers on time-dependent scheduling [21] and energy-efficient scheduling [18,19,45]. Our literature review will focus on the following two categories mainly involved in the proposed scheduling problem: (1) parallel machine scheduling problems with step-deteriorating jobs and (2) energy-efficient parallel machine scheduling.

Parallel machine scheduling problems with step-deteriorating jobs

The single machine scheduling problem with step-deteriorating jobs was firstly studied by Sundararaghavan and Kunathur [46], then followed by many researchers. For parallel machine scheduling problems with step-deteriorating jobs, the studied objectives in the literature include total completion time [12,30], total tardiness [24], the total net revenue [39]. Since minimizing total completion time or total weighted tardiness on a single machine with deteriorating threshold has been proved to be NP-hard [11,25], it is native to adopt the heuristics or meta-heuristics to solve parallel machine scheduling problems. Such methods include a variable neighborhood search [12], a hybrid variable neighborhood search algorithm [39], reformulation based on set partitioning [30], population-based cuckoo search algorithm [24]. Recently, some researches related to parallel machine scheduling with time-dependent deterioration have started to model the actual processing situations, including work-related fatigue [55], machine depreciation [15] et al..

Batch scheduling with step-deteriorating jobs where machines can process several jobs simultaneously also received the attention of researchers. In batch processing, the jobs are processed in batches, and the processing time is a step function of its waiting time. Leung et al. [31] considered parallel machine batching scheduling problem by minimizing the sum of completion times for all jobs, and suggested some polynomial-time algorithms for special cases. Besides, variable neighborhood search is also adopted to solve batching scheduling problems [38,40].

Based on the above literature review, there is no literature to consider energy consumption in parallel machine scheduling problems with step-deteriorating jobs. That means the existing models and algorithms for solving parallel machine scheduling problems with deteriorating jobs can not be directly used to handle the situation in question. The production process with step-deteriorating jobs may be sped up or more energy efficient via selecting a different pre-processing pattern, as alluded in the example of a prilling process of urea in “Introduction”. Consequently, the scheduling optimization must consider the extra energy consumption brought by the pre-processing in addition to production efficiency, espe-

cially in the era of rising concerns for energy saving and climate change.

Energy-efficient parallel machine scheduling problems

Due to the serious environmental challenges of global climate change, energy-efficient scheduling has received much attention in manufacturing. As one of the general manufacturing environments, parallel machines have attracted much attention to improving productivity via scheduling. Here, an overview of recent parallel machine scheduling problems that deal with energy consumption is exposed.

In literature, parallel machine scheduling problems involving energy consumption concern objectives such as the total penal cost of tardy jobs and the extra energy consumption of machines [32], the total electricity cost with bounded makespan [10], the total energy consumption and the makespan [3,51,53,58], the total tardiness and total energy consumption [37], the makespan and the total carbon emission [56], the total energy consumption and total completion time [57].

Methods used to solve an energy-related parallel machine scheduling problem include dispatching rules [32], two-stage heuristic algorithm [10,57], a fixed and relax algorithm [43], an imperialist competitive algorithm [37], a memetic differential evolution algorithm [53], the distribution evolution memetic algorithm [56], an evolutionary algorithm [58], and a bi-objective local search algorithm [3].

Although there is no direct literature for the problem studied in this paper, some scheduling studies with the deteriorating effect and energy saving have been published. Huang and Yu [29] studied a two-stage flow shop scheduling problem with deteriorating maintenance and aimed to minimize the makespan with limited energy consumption via particle swarm optimization. Abedi et al. [1] studied the linear deterioration effect on machines for a job-shop scheduling problem and determined the optimal speed level to process each operation for minimizing the total weighted tardiness and total energy consumption simultaneously via a multi-objective memetic algorithm. In Wu et al. [54], the step-deteriorating effect is introduced into the flexible job-shop scheduling problem, and a hybrid pigeon-inspired optimization is developed to optimize the energy consumption without delaying the makespan. Furthermore, Tigane et al. [48] implemented a non-dominated sorting-based genetic algorithm for a parallel unrelated machine scheduling problem with linear deterioration jobs and minimized the makespan and the total energy consumption. These closely related studies aim to develop meta-heuristics for the scheduling problems considering deteriorating effect and energy consumption. Only one paper, among these four papers, formulated the deteriorating effect by step-wise functions. They are significantly differ-

ent from our work regarding the problem description and the designed solution approach.

Problem description and formulation

Problem description

The studied parallel machine scheduling problem with step-deteriorating jobs to simultaneously minimize total tardiness and total energy consumption is dubbed PMSPSDEC. Consider a set $N := \{1, \dots, n\}$ of jobs on a set $M := \{1, \dots, m\}$ of parallel machines. Each machine can handle only one job at a time, and each job cannot be processed on more than one machine simultaneously. No preemption is allowed. All jobs and machines are available at time zero. Furthermore, machines breakdown and their preventive maintenance are not considered in this problem.

The temperature of a job while waiting to be processed will decrease. If the starting time s_j of job j is less than or equal to the deteriorating threshold h_j , its temperature remains suitable for being directly handled by a machine. In this case, job j only requires a processing time a_j . Otherwise, an extra penalty time b_j will be incurred for pre-processing. That is to say, the actual processing time p_j of job j is a step function of its starting time s_j and deteriorating threshold h_j . Alternatively, assistant equipment (such as a holding furnace) is used to keep the job status at a level such that a machine can start to work on it at once if available, but a continuous energy supply is required. In that case, the machine only consumes a base processing time to finish the job.

Either in the case of reheating or using holding equipment, extra energy will be consumed. When using assistant equipment, the amount of the additional energy consumption is calculated: $E_j = e_0 \times \text{Max}\{0, s_j - h_j\}$, where e_0 is the unit energy consumed by the assistant equipment and the second part is the duration of the waiting phase when beyond the deteriorating threshold.

On the other hand, if a job does not use assistant equipment after its deteriorating threshold, then more energy consumption will be needed for its penalty processing time due to the reheating need. Under this situation, the extra energy consumption amount is determined: $E_j = q_j \times b_j$, in which quantity q_j means the unit energy consumption needed by its penalty processing time.

We want to emphasize that extra energy consumption E_j does not include the energy consumption needed by jobs' base processing time. The decision is to assign and schedule the jobs to each machine and select an appropriate pre-processing pattern before handling a job. The objective function is to minimize the weighted sum of total tardiness cost and extra energy consumption cost.

Let T_j denote the tardiness of job j . Using the three-field notation, the problem can be denoted by $Pm|SD, PE|TTC + TEC$, where Pm indicates identical parallel machines; in the second field, SD indicates the step-deteriorating jobs, and PE represents the processing energy consumption; in the third field, the scheduling criterion is indicated that includes total tardiness cost TTC and total extra energy consumption cost TEC .

Regarding the computational complexity, it is known that a single machine scheduling problem of total weighted tardiness with step-deteriorating jobs is shown to be NP-hard in the strong sense [25]. Since our problem is a generalization of a single machine scheduling problem, and an additional objective of minimizing total extra energy consumption is considered, the same complexity status holds at least. Next a mixed integer programming (MIP) model is presented and a toy example is given for the studied problem.

