



A new approximation algorithm for unrelated parallel machine scheduling with release dates

Zhi Pei¹ · Mingzhong Wan^{1,2} · Ziteng Wang²

© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract

In the current study, an unrelated parallel machine scheduling problem with release dates is considered, which is to obtain a job assignment with minimal sum of weighted completion times. Although this problem is NP-hard in the strong sense, which renders the optimality seeking a formidable task within polynomial time, a 4-approximation algorithm based on the constant worst-case bound is devised and proved in comparison with the existing 16/3-approximation (Hall et al. in *Math Oper Res* 22(3):513–544, 1997). In the newly proposed algorithm, the original scheduling problem is divided into several sub-problems based on release dates. For each sub-problem, a convex quadratic integer programming model is constructed in accordance with the specific problem structure. Then a semi-definite programming approach is implemented to produce a lower bound via the semi-definite relaxation of each sub-problem. Furthermore, by considering the binary constraint, a branch and bound based method and a local search strategy are applied separately to locate the optimal solution of each sub-problem. Then the solution of the original scheduling problem is derived by integrating the outcomes of the sub-problems via the proposed approximation algorithm. In the numerical analysis, it is discovered that the proposed methods could always obtain a scheduling result within 1% of the optimal solution. And an asymptotic trend could be observed where the quality of solutions improves even further as the scale of the scheduling problem increases.

Keywords Unrelated parallel machine scheduling · Release dates · Semi-definite programming · Approximation algorithm · Branch and bound

1 Introduction

Parallel machine scheduling, as a challenging but fruitful field, exerts a great influence on improving the productivity of manufacturing systems. In the present paper, a special scenario, unrelated parallel machine scheduling with release dates, stemming from the general parallel

✉ Zhi Pei
peizhi82@163.com

¹ Department of Industrial Engineering, Zhejiang University of Technology, Hangzhou, China

² Department of Industrial and Systems Engineering, Northern Illinois University, DeKalb, USA

machine scheduling setting is investigated. The underlined scheduling problem considers a circumstance where n jobs are assigned to m unrelated parallel machines. Specifically, each job retains its unique release date, priority weight, and unrelated processing time on each machine. Intuitively, each job is only allowed to start processing after its release date. The aim of this research is to find a high quality assignment which yields a minimized sum of weighted job completion times(TWCT).

In literature, extensive studies have been conducted in the area of parallel machine scheduling without release dates. Chen and Powell (1999) considered a column generation approach for parallel machine scheduling problems. Similarly, van Den Akker et al. (1999) reported a column generation based method for $P||\sum w_j C_j$ problem. Lancia (2000) investigated a scheduling problem under unrelated parallel machines setting with a B&B algorithm to obtain the minimal makespan. Schulz and Skutella (2002) provided a $(2 + \epsilon)$ -approximation algorithm and a $(3/2 + \epsilon)$ -approximation algorithm for the $Rm||\sum w_j C_j$ problem. Another approximation algorithm for $Rm||\sum w_j C_j$ problem is obtained by Unlu and Mason (2010). Since the unrelated parallel machine scheduling is of NP-hard nature, Mokotoff and Jimeno (2002) designed heuristics for unrelated parallel machine scheduling problem based on partial enumeration method. Rodriguez et al. (2012) applied the GRASP method for $Rm||\sum w_j C_j$ problem, where high-quality solutions are obtained within reasonable computational time. Recently, Chen (2015) introduced three heuristics for $Rm||\sum w_j C_j$ problem. For these heuristics, two are based on record-to-record travel, and the other is based on random descent search. Later, Djurasevic et al. (2016) provided an automatized synthesis of heuristics for the unrelated parallel machines base on genetic programming.

Within the domain of unrelated parallel machine scheduling, more extensions are made by Skutella et al. (2016), Mir and Rezaeian (2016) and Cheng and Huang (2017). In particular, several studies focused on the unrelated parallel machine scheduling problem with heterogeneous constraints. Dietrich and Escudero (1993) studied the unrelated parallel machine scheduling problem with setup time. Arnaout (2019) also considered the unrelated parallel machines with sequence-dependent setup times, and the objective of this study is to minimize the makespan. The unrelated parallel machine scheduling problem with time-window was considered by Lee and Sherali (1994). After that, Daniels et al. (1997) proposed a heuristic for the parallel machine scheduling problem with flexible-resources. Recently, Siepak and Józefczyk (2014) studied the unrelated parallel machine scheduling problem with interval processing times to minimize maximal total regret.

To the best of our knowledge, the first exact approach to deal with scheduling problem $Rm||\sum w_j C_j$ is proposed by Azizoglu and Kirca (1999b), and the same authors Azizoglu and Kirca (1999a) also designed two B&B methods to handle the non-identical parallel machine scheduling. Later, Skutella (2001) provided a convex quadratic integer programming based method for the same problem setting, where a non-convex quadratic integer programming model is transformed to the convex formulation by changing the positions of certain independent variables and their respective coefficients. Motivated by Skutella's method, in the present paper, the quadratic programming methodology will be applied to handle $Rm|r_j|\sum w_j C_j$, which is divided into several sub-problems without release dates. Meanwhile, van Den Akker et al. (1999) implemented the well-known Smith (1956) weighted shortest processing time(WSPT) rule to each of the machines separately. Bülbül and Şen (2017) extended this formulation to Rm-TWCT setting, and they managed to prove that in the optimal solution the jobs assigned to each machine should be sequenced in WSPT order, which indeed acts as a catalyst that inspires us to employ the divide-and-conquer framework.

Recently, the approximation algorithm design for scheduling problems with release dates are considered in the literature. Specifically, several latest research investigated on the single

machine scheduling with release dates. For instance, Skutella (2016) proposed a 2.542-approximation for precedence constrained single machine scheduling with release dates and an objective of total weighted completion time. Whereafter, Sitters and Yang (2018) considered a $(2 + \epsilon)$ -approximation for precedence constrained single machine scheduling with release dates and an objective of total weighted completion time.

For the more general case, the parallel machine scheduling with release dates are considered. Zhang et al. (2016) proposed a 2-approximation algorithm for two unrelated machine scheduling problems with machine dependent release dates to minimize the makespan. Yalaoui and Chu (2006) provided a B&B method for $Pm|r_j| \sum w_j C_j$ problem. Another B&B algorithm for $P|r_j| \sum w_j C_j$ problem is introduced in Nessah et al. (2008). Jäger (2018) proposed an approximate algorithm for identical parallel machine scheduling with precedence constraints and release dates.

In terms of computational complexity, Lenstra et al. (1977) proved that the single machine scheduling problem with release dates, which is a special case of $Rm|r_j| \sum w_j C_j$ problem, is NP-hard. Thus the scheduling problem $Rm|r_j| \sum w_j C_j$ is also NP-hard in nature, meaning the optimal schedule cannot be derived in polynomial time. Moreover, Hoogeveen et al. (1998) proved that $Rm|r_j| \sum w_j C_j$ is MaxSNP-hardness, which means that $Rm|r_j| \sum w_j C_j$ problem does not have polynomial time approximation algorithm. Due to the complexity of the unrelated parallel machine scheduling with release dates, the first approximation algorithm for $Rm|r_j| \sum w_j C_j$ appeared in Phillips et al. (1997), and they demonstrated an $(16+\epsilon)$ algorithm with performance guarantee. In the same year, Hall et al. (1997) also considered the problem $Rm|r_j| \sum w_j C_j$, and improved the constant worst-case bound to $16/3$ based on the LP-relaxation. To the best of our knowledge, it is believed that the approximation ratio of $16/3$ is the best known bound for the scheduling problem under constant worst-case setting, which is recently reassured by Skutella et al. (2016). Although after the year of 1997, there were several studies focusing on efficiently solving this scheduling problem, only heuristic based algorithms were devised with no approximation rate guaranteed. For instance, Vredeveld and Hurkens (2002) conducted an empirical comparison of the polynomial-time approximation algorithms and local search heuristics for the scheduling problem $Rm|r_j| \sum w_j C_j$. Later, Tang and Zhang (2011) proposed a Lagrangian relaxation based algorithm for the same problem setting. A set of dispatching rules for $Rm|r_j| \sum w_j C_j$ appeared in Lin and Lin (2013). More recently, Gao et al. (2019) considered the two-agent scheduling on a parallel-batch machine to minimize makespan, where jobs have release dates. However, it is necessary to mention that in Skutella et al. (2016), the authors pointed out that the best-known approximation algorithm is provided by Skutella (2001) and Schulz and Skutella (2002) instead of Hall et al. (1997). Although the approximation rates are significantly improved over it is in Hall et al. (1997), it is worth noticing that the bound is the expected guarantee obtained via random rounding, which is equivalent to state that the underlined algorithms have an expected performance guarantee $2 + \epsilon$ for the expected objective function values, but not for the constant worst-case scenario. Interested readers could always refer to Li and Yang (2009) and Rodriguez et al. (2013) for their excellent literature reviews.

As for semi-definite programming (SDP) method, Lovász and Schrijver (1991) and Balas et al. (1994) provided a theoretic framework and a practical method to construct higher-dimensional polyhedra (or, in some cases, convex sets), in which the projection approximates the convex hull of 0–1 valued solutions of a system of linear inequalities. Sherali and Adams (1994) provided an SDP relaxation model for dealing with mixed integer 0–1 programming problem. For more extensive literature review, the readers can refer to Rendl (2016) for the latest development of SDP relaxation appeared in the assignment and ordering problems.

In this research, we consider two new approximation algorithms, namely the divided semi-definite programming (D-SDP) and the fast divided semi-definite programming (FD-SDP) algorithm, for $Rm|r_j| \sum w_j C_j$ problem. In both algorithms, the $Rm|r_j| \sum w_j C_j$ problem is divided into several $Rm|| \sum w_j C_j$ sub-problems according to their respective release dates. Each sub-problem, relaxed through semi-definite programming (SDP), can then be solved by tailored B&B or LS method. In this way, a series of high quality exact solutions to those sub-problems could be obtained. Then the schedule of the original problem could be constituted with those sub-problem solutions.

The remainder of this paper is organized as follows. In Sect. 2, we consider a mixed integer programming (MIP) model of $Rm|r_j| \sum w_j C_j$ problem. The descriptions of our approximation algorithms are stated in Sect. 3, and the performance of these two algorithms are also reported in this section. In Sect. 4, the computational results are presented in comparison with other published approaches. Sect. 5 concludes the paper.

2 Problem formulation

In this section, a detailed formulation for unrelated parallel machine scheduling problem with release dates is provided. Besides, an MIP model in accordance with the scheduling formulation will be proposed. Despite only a limited number of independent variables are contained in the MIP model, due to the curse of dimensionality, the performance of commercial MIP solvers to address this model at large instances is still rather unsatisfactory. For instance, it takes too much CPU time and memory of computer, if large instances of this MIP model are attacked by CPLEX, which will eventually render the computational process hard to implement under practical settings. For that very reason, in this paper, we will use the outcomes of the commercial solver to compare with the solutions of our two efficient approximation algorithms, which will be illustrated in detail in Sect. 3.

In the formulation of unrelated parallel scheduling problem with release dates, there is a job set, $N = \{1, 2, \dots, n\}$, to be processed on a group of unrelated parallel machines, $M = \{1, 2, \dots, m\}$. For each job $j \in N$ manufactured on machine $i \in M$, its processing time is represented as p_{ij} . To illustrate the priority of the jobs to be handled, each job has its respective weight w_j , $j \in N$ and release date r_j , $j \in N$. The overall objective of this scheduling problem is to minimize the total weighted completion times.

