European Journal of Operational Research 291 (2021) 128–146

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A combinatorial Benders decomposition algorithm for parallel machine scheduling with working-time restrictions

Kan Fang[a], Shijin Wang[b,*], Michael L. Pinedo[c], Lin Chen[d], Feng Chu[e,f]

[a] College of Management and Economics, Tianjin University, Tianjin 300072, China
[b] School of Economics and Management, Tongji University, Shanghai, NY 200092, China
[c] Stern School of Business, New York University, New York 10012, USA
[d] Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA
[e] IBISC, Univvry, University of Paris-Saclay, Paris, France
[f] School of Economics and Management, Fuzhou University, Fuzhou 350000, China

ARTICLE INFO

Article history:
Received 25 November 2019
Accepted 24 September 2020
Available online 3 October 2020

Keywords:
Scheduling
Parallel machine
Maximum consecutive working-time
Minimum break time
Combinatorial Benders decomposition

ABSTRACT

This paper addresses a parallel machine scheduling problem with restrictions on employees' working-times and break times. Tasks must be processed by employees nonpreemptively on unrelated parallel machines with different thresholds that specify for each employee the maximum total and consecutive working-time, and the minimum break time. The objective is to minimize the weighted sum of the makespan, the machine depreciation costs, and the labor costs. To solve this problem, a mixed integer linear programming model is formulated, and two different decomposition-based exact algorithms are implemented as well as a list scheduling (LS)-based heuristic method. Extensive computational experiments are performed on randomly generated instances, and the results demonstrate the efficiency of our proposed combinatorial Benders decomposition approach.

1. Introduction

With an increasing demand for better working conditions and a growing concern with respect to workers' health and safety, regulatory authorities in various countries have started to implement stricter labor and employment laws regarding employees' working hours. As shown in the literature, long working hours, especially "very extended" or "extremely extended" hours, could entail significant physiological and psychological risks to individual employees, which may lead to fatigue and stress (e.g. Waersted & Westgaard, 1991). This has become a major public policy issue. To protect employees against such "adverse insecurities", many regulatory systems have incorporated various upper limits on working-time durations, such as maximum daily hours, maximum weekly hours, and maximum permitted overtime hours (Lee, McCann, & Messenger, 2007). As a result, the additional restrictions on working-time durations make many employee scheduling problems more complicated and difficult to solve, and present interesting modeling and algorithmic challenges.

In the past decade, more and more researchers have started to incorporate different working time regulations into various employee scheduling problems that arise in a variety of service industries, including the scheduling of nurses in hospitals, crew members in transportation environments, and so on. To determine a feasible timetable for each employee within a planning horizon, the scheduler not only has to provide a suitable schedule for the employees in order to satisfy service requirements, but he/she also has to comply with working regulations and other cost constraints. We give a few examples. Rodrigues, de Souza, and Moura (2006) proposed a computational tool to solve an urban transportation problem to meet passenger demand and minimize operational costs, while satisfying a number of labor and safety regulations concerning maximum working hours and rest periods. Saddoune, Desaulniers, Elhallaoui, and Soumis (2011) investigated an integrated crew scheduling problem to determine least cost schedules that cover all flights and meet various safety and collective agreement rules regarding working time durations of crew members. Braekers, Hartl, Parragh, and Tricoire (2016) considered a home care routing and scheduling problem, in which each nurse can work overtime at a certain additional cost within a given maximum working time, and the objective is to minimize total costs as well as client inconvenience. Ağralı, Taşkın, and Ünal (2017) studied an employee scheduling problem with flexible em-

* Corresponding author.
E-mail addresses: kfang@tju.edu.cn (K. Fang), shijinwang@tongji.edu.cn (S. Wang), mpinedo@stern.nyu.edu (M.L. Pinedo), Lin.Chen@ttu.edu (L. Chen), feng.chu@univ-evry.fr (F. Chu).

https://doi.org/10.1016/j.ejor.2020.09.037
0377-2217/

ployee availability as well as flexible demand, under several legislative constraints such as maximum total working time within a week. For further pointers to the general employee (personnel) scheduling problem, we refer the reader to the surveys by Ernst, Jiang, Krishnamoorthy, and Sier (2004), Brucker, Qu, and Burke (2011) and Van den Bergh, Beliën, De Bruecker, Demeulemeester, and De Boeck (2013).

It is noteworthy that the most recent research on employee scheduling only imposes upper limits on the total working hours within a day, a week or some other specified planning horizon. However, *consecutive working-time*, that is, the total working time of an employee without a break, also plays an important role in employees' fatigue and stress as well as in potential risks with regard to health and safety. For example, the new regulations in the European Union regarding drivers' working hours (i.e. EC regulation 561/2006) require that for each driver a break or a rest period must be scheduled after an accumulated driving time of four-and-a-half hours (Goel, 2009). Krempels and Panchenko (2006) considered an operating theatre scheduling problem in which each surgical team is restricted within a shift by the maximum working time allowed without a break.

On the other hand, as we know, extended working hours can also be found in a wide variety of manufacturing settings, in which different products have to share machines, resources and workers, due to the commonality or similarities of components. Therefore, it may be of interest to integrate other decision making such as resource (e.g., machine) scheduling into the employee scheduling problem, or vice versa. For example, Lodree Jr., Geiger, and Jiang (2009) established a framework to incorporate human characteristics and behaviors into the machine scheduling paradigm, and discussed potential interdisciplinary research opportunities in scheduling and human factors; Guyon, Lemaire, Pinson, and Rivreau (2010) recommended that an appropriate employee timetable has to be built up together with the production schedule; Edis, Oguz, and Ozkarahan (2013) suggested that the machine scheduling problem with additional resources, such as machine operators, still remains an important area of research.

Given the intrinsic close interrelationships between employee scheduling and machine scheduling, an increasing amount of work has focused on how to assign the machines and their corresponding operators to different tasks in order to achieve productivity and/or service goals. For example, Huq, Cutright, and Martin (2004) developed a mixed integer programming model to determine the lot size that minimizes the makespan for a fixed daily workload in a multi-processor flow shop. Artigues, Gendreau, Rousseau, and Vergnaud (2009) studied an integrated employee timetabling and job shop scheduling problem, and proposed various exact hybrid methods that are based on integer linear programming and constraint programming. Guyon et al. (2010) developed two exact methods based on Benders decomposition and cut generation to solve an integrated employee timetabling and production scheduling problem, in which tasks can be interrupted and processed by different operators, and with as objective to schedule the jobs and to assign a work pattern to each operator that satisfies the need for operators at minimum cost. In a follow-up work, Guyon, Lemaire, Pinson, and Rivreau (2014) analyzed a minimum-cost integrated employee timetabling and job-shop scheduling problem, and proposed new exact methods based on cut generation approaches. Agnetis, Murgia, and Sbrilli (2014) considered a job shop scheduling problem with human operators in handicraft production to minimize the makespan, and proposed two different heuristics for decomposing the problem. Ahmadi-Javid and Hooshangi-Tabrizi (2017) studied a ternary-integration job-shop scheduling problem with employee timetabling and heterogeneous transporters to minimize the makespan, and developed an Anarchic Society Optimization algorithm to solve the problem. Dolgui, Kovalev, Kovalyov, Malyutin, and Soukhal (2018) investigated a workforce assignment problem in a paced assembly line to minimize the maximum number of workers employed at any time, subject to constraints regarding the cycle time of the line and the number of workers assigned to each operation.

Although a great deal of research has been done on the integration of employee and machine scheduling, our literature review suggests that research on incorporating restrictions with regard to working-time durations into such problems still appears to be rather sparse. The only exception we found is the work by Fischetti, Martello, and Toth (1989), who considered a fixed job schedule problem with working-time constraints, in which each of the tasks requires processing without interruption within its time window, and the objective is to perform all tasks with a minimum number of processors, so that no processor (e.g., crew member in a bus company) has to work longer than the given working time limit.

Researchers have also studied various employee scheduling problems that assume a heterogeneous workforce. In these problems, employees may either possess different skill sets or have different skill levels and thus may perform tasks at different speeds, and may therefore incur different labor costs. For example, Valls, Pérez, and Quintanilla (2009) proposed a skilled workforce scheduling problem at a service centre, in which a task may have a duration that depends on the worker to whom it has been assigned, and the goal is to minimize constraint violations according to the preferences set by the decision-maker. Othman, Bhuiyan, and Gouw (2012) studied a workforce planning problem in a job shop environment and integrated workers' differences into the problem so as to minimize the total costs. Benavides, Ritt, and Miralles (2014) considered a flow shop scheduling problem with heterogeneous workers to minimize the makespan, in which workers may be unable to operate a subset of the machines, or may have different execution times for the same operation. For more pointers to the literature on scheduling problems with workers that have different skill sets, we refer the reader to the survey by De Bruecker, Van den Bergh, Beliën, and Demeulemeester (2015).

In this work, we consider an integrated employee and parallel machine scheduling problem, in which a set of n jobs has to be processed nonpreemptively on m machines by ℓ employees. Note that in several types of scheduling environments, such as the operating room scheduling problem, a surgeon (employee) in an operating room (machine) cannot be interrupted before the completion of the current surgery (job) (Freeman, H., & J., 2016). Therefore, it is reasonable to assume that employees are not allowed to terminate their shifts during the execution of a job.

Similar to previous studies in the literature, we also follow the general regulations regarding employees' working hours by imposing an upper limit on each employee's total working-time in order to avoid highly unbalanced schedules or potential inequities in workloads. In addition, because of the different skill levels of the employees, the processing time of each job on a machine may depend on the employee, and there may also be restrictions with regard to maximum consecutive working-time and minimum break time for each employee. The objective of our problem is to minimize the weighted sum of the makespan-related costs plus the machine depreciation and labor costs. For simplicity, we refer to this problem as the *integrated employee and parallel machine scheduling problem with consecutive working-time constraints*, or the IEMSCW problem for short.

To solve the IEMSCW problem, we first propose in Section 2 a mixed integer linear programming (MILP) formulation with time-indexed variables. In Section 3, we develop a time-decomposition method using a "divide-and-conquer" approach in our search for optimal solutions. In order to find optimal solutions even for large-scale instances, we design in Section 4 a combinatorial Benders

decomposition method, followed in Section 5 by a list scheduling (LS)-based heuristic method for generating feasible schedules. Then, in Section 6 we conduct extensive computational experiments. The results demonstrate the efficiency of the combinatorial Benders decomposition approach. Finally, we conclude our research and discuss possible future research directions in Section 7.

2. Mathematical description of the problem

As mentioned in the introduction, we refer to the problem that we study as the *integrated employee and parallel machine scheduling problem with consecutive working-time constraints*, or the IEMSCW problem for short. In this problem, there is a set $\mathcal{J} = \{1, 2, \ldots, n\}$ of jobs, a set $\mathcal{M} = \{1, 2, \ldots, m\}$ of machines, and a set $\mathcal{W} = \{1, 2, \ldots, \ell\}$ of employees. We assume that each job $j \in \mathcal{J}$ must be processed nonpreemptively, and can only be processed by exactly one employee $w \in \mathcal{W}$ on some machine $i \in \mathcal{M}$, with an associated processing time of p_{ijw} depending on both the employee's skill level and the machine setting. Employees cannot terminate their shifts during the execution of a job. In addition, each machine $i \in \mathcal{M}$ incurs an associated depreciation cost f_i per unit time when it is processing a job and no cost when it is idle, and each employee $w \in \mathcal{W}$ has an associated labor cost c_w per unit time when he/she is working and no cost when he/she is idle. There also is a cost θ per unit time related to the makespan $C_{\max}$.

Due to the labor and employment laws imposed by the regulatory authorities, we assume that an employee $w \in \mathcal{W}$ cannot process jobs consecutively without a break for more than d_w time units. To guarantee sufficient rest for the workers, when employee w starts to take a break after he/she has finished a job, we require at least a minimum break time of b_w time units. Moreover, there is also an upper limit on the total working time of each employee w, i.e., ϕ_w time units, so as to balance the workload of employees and ensure work equity between them. The objective is to find a feasible schedule that minimizes the weighted sum of all costs, including the depreciation costs of machines, the labor costs of employees and the makespan-related costs. Without loss of generality, we assume that under the above working-time restrictions, at least one feasible assignment plan of jobs to machines and employees exists for the IEMSCW problem.

2.1. A time-indexed formulation of the IEMSCW problem

To begin with, we let $T = \sum_{j=1}^{n} \max_{(i,w)} \{p_{ijw}\}$ be the length of the planning horizon, and $\mathcal{T} = \{0, 1, 2, \ldots, T-1\}$ be the corresponding set of time instances, that is, the planning horizon is divided into T periods, and each period t begins at time instant t and ends at time instant $t+1$, for $t \in \mathcal{T}$. In what follows, we propose a mixed integer linear program for the IEMSCW problem. We define the following decision variables:

- x_{ijwt} is equal to 1 if job j starts processing on machine i at time t by employee w, and 0 otherwise;
- β_{wt} is equal to 1 if employee w starts break at time instant t after finishing some job, and 0 otherwise. Obviously, we have $\beta_{w0} = 0$.
- z_{wt} is equal to 1 if employee w is working at time t, and 0 otherwise;
- $C_{\max}$ is the makespan of the schedule.