Mathematical formulation

In Table 1, the notations are introduced for the model. Then, a mixed integer linear programming model of the problem under consideration is presented. The proposed MIP model consists of objective function (1) and constraints (2)–(13).

$$\text{Minimize } F = \sum_{j \in N} \alpha_j T_j + \beta \sum_{j \in N} E_j, \quad (1)$$

subject to

$$\sum_{k \in M} y_{jk} = 1 \quad \forall j \in N, \quad (2)$$

$$y_{ik} + y_{jk} \leq 1 + x_{ij} + x_{ji} \quad \forall i, j \in N, i \neq j, k \in M, \quad (3)$$

$$s_i + a_i + b_i \cdot (1 - u_i - z_i) - s_j \leq \Delta \cdot (1 - x_{ij}) \quad \forall i, j \in N, \quad (4)$$

$$s_j + a_j + b_j \cdot (1 - u_j - z_j) - d_j \leq T_j \quad \forall j \in N, \quad (5)$$

$$s_j + \Delta \cdot (z_j - 1) \leq h_j \quad \forall j \in N, \quad (6)$$

$$h_j + \epsilon - \Delta \cdot z_j \leq s_j \quad \forall j \in N, \quad (7)$$

$$u_j + z_j \leq 1 \quad \forall j \in N, \quad (8)$$

$$e_0 \times (s_j - h_j) - \Delta \cdot (1 - u_j) \leq e_j \quad \forall j \in N, \quad (9)$$

$$e_j \leq \Delta \cdot u_j \quad \forall j \in N, \quad (10)$$

$$e_j + q_j \cdot b_j \cdot (1 - u_j - z_j) \leq E_j \quad \forall j \in N, \quad (11)$$

$$x_{ij}, y_{jk}, z_j, u_j \in \{0, 1\} \quad \forall i, j \in N, k \in M, \quad (12)$$

$$T_j, E_j, s_j \geq 0 \quad \forall j \in N, \quad (13)$$

In the above formulation, the objective function (1) minimizes the sum of total tardiness cost and the extra energy cost. Constraint (2) ensures every job is exactly assigned to a machine. Constraint (3) enforces that jobs i and j cannot

Table 1 Notation

Indices	
i, j	Indexes of jobs, $i, j \in N$
k	Index of a machine, $k \in M$
Parameters	
N	Set of all jobs
M	Set of all machines
a_j	Base processing time of job j
h_j	Deteriorating threshold of job j
b_j	Extra penalty processing time of job j
d_j	Due date of job j
e_0	Energy consumption (per time unit) of the holding equipment for a job
q_j	Energy consumption (per time unit) when reheating job j
α_j	Tardiness penalty cost for job j
β	unit energy cost, such as hourly electricity price
Δ	Large positive number
ϵ	Very small positive number
Decision variables	
x_{ij}	Equals to 1 if and only if job i is completed before job j starts, and to 0 otherwise
y_{jk}	Equals to 1 if and only if job j is processed on machine k
z_j	Equals to 1 if and only if job j is started no later than its deteriorating threshold, and to 0 otherwise. When $z_j = 1$, $b_j = u_j = 0$.
u_j	Equals to 1 if and only if the status of job j is maintained by a holding equipment, and to 0 otherwise. When $u_j = 1$, $z_j = b_j = 0$.
Dependent variables	
p_j	Actual processing time of job j . In our models, it can be replaced by $a_j + b_j \times (1 - u_j - z_j)$.
e_j	Extra energy consumed by job j during the waiting phase in a holding machine
E_j	Total extra energy consumption for completing job j
T_j	Tardiness of job j
s_j	Starting time of job j

be processed simultaneously if they are assigned to the same machine. Constraint (4) sets jobs' starting times and ensures that they are harmonized with the sequence variables x_{ij} . The computation of tardiness T_j of each job is given in constraint (5). Constraints (6) and (7) enforce the following: if the starting time of job j ($j \in N$) is less than or equal to the deteriorating threshold ($s_j \leq h_j$), then variables $z_j = 1$, otherwise $z_j = 0$. Constraint (8) ensures u_j and z_j for each $j \in N$ are mutually exclusive, i.e., when $z_j = 1$ then variable $u_j = 0$; and vice versa, when $u_j = 1$, $z_j = 0$. Constraint (9) calculates the energy consumption for preserving the status of a job. If job j does not use holding equipment, the extra energy consumption e_j during the waiting processing phase is set to 0 via constraint (10). In addition to e_j , the energy consumption brought by the penalty processing time is determined by constraint (11). Constraints (12) and (13) define the domains of the decision variables.

Example of a schedule

Consider an example of PMSPSDEC depicted in Table 2. There are $n = 5$ jobs and $m = 2$ machines. Moreover, let $e_0 = 20$ be the energy consumed by the holding equipment per unit time and $\beta = 0.01$ be the unit energy cost. An optimal solution is shown in Fig. 1. Jobs 1 and 2 are assigned to machine 1 and jobs 3, 4 and 5 are assigned to machine 2. In the optimal solution, job 1 is processed after its deteriorating threshold, and its processing status is kept by holding equipment. Job 5 is also processed after its deteriorating threshold, but it adopts an extra penalty time to decrease the impact on the objective function. In that case, jobs 1 and 5 have tardiness values of $8 = (19 - 11)$ and $11 = (31 - 20)$, respectively. Therefore, the total weighted tardiness cost is $13.5 = ((19 - 11) \times 1 + (31 - 20) \times 0.5)$. On the other hand, the extra energy consumption is $82 = ((10 - 6) \times 20 + 1 \times 2)$. Therefore the solution illustrated in Fig. 1 has an objective value of $13.5 + 82 \times 0.01 = 14.32$.

Table 2 Example problem data

Job(<i>j</i>)	1	2	3	4	5
Base processing time a_j	9	10	8	7	15
Deteriorating threshold h_j	6	5	8	9	10
Penalty time b_j	3	5	8	7	1
Due date d_j	11	13	15	14	20
unit processing energy consumption q_j	2	2	3	4	2
tardiness penalty cost α_j	1	2	2.5	2	0.5

Fig. 1 The Gantt chart of an optimal solution for the toy example **Fig. 2** The Gantt chart of an optimal solution for the toy example without using holding equipment

To highlight the significance of adopting an alternative approach of using a holding machine, the above example without considering holding equipment is also solved. The corresponding optimal solution is shown in Fig. 2. The optimal assignment is the same as the previous situation. However, job 1 must be penalized because its starting time is later than its deteriorating threshold. In this case, the energy consumption is $6 = 2 \times 3$. The extra energy consumption for job 5 is $2 = 2 \times 1$. Therefore, the objective value is $16.58 = (22 - 11) \times 1 + (31 - 20) \times 0.5 + (6 + 2) \times 0.01$. Compared with the previous situation, tardiness becomes significantly more prominent when there is no holding equipment. The above result demonstrates the alternative approach of adopting holding equipment is beneficial to fulfill the orders for manufacturing factories while maintaining a lower cost.

Logic-based Benders decomposition approach

Logic-based Benders decomposition (LBBD) is a substantial generalization of classical Benders decomposition that,

Fig. 3 A flowchart of the proposed LBBD framework

in principle, allows a sub-problem to be any optimization problem rather than specifically a linear or nonlinear programming problem. LBBD provides a natural means to combine different kinds of problem formulations and solvers [28]. For example, the master problem might be amenable to a MIP formulation and solver, while the sub-problems might be better suited for constraint programming (CP). The MIP/CP combination is probably the most popular because problems frequently are decomposed into an assignment problem suitable for MIP and a scheduling problem on which CP methods tend to excel. Based on the framework of LBBD, different solution approaches for the master problem and sub-problems may be designed separately. Due to its flexibility in solving a sub-problem, LBBD has been used to solve various complex combination optimization problems, including inventory location [52], home healthcare delivery [27], parallel machine scheduling [34,47,49], integrated process planning and scheduling [4], gantry crane scheduling [26] and so on. Furthermore, LBBD, as a decomposition framework, can be easily stretched using heuristics to solve its master problem or sub-problems.

We keep the assignment decision in the master problem, and the sequence decision is left to the sub-problems for the problem in question. A sub-problem always has a feasible solution, but such a solution may not be optimal from the view of the entire problem. If the sum of the solution values of all sub-problems is equal to that of the master problem, then an optimal solution is found, and the LBBD algorithm terminates. Otherwise, a Benders optimality cut is developed and added to the master problem. In our problem, there are no feasibility Benders cut because a subproblem always has a feasible solution. Figure 3 describes the general scheme for the logic-based Benders decomposition for our problem. Now, we describe the details of the proposed LBBD approaches.

