Journal of Scheduling (2022) 25:577–587
https://doi.org/10.1007/s10951-022-00743-9

On the tractability of hard scheduling problems with generalized due-dates with respect to the number of different due-dates

Gur Mosheiov[1] · Daniel Oron[2] · Dvir Shabtay[3]

Accepted: 8 April 2022 / Published online: 25 June 2022

Abstract

We study two $\mathcal{NP}$-hard single-machine scheduling problems with generalized due-dates. In such problems, due-dates are associated with positions in the job sequence rather than with jobs. Accordingly, the job that is assigned to position j in the job processing order (job sequence), is assigned with a predefined due-date, δ_j. In the first problem, the objective consists of finding a job schedule that minimizes the maximal absolute lateness, while in the second problem, we aim to maximize the weighted number of jobs completed exactly at their due-date. Both problems are known to be strongly $\mathcal{NP}$-hard when the instance includes an arbitrary number of different due-dates. Our objective is to study the tractability of both problems with respect to the number of different due-dates in the instance, ν_d. We show that both problems remain $\mathcal{NP}$-hard even when $\nu_d = 2$, and are solvable in pseudo-polynomial time when the value of ν_d is upper bounded by a constant. To complement our results, we show that both problems are fixed parameterized tractable (FPT) when we combine the two parameters of number of different due-dates (ν_d) and number of different processing times (ν_p).

Keywords Scheduling · Single machine · Generalized due-dates · $\mathcal{NP}$-hard · Pseudo-polynomial time algorithm · Parameterized complexity.

1 Introduction

In most classical scheduling problems involving due-date related performance measures, the due-dates are given as a set of predefined job-related parameters, i.e., each job has its own predefined due-date given by the instance. When scheduling with generalized due-dates (*gdd*'s), the due-date of each job is defined only after the scheduling decisions are made. Accordingly, due-dates are associated with positions in the job processing order, and each job is assigned with a due-date based on its position in this order. Yin et al. (2012) pointed out that scheduling with *gdd* arises in cases where there are milestones in a serial project, each indicating the number of operations that are required to be completed up to a certain point in time. Browne et al. (1984), Hall (1986) and Stecke and Solberg (1981) describe situations in which generalized due-dates arise in practical settings, including in public utility planning, survey design and flexible manufacturing.

✉ Daniel Oron
daniel.oron@sydney.edu.au

Gur Mosheiov
msomer@huji.ac.il

Dvir Shabtay
dvirs@bgu.ac.il

1 School of Business Administration, The Hebrew University, Jerusalem, Israel

2 The University of Sydney Business School, 2006 NSW, Australia

3 Department of Industrial Engineering and Management , Ben-Gurion University of the Negev, Beer-Sheva, Israel

1.1 Literature review and problem definition

The field of scheduling with *gdd* was first introduced by Hall (1986) who analyzed several scheduling problems with *gdd* on a single machine and on identical parallel machines. He showed that some problems that are solvable in polynomial time for the case of job-related due-dates remain so when generalized due-dates are considered. This includes the problems of the minimizing maximum lateness and the number of tardy jobs on a single machine. He also showed, however, that for some other problems the complexity status changes. For example, Hall showed that although the problem of minimizing the total tardiness on a single machine is $\mathcal{NP}$-hard when due-dates are job-related (see Du & Leung, 1990), it is solvable in polynomial time with *gdd*. Other results on scheduling under the assumption of *gdd* appear in Sriskandarajah (1990), Hall et al. (1991), Tanaka and Vlach (1999), Gao and Yuan (2006), Yin et al. (2012) and Gerstl and Mosheiov (2020) .

In many cases where a scheduling problem with *gdd* is found to be $\mathcal{NP}$-hard, the reduction is done to an instance that includes an arbitrary number of different due-dates, ν_d (see, e.g., Tanaka & Vlach, 1999; Gerstl & Mosheiov, 2020). However, in real-life applications, especially in cases where delivery costs are very high, the value of ν_d may be of limited size. Therefore, it is interesting to investigate whether or not the problems become tractable for bounded values of ν_d. We focus on two such problems, where the scheduling criterion is non-regular and follows the concept of just in time (*JIT*). The concept of *JIT* is used whenever both job earliness and tardiness should be avoided, i.e., when it is desirable to complete a job's processing at, or as close as possible, to its due-date. In the first problem, our aim is to find a schedule that minimizes the maximum absolute lateness, while in the second problem we seek a schedule that maximizes the weighted number of jobs completed exactly at their due-dates.

The two problems we consider in this paper are defined as follows. We are given a set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ of n jobs to be scheduled non-preemptively on a single machine. Let p_j be the processing time of job J_j on the single machine. A feasible job schedule S is defined by (i) a processing permutation $\pi = \{J_{[1]}, J_{[2]}, \ldots, J_{[n]}\}$ of the n jobs on the single machine, where $[j]$ is the index of the job in the jth position in π, and by (ii) a feasible set of processing intervals, $(S_{[j]}, C_{[j]} = S_{[j]} + p_{[j]}]$ for $j = 1, \ldots, n$, satisfying that $S_{[j]} \geq C_{[j-1]} = S_{[j-1]} + p_{[j-1]}$ for $j = 1, \ldots, n$, where $S_{[j]}$ and $C_{[j]}$ are the start time and the completion time of job $J_{[j]}$, respectively, and $S_{[0]} = p_{[0]} = 0$ by definition. Given S, the due-date assigned to job $J_{[j]}$ is δ_j. Accordingly, the lateness of job $J_{[j]}$ is defined by $L_{[j]} = C_{[j]} - \delta_j$. Moreover, we say that job $J_{[j]}$ is a *JIT* job if $C_{[j]} = \delta_j$, and by $\mathcal{E} = \{J_j \in \mathcal{J} \,|C_{[j]} = \delta_j\}$, we denote the set of *JIT* jobs.

In the first problem, our objective is to find a feasible schedule that minimizes the maximum absolute lateness, given by $\max_{J_j \in \mathcal{J}}\{|L_j|\}$, while in the second problem we aim to find a solution that maximizes the weighted number of *JIT* jobs, given by $\Sigma_{J_j \in \mathcal{E}} w_j$, where w_j is the weight of job J_j representing the gain obtained from completing job J_j in a *JIT* mode. Using the classical three-field notation for scheduling problems (see Graham et al., 1979), we denote the first problem we study by $1\,|gdd|\max_{J_j \in \mathcal{J}}\{|L_j|\}$ and the second problem by $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$. We note that $|L_{[j]}| = \max\{E_{[j]}, T_{[j]}\}$, where $E_{[j]} = \max\{0, \delta_j - C_{[j]}\}$ is the earliness of job $J_{[j]}$, and $T_{[j]} = \max\{0, C_{[j]} - \delta_j\}$ is the tardiness of job $J_{[j]}$. By $1\,||\max_{J_j \in \mathcal{J}}\{|L_j|\}$ and $1\,||\,\Sigma_{J_j \in \mathcal{E}} w_j$, we refer to the variant of the same problems where due-dates are job-related.

When due-dates are job-related, the resulting $1\,||\max_{J_j \in \mathcal{J}}\{|L_j|\}$ and $1\,||\,\Sigma_{J_j \in \mathcal{E}} w_j$ problems are solvable in polynomial time (see Garey et al., 1988 and Lann and Mosheiov, 1996). However, both problems are strongly $\mathcal{NP}$-hard with generalized due-dates (see Tanaka & Vlach, 1999 and Gerstl & Mosheiov, 2020), and for the $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ problem, the strongly $\mathcal{NP}$-hardness result holds even if all weights are identical (i.e., even when the objective is simply to maximize the number of *JIT* jobs, $|\mathcal{E}|$).

1.2 Research objectives and paper organization

The reductions used by Tanaka and Vlach (1999) and Gerstl and Mosheiov (2020) to prove that problems $1\,|gdd|\max_{J_j \in \mathcal{J}}\{|L_j|\}$ and $1\,|gdd|\,|\mathcal{E}|$ are strongly $\mathcal{NP}$-hard are done by constructing instances that include an arbitrary number of different due-dates. We aim to see whether the problems become easier to solve when the number of different due-dates in the instance is of a limited size. Our analysis is done from both a classical (see Garey & Johnson, 1979) and a parameterized (see, e.g., Downey, 1999 & Niedermeier, 2006) complexity point of view. To do so, let υ_d be the number of different due-dates in the instance. From a classical complexity perspective, we aim to determine whether or not the problems are solvable in polynomial time when υ_d is upper bounded by a constant. If not, we aim to determine whether the problem remains $\mathcal{NP}$-hard in the strong sense, or it is solvable in pseudo-polynomial time. We also aim to determine the complexity of each problem with respect to (wrt.) υ_d in the sense of parameterized complexity.