To facilitate the problem formulation, an MIP model for $Rm|r_j| \sum w_j C_j$ problem will be constructed based on the model proposed by Lin and Lin (2013). It is assumed that all jobs have to be completed by time t_{max} , where $t_{max} \geq \sum_{j \in N} r_j + \sum_{i \in M} \sum_{j \in N} p_{ij}$. Then a set of discrete time indexes is denoted as $T = \{1, 2, \dots, t_{max}\}$. In the formulation, C_j stands for the completion time of job j . The binary decision variable x_{ijt} is equal to 1 if job j starts processing on machine i at time t , and 0 otherwise, where $t \in T$. The detailed MIP model is as below,

$$MIP : \min \sum_{j \in N} w_j C_j \quad (1)$$

$$s.t. \quad \sum_{i=1}^m \sum_{t=r_j}^{t_{max}-p_{ij}+1} x_{ijt} = 1, \quad j \in N \quad (2)$$

$$\sum_{j=1}^n \sum_{t=\max\{r_j, t-p_{ij}+1\}}^{\max\{r_j, t\}} x_{ijs} \leq 1, \quad i \in M, t \in T \quad (3)$$

$$\sum_{i=1}^m \sum_{t=0}^{r_j} x_{ijt} = 0, \quad j \in N \quad (4)$$

$$C_j = \sum_{i=1}^m \sum_{t=0}^{t_{\max} - p_{ij} + 1} x_{ijt}(t - 1 + p_{ij}), \quad j \in N \quad (5)$$

$$x_{ijt} \in \{0, 1\}, \quad i \in M, t \in T, j \in N \quad (6)$$

In the above *MIP* model, the number of independent variables x_{ijt} increases with the value of $t_{\max}$. To simplify the computation process, the number of independent variables is reduced by shrinking the span of the planning horizon T , where $t_{\max} \in T$ is an upper bound of the makespan. Function (1) minimizes the total weighted completion time. Constraint (2) ensures that each job can be processed after its release date on only one machine. Constraint (3) guarantees that each job can be processed exactly once. In Constraint (4), jobs can not be processed before their release dates. Constraint (5) defines the value of completion time. Constraint (6) denotes the boolean variables $x_{ijt} = 0$ if and only if job j starts processing on machine i at time t .

Lemma 1 Azizoglu and Kirca (1999b) In $Rm|| \sum w_j C_j$, for each job $j \in N$ on machine i , we have

$$C_{ij} \leq \frac{1}{m} \left(\sum_j \max_i p_{ij} + \sum_{i'} \max_j p_{i'j} \right). \quad (7)$$

where $i \neq i', i, i' \in M$, C_{ij} is the completion time of job j on machine i .

Based on Lemma 1, further extension could be achieved for $Rm|r_j| \sum w_j C_j$ problem to locate a better value of $t_{\max}$. Let $C_{\max,i} = \max_{j \in N} C_{ij}$, which is the makespan of machine i . This tighter bound of $C_{\max,i}$ could be illustrated through the following lemma.

Lemma 2 In $Rm|r_j| \sum w_j C_j$, for each job $j \in N$ on machine i , we have

$$C_{\max,i} \leq \frac{1}{m} \left(\sum_j \max_i p_{ij} + \sum_{i'} \max_j p_{i'j} \right) + r_l. \quad (8)$$

where $i \neq i', i, i' \in M$; r_l is the release date of the last job.

Proof Consider the worst case scenario for makespan minimization, it is assumed $r_j = r_l, \forall j \in N$. Then this special form of unrelated parallel machine scheduling problem $Rm|r_j| \sum w_j C_j$ is equivalent to an $Rm|| \sum w_j (C_j + r_l)$ problem. Hence we have the following inequality.

$$C_{\max,i} \leq C_{ij} + r_l.$$

Based on the Inequality (7) of Lemma 1, we have

$$C_{\max,i} \leq C_{ij} + r_l \leq \frac{1}{m} \left(\sum_j \max_{i'} p_{i'j} + \sum_{i' \neq i} \max_j p_{i'j} \right) + r_l.$$

□

Based on Lemma 2, a better upper bound of the $C_{max,i}$ could be obtained, and it can effectively reduce the number of independent variables. As a result, the MIP model mentioned above could be solved more efficiently in terms of CPU time due to the reduction of t_{max} . To better depict the refined planning horizon T , the following corollary is introduced, which is directly derived from Lemma 2.

Proposition 1 In $Rm|r_j| \sum w_j C_j$, the improved upper bound of completion time t_{max} is

$$t_{max} = \max_i C_{max,i} = \max_i \left\{ \frac{1}{m} \left(\sum_j \max_i p_{ij} + \sum_{i'} \max_j p_{i'j} \right) + r_l \right\}. \quad (9)$$

where $i \neq i'$, $i, i' \in M$, $j \in N$.

Therefore, in comparison with the original t_{max} in the MIP formulation, where $t_{max} \geq \sum_{j \in N} r_j + \sum_{i \in M} \sum_{j \in N} p_{ij}$, the number of independent variables could be reduced substantially via Proposition 1. As a result, in small-scale instance, the aforementioned MIP model could be solved by commercial solver much faster. However, due to the intrinsic complexity of the MIP model, the memory requirements and the CPU time would increase rapidly with the number of jobs and machines even though the planning horizon T has been improved. To tackle that issue, new approximation algorithms are needed to solve the MIP model with medium and large instances.

3 The approximation algorithm

In literature, Lenstra et al. (1977) proved that the single machine scheduling problem with release dates, $1|r_j| \sum w_j C_j$, is strongly NP-hard. Therefore, $Rm|r_j| \sum w_j C_j$ problem, as a more generalized instance of $1|r_j| \sum w_j C_j$ is also NP-hard in the strong sense. Moreover, Hoogeveen et al. (1998) proved that $Rm|r_j| \sum w_j C_j$ problem is MaxSNP-hardness, which means that this unrelated parallel machine scheduling problem cannot be solved in polynomial time.

In the present paper, high quality approximations of the optimal solution to $Rm|r_j| \sum w_j C_j$ problem are provided by two new algorithms, namely D-SDP and FD-SDP. In both of these two algorithms, we first group the jobs according to the non-decreasing order of their release dates. Then the original problem is divided into several sub-problems based on the ordered release dates, and these sub-problems could be fully addressed via semi-definite programming based algorithms. After that, the approximation of the optimal solution to the original problem could be obtained by combining the solutions of the sub-problems. More importantly, D-SDP algorithm is proved to be a 4-approximation algorithm under the constant worst case scenario. In the following, we will introduce the divide and combine rules and the SDP based algorithm for the decomposed sub-problems in detail.

3.1 Divide and combine

In this sub-section, the divide and combine framework will be employed to decompose the original $Rm|r_j| \sum w_j C_j$ problem into a series sub-problems. Without loss of generality, a set of jobs are indexed with regard to their release dates, i.e. $r_1 \leq r_2 \leq \dots \leq r_n$, where r_j is the release date of job j . In order to describe the sub-problem, we firstly define the stage date of jobs.

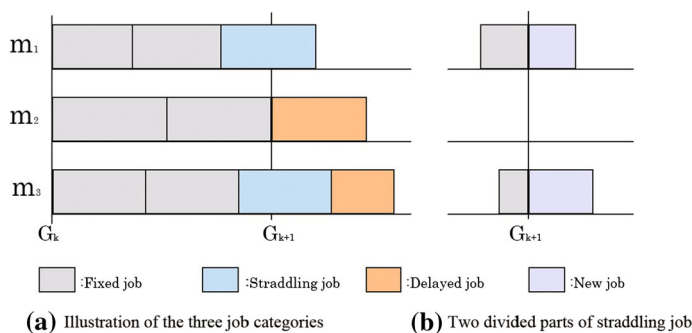


Fig. 1 Illustration of the three job categories

Definition 1 (Stage date) In $Rm|r_j|\sum w_j C_j$, each job $j \in N$ has a stage date, only after which the job can be processed. The initial stage date of job j is generated as below,

$$G_k = r_j, \forall k \in K.$$

where $i \in M$, $p_{max} = \max_{i,j} p_{ij}$, $K = \{1, 2, \dots, k_{max}\}$, k_{max} is the number of unequal release dates.

As it is mentioned above, in the newly devised algorithms, the sub-problems are defined by their respective stage dates in the non-decreasing order. Without loss of generality, a set of jobs are indexed with regard to their stage dates, i.e. $G_1 \leq G_2 \leq \dots \leq G_k \leq G_{max}$. Then the sub-problem is defined as follows.

Definition 2 (Sub-problem) In $Rm|r_j|\sum w_j C_j$, the k_{th} sub-problem is defined as the $Rm|r_j|\sum w_j C_j$ problem for $j \in N_k$, $\forall k \in K$, where N_k is the job set with identical stage date G_k .

For each job $j \in N$, the stage date G_k is employed to construct job groups, in each of which the jobs share the same stage date. For the k_{th} such job group N_k , scheduling jobs in it is defined as the k_{th} sub-problem with the stage date of the sub-problem G_k . More details about how to handle the sub-problem will be introduced in Subsection 3.2.

In the sub-problem, we cannot guarantee that all the jobs in one sub-problem would finish their processing before the next stage date. Consequently, the next sub-problem would not be a simple $Rm|r_j = G_{k+1}|\sum w_j C_j$ problem, if $C_j > G_{k+1}$, $\exists$ job $j \in N_k$. Before addressing that issue, three job type sets in the sub-problems have been defined based on the completion time versus the next stage date are listed below.

Definition 3 (Job sets type) In the k_{th} sub-problem, we have

- Fixed job set: $FJS_k = \{j | C_j \leq G_{k+1}\}$;
 - Straddling job set: $SJS_k = \{j | C_j - p_{ij} \leq G_{k+1}, C_j \geq G_{k+1}\}$;
 - Delayed job set: $DJS_k = \{j | C_j \geq G_{k+1}\}$.
- where $i \in M$, $k \in K$, $j \in N_k$.

An illustration of these three kinds of jobs is shown in Fig. 1a. For the fixed and straddling jobs, their scheduling plans remain unchanged, which means that these two types of jobs are still considered in the k_{th} sub-problem. While for delayed jobs, it is straightforward to presume that their stage date has now shifted to G_{k+1} . Meanwhile, the engaged stages cannot be simply modeled as an $Rm|G_k|\sum w_j C_j$ or an $Rm|G_{k+1}|\sum w_j C_j$ problem, since

the straddling jobs are spanning across at least two stages, the residue processing left from the last stage will keep occupying the same machine, which violates the assumption that all the machines are available when the stage starts.

To simplify the situation incurred by delayed jobs, it is assumed that the delayed jobs are divided into several segments, each of which is fully contained in one stage. Then every delayed job segment could be considered as an independent new job, and what we have to do is to guarantee that the delayed job segments are manufactured in one machine continuously without interruption. An illustration of the new jobs yielded by dividing straddling jobs is shown in Fig. 1b.

Lemma 3 For each $j \in SJS_k$ processed on machine i , job j is divided into two segments, $j^a \in SJS1_k$ and $j^b \in SFS2_k$ with new processing times and weights:

$$\begin{aligned} \text{Segment1 : } j^a & \begin{cases} p_{ij^a} = G_{k+1} - (C_j - p_{ij}); \\ w_{j^a} = w_j. \end{cases} \\ \text{Segment2 : } j^b & \begin{cases} p_{i'j^b} = \begin{cases} C_j - G_{k+1}, & \text{if } i' = i; \\ P_j, & \text{otherwise.} \end{cases} \\ w_{j^b} = W_j. \end{cases} \end{aligned}$$

where $i, i' \in M$, P_j and W_j are two sufficiently large real numbers.

Moreover, we will show that under what conditions the straddling jobs and their respective new jobs will be processed in a continuous manner. In other words, these straddling jobs in the original scheduling problem would not be interrupted by the stage date given certain values of P_j and W_j . Bülbül and Şen (2017) extended the WSPT rule to the unrelated parallel machine scheduling problem, and they have proved that the jobs assigned to each machine are sequenced in the WSPT order in the optimal solution of $Rm||\sum w_j C_j$ problem. Based on this extension, the following theorem is introduced.

Theorem 1 (Nonpreemption) For each job $j \in SJS_k$, it obeys the rule of nonpreemption, given the following conditions

$$\begin{aligned} \frac{W_j}{P_j} & \geq \max_{j'} \frac{w_{j'}}{p_{j'}}, \\ P_j & \geq C_j - G_{k+1}. \end{aligned}$$

where $j' \in N_{k+1}$.