The master problem

The master problem (MP) aims at determining an assignment of jobs to machines. Therefore, the assignment constraint is considered as the key part of the MP. Besides, the relaxed sub-problems should be contained in the MP to improve a lower bound [28]. First, we used the MIP provided in “Mathematical formulation” to form the mathematical model of the MP, as follows.

$$\text{(Master-Problem) Minimize } F, \quad (14)$$

subject to

$$\sum_{k \in M} y_{j,k} = 1 \quad \forall j \in N, \quad (15)$$

$$y_{ik} + y_{jk} \leq 1 + x_{ij} + x_{ji} \quad \forall i, j \in N, i \neq j, k \in M, \quad (16)$$

$$F \geq \sum_{j \in N} \alpha_j T_j + \beta \sum_{j \in N} E_j \quad \forall j \in N, \quad (17)$$

$$\text{relaxed sub-problems with } 0 \leq x_{ij}, z_j, u_j \leq 1, \quad \forall i, j \in N, \quad (18)$$

$$y_{j,k} \in \{0, 1\} \quad \forall i, j \in N, k \in M. \quad (19)$$

Objective function (14) is the same as that of the MIP. Constraint (15) ensures that each job is assigned to exactly one machine. Constraint (16) gives the link of the assignment variable y_{jk} and the sequence variable x_{ij} . Constraint (17) determines a lower bound of the objective. The tardiness (T_j) and extra energy consumption (E_j) in (17) are calculated using the relaxed sub-problems (18).

For the decision variables, only assignment variable y_{jk} is binary and variables x_{ij} , z_j and u_j are relaxed to be continuous as shown in Constraint (19).

Sub-problems

After solving the MP and determining a job-to-machine assignment, we have a set of values for variables y_{jk} , $j \in N, k \in M$. Now, consider these values as input parameters to the sub-problems. Let v indicate the current iteration of the Benders decomposition approach. For each machine k , we generate a sub-problem based on the value of the binary variables y_{jk} , $j \in N$. The sequence of jobs on a machine does not affect another machine for a given assignment. Thus, we can solve m sub-problems independently and in parallel. Let $N_{vk} = \{j | y_{jk} = 1\}$ be the set of jobs assigned to machine k in iteration v . Each generated sub-problem is a single machine scheduling problem and can be modeled similarly to a time-dependent traveling salesman problem (TDTSP) [5,44]. Since the processing time of a job depends on its starting time, unfortunately, we cannot use the most powerful TSP solver-Concorde. Consequently, we propose

two solution approaches here to solve it. Next, we present a mixed integer programming of a sub-problem for a given machine k .

Mixed integer programming model

Here, a linear ordering (LO) formulation is adopted to describe the single machine scheduling problem. The LO formulation is presented as follows.

$$\text{(Sub-Problem MIP) Minimize}$$

$$F_k = \sum_{j \in N_{vk}} \alpha_j T_j + \beta \sum_{j \in N_{vk}} E_j, \quad (20)$$

subject to

constraints (5)–(11), and additionally,

$$x_{ij} + x_{ji} = 1 \quad \forall i, j \in N_{vk} (i \neq j), \quad (21)$$

$$x_{ij} + x_{jl} + x_{li} \leq 2 \quad \forall i, j, l \in N_{vk} (i \neq j, j \neq l, i \neq l), \quad (22)$$

$$\sum_{i \in N_{vk}} p_i x_{ij} \leq s_j \quad \forall j \in N_{vk}, \quad (23)$$

$$x_{ij}, z_i, u_i \in \{0, 1\} \quad \forall i, j \in N_{vk}, \quad (24)$$

$$T_j, E_j, s_j \geq 0 \quad \forall j \in N_{vk}. \quad (25)$$

In the above sub-problem, constraint (21) ensures that either job i is positioned before job j or vice versa. Constraint (22) enforces the transitivity constraints that guarantee a linear ordering between three jobs i , j and l . Constraint (23) determines the starting time of job j based on its predecessors. Note that constraint (23) is non-linear due to the quadratic term $p_i \cdot x_{ij}$. The product term between a binary variable x_{ij} and a continuous variable p_i can be linearized via the method in the paper [25]. The above three constraints aim to determine a sequence of jobs and provide an equivalent expression of the disjunctive constraints in the original MIP model proposed in Sect. 3.2. Based on our preliminary tests, the above linear ordering formulation performs better than directly using the original MIP model in solving sub-problems.

As shown by the domains of the decision variables defined in (24), we still have to deal with three sets of binary variables x_{ij} , z_j and u_j , which still make challenging to solve the sub-problems based on a standard solver. Ideally, the LBB needs the solution information of sub-problems generated in a short time. To solve a sub-problem efficiently, the most common approach is to use a constraint programming technique in logic-based Benders decomposition [13,14,26]. However, we tried to integrate the CP model into the decomposition framework and found CP still does not perform satisfactorily.

In principle, as long as the sub-problems provide smaller upper bounds of F , while the master problem finds larger

lower bounds, then the LBB iterations proceed in the converging direction. Bearing this in mind, we designed a heuristic based on Tabu search to solve a sub-problem in the next section. Adopting a heuristic leads to efficiently solving the subproblems while maintaining the LBB iterations being converging in the right direction.

Tabu search

Tabu search uses a local neighborhood search structure to iteratively move from one solution to an improved one until some stopping criterion has been met. To avoid being trapped in a local minimum, Tabu search carefully explores the neighborhood of each solution through the use of a tabu list. A tabu list is built to forbid the selection of already visited solutions. Tabu search uses an aspiration criterion to accept an improved solution and override the tabu status when the search process finds an improved solution. Since the Tabu search was invented, it has been used to solve various combinatorial optimization problems, including single machine total tardiness problem [6,8,16,50]. If we use Tabu search to solve a sub-problem, the proposed decomposition algorithm becomes a heuristic without the promise of finding an optimal solution.

The performance of Tabu search depends on the quality of an initial solution and the neighborhood search strategy. Here, we used a heuristic based on *sample dispatching rule* (dubbed HBSD) to determine an initial sequence. First, the job with the smallest due date is chosen to occupy the first position of the sequence. The completion time of that job is used to calculate the increase to the objective function by appending a job from the remaining unscheduled jobs. Then the job with the smallest increment of the objective function value is chosen to occupy the next position. The selecting operation is repeated until there is no un-scheduled job. The procedure details of the heuristic is described in Algorithm 1.

Algorithm 1 The heuristic based on sample dispatching rule

Require: the parameters of the jobs in N_{vk}

Ensure: the jobs sequence π_{vk} on machine k

- 1: $\pi_{vk} \Rightarrow \{\}$
 - 2: Find the job j with the smallest due date
 - 3: $\pi_{vk} \Rightarrow \pi_{vk} \cup \{j\}$ and $N_{vk} \Rightarrow N_{vk} \setminus \{j\}$
 - 4: $c_j = a_j$
 - 5: **while** N_{vk} is non-empty **do**
 - 6: Update the completion time c_j ($j \in N_{vk}$) based on the completion time of the last job in π_{vk}
 - 7: Calculate the increment f_{ij} of the objective function with tardiness and energy consumption by appending a job from the set of unscheduled jobs based on c_j ($j \in N_{vk}$)
 - 8: Find the job with the smallest value f_{ij} from N_{vk}
 - 9: $\pi_{vk} \Rightarrow \pi_{vk} \cup \{j\}$ and $N_{vk} \Rightarrow N_{vk} \setminus \{j\}$
 - 10: **end while**
-

To generate a neighbor, several neighborhood operators are available for single machine scheduling problems [23]. Here, an *adjacent pairwise swap* is adopted since it is straightforward. The tabu list is defined as a circular First In First Out (FIFO) queue of fixed length. The length of the tabu list is set at $2\sqrt{|N_{vk}|}$. The algorithm is stopped via the maximum number of non-improving moves. If the best solution is not improved in subsequent $|N_{vk}|$ iterations, the algorithm stops. The above two parameters of the Tabu search algorithm were tested in Eren and Güner [16].