Given an $\mathcal{NP}$-hard problem and a parameter k, in parameterized complexity we aim to determine whether the problem has an algorithm running in $f(k)n^{O(1)}$ time, where $f(k)$ is a function that depends solely on k (and thus independent of the number of jobs n). Such an algorithm is referred to as a *Fixed Parameter Tractable* (*FPT*) algorithm wrt. k. Note that an FPT algorithm is always faster than an $n^{f(k)}$ algorithm, for any monotone increasing function $f(k)$, when n and k are sufficiently large. Thus, for example, an *FPT* algorithm is preferable over an algorithm that is polynomial whenever k is constant.

In Sections 2 and 3, we analyze the tractability of problems $1\,|gdd|\max_{J_j \in \mathcal{J}}\{|L_j|\}$ and $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ wrt. ν_d. We prove that problems $1\,|gdd|\max_{J_j \in \mathcal{J}}\{|L_j|\}$ and $1\,|gdd|\,|\mathcal{E}|$ are both $\mathcal{NP}$-hard even if $\nu_d = 2$. Unless P=$\mathcal{NP}$, this result rules out the possibility to construct *FPT* algorithms for problems $1\,|gdd|\max_{J_j \in \mathcal{J}}\{|L_j|\}$ and $1\,|gdd|\,|\mathcal{E}|$ wrt. ν_d. We also provide pseudo-polynomial time algorithms to solve problems $1\,|gdd|\max_{J_j \in \mathcal{J}}\{|L_j|\}$ and $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ when ν_d is upper bounded by a constant. These results lead to the conclusion that both problems are strongly $\mathcal{NP}$-hard only for an arbitrary value of ν_d, while they both become ordinary $\mathcal{NP}$-hard when ν_d is upper bounded by a constant. As both problems do not admit an *FPT* algorithm wrt. ν_d, we further analyze the case where we combine parameter ν_d with another parameter, ν_p, which represents the number of different processing times in the instance. We provide *FPT* algorithms for problems $1\,|gdd|\max_{J_j\in\mathcal{J}}\{|L_j|\}$ and $1\,|gdd|\,|\mathcal{E}|$ wrt. the combined parameter. A summary and future research section concludes our paper.

2 Maximal absolute lateness

The $1\,|gdd|\max\{|L_j|\}$ problem is strongly $\mathcal{NP}$-hard when due-dates are arbitrary (see Tanaka & Vlach, 1999). When there is a common due-date for all jobs (i.e., $\nu_d = 1$), a simple $O(n)$ time algorithm provides the optimal schedule for the corresponding $1\,|gdd, \delta_j = \delta|\max\{|L_j|\}$ problem (see Cheng, 1987). In the following three subsections, we complement these two results by showing that the $1\,|gdd|\max\{|L_j|\}$ problem is (i) $\mathcal{NP}$-hard even when we have only two distinct due-dates in the instance; (ii) solvable in pseudo-polynomial time when the value of ν_d is upper bounded by a constant; and (iii) fixed parameterized tractable (FPT) when we combine the two parameters consisting of the number of different due-dates (ν_d) and number of different processing times (ν_p).

2.1 $\mathcal{NP}$-hardness for the two distinct due-dates case

In this subsection, we show that the $1\,|gdd|\max\{|L_j|\}$ is $\mathcal{NP}$-hard even when $\nu_d = 2$. The reduction is done from the ordinary $\mathcal{NP}$-hard Equal Size Partition problem, which is defined below.

Definition 1 Equal Size Partition: Given a set of $2t$ positive integers $\mathcal{A} = \{a_1, a_2, \ldots, a_{2t}\}$ with $\Sigma_{j=1}^{2t} a_j = 2B$ and $0 < a_j < B$. Can $\mathcal{A}$ be partitioned into two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\Sigma_{a_j\in\mathcal{A}_i} a_j = B$ and $|\mathcal{A}_i| = t$ for $i = 1, 2$?

Theorem 1 *The $1\,|gdd|\max\{|L_j|\}$ is $\mathcal{NP}$-hard even when there are only two distinct due-dates (i.e., even when $\nu_d = 2$).*

Proof Given an instance for the $\mathcal{NP}$-hard Equal Size Partition problem, we construct the following instance for our $1\,|gdd|\max\{|L_j|\}$ problem: The instance includes $n = 2t + 2$ jobs. The processing times are:

$$p_j = \begin{cases} a_j \text{ for } & j = 1, \ldots, 2t \\ 2B \text{ for } & j = 2t + 1, 2t + 2 \end{cases} \tag{1}$$

and the due-dates are

$$\delta_j = \begin{cases} \underline{\delta} = 2.5B \text{ for } & j = 1, \ldots, t + 1 \\ \overline{\delta} = 5.5B \text{ for } & j = t + 2, \ldots 2t + 2 \end{cases}. \tag{2}$$

In the decision version of the problem, we ask whether there exists a feasible schedule with $\max\{|L_j|\} \le B/2$. Note that there are only two distinct due-dates in the constructed instance of the $1\,|gdd|\max\{|L_j|\}$ problem.

Given a solution that provides a *YES* answer for the Equal Size Partition, we construct the following solution S to our $1\,|gdd|\max\{|L_j|\}$ problem. We set $\mathcal{J}_i = \{J_j\,|a_j \in \mathcal{A}_i\}$ for $i = 1, 2$. Then, we schedule the jobs in the following order: $J_{2t+1}, \mathcal{J}_1, J_{2t+2}, \mathcal{J}_2$ with no machine idle times. The processing order in each $\mathcal{J}_i$ $(i = 1, 2)$ is arbitrary. Since $|\mathcal{J}_i| = |\mathcal{A}_i| = t$, all jobs in $J_{2t+1} \cup \mathcal{J}_1$ share the same due-date of $\underline{\delta} = 2.5B$, and all jobs in $J_{2t+2} \cup \mathcal{J}_2$ share the same due-date of $\overline{\delta} = 5.5B$. Therefore, in S:

- Job J_{2t+1} is scheduled during time interval $(0, 2B]$. Thus, $|L_{2t+1}| = |2B - 2.5B| = 0.5B$.
- Job set $\mathcal{J}_1$ is scheduled during time interval $(2B, 3B]$. Therefore, $\max_{J_j\in\mathcal{J}_1}\{|L_j|\} = \max\{|2B + p_{[2]} - 2.5B|, 3B - 2.5B\}$, where $[j]$ is the index of the jth job in the processing order. The fact that $p_{[2]} = a_{[2]} < B$ implies that $\max_{J_j\in\mathcal{J}_1}\{|L_j|\} = 3B - 2.5B = 0.5B$.
- Job J_{2t+2} is scheduled during time interval $(3B, 5B]$. Thus, $|L_{2t+2}| = |5B - 5.5B| = 0.5B$.
- Job set $\mathcal{J}_2$ is scheduled during time interval $(5B, 6B]$. Therefore, $\max_{J_j\in\mathcal{J}_2} |L_j| = \max\{|5B + p_{[t+3]} - 5.5B|, 6B - 5.5B\}$. The fact that $p_{[t+3]} = a_{[t+3]} < B$ implies that $\max_{J_j\in\mathcal{J}_2}\{|L_j|\} = 6B - 5.5B = 0.5B$.

It follows that in schedule S, $\max_{J_j\in\mathcal{J}}\{|L_j|\} = 0.5B$, and we have a *YES* answer for the constructed instance of our $1\,|gdd|\max\{|L_j|\}$ problem.

Consider next a solution S that provides a *YES* answer for the constructed instance of our $1\,|gdd|\max\{|L_j|\}$ problem.

Lemma 1 *In S, there are no machine idle times.*

Proof If S includes machine idle times of a total length of $\Delta > 0$, then the last job is completed at time $6B + \Delta$. As the last scheduled job is assigned a due-date of $\overline{\delta} = 5.5B$, its absolute lateness equals to $0.5B + \Delta > 0.5B$, contradicting the fact that S provides a *YES* answer for the $1\,|gdd|\max\{|L_j|\}$ problem. □

Now, let $\mathcal{J}_1$ be the set of $t + 1$ jobs that are scheduled first in S, and let $\mathcal{J}_2$ be the set of $t + 1$ jobs that are scheduled last in S. All jobs in $\mathcal{J}_1$ share the same due-date of $\underline{\delta} = 2.5B$, and all jobs in $\mathcal{J}_2$ share the same due-date of $\overline{\delta} = 5.5B$.