The IEMSCW problem can now be formulated as follows.

$$\text{minimize} \quad \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}}\sum_{t=0}^{T-p_{ijw}} f_i p_{ijw} x_{ijwt} + \sum_{w\in\mathcal{W}}\sum_{t=0}^{T-1} c_w z_{wt} + \theta C_{\max} \tag{1a}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{M}}\sum_{w\in\mathcal{W}}\sum_{t=0}^{T-p_{ijw}} x_{ijwt} = 1 \quad \forall j \in \mathcal{J}; \tag{1b}$$

$$\sum_{w\in\mathcal{W}}\sum_{j\in\mathcal{J}}\sum_{s=\max\{0,t-p_{ijw}+1\}}^{t} x_{ijws} \le 1 \quad \forall i \in \mathcal{M}, t \in \{0, \ldots, T-1\}; \tag{1c}$$

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\sum_{s=\max\{0,t-p_{ijw}+1\}}^{t} x_{ijws} \le 1 \quad \forall w \in \mathcal{W}, t \in \{0, \ldots, T-1\}; \tag{1d}$$

$$z_{wt} = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\sum_{s=\max\{0,t-p_{ijw}+1\}}^{t} x_{ijws} \quad \forall w \in \mathcal{W}, t \in \{0, \ldots, T-1\}; \tag{1e}$$

$$\sum_{s=t}^{t+d_w} z_{ws} \le d_w \quad \forall w \in \mathcal{W}, t \in \{0, \ldots, T-d_w-1\}; \tag{1f}$$

$$\sum_{t\in\mathcal{T}} z_{wt} \le \phi_w \quad \forall w \in \mathcal{W}; \tag{1g}$$

$$z_{wt} \ge \beta_{w,t+1} \quad \forall w \in \mathcal{W}, t \in \{0, \ldots, T-2\}; \tag{1h}$$

$$z_{wt} + \beta_{wt} \le 1 \quad \forall w \in \mathcal{W}, t \in \{0, \ldots, T-1\}; \tag{1i}$$

$$z_{wt} - z_{w,t+1} \le \beta_{w,t+1}, \quad \forall w \in \mathcal{W}, t \in \{0, \ldots, T-2\}; \tag{1j}$$

$$\sum_{s=t}^{t+b_w-1} (1 - z_{ws}) \ge b_w \beta_{wt} \quad \forall w \in \mathcal{W}, t \in \{1, \ldots, T-b_w\}; \tag{1k}$$

$$C_{\max} \ge \sum_{i\in\mathcal{M}}\sum_{w\in\mathcal{W}}\sum_{t=0}^{T-p_{ijw}} (t + p_{ijw}) x_{ijwt} \quad \forall j \in \mathcal{J}; \tag{1l}$$

$$x_{ijwt}, \beta_{wt}, z_{wt} \in \{0, 1\} \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, w \in \mathcal{W}, t \in \mathcal{T}; \tag{1m}$$

The objective (1a) minimizes the total costs including machine depreciation costs, labor costs, and makespan-related costs. Constraints (1b) ensure that each job starts its processing on exactly one machine with exactly one employee and at exactly one time instant. Constraints (1c) ensure that at most one job can be processed at any time instant on any machine. Constraints (1d) ensure that at most one job can be processed at any time instant by any employee. Constraints (1e) give the relationship between z_{wt} and x_{ijwt}. Constraints (1f) ensure that the consecutive working-time of each employee is not longer than the given threshold d_w. Constraints (1g) impose an upper limit on the total working time for each employee within the given planning horizon. Constraints (1h)–(1j) define the relationship between z_{wt} and β_{wt}. Constraints (1k) force the minimum duration of a break time of employee w to be at least b_w after processing some job. Constraints (1l) define the makespan of the schedule. Constraints (1m) present the ranges of the decision variables. For simplicity, we refer to the above model (1) as the MILP model, and denote $D = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}}\sum_{t=0}^{T-p_{ijw}} f_i p_{ijw} x_{ijwt}$, $L = \sum_{w\in\mathcal{W}}\sum_{t=0}^{T-1} c_w z_{wt}$, and $M = \theta C_{\max}$ as the costs associated with machine depreciation, labor, and makespan, respectively.

Remark 1. It should be noted that there is a slight difference between the break time and the idle time of an employee. A break time is invoked for an employee only if this employee just has finished some job, and does not start with the processing of another job immediately; an idle time can be any non-working time, including a break time. To be specific, within the time interval in which an employee does not start processing his/her first job, this employee is idle but not on a break. Such a difference matches our intuition, since it is unreasonable to force employee w to take a break (i.e., of at least b_w time units) before he/she starts processing his/her first job.

Remark 2. In addition to the discrete time model described above, the IEMSCW problem can also be formulated as a continuous time model by directly assigning jobs to positions, which is a generalization of the original single machine formulation proposed by Lasserre and Queyranne (1992). Such a formulation is usually referred to as an "assignment and positional formulation". However, according to our experimental study, such a formulation has a much worse performance than the time-indexed formulation described above. For more detail, please see Appendix A.

3. A time-decomposition based approach

It is easy to see that when we set $\ell = +\infty$, $p_{ijw} = p_j$, $f_i = c_w = 0$, and $\theta = 1$, the IEMSCW problem becomes the $Pm||C_{\max}$ problem, which is already NP-hard in the ordinary sense with $m \geq 2$ (Pinedo, 2016). Therefore, it is to be expected that the above MILP model cannot solve the IEMSCW problem efficiently within a reasonable computation time for medium- and large-size instances. In this section, we develop a time-decomposition based method, which is based on a "divide-and-conquer" approach in its search for optimal solutions to the IEMSCW problem.

As we can see, in the IEMSCW problem, we have to make the following two types of decisions, namely (i) assignment decisions that specify the associated machine and employee for the processing of each job, and (ii) scheduling decisions that determine the sequence and the start time of each job as well as the makespan of the entire schedule. Therefore, one idea for solving the IEMSCW problem is based on a decomposition into two subproblems: one subproblem focuses on the minimization of the costs related to machine and employee assignment, and the other focuses on the minimization of the makespan $C_{\max}$ of the schedule.

Inspired by the above observation, within a given planning horizon $\widetilde{T}$, we first search for a feasible solution with minimum machine depreciation costs and labor costs by solving the following model of (2):

$$\text{minimize} \quad DL = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}}\sum_{t=0}^{\widetilde{T}-p_{ijw}} f_i p_{ijw} x_{ijwt} + \sum_{w\in\mathcal{W}}\sum_{t=0}^{\widetilde{T}-1} c_w z_{wt} \quad (2a)$$

$$\text{subject to} \quad \text{Constraints } (1b)-(1k), \text{ and } (1m). \quad (2b)$$

Let $\underline{\text{DL}}$ be the corresponding optimal objective of model (2) when $\widetilde{T} = T$. Obviously, $\underline{\text{DL}}$ is a lower bound on the sum of machine depreciation costs and labor costs. Then, we calculate the corresponding makespan determined by the solutions generated with model (2), and continue this process by iteratively decreasing the length of the planning horizon $\widetilde{T}$, as shown in Algorithm 3.1, in which C_j (see Step 4) is the completion time of job j within the planning horizon $\widetilde{T}$. For simplicity, we refer to Algorithm 3.1 as Algorithm TD.

Algorithm 3.1 A time-decomposition based approach for the IEMSCW problem.

1: Initialize: $\widetilde{T} \leftarrow T$, $\mathcal{O} \leftarrow \emptyset$.
2: **while** model-(2) is feasible within the planning horizon $\widetilde{T}$ **do**
3: Solve model-(2) and determine its optimal solutions $x^*_{ijwt}(\widetilde{T})$and $z^*_{wt}(\widetilde{T})$. Denote associated machine depreciation costs asD$(\widetilde{T})$, and labor costs as L$(\widetilde{T})$.
4: Using $x^*_{ijwt}(\widetilde{T})$ to calculate $C_j(\widetilde{T})$, and set $C_{\max}(\widetilde{T}) \leftarrow \max_j\{C_j(\widetilde{T})\}$ as the makespan.
5: Calculate the makespan-related costs $\text{M}(\widetilde{T}) \leftarrow \theta C_{\max}(\widetilde{T})$.
6: $\mathcal{O} \leftarrow \mathcal{O} \cup \{\text{D}(\widetilde{T}) + \text{L}(\widetilde{T}) + \text{M}(\widetilde{T})\}$.
7: Update $\widetilde{T} \leftarrow C_{\max}(\widetilde{T}) - 1$.
8: Calculate $\widetilde{T}^* \leftarrow \arg\min\{\mathcal{O}\}$.
9: Output the corresponding assignment and scheduling decisions $x^*_{ijwt}(\widetilde{T}^*)$ and $z^*_{wt}(\widetilde{T}^*)$.

4. A combinatorial Benders decomposition based approach

Given the structure of the IEMSCW problem, it seems that decomposing this problem into two subproblems, i.e., the assignment and the scheduling subproblems, may be a possible approach to obtain computational speedups. Benders decomposition is such a kind of partitioning method applicable to MIPs (Benders, 1962). As we know, Benders decomposition algorithms have demonstrated its efficiency in solving a wide range of difficult problems, including planning and scheduling problems (Canto, 2008; Hooker, 2007). In particular, as a variant of the classical Benders decomposition method, the combinatorial Benders decomposition algorithm, which was originally studied in the seminal work of Hooker (2000), has extended itself to solve various mixed-integer programming models with special structures (Chen, Lee, & Cao, 2012; Codato & Fischetti, 2006). To enhance the performance of the Benders decomposition algorithm and speed up the search process, instead of using information with respect to the dual to generate cuts, the combinatorial Benders decomposition method iteratively excludes the current solution of the master problem from further consideration via combinatorial or feasibility cuts, and continues such procedure until an optimal solution is identified (Rahmaniani, Crainic, Gendreau, & Rei, 2017).

Until recently, there have been a number of successful applications of combinatorial Benders decomposition algorithms to various optimization problems including the quayside operation problem at container terminals Chen et al. (2012), the problem of decomposing intensity modulated radiation therapy fluency maps using rectangular apertures Taşkın and Cevik (2013), the lock scheduling problem Verstichel, Kinable, De Causmaecker, and Vanden Berghe (2015), and assembly line balancing problems with setups Akpinar, Elmi, and Bektaş (2017), among others. These papers illustrate the potential of the combinatorial Benders decomposition method.

In this paper, we propose a combinatorial Benders decomposition based approach to solve the IEMSCW problem. For simplicity, we refer to the algorithm that implements this approach as Algorithm CBD.

4.1. The master and slave problems of Algorithm CBD

We first present the master problem (i.e. model (3)) that determines the assignment of jobs. We define y_{ijw}=1 if job j is assigned to machine i and processed by employee w, and 0 otherwise. For simplicity, we also define

$$F = \max\left\{\max_{i\in\mathcal{M}}\left\{\sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}} y_{ijw}p_{ijw}\right\}, \max_{w\in\mathcal{W}}\left\{\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}} y_{ijw}p_{ijw}\right\}\right\},$$

that is, F is the maximal value of the total job processing time on each machine and by each employee. The objective is to minimize the weighted sum of the corresponding machine depreciation

tion costs, labor costs and costs related to the value of F. Then, the master problem can be formulated as follows:

$$[\textbf{master}] \quad \text{minimize} \quad f_M = \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}}(f_i + c_w)p_{ijw}y_{ijw} + \theta F; \tag{3a}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{M}}\sum_{w\in\mathcal{W}} y_{ijw} = 1 \quad \forall j \in \mathcal{J}; \tag{3b}$$

$$F \geq \sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}} y_{ijw}p_{ijw} \quad \forall i \in \mathcal{M}; \tag{3c}$$

$$F \geq \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}} y_{ijw}p_{ijw} \quad \forall w \in \mathcal{W}; \tag{3d}$$

$$F \leq T; \tag{3e}$$

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}} y_{ijw}p_{ijw} \leq \phi_w \quad \forall w \in \mathcal{W}; \tag{3f}$$

$$y_{ijw} \in \{0,1\} \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, w \in \mathcal{W}. \tag{3g}$$

Constraints (3b) ensure that each job can only be assigned to exactly one machine and one employee. Constraints (3c) and (3d) define the value of F. Constraints (3e) ensure that the total processing times of jobs assigned to each machine cannot be more than the length of the given planning horizon. Constraints (3f) ensure that the maximum total working-time of each employee is not greater than the given threshold. It is easy to see that once variables y_{ijw} have been fixed, the value of F is also determined.

Suppose a solution $\mathcal{S} = \{y^*_{ijw}, F^*\}$ of the master problem as well as its objective value f^*_M have been obtained. We now try to find a feasible solution, if any, to the following slave problem (i.e. model (4)). Model (4) is mainly used to test feasibility of the solution $\mathcal{S} = \{y^*_{ijw}, F^*\}$, it does not matter what the objective function is, hence we set the objective function as minimize 0.

$$[\textbf{slave}] \quad \text{minimize} \quad 0 \tag{4a}$$

$$\text{subject to} \quad \sum_{t=0}^{T-p_{ijw}} x_{ijwt} = y^*_{ijw} \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, w \in \mathcal{W}; \tag{4b}$$

$$(t+1)z_{wt} \leq C_{\max} \quad \forall w \in \mathcal{W}, t \in \mathcal{T}; \tag{4c}$$

$$C_{\max} \leq F^*; \tag{4d}$$

$$\text{Constraints } (1c) - (1k); \tag{4e}$$

$$x_{ijwt}, z_{wt}, \beta_{wt} \in \{0,1\} \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, w \in \mathcal{W}, t \in \mathcal{T}. \tag{4f}$$

Constraints (4b) establish the relationship between variable x_{ijwt} and the value of y^*_{ijw} obtained from the master problem. Constraints (4c) and (4d) restrict the range of $C_{\max}$ in the slave problem. Constraints (4e) ensure that if a solution to the slave problem has been found, it must satisfy all the predefined restrictions regarding employees' working hours.

Obviously, if the slave problem returns a feasible solution, then the solution of $\{y^*_{ijw}, F^*\}$ is also optimal to the master problem, and thus we solve the original IEMSCW problem. Otherwise, the current assignment plan of jobs, i.e., y^*_{ijw}, should be forbidden over the current time horizon with length F^*.

Unfortunately, the problem of finding a feasible solution to the slave problem is not polynomial-time solvable. To illustrate, we consider the following problem (i.e. the slave(w) problem), in which we only search for a feasible schedule of jobs for each individual employee $w \in \{1, 2, \ldots, \ell\}$. Let $\mathcal{J}_w = \{j : y^*_{ijw} = 1\}$, and $\mathcal{U}_w = \{(i, j, w) | y^*_{ijw} = 1\}$. For each $j \in \mathcal{J}_w$, we define $p'_j = p_{ijw}$ if $y^*_{ijw} = 1$. Now the slave(w) problem becomes a search for a feasible schedule of jobs in $\mathcal{J}_w$ with processing times $\{p'_j\}$, so that working hours restrictions are satisfied, and the makespan is at most F^*.

Theorem 1. *The slave(w) problem is NP-hard in the strong sense.*

Proof. We reduce any instance of the 3-PARTITION problem, which is strongly NP-hard (Garey & Johnson, 1979), to an instance of the slave(w) problem. The 3-PARTITION problem is described as follows: Given a set $\mathcal{S} = \{1, 2, \ldots, 3m\}$ and positive integers $A_1, \ldots, A_{3m}, B$ such that $B/4 < A_j < B/2$ for all $j \in \mathcal{S}$ and $\sum_{j\in\mathcal{S}} A_j = mB$, does there exist a partition of $\mathcal{S}$ with m 3-element subsets $\mathcal{S}_1, \ldots, \mathcal{S}_m$ such that $\sum_{j\in\mathcal{S}_i} A_j = B$ for all $i = 1, \ldots, m$? Given any instance $\mathcal{I}_1$ of 3-PARTITION, we construct an instance $\mathcal{I}_2$ of the slave(w) problem as follows: The set of jobs is $\mathcal{J}_w = \{1, 2, \ldots, 3m\}$, the processing time of each job $j \in \mathcal{J}_w$ is:

$$p'_j = A_j \quad \text{for } j = 1, \ldots, 3m,$$

and the value of F^* is $mB + (m-1)$.