In the above neighborhood operations, a permutation of jobs is used to express the solution of sub-problem k . Once the sequence is obtained, the objective value of the specific sub-problem can be determined by a decoding scheme. Before we state a decoding scheme, we prove a property for job j in a job sequence when its starting time is later than the deteriorating threshold.

Proposition 1 *When the starting time s_j of job j is later than its deteriorating threshold h_j , if $e_0 \times (s_j - h_j) \leq b_j \times q_j$, then job j must be kept in holding equipment during the waiting phase in an optimal schedule.*

Proof When $e_0 \times (s_j - h_j) \leq b_j \times q_j$, if adopting the penalty processing pattern, the extra energy consumption brought into the objective function is greater than that of a solution if job j using holding equipment. Thus, in this case, it is clear that job j should be kept in holding equipment from the perspective of minimizing the cost related to the extra energy consumption.

Meanwhile, in this case, job j would consume only a base processing time a_j when using holding equipment to maintain its status. While if adopting the penalty processing pattern, the processing time p_j of job j becomes $a_j + b_j$ and thus, it is possible to cause an additional increase of the objective function value in terms of tardiness.

Therefore, adopting holding equipment for job j to maintain the job's status must be optimal in a schedule if the energy condition stated in the proposition is met. $\square$

The above property can be used to determine a decoding scheme for job sequence in Tabu search: the actual processing time of job j is just a base processing time if its starting time s_j is less than or equal to its deteriorating threshold h_j . If $s_j > h_j$ and $e_0 \times (s_j - h_j) \leq b_j \times q_j$, the job is kept in a holding equipment, otherwise it has to accept the penalty processing and the actual processing time is $a_j + b_j$. Once all jobs in the sequence list are scanned, the objective value can be obtained for sub-problem k . We must note the fact that the above decoding strategy is short-sighted because only one of the remaining jobs is analyzed each time by its effect on the increment of the objective function during the Tabu search. Thus, the obtained Tabu-searched solution may not be optimal for the corresponding sub-problem. This claim

will be verified later in the computational results delivered by the algorithms with sub-problems solved by Tabu search.

Benders cut

After solving all the sub-problems, the LBBD algorithm ends up with either optimality or near-optimality. When the objective function value obtained by the master problem and the sum of the sub-problems are identical, the algorithm reaches optimality and terminates. When the sum of the objective values of the SPs is greater than the objective value of the MP, the SPs deliver an upper bound, and the MP provides a lower bound. In this case, the algorithm has to continue the search for a better solution by adding a Benders optimality cut into the MP. The LBBD algorithm then continues solving the master model with the new cut. When it hits upon another assignment (integer solution), the sub-problems are again solved, a new cut is generated, and the MP is solved until no more unexplored and un-fathomed solutions remain for the MP or the objective value of the MP reaches the sum of that of the SPs.

In our problem, as the SPs are always feasible, there is no need to generate any feasibility cut during the optimizing process. To improve the objective value of the MP and/or generate different assignments for SP, we developed the following two types of Benders cuts.

Optimality cut (dubbed OC):

$$F \geq \sum_{k \in M} (\bar{F}_{vk} - \bar{F}_{vk} \cdot \sum_{j \in N_{vk}} (1 - y_{j,k})). \quad (26)$$

In constraint (26), $\bar{F}_{v,k}$ denotes the corresponding objective value incurred by machine k at iteration v . The above cut enforces either a different job-to-machine assignment in the MP is generated, or the objective value F must be non-decreasing.

Proposition 2 *Constraint (26) is a valid Benders optimality cut.*

Proof A valid Benders optimality cut must hold two properties: 1) the cut should not remove any integer solution; and 2) the objective of the incumbent solution of the MP is bounded by the optimal solution of the SP. It is clear that Constraint (26) does not cut off any unexplored assignment solution and only limits the objective value of the incumbent solution of the MP (i.e., a solution with the same set of assignments) to the sum of the optimal solution values delivered by SPs. $\square$

In addition to the Benders cut (26), a greedy assignment cut (dubbed GC)

$$\sum_{j \in N_{vk'}} y_{jk'} \leq |N_{vk'}| - 1, \quad k' := \arg \max_{k \in M} F_k \quad (27)$$

is added to the master problem to accelerate the iteration. The constraint enforces the set of jobs assigned to the machine with the largest portion in the objective value that must remove a job in the next iteration. Although we considered the cut mentioned in Tadumadze et al. [47] to change the set of jobs assigned to every machine by removing a job, the numerical test reports that it does not perform better than the greedy cut (27). We want to point out that constraint (27) does not warrant an optimal solution as it does not consider the impact on the objective function value by removing a job from other machines, which potentially can lead to a better job-to-machine assignment. This suboptimality of the GC will be illustrated in the computing experiments in “Computational results”.

Furthermore, an upper bound cut is added to the MP once a new upper bound has been found to warrant decreasing upper bounds. Denote by UB^* the currently found best upper bound from solving sub-problems. Let $UB^v := \sum_{k \in M} F(N_{vk})$ be an upper bound of the objective value obtained by the sub-problems at iteration v . The upper bound constraint of

$$F \leq \min\{UB^*, UB^v\}$$

is added into the MP model.

In this way, those solutions of MP model which are not less than the best upper bound $\min\{UB^*, UB^v\}$ will be ruled out in further searches. Similarly, if LB_μ represents the lower bound obtained by the MP at iteration μ , the following lower bound cut

$$F \geq LB_\mu, \quad \mu = 1, \dots, v$$

is added into the MP.

The above four cuts are iteratively added to the MP model and once a new job-to-machine assignment is found in the MP, the sub-problems are solved until either that the lower bound from the MP meets the upper bound from the SPs, or that no more new and un-fathomed solutions remain for the MP model. By that time, the algorithm terminates and the best upper bound is optimal. If the algorithm is interrupted in advance by an imposed stopping criterion, the found best upper bound provides a feasible solution.

Initial solution for LBBD

Generally, a good initial solution for LBBD can improve its performance. Here, a dispatching rule mentioned in Algorithm 1 is further extended to a parallel machine scheduling environment. As in a single machine, the first job is chosen using the earliest due date, and it is assigned to any machine. Then, the earliest available machine k is found and the increment of the objective function caused by adding a job from the remaining jobs are calculated based on the completion

time of the last scheduled job on machine k as in step 6 of Algorithm 1. The job with the smallest increment is chosen and is assigned to machine k . After that, the related parameters are updated. The above procedure is iterated until the set of un-scheduled jobs becomes empty. Once the above approach is made, we can obtain a feasible solution for the original problem. We use the solution to set the start values of the related decision variables y , x , u and z .

Computational study

This section aims to evaluate the performance of the proposed algorithms. However, there are no standard benchmark instances for the PMSPSDEC. We generated three groups of instances to perform the computational experiments, including small-sized, medium-sized and large-sized sets based on our knowledge of a local fertilizer plant. The instance generation is described in “Instance generation”. Tests on the computational performance of the proposed MIP model and our decomposition methods are detailed in “Computational results”. All methods are implemented using the C# programming language (.Netcore 2.1) and run on a PC with an Intel i7-4600U 2.10 GHz CPU and 16 GB of RAM. All MIP

models are solved by Gurobi 9.0. The proposed logic-based Benders decomposition algorithms are implemented using the lazy constraint callback function available in Gurobi in which cuts are added to the master problem when an integer solution to the job-to-machine assignment is found in the MP.