Lemma 2 *Set $\mathcal{J}_1$ includes exactly a single job out of the pair $\{J_{2t+1}, J_{2t+2}\}$.*

Proof By contradiction, assume that set $\mathcal{J}_1$ includes either none or both jobs in $\{J_{2t+1}, J_{2t+2}\}$. If $\mathcal{J}_1$ includes none of the two jobs in $\{J_{2t+1}, J_{2t+2}\}$, then based on Lemma 1 the first job to be schedule will complete at time $C_{[1]} = a_{[1]} < B$. Then, $|L_{[1]}| = |C_{[1]} - \delta_1| > |B - 2.5B| = 1.5B$, contradicting the fact that S provides a *YES* answer for the $1\,|gdd|\max\{|L_j|\}$ problem. If $\mathcal{J}_1$ includes both $\{J_{2t+1}, J_{2t+2}\}$, the completion of the last job in $\mathcal{J}_1$ will be at time $C_{[t+1]} > p_{2t+1} + p_{2t+2} = 4B$. Therefore, $|L_{[t+1]}| = |C_{[t+1]} - \delta_{t+1}| > |4B - 2.5B| = 1.5B$, contradicting the fact that S provides a *YES* answer for the $1\,|gdd|\max\{|L_j|\}$ problem. Accordingly $\mathcal{J}_1$ includes exactly a single job out of the pair $\{J_{2t+1}, J_{2t+2}\}$. □

Following Lemma 2, and without loss of generality, assume that J_{2t+1} is included in $\mathcal{J}_1$. Moreover, let $\widehat{\mathcal{J}}_1 = \mathcal{J}_1 \setminus \{J_{2t+1}\}$. Note that $|\widehat{\mathcal{J}}_1| = t$.

Lemma 3 *In schedule S, the total processing time of all jobs in $\widehat{\mathcal{J}}_1$ is exactly B (i.e., $\Sigma_{J_j \in \widehat{\mathcal{J}}_1} p_j = B$).*

Proof By contradiction, assume that $\Sigma_{J_j \in \widehat{\mathcal{J}}_1} p_j > B$. In such a case $C_{[t+1]} = p_{2t+1} + \Sigma_{J_j \in \widehat{\mathcal{J}}_1} p_j > 2B + B = 3B$. Thus, $|L_{[t+1]}| = |C_{[t+1]} - \delta_{t+1}| > |3B - 2.5B| = 0.5B$, contradicting the fact that S provides a *YES* answer for the $1\,|gdd|\max\{|L_j|\}$ problem. On the other hand, if $\Sigma_{J_j \in \widehat{\mathcal{J}}_1} p_j < B$, it implies (based on Lemma 1) that job $J_{[t+2]}$ will start at time $p_{2t+1} + \Sigma_{J_j \in \widehat{\mathcal{J}}_1} p_j < 3B$. Thus, $C_{[t+2]} = p_{2t+1} + \Sigma_{J_j \in \widehat{\mathcal{J}}_1} p_j + p_{[t+2]} < 3B + 2B = 5B$. Therefore, $|L_{[t+2]}| = |C_{[t+2]} - \delta_{t+2}| > |5B - 5.5B| = 0.5B$, contradicting the fact that S provides a *YES* answer for the $1\,|gdd|\max\{|L_j|\}$ problem. □

Now, let $\mathcal{A}_1 = \{a_j \mid J_j \in \widehat{\mathcal{J}}_1\}$ and $\mathcal{A}_2 = \mathcal{A} \setminus \mathcal{A}_1$. The fact that $|\mathcal{A}_1| = |\widehat{\mathcal{J}}_1| = t$ and that $\Sigma_{a_j \in A_1} a_j = \Sigma_{J_j \in \widehat{\mathcal{J}}_1} p_j = B$ implies that we have a *YES* answer for the Equal Size Partition problem. □

2.2 A pseudo-polynomial time algorithm for a constant number of different due-dates

In this section, we develop a pseudo-polynomial time algorithm to solve the $1\,|gdd|\max\{|L_j|\}$ problem when the number of different due-dates, ν_d, is bounded by a constant. We assume that $\delta_1 < \delta_2 < \cdots < \delta_{\nu_d}$, and begin with few notations. Let m_i denote the number of positions in the job processing order having a due-date of δ_i for $i = 1, \ldots, \nu_d$. Moreover, let $l_r = \Sigma_{i=1}^{r} m_i$ denote the number of positions having due-date not greater than δ_r for $r = 1, \ldots, \nu_d$, with $l_{\nu_d} = n$ by definition. Given a solution, by $\mathcal{J}_i$ we denote the set of m_i jobs assigned to due-date δ_i $(i = 1, \ldots, \nu_d)$. Our pseudo-polynomial time algorithm exploits the properties in following two lemmas:

Lemma 4 *There exists an optimal schedule for the $1\,|gdd|\max\{|L_j|\}$ problem, which includes no machine idle times between the processing of any two consecutive jobs in each $\mathcal{J}_i$ $(i = 1, \ldots, \nu_d)$.*

Proof Consider an optimal solution S^*, which includes machine idle time of duration Δ between the processing of jobs $J_{[j]}$ and $J_{[j+1]}$ both belong to the same set $\mathcal{J}_i$ (i.e., $j \in \{l_{i-1} + 1, \ldots, l_i\}$). Define an alternative solution S' out of S^*, by starting the processing of jobs $\{J_{[j+1]}, .., J_{[l_i]}\}$ Δ time units earlier (i.e., by eliminating the idle time between jobs $J_{[j]}$ and $J_{[j+1]}$), while keeping the schedule of all other jobs unchanged.

As all jobs in $\mathcal{J}_i$ share the same due-date of d_i, job $J_{[l_{i-1}+1]}$ has the maximum earliness value among all jobs in $\mathcal{J}_i$, while job $J_{[l_i]}$ has the maximal tardiness value among all jobs in $\mathcal{J}_i$. As the schedule of job $J_{[l_{i-1}+1]}$ is identical in both S' and S^*, the maximum earliness among all jobs in $\mathcal{J}_i$ is the same in both S' and S^*. However, job $J_{[l_i]}$ is completed Δ earlier in S' comparing to its completion time in S^*. Therefore, the maximum tardiness among all jobs in $\mathcal{J}_i$ is not greater in S' than it is in S^*. Thus, the absolute lateness among all jobs in S' is not greater than it is in S^*. Accordingly, schedule S' is optimal as well. By repeating this procedure for any pair of consecutive jobs in S^* sharing the same due-date and having idle time between them, we complete the proof. □

Lemma 5 *There exists an optimal schedule for the $1\,|gdd|\max\{|L_j|\}$ problem, where the job with the largest processing time among all jobs in each set $\mathcal{J}_i$ is scheduled first (within its job set), while the processing order of all other jobs in $\mathcal{J}_i$ can be arbitrary.*

Proof As all jobs in $\mathcal{J}_i$ share the same due-date of d_i, the first scheduled job in $\mathcal{J}_i$, which is job $J_{[l_{i-1}+1]}$, has the maximum earliness value among all jobs in $\mathcal{J}_i$, while the last scheduled job in $\mathcal{J}_i$, which is job $J_{[l_i]}$, has the maximal tardiness value among all jobs in $\mathcal{J}_i$. Consider now an optimal solution, S^*, which follows the property in Lemma 4. It follows that in S^*

$$\begin{aligned}\max_{J_j \in \mathcal{J}_i}\{|L_j|\} &= \max\{E_{[l_{i-1}+1]}, T_{[l_i]}\}\\ &= \max\{\max\{0, \delta_i - A_i - p_{[l_{i-1}+1]}\},\\ &\quad \max\{0, A_i + P(\mathcal{J}_i) - \delta_i\}\},\end{aligned}$$

where A_i is the start time of set $\mathcal{J}_i$ in S^*, and $P(\mathcal{J}_i) = \Sigma_{J_j \in \mathcal{J}_i} p_j$. The lemma now follows from the fact the value of $T_{[l_i]}$ is independent of the internal schedule of the jobs in $\mathcal{J}_i$, and that the value of $E_{[l_{i-1}+1]}$ is a non-increasing function of $p_{[l_{i-1}+1]}$. □

Consider now a feasible partition of job set $\mathcal{J}$ into the subsets $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{\nu_d}$, and let (i) $P_i = \Sigma_{J_j \in \mathcal{J}_i} p_j$ represent the total processing time of all jobs assigned to set $\mathcal{J}_i$; and

(ii) and $\alpha_i = \max_{J_j \in \mathcal{J}_i} p_j$ represent the maximal processing time among all jobs assigned to set $\mathcal{J}_i$. Given the values of P_i and α_i for $i = 1, \ldots, \nu_d$, we can compute the optimal starting times of job sets $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{\nu_d}$ by using a *Linear Programming* (*LP*) formulation, consisting solely of continuous variables. In the formulation, S_i is a continuous variable representing the start time of set $\mathcal{J}_i$ on the single machine $(i = 1, \ldots, \nu_d)$, and Z is a continuous variable representing the value of the maximum absolute lateness. To ensure that the processing intervals of the sets do not overlap, we include the set of constraints that

$$S_i \geq S_{i-1} + P_{i-1} \text{ for } i \in \{1, \ldots, \nu_d\} \tag{3}$$

with $S_0 = P_0 = 0$ by definition. Due to Lemmas 4 and 5, and due to the fact that all jobs in $\mathcal{J}_i$ share the same due-date of δ_i, the maximal earliness value of a job belonging to $\mathcal{J}_i$ is equal to $\max\{0, \delta_i - S_i - \alpha_i\}$, and the maximal tardiness value of a job belonging to $\mathcal{J}_i$ is equal to $\max\{0, S_i + P_i - \delta_i\}$. Therefore, we include the following set of constraints as well

$$Z \geq \delta_i - S_i - \alpha_i \text{ for } i \in \{1, \ldots, \nu_d\}, \tag{4}$$

and

$$Z \geq S_i + P_i - \delta_i \text{ for } i \in \{1, \ldots, \nu_d\}. \tag{5}$$

Thus, given a feasible partition of job set $\mathcal{J}$ into the subsets $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{\nu_d}$ represented by the vector $(P_1, P_2, \ldots, P_{\nu_d}, \alpha_1, \alpha_2, \ldots, \alpha_{\nu_d})$, one can compute the optimal starting time of the job sets, and the minimal objective value by solving a *LP* problem of finding a solution that minimizes Z, subject to the set of constraints in (3)–(5). Using Karmarkar's Method (see Karmarkar, 1984), this can be done in $O(\nu_d^{3.5})$ time (note that we have ν_d continuous variables in the *LP* formulation). Thus, the following holds:

Lemma 6 *Given a vector $(P_1, P_2, \ldots, P_{\nu_d}, \alpha_1, \alpha_2, \ldots, \alpha_{\nu_d})$ representing a feasible partition of $\mathcal{J}$ into the sets $\mathcal{J}_1$, $\mathcal{J}_2$, $\ldots$, $\mathcal{J}_{\nu_d}$, one can compute the optimal starting time of each of the sets $\mathcal{J}_i$, $\mathcal{J}_2$, $\ldots$, $\mathcal{J}_{\nu_d}$, and the minimal objective value in $O(\nu_d^{3.5})$ time.*

It follows from Lemma 6 that we can solve our $1\,|gdd|\,\max\{|L_j|\}$ problem, by finding all possible $(P_1, P_2, \ldots, P_{\nu_d}, \alpha_1, \alpha_2, \ldots, \alpha_{\nu_d})$ vectors, each representing at least a single feasible partition of $\mathcal{J}$ into the sets $\mathcal{J}_i$, $\mathcal{J}_2$, $\ldots$, $\mathcal{J}_{\nu_d}$. Then, for each such vector, we can find the optimal objective value by solving the *LP* formulation. Following this process, we select as an optimal solution a schedule that corresponds to the vector which yields the minimum objective value among all the vectors.

Next, we present a state generation process that constructs all possible $(P_1, P_2, \ldots, P_{\nu_d}, \alpha_1, \alpha_2, \ldots, \alpha_{\nu_d})$ vectors representing feasible partitions of $\mathcal{J}$ into the sets $\mathcal{J}_1$, $\mathcal{J}_2$, $\ldots$, $\mathcal{J}_{\nu_d}$. We begin by re-indexing our jobs according to the longest processing time (*LPT*) rule such that $p_1 \geq p_2 \geq \cdots \geq p_n$. Now, let state $(j, k_1, \ldots, k_{\nu_d}, P_1, \ldots, P_{\nu_d}, \alpha_1, \ldots, \alpha_{\nu_d})$ represent a feasible partition of job set $\{J_1, \ldots, J_j\}$ into the sets $\mathcal{J}_1$, $\mathcal{J}_2$, $\ldots$, $\mathcal{J}_{\nu_d}$, where $k_i \leq m_i$ represents the number of jobs assigned to $\mathcal{J}_i$ $(i \in \{1, \ldots, \nu_d\})$. Let $\mathcal{L}_j$ represent all possible states on job set $\{J_1, \ldots, J_j\}$.

We initialize our state generation process by including a single state $(0, k_1 = 0, \ldots, k_{\nu_d} = 0, P_1 = 0, \ldots, P_{\nu_d} = 0, \alpha_1 = 0, \ldots, \alpha_{\nu_d} = 0)$ in $\mathcal{L}_0$. Then, for $j = 1, \ldots, n$, we construct $\mathcal{L}_j$ from $\mathcal{L}_{j-1}$ as follows: starting from each $(j-1, k_1, \ldots, k_{\nu_d}, P_1, \ldots, P_{\nu_d}, \alpha_1, \ldots, \alpha_{\nu_d}) \in \mathcal{L}_{j-1}$, we include at most ν_d states in $\mathcal{L}_j$ each representing a feasible assignment of J_j into one of the sets, $\mathcal{J}_1$, $\mathcal{J}_2$, $\ldots$, $\mathcal{J}_{\nu_d}$. Accordingly, for $r = 1, \ldots, \nu_d$, if $k_r < m_r$ we assign job J_j to set $\mathcal{J}_r$ and therefore include the state $(j, k'_1, \ldots, k'_{\nu_d}, P'_1, \ldots, P'_{\nu_d}, \alpha'_1, \ldots, \alpha'_{\nu_d})$ in $\mathcal{L}_j$ with (i) $k'_i = k_i$, $P'_i = P_i$ and $\alpha'_i = \alpha_i$ for $i \in \{1, \ldots, \nu_d\} \setminus \{r\}$; (ii) $k'_r = k_r + 1$; (iii) $P'_r = P_r + p_j$; and (iv) $\alpha'_r = p_j$ if $\alpha_r = 0$ and $\alpha'_r = \alpha_r$ if $\alpha_r > 0$. At the end of the state generation process, set $\mathcal{L}_n$ includes the set of all possible state vectors. To conclude the above analysis, the following algorithm can be used to solve the $1\,|gdd|\,\max\{|L_j|\}$ problem:

Algorithm 1 *An optimization algorithm for solving the* $1\,|gdd|\max\{|L_j|\}$ *problem.*
Input: n, $(\delta_1, ..., \delta_{\nu_d})$, $(m_1, ..., m_{\nu_d})$, $(p_1, ..., p_n)$.
Initialization:
Set $\mathcal{L}_0 = \{(0, k_1 = 0, ..., k_{\nu_d} = 0, P_1 = 0, ..., P_{\nu_d} = 0, \alpha_1 = 0, ..., \alpha_{\nu_d} = 0)\}$
and $\mathcal{L}_j = \emptyset$ *for* $j = 1, ..., n$. *Set* $Opt = \emptyset$ *and* $Opt\_value = \infty$.
Step 1: *Re-index the jobs according to the LPT rule such that* $p_1 \geq p_2 \geq ... \geq p_n$.
Step 2:
For $j = 1$ *to* n *do*
For any $(j-1, k_1, ..., k_{\nu_d}, P_1, ..., P_{\nu_d}, \alpha_1, , ..., \alpha_{\nu_d}) \in \mathcal{L}_{j-1}$ *do*
For $r = 1$ *to* ν_d *do*
If $k_r < m_r$ *then*
Include state $(j, k'_1, ..., k'_{\nu_d}, P'_1, ..., P'_{\nu_d}, \alpha'_1, ..., \alpha'_{\nu_d})$ *in* $\mathcal{L}_j$, *where*
$k'_i = k_i$, $P'_i = P_i$ *and* $\alpha'_i = \alpha_i$ *for* $i \in \{1, ..., \nu_d\}\diagdown\{r\}$; $k'_r = k_r + 1$; $P'_r = P_r + p_j$; *and*
$\alpha'_r = p_j$ *if* $\alpha_r = 0$ *and* $\alpha'_r = \alpha_r$ *if* $\alpha_r > 0$.
End if
End For
End For
End For
Step 3: *For any* $(n, m_1, ..., m_{\nu_d}, P_1, P_2, ..., P_{\nu_d}, \alpha_1, \alpha_2, ..., \alpha_{\nu_d}) \in \mathcal{L}_n$ *solve the* LP *formulation of minimizing* Z, *subject to the set of constraints in (3)-(5). Let* Z^* *be the optimal solution value.*
If $Z^* < Opt\_value$, *set* $Opt = (n, m_1, ..., m_{\nu_d}, P_1, P_2, ..., P_{\nu_d}, \alpha_1, \alpha_2, ..., \alpha_{\nu_d})$ *and* $Opt\_value = Z^*$.
Step 4: *Track back the optimal assignment of jobs into the job sets* $\mathcal{J}_1$, $\mathcal{J}_2$, ..., $\mathcal{J}_{\nu_d}$ *out of* Opt.