Proof For each job $j \in SJS_k$ which is processed on machine i and has the stage date G_k . In the k_{th} sub-problem, the job j^b is generated from job j . Its processing time and weight is generated by Lemma 3.

For job j^b , we have the conditions $\frac{W_j}{P_j} \geq \max_{j'} \frac{w_{j'}}{p_{j'}}$ for $j' \in N_{k+1}$. Thus according to the extended WSPT rule, job j^b in the $(k+1)_{th}$ sub-problem generated by its respective straddling job which starts processing in the k_{th} sub-problem would be scheduled to be the first to process, which means that job j^b could select any machine $i \in M$. Moreover, due to the processing time of the job satisfies $p_{i'j^b} = P_j \geq C_j - G_{k+1}, \forall i' \in M \setminus \{i\}$, thus $p_{ij^b} = C_j - G_{k+1} \leq p_{i'j^b}$, which means that the straddling job must be processed on the same machine to guarantee the optimal scheduling. $\square$

In the above paragraphs, the divide method has been introduced in detail. Then we need to consider how to combine the solutions of each sub-problem to obtain the scheduling

plan of MIP . Based on Theorem 1, it is apparent that the k_{th} sub-problem can be solved as an $Rm|r_j = G_k|\sum w_j C_j$ problem with some jobs generated from straddling jobs. In each sub-problem, the starting time of each job before the next stage date stays unchanged. Equivalently, we will not change the start time of fixed jobs and straddling jobs. Therefore, the solution of the original problem MIP can be derived by combining the solutions of all the sub-problems. Furthermore, the number of jobs in each sub-problem would increase if the release dates are densely distributed, because in that case the number of straddling jobs would accumulate. Although it should be pointed out that this divide and combine algorithm may not be able to obtain the optimal solution of the problem MIP directly, in the following chapters, it is demonstrated that this approximation algorithm could indeed provide high quality solutions in computational instances.

3.2 SDP model for sub-problem

To locate high quality solutions of $Rm|r_j|\sum w_j C_j$ problem, in our algorithms, we consider dividing the original problem into several sub-problems. Moreover, each sub-problem can be formulated as an $Rm||\sum w_j C_j$ problem. To handle the sub-problems, we firstly consider the basic $Rm||\sum w_j C_j$ model, which was introduced by Skutella (2001).

$$QIP_k : \min \sum_{j,j' \in N} \sum_{i \in M} x_{ij} w_j \left(p_{ij} + \sum_{j' \prec_i j} x_{ij'} p_{ij'} \right) \quad (10)$$

$$s.t. \sum_{i \in M} x_{ij} = 1, \quad j \in N_k \quad (11)$$

$$x_{ij} \in \{0, 1\}, \quad i \in M, j \in N_k \quad (12)$$

where x_{ij} equals 1 if job j is processed on machine i , and $x_{ij} = 0$ otherwise. The precedence $j' \prec_i j$ holds if $w_j/p_{ij} \leq w_{j'}/p_{ij'}$. Function (10) minimizes the total weighted completion time for all jobs in the k_{th} sub-problem. Constraint (11) ensures that each job can be processed exactly once. Constraint (12) denotes the boolean decision variables.

Since the formulation QIP_k is well defined, and a 2-approximation scheme is available to handle the QIP_k model, if we could prove that the sub-problem QIP_k is equivalent to $Rm||\sum w_j C_j$ problem, then the solution procedure would be greatly simplified.

In our research, each sub-problem can be illustrated as below, which has the similar formulation as QIP_k model:

$$Q\tilde{I}P_k : \min \sum_{j,j' \in N} \sum_{i \in M} x_{ij} w_j \left(p_{ij} + \sum_{j' \prec_i j} x_{ij'} p_{ij'} + G_k \right) \quad (13)$$

$$s.t. \sum_{i \in M} x_{ij} = 1, \quad j \in N_k \quad (14)$$

$$x_{ij} \in \{0, 1\}. \quad i \in M, j \in N_k \quad (15)$$

where $x_{ij} = 1$ if job j is processed on machine i , and $x_{ij} = 0$ otherwise; According to the WSPT rule, the precedence relation $j' \prec_i j$ stands if $w_j/p_{ij} \leq w_{j'}/p_{ij'}$. Function (13) minimizes the sum of weighted completion times plus stage dates for all jobs in the k_{th} sub-problem.

Based on the above method, the original problem can be decomposed into K sub-problems. The stage date of the $(k+1)_{th}$ sub-problem is larger than it is of the k_{th} sub-problem,

$\forall k \in \{1, 2, \dots, k_{max} - 1\}$. Moreover, the optimal solution of each sub-problem QIP_k should be a local optimum for the job set N_k in the original problem. Hence to combine all the sub-problems together would yield a relaxation of the original problem.

As stated in Definition 2, the jobs in the same sub-problem share an equivalent stage date. In other words, all the jobs in the k_{th} sub-problem would be released at G_k , even though the stage date and release time are defined differently. As a result, these sub-problems could be expressed in the form $Rm|r_j = G_k | \sum w_j C_j$, where each job in N_k is released at G_k . In the following proposition, we'll try to show that each sub-problem could be formulated as an $Rm || \sum w_j C_j$ problem.

Proposition 2 *In the k_{th} sub-problem, we have*

$$x_{ij}^* = \tilde{x}_{ij}^*, \forall i \in M, j \in N_k.$$

where x^* is the optimal solution of QIP_k and $\tilde{x}^*$ is the optimal solution of $Q\tilde{I}P_k$.

Proof It is observed that the sub-problem $Q\tilde{I}P_k$ has an objective function quite similar to QIP_k model. To fully exploit this feature, the objective function of $Q\tilde{I}P_k$ model is reformulated as follows,

$$\begin{aligned} & \sum_{j,j' \in N} \sum_{i \in M} x_{ij} w_j \left(p_{ij} + \sum_{j' < i j} x_{ij'} p_{ij'} + G_k \right) \\ &= \sum_{j,j' \in N} \sum_{i \in M} x_{ij} w_j \left(p_{ij} + \sum_{j' < i j} x_{ij'} p_{ij'} \right) + \sum_{j \in N} \sum_{i \in M} x_{ij} w_j G_k \\ &= \sum_{j,j' \in N} \sum_{i \in M} x_{ij} w_j \left(p_{ij} + \sum_{j' < i j} x_{ij'} p_{ij'} \right) + \sum_{j \in N} w_j G_k. \end{aligned}$$

where the value of $\sum_{j \in N} w_j G_k$ is equal to $\sum_{j \in N} \sum_{i \in M} x_{ij} w_j G_k$ because $\sum_{i \in M} x_{ij} = 1$ in Constraint (14). Besides, it is found that the value of $\sum_{j \in N} w_j G_k$ does not depend on variables x_{ij} . Also, since $Q\tilde{I}P_k$ model has exactly the same sets of constraints as QIP_k model, thus the QIP_k model and $Q\tilde{I}P_k$ model share the same solution space. Therefore, given the same set of jobs $j \in N_k$, we have $x_{ij}^* = \tilde{x}_{ij}^*, \forall i \in M$. $\square$

It is proved that each sub-problem can be modeled as an $Rm || \sum w_j C_j$ problem in Proposition 2. Now we focus on the sub-problems and try to find the optimal solution of each sub-problem. As it is recalled, a quadratic integer programming model for the sub-problem is demonstrated in Sect. 3.2. In Skutella (2001), the author claimed that a semi-definite relaxation based method could be used to tackle $Rm || \sum w_j C_j$ problem, which is the equivalent counterpart of the k_{th} sub-problem $Q\tilde{I}P_k$. To handle that QIP_k model, Skutella also proposed a convex quadratic method, which is formulated as follows,

$$CQIP_k : \quad \min \quad \frac{1}{2} c^T x + \frac{1}{2} x^T D x \quad (16)$$

$$s.t. \quad \sum_{i \in M} x_{ij} = 1, \quad j \in N_k \quad (17)$$

$$x_{ij} \in \{0, 1\}, \quad i \in M, j \in N_k \quad (18)$$

where $x_{ij} = 1$ if job j is processed on machine i , and $x_{ij} = 0$ otherwise. $x \in R^{mn}$ denotes the vector of ordered variables x_{ij} with respect to the natural order of the machines. For each machine, the jobs are sequenced according to the WSPT rule. Note that for each machine i , the jobs are ordered according to the relation $<_i$. The vector $c \in R^{mn}$ is derived by $c_{ij} = w_j p_{ij}$, and $D = (d_{(ij)(i'j')})$ is a symmetric $mn \times mn$ matrix denoted as below,

$$d_{(ij)(i'j')} = \begin{cases} w_j p_{ij} & \text{if } i = i' \text{ and } j = j' \\ w_j p_{ij'} & \text{if } i = i' \text{ and } j' <_i j \\ w_{j'} p_{ij} & \text{if } i = i' \text{ and } j <_i j' \\ 0 & \text{if } i \neq i' \end{cases}$$

It is worth noticing that matrix D is a semi-definite matrix. Due to the quadratic objective function and 0–1 boolean variables included in the above modeling, an SDP relaxation formulation is developed by relaxing the 0–1 constraints. The advantages to apply the SDP relaxation based method are twofold. On one hand, semi-definite optimization problems are in principle solvable in polynomial time by CSDP method proposed by Borchers (1999); On the other hand, the modeling power of SDP relaxation based methods allows us to deal with optimization models with a quadratic objective function. As a result, the following SDQP model is proposed.

$$SDQP_k : \min \frac{1}{2} c^T x + \frac{1}{2} D \cdot X \quad (19)$$

$$s.t. \quad Ax = e \quad (20)$$

$$X - xx^T \succeq 0 \quad (21)$$

$$\text{diag}(X) = x \quad (22)$$

$$X \in S^{mn \times mn} \quad x \in R^{mn} \quad (23)$$

where $A \in R^{mn \times mn}$ is an all one matrix, $e \in R^{mn}$ is an all one vector, and x is a vector of x_{ij} . $S^{mn \times mn}$ represents the set of $mn \times mn$ real symmetric matrix, and R^{mn} represents the set of mn real vectors.

In the above $CQIP_k$ model, the 0–1 constraints are replaced by Constraint (21) in $SDQP_k$. Moreover, the boolean variables $x_{ij} \in \{0, 1\}$ in $CQIP_k$ model defined above could also be replaced by a series of valid inequalities, in the form of $x_{ij}^2 = x_{ij}$. Then this semi-definite relaxation could be further tightened by adding Constraint (22), $\text{diag}(X) = x$, which stands for the cutting plane of the feasible domain in $SDQP_k$ model.

3.3 D-SDP and FD-SDP approximation algorithms

In this section, we will introduce our two approximation algorithms in detail, namely D-SDP and FD-SDP. Albeit the two proposed approximation algorithms share the divide and combine approach, they diverge to separate routes while rounding the non-integer solutions of the SDQP model solved by semi-definite programming methods, which then lead to different performances in terms of solution accuracy and computation time.

The D-SDP Algorithm.

In D-SDP algorithm, the $Rm|r_j| \sum w_j C_j$ problem is divided into several $Rm|| \sum w_j (C_j + G_k)$ sub-problems based on the jobs' stage dates. For the k_{th} sub-problem, it is solved by a tailored B&B method based on the bound generated by solving $SDQP_k$. After the optimal solution of the k_{th} sub-problem is obtained, the jobs in the k_{th} sub-problem is handled by the method proposed in Sect. 3.1. In particular, delayed jobs and a part of straddling

jobs are moved to the $(k + 1)_{th}$ sub-problems to be considered. Finally, the solution of $Rm|r_j|\sum w_j C_j$ problem is obtained by combining the solution of each sub-problem, after all the sub-problems have been solved. The pseudo code of D-SDP algorithm is shown as follows.

