A New Approximation Algorithm for Unrelated Parallel Machine Scheduling Problem with Release Dates

Zhi Pei[a]*, Mingzhong Wan[a,b], Ziteng Wang[b]

[a] *Department of Industrial Engineering, Zhejiang University of Technology, Hangzhou 310014, China*

[b] *Department of Industrial and Systems Engineering, Northern Illinois University, DeKalb, IL 60115, USA*

Abstract

In this research, we consider the unrelated parallel machine scheduling problem with release dates. The goal of this scheduling problem is to find an optimal job assignment with minimal sum of weighted completion times. As it is demonstrated in the present paper, this problem is NP-hard in the strong sense. Albeit the computational complexity, which renders the optimality seeking a formidable task within polynomial time, a 4-approximation algorithm is devised and proved in comparison with the 16/3-approximation(Hall et al. 1997). In the newly proposed algorithm, the original scheduling problem is divided into several sub-problems based on the release dates. For each sub-problem, a convex quadratic integer programming (CQIP) 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 0-1 constraint, a branch and bound (B&B) based method and a local search (LS) strategy are applied separately to locate the integer solution of each sub-problem. Consequently, the solution of the original scheduling problem can be constituted by integrating the outcomes of the sub-problems with the help of the proposed approximation algorithm. In the case study section, the computational efficiency and accuracy of the presented algorithm are verified over wide range of instance sizes in terms of CPU time and quality of solutions.

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

*Corresponding author. Tel:+8618858123635. E-mail address: peizhi82@163.com

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 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. Van den Akker, et al.(1999) implemented the well known Smith's(1956) weighted shortest processing time(WSPT) rule to each of the m machines separately. Sen and Bulbul(2017) extended this formulation to Rm-TWCT setting. They managed to prove that in the optimal solution the jobs assigned to each machine should be sequenced in WSPT order, which really acts as an essential inspiration that allows us to divide the $Rm|r_j|\sum w_jC_j$ problem into a series of sub-problems. 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 $P||\sum w_jC_j$ problem. Yalaoui and Chu (2006) provided a B&B method for $Pm|r_j|\sum w_jC_j$ problem. Another B&B algorithm for $P|r_j|\sum w_jC_j$ problem is introduced in Nessah, et al.(2008). Lancia (2000) investigated a scheduling problem under unrelated parallel machines environment. They provided a B&B algorithm to solve the scheduling problem with release dates, and applied it to two unrelated parallel machines with the minimal makespan. Schulz and Skutalla (2001) provided a (2+ϵ)-approximation algorithm and a (3/2+ϵ)-approximation algorithm for the $Rm||\sum w_jC_j$ problem. Another approximation algorithm for $Rm||\sum w_jC_j$ problem is obtained by Unlu and Mason(2010). Since the unrelated parallel machine scheduling is of NP-hard nature, Chen (2015) designed three heuristics for $Rm||\sum w_jC_j$ problem. For these heuristics, two are based on record-to-record travel and the other one is based on random descent search. Recently, Zhang (2016) proposed a 2-approximation algorithm for two unrelated machine scheduling problems with machine dependent release dates to minimize the makespan. Durasevic(2016) provided an automatized synthesis of heuristics for the unrelated parallel machines base on genetic programming. Moreover, several other important scheduling problems in unrelated parallel machine environment are considered by Skutella (2016), Mir (2016) and Cheng (2017).

Azizoglu and Kirca (1999a,b) designed two B&B methods to handle the non-identical parallel machine environment. To the best of our knowledge, the first exact approach to deal with scheduling problem $Rm||\sum w_jC_j$ is proposed by Azizoglu and Kirca (1999b). 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. Enlightened by Skutella's method, in the present paper, we would extend it to a more special scheduling setting, where $Rm|r_j|\sum w_jC_j$ is divided into some sub-problems without 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 $Rm|r_j|\sum w_jC_j$ problem, is NP-hard. Thus apparently the scheduling problem $Rm|r_j|\sum w_jC_j$ is also NP-hard in nature, meaning the optimal schedule can't be derived in polynomial time. Due to the complexity of the unrelated parallel machine scheduling with release dates, the first approximation algorithm for $Rm|r_j|\sum w_jC_j$ problem appeared in Phillips et al.(1997). They demonstrated an $(16+\epsilon)$algorithm with performance guarantee. In the same year, Hall et al.(1997) also considered the problem $Rm|r_j|\sum w_jC_j$, and improved the solution bound by a 16/3-approximation algorithm base on LP-relaxation. Afrati et al.(1999) presented a polynomial time approximation schemes to solve the $Rm||\sum w_jC_j$ and $Rm|r_j, pmtn|\sum w_jC_j$ problems. Skutella(2001) also presented a polynomial time 2-approximation schemes to tackle a special case of $Rm|r_j|\sum w_jC_j$ problem, where the job release time is considered the same as starting time. In the work, he also claimed that the polynomial approximation bound for $Rm|r_{ij}|\sum w_jC_j$ was difficult to attain. Vredeveld and Hurkens (2002) conducted an empirical comparison of the polynomial-time approximation algorithms and local search heuristics for the problem $Rm|r_j|\sum w_jC_j$. Later, Tang and Zhang (2011) proposed a Lagrangian relaxation based algorithm for $Rm|r_j|\sum w_jC_j$ problem. A set of dispatching rules for $Rm|r_j|\sum w_jC_j$ problem appeared in Lin and Lin (2013). More recently, Avdeenko (2017) considered an approximation algorithm based on the scheme of modified dynamic programming with adaptive narrowing down of search domain, ensuring its computational efficiency, for scheduling unrelated parallel machines with release dates. Interested readers could always refer to Li and Yang(2009) and Rodriguez, et al.(2013) for their excellent reviews.

As for semi-definite programming (SDP) method, Lovasz 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,H.D.&Adams,W.P.(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,F.(2016) for the latest development of SDP relaxation appeared in the assignment and ordering problems.

In this research, we consider two new approximation algorithm, namely the divided semi-definite programming (D-SDP) and the fast divided semi-definite programming (FD-SDP) algorithm, for $Rm|r_j|\sum w_jC_j$ problem. In both algorithms, the $Rm|r_j|\sum w_jC_j$ problem is divided into several $Rm||\sum w_jC_j$ 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 Section 2, we consider a mixed integer programming (MIP) model of $Rm|r_j|\sum w_jC_j$ problem. The descriptions of our approximation algorithms are stated in Section 3, and the performance of these two algorithms are also reported in this section. In Section 4, the computational results are presented in compare with other published approaches. Section 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 algorithms for small sized instances. Furthermore, two efficient approximation algorithms for both medium and large instances of the unrelated parallel machine scheduling problem with release dates are proposed in the next section.

Before introducing the algorithms to tackle the unrelated parallel machine scheduling problem with release dates, we have to firstly construct the problem formulation. Fundamentally, there are a set of jobs, $N = \{1, 2, ..., n\}$, to be processed on m unrelated parallel machines, $M = \{1, 2, ..., m\}$. 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, $j \in M$ and release date r_j, $j \in M$. The overall objective of this scheduling problem is to minimize the total weighted completion times.