Theorem 2 *Algorithm 1 solves the* $1\,|gdd|\max\{|L_j|\}$ *problem in pseudo-polynomial time when the value of* ν_d *is upper bounded by a constant.*

Proof The correctness of the algorithm follows from the discussion in this section. Step 1 requires a sorting operation, and therefore can be done in $O(n \log n)$ time. The facts that (i) the value of P_i $(i \in \{1, \ldots, \nu_d\})$ is upper bounded by $P_\Sigma = \sum_{J_j \in \mathcal{J}} p_j$, and that (ii) there are $O(n)$ different possible values for each α_i, implies that there are at most $O((nP_\Sigma)^{\nu_d})$ states in each $\mathcal{L}_j$. In Step 2, we construct at most ν_d states in $\mathcal{L}_j$ out of state in $\mathcal{L}_{j-1}$, each of which requires $O(\nu_d)$ time. Thus, each iteration $j \in \{1, \ldots, n\}$ of Step 2 requires $O((\nu_d)^2(nP_\Sigma)^{\nu_d})$ time, and the overall complexity of Step 2 is $O((\nu_d)^2 n^{\nu_d+1}(P_\Sigma)^{\nu_d})$. In Step 3, for each state in $\mathcal{L}_n$, we solve an *LP* formulation, which requires $O(\nu_d^{3.5})$ time (see Lemma 6). The fact that we have $O((nP_\Sigma)^{\nu_d})$ states in each $\mathcal{L}_j$, implies that Step 3 can be done in $O(\nu_d^{3.5}(nP_\Sigma)^{\nu_d})$ time. In Step 4 we track back the optimal assignment of jobs into the job sets $\mathcal{J}_1$, $\mathcal{J}_2$, . . . , $\mathcal{J}_{\nu_d}$ out of Opt. This can easily be done in a linear time, and therefore the time complexity of Algorithm 1 is $O((\nu_d)^2 \max\{n, (\nu_d)^{1.5}\}(nP_\Sigma)^{\nu_d})$. If the value of ν_d is upper bounded by a constant, this time complexity reduces to $O(n^{\nu_d+1}(P_\Sigma)^{\nu_d})$, which is pseudo-polynomial. □

2.3 An FPT algorithm for the combined parameter (ν_d, ν_p)

In this section, we prove that the following result holds:

Theorem 3 *The* $1\,|gdd|\max_{J_j \in \mathcal{J}}\{|L_j|\}$ *problem is* FPT *wrt. the combined parameter* (ν_d, ν_p).

The proof of Theorem 3 is done by providing a *Mixed Integer Linear Programming* (*MILP*) formulation for the $1\,|gdd|\max_{J_j \in \mathcal{J}}\{|L_j|\}$ problem with $O(\nu_d\nu_p)$ integer variables. Then, Theorem 3 directly holds from the result by Lenstra (1983) who showed that the problem of solving an *MILP* is *FPT* wrt. the number of integer variables.

For ease of presentation, we represent the vector of n due-dates in a modified manner by two vectors of size ν_d: $(\delta_1, \delta_2, \ldots, \delta_{\nu_d})$ and $(m_1, m_2, \ldots, m_{\nu_d})$. The first includes the ν_d distinct due-dates in the instance. In the second, m_i represents the number of positions in the sequence having a due-date of δ_i. We also represent the vector of n processing times in a modified manner by two vectors of size ν_p: $(p_1, p_2, \ldots, p_{\nu_p})$ and $(n_1, n_2, \ldots, n_{\nu_p})$, where the first includes the ν_p distinct processing times in the instance, and n_i is the number of jobs having processing time of p_i for $i = 1, \ldots, \nu_p$. Without loss of generality, we assume that the due-dates are numbered according to the *EDD* rule, such that $\delta_1 < \delta_2 < \cdots < \delta_{\nu_d}$, and that the processing times are numbered according to the *SPT* rule, such that $p_1 < p_2 < \cdots < p_{\nu_p}$.

Now, let x_{ij} be an integer decision variable representing the number of jobs with processing time p_j allocated to due-date δ_i ($i \in \{1, \ldots, \nu_d\}$, $j \in \{1, \ldots, \nu_p\}$). Accordingly, we have the following set of constraints:

$$\sum_{i=1}^{\nu_d} x_{ij} = n_j \text{ for } j = 1, \ldots, \nu_p \;, \tag{6}$$

and

$$\sum_{j=1}^{\nu_p} x_{ij} = m_i \text{ for } i = 1, \ldots, \nu_d \tag{7}$$

Now, let S_i ($i = 1, \ldots, \nu_d$) be a continuous decision variable representing the start time of the jobs in set $\mathcal{J}_i$. Due to Lemma 4, the completion time of all jobs in set $\mathcal{J}_i$ is at time $S_i + \sum_{j=1}^{\nu_p} p_j x_{ij}$. Therefore, we include the following set of constraints that ensures that processing time intervals do not overlap:

$$S_{i+1} \geq S_i + \sum_{j=1}^{\nu_p} p_j x_{ij} \text{ for } i = 0, \ldots, \nu_d - 1, \tag{8}$$

with $S_0 = 0$ by definition.

Let Z be a continuous decision variable representing the maximum absolute lateness, and let α_i be a continuous decision variable representing the largest processing time among all processing times of the jobs in $\mathcal{J}_i$ ($i \in \{1, \ldots, \nu_d\}$). Due to Lemma 5, and the fact that the first scheduled job in $\mathcal{J}_i$ has the maximum earliness value among all jobs in $\mathcal{J}_i$, we include the set of constraints that

$$Z \geq \delta_i - S_i - \alpha_i \text{ for } i = 1, \ldots, \nu_d. \tag{9}$$

Moreover, due to Lemmas 4–5, and the fact that the last scheduled job in $\mathcal{J}_i$ has the maximal tardiness value among all jobs in $\mathcal{J}_i$, we also include the following set of constraints as well

$$Z \geq S_i + \sum_{j=1}^{\nu_p} p_j x_{ij} - \delta_i \text{ for } i = 1, \ldots, \nu_d. \tag{10}$$

To ensure that α_i is indeed the largest processing time among all processing times of the jobs in $\mathcal{J}_i$, we define y_{ij} as a binary variable that is equal to 1 when $x_{ij} \geq 1$ and is equal 0 otherwise ($i \in \{1, \ldots, \nu_d\}$, $j \in \{1, \ldots, \nu_p\}$). Then, we include the following set of constraints:

$$y_{ij} \leq x_{ij} \text{ for } i = 1, \ldots, \nu_d \text{ and } j = 1, \ldots, \nu_p; \tag{11}$$

and also the following set of constraints:

$$\alpha_i \leq p_j + \sum_{l=j+1}^{\nu_p} p_l y_{il} \text{ for } i = 1, \ldots, \nu_d \text{ and } j = 1, \ldots, \nu_p. \tag{12}$$

Now, consider the problem $\mathcal{P}$ of minimizing Z subject to the set of constraints in (6)–(12). It implies from the objective, and the set of constraints in (9)–(10) that in an optimal solution for $\mathcal{P}$

$$Z = \max_{i=1,\ldots,\nu_d} \{\delta_i - S_i - \alpha_i, S_i + \sum_{j=1}^{\nu_p} p_j x_{ij} - \delta_i\}, \tag{13}$$

which is exactly the value of the maximal absolute lateness of any solution that satisfy the properties in Lemmas 4 and 5 if indeed α_i receives the largest processing time among all processing times of the jobs in $\mathcal{J}_i$. To prove that, we note that Z is a non-increasing function of α_i for $i = 1, .., n$. Therefore, there exists an optimal solution for problem $\mathcal{P}$, where α_i is the largest value satisfying the set of constraints in (12). As the right hand side of the set of constraints in (12) is an increasing function of the y_{il} variables, there exists an optimal solution in which $y_{ij} = 1$ if $x_{ij} \geq 1$, and $y_{ij} = 0$ if $x_{ij} = 0$.

Assume now that p_q is the largest processing time among all processing times of the jobs assigned to $\mathcal{J}_i$ ($q \in \{1, \ldots, \nu_p\}$). It follows that $y_{iq} = 1$, and that $y_{ij} = 0$ for $j = q+1, \ldots, \nu_p$. Therefore, $p_j + \sum_{l=j+1}^{\nu_p} p_l y_{il} > p_q$ for any $j \in \{1, \ldots, q-1\}$, $p_q + \sum_{l=q+1}^{\nu_p} p_l y_{il} = p_q$ and $p_j + \sum_{l=j+1}^{\nu_p} p_l y_{il} = p_j > p_q$ for any $j \in \{q+1, \ldots, \nu_p\}$. Thus, the set of constraints in (12) reduces to $\alpha_i \leq p_q$, and since the value of Z is a non-increasing function of α_i for $i = 1, .., n$, there exists an optimal solution for $\mathcal{P}$ in which $\alpha_i = p_q$.

The fact that solving $\mathcal{P}$ provides an optimal solution for the $1\,|gdd|\max_{J_j \in \mathcal{J}}\{\left|L_j\right|\}$ problem, and that $\mathcal{P}$ includes $O(\nu_d \nu_p)$ integer variables, implies that we can use Lenstra's algorithm (1983) algorithm to solve the $1\,|gdd|\max_{J_j \in \mathcal{J}}\{\left|L_j\right|\}$ problem wrt. the combined parameter (ν_d, ν_p), and Theorem 3 follows.