Instance generation

In this section, three groups of instances are generated based on the number of jobs suggested by Gacias et al. [17], Ozer and Sarac [36]. The number n of jobs takes 8 with $m = \{2, 3\}$ for small-sized problems, takes 40 with $m = \{2, 6\}$ for medium-sized problem and takes 100 with $m = \{2, 8\}$ for large-sized problems. This scheme of generating instances is an adaptation of the scheme proposed by Guo et al. [25] for tardiness minimization scheduling problems. The problem instances, as well as the detailed computational results for every instance, are available by visiting the following link: <https://github.com/pengguo318/PMSPSDEC>.

For each job, the base processing time a_j is drawn from a uniform distribution over the interval $[1, 100]$. The penalty time b_j is picked up from a uniform distribution on $[1, 10]$. The deteriorating threshold h_j is generated randomly from the uniform distribution over the interval $H := [1, A]$,

Table 3 Computational results of small instances

Instance		MIP	LBBD-MIP-OC	LBBD-MIP-GC	LBBD-TS-OC	LBBD-TS-GC
Size	Group					
8×2	1	0/5/0.90	0/5/3.74	0/5/1.83	0/5/0.12	0/5/0.05
	2	0/5/0.45	0/5/3.72	0/5/2.09	0/5/0.11	0/5/0.06
	3	0/5/0.47	0/5/4.24	0.22/4/2.54	1.25/2/0.10	1.47/1/0.06
	4	0/5/0.30	0/5/3.43	0/5/1.89	0/5/0.09	0/5/0.06
	5	0/5/0.42	0/5/3.66	0.14/4/2.34	1.60/2/0.11	1.72/2/0.05
	6	0/5/0.54	0/5/5.54	0.64/4/3.61	8.55/2/0.11	9.37/2/0.05
	7	0/5/0.24	0/5/3.67	0/5/2.75	12.07/4/0.08	12.07/4/0.06
	8	0/5/0.47	0/5/4.76	0.45/4/3.03	0.26/4/0.10	0.71/3/0.06
	9	0/5/0.32	0/5/3.80	0/5/2.74	0/5/0.09	0/5/0.05
	10	0/5/0.40	0/5/2.94	0/5/2.49	6.73/4/0.08	6.73/4/0.05
8×3	1	0/5/0.26	0/5/11.64	0.02/4/1.20	0/5/1.32	0.02/4/0.10
	2	0/5/0.25	0/5/9.39	4.57/3/1.47	0/5/0.53	4.57/3/0.10
	3	0/5/0.09	0/5/3.02	0.88/4/1.48	0/5/0.25	0.88/4/0.09
	4	0/5/0.06	0/5/1.28	0/5/1.58	0/5/0.10	0/5/0.11
	5	0/5/0.41	0/5/12.23	3.51/1/2.33	3.81/4/0.57	7.01/1/0.12
	6	0/5/0.15	0/5/3.99	5.14/3/1.83	0/5/0.26	3.65/3/0.12
	7	0/5/0.05	0/5/1.07	0/5/1.19	0/5/0.09	0/5/0.08
	8	0/5/0.09	0/5/2.29	0/5/1.26	13.96/4/0.16	13.96/4/0.09
	9	0/5/0.10	0/5/2.53	0/5/1.29	0/5/0.19	0/5/0.09
	10	0/5/0.05	0/5/1.11	0/5/1.56	0/5/0.08	0/5/0.08
Avg/Sum/Avg		0/100/0.30	0/100/4.40	0.78/86/2.02	2.41/86/0.23	3.11/75/0.08

Note: The gap concerning optimal solution value/number of optimal solutions found/computational time in seconds

Table 4 Optimal solution values and gap values for some small instances

Instance				MIP	LBB-D-MIP-OC	LBB-D-MIP-GC	LBB-D-TS-OC	LBB-D-TS-GC
Size	Group	Index	Optimal	Gap (%)	Gap (%)	Gap (%)	Gap (%)	Gap (%)
8×2	3	1	239.18	0	0	1.10	0	1.10
	3	2	219.52	0	0	0	3.38	3.38
	3	3	22.9	0	0	0	2.27	2.27
	3	4	160.51	0	0	0	0.60	0.60
	5	1	369.68	0	0	0	1.54	1.54
	5	3	35.58	0	0	0	4.83	4.83
	5	5	113.58	0	0	0.70	1.64	2.23
	6	3	228.58	0	0	0	3.85	3.85
	6	4	25.3	0	0	0	37.67	37.67
	6	5	133.24	0	0	3.22	1.22	5.34
	7	4	2.37	0	0	0	60.34	60.34
	8	1	251.54	0	0	0	1.29	1.29
	8	3	23.49	0	0	2.26	0	2.26
	10	1	74.38	0	0	0	33.66	33.66
8×3	1	5	324.8	0	0	0.11	0	0.11
	2	2	96.2	0	0	22.22	0	22.22
	2	3	368.47	0	0	0.65	0	0.65
	3	3	38.04	0	0	4.42	0	4.42
	5	1	107.81	0	0	1.96	0	0.97
	5	2	100.5	0	0	3.84	0	3.84
	5	3	125.7	0	0	3.20	0	3.20
	5	5	30.99	0	0	8.55	19.04	27.04
	6	4	37.83	0	0	9.99	0	9.99
	6	5	91.36	0	0	15.70	0	8.27
Average	8	1	3.84	0	0	0	69.79	69.79
			–	0.00	0.00	3.12	9.64	12.43

where $A = \sum_{j \in N} a_j / m$. Based on the instance generation method adopted by Biskup et al. [7], the due date d_j is randomly drawn from the uniform distribution on the interval $a_j + [TF \times A, RD \times C'_{max}]$, where C'_{max} is the value of the maximum completion time obtained by scheduling the jobs in the ascending order of ratios $a_j/b_j (j \in N)$ by assigning an unscheduled job to the earliest available machine, and TF and RD are range parameters defined by $TF \in \{0, 0.2, 0.4, 0.6\}$ and $RD \in \{0.2, 0.4, 0.6, 0.8\}$ with $TF < RD$. The tardiness penalty cost α_j is generated from a uniform distribution on $[0.5, 5]$. The unit energy consumption q_j is generated from a uniform distribution on the interval $[1, 6]$. Furthermore, the energy consumption e_0 of the holding equipment is drawn from the uniform distribution $U[20, 40]$. Moreover, the unit energy cost β is set to 0.01. We generated 50 instances for each pair of (n, m) with 5 problems for each pair (TF, RD) . For each combination (n, m) , the 50 instances are divided into 10 representative groups with all possible combinations of the

two parameters TF and RD (denoted $\{\text{index}:(TF, RD)\}$), including $\{1:(0,0.2)\}$, $\{2:(0,0.4)\}$, $\{3:(0,0.6)\}$, $\{4:(0,0.8)\}$, $\{5:(0.2,0.4)\}$, $\{6:(0.2,0.6)\}$, $\{7:(0.2,0.8)\}$, $\{8:(0.4,0.6)\}$, $\{9:(0.4,0.8)\}$, and $\{10:(0.6,0.8)\}$, where $TF < RD$.