Algorithm 1: D-SDP

```

1 Initialization: Arrange the jobs in non-decreasing order with respect to their release
  dates. Find the largest processing time  $p_{max} = \max\{p_{ij}\}$  for all  $i \in M, j \in N$ , where
  set  $P$  is used to record the confirmed scheduling plan;
2 Let  $G_1 \leftarrow r_1$  and  $k \leftarrow 2$ ;
3 for  $j=2$  to  $n$  do
4   if  $G_k = r_j$  then
5     | Set  $G_k \leftarrow r_j$ 
6   else
7     | Set  $k \leftarrow k + 1, G_k \leftarrow r_j$ 
8 for  $k=1$  to  $n$  do
9   | The  $k_{th}$  sub-problem is composed by the jobs with equality  $G_k$ .
10 Let  $k_{max}$  be the number of sub-problems;
11 for  $k = 1$  to  $k_{max}$  do
12   | Solve the  $SDQP_k$  model of  $k_{th}$  sub-problem, and find the exact solution by B&B
    algorithm;
13   | Include the scheduling plan of fixed jobs and straddling jobs into the overall result;
14   | If there is a straddling job  $j$  processed on machine  $i$ , then set a new job with stage
    date  $G_{k+1}$ , which has the processing time  $p_{lj} = C_j - G_{k+1}$ ; Otherwise,  $\forall l = i$ ,
     $p_{lj} = P_j$ , and the weight is  $w_j = W_j$ ;
15   | If there exists delayed jobs, let the stage dates of these jobs be  $G_{k+1}$ ;
16 Obtain the scheduling result by combining the scheduling plans of jobs in set  $P$ ;
17 Calculate the sum of weighted completion times of the original jobs in set  $P$ .
```

The pseudo code for the tailored B&B method is in Step 12 of D-SDP algorithm. In the B&B method, the cutting plane $x_{ij} = 1$ is added to $SDQP_k$ model to calculate the bound of the target branch.

In D-SDP algorithm, an objective value which is closer to the exact optimal solution could be obtained for both medium scale and large scale instances. The CPU time of this algorithm will only increase with the number of jobs and machines but not the value of t_{max} . However, the B&B method in D-SDP algorithm will incur higher cost, since the semi-definite programming based approach needs to be revisited multiple times, which is quite time consuming compared to one time visit. To resolve that issue, a local search based algorithm will be introduced, which is significantly faster, and the quality of solutions is acceptable even though it might not obtain the optimal solution for each sub-problem.

Algorithm 2: Branch and Bound**Input:** The relaxed scheduling plan P and its objective value**Output:** The exact scheduling plan $\tilde{P}$

```

1 for  $j=1$  to number of jobs in the  $k_{th}$  sub-problem do
2   If  $x_{ij}$  is the maximum for all  $i \in M$ , then set  $x_{ij} \leftarrow 1$ , otherwise set  $x_{ij} \leftarrow 0$ ;
3   Add the scheduling plan to  $\tilde{P}$ ;
4 Let the initial bound  $B$  equal to the objective value of the scheduling plan  $\tilde{P}$ ;
5 Let the nodes which have not been visited or fathomed be active nodes and set
    $V_{min} \leftarrow \text{inf}$ ;
6 for  $\tilde{j}=1$  to number of jobs in  $k_{th}$  sub-problem do
7   for  $\tilde{i}=1$  to  $m$  do
8     while current node is active node do
9       Add the constraint  $x_{\tilde{i}\tilde{j}} = 1$  to the  $SDQP_k$  model of its parental node;
10      Solve the  $SDQP_k$  model of the current node and let  $LB_{\tilde{i}\tilde{j}}$  be the lower bound
        of the current node;
11      if  $LB_{\tilde{i}\tilde{j}} \leq B$  then
12        Set  $B \leftarrow LB_{\tilde{i}\tilde{j}}$ ;
13      else
14        if  $\tilde{i} \neq m$  then
15          Fathom the branch of the current node;
16        else
17          If  $LB_{\tilde{i}\tilde{j}} \leq V_{min}$ , let  $V_{min} \leftarrow LB_{\tilde{i}\tilde{j}}$ ;
18          Let scheduling plan to  $\tilde{P} \leftarrow$  scheduling plan of the current node;
19 Let  $\tilde{P}$  be the best exact scheduling plan and  $V_{min}$  denote its value.

```

The FD-SDP Algorithm.

In D-SDP algorithm, the step of the B&B method would increase the CPU time of D-SDP algorithm dramatically. So it is desirable to replace this B&B algorithm in Step 12 of D-SDP with a faster algorithm, which ideally could guarantee high quality solutions of the sub-problems. To serve that purpose, we replace the B&B algorithm by a tailored local search (LS) method to find the exact solutions of the sub-problems, which is named FD-SDP algorithm.

In the LS method of FD-SDP algorithm, the relaxed solution of $SDQP_k$ is rounded firstly. That is, let $x_{ij} = 1, \forall j \in N$, where $i = \text{argmax}(x_{ij})$. Then the rounding result is compared by changing the assignment of one job. Consequently, the best rounding solution is known as the result of LS method. The pseudo code of LS method is illustrated as follows.

Algorithm 3: Local Search**Input:** The relaxed solution P **Output:** The exact scheduling plan $\tilde{P}$

- 1 For each job in the relaxed solution P , let $x_{i_{\max}j} \leftarrow 1$ if $x_{i_{\max}j} > x_{ij}$, $i \in M \setminus \{i_{\max}\}$; Otherwise, let $x_{i_{\max}j} \leftarrow 0$. Let V_{int} be the objective function value w.r.t. the exact solution $\tilde{P}$ by rounding the solution P ;
- 2 **for** $\tilde{j}=1$ to the number of jobs in the k_{th} sub-problem **do**
- 3 **for** $\tilde{i}=1$ to M **do**
- 4 Let $x_{\tilde{i}\tilde{j}} \leftarrow 1$ and $x_{i\tilde{j}} \leftarrow 0$ for all $i \in M \setminus \{\tilde{i}\}$. Then let $\tilde{V}$ be the value of the current scheduling plan. **if** $\tilde{V} < V_{int}$ **then**
- 5 Let $V_{int} \leftarrow \tilde{V}$, and save the current scheduling plan as $\tilde{P}$

3.4 Performance analysis

In this section, we will analyze the performance of D-SDP algorithm for each sub-problem, and estimate the approximation quality of the D-SDP algorithm. Meanwhile, since the proposed local search based method is employed to find the exact solution in FD-SDP algorithm, the performance guarantee for the FD-SDP algorithm will not be discussed.

To fully stress the performance evaluation of the approximation algorithm D-SDP, the k_{th} sub-problem of the original scheduling problem has been decomposed into several cases with respect to the existence of straddling jobs. There are in general two different scenarios for the k_{th} sub-problem. Moreover, the second case can be further divided into two conditions by the existence of delayed jobs.

Case 1: In the k_{th} sub-problem, $SJS_k = \emptyset$ and $DJS_k = \emptyset$. This illustration of which is shown in Fig. 2a.

Case 2: In the k_{th} sub-problem, $SJS_k \neq \emptyset$.

Case 2.1: $DJS_k = \emptyset$, which is shown in Fig. 2b.

Case 2.2: $DJS_k \neq \emptyset$, which is demonstrated in Fig. 2c.

In Proposition 2, we have proved that each sub-problem can be solved as an $Rm|| \sum w_j(C_j + G_k)$ problem, which can obtain the optimal solution based on the B&B method of D-SDP

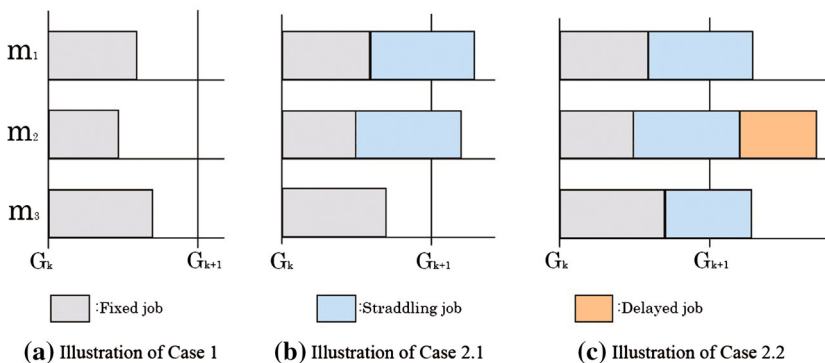
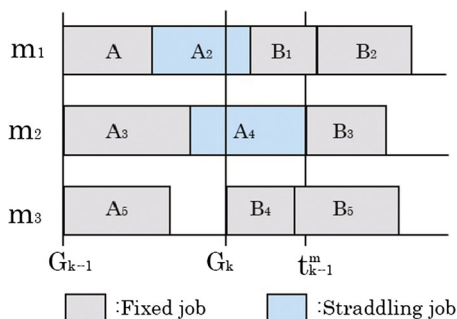
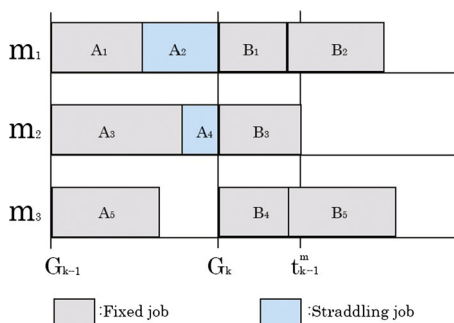


Fig. 2 Illustrations of the three different scenarios in the k_{th} sub-problem

Fig. 3 The sample scheduling plan derived by D-SDP algorithm**Fig. 4** The lower bound job allocation of the k_{th} sub-problem

algorithm. Particularly, for each $j \in N_k$ in Case 1, it will not affect the scheduling outcome of the $(k+1)_{th}$ sub-problem. It means that $\forall k \in K$, if any job $j \in FJS_k$, then D-SDP algorithm could be used to obtain the optimal solution of the original problem. In the following, we will focus on the approximation of D-SDP algorithm via Case 2.1 and Case 2.2.

Several frequently used signs and symbols are denoted as below,

v^o : The global optimum of the original problem.

v_k^o : The local optimum of QIP_k , $\forall j \in N_k \setminus DJS_k \cup SJS_{2k-1}$, which can be obtained via D-SDP algorithm.

v^D : The objective value of the scheduling plan via D-SDP algorithm, and $v^D = \sum_{k \in K} v_k^o$.

In what follows, the relationship between v^o and v^D will be proved to be $v^D \leq \epsilon \cdot v^o$, where ϵ is a specifically constructed ratio. Firstly, we considered the approximation of v_k^o in each sub-problem.

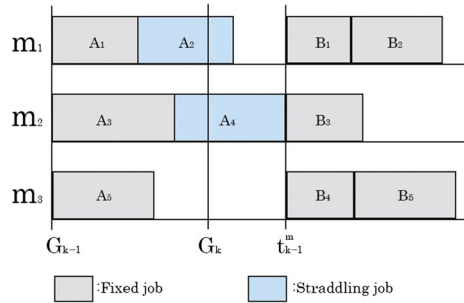
Figure 3 depicts the scheduling plan obtained by D-SDP algorithm for an instance of Case 2.1, where jobs $A_1, \dots, A_5$ have stage dates G_{k-1} , jobs $B_1, \dots, B_5$ have stage dates G_k , and t_{k-1}^m is the makespan of the scheduling plan for the $(k-1)_{th}$ sub-problem. Although the exact optimal solution to the MIP could hardly be obtained due to the curse of dimensionality, we will try to provide an upper bound and a lower bound of the objective value $\sum w_j C_j$ for the k_{th} sub-problem handled by D-SDP algorithm.

To begin with, the proposed lower bound for the local optimal solution of the k_{th} sub-problem is demonstrated in the following lemma, which by definition should be smaller than the optimal solution of $SDQP_k$. This situation is shown in Fig. 4, assuming all the machines are idle after G_k .

Lemma 4 In the k_{th} sub-problem, we have

$$v_k \leq v_k'. \quad (24)$$

Fig. 5 The upper bound job allocation of the k_{th} sub-problem



where v_k denotes the optimal objective value of $Q\tilde{I}P_k \forall j \in N_k$, and $v_{k'}$ denotes the optimal value of $Q\tilde{I}P_k \forall j' \in N'_k$, and N'_k is another job set that $N_k \subseteq N'_k$.