3 Maximal weighted number of *JIT* Jobs

Gerstl and Mosheiov (2020) prove that the following theorem holds:

Theorem 4 *(Gerstl & Mosheiov, 2020) The $1\,|gdd|\,|\mathcal{E}|$ problem is strongly $\mathcal{NP}$-hard when the number of due-dates is arbitrary.*

Consider next the case where all due-dates are common, i.e., $\delta_j = \delta$ for $j = 1, \ldots, n$. For this case, it is obvious that at most a single job can be a *JIT* job, i.e., that for any feasible schedule we have that $|\mathcal{E}| \leq 1$. Now, let $\mathcal{J}' = \{J_j \in \mathcal{J} \,\big|\, p_j \leq \delta\}$ be the subset of jobs which can be scheduled in a *JIT* mode. It implies that among all jobs in $\mathcal{J}'$ it is optimal to schedule the one of maximal weight in a *JIT* mode. Therefore, the following corollary holds:

Corollary 1 *The $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ problem is solvable in $O(n)$ time when $\nu_d = 1$.*

In the following three subsections, we complement the results in Theorem 4 and Corollary 1 by showing that (i) the $1\,|gdd|\,|\mathcal{E}|$ problem is $\mathcal{NP}$-hard even when we have only two distinct due-dates in the instance; (ii) the $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ problem is solvable in pseudo-polynomial time when the value of ν_d is upper bounded by a constant; and (iii) the $1\,|gdd|\,|\mathcal{E}|$ problem is *FPT* wrt. (ν_d, ν_p). We leave the question of whether the $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ is *FPT* wrt. (ν_d, ν_p) open.

3.1 $\mathcal{NP}$-hardness for the two distinct due-dates case

In this subsection, we show that the $1\,|gdd|\,|\mathcal{E}|$ is $\mathcal{NP}$-hard even when $\nu_d = 2$. The reduction is done from the ordinary $\mathcal{NP}$-hard Partition problem, which is defined below.

Definition 2 Partition: Given a set of t positive integers $\mathcal{A} = \{a_1, a_2, \ldots, a_t\}$ with $\Sigma_{j=1}^{t} a_j = 2B$ and $0 < a_j < B$. Can $\mathcal{A}$ be partitioned into two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\Sigma_{a_j \in \mathcal{A}_i} a_j = B$ for $i = 1, 2$?

Theorem 5 *The $1\,|gdd|\,|\mathcal{E}|$ problem is $\mathcal{NP}$-hard even when there are only two distinct due-dates (i.e., even when $\nu_d = 2$).*

Proof The proof is based on a reduction from the $\mathcal{NP}$-hard Partition problem. Given an instance for the Partition problem, we construct the following instance to our scheduling problem with a set $\mathcal{J} = \{J_1, \ldots, J_n\}$ of $n = t$ jobs. The processing times are

$$p_j = a_j \tag{14}$$

for $j = 1, \ldots, t$. The due-dates are

$$\delta_j = \begin{cases} B & \text{for } j = 1, \ldots, t-1 \\ 2B & \text{for } \quad j = t \end{cases}. \tag{15}$$

In the decision version of the problem, we ask whether there is a feasible schedule with $|\mathcal{E}| = 2$. Note that there are only two distinct due-dates in the constructed instance of the $1\,|gdd|\,|\mathcal{E}|$ problem. Therefore, $|\mathcal{E}| \leq 2$ in any feasible schedule.

Consider a solution that provides a *YES* answer for the Partition problem, we construct the following solution S to our $1\,|gdd|\,|\mathcal{E}|$ problem. We set $\mathcal{J}_i = \{J_j \,|a_j \in \mathcal{A}_i\}$ for $i = 1, 2$. Then, we schedule all jobs in $\mathcal{J}_1$ before any job in $\mathcal{J}_2$ with no machine idle times. The processing order within each $\mathcal{J}_i$ $(i = 1, 2)$ is arbitrary. As $\Sigma_{J_j \in \mathcal{J}_i} p_j = \Sigma_{J_j \in \mathcal{A}_i} a_j = B$, the completion time of the last job in $\mathcal{J}_1$ is exactly at time B. Moreover, the fact that $|\mathcal{J}_1| < n$, implies that the last job to be scheduled in $\mathcal{J}_1$ is assigned with a due-date of B (see eq. (15)). Therefore, the last scheduled job in $\mathcal{J}_1$ is completed in a *JIT* mode. Moreover, as we schedule all jobs with no idle times, the last job to be scheduled in $\mathcal{J}_2$ is completed at time *2B*, which is also the due-date of the last scheduled job (see eq. (15)). Accordingly, in S we have that $|\mathcal{E}| = 2$, and therefore S provides a *YES* answer for the constructed instance of our $1\,|gdd|\,|\mathcal{E}|$ problem.

Consider next a solution S that provides a *YES* answer for the constructed instance of our $1\,|gdd|\,|\mathcal{E}|$ problem. It implies that the last scheduled job is completed exactly at time $2B$. Thus, in S, there are no machine idle times. The fact that $|\mathcal{E}| = 2$ in S implies that there is job which completes exactly at time B in schedule S. Let $\mathcal{J}_1$ be the set of jobs that are completed no later than time B in S, and let $\mathcal{J}_2$ be the set of all other jobs. As there are no machine idle time in S, we have that $\Sigma_{J_j \in \mathcal{J}_i} p_j = B$ for $i = 1, 2$. Therefore, by setting $\mathcal{A}_i = \{a_j \,\big|J_j \in \mathcal{J}_i\}$ for $i = 1, 2$, we have a solution that provides a *YES* answer for the Partition problem. □

3.2 A pseudo-polynomial time algorithm for a constant number of different due-dates

Let $\mathcal{L}$ be the set of all $l = (l_1, l_2, \ldots, l_r)$ vectors satisfying that (i) $l_i \in \{1, \ldots, n\}$; and (ii) $\delta_{l_i - 1} < \delta_{l_i}$ for $i = 1, \ldots, r$ with $l_0 = 0$ by definition (note that $r \leq \nu_d$). There are $O(n^{\nu_d})$ such vectors, each includes a subset of positions in the job sequence having different due-dates. It implies that any feasible $\mathcal{E}$ set includes jobs that are scheduled in the positions of some $l \in \mathcal{L}$. Therefore, we solve the $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ problem, by partitioning it into a set of $O(n^k)$ (where k is the number of due-dates) subproblems each corresponding to a different $(l_1, l_2, \ldots, l_r) \in \mathcal{L}$. Let $P(l)$ be the subproblem corresponding to vector l. Our $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ problem reduces to finding the vector $l \in \mathcal{L}$ with the best feasible solution (if such a solution exists) satisfying that the set of *JIT* jobs are scheduled in positions $l_1, l_2, \ldots, l_r$.

Given a subproblem $P(l)$, corresponding to vector $(l_1, l_2, \ldots, l_r) \in \mathcal{L}$, we let $n_i = l_i - l_{i-1}$ for $i = 1, \ldots, r+1$, where $l_{r+1} = n$ by definition. Let S be a feasible solution for the corresponding subproblem with $\mathcal{A}_i$ being the set of n_i jobs scheduled in positions $\{l_{i-1}+1, \ldots, l_i\}$ for $i = 1, \ldots, r+1$. Note that all jobs in each $\mathcal{A}_i$ $(i = 1, \ldots, r)$ share the same due-date of δ_{l_i}. Accordingly, only a single job in each $\mathcal{A}_i$ $(i = 1, \ldots, r)$ can be completed at the common due-date, δ_{l_i}. It follows that $\Sigma_{J_j \in \mathcal{A}_i} p_j \leq \delta_{l_i} - \delta_{l_i - 1}$ for $i = 1, \ldots, r$, as otherwise we cannot complete the last job in $\mathcal{A}_i$ at time δ_{l_i}, given that the last job in $\mathcal{A}_{i-1}$ is completed at time $\delta_{l_i - 1}$. The following lemma obviously holds:

Lemma 7 *Given a feasible solution S for problem $P(l)$ with n_i jobs in each set $\mathcal{A}_i$ and with $\Sigma_{J_j \in \mathcal{A}_i} p_j \leq \delta_{l_i} - \delta_{l_i - 1}$ for $i = 1, \ldots, r$, it is optimal to schedule all jobs in $\mathcal{A}_i$ during* *time interval $(\delta_{l_i} - \Sigma_{J_j \in \mathcal{A}_i} p_j, \delta_{l_i}]$ with the last job, which is the only one in $\mathcal{A}_i$ being scheduled in a JIT mode, is the one of maximal weight among all jobs in $\mathcal{A}_i$.*

We solve each subproblem $P(l)$ by using a dynamic programming procedure, which starts by re-indexing the jobs in a non-increasing order of weight, such that $w_1 \geq w_2 \geq \cdots \geq w_n$. Now, let $F_j(x_1, \ldots, x_{r+1}, q_1, \ldots, q_{r+1})$ be the maximal gain that can be obtained out of all feasible partial schedule on job set $\mathcal{J}_j = \{J_1, \ldots, J_j\}$ with q_i jobs the assigned to set $\mathcal{A}_i$ and $x_i = \Sigma_{J_j \in \mathcal{A}_i} p_j$ for $i = 1, \ldots, r+1$. It follows from the feasibility of the partial schedule that the following conditions holds:

Condition 1 *$x_i \leq \delta_{l_i} - \delta_{l_i - 1}$ for $i = 1, \ldots, r$, as otherwise we cannot complete the last job in $\mathcal{A}_i$ at time δ_{l_i}, given that the last job in $\mathcal{A}_{i-1}$ is completed at time $\delta_{l_i - 1}$.*

Condition 2 *$x_{r+1} = \Sigma_{l=1}^{j} p_l - \Sigma_{i=1}^{r} x_i$, and $q_{r+1} = j - \Sigma_{i=1}^{r} q_i$ as any job has to be assigned to one of the $\mathcal{A}_i$ sets $(i = 1, \ldots, r+1)$.*

Condition 3 *For any $j = 1, \ldots, n$ and $i = 1, \ldots, r+1$, $q_i \leq n_i$ and $q_i + (n - j) \geq n_i$; as otherwise we cannot construct a complete solution with n_i jobs in $\mathcal{A}_i$ for $i = 1, \ldots, r$.*

Based on Lemma 7 and the fact that the jobs are re-indexed such that $w_1 \geq w_2 \cdots \geq w_n$, we can compute $F_j(x_1, \ldots, x_{r+1}, q_1, \ldots, q_{r+1})$ for $j = 1, \ldots, n$ and for any set of x_i and q_i values satisfying the conditions in (1)–(3) by using the following recursion:

$$F_j(x_1, \ldots, x_{r+1}, q_1, \ldots, q_{r+1}) = \max_{i=1,\ldots,r+1} \begin{cases} F_{j-1}(x_1, \ldots, x_i - p_j, \ldots, x_{r+1}, q_1, \ldots, q_i - 1, \ldots, q_{r+1}) + w_j & \text{if } i < r+1 \text{ and } q_i = 1 \\ F_{j-1}(x_1, \ldots, x_i - p_j, \ldots, x_{r+1}, q_1, \ldots, q_i - 1, \ldots, q_{r+1}) & \text{otherwise} \end{cases} \quad (16)$$

with the initial condition that

$$F_0(x_1, \ldots, x_{r+1}, q_1, \ldots, q_{r+1}) = \begin{cases} 0 & \text{if } x_i = q_i = 0 \text{ for } i = 1, \ldots, r+1 \\ -\infty & \text{otherwise} \end{cases} \quad (17)$$

At the end of the computing process, the optimal solution for $P(l)$ is given by

$$F^*(l) = \max\{F_n(x_1, \ldots, x_{r+1}, n_1, \ldots, n_{r+1}) \mid x_i \leq \delta_{l_i} - \delta_{l_i - 1} \text{ for } i = 1, \ldots, r \text{ and } \Sigma_{i=1}^{r+1} x_i = \Sigma_{l=1}^{n} p_l\}. \quad (18)$$

To conclude, we can use the following algorithm to solve our $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ problem:

Algorithm 2 *An optimization algorithm for solving the $1\,|gdd|\,\Sigma_{J_j \in \mathcal{E}} w_j$ problem.*

Initialization *Set $F^* = 0$.*

Step 1 *Determine $\mathcal{L}$ which is the set of all $l = (l_1, l_2, \ldots, l_r)$ vectors satisfying that (i) $l_i \in \{1, \ldots, n\}$; and (ii) $\delta_{l_i - 1} < \delta_{l_i}$ for $i = 1, \ldots, r$ with $l_0 = 0$ by definition.*

Step 2

For any $l \in \mathcal{L}$ do:

Calculate $n_i = l_i - l_{i-1}$ for $i = 1, \ldots, r+1$, with $l_0 = 0$ and $l_{r+1} = n$.

For $j = 1$ up to $j = n$ do

For any set of non-negative integers $(x_1, \ldots, x_r, q_1, \ldots, q_r)$ satisfying $x_i \leq \delta_{l_i} - \delta_{l_i - 1}$, that $j \geq \Sigma_{i=1}^{r} q_i$, that $\Sigma_{l=1}^{j} p_l \geq \Sigma_{i=1}^{r} x_i$, and that $q_i \leq n_i$ and $q_i + (n - j + 1) \geq n_i$ for $i = 1, \ldots, r$ do:

Compute $x_{r+1} = \Sigma_{l=1}^{j} p_l - \Sigma_{i=1}^{r} x_i$, and $q_{r+1} = j - \Sigma_{i=1}^{r} q_i$.

Compute $F_j(x_1, \ldots, x_{r+1}, q_1, \ldots, q_{r+1})$ by (16), with the initial condition in (17).

End For

End For

Calculate $F^*(l)$ by (18).

If $F^*(l) > F^*$, then update $F^* = F^*(l)$.

Step 3 *Determine the optimal assignment of jobs to sets $\mathcal{A}_i$ for $i = 1, \ldots, r+1$ by tracking the decisions that lead to the optimal solution value, F^*.*

Output *The optimal solution value is F^*. For $i = 1, .., r$ schedule the jobs in each $\mathcal{A}_i$ during time interval $(\delta_{l_i} - \Sigma_{J_j \in \mathcal{A}_i} p_j, \delta_{l_i}]$ where the last schedule job in each interval $(\delta_{l_i} - \Sigma_{J_j \in \mathcal{A}_i} p_j, \delta_{l_i}]$ is the job of maximal weight (smallest index) among all jobs in $\mathcal{A}_i$. Schedule the jobs in $\mathcal{A}_{r+1}$ during time interval $(\delta_{l_r}, \delta_{l_r} + \Sigma_{J_j \in \mathcal{A}_i} p_j]$.*

Theorem 6 *Algorithm 2 solves the* $1\,|gdd|\,\Sigma_{J_j\in\mathcal{E}} w_j$ *problem in* $O(\nu_d n^{2\nu_d+1}\Pi_{i=1}^{\nu_d}\delta_{l_i})$.

Proof The fact that Algorithm 2 provides the optimal solution follows from the discussion in this section. Step 1 requires $O(n^{\nu_d})$ time as the number of possible l vectors. In Step 2, for any $l\in\mathcal{L}$, we compute $F_j(x_1,\ldots,x_{r+1},q_1,\ldots,q_{r+1})$ by (16). The fact that $r=O(\nu_d)$, that $\delta_{l_i}-\delta_{l_i-1}=O(\delta_{l_i})$ and that $q_i\le n_i=O(n)$, implies that we compute $O(n^{\nu_d}\Pi_{i=1}^{\nu_d}\delta_{l_i})$ different values of $F_j(x_1,\ldots,x_{r+1},q_1,\ldots,q_{r+1})$ at any stage $j\in\{1,\ldots,n\}$, and $O(n^{\nu_d+1}\Pi_{i=1}^{\nu_d}\delta_{l_i})$ values of $F_j(x_1,x_2,\ldots,x_{r+1},q_1,q_2,\ldots,q_{r+1})$ in total. As the computation of each $F_j(x_1,\ldots,x_{r+1},q_1,\ldots,q_{r+1})$ by (16) requires $O(\nu_d)$ time, the time required in Step 2 to compute all $F_j(x_1,\ldots,x_{r+1},q_1,\ldots,q_{r+1})$ values for a given $l\in\mathcal{L}$ is $O(\nu_d n^{\nu_d+1}\Pi_{i=1}^{\nu_d}\delta_{l_i})$. As calculating $F^*(l)$ by (18) can be done in $O(n^{\nu_d}\Pi_{i=1}^{\nu_d}\delta_{l_i})$ time, applying Step 2 for a given l vector requires $O(\nu_d n^{\nu_d+1}\Pi_{i=1}^{\nu_d}\delta_{l_i})$ time. The fact that we repeat Step 2, for any $l\in\mathcal{L}$, and that $|\mathcal{L}|=O(n^{\nu_d})$, implies that Step 2 requires $O(\nu_d n^{2\nu_d+1}\Pi_{i=1}^{\nu_d}\delta_{l_i})$ time. This time complexity reduces to $O(n^{\nu_d+1}\Pi_{i=1}^{\nu_d}\delta_{l_i})$ when the value of ν_d is upper bounded by a constant. The theorem now follows from the fact that tracking the decisions that lead to the optimal solution value in Step 3 require only linear time. □

3.3 An FPT algorithm for the $1\,\left|gdd\right|\,|\mathcal{E}|$ problem wrt. the combined parameter (ν_d, ν_p)

In this section, we prove that the following result holds:

Theorem 7 *The* $1\,|gdd|\,|\mathcal{E}|$ *problem is* FPT *wrt. the combined parameter* (ν_d,ν_p).