Computational results

The time limit for solving a MIP model is set to 600 CPU seconds. Similarly, the same time limit is imposed on the LBB-D when dealing with medium and large instances. Moreover, if MIP solves a sub-problem in the LBB-D, the time limit is set to 10 s for a feasible solution within the 600 s. Note that the 600s time-limit aims at practical applications adopting our proposed approaches and is considered to be acceptable by a shop manager we consulted. Besides, we tested medium-sized instances with a time-limit of 3600 s, of which the computational results are not significantly different from the results using 600-s time-limit. For our proposed algorithms, several abbreviations are used to denote different

Table 5 Computational results of medium instances

Instance		MIP	LBBD-MIP-OC	LBBD-MIP-GC	LBBD-TS-OC	LBBD-TS-GC
Size	Group					
40×2	1	3.68/0	0.58/1	0.40/2	0.34/2	0.73/0
	2	7.57/0	0.62/0	0.54/1	0.41/2	0.38/2
	3	30.10/0	1.35/0	1.59/2	0.42/3	2.35/0
	4	49.03/0	2.91/1	4.19/0	0.60/3	1.72/1
	5	9.85/0	0.43/2	0.62/2	0.77/1	0.89/1
	6	27.86/0	3.13/0	2.22/0	0.33/4	1.78/1
	7	77.58/0	10.41/0	4.37/1	0.58/3	3.57/1
	8	28.19/0	1.68/1	1.24/1	0.10/3	0.74/1
	9	118.10/0	22.74/0	21.88/0	1.55/3	5.96/2
	10	106.03/0	28.24/0	27.76/0	1.25/2	1.30/3
40×6	1	1.02/1	3.12/0	0.97/2	3.36/0	0.24/2
	2	1.02/4	12.42/0	6.52/0	8.56/0	1.97/1
	3	4.52/3	28.97/0	17.24/1	18.71/0	5.83/1
	4	21.01/2	103.02/0	61.81/0	74.87/0	2.49/3
	5	3.96/0	7.55/0	5.28/0	4.55/0	0/5
	6	7.01/1	24.30/0	10.57/0	14.16/0	1.26/4
	7	103.37/1	514.27/0	224.16/0	245.52/0	3.17/4
	8	4.54/3	33.98/0	19.15/0	12.57/0	3.10/2
	9	12.49/2	1352.76/0	36.02/0	57.57/1	4.04/2
	10	134.63/0	6666.04/0	1096.58/0	43.75/1	0.94/4
Avg/Sum		37.58/17	441/5	77.15/12	24.50/28	2.12/40

The gap concerning the best solution value found/number of the best solutions found

methods. LBBD-MIP-OC denotes the LBBD method with sub-problems solved via MIP and optimality cut (26). LBBD-TS-GC means the LBBD method with sub-problems solved via TS and greedy assignment cut (27). LBBD-MIP-GC and LBBD-TS-OC have similar meanings.

Apart from the above algorithms, we also tested the LBBD using both OC and GC combined. However, the preliminary tests show that the LBBD using both OC and GC combined performs worse than the methods with only one of the two Benders cut. This is because the greedy assignment cut is a suboptimal Benders cut as we pointed out earlier. If using both OC and GC in the LBBD, it is possible that more unexplored assignment solutions are ruled out in an iteration. In that case, some potentially better solutions may be missed, and hence the quality of the final delivered solution may be degraded.

For the computational results, the percentage gap with respect to the best-found solution value ($\text{gap} = 100 \times (F - F_{\text{best}})/F_{\text{best}}$), the number of best solutions and the average running time (in seconds) are listed as a triple in each cell separated by a forward slash in a cell in the table. If an optimal solution is available, the gap calculation utilizes the optimal solution value.

Small-sized instances

Table 3 lists the computational results for small instances. The MIP and LBBD-MIP-OC obtain the optimal solutions for all small instances. Since the LBBD consumes most of the running time to invoke a Benders cut, the computational time of LBBD-MIP-OC is significantly longer than that of MIP, and has an average value of 4.40 s. The LBBD-MIP-GC delivers only 86 out of 100 optimal solutions. These results confirm that GC is a suboptimal Benders cut, and it may overlook some further improvement opportunities for machines besides the machine with the maximum portion of the objective value. Furthermore, LBBD-TS-OC also yields 86 out of 100 optimal solutions. This observation verifies that the obtained pre-processing pattern is not optimal based on the decoding scheme used in Tabu search to solve a sub-problem. However, the mean running times of LBBD-TS-OC and LBBD-TS-GC are less than that consumed by MIP, and are 0.23 s and 0.08 s, respectively. The shorter running time exhibited by LBBD-TS-OC and LBBD-TS-GC indicates that LBBD with Tabu search for sub-problems is promising to solve larger instances in terms of running time.

To further explore the performance of the mentioned five methods in solving our problem, we listed the detailed results

for these instances in which at least one method cannot solve them to optimality in Table 4. There are 2 instances where LBBD-TS-OC and LBBD-TS-GC have a gap of more than 50%. We believe that this is partly due to their very small optimal solution values of less than 5. With such a small solution value, numerically, it is prone to make a big gap when only a slight improvement occurs. Moreover, it is also observed that it is possible to lose an optimal solution when using MIP and GC with an average gap of 3.12%.

Medium-sized instances

Table 5 lists the computational results of medium-sized instances. Since all methods consume the given time limitation (600s), the running time is ignored in the medium-sized and later large-sized instances. Meanwhile, the listed gap is related to the best solution value found among the five methods. It can be seen that the LBBD-TS-GC and LBBD-TS-OC perform better than MIP. The best average gap is 2.12% given by LBBD-TS-GC. Since the generated sub-problems are still NP-hard, obtaining a suitable solution within 10s via MIP is hard. This can be observed from the larger gaps by LBBD-MIP-OC and LBBD-MIP-GC compared with the other three methods. There is a huge gap for the instances of group 10 of 40×6 . We delved deeper into the result of every instance in that group and found that the objective value of each instance in that group is very small. For the number of best solutions found, the LBBD-TS-GC performs the best and delivers the best solutions for 40 out of the 100 instances. Based on the performance of the LBBD with sub-problems solved by Tabu search, we believe that the method can be used to solve medium-sized instances.

Large-sized instances

Table 6 gives the computational results of large instances. Due to the worse performance of the LBBD-MIP-OC and LBBD-MIP-GC, the two methods are abandoned in the test of large instances. It is observed that the LBBD-TS-OC and LBBD-TS-GC outperform MIP significantly in terms of the gap. Although greedy assignment cut cannot ensure the optimality of a solution, the corresponding LBBD-TS-GC still performs the best among the listed three methods, and its average gap value for the 100 instances listed in Table 6 is 0.81% as shown in the last row of Table 6. The MIP cannot deliver the best solution for large instances, and its mean gap value is 250.70%. In contrast, LBBD-TS-OC gives the best solutions for 45 out of 100 instances, and LBBD-TS-GC has 61 instances with the best solutions found. Finally, It can be summarized that LBBD-TS-GC has the best performance for large instances, which confirms that it is possible to obtain reasonable solutions when reducing the number of jobs on the machine with the maximum proportion of objective value.

Table 6 Computational results of large instances

Instance		MIP	LBBD-TS-OC	LBBD-TS-GC
Size	Group			
100×2	1	79.59/0	0.07/1	0.00/4
	2	94.29/0	0.23/2	0.18/4
	3	177.74/0	0.75/3	0.31/2
	4	434.46/0	0.35/4	2.29/1
	5	107.48/0	0.14/3	0.25/2
	6	189.22/0	0/5	0.16/2
	7	780.72/0	1.28/3	1.72/2
	8	270.58/0	0.41/1	0.26/4
	9	765.92/0	0.37/4	1.89/1
	10	573.69/0	0.18/3	0.32/2
100×8	1	23.68/0	0.61/1	0.29/4
	2	47.17/0	0.77/4	0.40/1
	3	59.82/0	2.47/1	0.27/4
	4	92.62/0	2.50/2	0.61/4
	5	70.73/0	1.37/2	0.25/3
	6	94.91/0	4.93/2	0/5
	7	179.10/0	8.62/1	0.96/4
	8	135.73/0	4.44/1	1.45/4
	9	604.75/0	36.58/1	4.14/4
	10	231.82/0	29.75/1	0.55/4
Avg/Sum		250.70/0	5.00/45	0.81/61

The gap concerning the best solution value found/number of the best solutions found

Fig. 4 Mean plot of the rates of tardiness cost and energy cost for instances 40×2 with different values of β

Economic analysis on energy consumption

In this subsection, the effect of the unit energy cost β is discussed for finding the trade-off between tardiness and energy consumption. To achieve the aim, we used the medium instances with $n = 40$ and $m = \{2, 6\}$ to obtain computational data. Here, only the instances with index 1 are used and the value of parameter β is picked from the set $\{0.001, 0.005, 0.01, 0.04, 0.08, 0.2\}$. Recall in our model, and the objective value consists of two parts: the total tardiness cost and the