In addition, the maximum consecutive working-time and minimum break time for employee w are B and 1, respectively. Then there exists a feasible schedule of jobs for employee w if and only if this employee always achieves his/her maximum consecutive working-time B when he/she is working, and then only takes the minimum break time, i.e., one unit time. This can be done if and only if the 3-PARTITION problem has a solution, therefore the theorem follows. □

Although the general slave(w) problem is strongly NP-hard, during our implementation of Algorithm CBD, we can still solve the following optimization model to check if the given assignment plan is feasible for each specific employee $w_k \in \mathcal{W}$ within a fairly small amount of computational time:

$$[\textbf{slave}(\mathbf{w_k})] \quad \text{minimize} \quad 0 \tag{5a}$$

$$\text{subject to} \quad \sum_{t=0}^{T-p_{ijw_k}} x_{ijw_kt} = y^*_{ijw_k} \quad \forall i \in \mathcal{M}, j \in \mathcal{J}; \tag{5b}$$

$$(t+1)z_{w_kt} \leq C_{\max} \quad \forall t \in \mathcal{T}; \tag{5c}$$

$$C_{\max} \leq F^*; \tag{5d}$$

$$\text{Constraints } (1c) - (1k) \quad \text{where } w = w_k; \tag{5e}$$

$$x_{ijw_kt}, z_{w_kt}, \beta_{w_kt} \in \{0,1\} \quad \forall i \in \mathcal{M}, j \in \mathcal{J}, t \in \mathcal{T}. \tag{5f}$$

4.2. Combinatorial Benders cuts: a heuristic generation method

For simplicity, we denote the optimal solution of the master problem in the tth iteration as $\{y^{(t)}_{ijw}, F^{(t)}\}$, and the corresponding optimal objective as $f^{(t)}_M$. When an infeasible slave problem is encountered, we know that the current assignment plan of

jobs $y_{ijw}^{(t)}$ should be forbidden over the time horizon with length $F^{(t)}$. Let $\mathcal{U}^{(t)} = \left\{(i,j,w)|y_{ijw}^{(t)} = 1,\ \text{slave is infeasible}\right\}$ and $\overline{\mathcal{U}}^{(t)} = \left\{(i,j,w)|y_{ijw}^{(t)} = 0,\ \text{slave is infeasible}\right\}$, then we can add the following combinatorial Benders cut into the master problem:

$$\sum_{(i,j,w)\in\overline{\mathcal{U}}^{(t)}} y_{ijw} + \sum_{(i,j,w)\in\mathcal{U}^{(t)}} (1 - y_{ijw}) \geq 1, \tag{6}$$

which can be used to prevent the above assignment plan to the job schedule, that is, to make a schedule feasible for all employees, we need to remove at least one element in $\mathcal{U}^{(t)}$ from the current assignment plan, and replace it with some other element in $\overline{\mathcal{U}}^{(t)}$.

In particular, when not only the slave problem is infeasible, but also the slave(w_k) problem is infeasible for some $w_k \in \mathcal{W}$, then we let $\mathcal{U}_{w_k}^{(t)} = \left\{(i,j,w_k)|y_{ijw_k}^{(t)} = 1,\ \text{slave}(w_k)\ \text{is infeasible}\right\}$, and add the following additional combinatorial Benders cuts into the master problem:

$$\sum_{(i,j,w_k)\in\mathcal{U}_{w_k}^{(t)}} (1 - y_{ijw_k}) \geq 1, \tag{7}$$

which prevents the assignment plan of $\mathcal{U}_{w_k}^{(t)}$ to employee w_k. That is, to make a schedule feasible for employee w_k, we need to remove at least one element from $\mathcal{U}_{w_k}^{(t)}$.

In addition, note that each employee has a consecutive working-time restriction, if for some employee w_k, we have $\sum_{(i,j,w_k)\in\mathcal{U}_{w_k}^{(t)}} y_{ijw_k}^{(t)} p_{ijw_k} > d_{w_k}$, then this employee has to take breaks for at least $\left\lceil (\sum_{(i,j,w_k)\in\mathcal{U}_{w_k}^{(t)}} y_{ijw_k}^{(t)} p_{ijw_k})/d_{w_k} - 1 \right\rceil$ times. For simplicity, we define

$$\underline{F}_{w_k}^{(t)} = \sum_{(i,j,w_k)\in\mathcal{U}_{w_k}^{(t)}} y_{ijw_k}^{(t)} p_{ijw_k} + \left(\left\lceil \left(\sum_{(i,j,w_k)\in\mathcal{U}_{w_k}^{(t)}} y_{ijw_k}^{(t)} p_{ijw_k} \right)/d_{w_k} - 1 \right\rceil\right) b_{w_k}.$$

Then, given the current assignment plan of jobs $y_{ijw_k}^{(t)}$, the minimal completion time by employee w_k is at least $\underline{F}_{w_k}^{(t)}$. However, it may be the case that $F^{(t)}$ satisfies Constraints (3d) for employee w_k. That is, $F^{(t)} \geq \sum_{(i,j,w_k)\in\mathcal{U}_{w_k}^{(t)}} y_{ijw}^{(t)} p_{ijw}$, while $F^{(t)} < \underline{F}_{w_k}^{(t)}$ also holds, which is obviously an infeasible solution to employee w_k. To avoid such infeasible assignment of jobs to employee w_k with the solution of $\{y_{ijw_k}^{(t)}, F^{(t)}\}$, we can further add the following strenghthened combinatorial Benders cuts into the master problem:

$$\sum_{(i,j,w_k)\in\tilde{\mathcal{U}}_{w_k}^{(t)}} y_{ijw_k} p_{ijw_k} + \left(\left\lceil \left(\sum_{(i,j,w_k)\in\tilde{\mathcal{U}}_{w_k}^{(t)}} y_{ijw_k} p_{ijw_k} \right)/d_{w_k} - 1 \right\rceil\right) b_{w_k} \leq F^{(t)}, \tag{8}$$

where $\tilde{\mathcal{U}}_{w_k}^{(t)} = \{(i,j,w_k)|y_{ijw_k}^{(t)} = 1, F^{(t)} < \underline{F}_{w_k}^{(t)}\}$.

4.3. Lower and upper bounds on the F values

As we can see, the constraints on the value of F variable in the initial master problem are quite loose. In fact, we can use some auxiliary optimization models to generate tighter restrictions on the range of F values.

Consider the following optimization model:

$$\text{minimize} \quad C_{\max} \tag{9a}$$

$$\text{subject to} \quad \text{Constraints } (1b) - (1m). \tag{9b}$$

Note that the above optimization model (9) is much easier to solve than the IEMSCW problem. We let $\underline{C}_{\max}^{*}$ be the corresponding optimal objective for model (9), then obviously the makespan of any feasible schedule for the IEMSCW problem should be at least $\underline{C}_{\max}^{*}$. Therefore, we can obtain a lower bound on the F value as follows:

$$F \geq \underline{C}_{\max}^{*}. \tag{10}$$

On the other hand, consider the following optimization model:

$$\text{minimize} \quad C_{\max} \tag{11a}$$

$$\text{subject to} \quad \text{Constraints } (4b) - (4c) \text{ and } (4e) - (4f). \tag{11b}$$

The above optimization model (11) calculates the minimum makespan under the given assignment plan of jobs (i.e., $y_{ijw}^{(t)}$). If model (11) is feasible, we let $\underline{C}_{\max}^{(t)}$ be the corresponding optimal objective to model (11), and define

$$U_B^{(t)} = f_M^{(t)} - \theta F^{(t)} + \theta \underline{C}_{\max}^{(t)}. \tag{12}$$

In addition, if $\{\underline{x}^{(t)}, \underline{z}^{(t)}, \underline{\beta}^{(t)}\}$ is the corresponding feasible solution to the optimization model (11), then the solution $\{\underline{x}^{(t)}, \underline{z}^{(t)}, \underline{\beta}^{(t)}, \underline{C}_{\max}^{(t)}\}$ is obviously a feasible solution to the IEMSCW problem, with objective value $U_B^{(t)}$, which is an upper bound on the total cost for the IEMSCW problem.

Note that $\underline{DL}$ is a lower bound on the sum of machine depreciation and labor costs (see Section 3), therefore the makespan related cost is at most $U_B^{(t)} - \underline{DL}$, which means that the maximum makespan is at most $\left(U_B^{(t)} - \underline{DL}\right)/\theta$. Therefore, we can obtain the following upper bound on the F value:

$$F \leq \left\lfloor (U_B^{(t)} - \underline{DL})/\theta \right\rfloor. \tag{13}$$

For simplicity, we define $\underline{F} = \underline{C}_{\max}^{*}$ and $\overline{F} = \left\lfloor (U_B^{(t)} - \underline{DL})/\theta \right\rfloor$ as the lower and upper bounds on the F values, respectively. Then, by incorporating the above lower and upper bounds on the F values, we can significantly shorten the length of the time horizon during each iteration, and thus speed up the entire search process.

4.4. Monotonic search strategy: avoiding stagnation situations

It should be noted that all of the generated combinatorial Benders cuts in Section 4.2 remain valid only when the length of the new time horizon in the $(t+1)$th iteration does not increase, that is, $F^{(t+1)} \leq F^{(t)}$. Otherwise, if we obtained a new solution to the master problem in the $(t+1)$th iteration, by adding all the feasibility cuts generated after the tth iteration, and have $F^{(t+1)} > F^{(t)}$, then the previous assignment plan of jobs (i.e., $y_{ijw}^{(t)}$) may actually be feasible over the new time horizon with length $F^{(t+1)}$. Therefore, each time when the value of F strictly increases, we have to remove all the previously generated combinatorial Benders cuts, and restart the search process for feasibility cuts with the new value of F from scratch.

Meanwhile, it may be the case that the solution in the $(t+2)$th iteration (i.e., $\{y_{ijw}^{(t+2)}, F^{(t+2)}\}$) satisfies $F^{(t+2)} < F^{(t)} < F^{(t+1)}$. That is, in the new solution generated in the $(t+2)$th iteration, the time horizon is shortened with a lower makespan-related cost, at the expense of higher machine depreciation costs and labor costs. As a result, the search process may stagnate in local optima because of the fluctuations in the values of F and the removal of combinatorial Benders cuts when the value of F increases.

To avoid such situations, we implement the following monotonic search strategy: We separately solve the master problem

into two opposite directions. That is, during the implementation of Algorithm CBD, we search for the optimal solutions in non-decreasing and nonincreasing orders of the F values, respectively. For simplicity, we call them the "master-upward" and "master-downward" problems, and denote the corresponding slave problem in the upward and downward directions as "slave-upward" and "slave-downward" problems, respectively.

In particular, during any two consecutive iterations in the "master-upward" (resp. "master-downward") problem, the value of F in fact can only be increased (resp. decreased) by at most 1. The reason is the following: suppose that we have obtained a solution $(y_{ijw}^{(t)}, F^{(t)})$ by solving the "master-upward" problem in the tth iteration, and the corresponding slave problem is infeasible. Then, we add all the combinatorial Benders cuts that are generated in Section 4.2, and obtain a new solution $(y_{ijw}^{(t+1)}, F^{(t+1)})$ for the master problem. Now, if we have $F^{(t+1)} - F^{(t)} \geq 2$, then since the value of F increases monotonically in the "master-upward" problem, we know that for any iteration $t' > t+1$, the value of $F^{(t')}$ must be greater than or equal to $F^{(t+1)}$, which means that any solution (y'_{ijw}, F') that satisfies $F^{(t)} < F' < F^{(t+1)}$ will be skipped. However, it may be the case that the corresponding slave problem of solution (y'_{ijw}, F') is feasible, and the minimal total cost may be derived from this solution. Similar arguments hold for search processes in the "master-downward" problem.

Based on the above observation, starting from the value of $F^{(0)}$, which was obtained by solving the initial "master-upward" problem, we solve its associated "slave-upward" problem and check if it is feasible. If not, we add valid combinatorial Benders cuts into the master problem, and continue the above process until the slave problem becomes feasible, or the master problem becomes infeasible. To be specific, suppose in iteration t, the corresponding slave problem of solution $(y_{ijw}^{(t)}, F^{(t)})$ is feasible, we calculate the corresponding objective value of the master problem, and set it as the new upper bound $U_B^{(t)}$ on the total cost, with an updated value of $\overline{F}$. Then, in the next iteration, we update the value of F from $F^{(t)}$ to $F^{(t)}+1$ in the objective function of the master problem, and add the following constraint to the master problem

$$f_M \leq U_B^{(t)} - 1, \tag{14}$$

so as to ensure that the total cost is at most $U_B^{(t)} - 1$ in the following iterations, and continue such process until the value of F becomes larger than $\overline{F}$. In the "master-downward" problem, we repeat similar process starting from the value of $F^{(0)} - 1$, and decrease the value of F by at most 1 during any two consecutive iterations. This way, we can avoid any stagnation situations, and make it possible to find an optimal solution of the IEMSCW problem during the search process.

4.5. The algorithmic outline of Algorithm CBD

The overall procedure of the combinatorial Benders decomposition based approach (i.e., Algorithm CBD) is summarized as follows (see Algorithm 4.1).

Algorithm 4.1 The algorithmic outline of Algorithm CBD for the IEMSCW problem.