To facilitate the problem formulation, the MIP model for $Rm|r_j|\sum w_jC_j$ problem proposed by Lin and Lin (2013) will be employed. It is assumed that all jobs have to be completed by time T, where $T \geq \sum_{j\in N} r_j + \sum_{i\in M}\sum_{j\in N} p_{ij}$. In the formulation, C_j denotes 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 shown as below,

$$
\begin{aligned}
&min \quad \sum_{j\in N} W_j C_j \\
&s.t. \quad \sum_{i=1}^{m}\sum_{t=r_j}^{T-p_{ij}+1} x_{ijt} = 1, && j = 1, 2, ..., n \\
&\sum_{j=1}^{n}\sum_{s=\max(r_j, t-p_{ij}+1)}^{t} x_{ijs} \leq 1, && i = 1, 2, ..., m, t = 0, 1, ..., T \\
&\sum_{i=1}^{m}\sum_{t=0}^{r_j} x_{ijt} = 0, && j = 1, 2, ..., n \\
&C_j = \sum_{i=1}^{m}\sum_{t=0}^{T-p_{ij}+1} x_{ijt}(t - 1 + p_{ij}), && j = 1, 2, ..., n \\
&x_{ijt} \in \{0, 1\}, && i = 1, 2, ..., m, t = 0, 1, ..., T, j = 1, 2, ..., n
\end{aligned}
$$

In the above MIP model, the number of independent variables x_{ijt} increases with the value of T. To simplify the computation process, the number of independent variables is reduced by shrinking the span of the planning horizon T, which is an upper bound of the makespan. Azizoglu and Kirca (1999b) found that the total processing time on machine i of $Rm||\sum w_jC_j$ problem is less than $\frac{1}{m}\{\sum_j \max_q\{p_{qj}\} + \sum_{q\neq i}\max_j\{p_{qj}\}\}$, where q stands for a machine other than i. Based on that conclusion, further extension could be achieved for $Rm|r_j|\sum w_jC_j$ problem to locate a better value of T, and this tighter bound of the makespan could be illustrated through the following theorem,

Theorem 1. In $Rm|r_j|\sum w_jC_j$ problem,the completion time of jobs processed on machine i does not exceed $\frac{1}{m}\{\sum_j \max_q\{p_{qj}\} + \sum_{q\neq i}\max_j\{p_{qj}\}\} + r_l$, where i and q are different machines, i,q $\in$ M, j $\in$ N; r_l is the release time of the last job.

Proof. Considering the worst scenario for makespan minimization, it is assumed that all jobs would be available at the latest release time r_l. Then this special form of unrelated parallel machine scheduling problem $Rm|r_l|\sum w_jC_j$ is equivalent to an $Rm||\sum w_jC_j$ problem, in which the starting time of the first job on each machine is r_l. Hence the completion time of jobs processed on machine i does not exceed the bound $\frac{1}{m}\{\sum_j \max_q\{p_{qj}\} + \sum_{q\neq i}\max_j\{p_{qj}\}\} + r_l$. $\square$

Based on Theorem 1, a better upper bound of the makespan 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 decrease of T. To better depict the refined planning horizon, the following corollary is employed, and it could be directly derived from Theorem 1.

Corollary 1. In $Rm|r_j|\sum w_jC_j$ problem, let's assume that all jobs have to be completed by time T, then it is required that

$$T = \max_{i,q\in M, j\in N}\{\frac{1}{m}\{\sum_j \max_q\{p_{qj}\} + \sum_{q\neq i}\max_j\{p_{qj}\}\} + r_l\}, \text{where } i \neq q.$$

Therefore, in comparison with the original T value in the MIP formulation, where $T \geq \sum_{j\in N} r_j + \sum_{i\in M}\sum_{j\in N} p_{ij}$, the number of independent variables could be reduced substantially via Corollary 1. As a result, in small-scale instance, the aforementioned MIP model could be solved by commercial solver more quickly. 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 algorithms should be developed to solve the MIP model with medium and large instances.

3. D-SDP Approximation Algorithm

In literature, Lenstra et al. (1977) proved that the single machine scheduling problem with release dates, $1|r_j|\sum w_jC_j$, is strongly NP-hard. Therefore, $Rm|r_j|\sum w_jC_j$ problem, as a more generalized instance of $1|r_j|\sum w_jC_j$ is also NP-hard in the strong sense, 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 $Rm|r_j|\sum w_jC_j$ problem are provided with 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 MIP problem is divided into several sub-problems base on the ordered release dates, and these sub-problems could be fully addressed via semi-definite programming based algorithm respectively. 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, the D-SDP algorithm derived is proved to be a 4-approximation algorithm. In the following, we will introduce the divide and combine rules and the SDP based algorithm for the defined 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 $Rm|r_j|\sum w_jC_j$ problem into a series subproblems. As it is mentioned above, In the newly devised algorithms, the subproblems are defined by their respective release dates in non-decreasing order. Without loss of generality, a set of jobs are indexed with regard to their release time, i.e. $r_1 \leq r_2 \leq ...r_j \leq r_n$, where r_j is the release date of job j. To help construct the subproblem, the following definition is introduced to denote the stage time of a job and the stage time of a sub-problem.

Definition 1. Stage time for jobs and sub-problems

g_j: Suppose that g_j is the earliest possible time that job j can initiate its processing within a sub-problem, then g_j is denoted as the stage time of job j for $\forall j \in \{1, 2, ..., n\}$.

G_k: Suppose that all jobs in k_{th} sub-problem could start processing after G_k, then it is denoted as the stage time of k_{th} sub-problem.

For convenience's sake, the stage time has been manipulated so as to avoid the release time of early stages approaching r_1, which represents the starting point of planning horizon. Suppose $r_1 = 0$, otherwise the entire planning horizon could be shifted backwards r_1 time units without interfering the scheduling outcome. Suppose $p_{max} = \max_{i\in M, j\in N} p_{ij}$, and the stage time of each job is constructed as follows: if $r_j \leq p_{max}$ and $r_j \neq r_1$ for all $j \in \{2, 3, ..., n\}$, then $g_j = p_{max}$; otherwise let $g_j = r_j$.

Definition 2. Sub-problem QIP$_k$

For $\forall$ job j $\in$ N, its stage time g_j is employed to construct job groups, in each of which the jobs share the same stage time. For the k_{th} such job group N_k, scheduling jobs in it is defined as the

k_{th} sub-problem with the stage time of the sub-problem G_k. Then the k_{th} sub-problem can be illustrated as below:

$$QIP_k: \quad min \quad \sum_{j,j' \in N} \sum_{i \in M} x_{ij} w_j (p_{ij} + \sum_{j' \prec_i j} x_{ij'} p_{ij'} + G_k) \tag{1}$$

$$s.t. \quad \sum_{i \in M} x_{ij} = 1, \qquad j \in N_k \tag{2}$$

$$x_{ij} \in \{0,1\}. \qquad i \in M, j \in N_k \tag{3}$$

where x_{ij} equals 1 if job j is processed on machine i, and $x_{ij} = 0$ otherwise; According to the weighted SPT rule, the precedence relation $j' \prec_i j$ stands if $w_j/p_{ij} \leq w_{j'}/p_{ij'}$. The objective function (1) minimizes the total weighted completion time for all the jobs in N_k. Constraint (2) ensures that each job can be processed exactly once. Constraint (3) denotes the boolean decision variables.

Let K represent the number of unequal stage time of jobs, then the original problem can be decomposed into K sub-problems. And the stage time of the $k + 1_{th}$ sub-problem is larger than it is of the k_{th} sub-problem, for all $k \in \{1, 2, ..., K - 1\}$. 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.

Apparently, according to Definition 1, the jobs in the same sub-problem share an equivalent stage time. In other words, all the jobs in the k_{th} sub-problem would be released at G_k, even though the stage time 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 what follows, we'll try to prove that each sub-problem could be solved as an $Rm|| \sum w_j C_j$ problem.

Lemma 1. The scheduling problem $Rm|| \sum w_j C_j$ could be formulated as a QIP model as below, which is first introduced by Skutella (2001).

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

$$s.t. \quad \sum_{i \in M} x_{ij} = 1, \qquad j = 1, 2, ..., n$$

$$x_{ij} \in \{0,1\}, \qquad i = 1, 2, ..., m, j = 1, 2, ..., n$$

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 $w_j/p_{ij} \le w'_j/p_{ij'}$.

Since the formulation in Lemma 1 is well defined, and a 2-approximation scheme is available to handle the QIP model, if we could prove that the sub-problem QIP_k is equivalent to $Rm||\sum w_jC_j$ problem, then the solution procedure would be greatly simplified. In the following theorem, we'll try to fulfil that conjecture.