Table 7 Tardiness and energy consumption for medium instances with index 1

Instance		F(T)						F(E)					
Size	Group	0.001	0.005	0.01	0.04	0.08	0.2	0.001	0.005	0.01	0.04	0.08	0.2
40×2	1	14892.5	15088.9	15222.2	15361	15335.4	15354.2	33.257	50.23	43.52	5.92	9.76	24.4
	2	20711.6	21182.1	21446.3	21312.8	21499.6	21594.4	41.833	53.12	3.09	22.44	10.72	32
	3	5273.67	5670.05	5758.39	5784.86	5820.88	5837.68	64.125	30.255	7.7	10.72	19.68	49.2
	4	2962.93	3306.39	3205.01	3228.71	3315.99	3161.33	34.6	0.79	18.63	6.72	14.08	37
	5	13847.2	14246.2	14562.9	14486.7	14486.4	14437.7	76.248	24.075	2.88	10.56	22.8	54.4
	6	5662.6	5718.17	5730.03	5807.06	5867.53	5805.63	15.4	21.26	25.43	3.88	7.12	17.8
	7	2527.11	2987.9	3062.28	3244.08	3035.99	2978.19	57.524	26.015	2.81	11.88	21.36	57.4
	8	3224.68	3259.51	3292.75	3368.29	3423.68	3415.45	14.485	9.995	15.23	3.68	9.6	21.2
	9	1948.14	2095.86	2095.86	2080.88	2063.62	2081.37	32.873	8.455	16.91	11.24	10.48	29.6
	10	956.47	956.12	975.12	941.26	958.95	964.34	14.736	0.905	2.09	6.92	13.84	36.2
40×6	1	4701.98	4686.7	4738.05	4939.58	4903.61	4883.23	21.682	37.535	55.98	5.84	14.56	24
	2	3306.93	3454.96	3335.3	3436.93	3437.58	3420.49	30.491	23.39	3.99	8.52	15.36	45.6
	3	602.72	735.6	724.11	698.8	776.14	686.76	11.78	29.15	27.39	15.04	10.88	26.4
	4	263.7	476.25	331.6	393.16	433.26	454.2	18.551	17.08	18.02	11.48	36.88	52.6
	5	1867.11	1827.69	1853.64	1877.3	1911.97	1941.99	16.423	27.675	27.73	5.6	13.84	24
	6	945.27	1084.79	1050.23	1097.26	1090.28	1183.93	34.262	18.98	32.41	8.44	16.08	38.6
	7	0	0	2.64	0	3.89	0	0.105	1.615	5.14	4.48	10.08	20.2
	8	911.41	860.06	951.65	989.7	906.41	966.45	22.436	15.375	0.53	2.76	5.04	15.8
	9	75.76	72.79	79.17	60.78	65.73	78.57	7.043	0.275	0.61	2.44	5.36	10.8
	10	0	0	0	0	0	0	0.069	0.255	0.51	2.24	4.72	12.4

extra energy consumption cost. Here, let the total tardiness cost denoted by $F(T) := \sum_{j \in N} \alpha_j T_j$ and the total extra energy consumption cost denoted by $F(E) := \beta \sum_{j \in N} E_j$.

In Table 7, we listed the two costs for analyzing the relationship of the two values with parameter β . It is observed that the value of $F(T)$ increases when the unit energy cost increases. After a specific value of β , the tardiness cost starts to reduce. Overall, the change of tardiness cost is not significant with the change of β . However, the energy cost firstly decreases and then increases after a specific value of β in the interval from 0.001 to 0.2. When the number of machines becomes 6, the trend of the mentioned two costs is slightly different because the tardinesses are 0 for some instances.

To further analyze their general trends, the two costs are normalized via the following approach. Two rates $\text{rate}_{F(T)}$ and $\text{rate}_{F(E)}$ are introduced by using the maximum value among the results using different β values. Then, all values listed in Table 7 can be transformed into the values within the range $[0, 1]$. Since the tardiness cost is 0 for some instances with 40×6 , the transformation is made for the instances with 2 machines. Moreover, the mean values of the values of $\text{rate}_{F(T)}$ and $\text{rate}_{F(E)}$ of 10 instances are shown in Fig. 4 for each value of β , respectively. Figure 4 clearly shows the change trends of the two costs with the increase of the unit energy cost. Based on the results of Table 7, the objective

value is also analyzed via the above procedure, and it is observed that the objective value increases with the increment of the value of β , and the corresponding values are $[0.930, 0.971, 0.979, 0.987, 0.988]$. Since the rates of the objective values are very close to the values of $\text{rate}_{F(T)}$, they were not plotted in the figure. However, it can be said that the total cost incurred will increase if the energy cost rises continuously.

Conclusions

In this paper, an identical parallel machine scheduling problem with step-deteriorating jobs motivated by the urea production process was considered. We aimed to find a schedule for minimizing the weighted sum of the total tardiness cost and the total extra energy consumption cost when facing two different pre-processing patterns. A mixed-integer programming model was formulated to obtain the exact solutions for small-sized instances. To solve large-sized problems, we designed a logic-based Benders decomposition framework. In our algorithm, we tried two solution approaches to deal with the generated sub-problems, including mixed-integer programming and Tabu search. Furthermore, we also developed two Benders cuts for the connection between the

master problem and sub-problems. The proposed LBBD with a greedy assignment cut and Tabu search performs better than other methods when solving larger instances based on the computational analysis. Moreover, the unit energy cost sensitivity is analyzed, and the change trends of the total tardiness cost and the energy cost are discussed.

Future research should develop a multi-objective optimization method to analyze the proposed problem and introduce some efficient meta-heuristics to solve the problem within a short running time. Furthermore, we will try to extend the studied situation to a more complex environment, including job-shop, flexible job-shop, etc.