1: Initialize: $f_M^* \leftarrow 0$, $\mathcal{O} \leftarrow \emptyset$, and U_B is a sufficient large number.
2: Solve the optimization model~(9) to determine the value of $\underline{F}$, and add valid inequality~(10) to the initial master problem.
3: Solve the initial master problem (i.e., model~(3)), and calculate $\mathcal{S}^{(0)} = \{f_M^{(0)}, y_{ijw}^{(0)}, F^{(0)}\}$.
4: Solve the optimization model~(11),and calculate the value of $\overline{F}$ by inequality~(13).
5: **while** $F^{(t)} \leq \overline{F}$ **do**
6: **while** the "slave-upward" problem is infeasible with the generated tth solution $\{y_{ijw}^{(t)}, F^{(t)}\}$ **do**
7: Add combinatorial Benders cuts~(6) to the "master-upward" problem.
8: **for** each $w \in \mathcal{W}$, solve model~(5) **do**
9: **if** model~(5) is infeasible for w_k **then**
10: Add combinatorial Benders cut~(7) to the "master-upward" problem.
11: **if** $F^{(t)} < \underline{F}_w^{(t)}$ **then**
12: Add combinatorial Benders cut~(8) to the "master-upward" problem.
13: Fix $F^{(t+1)} \leftarrow F^{(t)}$ in model~(3), and solve the "master-upward" problem again.
14: Update $U_B^{(t+1)} \leftarrow U_B^{(t)}$.
15: **if** the "master-upward" problem is feasible **then**
16: Calculate an updated solution of $\{f_M^{(t+1)}, y_{ijw}^{(t+1)}\}$ with the fixed value of $F^{(t+1)}$.
17: **else**
18: $F^{(t+1)} \leftarrow F^{(t)} + 1$.
19: Remove all the combinatorial Benders cuts from model~(3), and solve the "master-upward" problem again with the fixed value of $F^{(t+1)}$.
20: Break.
21: Record the solution of $\{x^{(t)}, z^{(t)}, \beta^{(t)}\}$ when the slave problem becomes feasible.Update $U_B^{(t+1)} \leftarrow \min\{f_M^{(t)}, U_B^{(t)}\}$ and calculate the value of $\overline{F}$ by Inequation (13).
22: $F^{(t+1)} \leftarrow F^{(t)} + 1$.
23: Remove all the combinatorial Benders cuts frommodel~(3). Add constraint~(14) intothe "master-upward" problem, and solve it again with the fixed value of$F^{(t+1)}$.
24: $F^{(t)} \leftarrow F^{(0)} - 1$.
25: **while** $F^{(t)} \geq \underline{F}$ **do**
26: Repeat similar procedures as in Steps 5–23 in the downward direction, except thatwe decrease the value of F during each iteration in a similar way.
27: Output the corresponding optimal assignment decisions$\{y_{ijw}^*, F^*\}$, and its associated optimal objective value f_M^*.

5. An LS-based heuristic approach: scheduling incompatible jobs on parallel dedicated machines with working-time restrictions

From Section 4.1, we know that once we solved the initial master problem, we can obtain an assignment plan of jobs to machines and employees, with a minimal lower bound $f_M^{(0)}$ on total costs. In particular, if we can find a feasible schedule of jobs within the time horizon $[0, F^{(0)}]$, then we immediately obtain an optimal schedule for the IEMSCW problem. However, we expect that this may not always happen since the value of $F^{(0)}$ is obtained by scheduling jobs on machines or by employees without any idleness, and usually that may not satisfy the working-time restrictions.

Inspired by the above observation, one natural question that can be raised is the following: once we solved the initial master problem and fixed the corresponding assignment of jobs to machines and employees, can we easily find a feasible schedule that minimizes the makespan while satisfying the working-time restrictions, and then use such a schedule as an approximate solution to the IEMSCW problem? Note that in the above problem, the jobs to be processed on each machine are known in advance. That is, the machines are *parallel* and *dedicated*. Meanwhile, the jobs that are assigned to the same employee cannot be processed simultaneously, i.e., some of the jobs are *incompatible*, and only compatible jobs can be processed in parallel. For simplicity, we refer to

the above problem as the problem of *scheduling incompatible jobs on parallel dedicated machines with working-time restrictions*. Using the standard three-field scheduling notation (Graham, Lawler, Lenstra, & Kan, 1979), we denote this problem by *PDm|incmp, consec, break*|$C_{\max}$.

There have been several studies of scheduling problems for parallel dedicated machines, in which jobs are subject to various types of resource constraints or even have incompatibility relationships. For example, Kellerer and Strusevich (2003) studied the problem of scheduling jobs on parallel dedicated machines subject to a single resource constraint to minimize the makespan. In a follow-up work, Kellerer and Strusevich (2004) considered scheduling problems with parallel dedicated machines subject to multiple resource constraints. Moreover, Kellerer and Strusevich (2008) further considered a problem of scheduling jobs on parallel dedicated machines to minimize the makespan, in which jobs may be assigned an additional resource with reduced processing time, and no two jobs are allowed to use the resource simultaneously. In all of the above work, the authors investigated the complexity results for different variants of the problems, proposed different heuristic algorithms and analyzed the worst-case behaviors of these algorithms. In addition, Grigoriev, Sviridenko, and Uetz (2007) proposed approximation algorithms for an unrelated parallel dedicated machine scheduling problem, in which the processing time of jobs can be reduced by utilizing a discrete renewable resource. More recently, Lushchakova and Strusevich (2010) proposed a linear-time algorithm for the problem of scheduling incompatible tasks on two machines. However, according to our literature review, no work has been done on scheduling problems with parallel dedicated machines subject to working-time constraints.

From Theorem 1, it is trivial to see that problem *PD*1|*incmp, consec, break*|$C_{\max}$ is strongly NP-hard. In fact, we can further show that it is still NP-hard in the ordinary sense even when jobs can be processed preemptively (see Appendix B for proof).

Theorem 2. *Problem PD*1|*incmp, consec, break, pmtn*|$C_{\max}$ *is NP-hard in the ordinary sense.*

Given the above complexity results, we know how difficult it could be to find an optimal solution in polynomial time even for the *PDm|incmp, consec, break, pmtn*|$C_{\max}$ problem, which provides a lower bound on the makespan of the general *PDm|incmp, consec, break*|$C_{\max}$ problem. Then, one interesting question we may ask is: does there exist any approximate algorithm such that the makespan of the schedule generated is within the value of $\rho F^{(0)}$, where $\rho > 1$ is a constant? If this condition holds, we let σ be the feasible schedule generated by the approximate algorithm, and $C^{\sigma}_{\max}$ be its corresponding makespan obtained, and we have $C^{\sigma}_{\max} \le \rho F^{(0)}$. Obviously, the objective value of the algorithm is $f_M^{(0)} - \theta F^{(0)} + \theta C^{\sigma}_{\max}$. For simplicity, we define OPT as the optimal objective value of the IEMSCW problem, and we have

$$\frac{f_M^{(0)} - \theta F^{(0)} + \theta C^{\sigma}_{\max}}{OPT} \le \frac{f_M^{(0)} - \theta F^{(0)} + \theta C^{\sigma}_{\max}}{f_M^{(0)} - \theta F^{(0)} + \theta F^{(0)}} \le \frac{C^{\sigma}_{\max}}{F^{(0)}} \le \rho,$$

that is, such algorithm will also have a performance ratio of ρ for the IEMSCW problem.

Unfortunately, we have no clue how to answer the question above. The reason is the following: when each employee has to process at least one job on each machine, and if the minimum break time of each employee is sufficiently long, while the total processing time of jobs for each employee is less than this employee's maximum consecutive working time, then the *PDm|incmp, consec, break*|$C_{\max}$ problem includes the problem of minimizing the makespan for a no-wait open shop as a special case. Until recently, the approximability of makespan minimization in no-wait open shops remains a major open problem, and few theoretical results have been obtained so far (Allahverdi, 2016).

As a result, we instead try to design some simple and intuitive heuristics to search for a feasible schedule of the IEMSCW problem, and test its performance empirically. One typical idea is to employ a so called *list scheduling* heuristic, which has been often implemented in searches for feasible solutions to various parallel machine scheduling problems (e.g. Mokotoff, Jimeno, & Gutiérrez, 2001). In this work, we propose the following greedy list scheduling algorithm (Algorithm 5.1) for the IEMSCW problem, in which

Algorithm 5.1 The list scheduling based heuristic algorithm for the IEMSCW problem.

1: Solve the initial master problem, and determine the corresponding assignment plan of jobs to machines and employees according to the values of $y^{(0)}_{ijw}$.
2: Generate a list of the employees in an arbitrary order.
3: Define $\mathcal{J}_w = \{j \in \mathcal{J} | \exists i \in \mathcal{M}, j \in \mathcal{J},$ s.t. $y^{(0)}_{ijw} = 1\}$ be theset of jobs that are assigned to employee $w \in \mathcal{W}$, and$\mathcal{M}_w = \{i \in \mathcal{M} | \exists j \in \mathcal{J}_w,$ s.t. $y^{(0)}_{ijw} = 1\}$ be the set of machines that are used toprocess the jobs in $\mathcal{J}_w$.
4: Initialize: $t_w \leftarrow 0$ for $w \in \mathcal{W}$.
5: **for** Each employee w according to the employee list **do**
6: **while** $\mathcal{J}_w$ is not empty **do**
7: – Check if some yet unscheduled job $j' \in \mathcal{J}_w$ can bestarted at time t_w on an idle machine $i' \in \mathcal{M}_w$ without violating theconsecutive working-time restriction. If yes, schedule job j' to start attime t_w on machine i'; ties are broken arbitrarily. Update$\mathcal{J}_w \leftarrow \mathcal{J}_w \backslash \{j'\}$ and the associated machine set $\mathcal{M}_w$, update $t_w \leftarrow t_w + p_{i'j'w}$.
8: – If no job can be scheduled on any of the machines in $\mathcal{M}_w$ at time t_w, update t_w to the next machine available time.

we first generate a list of the employees by sorting them in an arbitrary order, and then process the jobs according to the order of employees. That is, we will not consider processing the jobs that belong to an employee until all the jobs of the previous employee on the list have been scheduled. In addition, all the jobs belonging to the same employee will be processed with no inserted idle time according to their machines' available times, while taking the working-time restrictions into account. This way, we aim to ensure that each employee can process jobs as much as possible consecutively. For simplicity, we refer to this algorithm as Algorithm LS.

6. Experimental study

6.1. Computational environment

To test the performance of the MILP model and the three algorithms we proposed, i.e., Algorithms TD, CBD and LS, we empirically conducted computational experiments on randomly generated instances. We considered instances in which the number of jobs n was 15, 20, 30 and 50, and the number of employees ℓ and machines m were chosen from the following combinations:

$$(\ell, m) \in \{(3,3), (3,6), (6,3), (5,5), (5,10), (10,5)\}.$$

To better evaluate the performance of these algorithms, we considered two different manufacturing environments: the stable (Case I) and unstable (Case II) environments. For each of the above combinations and manufacturing environments, we randomly generated 5 instances, for a total of $4 \times 6 \times 2 \times 5 = 240$ instances. We also randomly generated the corresponding parameter inputs, i.e. the values of $p_{ijw}, b_w, d_w,$ and ϕ_w as follows, in which we assume that $1 \le b_w \le d_w \le \phi_w$, as this matches typical manufacturing environments.

Table 1
Comparison of computational results by MILP-E, Algorithms TD, CBD and LS with $n = 15$.

ℓ	m	#	Case I					Case II				
			CPU running time (s)			Objective value		CPU running time (s)			Objective value	
			MILP-E	Alg TD	Alg CBD	OPT	Alg LS	MILP-E	Alg TD	Alg CBD	OPT	Alg LS
3	3	1	8	19	6	153	164	58	61	67	196	199
		2	288	24	6	140	143	88	41	21	185	189
		3	4	22	5	145	155	10	52	10	184	188
		4	3	16	5	147	152	13	41	22	179	186
		5	4	12	3	94	94	9	45	9	263	278
3	6	1	11	12	4	110	116	17	45	20	129	138
		2	6	27	24	107	111	13	61	12	119	119
		3	6	23	4	102	102	18	49	11	110	110
		4	10	11	5	81	84	14	73	7	120	120
		5	29	16	8	103	103	19	45	14	116	116
6	3	1	7	37	10	78	83	15	82	12	112	118
		2	6	34	15	148	151	13	68	11	81	81
		3	7	23	6	110	110	13	66	15	154	154
		4	7	22	5	96	99	14	108	13	131	134
		5	6	29	6	102	105	12	87	12	176	185
5	5	1	15	35	8	70	73	17	60	24	108	108
		2	10	31	7	75	81	19	119	12	105	108
		3	14	109	47	86	87	16	155	18	111	120
		4	9	45	9	75	75	16	75	16	120	129
		5	15	27	10	100	100	20	163	16	94	97
5	10	1	19	71	15	56	59	54	472	52	79	82
		2	31	71	28	78	78	42	181	35	96	99
		3	18	47	16	89	95	38	364	46	117	117
		4	19	79	20	90	93	47	221	42	103	109
		5	23	50	22	89	92	44	181	48	102	102
10	5	1	227	82	24	80	87	61	522	52	73	79
		2	29	126	27	82	82	39	597	47	94	100
		3	15	137	23	55	55	32	224	48	84	90
		4	128	236	24	103	103	31	357	54	114	114
		5	144	239	25	89	89	36	382	45	82	88
Avg CPU time			37.3	57.1	13.9	/	/	27.9	166.6	27.0	/	/
# solved			30	30	30	/	/	30	30	30	/	/

- **Stable environment (Case I):** The processing times of jobs p_{ijw} were randomly generated from the uniform distribution on $\{1, \ldots, 6\}$, b_w from the uniform distribution on $\{1, 2\}$, and d_w from the uniform distribution on $\{4, \ldots, 10\}$. It should be noted that the values of ϕ_w should be appropriately determined within a reasonable range, otherwise Constraints (1g) may be too loose to take effect on generating the solutions or be too strict which results in no feasible solutions. Through some examination, we decided to randomly generate the values of ϕ_w from the uniform distribution on $\{\lceil 3n/\ell \rceil - 3, \ldots, \lceil 3n/\ell \rceil + 3\}$. This way, the average value of the total processing time of jobs assigned to each employee will keep at a similar level as the average value of the maximum total working-time of employees. To check if the instance we generated is feasible, we can simply solve the MILP model (1), in which the objective function is set as 0. If the corresponding instance is infeasible, then we repeat the above procedure to generate a new one.
- **Unstable environment (Case II):** The processing times of jobs p_{ijw} were randomly generated from the uniform distribution on $\{1, \ldots, 10\}$, b_w from the uniform distribution on $\{1, 2, 3\}$, and d_w from the uniform distribution on $\{6, \ldots, 15\}$. In addition, the values of ϕ_w were randomly generated from the uniform distribution on $\{\lceil 5n/\ell \rceil - 5, \ldots, \lceil 5n/\ell \rceil + 5\}$. If the corresponding instance is infeasible, then we repeat the above procedure to generate a new one.

For both of the above two cases, we randomly generated the cost coefficients c_w and f_i from the uniform distribution on $\{1, \ldots, 6\}$ and the value of θ was given as 3, i.e. the mean value of c_w and f_i, so as to provide a similar weight for the makespan-related costs compared with the other two types of the costs.