Proof Firstly, assume that $N'_k = N_k \cup \{j'_a\}$. If $v_k > v_{k'} - w_{j'_a} C_{j'_a}$, then v_k can be reduced by using the optimal solution, while ignoring the job j'_a of $Q\tilde{I}P_k$ for $j' \in N'_k$. Apparently, it contradicts the optimum condition of v_k . Hence we have

$$v_k \leq v_{k'} - w_{j'_a} C_{j'_a}.$$

Without loss of generality, assume that $N'_k = N_k \cup \{j'_1, j'_2, \dots, j'_s\}$. Based on the above proof, we have

$$v_k \leq v_{k'} - (w_{j'_1} C_{j'_1} + w_{j'_2} C_{j'_2} + \dots + w_{j'_s} C_{j'_s}) \leq v_{k'}.$$

□

Theorem 2 (Lower bound) In the k_{th} sub-problem, we have

$$v_k^{lb} \leq v_k^o. \quad (25)$$

where v_k^{lb} is the optimal value of $Q\tilde{I}P_k$ by assuming all machines are idle at G_k .

Proof When we ignore the existence of the straddling jobs in the $(k-1)_{th}$ stage, in the best scenario, it could be assumed that all the machines are idle at G_k . Let N_k^{lb} denote the job set that $N_k^{lb} \cup \{j^b | j \in N_{k-1}\}$. Moreover, since Inequality (24) holds in Lemma 4, hence we have $v_k^{lb} \leq v_k^o$. □

Theorem 2 demonstrates how the lower bound of the original problem MIP is constructed. In the following, we will devise an upper bound for the optimal solution of the k_{th} sub-problem to completely restrain the optimal value of the $(k+1)_{th}$ sub-problem. This upper bound condition is shown in Fig. 5.

Theorem 3 (Upper bound) In the k_{th} sub-problem, we have

$$v_k^o \leq v_k^{ub}. \quad (26)$$

where v_k^{ub} is the optimal value of $Q\tilde{I}P_k$ by assuming all the machines are idle at $\max\{G_k, t_{k-1}^m\}$.

Proof The assumption, that all machines are idle after $\max\{G_k, t_{k-1}^m\}$, can be known as that there is a set of jobs N_k^{ub} should be produced in the idle time period $[G_k, \max\{G_k, t_k^m\})$. Then v_k^o is the optimal value of $Q\tilde{I}P_k \forall j \in N_k$, and v_k^{ub} is the optimal value of $Q\tilde{I}P_k \forall j \in \{N_k \cup N_k^{ub}\}$. Moreover, Inequality (4) holds in Lemma 4. Hence we have $v_k^o \leq v_k^{ub}$. □

In Theorem 2 and Theorem 3, both the lower bound and the upper bound for the optimal solution of the k_{th} sub-problem have been proposed. Moreover, according to Proposition 2, it could be concluded that the jobs in the optimal scheduling plan of Theorem 2 and Theorem 3 should have exactly the same sequence, given that the jobs in each stage all start processing at the same time. Then the following lemma is provided to compare the objective values, $\sum w_j C_j$, of these two scheduling.

Lemma 5 *In the k_{th}^a and k_{th}^b sub-problems, which have the same job set N_k , with different stage dates. Their objective values satisfy*

$$v_{ka}^o \leq \frac{G_a}{G_b} v_{kb}^o. \quad (27)$$

where $G_a \leq G_b$ are unequal stage dates, v_{ka}^o is the objective value of $Q\tilde{I}P_k$ with job set N_k and stage date G_a ; v_{kb}^o is the objective value of $Q\tilde{I}P_k$ with job set N_k and stage date G_b .

Proof According to Proposition 2, the same set of jobs in stage G_a should have the identical scheduling plan as they do in stage G_b . It means that

$$\frac{w_j C_j^a}{w_j C_j^b} = \frac{G_a + p_{ij} + \tilde{p}_{ij}}{G_b + p_{ij} + \tilde{p}_{ij}} \leq \frac{G_a}{G_b}, \forall j \in N_k.$$

where C_j^a is the completion time of job j in stage G_a , and C_j^b is the completion time of job j in stage G_b ; $\tilde{p}_{ij}$ is the total processing time of jobs processed before job j on machine i in the same stage. Thus the following inequality holds,

$$\frac{\sum w_j C_j^a}{\sum w_j C_j^b} \leq \frac{G_a}{G_b}.$$

Since $v_{ka}^o = \sum w_j C_j^a$, and $v_{kb}^o = \sum w_j C_j^b$, thus we have $v_{ka}^o \leq \frac{G_a}{G_b} v_{kb}^o$. $\square$

Based on Lemma 5, we can evaluate the relationship between the objective values of the two sub-problems with identical job sets but unequal stage dates. This conclusion can be used in comparing the lower bound in Theorem 2 and the upper bound in Theorem 3. And the comparison result is shown in the following theorem.

Theorem 4 (ϵ_k -approximation) *In the k_{th} sub-problem, v_k^o has the approximation to v_k^{lb} satisfying,*

$$v_k^o \leq \epsilon_k v_k^{lb}. \quad (28)$$

where $k \in K \setminus \{1\}$, $\epsilon_k = \frac{\max\{G_k, t_{k-1}^m\}}{G_k}$, t_{k-1}^m is the makespan of the scheduling plan for the $(k-1)_{th}$ sub-problem.

Proof As it is shown in Theorem 3, the relation $v_k^o \leq v_k^{ub}$ holds. According to Proposition 2, the jobs in the optimal scheduling plan of the lower bound in Theorem 2 and the upper bound in Theorem 3 have the identical scheduling outcome. It means that each job in the optimal scheduling within the upper bound starts processing $\max\{G_k, t_k^m\} - G_k$ later than the job of the optimal scheduling plan within the lower bound. Based on Lemma 4, it is obtained that

$$v_k^{ub} \leq \frac{\max\{G_k, t_{k-1}^m\}}{G_k} v_k^{lb}.$$

Moreover, Theorem 2 shows that $v_k^o \leq v_k^{ub}$, hence we have

$$v_k^o \leq v_k^{ub} \leq \frac{\max\{G_k, t_{k-1}^m\}}{G_k} v_k^{lb}.$$

□

In Theorem 4, we proved that $v_k^o \leq \frac{\max\{G_k, t_{k-1}^m\}}{G_k} v_k^{lb}$, $\forall k \in K \setminus \{1\}$. However, the value of ϵ_k might be extremely large when G_k approaches 0. Hence, a better upper bound for the optimal objective value, for sub-problems with small stage time, should be developed. The following assumption is included to set a limit for $\epsilon = \max_k(\epsilon_k)$.

Assumption 1 In $Rm|r_j| \sum w_j C_j$, it is assumed that

$$G_k = \begin{cases} p_{max}, & \text{if } r_j \leq p_{max}; \\ r_j, & \text{otherwise.} \end{cases}$$

where $k \in K$, $j \in N$, $p_{max} = \max_{i,j} p_{ij}$.

Based on Assumption 1, the job set of the first sub-problem is denoted as $N_1 = \{j | r_j \leq p_{max}, j \in N\}$. Specifically, for each job $j \in N_1$, it only can be processed after G_1 instead of r_j . Hence the optimal solution of the first sub-problem based on Assumption 1 can be known as the upper bound of the optimal solution of the first sub-problem without Assumption 1. However, the processing of jobs in the k_{th} sub-problem, $\forall k \in K \setminus \{1\}$, will be postponed. Then the relationship between the optimal objective values of $Rm|r_j| \sum w_j C_j$ with and without Assumption 1 will be discussed.

Lemma 6 In $Rm|r_j| \sum w_j C_j$, we have

$$\hat{v}^o \leq \mu v^o. \quad (29)$$

where $\mu \leq 2$, and $\hat{v}^o$ is the optimal objective value which satisfies $G_1 = p_{max}$, $\forall j \in N$, $r_j \leq p_{max}$.

Proof In Assumption 1, we let $G_1 = p_{max}$ if $r_j \leq p_{max}$, $\forall i \in M$, $j \in N$. Then we could conduct a worst case analysis to evaluate the approximation performance of this algorithm. For the worst case scenario, all the jobs in the first stage are released extremely close to time zero, but will be processed after G_1 at the earliest. Thus it is equivalent to postpone the completion time of any job $j \in N$ by p_{max} .

In that case, we have $\hat{v}^{ub} = \sum w_j C_j + \sum w_j p_{max}$, where $\hat{v}^{ub}$ represents an upper bound of $\hat{v}^o$. Moreover, $v^o = \sum w_j C_j$ indicates that it is the optimal solution to the original scheduling problem. Then we have

$$\hat{v}^{ub} = \mu v^o, \text{ where } \mu = \frac{\sum w_j C_j + p_{max} \sum w_j}{\sum w_j C_j}.$$

In the actual manufacturing process, since jobs released in the first stage would not all be postponed by p_{max} , the optimal solution $\hat{v}^o \leq \hat{v}^{ub}$. Therefore, for more general cases where the situations are less extreme, the numerator part $\sum w_j C_j + p_{max} \sum w_j$ in the above fractional expression would be reduced, which is equivalent to the decrease of $\hat{v}^{ub}$, then the following inequality stands.

$$\hat{v}^o \leq \hat{v}^{ub} = \mu v^o, \text{ where } \mu = \frac{\sum w_j C_j + p_{max} \sum w_j}{\sum w_j C_j}.$$

Moreover, the inequality $\mu \leq 2$ holds, because $p_{\max} \leq C_j$ for all $j \in N$. As a result, the inequality $\mu \leq 2$ holds in the proposed stage construction model. $\square$

In Lemma 6, the most significant influence on the approximation bound exerted by the above postponement operation in Assumption 1 has been obtained via the worst case analysis. Based on the conclusions derived in Lemma 6, we could eventually evaluate the approximation performance of D-SDP algorithm as illustrated below.

Theorem 5 (4-Approximation) *D-SDP is a 4-approximation algorithm.*

Proof In the k_{th} sub-problem, the upper bound of the objective value of the scheduling plan for the k_{th} sub-problem could be obtained by D-SDP algorithm, $v_k^o \leq \epsilon_k v_k^{lb}$, where $\epsilon_k = \frac{\max\{G_k, t_{k-1}^m\}}{G_k}$, as it is illustrated in Theorem 4. Although the stage dates for the jobs whose release dates are less or equal to G_1 are deliberately set to $p_{\max}$ by the divide and combine method, it is firstly assumed that the operations will not affect the successive stages. Let $t_0^m = 0$, then we have the following inequality,

$$v^D \leq \sum_{k \in K} \frac{\max\{G_k, t_{k-1}^m\}}{G_k} v_k^{lb}$$

Counting into the effects of the divide and combine method, after setting the first stage date to be $p_{\max}$ for all the jobs whose release dates are less or equal to G_1 , the optimal objective value, $\hat{v}^o$, will be larger than the sum of the lower bound of each sub-problem ($\sum v_k^{lb}$) without considering the postponement effect.

$$\sum_{k \in K} \frac{\max\{G_k, t_{k-1}^m\}}{G_k} v_k^{lb} \leq \max_{k \in K} \left\{ \frac{\max\{G_k, t_{k-1}^m\}}{G_k} \right\} \hat{v}^o$$

Moreover, since Lemma 6 demonstrates that $\hat{v}^o \leq \mu v^o$ for the relationship between the optimal solution with/without considering divide and combine method, thus the following inequality holds,

$$\max_{k \in K} \left\{ \frac{\max\{G_k, t_{k-1}^m\}}{G_k} \right\} \hat{v}^o \leq \max_{k \in K} \left\{ \frac{\max\{G_k, t_{k-1}^m\}}{G_k} \right\} \mu v^o.$$

Finally, for the k_{th} sub-problem, since $t_k^m \leq G_k + p_{\max}$, $\max\{G_k, t_{k-1}^m\}/G_k \leq 2$ holds. In addition, based on Lemma 6 we have $\mu \leq 2$, thus the approximation rate $v^D \leq 4v^o$ is obtained. $\square$

Actually, for the more general situation, the value of $\epsilon = \max_k(\epsilon_k)$ would approach the average value of ϵ_k of the last several stages; and μ would get close to 1 when the number of jobs becomes sufficiently large. To validate the derived theorem, a special case is considered where identical groups of jobs are allocated into every stage. Then the sum of the weighted completion times for the k_{th} stage will increase with the value of k . It means that the larger k is, the more influence ϵ_k will have for the overall ϵ . Moreover, when the value of G_k is large enough, μ approaches 1, which means that D-SDP performance will be enhanced with the increase of job quantity. Furthermore, the computation results in the next section will also show that the gap between the optimal solution and our proposed algorithm is within 1% for most instances.