Theorem 2. Each sub-problem QIP_k decomposed from the original MIP model is equivalent to an $Rm||\sum w_jC_j$ problem.

Proof. It is observed that the sub-problem QIP_k denoted in Definition 2 has an objective function quite similar with the QIP model presented in Lemma 1. To fully exploit the structure feature, the objective function of QIP_k model is reformulated as follows,

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

where the value of $\sum_{j\in N} w_jG_k$ is equal to $\sum_{j\in N}\sum_{i\in M} x_{ij}w_jG_k$ because $\sum_{i\in M} x_{ij} = 1$ in constraint (2). Besides, it is found that the value of $\sum_{j\in N} w_jG_k$ does not depend on variables x_{ij}. Also, since the QIP_k model has exactly the same sets of constraints as the QIP model, thus the QIP model and QIP_k model should have the same solution space. Furthermore, given the same job group N_k, the sub-problem formulated as QIP_k will surely yield the same optimal solution as the QIP modeling of $Rm||\sum w_jC_j$. □

As it is pointed out by Skutella (2001), a semi-definite relaxation based method could be used to tackle $Rm||\sum w_jC_j$ problem, which is the equivalent counterpart of the k_{th} sub-problem QIP_k. However, in the sub-problem QIP_k illustrated in Definition 2, we cannot guarantee that all the jobs in one sub-problem would finish their processing before the next stage time. Consequently, the next sub-problem would not be a simple $Rm|G_{k+1}|\sum w_jC_j$ problem, if $\exists$ job h $\in N_k$ satisfies $G_k + p_{ih} > G_{k+1}$. Before addressing that issue, the detailed categorization of the jobs in one sub-problem based on the finishing time versus the next stage time is listed below,

Definition 3. Three job types in the k_{th} sub-problem, where $k \in \{1, 2, ..., K-1\}$.

1) Fixed job: The job which is completed before G_{k+1};

2) Straddling job: The job which has a stage time $g_j < G_{k+1}$ and will be finished after G_{k+1}.

3) Delayed job: The job which will start processing after G_{k+1}.

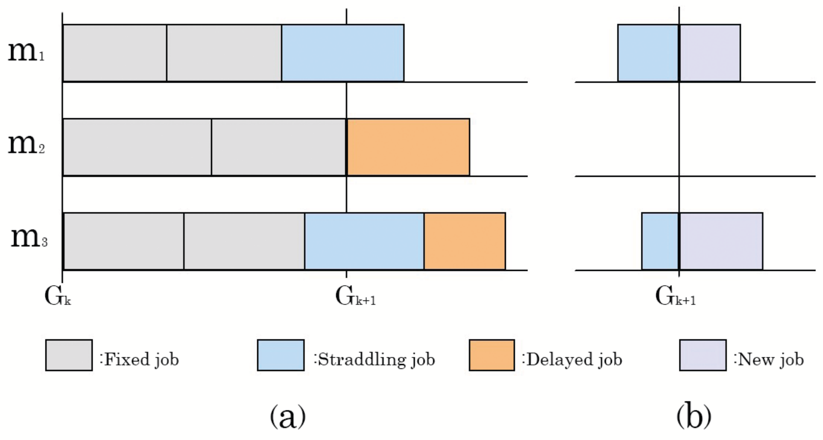


Figure 1: Illustration of the three job categories.

An illustration of these three kinds of jobs is shown in Fig.1(a). For the fixed and straddling jobs, their production plan 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 time has now shifted to G_{k+1}. Meanwhile, the engaged stages cannot be simply modeled as a $Rm|G_k|\sum w_j C_j$ or a $Rm|G_{k+1}|\sum w_j C_j$ 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 produced in one machine continuously without interruption. An illustration of the new jobs yielded by dividing straddling jobs is shown in Fig.1(b).

Lemma 2. For an arbitrary straddling job j which is produced on machine i in stage k initially, it is assumed that there is a new job j' which has the processing time $p_{lj'} = C_j - G_{k+1}$ if l = i, which represents that the new job j' remains on machine i in both stages k and k+1; Otherwise, if the new job divided from job j shifts to another machine, then we have $p_{lj'} = Q_1$. To fully demonstrate a new job j', the weight with respect to its completion time should be included. Here we assume the weight as $w_{lj'} = Q_2$ for all $l \in M$.

Moreover, we will show that under what conditions the straddling jobs and their respective new jobs will be produced in a continuous manner. In other words, these straddling jobs in the original scheduling problem would not be interrupted by the stage time given certain values of Q_1 and Q_2.

Theorem 3. For $\forall$ job $j \in$ straddling job set starting in stage k, if $Q_1 \geq C_j - G_{k+1}$, and $Q_2 \geq \min \frac{w_j}{p_j} \times (C_j - G_{k+1})$, then to interrupt the processing of job j at G_{k+1} would not be part of the optimal scheduling.

Proof. Assume that there is a straddling job j which is processed on machine i and has the stage time G_k. In $k + 1_{th}$ stage, the new job j' generated from job j has weight Q_2. Then the processing time of job j' would be $C_j - G_{k+1}$ if this job is continuously processed on machine i, otherwise, the processing time would be represented by Q_1. Sen and Bulbul (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 $Rm||\sum w_jC_j$ problem. Since we have the conditions $\frac{Q_2}{Q_1} \geq \min \frac{w_j}{p_j}$ and $Q_1 \geq C_j - G_{k+1}$ stand, which could easily deduce the following inequality, $\frac{Q2}{C_j - G_{k+1}} \geq \min \frac{w_j}{p_j}$. Thus according to the WSPT rule, the new job j' in the $k+1_{th}$ stage generated by its respective straddling job initiated in stage k would be scheduled as the earliest, which means that job j' could select any machine $i \in M$. Moreover, due to the processing time of the new job satisfies $Q_1 \geq C_j - G_{k+1}$, thus $p_{ij'} = C_j - G_{k+1}$ is the shortest possible processing time of the new job on any machine, which means that the straddling job must be processed on the same machine to guarantee the optimal scheduling. □

Based on Theorem 3, it is apparent that each sub-problem QIP_k can be solved as an $Rm||\sum w_jC_j$ problem with some new jobs generated from straddling jobs. In each sub-problem, the starting time of each job before the next stage time stays unchanged. Equivalently, we will not change the start time of fixed jobs and straddling jobs. Therefore, the solution of the original problem QIP 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 QIP 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 $Rm|r_j|\sum w_jC_j$ problem, in our algorithms, we consider dividing the original problem into several sub-problems. Then it is proved that each sub-problem can be solved as an $Rm||\sum w_jC_j$ problem in Theorem 2. Now we will focus on the sub-problem and try to find the exact solution of these sub-problems. As it is recalled, a quadratic integer programming (QIP) model for the sub-problem is demonstrated in Lemma 1. To tackle that QIP model, Skutella(2001) also proposed a convex quadratic method, which is formulated as follows,

$$
\begin{array}{rll}
CQIP: \quad min & \frac{1}{2}c^T x + \frac{1}{2}x^T D x & \\
s.t. & \sum_{i \in M} x_{ij} = 1, & j = 1,2,...,n \\
& x_{ij} \in \{0,1\}, & i = 1,2,...,m, j = 1,2,...,n
\end{array}
$$

where x_{ij} equals 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 $\prec_i$. 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:

$$
d_{(ij)(i'j')} = \begin{cases} w_j p_{ij} & \text{if } i = \text{i' and } j = j' \\ w_j p_{ij'} & \text{if } i = \text{i' and } j' \prec_i j \\ w'_j p_{ij} & \text{if } i = \text{i' and } j \prec_i j' \\ 0 & \text{if } i \neq \text{i'} \end{cases}
$$

It is worth noting that matrix D is a semi-definite matrix. Due to the quadratic objective function and 0-1 Boolean variables included in the above modeling, we devise an SDP relaxation formulation by first relaxing the 0-1 variables. 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 Briab (1999); On the other hand, the modeling power of SDP relaxation based methods allows us to naturally handle the quadratic objective function. As a result, the following SDQP model is proposed in the present paper.

$$
\begin{aligned}
SDQP: \quad & min \quad \frac{1}{2}cx + \frac{1}{2}D \cdot X && (4)\\
& s.t. \quad Ax = 1 && (5)\\
& \qquad X - xx^T \succeq 0 && (6)\\
& \qquad diag(X) = x && (7)\\
& \qquad X \in S_{mn\times mn} \quad x \in R^{mn} && (8)
\end{aligned}
$$

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

In the above CQIP model, the 0-1 constraints are replaced by constraint (6) in SDQP. Moreover, the Boolean variables $x_{ij} \in \{0, 1\}$ in CQIP 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 the constraints (7), $diag(X) = x$, which stands for the cutting plane of the feasible domain in SDQP model.