To perform these experiments, we used Gurobi Optimizer 9.0.2 to solve the MILP model and the Python programming language to implement Algorithms TD, CBD and LS on a computer with a 2.8GHz Intel Core i7 processor and 16 GB of RAM running the OS X 10.14 operating system. According to our experimental study, we found that most of the instances with $n \leq 30$ can be solved optimally by at least one of the proposed solution approaches (i.e., Algorithms TD and CBD) within 30 minutes. To make a better comparison on the performance of the MILP model and the two exact algorithms with similar computational time, we set a 1800 seconds time limit on each instance. For each instance, if it cannot be solved optimally within the given time limit, we recorded the corresponding optimality gap.

In particular, as we can see, the variable z_{wt} in the MILP model (1) is merely used for convenience and can be eliminated by replacing it with an equivalent term as per Constraints (1e). For simplicity, we denote the corresponding MILP model that eliminates the variable z_{wt} as the MILP-E model. By comparing the performance of these two models over all the instances, we found that in general the MILP-E model outperforms the original MILP model, as fewer variables and constraints are included in the MILP-E model. Therefore, in what follows, we only recorded the computational results obtained by the MILP-E model, and compared its performance with the other algorithms.

6.2. Comparison between Algorithms TD, CBD, LS and the MILP-E model

We first focused on small size instances, i.e., $n = 15$, to compare the performance of the MILP-E model and our proposed algorithms. Table 1 shows the CPU running time and the objective

Table 2
Comparison of computational results by MILP-E, Algorithms TD, and CBD with $n = 20$.

ℓ	m	#	Case I							Case II						
			MILP-E				Alg TD	Alg CBD	OPT	MILP-E				Alg TD	Alg CBD	OPT
			Time (s)	Best	LB	Gap	Time (s)	Time (s)		Time (s)	Best	LB	Gap	Time (s)	Time (s)	
3	3	1	28	186	186	0	37	31	186	534	204	204	0	83	94	204
		2	–	189	181	4.2%	46	12	189	28	197	197	0	156	127	197
		3	7	232	232	0	25	4	232	–	203	191	5.9%	93	32	203
		4	22	173	173	0	29	120	173	34	248	248	0	85	18	248
		5	–	172	171	0.6%	32	39	172	–	265	257	3.0%	138	149	265
3	6	1	34	133	133	0	68	123	133	31	158	158	0	123	40	158
		2	580	147	147	0	59	71	147	28	221	221	0	127	21	221
		3	29	138	138	0	96	268	138	31	171	171	0	148	42	171
		4	18	150	150	0	83	190	150	25	184	184	0	220	38	184
		5	66	181	181	0	129	32	181	82	194	194	0	134	24	194
6	3	1	13	155	155	0	55	8	155	26	212	212	0	428	27	212
		2	316	127	127	0	62	11	127	77	181	181	0	494	30	181
		3	457	123	123	0	67	15	123	36	173	173	0	225	26	173
		4	73	155	155	0	78	9	155	34	130	130	0	182	25	130
		5	62	93	93	0	57	20	93	22	132	132	0	306	44	132
5	5	1	31	133	133	0	119	73	133	76	124	124	0	360	90	124
		2	91	102	102	0	68	20	102	39	119	119	0	274	40	119
		3	20	127	127	0	101	23	127	34	140	140	0	242	29	140
		4	–	177	175	1.1%	209	38	177	40	145	145	0	309	37	145
		5	24	133	133	0	62	16	133	39	114	114	0	523	49	114
5	10	1	417	147	147	0	252	95	147	70	86	86	0	465	62	86
		2	631	111	111	0	144	423	111	68	109	109	0	428	56	109
		3	54	87	87	0	138	36	87	59	80	80	0	857	92	80
		4	240	100	100	0	96	29	100	60	123	123	0	821	61	123
		5	59	123	123	0	162	86	123	88	120	120	0	387	60	120
10	5	1	120	97	97	0	448	47	97	71	90	90	0	1487	125	90
		2	46	97	97	0	469	54	97	–	124	119	4.0%	777	143	124
		3	–	108	105	2.8%	507	233	108	–	107	106	0.9%	1283	145	107
		4	32	82	82	0	433	46	82	240	104	104	0	1414	128	104
		5	201	135	135	0	189	43	135	72	127	127	0	752	84	127
Avg value			>362.2	/	/	0.3%	143.2	73.8	/	>304.8	/	/	0.5%	444.0	64.6	/
# solved			26				30	30	/	26				30	30	/

–: for these instances, the optimal solutions cannot be obtained within 1800 seconds.

values obtained by solving the MILP-E model and by implementing Algorithms TD (denoted by "Alg TD" in the table), CBD (denoted by "Alg CBD" in the table), and LS (denoted by "Alg LS" in the table). In addition, we also denote "OPT" as the optimal objective value, which is obtained by at least one of the above three solution approaches within the time limit.

From this table, we can see that when $n = 15$, all the instances can be solved optimally either by the MILP-E model or by Algorithms TD and CBD. In particular, the CPU running time of Algorithm CBD is comparatively smaller than the ones by Algorithm TD and by the MILP-E model. In addition, we can see that the CPU running times of Algorithms TD and CBD for most of the instances in Case I (stable environment) are generally smaller than the ones in Case II (unstable environment). This is to be expected, since the length of the planning horizon in Case II is much longer, and more alternative solutions can be found during the search process. As a result, more iterations may be required to obtain optimal solutions.

On the other hand, for the above instances that we generated, we found that the LS-based heuristic approach can obtain near-optimal objective values, which illustrates that Algorithm LS could be a promising alternative to solve the IEMSCW problem when we only have very limited computational time. Since the main focus of our paper is to investigate the exact approaches to the IEMSCW problem, we will not further compare Algorithm LS with other algorithms in the remaining of this paper.

When the number of jobs increases, i.e., $n = 20$ and $n = 30$, Tables 2 and 3 show the corresponding computational results. In the tables, we denote "Time (s)" as the computation time in CPU seconds. If an instance cannot be solved optimally within the time limit, then we used 1,800 as the computation time for the instance to compute the average value (denoted by "Avg value") of computation time, and then added a "$>$" sign before the calculated value. From these two tables, we can see that the performance of Algorithm CBD is generally better than the ones of Algorithm TD and the MILP-E model since it can solve most of the instances optimally, and its average CPU running time is smaller than those of the other two methods. Even for the instances that cannot be solved optimally by Algorithm CBD, it can still obtain near-optimal solutions with comparatively small optimality gaps. The only exceptions are the instances in Case I with $n = 30$, for which Algorithm CBD can only solve 16 out of 30 instances optimally, while the other two methods have larger numbers of solved instances.

It should be noted that when $n \geq 20$, Algorithm CBD seems to have a better performance on the instances in Case II than the ones in Case I. The reason for such phenomenon may be that: due to our proposed instance generation method, we know that the variation of the input data in Case II is higher than that in Case I. Therefore, when the size of instances increases, the valid combinatorial Benders cuts that we generated could be much tighter for the instances in Case II than the ones in Case I, since more possible infeasible solutions could be excluded by adding such cuts. Meanwhile, the performance of Algorithm CBD on instances with $(\ell, m) \in \{(6, 3), (10, 5)\}$ is generally better than the one on instances with $(\ell, m) \in \{(3, 6), (5, 10)\}$. This is reasonable, since the instances in the first set have larger numbers of employees, and as a result, fewer jobs may be assigned to each employee.

In addition, from all of these three tables, it seems that we cannot simply claim either Algorithm TD outperforms the MILP-E model, or vice versa. On one hand, we can see that even for small size instances, say $n = 15$, the average CPU running times of Al-

Table 3
Comparison of computational results by MILP-E, Algorithms TD, and CBD with $n = 30$.

ℓ	m	#	Case I										Case II												
			MILP-E				Alg TD	Alg CBD				OPT	MILP-E				Alg TD				Alg CBD				OPT
			Time (s)	Best	LB	Gap	Time (s)	Time (s)	Best	LB	Gap		Time (s)	Best	LB	Gap	Time (s)	Best	LB	Gap	Time (s)	Best	LB	Gap	
3	3	1	–	289	275	4.8%	115	115	289	289	0	289	–	431	420	2.6%	341	431	431	0	67	431	431	0	431
		2	1084	275	275	0	109	1181	275	275	0	275	–	314	298	5.1%	366	314	314	0	255	314	314	0	314
		3	34	319	319	0	162	–	324	318	1.9%	319	50	338	338	0	415	338	338	0	38	338	338	0	338
		4	43	215	215	0	112	270	215	215	0	215	–	313	273	12.8%	331	311	311	0	136	311	311	0	311
		5	–	294	273	7.1%	101	55	294	294	0	294	999	401	401	0	288	401	401	0	52	401	401	0	401
3	6	1	197	318	318	0	239	117	318	318	0	318	–	212	183	13.7%	738	212	212	0	–	215	206	4.4%	212
		2	–	188	166	11.7%	331	–	192	183	4.9%	188	84	348	348	0	616	348	348	0	–	350	344	1.7%	348
		3	536	203	203	0	399	–	204	198	3.0%	203	1025	270	270	0	1043	270	270	0	619	270	270	0	270
		4	59	243	243	0	149	777	243	243	0	243	703	232	232	0	323	232	232	0	101	232	232	0	232
		5	602	232	232	0	175	806	232	232	0	232	516	227	227	0	688	227	227	0	189	227	227	0	227
6	3	1	–	146	126	13.7%	243	47	146	146	0	146	101	237	237	0	1491	237	237	0	67	237	237	0	237
		2	–	134	116	13.4%	221	–	135	129	4.7%	134	–	224	207	7.6%	574	223	223	0	362	223	223	0	223
		3	–	199	176	11.6%	235	158	199	199	0	199	109	220	220	0	944	220	220	0	48	220	220	0	220
		4	39	297	297	0	303	42	297	297	0	297	–	265	251	5.3%	1779	265	265	0	93	265	265	0	265
		5	1584	217	217	0	215	36	217	217	0	217	863	214	214	0	857	214	214	0	43	214	214	0	214
5	5	1	–	223	220	1.3%	776	–	225	219	2.7%	223	–	271	249	8.1%	860	271	271	0	137	271	271	0	271
		2	–	125	111	11.2%	373	–	128	122	4.9%	125	86	294	294	0	1387	294	294	0	1054	294	294	0	294
		3	571	150	150	0	306	–	150	144	4.2%	150	–	159	145	8.8%	989	159	159	0	85	159	159	0	159
		4	131	131	131	0	293	–	137	125	9.6%	131	202	223	223	0	1261	223	223	0	106	223	223	0	223
		5	253	148	148	0	327	–	150	144	4.2%	148	169	222	222	0	1525	222	222	0	1374	222	222	0	222
5	10	1	–	178	162	9.0%	784	–	179	176	1.7%	178	–	145	131	9.7%	–	941	143	558.0%	–	149	143	4.2%	–
		2	1444	211	211	0	702	–	213	207	2.9%	211	–	159	143	10.1%	–	935	158	491.8%	387	159	159	0	159
		3	–	146	132	9.6%	649	–	147	141	4.3%	146	941	143	143	0	–	512	143	258.0%	121	143	143	0	143
		4	–	172	160	7.0%	629	–	175	169	3.6%	172	1107	222	222	0	–	957	215	345.1%	–	224	215	4.2%	222
		5	81	177	177	0	501	45	177	177	0	177	192	154	154	0	–	739	154	379.9%	171	154	154	0	154
10	5	1	156	146	146	0	1039	91	146	146	0	146	–	161	159	1.2%	–	876	161	444.1%	211	161	161	0	161
		2	–	147	128	12.9%	655	115	147	147	0	147	–	157	141	10.2%	–	900	157	473.2%	478	157	157	0	157
		3	258	130	130	0	661	90	130	120	0	130	–	162	143	11.7%	–	893	161	454.7%	749	162	162	0	162
		4	116	147	147	0	978	–	148	143	3.5%	147	–	197	176	10.7%	–	904	196	361.2%	448	197	197	0	197
		5	1172	189	189	0	728	117	189	189	0	189	228	195	195	0	–	952	195	388.2%	166	195	195	0	195
Avg value			> 998.7	/	/	3.8%	402.0	> 975.4	/	/	1.9%	/	> 1085.8	/	/	3.9%	> 1160.5	/	/	138.5%	> 491.9	/	/	0.5%	/
# solved			18				30	16				/	16				20				26				/

–: for these instances, the optimal solutions cannot be obtained within 1800 seconds.

Table 4
Comparison of computational results by the MILP-E model, Algorithms TD, and CBD with $n = 50$.