The proof of Theorem 7 is based on breaking down the $1\,|gdd|\,|\mathcal{E}|$ problem into a set of $O(2^{\nu_d})$ subproblems and providing a *MILP* formulation with $O(\nu_d\nu_p)$ integer variables for each of the subproblems. Then the fact that Theorem 7 holds follows directly from the result by Lenstra (1983) that shows that the problem of solving an *MILP* is *FPT* wrt. the number of integer variables.

For ease of presentation, we represent the vector of n due-dates in a modified manner by two vectors of size ν_d: $(\delta_1,\delta_2,..,\delta_{\nu_d})$ and $(m_1,m_2,..,m_{\nu_d})$. The first includes the ν_d distinct due-dates in the instance. In the second, m_i represents the number of positions in the sequence having a due-date not greater than δ_i. We order the due-dates in the first vector according to the *EDD* rule, such that $\delta_1<\delta_2<\cdots<\delta_{\nu_d}$. By definition, we also have that $m_1<m_2<\cdots<m_{\nu_d}$ with $m_{\nu_d}=n$. We also represent the vector of n processing times in a modified manner by two vectors of size ν_p: $(p_1,p_2,..,p_{\nu_p})$ and $(n_1,n_2,..,n_{\nu_p})$, where the first includes the ν_p distinct processing times in the instance, and n_i is the number of jobs having processing time of p_i for $i=1,\ldots,\nu_p$.

Given a feasible schedule, we say that due-date δ_i is *active* if there exists a job completed exactly at time δ_i. Now, let Δ be a set that includes all subsets of set $(\delta_1,\delta_2,..,\delta_{\nu_d})$. Note that $|\Delta|=O(2^{\nu_d})$. We partition our $1\,|gdd|\,|\mathcal{E}|$ problem into a set of $O(2^{\nu_d})$ subproblems, each corresponding to a specific set of distinct due-dates $\delta\in\Delta$. In each such subproblem, we aim to find if there exists a feasible solution with all due-dates in δ being active. Let $P(\delta)$ be the subproblem that corresponds to vector δ. Our $1\,|gdd|\,|\mathcal{E}|$ problem reduces to finding the set $\delta\in\Delta$ of maximal cardinality for which there is a feasible solution for $P(\delta)$.

Given $\delta\in\Delta$, we next show that each $P(\delta)$ problem can be represented as an *MILP* with only $O(\nu_d\nu_p)$ integer variables. Let $\delta=(\delta_{[1]},\delta_{[2]},\ldots,\delta_{[k]})$ (note that $k\le\nu_d$). We define x_i as a positive integer variable that represents the position in the sequence of the job completed at $\delta_{[i]}$ for $i=1,..,k$. As there are exactly m_i positions with due-date not larger than δ_i, we include the following set of constraints for $i=1,\ldots,k$:

$$m_{[i-1]}+1\le x_i\le m_{[i]}.$$

We also define y_{ij} as a nonnegative integer variable which represents the number of jobs having processing time of p_j that are assigned to positions $x_{i-1}+1,\ldots,x_i$ in the sequence. Accordingly, for each $i=1,\ldots,k$ we include the constraint that

$$\sum_{j=1}^{\nu_p} y_{ij}=x_i-x_{i-1},$$

with $x_i=0$ by definition. To make sure that $\delta_{[i]}$ is indeed an active due-date, we also include the following set of constraints for $i=1,\ldots,k$:

$$\sum_{j=1}^{\nu_p} p_j y_{ij}\le\delta_{[i]}-\delta_{[i-1]}.$$

Finally, to ensure that we do not assign more than n_j jobs with processing time p_j, up to position k, we include the following set of constraints for $j=1,\ldots,\nu_p$:

$$\sum_{i=1}^{k} y_{ij}\le n_j.$$

As $k\le\nu_d$, the above formulation includes $O(\nu_d+\nu_d\nu_p)=O(\nu_d\nu_p)$ integer variables in each subproblem $P(\delta)$.

4 Summary and future research

Many scheduling problems with due-date related objective function are $\mathcal{NP}$-hard when the number of different due-dates in the instance, ν_d, is arbitrary. However, in many real-life problems the number of due-dates is a bounded parameter due to many reasons, such as high shipment costs, and work agreements. Therefore, there is great interest in studying the complexity of such $\mathcal{NP}$-hard problems with respect to the number of due-dates in the instance. We consider two such single-machine scheduling problems with generalized due-dates. The first, denoted by $1\,|gdd|\,\max\{\left|L_j\right|\}$, focuses on minimizing the maximal absolute lateness, and the second, denoted by $1\,|gdd|\,\Sigma_{J_j\in\mathcal{E}}w_j$, consists of maximizing the weighted number of jobs completed in a *JIT* mode (i.e., exactly at their due-date). Both problems are known to be strongly $\mathcal{NP}$-hard for arbitrary values of ν_d. We show that both problems are solvable in polynomial time when $\nu_d = 1$. Moreover, we show that problems $1\,|gdd|\,\max\{\left|L_j\right|\}$ and $1\,|gdd|\,|\mathcal{E}|$ are $\mathcal{NP}$-hard when $\nu_d = 2$. We compliment these two results by showing that problems $1\,|gdd|\,\max\{\left|L_j\right|\}$ and $1\,|gdd|\,\Sigma_{J_j\in\mathcal{E}}w_j$ are solvable in pseudo-polynomial time when the value of ν_d is bounded by a constant.

The fact that problems $1\,|gdd|\,\max\{\left|L_j\right|\}$ and $1\,|gdd|\,|\mathcal{E}|$ are $\mathcal{NP}$-hard even if $\nu_d = 2$ rules out the possibility of constructing an *FPT* algorithm for neither problem wrt. ν_d, unless $\mathcal{P} = \mathcal{NP}$. We show, however, that both problems are *FPT* wrt. the combined parameter (ν_d, ν_p) where ν_p is the number of different processing times in the instance. We leave open the question whether the $1\,|gdd|\,\Sigma_{J_j\in\mathcal{E}}w_j$ problem is *FPT* when we combine parameters ν_d and ν_p.

In future research, we aim to study the complexity status of other $\mathcal{NP}$-hard scheduling problems with due-date related objective function wrt. to ν_d, hoping to provide efficient algorithms to solve practical instances with limited ν_d values.

Acknowledgements The first author was supported by the Israel Science Foundation (Grant No.2505/19), by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project Number 452470135 and by the Recanati Fund of The School of Business Administration, The Hebrew University of Jerusalem, Israel.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

References

Browne, J., Dubois, D., Rathmill, K., Sethi, S. P., & Stecke, K. (1984). Classification of flexible manufacturing systems. *The FMS Magazine, 2*, 114–117.

Cheng, T. C. E. (1987). Minimizing the maximum deviation of job completion time about a common due-date. *Computers and Mathematics with Applications, 14*, 279–283.

Downey, R., & Fellows, M. (1999). *Parameterized complexity*. Springer.

Du, J., & Leung, J. Y. T. (1990). Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research, 15*, 483–495.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability - a guide to the theory of NP completeness*. Freeman.

Garey, M. R., Tarjan, R. E., & Wilfong, G. T. (1988). One-processor scheduling with symmetric earliness and tardiness penalties. *Mathematics of Operations Research, 13*, 330–348.

Gerstl, E., & Mosheiov, G. (2020). Single machine scheduling to maximize the number of on-time jobs with generalized due-dates. *Journal of Scheduling, 23*(3), 289–299.

Gao, Y., & Yuan, J. (2006). Unary NP-hardness of minimizing total weighted tardiness with generalized due-dates. *Operations Research Letters, 44*, 92–95.

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

Hall, N. G. (1986). Scheduling problems with generalized due dates. *IIE Transactions, 18*, 220–222.

Hall, N. G., Sethi, S. P., & Sriskandarajah, C. (1991). On the complexity of generalized due-date scheduling problems. *European Journal of Operational Research, 51*, 100–109.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica, 4*(4), 373–395.

Lann, A., & Mosheiov, G. (1996). Machine scheduling to minimize the number of early and tardy jobs. *Computers and Operations Research, 23*, 765–781.

Lenstra, H. L. (1983). Integer programming with a fixed number of variables. *Mathematics of Operations Research, 8*(4), 538–548.

Niedermeier, R. (2006). *Invitation to fixed-parameter algorithms. Oxford lecture series in mathematics and its applications*. Oxford Univerity Press.

Sriskandarajah, C. (1990). A note on the generalized due-dates scheduling problems. *Naval Research Logistics, 37*, 587–597.

Stecke, K. E., & Solberg, J. J. (1981). Loading and control policies for a flexible manufacturing system. *International Journal of Production Research, 19*, 481–490.

Tanaka, K., & Vlach, M. (1999). Minimizing maximum absolute lateness and range of lateness under generalized due-dates on a single machine. *Annals of Operations Research, 86*, 507–526.

Yin, Y., Cheng, S. R., Cheng, T. C. E., Wu, C. C., & Wu, W. H. (2012). Two-agent single-machine scheduling with assignable due-dates. *Applied Mathematics and Computation, 219*, 1674–1685.