3.3. The two 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 Step 6 of this algorithm, we postpone the stage time of jobs which are released before p_{max} to avoid the plumbing of solution quality when the number of jobs is limited. Moreover, 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. However, the B&B algorithm in Step 9

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  for $j$=1 to $n$ do
3      If $r_j < p_{max}$ and $r_j \neq 0$, then set $g_j \leftarrow p_{max}$; otherwise, set $g_j \leftarrow r_j$.
4  Let $S_1 = s_1$ and $k \leftarrow 1$;
5  for $j$=2 to $n$ do
6      If $g_j == g_{j-1}$, then add job j to $k_{th}$ sub-problem and set $G_k \leftarrow g_j$; otherwise set
       $k \leftarrow k + 1$ then add job j to $k_{th}$ sub-problem.
7  Let $k_{max}$ be the number of sub-problems;
8  for $k = 1$ to $k_{max}$ do
9      Solve the SDQP model of $k_{th}$ sub-problem, and find the exact solution by branch
       and bound algorithm;
10     Include the scheduling plan of fixed jobs and straddling jobs into the overall result;
11     If there is a straddling job j processed on machine i, then set a new job with stage
       time $G_{k+1}$, which has the processing time $p_{lj} = C_j - G_{k+1}$ for l = i, $p_{lj} = Q_1$
       otherwise; and the weight is $w_{ij} = Q_2$ for all $i \in M$;
12     If there exists delayed jobs, let the stage times of these jobs be $G_{k+1}$;
13 Obtain the scheduling result by combining the scheduling plans of jobs in set P;
14 Calculate the sum of weighted completion times of the original jobs in set P.
```

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 $k_{th}$ sub-problem do
2      If $x_{ij}$ is maximal 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 value of the scheduling plan $\tilde{P}$;
5  Let the nodes which are not visited or fathomed be active nodes and set $V_{min} \leftarrow$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 model of its parental node;
10             Solve the SDQP 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.
```

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 tackle that problem, in the following we will introduce a local search based algorithm which is significantly faster, and the quality of solution is acceptable even thought it would not obtain the optimal solution in each sub-problem.

The FD-SDP Algorithm.