ℓ	m	#	Case I												Case II											
			MILP-E				Alg TD				Alg CBD				MILP-E				Alg TD				Alg CBD			
			Time (s)	Best	LB	Gap	Time (s)	Best	LB	Gap	Time (s)	Best	LB	Gap	Time (s)	Best	LB	Gap	Time (s)	Best	LB	Gap	Time (s)	Best	LB	Gap
3	3	1	–	407	368	9.6%	1160	407	407	0	792	407	407	0	–	631	529	16.2%	–	1794	623	188.0%	–	625	620	0.1%
		2	–	402	363	9.7%	972	402	402	0	–	405	397	2.0%	–	662	583	11.9%	–	1902	637	198.6%	–	665	627	6.1%
		3	–	427	362	15.2%	–	675	427	58.1%	125	427	427	0	–	674	570	15.4%	–	1512	651	132.3%	–	663	651	1.8%
		4	–	547	502	8.2%	–	890	544	63.6%	1555	547	547	0	–	534	454	15.0%	–	1540	520	196.2%	–	527	520	1.3%
		5	499	552	552	0	–	1024	552	85.5%	53	552	552	0	–	691	623	9.8%	–	1673	669	150.1%	959	682	682	0
3	6	1	–	379	344	9.2%	–	1050	369	184.6%	–	381	369	3.3%	–	421	369	12.4%	–	1730	400	332.5%	–	427	400	6.8%
		2	1173	335	335	0	–	426	323	31.9%	–	341	323	5.6%	–	495	429	13.3%	–	1738	468	271.4%	–	495	468	5.8%
		3	–	310	261	15.8%	–	567	291	94.8%	–	306	291	5.2%	–	431	368	14.6%	–	1717	401	328.2%	–	428	401	6.7%
		4	–	267	229	14.2%	1418	267	267	0	–	269	251	7.2%	–	592	555	6.3%	–	1885	583	223.3%	–	594	579	2.6%
		5	–	235	195	17.0%	1121	235	235	0	–	240	222	8.1%	–	451	407	9.8%	–	1719	448	283.7%	–	454	448	1.3%
6	3	1	–	341	311	8.7%	–	509	339	50.1%	154	341	341	0	582	514	514	0	–	1868	514	263.4%	189	514	514	0
		2	–	412	382	7.3%	–	902	411	119.5%	1305	412	412	0	–	363	325	10.5%	–	1682	363	363.4%	–	369	363	1.7%
		3	–	285	251	11.9%	–	526	285	84.6%	116	285	285	0	–	379	348	8.2%	–	1701	376	352.4%	–	382	376	1.6%
		4	–	359	351	2.2%	–	982	359	173.5%	239	359	359	0	–	591	553	6.4%	–	1911	588	225.0%	–	594	588	1.0%
		5	–	414	355	14.3%	–	519	414	25.4%	203	414	414	0	–	465	431	7.3%	–	1802	465	287.5%	248	465	465	0
5	5	1	–	302	272	9.9%	–	718	301	138.5%	–	310	301	3.0%	–	302	262	13.2%	–	1658	292	467.8%	–	301	292	3.1%
		2	–	424	390	8.0%	–	743	415	79.0%	–	436	415	5.1%	–	478	459	4.0%	–	1820	476	282.4%	1278	478	478	0
		3	–	377	337	10.4%	–	877	369	137.7%	–	379	369	2.7%	–	367	338	7.9%	–	1751	365	379.7%	–	367	365	0.1%
		4	–	251	216	14.3%	–	502	247	103.2%	–	259	247	4.9%	–	443	419	5.4%	–	1837	440	317.5%	–	449	440	2.0%
		5	–	498	491	1.4%	–	1263	492	156.7%	–	510	492	3.7%	–	348	326	6.3%	–	1759	346	408.4%	–	352	346	1.7%
5	10	1	–	273	239	12.5%	–	1048	265	295.5%	–	274	265	3.4%	–	266	238	10.5%	–	1684	262	542.7%	–	271	262	3.4%
		2	–	245	206	15.9%	–	928	234	296.6%	–	246	234	5.1%	–	322	293	9.0%	–	1726	316	446.2%	–	322	316	1.9%
		3	–	193	165	14.5%	–	952	183	420.2%	–	201	183	9.8%	776	357	357	0	–	1782	356	400.6%	945	357	357	0
		4	–	293	261	10.9%	–	1036	291	256.0%	–	303	291	4.1%	–	302	275	8.9%	–	1721	297	479.5%	–	309	297	4.0%
		5	–	231	194	16.0%	–	1020	220	363.6%	–	244	220	10.9%	–	294	255	13.3%	–	1688	288	486.1%	–	294	288	2.1%
10	5	1	–	274	245	10.6%	–	1067	263	305.7%	–	275	263	4.6%	–	318	285	10.4%	–	1711	317	439.7%	1390	318	318	0
		2	–	248	220	11.3%	–	1049	248	323.0%	437	248	248	0	–	324	295	9.0%	–	1719	324	430.6%	645	324	324	0
		3	–	257	231	10.1%	–	1064	257	314.0%	569	257	257	0	945	302	302	0	–	1709	300	469.7%	–	306	300	2.0%
		4	–	350	313	10.6%	–	1119	350	219.7%	522	350	350	0	–	343	322	6.1%	–	1766	343	414.9%	–	349	343	1.7%
		5	1697	345	345	0	–	1175	341	244.6%	–	347	341	0	–	213	189	11.3%	–	1631	210	676.7%	–	219	210	4.3%
Avg value			> 1732.3	/	/	10.0%	> 1715.7	/	/	154.2%	> 1282.3	/	/	3.0%	> 1696.8	/	/	8.9%	> 1800	/	/	348.0%	> 1568.5	/	/	2.1%
# solved			3				4				12				3				0				7			

–: for these instances, the optimal solutions cannot be obtained within 1800 seconds.

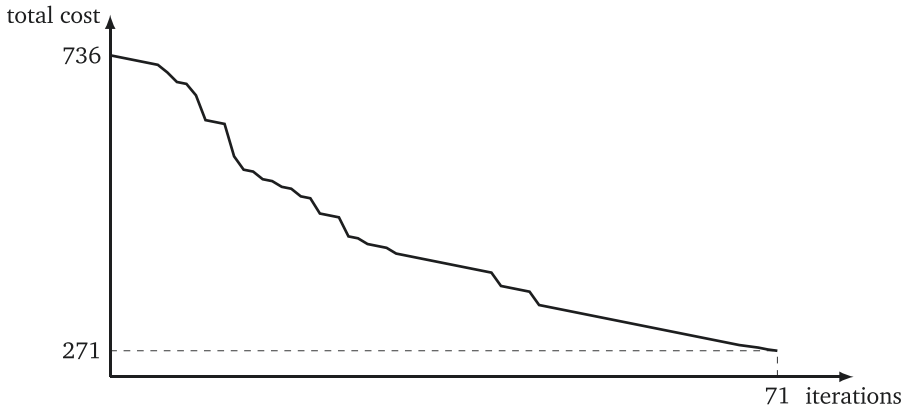

Fig. 1. The search process of Algorithm TD.

Table 5
Comparison of computational times between Algorithms CBD-O and CBD-M with $n = 15$ and $n = 20$.

ℓ	m	#	$n = 15$				$n = 20$			
			Case I		Case II		Case I		Case II	
			Alg CBD-O	Alg CBD-M	Alg CBD-O	Alg CBD-M	Alg CBD-O	Alg CBD-M	Alg CBD-O	Alg CBD-M
3	3	1	6	11	67	90	31	61	94	113
		2	6	3	21	27	12	8	127	67
		3	5	7	10	7	4	5	32	56
		4	5	9	22	20	120	113	18	32
		5	3	4	9	8	39	143	149	412
3	6	1	4	5	20	28	123	207	40	94
		2	24	15	12	12	71	198	21	20
		3	4	4	11	11	268	716	42	69
		4	5	5	7	7	190	276	38	45
		5	8	11	14	10	32	72	24	34
6	3	1	10	13	12	14	8	16	27	31
		2	15	13	11	12	11	13	30	34
		3	6	8	15	20	15	13	26	32
		4	5	6	13	18	9	11	25	23
		5	6	7	12	14	20	40	44	57
5	5	1	8	15	24	24	73	147	90	177
		2	7	7	12	18	20	23	40	61
		3	47	31	18	20	23	29	29	35
		4	9	7	16	17	38	18	37	35
		5	10	11	16	19	16	14	49	59
5	10	1	15	16	52	56	95	68	62	59
		2	28	31	35	32	423	911	56	61
		3	16	14	46	45	36	79	92	97
		4	20	20	42	37	29	59	61	66
		5	22	16	48	51	86	618	60	65
10	5	1	24	25	52	50	47	151	125	161
		2	27	29	47	42	54	125	143	175
		3	23	23	48	53	233	425	145	205
		4	24	24	54	51	46	89	128	153
		5	25	21	45	46	43	61	84	91
Avg CPU time			13.9	13.7	27	28.6	73.8	157.0	64.6	87.3

gorithm TD for Case I and Case II are 57.1 and 166.6 seconds, respectively, which are comparatively larger than the ones by solving the MILP-E model. One reason for such results may be that: when the size of instances is small, model (1) itself can be solved efficiently within a fairly small CPU running time, therefore, the speedup factor of model (2) becomes less significant since similar computational time may be required to solve model (2) as the one by solving the MILP-E model directly. Considering that multiple iterations will be always required during the implementation of Algorithm TD, as a result, the overall CPU running time of Algorithm TD could be larger than the one by using the MILP-E model.

On the other hand, when n increases, for all the instances with $n = 20$, and the stable instances with $n = 30$, we found that some of the instances cannot be solved optimally by the MILP-E model, while Algorithm TD can still solve these instances optimally. Meanwhile, among all the instances that can be solved optimally by both methods, we find that there exists considerable amount of them for which the corresponding CPU running time of the MILP-E model are comparatively much smaller than the ones of Algorithm TD.

Moreover, for the instances in Case II with $n = 30$, we can see that some of them cannot be solved optimally by Algorithm TD. For simplicity, we let V_{TD} be the current best objective value among all the generated feasible solutions by Algorithm TD, and $V_{LB} = \min\{f_M^{(0)}, \underline{DL}\}$, and we define the optimality gap of Algorithm TD as follows:

$$\text{Optimality gap of Algorithm TD} = \frac{V_{TD} - V_{LB}}{V_{LB}} \times 100\%.$$

Note that Algorithm TD searches feasible solutions starting from the end of the time horizon. Therefore, the initial total cost generated by this algorithm is very high. If Algorithm TD fails to sig-

Table 6
Comparison of CPU running times between Algorithms CBD-O and CBD-S with $n = 15$ and $n = 20$.

ℓ	m	#	$n = 15$				$n = 20$			
			Case I		Case II		Case I		Case II	
			Alg CBD-O	Alg CBD-S	Alg CBD-O	Alg CBD-S	Alg CBD-O	Alg CBD-S	Alg CBD-O	Alg CBD-S
3	3	1	6	6	67	90			94	121
		2			21	21			127	88
		3	5	5						
		4					120	122		
		5							149	930
3	6	1					123	145		
		2					71	68		
		5					32	20		
6	3	2	15	16						
5	5	1					73	265		
		3	47	37						

nificantly reduce the value of T during the search process within the time limit, then its optimality gap can be extremely large (e.g., 558.0% for the first instance of the combination $(\ell, m, n) = (5, 10, 30)$). For more detail, please see Fig. 1 in Section 6.3.3 for a graphical view of the search process of Algorithm TD for an illustrative example.

To further test the performance of Algorithm CBD, we also conducted our computational experiments on the instances with $n = 50$. Table 4 shows the corresponding computational results. From this table, we can see that when the size of instances continuously increases, Algorithm TD could consume prohibitive CPU running time to calculate the optimal solutions, and the optimality gaps are extremely large for most of the test instances. On the other hand, Algorithm CBD still outperforms the MILP-E model in terms of both the number of solved instances and the average optimality gap, which indicates the efficiency of our proposed combinatorial Benders decomposition based approach in finding the optimal solutions or near-optimal solutions to the IEMSCW problem.

6.3. Evaluation of components in Algorithm CBD

As we have observed from Section 6.2, the combinatorial Benders decomposition algorithm we proposed could be quite effective in finding optimal solutions or near-optimal solutions to the IEMSCW problem. It should be noticed that all of the components in Algorithm CBD contribute to its good performance. In what follows, we conducted several experiments on Algorithm CBD to further identify the impact of each component.

6.3.1. Combinatorial Benders cuts: heuristic generation method vs. minimal infeasible subset

From Section 4.2, we know that the combinatorial Benders cut for each employee is generated by some straightforward heuristic methods, that is, we simply check if the obtained assignment plan of jobs is feasible to the corresponding employee. If not, we then add several combinatorial Benders cuts into the master problem. In fact, we can further find out the minimal infeasible subset of assignment plan to each employee, using a similar idea as in the work of Verstichel et al. (2015) (see Section 4.4 in their paper). To be specific, when an assignment plan $\mathcal{U}_w$ is not feasible to employee w, we enumerate each subset $\mathcal{U}'$ of $\mathcal{U}_w$ in nonincreasing order of the subset size, and check if the corresponding subset is still infeasible. If so, then we generate a new combinatorial Benders cut associated with $\mathcal{U}'$, and remove all the cuts associated with any superset that includes $\mathcal{U}'$, until no further action can be made. For simplicity, we call the corresponding combinatorial Benders decomposition algorithm with the above procedure to search for minimal infeasible subsets as Algorithm CBD-M, and call the original one as Algorithm CBD-O. Table 5 shows the corresponding computational results of CPU running times for the instances with $n = 15$ and $n = 20$.

From this table, it seems that incorporating such procedure to obtain minimal infeasible subset cannot improve the performance of Algorithm CBD for the IEMSCW problem. The reason may be that: the infeasible assignment plan $\mathcal{U}_w$ is already tight enough to each employee, and therefore is hard or even impossible to find a proper infeasible subset for it. As a result, when the size of instances increases, the way of enumeratively searching for any infeasible subset of $\mathcal{U}_w$, which includes a considerable amount of iterations, just becomes a waste of time.

6.3.2. Impact of the valid cuts generated by different types of constraints

As we can see from Section 4.2, we have proposed three different types of combinatorial Benders cuts, i.e., Constraints (6), (7) and (8). Through some examination, we found that when Algorithm CBD only includes the valid cuts generated by Constraint (6), then most of the instances with $n = 15$ cannot be solved optimally within the time limit, which indicates that such valid cuts are very loose. To further test the performance of the valid cuts that are generated by different types of constraints, we considered two variants of Algorithm CBD and compared their computational performance as follows: The first one is the original Algorithm CBD-O, which includes all types of the valid cuts proposed in Section 4.2, and the second one is the corresponding algorithm that only uses valid cuts generated by Constraints (6) and (7), and we call it Algorithm CBD-S for the sake of simplicity. It should be noticed that the combinatorial Benders cut can be generated by Constraint (8) only when the condition of $F^{(t)} < \underline{F}^{(t)}_{w_k}$ is satisfied. Therefore, we only recorded the computational results for the instances that the above condition is satisfied during some iteration. Table 6 shows the corresponding computational results, in which the blank part means that the corresponding instance has no such valid cuts. From this table, we can see that by adding Constraint (8) into the master problem, Algorithm CBD could significantly reduce the computational time to calculate the optimal solutions for some of the instances.

6.3.3. An illustrative example

To further illustrate the search processes of Algorithms CBD and TD, we also conducted some experiments on an illustrative instance, in which there are 30 jobs, 3 employees and 6 machines, with the following coefficient parameters: $c_w = \{2, 6, 4\}$, $d_w = \{6, 4, 7\}$, $b_w = \{1, 2, 1\}$, $\phi_w = \{12, 15, 18\}$, $f_i = \{1, 4, 5, 6, 4, 3\}, \theta = 3$, and the processing times (i.e., p_{ijw}) of jobs are given in Table C.8 (see Appendix C).

By implementing Algorithm CBD, we can obtain an optimal solution within a CPU running time of 40 seconds, with an optimal

objective value of 271. During the search process, the number of iterations searched in the upward directions is 4, and a feasible solution to the slave problem was found. Meanwhile, no iteration is conducted in the downward direction since we have $F^{(0)} = \underline{F}$ for this example.