In the next section, we will show the performance of our algorithms and compare it with the solution derived by CPLEX, the WCT-NEH algorithm which was proposed by Lin and Lin (2013), and the algorithm proposed by Hall et al. (1997).

4 Computational results

In this section, we present a series of computational experiments to test the performance of our algorithm. The MIP model and approximation algorithm of Hall et al. (1997) is solved by CPLEX 12.6. We use Corollary 1 to further reduce the CPU time for solving the MIP model. All other algorithms have been implemented by MATLAB R2016a. The experiments are performed on a PC with a 2.3-GHz CPU and 8 GB memory.

4.1 The comparison of D-SDP, FD-SDP and other existed algorithm

In order to conduct an intensive performance analysis, we consider job groups of numerical instances in terms of the number of jobs: $n \in \{20, 30, 40, 50, 60, 70, 80\}$ (medium scale instances) and $n \in \{100, 150, 200, 250\}$ (large scale instances). And the machine groups are divided into a set of different machines numbers: $m \in \{2, 3, 4, 5\}$. Besides, to assimilate the natural variations of the operation process, the processing times of jobs are uniformly distributed in the ranges $U[20, 50]$. And the weights of the jobs' completion times are also uniformly distributed in the range $U[2, 10]$. Finally, the release dates of jobs are again uniformly distributed in the ranges $U[1, P]$, where $P = \sum_{i=1}^m \sum_{j=1}^n p_{ij} / m$.

To conduct an in-depth analysis on the quality of solutions derived by the proposed algorithms, $V(\text{algorithm})$ denotes the objective function value of the solution of an algorithm. In small-medium instances, Gap1 is defined as $(V(D\text{-SDP}) - V(CPLEX)) / V(CPLEX) \times 100\%$, $(V(FD\text{-SDP}) - V(CPLEX)) / V(FD\text{-SDP}) \times 100\%$ is the value of Gap2, and the definition of Gap3 is $(V(WCT\text{-NEH}) - V(CPLEX)) / V(WCT\text{-NEH}) \times 100\%$. Gap2 is defined as $(V(CPLEX) - V(Hall(1997))) / V(CPLEX) \times 100\%$. For the cases where CPLEX cannot deliver an optimal solution in a reasonable period of time, NaN is used to represent the gap. Due to the curse of dimensionality, for large instances, CPLEX cannot obtain the optimal outcome most of the time.

In Tables 1 and 2, the averaged results over 5 trials for medium instances are reported. Each problem is solved by CPLEX, D-SDP, FD-SDP, WCT-NEH proposed by Lin and Lin (2013), and the approximation algorithm provided by Hall et al. (1997). It is observed that the CPU time of CPLEX based algorithm would increase rapidly with the number of jobs, machines and the value of t_{max} . For the instances where CPLEX is unable to obtain the optimal solution because the memory of PC is full, we set the CPU-time of CPLEX as infinity.

For each problem instance, Gap1 and Gap2 are usually less than 2%. D-SDP algorithm and FD-SDP algorithm always have similar solutions. Moreover, the CPU-times of D-SDP algorithm and FD-SDP algorithm would increase with the number of jobs, but the gap would decline with the number of jobs. The WCT-NEH algorithm can solve the problem rather quickly, but the Gap3 would increase rapidly with the number of jobs.

Figures 6 through 9 show the gaps and CPU times for small-medium instances on 2 to 5 unrelated parallel machines, respectively. It is observed that CPLEX could only find the exact solutions to the instances with very limited number of jobs and machines. In comparison, the WCT-NEH method could locate a solution quickly, but the quality of the solution is not assured. Moreover, the gap of WCT-NEH method would increase with the number of jobs. Similarly, the approximation algorithm of Hall can also derive the solution in a fast manner, but again the gap is not satisfactory. In Figs. 6a, 7a, 8a and 9a, the relationship of D-SDP and FD-SDP are zoomed in on the center of figures. It should be pointed out that the lines of D-SDP and FD-SDP overlapped in both Figs. 6b, 7b, 8b and 9b. To our relief, the proposed D-SDP and FD-SDP algorithms can always obtain a high quality solution for most cases,

Table 1 The performance of DP-SDP, QDP-SDP and WCT-NEH for $m = 2, 3$

m	n	CPLEX	D-SDP		FD-SDP		WCT-NEH		Hall (1997)	
		CPU time	CPU time	Gap1	CPU time	Gap2	CPU time	Gap3	CPU time	Gap4
2	20	5.93	29.90	0.48%	8.22	0.47%	0.01	9.51%	0.14	41.98%
	30	13.71	58.28	0.29%	9.27	0.33%	0.01	19.70%	0.20	37.53%
	40	22.45	71.01	0.23%	11.99	0.12%	0.02	24.77%	0.24	36.13%
	50	35.97	86.82	0.12%	15.11	0.12%	0.03	30.81%	0.26	43.31%
	60	52.20	108.89	0.09%	18.27	0.09%	0.04	29.77%	0.26	41.66%
	70	72.91	112.86	0.04%	20.97	0.04%	0.05	31.40%	0.27	34.55%
	80	91.87	136.66	0.04%	24.37	0.05%	0.08	35.83%	0.29	39.32%
	80	∞	190.09	NaN	26.50	NaN	0.09	NaN	0.42	NaN
3	20	9.17	56.03	0.67%	6.81	0.21%	0.01	3.25%	0.20	35.93%
	30	19.95	70.63	0.37%	10.01	0.11%	0.01	8.98%	0.24	45.53%
	40	36.70	95.21	0.24%	13.42	0.03%	0.02	12.76%	0.26	43.35%
	50	54.76	141.84	0.29%	16.68	0.08%	0.03	14.09%	0.31	44.08%
	60	79.82	142.51	0.25%	20.06	0.04%	0.04	17.54%	0.34	40.52%
	70	108.92	169.73	0.30%	23.50	0.04%	0.06	24.48%	0.40	35.38%
	80	∞	190.09	NaN	26.50	NaN	0.09	NaN	0.42	NaN

Table 2 The performance of DP-SDP, QDP-SDP and WCT-NEH for $m = 4, 5$

m	n	CPLEX	D-SDP		FD-SDP		WCT-NEH		Hall (1997)	
		CPU time	CPU time	Gap1	CPU time	Gap2	CPU time	Gap3	CPU time	Gap4
4	20	12.55	57.54	1.30%	7.10	0.48%	0.01	0.89%	0.25	28.51%
	30	27.46	86.00	0.75%	11.01	0.25%	0.01	1.86%	0.27	41.53%
	40	50.43	104.34	0.50%	13.98	0.12%	0.02	7.73%	0.32	41.81%
	50	76.35	134.73	0.23%	17.78	0.08%	0.04	8.10%	0.39	47.73%
	60	111.96	201.98	0.46%	21.89	0.02%	0.05	12.79%	0.39	39.36%
	70	156.09	233.82	0.19%	25.42	0.02%	0.07	13.25%	0.40	35.47%
	80	∞	334.99	NaN	28.69	NaN	0.09	NaN	0.42	NaN
	80	∞	455.65	NaN	32.63	NaN	0.11	NaN	0.54	NaN
5	20	17.01	81.66	0.15%	7.83	0.11%	0.01	0.23%	0.23	34.39%
	30	35.41	169.49	0.12%	12.20	0.09%	0.01	0.68%	0.32	41.30%
	40	62.62	202.85	0.04%	16.16	0.03%	0.02	2.24%	0.33	34.51%
	50	99.15	276.73	0.03%	20.44	0.02%	0.04	5.17%	0.36	44.27%
	60	151.99	304.75	0.03%	24.08	0.02%	0.06	5.79%	0.45	42.24%
	70	∞	415.97	NaN	28.24	NaN	0.09	NaN	0.53	NaN
	80	∞	455.65	NaN	32.63	NaN	0.11	NaN	0.54	NaN

even though D-SDP would require more time to tackle the problem because of the B&B algorithm.

Provided that the FD-SDP algorithm outperforms D-SDP in terms of computational time, it doesn't mean the inferiority of the quality of its outcome. It is observed from the numerical examples that the FD-SDP algorithm would have a better solution than D-SDP algorithm from time to time. In both medium and large scale instance, the solutions yielded by FD-SDP would be better than D-SDP occasionally. To dig deeper with the observed phenomenon, the