In D-SDP algorithm, the step of B&B method would increase the CPU time of D-SDP algorithm dramatically. So we would like to replace this B&B algorithm in Step 9 of D-SDP by a faster algorithm, which can guarantee high quality solutions of the sub-problems. To serve that purpose, we replace the B&B algorithm by a tailored local search (LS) algorithm to find the exact solutions of the sub-problems. This new approach is named FD-SDP algorithm, and it 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 $x_{i_{max}j} = 1$ if $x_{i_{max}j} > x_{i,j}, i \in m/i_{max}$;
  Otherwise, $x_{i_{max}j} = 0$. Let S be the exact solution $\tilde{P}$ by rounding the solution P;
2 Define that if two scheduling plans S and S' have k jobs processed on different machines,
  then the distance $\theta$ between the two solutions S and S' is $\theta(S, S') = k$. Search all of the
  solution space for scheduling plan S' which satisfies $\theta(S, S') = 1$, and find the best
  solution with the minimum total weighted completion time;
```

3.4.The performance analysis for D-SDP algorithm

In this part, 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, we can hardly provide a performance guarantee for the FD-SDP algorithm.

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: there is no straddling job in the scheduling plan of the k_{th} sub-problem, the illustration of which is shown in Fig.2(a).

Case 2: there are some straddling jobs in the scheduling plan of the k_{th} sub-problem.

Case 2.1: there is no delayed job in the scheduling plan of the k_{th} sub-problem, which is shown in Fig.2(b).

Case 2.2: there are some delayed jobs in the scheduling plan of the k_{th} sub-problem, which is demonstrated in Fig.2(c).

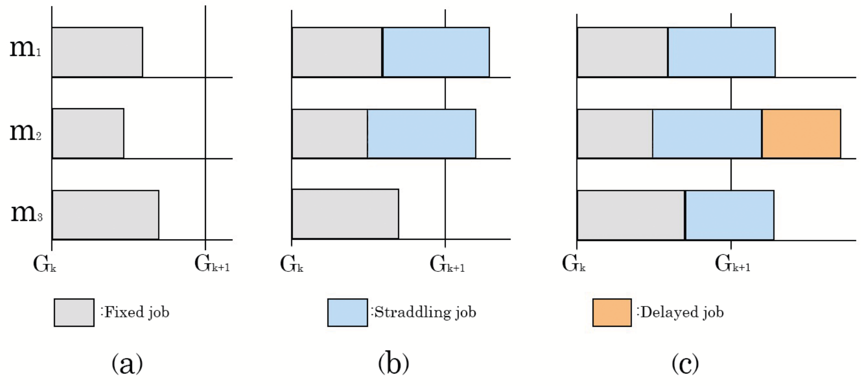


Figure 2: Illustrations of the three different scenarios in the k_{th} sub-problem

Theorem 4. If there is no straddling or delayed job in the sub-problem QIP_k of the original scheduling problem, then the result of the D-SDP algorithm for the $k + 1_{th}$ sub-problem is a part of the optimal scheduling plan.

Proof. Since there is no straddling or delayed job, the condition meets the requirement of Case 1 for the k_{th} sub-problem. And hence all jobs in the k_{th} stage can be finished before the next stage time G_{k+1}. As a result, the jobs processed in the k_{th} stage will not affect the completion times of the jobs in the $k + 1_{th}$ sub-problem. Then the $k + 1_{th}$ sub-problem can be viewed as a simple $Rm||\sum w_j C_j$ scheduling problem, which can be solved to optimal by the proposed D-SDP algorithm. $\square$

In Theorem 4, we have proved that D-SDP algorithm can be used to obtain the optimal solution in the special condition as demonstrated via Case 1. Then the approximation performance evaluation for D-SDP algorithm in Case 2.1 is provided based on Theorem 4. Let v_k^l and v^o, k$\in \{1, 2, ..., K\}$, be the values of the local optimum and global optimum for the k_{th} sub-problem, respectively. Apparently, the relationship between the two values is stated as $\sum v_k^l \leq v^o$. Furthermore, in the following theorem, we will show that in the k_{th} stage $v_k^D \leq (1+\epsilon_k)v_k^l$, where v_k^D is the value of the k_{th} sub-problem solved by D-SDP algorithm, and ϵ_k is a specifically constructed ratio. It should be pointed out that $v_1^D = v_1^l$, because no straddling job exists before the first stage. Moreover, we will prove that $v^D \leq (1+\max \epsilon_k)\mu v^o$, where v^D represents the value derived by the D-SDP algorithm for the original scheduling problem.

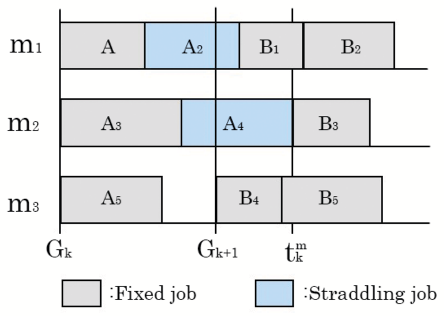


Figure 3: The sample scheduling plan derived by the D-SDP algorithm

Fig.3 depicts the scheduling plan obtained by D-SDP algorithm for an instance of Case 2.1, where jobs $A_1, ..., A_5$ have stage times G_k, jobs $B_1, ..., B_5$ have stage times G_{k+1}, and t_k^m is the makespan of the scheduling plan for the k_{th} sub-problem. Although the exact optimal solution to the original scheduling problem 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 $k+1_{th}$ sub-problem handled by D-SDP algorithm.

In the first place, the proposed lower bound for the optimal solution of the $k+1_{th}$ sub-problem is demonstrated in the following lemma, which by definition should be smaller than the optimal outcome of the original scheduling problem for jobs in the $k+1_{th}$ stage.

Lemma 3. Assume that all jobs in the $k+1_{th}$ stage can start processing after the release time r_{k+1}, regardless of the existence of straddling jobs in the k_{th} stage. Based on which the lower bound of the optimal $\sum w_j C_j$ value of the $k+1_{th}$ sub-problem is denoted as v_{k+1}^{lb}, and it means that $v_{k+1}^{lb} \leq v_{k+1}^l$. This situation is shown in Fig.4.

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 no straddling job is in the k_{th} stage. Moreover, in Theorem 4, it is demonstrated that the $k+1_{th}$ sub-problem can be solved to optimal by D-SDP algorithm in that scenario. Since the best scenario may not be attained in a general stage circumstance, once the influence of the straddling job has been exerted, the completion of the jobs in the corresponding stage would be further postponed. Therefore, $v^{lb}_{k+1} \leq v^{l}_{k+1}$ $\square$

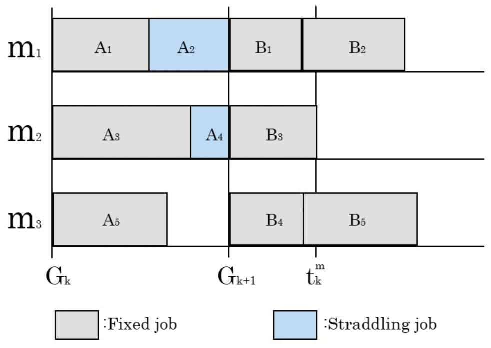


Figure 4: The sample scheduling plan according to the assumption in Lemma 3

Provided that the lower bound of the original scheduling problem has been proposed in Lemma 3, 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.

Lemma 4. Assume that all jobs in the $k+1_{th}$ stage can only start processing after the makespan t^{m}_{k} of the k_{th} stage. Let v^{ub}_{k+1} be denoted as the optimal $\sum w_j C_j$ value of the $k+1_{th}$ sub-problem based on this assumption. Thus v^{ub}_{k+1} is an upper bound of the optimal value of the original $k+1_{th}$ sub-problem, which means that $v^{D}_{k+1} \leq v^{ub}_{k+1}$. This situation is showed in Fig.5.

Proof. When the assumption stands, the best scenario is stated as all the jobs in the k_{th} stage sharing the same makespan t^{m}_{k}. However, in the general job group setting, not all of the jobs in the same stage complete at the same time. In other words, once a job in the k_{th} stage completes before t^{m}_{k}, then the optimal scheduling plan in the $k+1th$ stage would be to have a job started earlier than the k_{th} makespan. And hence the following inequality holds, $v^{ub}_{k+1} \geq v^{l}_{k+1}$. $\square$