This instance can also be solved optimally by Algorithm TD within a CPU running time of 203 seconds, while the MILP-E model cannot solve it within the time limit, with an optimality gap of 0.36%. It should be noticed that although the objective values obtained by these approaches are the same, their optimal solutions can be different. In addition, the number of iterations in Algorithm TD is 71. Fig. 1 shows the corresponding search process of Algorithm TD, from which we can see that at the very beginning of the search process, the total cost of the initial solution obtained by Algorithm TD is very high, and then gradually decreases during the search process. This observation may help explain why the optimality gap of Algorithm TD is extremely large for most of the instances with large sizes (see Tables 3 and 4).

7. Conclusions

In this paper, we studied an integrated employee and parallel machine scheduling problem with maximum consecutive working-time and minimum break time restrictions. The objective is to minimize the weighted sum of the makespan, the machine depreciation costs and the labor costs. To solve this problem, we proposed a mixed integer linear programming formulation, two different decomposition based exact solution methods, i.e., Algorithms TD and CBD, and an LS-based heuristic algorithm, i.e., Algorithm LS. To test the efficiency of our proposed solution approaches, we conducted extensive computational experiments on randomly generated instances. The computational results show that the combinatorial Benders decomposition based approach could be quite effective in finding optimal or near-optimal solutions to the IEMSCW problem within comparatively smaller CPU running times.

This work could be extended in several directions. First of all, this work considered the sum of the total costs related to job assignments and the weighted makespan as a single objective. This can be easily extended to a bi-objective problem with Pareto optimization to consider the tradeoff between the costs and machine utilization simultaneously. Second, it would also be interesting to see if any other stronger combinatorial Benders cuts could be generated by further investigating the structural properties of the optimal solutions to the IEMSCW problem. Third, employees' preferences with regard to the jobs' processing may also be included in the model to further reflect the priorities of the employees. Finally, as for alternative solution methods, column generation based methods may possibly be developed for the time-indexed formulation we proposed in this study, since column generation based methods have already been successfully implemented in many parallel machine scheduling problems, in which the problem could be reformulated as a set partitioning problem and a schedule for an individual employee may serve as a pricing problem.

Acknowledgment

The authors would like to thank the anonymous referees for their constructive comments which contributed to improve the quality of this paper. This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 71701144, 71571135, 71971155 and 91646118. The work was also supported by the Fundamental Research Funds for the Central Universities, and the Science & Technology Pillar Key Program of Tianjin Key Research and Development Plan (20YFZCGX00640).

Appendix A. An assignment and positional formulation for the IEMSCW problem

As we mentioned in Section 2.1, the IEMSCW problem can also be described by a continuous time formulation by directly assigning jobs to positions, which is a generalization of the original single machine formulation proposed by Lasserre and Queyranne (1992), and we call such formulation the *assignment and positional formulation*, or the AP formulation for short. Note that each employee has restrictions on his/her maximum consecutive working-time, minimum break time, and total working-time, so it is natural to define assignment and positional variables based on employees, but not on machines.

Meanwhile, we also need to ensure that at any time instant t, each machine cannot process any two jobs simultaneously. Fig. A.1 gives an example of a feasible schedule, in which job j is the kth job processed by employee w, job h is the ℓth job processed by employee u, and these two jobs are running concurrently during time period $[t_1, t_2)$, then we can obtain that jobs j and h must be processed on different machines, otherwise it becomes an infeasible schedule.

In order to keep track of which jobs are running concurrently at any time instant, we introduce binary variables to compare the relationship between the start and completion times of any two jobs that are assigned to different positions of employees. To be specific, we introduce the following variables:

- δ_{ijwk} is equal to 1 if job j is processed on machine i, and is the kth job that was processed by employee w, and 0 otherwise;
- S_{wk} is the start time of the kth job that was processed by employee w;
- C_{wk} is the completion time of the kth job that was processed by employee w;
- $x_{wku\ell}$ is equal to 1 if the start time of the kth job processed by employee w is less than or equal to the start time of the ℓth job processed by employee u (in other words, $S_{wk} \le S_{u\ell}$), and 0 otherwise;
- $y_{wku\ell}$ is equal to 1 if the completion time of the kth job processed by employee w is greater than the start time of the ℓth job processed by employee u (in other words, $C_{wk} > S_{u\ell}$), and 0 otherwise;
- $z_{wku\ell}$ is equal to 1 if the start time of the ℓth job processed by employee w occurs during the processing of the kth job processed by employee w (in other words, $S_{wk} \le S_{u\ell} < C_{wk}$), and 0 otherwise;
- σ_{wk} is equal to 1 if $C_{wk} = S_{w,k+1}$, and 0 otherwise;
- $\eta_{wk\ell}$ is equal to 1 if $\sigma_{wk} = \sigma_{w,k+1} = \cdots = \sigma_{w,\ell-1}$ for $k < \ell$, and 0 otherwise;
- $C_{\max}$ is the makespan of the schedule.

Then, the IEMSCW problem can be formulated as follows:

$$\text{minimize} \quad \sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\sum_{w\in\mathcal{W}}\sum_{k\in\mathcal{J}}(f_i + c_w)p_{ijw}\delta_{ijwk} + \theta C_{\max} \tag{A.1a}$$

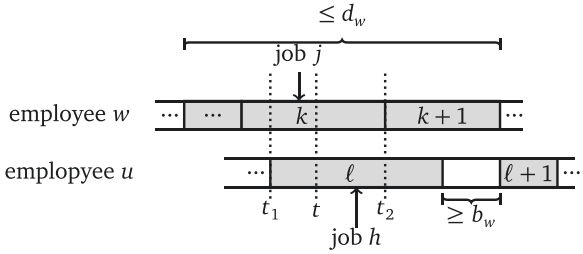

Fig. A.1. Concurrent jobs in a feasible schedule.

subject to $\sum_{i\in\mathcal{M}}\sum_{w\in\mathcal{W}}\sum_{k\in\mathcal{J}}\delta_{ijwk}=1 \quad \forall j\in\mathcal{J};$ (A.1b)

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\delta_{ijwk}\le 1 \quad \forall w\in\mathcal{W},k\in\mathcal{J};$$ (A.1c)

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\delta_{ijwk}\ge\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\delta_{ijw,k+1} \quad \forall w\in\mathcal{W},k\in\{1,2,\ldots,n-1\};$$ (A.1d)

$$C_{\max}\ge C_{wk} \quad \forall w\in\mathcal{W},k\in\mathcal{J};$$ (A.1e)

$$C_{w1}\ge\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}p_{ijw}\delta_{ijw1} \quad \forall w\in\mathcal{W};$$ (A.1f)

$$C_{wk}\ge C_{w,k-1}+\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}p_{ijw}\delta_{ijwk} \quad \forall w\in\mathcal{W},k\in\{2,\ldots,n\};$$ (A.1g)

$$C_{wk}=S_{wk}+\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}p_{ijw}\delta_{ijwk} \quad \forall w\in\mathcal{W},k\in\mathcal{J};$$ (A.1h)

$$S_{u\ell}-S_{wk}\le Mx_{wku\ell}-1 \quad \forall w,u\in\mathcal{W},k,\ell\in\mathcal{J};$$ (A.1i)

$$S_{wk}-S_{u\ell}\le M(1-x_{wku\ell}) \quad \forall w,u\in\mathcal{W},k,\ell\in\mathcal{J};$$ (A.1j)

$$C_{wk}-S_{u\ell}\le My_{wku\ell} \quad \forall w,u\in\mathcal{W},k,\ell\in\mathcal{J};$$ (A.1k)

$$S_{u\ell}-C_{wk}\le M(1-y_{wku\ell})-1 \quad \forall w,u\in\mathcal{W},k,\ell\in\mathcal{J};$$ (A.1l)

$$x_{wku\ell}+y_{wku\ell}=1+z_{wku\ell} \quad \forall w,u\in\mathcal{W},k,\ell\in\mathcal{J};$$ (A.1m)

$$z_{wku\ell}\le x_{wku\ell} \quad \forall w,u\in\mathcal{W},k,\ell\in\mathcal{J};$$ (A.1n)

$$z_{wku\ell}\le y_{wku\ell} \quad \forall w,u\in\mathcal{W},k,\ell\in\mathcal{J};$$ (A.1o)

$$\delta_{ijwk}+\delta_{ij'u\ell}+z_{wku\ell}\le 2 \quad \forall i\in\mathcal{M},j,j'\in\mathcal{J},w,u\in\mathcal{W},k,\ell\in\mathcal{J};$$ (A.1p)

$$S_{w,k+1}-C_{wk}\le M(1-\sigma_{wk}) \quad \forall w\in\mathcal{W},k\in\{1,\ldots,n-1\};$$ (A.1q)

$$C_{wk}-S_{w,k+1}\le M\sigma_{wk}-b_w \quad \forall w\in\mathcal{W},k\in\{1,\ldots,n-1\};$$ (A.1r)

$$\sum_{h=k}^{\ell-1}\sigma_{wh}\ge(\ell-k)\eta_{wk\ell} \quad \forall w\in\mathcal{W},k,\ell\in\mathcal{J}:k<\ell;$$ (A.1s)

$$1+\sum_{h=k}^{\ell-1}\sigma_{wh}\le\ell-k+\eta_{wk\ell} \quad \forall w\in\mathcal{W},k,\ell\in\mathcal{J}:k<\ell;$$ (A.1t)

$$\sum_{h=k}^{\ell}\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}p_{ijw}\delta_{ijwh}\le d_w+M(1-\eta_{wk\ell}) \quad \forall w\in\mathcal{W},k,\ell\in\mathcal{J}:k<\ell;$$ (A.1u)

$$\sum_{i\in\mathcal{M}}\sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{J}}p_{ijw}\delta_{ijwk}\le\phi_w \quad \forall w\in\mathcal{W};$$ (A.1v)

$$\delta_{ijwk},x_{wku\ell},y_{wku\ell},z_{wku\ell},\sigma_{wk},\eta_{wk\ell}\in\{0,1\} \quad \forall i\in\mathcal{M},j,k,\ell\in\mathcal{J},w,u\in\mathcal{W}.$$ (A.1w)

The objective (A.1a) minimizes the total costs including the machine depreciation costs, the labor costs, and the makespan-related costs. Constraints (A.1b) ensure that each job can only be processed on exactly one machine and by exactly one employee in exactly one position. Constraints (A.1c) ensure that each position by each employee contains at most one job processed on some machine. Constraints (A.1d) ensure that a job can be assigned to the $k+1$th position by employee w only if the kth position by employee w has been occupied. Constraints (A.1e) define the makespan. Constraints (A.1f)–(A.1h) ensure that the start and completion times are consistent with a parallel machine scheduling environment. Constraints (A.1i)–(A.1o) ensure that the positional variables x, y and z take their intended values. Constraints (A.1p) ensure that at any time instant any machine cannot process more than one jobs simultaneously. Constraints (A.1q) and (A.1r) define variable σ, and

Table A.7
Comparison of computational results by the TI and the AP formulations with $n=15$.

ℓ	m	#	Case I						Case II					
			TI formulation		AP formulation				TI formulation		AP formulation			
			Time (s)	OPT	Time (s)	Best	LB	Gap	Time (s)	OPT	Time (s)	Best	LB	Gap
3	3	1	8	153	–	161	141	12.4%	58	196	1066	196	196	0
		2	288	140	–	140	131	6.4%	88	185	–	185	182	1.62%
		3	4	145	–	149	139	6.7%	10	184	1015	184	184	0
		4	3	147	514	147	147	0	13	179	–	571	177	69%
		5	4	94	108	94	94	0	9	263	933	263	263	0
3	6	1	11	110	1091	110	110	0	17	129	596	129	129	0
		2	6	107	536	107	107	0	13	119	–	*	118	NA
		3	6	102	704	102	102	0	18	110	716	110	110	0
		4	10	81	129	81	81	0	14	120	–	*	119	NA
		5	29	103	266	103	103	0	19	116	588	116	116	0
6	3	1	7	78	–	*	69	NA	15	112	–	*	103	NA
		2	6	148	–	*	146	NA	13	81	–	*	74	NA
		3	7	110	–	*	107	NA	13	154	–	*	143	NA
		4	7	96	–	*	84	NA	14	131	–	*	121	NA
		5	6	102	–	*	92	NA	12	176	–	*	176	NA

–: for these instances, the optimal solutions cannot be obtained within 1800 seconds. *: for these instances, even a feasible solution cannot be obtained within 1800 seconds. NA: for these instances, the optimality gap cannot be obtained within 1800 seconds.

ensure that once an employee w takes a break, the break time should be at least b_w. Constraints (A.1s) and (A.1t) establish the relationship between σ and η. Constraints (A.1u) ensure that the maximum consecutive time of employee w should be less than d_w. Constraints (A.1v) provide the upper bound of total working time for each employee.

In addition, we can also add the following valid inequalities into the above model, which can considerably reduce the computational times to calculate the optimal solutions for some of the instances:

$$S_{w,k+1} - C_{wk} \le M \sum_{i\in\mathcal{M}} \sum_{j\in\mathcal{J}} \delta_{ijw,k+1} \quad \forall w \in \mathcal{W}, k \in \{1,\dots,n-1\}. \tag{A.1x}$$

For simplicity, we call the corresponding time-indexed formulation that eliminates the variable z_{wt} (i.e., the MILP-E model) the *TI formulation*. To evaluate the performance of the TI and AP formulations, we conducted experiments on the same small instances with $n = 15$, and $(\ell, m) \in \{(3, 3), (3, 6), (6, 3)\}$. Table A.7 shows the computational results obtained from these two formulations.

From this table, we can see that even for the small instances with $n = 15$, and $(\ell, m) \in \{(3, 3), (3, 6), (6, 3)\}$, the performance of the AP formulation is much worse than the one of the TI formulation both from the CPU running time and the ability of finding optimal or even feasible solutions within the given time limit. The reason may be that: due to the huge number of Big-M constraints, the AP formulation probably has a looser relaxation bound on the objective function than the TI formulation, and thus the computational time for obtaining optimal solutions increases. Based on the above observations, we believe that the AP formulation is not an appealing approach to formulate the IEMSCW problem.