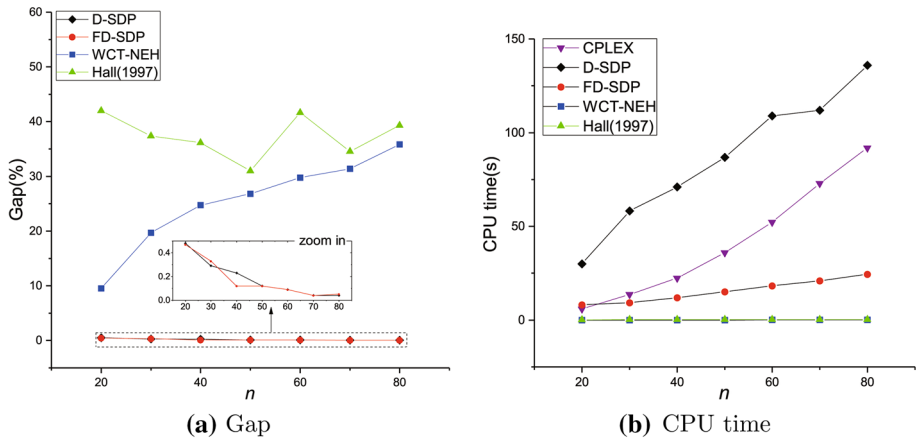


Fig. 6 The performance for $m = 2$

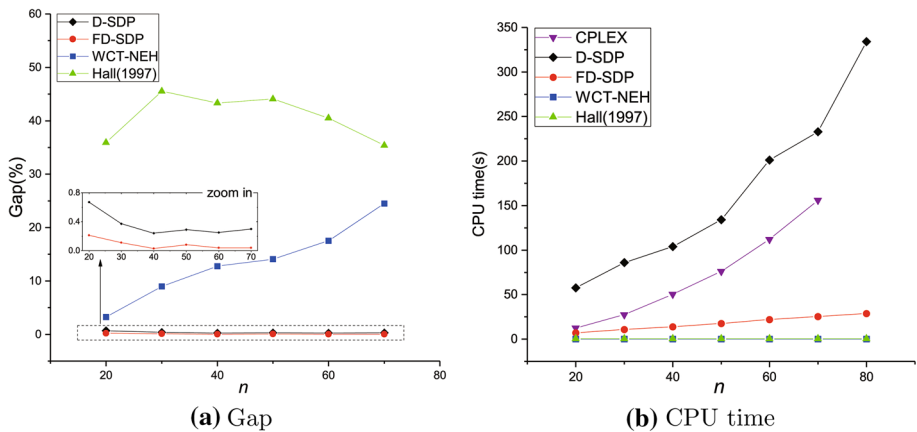
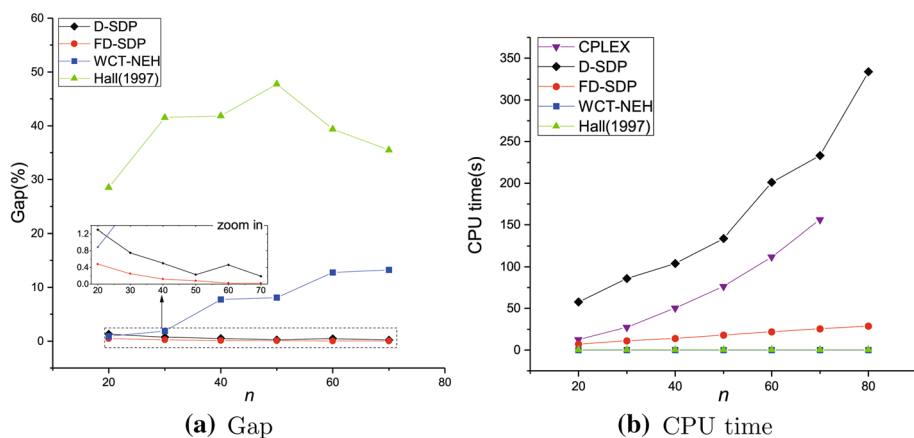
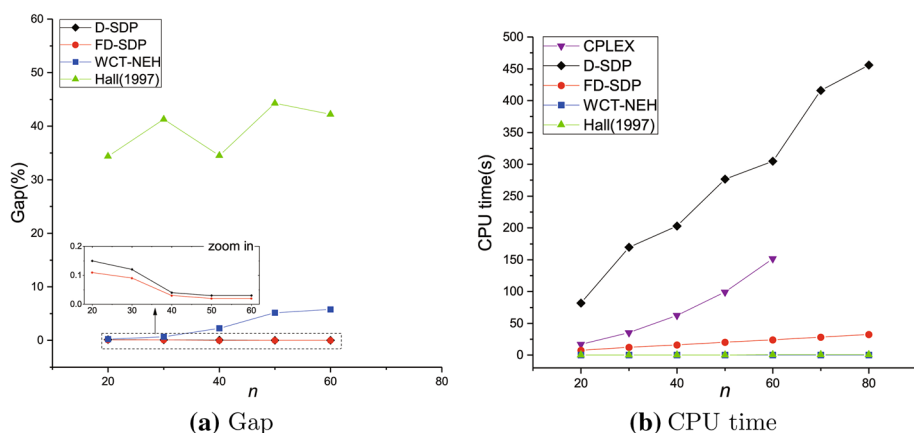


Fig. 7 The performance for $m = 3$

following discoveries may be worthwhile to pursue. For FD-SDP algorithm, the local search algorithm is applied to search for the solution of each sub-problem, which may not be the optimal solution because of the internal mechanism of local searching. However, since the aim of solving each sub-problem is to find the scheduling with the minimized sum of weighted completion time instead of the makespan, local search based algorithm can indeed obtain an ideal objective value sometimes. The reason for which is that the jobs of its successive stage will start processing earlier given there are straddling jobs in the current stage. And hence the successive sub-problem would have a smaller sum of weighted completion time.

For those instances where both algorithms could deliver valid results, we compare their performances with the proposed D-SDP algorithm. Here Gap-L1 is defined as $(V(D-SDP) - V(FD-SDP))/V(D-SDP) \times 100\%$, Gap-L2 is defined as $(V(WCT-NEH) - V(D-SDP))/V(WCT-NEH) \times 100\%$, and $(V(D-SDP) - V(Hall(1997)))/V(D-SDP) \times 100\%$ is the value of Gap-L3.

In Table 3, the averaged result over 5 trials for large instances are reported. Here we just compare the performance of D-SDP, FD-SDP, WCT-NEH, and the approximation algorithm

Fig. 8 The performance for $m = 4$ Fig. 9 The performance for $m = 5$

of Hall et al. (1997), because CPLEX usually fail to obtain the solution for large instances. It is observed that the CPU time of D-SDP and FD-SDP would increase continuously with the number of jobs and machines. Besides, it is found that FD-SDP is always more efficient than D-SDP, and the two proposed algorithms can obtain solutions of similar quality.

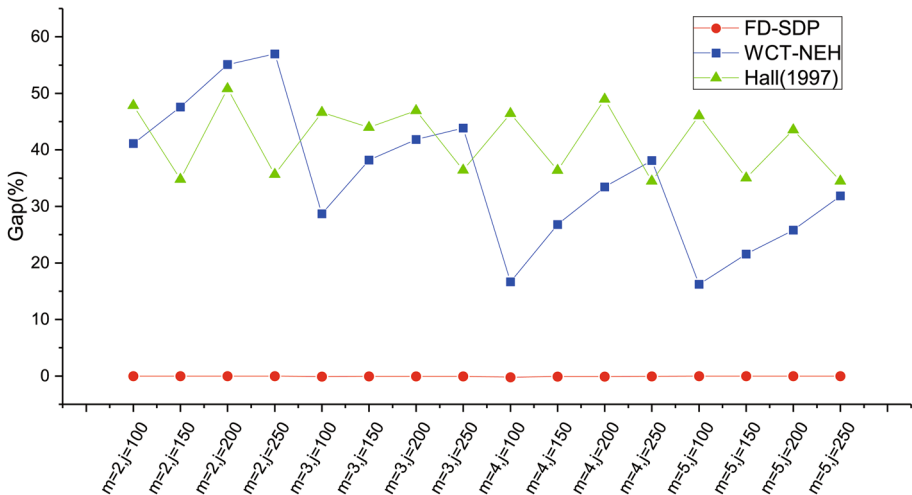
Figure 10 demonstrates the average relative gap for large sized instances. It seems that the FD-SDP algorithm is always much more efficient than other methods in this setting. In Fig. 11, however, it is witnessed that D-SDP algorithm would spend relatively more time in deriving the solution. In comparison, WCT-NEH method can always find a solution with less consumed time in these instances. Specially, the lines of WCT-NEH and Hall overlapped. The FD-SDP algorithm can solve these scheduling problems within a satisfactory time range.

4.2 The statistical analysis for D-SDP and FD-SDP algorithms

In this subsection, the statistical analysis for D-SDP and FD-SDP algorithm are provided. Several computational examples for medium scale instances are considered. Their optimal

Table 3 The performance of DP-SDP, QDP-SDP and WCT-NEH for large size instance

m	n	D-SDP	FD-SDP		WCT-NEH		Hall (1997)	
		CPU time	CPU time	Gap-L1 (%)	CPU time	Gap-L2 (%)	CPU time	Gap-L3 (%)
2	100	187.01	30.78	0.00	0.13	41.13	0.31	47.85
	150	283.74	45.29	0.00	0.36	47.57	0.45	34.81
	200	363.12	61.38	0.00	0.74	55.10	0.57	50.82
	250	463.26	80.38	0.00	1.83	56.95	0.67	35.66
3	100	258.10	34.16	-0.11	0.15	28.71	0.41	46.63
	150	316.77	49.51	-0.04	0.40	38.19	0.49	43.97
	200	575.62	68.09	0.05	0.83	41.81	0.67	46.96
	250	737.87	81.74	-0.05	1.54	43.85	0.82	36.42
4	100	551.21	35.76	-0.20	0.17	16.65	0.43	46.42
	150	644.17	53.69	-0.11	0.48	26.80	0.63	36.36
	200	883.83	89.46	-0.50	0.89	33.43	0.79	48.95
	250	980.20	73.17	0.07	1.56	38.13	0.83	34.46
5	100	607.63	39.90	0.00	0.18	16.22	0.60	46.04
	150	874.59	61.60	0.00	0.51	21.56	0.73	35.04
	200	1389.27	84.08	0.00	1.07	25.80	0.92	43.53
	250	1491.68	100.75	0.00	1.84	31.88	1.06	34.46

**Fig. 10** The average relative gap for large size instances

objective values are provided by MIP model solved by CPLEX 12.6. In both Figs. 12 and 13, the gap results over 24 combinations in terms of jobs and machines, each of which consists of 100 randomly generated instances, are reported for D-SDP and FD-SDP, respectively.

In Fig. 12, Gap is defined as $(V(D-SDP) - V(CPLEX))/V(D-SDP) \times 100\%$. In Fig. 12a, the extreme outliers exist for a few instances. But the gap values of all the extreme outliers would not exceed 9% in our computational examples. In Fig. 12b the box chart was zoomed in to better focus on the details. Generally, for the instances with identical number of machine,

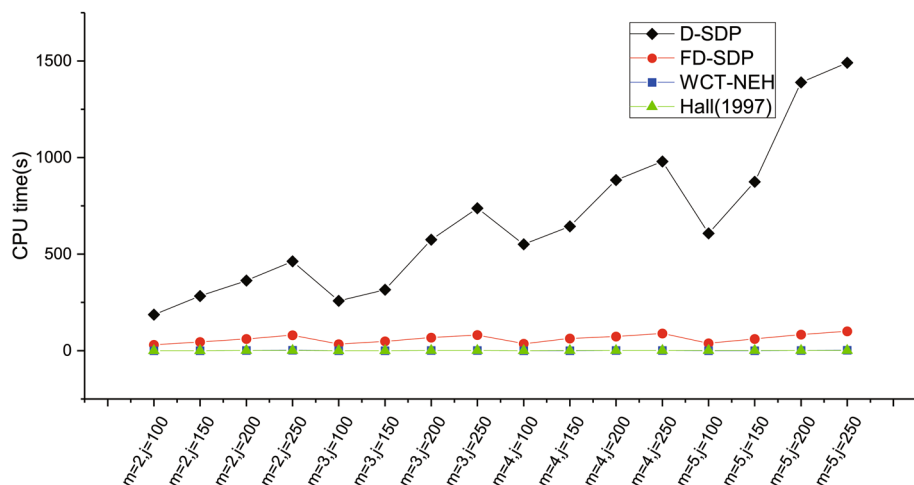


Fig. 11 The average CPU time for large size instances

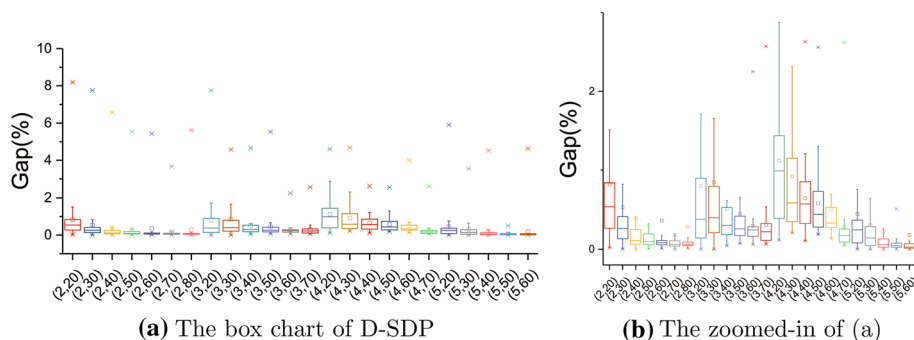


Fig. 12 The statistical analysis for D-SDP algorithm

the interquartile range would decrease with the increase of job quantity. Moreover, most gap values obtained with D-SDP algorithm in our computational examples would not exceed 3%.

In Fig. 13, Gap is defined as $(V(FD-SDP) - V(CPLEX)) / V(FD-SDP) \times 100\%$. Comparatively speaking, the extreme outliers in Fig. 13a are relatively larger than their counterparts in Fig. 12a. Mostly the values of the extreme outliers would not exceed 14% in our computational examples. As it is shown in Fig. 13b, for the instances with identical number of machine, the interquartile range would also decrease with the increase of job quantity. In general, most of the gap values would not exceed 2% in all of our computational examples. More importantly, an interesting fact could be observed in the above figures, which is the solution quality of FD-SDP turns better as the numbers of jobs and machines increase. Which is equivalent to state that the algorithm performs well as the size of the problem grows.