In Lemma 3 and Lemma 4, both the lower bound and the upper bound for the optimal solution of the k_{th} sub-problem have been proposed. Moreover, according to Theorem 2, it could be concluded that the jobs in the optimal scheduling plan of Lemma 3 and Lemma 4 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 $\sum w_j C_j$ values of these

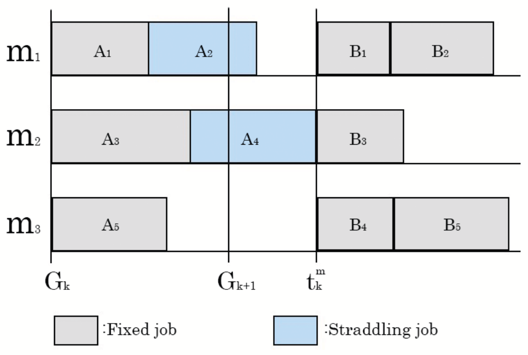


Figure 5: The sample scheduling plan according to the assumption in Lemma 4

two scheduling.

Lemma 5. Let v^{S_a} be the optimal $\sum w_j C_j$ value of a set of jobs in stage A which has the stage time S_a, and let v^{S_b} be the optimal $\sum w_j C_j$ value of the same set of jobs in stage B which has the stage time S_b, where $S_b \geq S_a$, then the inequality $v^{S_b} \leq \frac{S_b}{S_a} v^{S_a}$ holds.

Proof. According to Theorem 4, the same set of jobs in stage A should have the identical scheduling plan as they do in stage B. It means that for each job j, $\frac{C_j^b}{C_j^a} = \frac{s_b + p_{ij} + \tilde{p_{ij}}}{s_a + p_{ij} + \tilde{p_{ij}}} \leq \frac{s_b}{s_a}$, where C_j^a is the completion time of job j in stage A and C_j^b is the completion time of job j in stage B; $\tilde{p_{ij}}$ is the total processing time of jobs processed before job j on machine i in the same stage. Thus the relation $\frac{\sum w_j C_j^b}{\sum w_j C_j^a} \leq \frac{s_b}{s_a}$ holds, then we have $v^{s_b} \leq \frac{s_b}{s_a} v^{s_a}$. □

Based on Lemma 5, we can evaluate the approximation performance for the k_{th} sub-problem, where $k \in \{3, ..., K\}$. The 2_{nd} stage is analyzed separately because of the special construction method used in Definition 1. After including the influence of the 2_{nd} sub-problem, a worst case scenario analysis will be performed in Lemma 6.

Theorem 5. If there are only straddling jobs but no delayed jobs in the k_{th} sub-problem scheduling, then $v_{k+1}^D \leq \epsilon v_{k+1}^l$ for all $k \in \{2, 3, ..., K-1\}$, where $\epsilon = \frac{\max\{r_{k+1}, t_m^k\}}{r_{k+1}}$; t_k^m is the makespan of the scheduling plan for the k_{th} sub-problem.

Proof. As it is shown in Lemma 4, the relation $v_{k+1}^D \leq v_{k+1}^{ub}$ holds. According to Theorem 2,the jobs in the optimal scheduling plan as stated in Lemma 3 and Lemma 4 have the identical scheduling outcome. It means that each job in the optimal scheduling in Lemma 3 starts processing $\max\{r_{k+1}, t_k^m\} - r_{k+1}$ later than the job of the optimal scheduling plan in Lemma 4.

Based on Lemma 6, we can obtain that $v_{k+1}^{ub} \leq \frac{\max\{r_{k+1},t_k^m\}}{r_{k+1}} v_{k+1}^{lb}$. Moreover, Lemma 3 showed that $v_{k+1}^{lb} \leq v_{k+1}^{l}$. Therefore, $v_{k+1}^{D} \leq \frac{\max\{r_{k+1},t_k^m\}}{r_{k+1}} v_{k+1}^{lb} \leq \frac{\max\{r_{k+1},t_k^m\}}{r_{k+1}} v_{k+1}^{l}$. □

In Case 2.2, not only the straddling jobs exist, but also there is one or more delayed jobs in the subproblem, which is the most complicated condition of this scheduling problem. However, as it is demonstrated in the following lemma, Case 2.2 would be easily reduced to Case 2.1 by the assistance of D-SDP algorithm.

Lemma 6. In D-SDP algorithm, Case 2.2 is equivalent to Case 2.1.

Proof. Let J_k denote the set of jobs which are scheduled in the k_{th} sub-problem. Meanwhile, F_k, S_k, N_k and D_k represent the sets of fixed jobs, straddling jobs new jobs and delayed jobs in the scheduling plan of the k_{th} sub-problem, respectively. Apparently, the jobs in the sets F_k and S_k are considered in the k_{th} sub-problem, while the delayed jobs in set D_k will be switched to the $k+1_{th}$ sub-problem. Besides, the new jobs obtained by dividing the straddling jobs as defined in Lemma 2 will also be switched to the $k+1_{th}$ sub-problem. That is, the jobs in D_k and N_k will be reconsidered within the set $J_k' = J_k \cup D_k \cup N_k$ in the $k+1_{th}$ sub-problem. Consequently, there is no delayed job remained in the k_th sub-problem, and the new jobs will still be processed on the same machine as it is in the k_{th} sub-problem without preemptions. Hence, we could have the privilege to only consider the scheduling plan of fixed jobs and straddling jobs in each sub-problem, which renders Case 2.2 an equivalent situation to Case 2.1 in D-SDP algorithm. □

Lemma 7. Let v^o be the optimal objective value for the original problem and v^s be the value of the optimal solution with the execution of Step 1 to 3 in D-SDP, then $v^s \leq \mu v^o$ holds, where $\mu \leq 2$.

Proof. In Step 1 to 3 of D-SDP algorithm, let $r_j = p_{max}$ if $r_j \leq p_{max}$ and $r_j \neq 0$, where $p_{max} = \max p_{ij}$ for all $i \in M, j \in N$. Then we could conduct a worst case analysis to evaluate the approximation performance of this operation. For the worst case scenario, all the jobs in the second stage are released approaching time zero, and thus have to postpone their completion times by p_{max}. In that case, we have $v_{ub}^s = \sum w_j C_j + \sum w_j p_{max}$, where v_{ub}^s represents an upper bound of v^s. And $v^o = \sum w_j C_j$, indicating the optimal solution to the original scheduling problem. Since in the actual Step 1 to 3 operations, jobs released in the first stage would not be postponed by p_{max}, the optimal solution $v^s \leq v_{ub}^s$; Now suppose $\mu = \frac{\sum w_j C_j + p_{max} \sum w_j}{\sum w_j C_j}$, then the equation $v^s_{ub} = \mu v^o$ holds. 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 fractional expression would be reduced, which is equivalent to the value of v^s_{ub} decreases, then the inequality $v^s \leq v^s_{ub} \leq \mu v^o$ stands. Moreover, the inequality $\mu \leq 2$ holds, because $p_{max} \leq C_j$ for all $j \in N$ and $r_j \neq 0$; And when $r_j = 0$ we have $\mu = \frac{\sum w_j C_j}{\sum w_j C_j} = 1$. As a result the relation $\mu \leq 2$ stands in the proposed stage construction model. □

In Lemma 7, the most significant influences on the objective value exerted by the above postponement operation in Step 1 to 3 of D-SDP algorithm have been obtained via the worst case analysis. Based on the conclusions derived in Lemma 7, we can finally evaluate the approximation performance of the D-SDP algorithm as illustrated below.

Theorem 6. D-SDP is a 4-approximation algorithm.

Proof. In the $k+1_{th}$ sub-problem, the upper bound of the objective value of the scheduling plan for the $k+1_{th}$ sub-problem could be obtained by D-SDP algorithm, $v^D_{k+1} \leq \epsilon v^l_{k+1}$, where $\epsilon = \frac{\max\{r_{k+1}, t^m_k\}}{r_{k+1}}$, as it is illustrated in Theorem 5. Although the release time for the second stage is deliberately manipulated by Step 1 to 3 in D-SDP, we firstly assume that the operations will not affect the successive stages (such as 3_{rd} and above). Then we have the following,