Appendix B. Proof of Theorem 2

Proof. We reduce any instance of the PARTITION problem, which is NP-hard in the ordinary sense (Garey & Johnson, 1979), to an instance of problem $PD1|incmp, consec, break, pmtn|C_{\max}$. PARTITION can be described as follows: Given a set $\mathcal{S} = \{1, 2, \dots, m\}$ and positive integers $A_1, \dots, A_m$ with $\sum_{i\in\mathcal{S}} A_i = 2B$, does there exist a partition of $\mathcal{S}$ into two disjoint subsets $\mathcal{S}_1$ and $\mathcal{S}_2$ such that $\sum_{i\in\mathcal{S}_1} A_i = \sum_{i\in\mathcal{S}_2} A_i$=B? For simplicity, we also define $A_0 = B$. Given any instance $\mathcal{I}_1$ of PARTITION problem, we construct an instance $\mathcal{I}_2$ of problem $PD1|incmp, consec, break, pmtn|C_{\max}$ as follows:

The set of employees is $\mathcal{W} = \{w_0, w_1, \dots, w_m\}$. Employee w_0 needs to process 5 jobs with processing time $A_0 = B$, each employee w_i needs to process two identical jobs with processing time A_i for $i = 1, \dots, m$. The maximum consecutive working-time of employee $w_i \in \mathcal{W}$ is A_i for $i = 0, \dots, m$. The minimum break time of employee w_0 is B, and the minimum break time of employee w_i is $3B$ for $i = 1, \dots, m$. Given the above set of jobs, we can see that the total processing time of jobs is $9B$.

Now if there exists a schedule with a makespan of $9B$, then obviously employee w_0 must process his/her jobs non-preemptively during intervals $[2kB, (2k + 1)B]$ for $k = 0, 1, 2, 3, 4$, otherwise the makespan exceeds $9B$. As a result, each employee w_i can only process his/her jobs within blocks 1 to 4, for $i = 1, \dots, m$. (Fig. B.1)

Note that each employee w_i has to process two identical jobs, each with a processing time of A_i. Now if some employee w_i processes part of his/her first job in block 1, then this employee must process his/her first job entirely in block 1. If not, then the remaining part of this job has to be processed after at least $3B$ time, which means that the remaining part of the first job has to lie in block 3 or block 4. Although the remaining part of this job can be processed together with employee w_i's second job, given A_i as employee w_i's maximum consecutive working-time, at least part of the second job cannot be finished and has to be processed again after at least $3B$ time. This, however, implies that the remaining part of the second job cannot be processed in block 3 or block 4. Thus the makespan exceeds $9B$. Similarly, if some other employee w_j processes part of his/her first job in block 2, he/she must process this job entirely.

In addition, we also claim that employee w_i must process his/her second job entirely in block 3. Otherwise, suppose employee w_i processes part of his/her second job in block 4, note that because of the minimum break time restriction, an employee cannot process jobs in both blocks 3 and 4, as a result, employee w_i must process his/her second job entirely in block 4, and block 4 was occupied by a job with length A_i. Then, it is easy to see that the total processing time in block 2 cannot be greater than $B - A_i$, since for any employee, if he/she processes the first job in block 2, then the corresponding second job can only be processed in block 4. However, to derive a makespan of $9B$, we know that the total processing time of block 2 must be equal to B, which is a contradiction. Therefore, employee w_i must process his/her second job in block 3. This means that the set of employees assigned to block 1 is exactly the same as to block 3, and the employees assigned to block 2 is exactly the same as to block 4, in which each job must be processed nonpreemptively, and each of these blocks achieves a length of B. That is, there exists a feasible schedule for the constructed instance $\mathcal{I}_2$ with a makespan $9B$ if and only if there is a solution to instance $\mathcal{I}_1$ of the PARTITION problem, and the theorem follows. □

Appendix C. Processing times of jobs for the illustrative example in Section 6.3.3

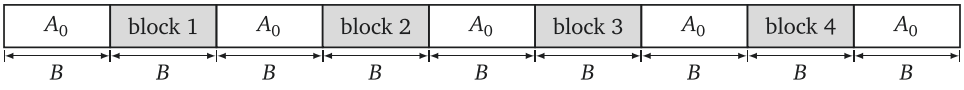

Fig. B.1. Schematic for proof of Theorem 2.

Table C.8
The processing times of jobs on machines by different employees.

Job	Employee 1						Employee 2						Employee 3					
	M_1	M_2	M_3	M_4	M_5	M_6	M_1	M_2	M_3	M_4	M_5	M_6	M_1	M_2	M_3	M_4	M_5	M_6
1	1	4	4	2	4	4	4	5	2	2	3	6	5	4	3	5	6	6
2	6	2	2	1	4	4	4	2	6	3	4	3	3	6	3	6	6	1
3	1	6	1	2	2	4	3	3	6	3	1	4	4	6	3	6	1	5
4	3	4	4	1	3	1	5	2	6	5	6	4	2	3	2	5	4	6
5	2	2	5	4	2	3	6	2	4	3	1	6	2	6	3	3	6	4
6	1	4	4	3	1	3	3	2	5	2	5	4	4	5	4	4	4	2
7	3	4	5	2	1	6	6	2	6	4	6	1	2	4	5	6	6	6
8	3	4	4	4	5	1	4	2	4	6	5	3	2	5	4	4	4	5
9	5	3	2	5	3	1	2	5	1	4	4	3	2	4	4	2	6	2
10	6	6	1	2	1	3	6	2	2	4	5	3	3	5	1	4	4	5
11	3	5	2	3	5	5	4	6	6	6	6	3	4	6	4	6	5	4
12	2	6	5	2	1	1	6	6	5	4	1	6	4	1	3	3	2	2
13	6	3	5	3	4	1	2	3	3	3	1	6	1	2	2	4	6	4
14	4	2	6	2	4	2	1	3	5	4	1	1	4	4	2	4	6	6
15	1	5	6	1	5	4	4	5	5	1	5	4	5	1	5	4	6	5
16	3	1	1	1	4	3	2	4	3	1	3	1	5	5	1	6	1	5
17	5	2	6	2	6	2	2	6	1	5	2	2	4	1	5	5	3	6
18	2	1	1	6	1	4	2	1	4	3	6	1	3	3	1	1	5	4
19	2	3	2	5	6	4	6	6	6	3	1	1	5	4	4	1	6	1
20	1	4	6	3	2	6	5	4	4	4	4	1	5	4	3	1	5	6
21	4	2	2	6	2	3	2	4	4	2	6	5	6	3	5	3	1	6
22	5	3	5	2	6	4	4	1	2	5	4	1	4	4	1	2	2	5
23	5	2	3	3	2	3	4	4	4	2	6	6	2	4	5	5	2	3
24	5	4	6	3	2	6	6	5	5	3	5	2	5	5	2	5	2	3
25	2	3	2	6	1	6	2	1	3	3	2	3	1	1	1	5	4	5
26	3	2	5	3	2	3	4	4	2	2	2	4	4	4	4	5	4	3
27	2	6	3	4	4	5	2	3	2	5	1	4	6	2	1	1	5	1
28	5	1	6	3	2	2	5	6	6	6	3	1	2	4	6	1	1	5
29	6	4	3	1	4	2	3	2	6	5	5	2	5	4	3	3	3	6
30	5	5	1	6	4	1	5	6	5	4	4	4	3	3	1	2	5	4

References

Agnetis, A., Murgia, G., & Sbrilli, S. (2014). A job shop scheduling problem with human operators in handicraft production. *International Journal of Production Research, 52*(13), 3820–3831.

Ağralı, S., Taşkın, Z. C., & Ünal, A. T. (2017). Employee scheduling in service industries with flexible employee availability and demand. *Omega, 66*, 159–169.

Ahmadi-Javid, A., & Hooshangi-Tabrizi, P. (2017). Integrating employee timetabling with scheduling of machines and transporters in a job-shop environment: A mathematical formulation and an anarchic society optimization algorithm. *Computers & Operations Research, 84*, 73–91.

Akpinar, S., Elmi, A., & Bektaş, T. (2017). Combinatorial benders cuts for assembly line balancing problems with setups. *European Journal of Operational Research, 259*(2), 527–537.

Allahverdi, A. (2016). A survey of scheduling problems with no-wait in process. *European Journal of Operational Research, 255*(3), 665–686.

Artigues, C., Gendreau, M., Rousseau, L.-M., & Vergnaud, A. (2009). Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. *Computers & Operations Research, 36*(8), 2330–2340.

Benavides, A. J., Ritt, M., & Miralles, C. (2014). Flow shop scheduling with heterogeneous workers. *European Journal of Operational Research, 237*(2), 713–720.

Benders, J. F. (1962). Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik, 4*(1), 238–252.

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E., & De Boeck, L. (2013). Personnel scheduling: A literature review. *European Journal of Operational Research, 226*(3), 367–385.

Braekers, K., Hartl, R. F., Parragh, S. N., & Tricoire, F. (2016). A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience. *European Journal of Operational Research, 248*(2), 428–443.

Brucker, P., Qu, R., & Burke, E. (2011). Personnel scheduling: Models and complexity. *European Journal of Operational Research, 210*(3), 467–473.

Canto, S. P. (2008). Application of Benders' decomposition to power plant preventive maintenance scheduling. *European Journal of Operational Research, 184*(2), 759–777.

Chen, J. H., Lee, D. H., & Cao, J. X. (2012). A combinatorial Benders' cuts algorithm for the quayside operation problem at container terminals. *Transportation Research Part E, 48*(1), 266–275.

Codato, G., & Fischetti, M. (2006). Combinatorial Benders' cuts for mixed-integer linear programming. *Operations Research, 54*(4), 756–766.

De Bruecker, P., Van den Bergh, J., Beliën, J., & Demeulemeester, E. (2015). Workforce planning incorporating skills: State of the art. *European Journal of Operational Research, 243*(1), 1–16.

Dolgui, A., Kovalev, S., Kovalyov, M. Y., Malyutin, S., & Soukhal, A. (2018). Optimal workforce assignment to operations of a paced assembly line. *European Journal of Operational Research, 264*(1), 200–211.

Edis, E. B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research, 230*(3), 449–463.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research, 153*(1), 3–27.

Fischetti, M., Martello, S., & Toth, P. (1989). The fixed job schedule problem with working-time constraints. *Operations Research, 37*(3), 395–403.

Freeman, N. K., H. , M. S., & J. , M. (2016). A scenario-based approach for operating theater scheduling under uncertainty. *Manufacturing & Service Operations Management, 18*(2), 245–261.

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W.H. Freeman.

Goel, A. (2009). Vehicle scheduling and routing with drivers' working hours. *Transportation Science, 43*(1), 17–26.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics, 5*, 287–326.

Grigoriev, A., Sviridenko, M., & Uetz, M. (2007). Machine scheduling with resource dependent processing times. *Mathematical Programming, 110*(1), 209–228.

Guyon, O., Lemaire, P., Pinson, E., & Rivreau, D. (2010). Cut generation for an integrated employee timetabling and production scheduling problem. *European Journal of Operational Research, 201*(2), 557–567.

Guyon, O., Lemaire, P., Pinson, E., & Rivreau, D. (2014). Solving an integrated job-shop problem with human resource constraints. *Annals of Operations Research, 213*(1), 147–171.

Hooker, J. N. (2000). *Logic-based methods for optimization: Combining optimization and constraint satisfaction*. John Wiley and Sons, New York.

Hooker, J. N. (2007). Planning and scheduling by logic-based Benders decomposition. *Operations Research, 55*(3), 588–602.

Huq, F., Cutright, K., & Martin, C. (2004). Employee scheduling and makespan minimization in a flow shop with multi-processor work stations: A case study. *Omega, 32*(2), 121–129.

Kellerer, H., & Strusevich, V. A. (2003). Scheduling parallel dedicated machines under a single non-shared resource. *European Journal of Operational Research, 147*(2), 345–364.

Kellerer, H., & Strusevich, V. A. (2004). Scheduling problems for parallel dedicated machines under multiple resource constraints. *Discrete Applied Mathematics, 133*(1–3), 45–68.

Kellerer, H., & Strusevich, V. A. (2008). Scheduling parallel dedicated machines with the speeding-up resource. *Naval Research Logistics, 55*(5), 377–389.

Krempels, K.-H., & Panchenko, A. (2006). An approach for automated surgery scheduling. In *Proceedings of the 6th international conference on the practice and theory of automated timetabling* (pp. 209–233).

Lasserre, J., & Queyranne, M. (1992). Generic scheduling polyhedral and a new mixed-integer formulation for single-machine scheduling. In *Proceedings of the 2nd ipco conference* (pp. 136–149).
Lee, S., McCann, D., & Messenger, J. C. (2007). *Working time around the world: Trends in working hours, laws, and policies in a global comparative perspective*. Routledge.
Lodree Jr. , E. J., Geiger, C. D., & Jiang, X. (2009). Taxonomy for integrating scheduling theory and human factors: Review and research opportunities. *International Journal of Industrial Ergonomics, 39*(1), 39–51.
Lushchakova, I. N., & Strusevich, V. A. (2010). Scheduling incompatible tasks on two machines. *European Journal of Operational Research, 200*(2), 334–346.
Mokotoff, E., Jimeno, J. L., & Gutiérrez, A. I. (2001). List scheduling algorithms to minimize the makespan on identical parallel machines. *Top, 9*(2), 243–269.
Othman, M., Bhuiyan, N., & Gouw, G. J. (2012). Integrating workersdifferences into workforce planning. *Computers & Industrial Engineering, 63*(4), 1096–1106.
Pinedo, M. L. (2016). *Scheduling: Theory, algorithms, and systems*. Springer.
Rahmaniani, R., Crainic, T. G., Gendreau, M., & Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research, 259*(3), 801–817.
Rodrigues, M. M., de Souza, C. C., & Moura, A. V. (2006). Vehicle and crew scheduling for urban bus lines. *European Journal of Operational Research, 170*(3), 844–862.
Saddoune, M., Desaulniers, G., Elhallaoui, I., & Soumis, F. (2011). Integrated airline crew scheduling: A bi-dynamic constraint aggregation method using neighborhoods. *European Journal of Operational Research, 212*(3), 445–454.
Taşkın, Z. C., & Cevik, M. (2013). Combinatorial Benders cuts for decomposing IMRT fluence maps using rectangular apertures. *Computers & Operations Research, 40*(9), 2178–2186.
Valls, V., Pérez, Á., & Quintanilla, S. (2009). Skilled workforce scheduling in service centres. *European Journal of Operational Research, 193*(3), 791–804.
Verstichel, J., Kinable, J., De Causmaecker, P., & Vanden Berghe, G. (2015). A combinatorial Benders' decomposition for the lock scheduling problem. *Computers & Operations Research, 54*, 117–128.
Waersted, M., & Westgaard, R. H. (1991). Working hours as a risk factor in the development of musculoskeletal complaints. *Ergonomics, 34*(3), 265–276.