In Fig. 14, the number of instances with specified gaps out of 100 randomly generated cases in each job/machine setting for D-SDP and FD-SDP, is demonstrated. In specific, Fig. 14a to (d) have depicted the number of instances, whose gaps are less than 0.5% and 1%, with unequal number of machines 2 to 5, respectively. As it is shown in Fig. 14, the number of instances, for D-SDP and FD-SDP with gaps less than 0.5%, would both increase with the

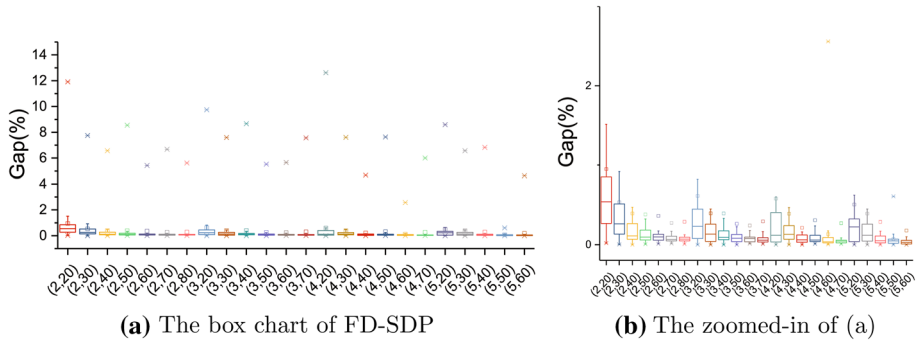


Fig. 13 The statistical analysis for FD-SDP algorithm

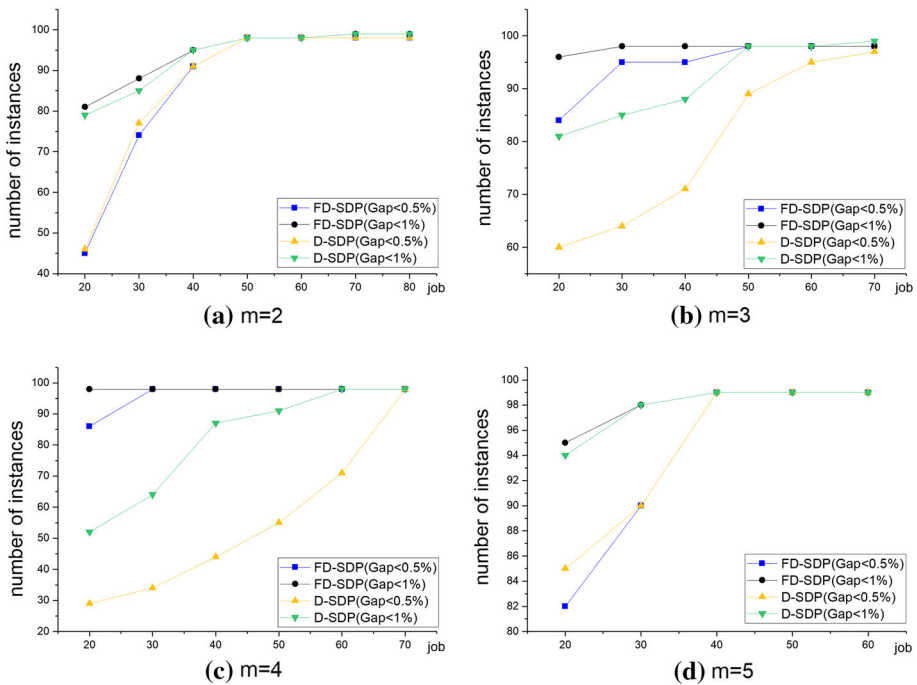


Fig. 14 The number of instances for D-SDP and FD-SDP with gaps less than 0.5% and 1%

number of jobs. Similarly, the number of instances, for D-SDP and FD-SDP with gap less than 1%, would also increase with the number of jobs.

5 Conclusion

In this research, the unrelated parallel machine scheduling with release dates to minimize the sum of weighted completion time is investigated. Two approximation algorithms, namely D-SDP and FD-SDP are proposed to tackle this NP-hard problem. In the divide and combine process, the $Rm|r_j| \sum w_j C_j$ problem is divided into a series of $Rm|| \sum w_j C_j$ sub-problems

based on their release dates. Then a QIP formulation is considered for each sub-problem, and an SDP relaxation is presented for each QIP model. After combining the solutions of the subproblems to constitute the solution of the primal problem, it is proved that D-SDP algorithm is a 4-approximation algorithm under constant worst-case scenario, which promotes the well-known 16/3-approximation given by Hall et al. In the computational results section, several different sized instances are constructed to verify the performance of our algorithms, in comparison with the outcomes of CPLEX and other approximation algorithms, such as WCT-NEH algorithm. Furthermore, during the extensive statistical analysis, it is observed that D-SDP and FD-SDP algorithms could almost always provide high quality solutions with less than 1% gaps. And the gap will further decrease as the size of the scheduling problem enlarges. It is also discovered that the computational time of FD-SDP is comparable to that of WCT-NEH or the approximation algorithm designed by Hall et al, which may incur relatively larger gaps in return.

Acknowledgements The authors are grateful for the financial support partly from the Natural Science Foundation of China under Grant Nos. 71871203, 51305400, Zhejiang Provincial Natural Science Foundation of China under Grant No. LY18G010017, and Northern Illinois University.

References

- Arnaout, J.-P. (2019). A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Annals of Operations Research*, 1–21. <https://doi.org/10.1007/s10479-019-03138-w>.
- Azizoglu, M., & Kirca, O. (1999a). On the minimization of total weighted flow time with identical and uniform parallel machines. *European Journal of Operational Research*, 113(1), 91–100.
- Azizoglu, M., & Kirca, O. (1999b). Scheduling jobs on unrelated parallel machines to minimize regular total cost functions. *IIE Transactions*, 31(2), 153–159.
- Balas, E., Ceria, S., Cornuejols, G., & Pataki, G. (1994). *Updated semi-definite constraints*. Tech. rep.: Technical Report, Carnegie Mellon University, Pittsburgh, USA.
- Borchers, B. (1999). Csdp, ac library for semidefinite programming. *Optimization Methods and Software*, 11(1–4), 613–623.
- Bülbül, K., & Şen, H. (2017). An exact extended formulation for the unrelated parallel machine total weighted completion time problem. *Journal of Scheduling*, 20(4), 373–389.
- Chen, J. (2015). Unrelated parallel-machine scheduling to minimize total weighted completion time. *Journal of Intelligent Manufacturing*, 26(6), 1099–1112.
- Chen, Z., & Powell, W. B. (1999). Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing*, 11(1), 78–94.
- Cheng, C., & Huang, L. (2017). Minimizing total earliness and tardiness through unrelated parallel machine scheduling using distributed release time control. *Journal of Manufacturing Systems*, 42, 1–10.
- Daniels, R. L., Hoopes, B. J., & Mazzola, J. B. (1997). An analysis of heuristics for the parallel-machine flexible-resource scheduling problem. *Annals of Operations Research*, 70, 439–472.
- Dietrich, B. L., & Escudero, L. F. (1993). On modelling the maximum workload allocation for parallel unrelated machines with setups. *Annals of Operations Research*, 43(7), 357–377.
- Djurasevic, M., Jakobovic, D., & Knezevic, K. (2016). Adaptive scheduling on unrelated machines with genetic programming. *Applied Soft Computing*, 48, 419–430.
- Gao, Y., Yuan, J., Ng, C., & Cheng, T. (2019). A further study on two-agent parallel-batch scheduling with release dates and deteriorating jobs to minimize the makespan. *European Journal of Operational Research*, 273(1), 74–81.
- Hall, L. A., Schulz, A. S., Shmoys, D. B., & Wein, J. (1997). Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22(3), 513–544.
- Hoogeveen, H., Schuurman, P., & Woeginger, G. J. (1998). Non-approximability results for scheduling problems with minsum criteria. In *International conference on integer programming and combinatorial optimization* (pp. 353–366). New York: Springer.
- Jäger, S. (2018). Approximating total weighted completion time on identical parallel machines with precedence constraints and release dates. *Operations Research Letters*, 46(5), 505–509.

- Lancia, G. (2000). Scheduling jobs with release dates and tails on two unrelated parallel machines to minimize the makespan. *European Journal of Operational Research*, 120(2), 277–288.
- Lee, Y., & Sherali, H. D. (1994). Unrelated machine scheduling with time-window and machine downtime constraints: An application to a naval battle-group problem. *Annals of Operations Research*, 50(1), 339–365.
- Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1, 343–362.
- Li, K., & Yang, S. (2009). Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied Mathematical Modelling*, 33(4), 2145–2158.
- Lin, Y., & Lin, C. (2013). Dispatching rules for unrelated parallel machine scheduling with release dates. *International Journal of Advanced Manufacturing Technology*, 67(1–4), 269–279.
- Lovász, L., & Schrijver, A. (1991). Cones of matrices and set-functions and 0–1 optimization. *SIAM Journal on Optimization*, 1(2), 166–190.
- Mir, M. S. S., & Rezaeian, J. (2016). A robust hybrid approach based on particle swarm optimization and genetic algorithm to minimize the total machine load on unrelated parallel machines. *Applied Soft Computing*, 41, 488–504.
- Mokotoff, E., & Jimeno, J. (2002). Heuristics based on partial enumeration for the unrelated parallel processor scheduling problem. *Annals of Operations Research*, 117(1–4), 133–150.
- Nessah, R., Yalaoui, F., & Chu, C. (2008). A branch-and-bound algorithm to minimize total weighted completion time on identical parallel machines with job release dates. *Computers & Operations Research*, 35(4), 1176–1190.
- Phillips, C., Stein, C., & Wein, J. (1997). Task scheduling in networks. *SIAM Journal on Discrete Mathematics*, 10(4), 573–598.
- Rendl, F. (2016). Semidefinite relaxations for partitioning, assignment and ordering problems. *Annals of Operations Research*, 240(1), 119–140.
- Rodriguez, F. J., Blum, C., García-Martínez, C., & Lozano, M. (2012). Grasp with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times. *Annals of Operations Research*, 201(1), 383–401.
- Rodriguez, F. J., Lozano, M., Blum, C., & García-Martínez, C. (2013). An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. *Computers & Operations Research*, 40(7), 1829–1841.
- Schulz, A. S., & Skutella, M. (2002). Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics*, 15(4), 450–469.
- Sherali, H. D., & Adams, W. P. (1994). A hierarchy of relaxations and convex hull characterizations for mixed-integer zeroone programming problems. *Discrete Applied Mathematics*, 52(1), 83–106.
- Siepak, M., & Józefczyk, J. (2014). Solution algorithms for unrelated machines minmax regret scheduling problem with interval processing times and the total flow time criterion. *Annals of Operations Research*, 222(1), 517–533.
- Sitters, R., & Yang, L. (2018). A $(2 + \epsilon)$ -approximation for precedence constrained single machine scheduling with release dates and total weighted completion time objective. *Operations Research Letters*, 46(4), 438–442.
- Skutella, M. (2001). Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48(2), 206–242.
- Skutella, M. (2016). A 2.542-approximation for precedence constrained single machine scheduling with release dates and total weighted completion time objective. *Operations Research Letters*, 44(5), 676–679.
- Skutella, M., Sviridenko, M., & Uetz, M. (2016). Unrelated machine scheduling with stochastic processing times. *Mathematics of Operations Research*, 41(3), 851–864.
- Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics*, 3(1–2), 59–66.
- Tang, L., & Zhang, Y. (2011). A new lagrangian relaxation algorithm for scheduling dissimilar parallel machines with release dates. *International Journal of Systems Science*, 42(7), 1133–1141.
- Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, 58(4), 785–800.
- van Den Akker, J. M., Hoogeveen, J. A., & van de Velde, S. L. (1999). Parallel machine scheduling by column generation. *Operations Research*, 47(6), 862–872.
- Vredevelde, T., & Hurkens, C. (2002). Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. *INFORMS Journal on Computing*, 14(2), 175–189.
- Yalaoui, F., & Chu, C. (2006). New exact method to solve the $pm|r_j|\sum c_j$ schedule problem. *International Journal of Production Economics*, 100(1), 168–179.

Zhang, X., Xu, D., Du, D., & Miao, C. (2016). Approximate algorithms for unrelated machine scheduling to minimize makespan. *Management*, 12(2), 771–779.

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