$$v^D \leq \sum_{k \in \{1,2,\ldots,K-1\}} \{\frac{\max\{r_{k+1}, t^m_k\}}{r_{k+1}} v^l_k\}$$

Counting into the effects of Step 1 to 3 in D-SDP algorithm, the optimum after manipulating the second stage, v^s, will be larger than the sum of local optimum without considering the postponement influence, $\sum v^l_k$.

$$\sum_{k \in \{1,2,\ldots,K-1\}} \{\frac{\max\{r_{k+1}, t^m_k\}}{r_{k+1}} v^l_k\} \leq \max_{k \in \{1,2,\ldots,K-1\}} \{\frac{\max\{r_{k+1}, t^m_k\}}{r_{k+1}}\} v^s$$

Moreover, since Lemma 7 has showed that $v^s \leq \mu v^o$ for the relationship between the optimal solution with/without considering Step 1 to 3 in D-SDP algorithm. Thus the following inequality holds,

$$\max_{k \in \{1,2,\ldots,K-1\}} \{\frac{\max\{r_{k+1}, t^m_k\}}{r_{k+1}}\} v^s \leq \max_{k \in \{1,2,\ldots,K-1\}} \{\frac{\max\{r_{k+1}, t^m_k\}}{r_{k+1}}\} \mu v^o$$

Finally, it should be pointed out that for each sub-problem k, $t^m_k \leq r_{k+1} + p_{max}$, then $\frac{\max\{r_{k+1}, t^m_k\}}{r_{k+1}} \leq 2$. In addition, from Lemma 7 we know $\mu \leq 2$, then the conclusion $v^D \leq 4v^o$ is obtained. □

Actually, for the more general situation, the value of ϵ would approach the average value of $\epsilon_{\tilde{k}}$ of the last several stages; And μ would get close to 1 when the number of jobs becomes large enough. Without loss of generality, let us consider such a special condition where the same number and characteristics of job exist in 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 powerful ϵ_k will be for ϵ. Moreover, when the value of r_k is large enough, ϵ_k will be close to 1, which means that the D-SDP performance will be enhanced with the increase of job quantity. Furthermore, the computation results in the next section will also validate that $\epsilon < 0.05$ 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 other approximation algorithms such as the one proposed by Hall et al.(1997).

4. Computational results

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%
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

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) would be solved by CPLEX 12.6. We use the Corollary 1 to further reduce the CPU time for solving the MIP model. All other algorithms would be implemented by MATLAB R2016a. The experiments are performed on a PC with a 2.3-GHz CPU and 8 GB memory.

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 $P = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}}{m}$.

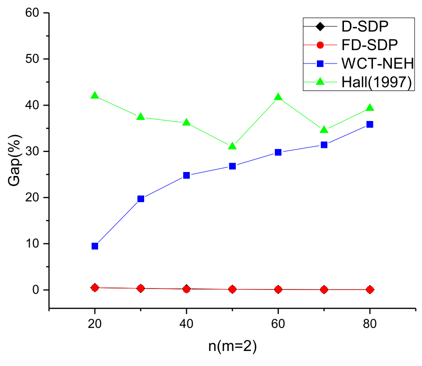
Table 2: The performance of DP-SDP, QDP-SDP and WCT-NEH for m=4,5

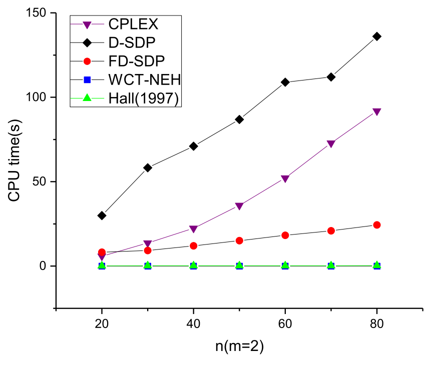
m	n	CPLEX	D-SDP		FD-SDP		WCT-NEH		Hall (1997)	
		CPU time	CPU time	Gap1	CPU time	Gap2	CPU time	Gap3	CPU time	Gap4
4	20	12.55	57.54	1.30%	7.10	0.48%	0.01	0.89%	0.25	28.51%
	30	27.46	86.00	0.75%	11.01	0.25%	0.01	1.86%	0.27	41.53%
	40	50.43	104.34	0.50%	13.98	0.12%	0.02	7.73%	0.32	41.81%
	50	76.35	134.73	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
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

Let V(algorithm) represent the value of the solution of an algorithm. In small-medium instances, Gap1 is defined as $\frac{\text{V(D-SDP)-V(CPLEX)}}{\text{V(CPLEX)}} \times 100\%$, Gap2 is defined as $\frac{\text{V(FD-SDP)-V(CPLEX)}}{\text{V(CPLEX)}} \times 100\%$, the definition of Gap3 is $\frac{\text{V(WCT-NEH)-V(CPLEX)}}{\text{V(CPLEX)}} \times 100\%$. Gap2 is defined as $\frac{\text{V(CPLEX)-V(Hall(1997))}}{\text{V(CPLEX)}} \times 100\%$. We record the gap by NaN if CPLEX can't deliver the result in a reasonable period of time. Due to the curse of dimensionality, for large instances, CPLEX cannot obtain the optimal outcome most of the time.

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

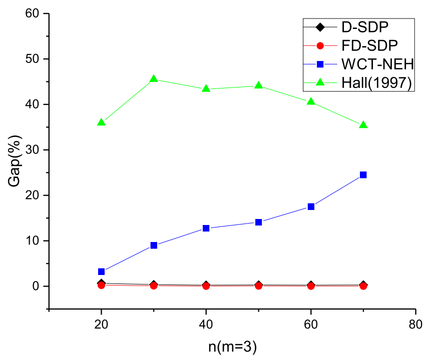
In Table 1 and 2, the averaged results over 5 trials for medium instances are reported. Each problem is solved by CPLEX, D-SDP, FD-SDP, WCT-NEH proposed by Lin and Lin(2013), and the approximation algorithm provided by Hall et al. (1997). It is observed that the CPU time of CPLEX based algorithm would increase rapidly with the number of jobs, machines and the value of T. 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.

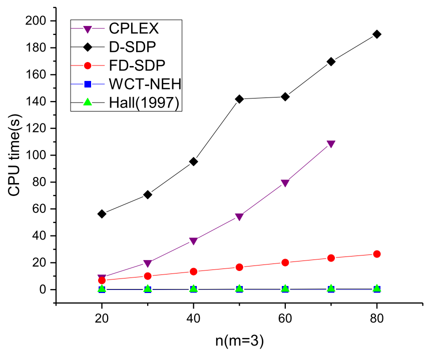




(a) Gap (b) CPU time

Figure 3. The performance for m=2.

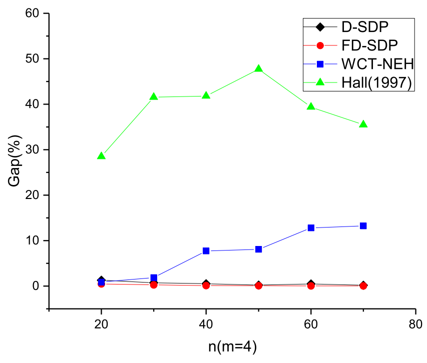




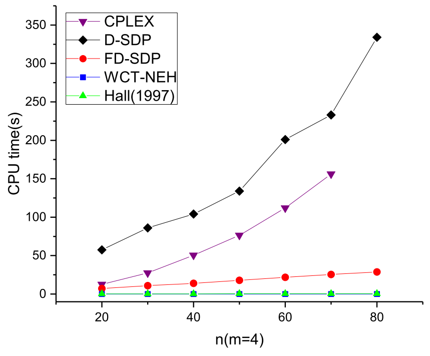
(a) Gap (b) CPU time

Figure 4. The performance for m=3.

For each problem instance, Gap1 and Gap2 are usually less the 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.

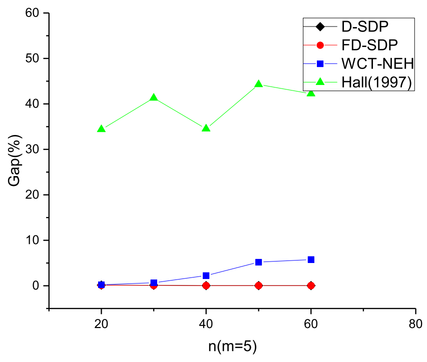


(a) Gap

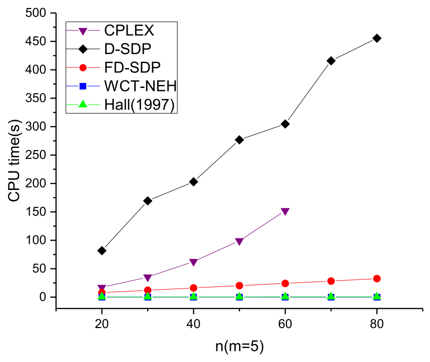


(b) CPU time

Figure 5. The performance for m=4.



(a) Gap



(b) CPU time

Figure 6. The performance for m=5.

Figure 3,4,5 and 6 show the gaps and CPU times for small-medium instances on 2,3,4 and 5 unrelated machine, 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. To our relief, the proposed D-SDP and FD-SDP algorithms can always obtain a high quality solution for most cases, 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. If we dig deeper, the following discoveries may be worthwhile to notice. 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 search method. However, since the aim of solving each sub-problem is to find the scheduling with minimized sum of weighted completion time instead of the makespan. Sometimes, local search can obtain an ideal makespan which means that the jobs of its successive stage will start processing earlier if there are straddling jobs in the current stage. And hence the successive sub-problem would have a smaller sum of weighted completion time.

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%

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 of

Hall (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 contiuously 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.

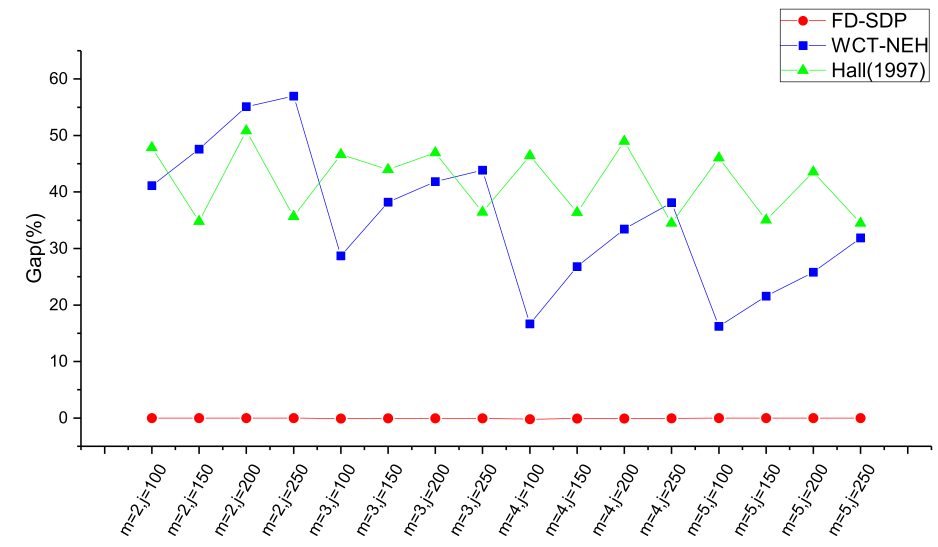


Figure 7. The average relative gap for large size instances.

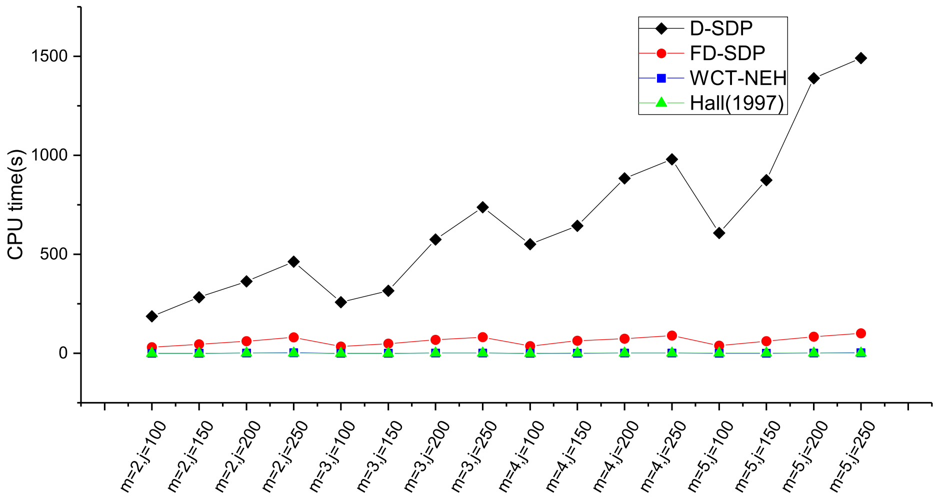


Figure 8. The average CPU time for large size instances.

Figure 7 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 Figure

8, 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. And FD-SDP algorithm can solve these scheduling problems within a satisfactory time range.

5. Conclusion