Acknowledgements This work has been supported by the National Natural Science Foundation of China (No. 51405403), and the Fundamental Research Funds for the Central Universities (No. 2682018CX09).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Abedi M, Chiong R, Noman N, Zhang R (2020) A multi-population, multi-objective memetic algorithm for energy-efficient job-shop scheduling with deteriorating machines. *Expert Syst Appl* 157:113348
2. Agnetis A, Billaut JC, Gawiejnowicz S, Pacciarelli D, Soukhal A (2014) Multiagent scheduling: Models and Algorithms. Springer, Berlin Heidelberg
3. Anghinolfi D, Paolucci M, Ronco R (2021) A bi-objective heuristic approach for green identical parallel machine scheduling. *Euro J Oper Res* 289(2):416–434
4. Barzanji R, Naderi B, Begen MA (2020) Decomposition algorithms for the integrated process planning and scheduling problem. *Omega* 93:102025
5. Bigras LP, Gamache M, Savard G (2008) The time-dependent traveling salesman problem and single machine scheduling problems with sequence dependent setup times. *Discrete Opt* 5(4):685–699
6. Bilge Ü, Kurtulan M, Kırac F (2007) A tabu search algorithm for the single machine total weighted tardiness problem. *Euro J Oper Res* 176(3):1423–1435
7. Biskup D, Herrmann J, Gupta JN (2008) Scheduling identical parallel machines to minimize total tardiness. *Int J Prod Econ* 115(1):134–142
8. Bożejko W, Grabowski J, Wodecki M (2006) Block approach-tabu search algorithm for single machine total weighted tardiness problem. *Comput Ind Eng* 50(1–2):1–14
9. BP, (2020). Statistical review of world energy 2020
10. Che A, Zhang S, Wu X (2017) Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *J Clean Prod* 156:688–697
11. Cheng T, Ding Q (2001) Single machine scheduling with step-deteriorating processing times. *Euro J Oper Res* 134(3):623–630
12. Cheng W, Guo P, Zhang Z, Zeng M, Liang J (2012) Variable neighborhood search for parallel machines scheduling problem with step deteriorating jobs. *Mathematical Problems in Engineering* 928312
13. Ciré AA, Coban E, Hooker JN (2016) Logic-based benders decomposition for planning and scheduling: a computational analysis. *Knowl Eng Rev* 31(5):440–451
14. Delorme M, Iori M, Martello S (2017) Logic based benders' decomposition for orthogonal stock cutting problems. *Comput Oper Res* 78:290–298
15. Delorme M, Iori M, Mendes NF (2021) Solution methods for scheduling problems with sequence-dependent deterioration and maintenance events. *Euro J Oper Res* 295(3):823–837
16. Eren T, Güner E (2007) Minimizing total tardiness in a scheduling problem with a learning effect. *Appl Math Modell* 31(7):1351–1361
17. Gacias B, Artigues C, Lopez P (2010) Parallel machine scheduling with precedence constraints and setup times. *Comput Oper Res* 37(12):2141–2151
18. Gahm C, Denz F, Dirr M, Tuma A (2016) Energy-efficient scheduling in manufacturing companies: a review and research framework. *Euro J Oper Res* 248(3):744–757
19. Gao K, Huang Y, Sadollah A, Wang L (2020) A review of energy-efficient scheduling in intelligent production systems. *Complex Intell Syst* 6:237–249
20. Gawiejnowicz S (2020a) Models and Algorithms of Time-Dependent Scheduling. Springer, Berlin Heidelberg
21. Gawiejnowicz S (2020b) A review of four decades of time-dependent scheduling: main results, new topics, and open problems. *J Scheduling* 23:3–47
22. Graham RL, Lawler EL, Lenstra JK, Kan A (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann Discrete Math* 5:287–326
23. Guo P, Cheng W, Wang Y (2014) A general variable neighborhood search for single-machine total tardiness scheduling problem with step-deteriorating jobs. *J Ind Manag Opt* 10(4):1071–1090
24. Guo P, Cheng W, Wang Y (2015) Parallel machine scheduling with step-deteriorating jobs and setup times by a hybrid discrete cuckoo search algorithm. *Eng Opt* 47(11):1564–1585
25. Guo P, Cheng W, Wang Y (2017) Scheduling step-deteriorating jobs to minimise the total weighted tardiness on a single machine. *Int J Syst Sci* 4(2):92–107
26. Guo P, He X, Luan Y, Wang Y (2021) Logic-based benders decomposition for gantry crane scheduling with transferring position constraints in a rail-road container terminal. *Eng Opt* 53(1):86–106
27. Heching A, Hooker JN, Kimura R (2019) A logic-based benders approach to home healthcare delivery. *Trans Sci* 53(2):510–522
28. Hooker JN (2007) Planning and scheduling by logic-based benders decomposition. *Oper Res* 55(3):588–602
29. Huang RH, Yu SC (2016) Two-stage multiprocessor flow shop scheduling with deteriorating maintenance in cleaner production. *J Clean Prod* 135:276–283
30. Lalla-Ruiz E, Voß S (2016) Modeling the parallel machine scheduling problem with step deteriorating jobs. *Euro J Oper Res* 255(1):21–33
31. Leung J, Ng C, Cheng T (2008) Minimizing sum of completion times for batch scheduling of jobs with deteriorating processing times. *Euro J Oper Res* 187(3):1090–1099
32. Li Z, Yang H, Zhang S, Liu G (2016) Unrelated parallel machine scheduling problem with energy and tardiness cost. *Int J Adv Manuf Technol* 84(1):213–226

33. Mensendiek A, Gupta JN, Herrmann J (2015) Scheduling identical parallel machines with fixed delivery dates to minimize total tardiness. *Euro J Oper Res* 243(2):514–522
34. Naderi B, Roshanaei V (2020) Branch-relax-and-check: a tractable decomposition method for order acceptance and identical parallel machine scheduling. *Euro J Oper Res* 286(3):811–827
35. NBS (2020) China energy statistical yearbook 2019. China Statistic Press
36. Ozer EA, Sarac T (2019) Mip models and a matheuristic algorithm for an identical parallel machine scheduling problem under multiple copies of shared resources constraints. *Top* 27(1):94–124
37. Pan Z, Lei D, Zhang Q (2018) A new imperialist competitive algorithm for multiobjective low carbon parallel machines scheduling. *Mathematical problems in engineering* 5914360
38. Pei J, Song Q, Liao B, Liu X, Pardalos PM (2020a) Parallel-machine serial-batching scheduling with release times under the effects of position-dependent learning and time-dependent deterioration. *Ann Oper Res* 298:407–444
39. Pei J, Wang X, Fan W, Pardalos PM, Liu X (2019) Scheduling step-deteriorating jobs on bounded parallel-batching machines to maximise the total net revenue. *J Oper Res Soc* 70(10):1830–1847
40. Pei J, Wei J, Liao B, Liu X, Pardalos PM (2020b) Two-agent scheduling on bounded parallel-batching machines with an aging effect of job-position-dependent. *Ann Oper Res* 294:191–223
41. Pinedo ML (2016) Scheduling: theory, algorithms, and systems, 5th edn. Springer, New York
42. Rahmanian N, Homayoonfard M, Alamdari A (2013) Simulation of urea prilling process: an industrial case study. *Chem Eng Commun* 200(6):764–782
43. Saberi-Aliabad H, Reisi-Nafchi M, Moslehi G (2020) Energy-efficient scheduling in an unrelated parallel-machine environment under time-of-use electricity tariffs. *J Clean Prod* 249:119393
44. Schneider J (2002) The time-dependent traveling salesman problem. *Physica A* 314(1–4):151–155
45. Shioura A, Shakhlevich NV, Strusevich VA, Primas B (2018) Models and algorithms for energy-efficient scheduling with immediate start of jobs. *J Scheduling* 21(5):505–516
46. Sundararaghavan P, Kunnathur A (1994) Single machine scheduling with start time dependent processing times: some solvable cases. *Euro J Oper Res* 78(3):394–403
47. Tadumadze G, Emde S, Diefenbach H (2020) Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines. *OR Spectrum* 42:461–497
48. Tigane M, Dahane M, Boudhar M (2019) Multiobjective approach for deteriorating jobs scheduling for a sustainable manufacturing system. *Int J Adv Manuf Technol* 101:1939–1957
49. Tran TT, Araujo A, Beck JC (2016) Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS J Comput* 28(1):83–95
50. Wan G, Yen BPC (2002) Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. *Euro J Oper Res* 142(2):271–281
51. Wang S, Wang X, Yu J, Ma S, Liu M (2018) Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. *J Clean Prod* 193:424–440
52. Wheatley D, Gzara F, Jewkes E (2015) Logic-based benders decomposition for an inventory-location problem with service constraints. *Omega* 55:10–23
53. Wu X, Che A (2019) A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega* 82:155–165
54. Wu X, Shen X, Li C (2019) The flexible job-shop scheduling problem considering deterioration effect and energy consumption simultaneously. *Comput Ind Eng* 135:1004–1024
55. Xu S, Hall NG (2021) Fatigue, personnel scheduling and operations: Review and research opportunities. *European Journal of Operational Research* Available online
56. Xue Y, Rui Z, Yu X, Sang X, Liu W (2019) Estimation of distribution evolution memetic algorithm for the unrelated parallel-machine green scheduling problem. *Memetic Comput* 11(4):423–437
57. Zandi A, Ramezani R, Monplaisir L (2020) Green parallel machines scheduling problem: A bi-objective model and a heuristic algorithm to obtain pareto frontier. *J Oper Res Soc* 71(6):967–978
58. Zhang L, Deng Q, Gong G, Han W (2020) A new unrelated parallel machine scheduling problem with tool changes to minimise the total energy consumption. *Int J Prod Res* 58(22):6826–6845

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.