In this research, we consider the unrelated parallel machines scheduling with release dates to minimize the sum of weighted completion time. This $Rm|r_j|\sum w_jC_j$ problem is divided into a series of $Rm||\sum w_jC_j$ sub-problems based on their release dates. The QIP model is considered for each sub-problem. Moreover, we give an SDP relaxation for QIP model, then we find the exact approximation solution by B&B method or LS method. The solution of the primal problem can be constituted by each solution of sub-problem. It is proved that the D-SDP algorithm is a 4-approximation algorithm promoting the existing 16/3-approximation given by Hall et al. In the computational results section, several different sizes of instances are used to test the performance of our algorithms in comparison with the outcomes of CPLEX, the existing approximation algorithms, and WCT-NEH algorithm. Furthermore, during the numerical case study, it is observed that D-SDP and FD-SDP algorithms usually could provide high quality solutions with smaller gaps, while the computational time of FD-SDP is comparable to it of WCT-NEH and the approximation algorithm of Hall, which may incur relatively larger gaps in return.

Acknowledgements

The authors are grateful for the financial support partly from the NSF Project (No. 51305400) of China, and from the NSF Projects (No. LQ12G01008, and LY18G010017) of Zhejiang Province.

References

[1] 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.

[2] Azizoglu,M.& Kirca,O.(1999b). Scheduling jobs on unrelated parallel machines to minimize regular total cost functions, IIE Transactions,31(2),153-159.

[3] Afrati,F.,Bampis,E.,Fishkin,AV. et al.(2000).Scheduling to minimize the average completion time of dedicated tasks. In Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science, FST TCS 2000, Lecture Notes in Computer Science. Springer Berlin Heidelberg: London, UK, 2000; 454C464.

[4] Afrati, F., Bampis, E., Sviridenko, M., Chekuri, C., Karger, D., & Kenyon, C., et al. (1999). Approximation schemes for minimizing average weighted completion time with release dates. Proceedings of Annual IEEE Symposium on Foundations of Computerence, 32-43.

[5] Avdeenko,T.V.; Mesentsev,Y.A.& Estraykh,I.V.(2017) Approximation algorithms for scheduling unrelated parallel machines with release dates,International Conference on Information Technologies in Business and Industry. Martyushev, N; Avramchuk, V& Faerman, V eds.,803.

[6] Balas,E.,Ceria,S.,Cornuejols,G.& Pataki,G.(1994). Updated semi- definite constraints. Mathematical Programming,58,295-324.

[7] Brian,B.(1999). CSDP, A C library for semidefinite programming. Optimization Methods and Software,11,613-623.

[8] Chen,ZL.& Powell,W.B.(1999). Solving parallel machine scheduling problem by column generation. INFORMS Journal on Computing,11(1),78-94.

[9] Chen,JF.(2015) Unrelated parallel-machine scheduling to minimize total weighted completion time. Journal of Intelligent Manufacturing,26(6),1099-1112.

[10] Cheng,CY.& Huang,LW.(2017) Minimizing total earliness and tardiness through unrelated parallel machine scheduling using distributed release time control. Journal of Manufacturing Systems,42,1-10.

[11] Durasevic,M. Jakobovic,D.& Knezevic,K.(2016) Adaptive scheduling on unrelated machines with genetic programming. Applied Soft Computing,48,419-430.

[12] Hall,LA.,Schulz,AS.,Shmoys,DB.& Wein,J.(1997).Scheduling to minimize average completion time: Off-line and on-line approximation algorithms,Mathematics of Operations Research,22(3),513-544.

[13] Li,K.& Shan,L.(2009).Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. Applied Mathematical Modelling,33(4),2145-2158.

[14] Lin,YK.& Lin,CW.(2013).Dispatching reles for unrelated parallel machine scheduling with release dates.International Journal of Advanced Manufacturing Technology,67,269-279.

[15] Lenstra,J.K.,Rinnooy Kan,A.H.G.,& Brucker,P.(1977). Complexity of Machine Scheduling Problem, Annals of Discrete Mathematics,1:343-262.

[16] 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.

[17] Lovase,L. & Schrijver,A.(1991). Cones of matrices and set functions and 0-1 optimization. SIAM Journal on Optimization,1(2),166-190.

[18] 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.

[19] Nessah,R.,Yalaoui,F.,&Chu,C.(2008). A branch-and-bound algorithm to minimize total weight completion time on identical parallel machines with job release dates. Computers & Operations Research,35(4),1176-1190.

[20] Rodriguez,F.J.,Lozano,M.,Blum,C.&Garcia-Martinez,C.(2013).An iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. Computers & Operations Research,40(7),1829-1841.

[21] Rendl,F.(2016).Semidefinite relaxations for partitioning, assignment and ordering problems. Annals of Operations Research,240(1),119-140.

[22] Sen,H.& BulBul,K.(2017). An exact extended formulation for the unrelated parallel machine total weighted completion time problem, Journal of Scheduling,20(4),373-389.

[23] Sherali,H.D. &Adams,W.P.(1994). A hierarchy of relaxations and convex hull characterizations for mixed-interger zero-one programming problems. Discrete applied Mathematics,52(1),83-106.

[24] Skutella,M.(2001).Convex quadratic and semidefinite programming relaxations in scheduling, Journal of The ACM,48:206-242.

[25] Smith, W.E. (1956). Various optimizers for single-stage production. Naval Research Logisitics Quarterly,3:59-66.

[26] Schulz,A.S.& Skutlla,M. (2001). Scheduling unrelated machines by randomized rounding. SIAM Journal on discrete mathematics,15(4),450-469.

[27] Skutella,M.; Sviridenko,M.& Uetz,M.(2016) Unrelated machine scheduling with stochastic processing times. Mathematics of Operations Research,41(3),851-864.

[28] Tang,LX.& Zhang,YY.(2011).A new Largrangian relaxation algorithm for scheduling dissimilar parallel machines with release dates. International Journal of Systems science,11(4),1133-1141.

[29] Unlu,Y.& Mason,S.J.(2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems.Computers & Industrail Engineering,58(4),785-800.

[30] Vredeveld,T.& Hurkens,C.(2002).Experimental comparison of approximation algorithms for scheduling unrelated parallel machines. INFORMS Journal on Computing,14(2),175-189.

[31] Van Den Akker,J.M.,Hoogeveen,J.A.,&Van De Velde,S.L.(1999).Parallel machine scheduling by colcumn generation. Operations Research,47(6),862-872.

[32] Yalaoui,F.& Chu,C. (2006) New exact method to solve the Pm/r(j)/Sigma C-j schedule problem. International Journal of Production Economics,100(1),168-179.

[33] Zhang,XZ.; Xu DC.& Du DL.(2016) Approximate algorithms for unrelated machine scheduling to minimize makespan. Joural of Industrial and Management Optimization,12(2),771-779.