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

Gur Mosheiov · Daniel Oron · Dvir Shabtay

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, \( \delta_j \). 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, \( \nu_d \). 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 \( \nu_d \) 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 (\( \nu_d \)) and number of different processing times (\( \nu_p \)).

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.

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 $NP$-hard, the reduction is done to an instance that includes an arbitrary number of different due-dates, $\nu_d$ (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 $\nu_d$ may be of limited size. Therefore, it is interesting to investigate whether or not the problems become tractable for bounded values of $\nu_d$. 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 $J = \{J_1, J_2, \ldots, J_n\}$ 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 $j$th position in $\pi$, and by (ii) a feasible set of processing intervals, $(S_{[j]}, C_{[j]} = S_{[j]} + p_{[j]}]$ for $j = 1, \ldots, n$, satisfying $S_{[j]} \geq C_{[j-1]} = S_{[j-1]} + p_{[j-1]}$ 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 $d_j$. Accordingly, the lateness of job $J_{[j]}$ is defined by $L_{[j]} = C_{[j]} - d_j$. Moreover, we say that job $J_{[j]}$ is a $JIT$ job if $C_{[j]} = d_j$, and by $E = \{J_j \in J | C_{[j]} = d_j\}$, 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 $max_{J_j \in J} \{L_{[j]}\}$, while in the second problem we aim to find a solution that maximizes the weighted number of $JIT$ jobs, given by $\Sigma_j \in E w_j$, 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 $1|gdd| max_{J_j \in J} \{L_{[j]}\}$ and the second problem by $1|gdd| \Sigma_j \in E w_j$. We note that $\{L_{[j]}\} = max\{E_{[j]}, T_{[j]}\}$, where $E_{[j]} = max\{0, d_j - C_{[j]}\}$ is the earliness of job $J_{[j]}$, and $T_{[j]} = max\{0, C_{[j]} - d_j\}$ is the tardiness of job $J_{[j]}$. By $1|gdd| max_{J_j \in J} \{L_{[j]}\}$ and $1|\Sigma_j \in E w_j$, we refer to the variant of the same problems where due-dates are job-related.

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

1.2 Research objectives and paper organization

The reductions used by Tanaka and Vlach (1999) and Gerstl and Mosheiov (2020) to prove that problems $1|gdd| max_{J_j \in J} \{L_{[j]}\}$ and $1|gdd| |E|$ are strongly $NP$-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 $\nu_d$ 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 $\nu_d$ is upper bounded by a constant. If not, we aim to determine whether the problem remains $NP$-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.) $\nu_d$ in the sense of parameterized complexity.

Given an $NP$-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 $1|gdd| max_{J_j \in J} \{L_{[j]}\}$ and $1|gdd| \Sigma_j \in E w_j$ wrt. $\nu_d$. We prove that problems $1|gdd| max_{J_j \in J} \{L_{[j]}\}$ and $1|gdd| |E|$ are both $NP$-hard even if $\nu_d = 2$. Unless $P=NP$, this result rules out the possibility to construct FPT algorithms for problems $1|gdd| max_{J_j \in J} \{L_{[j]}\}$ and $1|gdd| |E|$ wrt. $\nu_d$. We also provide pseudo-polynomial time algorithms to solve problems $1|gdd| max_{J_j \in J} \{L_{[j]}\}$ and $1|gdd| \Sigma_j \in E w_j$ when $\nu_d$ 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 \( v_d \), while they both become ordinary \( \mathcal{NP} \)-hard when \( v_d \) is upper bounded by a constant. As both problems do not admit an \( \text{FPT} \) algorithm wrt. \( v_d \), we further analyze the case where we combine parameter \( v_d \) with another parameter, \( v_p \), which represents the number of different processing times in the instance. We provide \( \text{FPT} \) algorithms for problems \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) and \( \{ |gdd|, |E| \} \) wrt. the combined parameter. A summary and future research section concludes our paper.

## 2 Maximal absolute lateness

The \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) 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., \( v_d = 1 \)), a simple \( O(n) \) time algorithm provides the optimal schedule for the corresponding \( |gdd|, \delta = \delta \max_{J \in \mathcal{J}} \{ |L_J| \} \) problem (see Cheng, 1987). In the following three subsections, we complement these two results by showing that the \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) 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 \( v_d \) is upper bounded by a constant; and (iii) fixed parameterized tractable (\( \text{FPT} \)) when we combine the two parameters consisting of the number of different due-dates (\( v_d \)) and number of different processing times (\( v_p \)).

### 2.1 \( \mathcal{NP} \)-hardness for the two distinct due-dates case

In this subsection, we show that the \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) is \( \mathcal{NP} \)-hard even when \( v_d = 2 \). The reduction is done from the ordinary \( \mathcal{NP} \)-hard \text{EQUAL SIZE PARTITION} problem, which is defined below.

**Definition 1** \text{EQUAL SIZE PARTITION}: Given a set of \( 2t \) positive integers \( \mathcal{A} = \{ a_1, a_2, \ldots, a_{2t} \} \) with \( \sum_{j=1}^{2t} a_j = 2B \) and \( 0 < a_j < B \). Can \( \mathcal{A} \) be partitioned into two subsets \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) such that \( \sum_{a_j \in \mathcal{A}_1} a_j = B \) and \( |\mathcal{A}_i| = t \) for \( i = 1, 2 \)?

**Theorem 1** The \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) is \( \mathcal{NP} \)-hard even when there are only two distinct due-dates (i.e., even when \( v_d = 2 \)).

**Proof** Given an instance for the \( \mathcal{NP} \)-hard \text{EQUAL SIZE PARTITION} problem, we construct the following instance for our \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) problem: The instance includes \( n = 2t + 2 \) jobs. The processing times are:

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

and the due-dates are

\[
\delta_j = \begin{cases} 
    \frac{\delta}{2} = 2.5B & \text{for } j = 1, \ldots, t + 1 \\
    \frac{3}{2} = 5.5B & \text{for } j = t + 2, \ldots 2t + 2
\end{cases}
\]

In the decision version of the problem, we ask whether there exists a feasible schedule with \( \max \{ |L_J| \} \leq B/2 \). Note that there are only two distinct due-dates in the constructed instance of the \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) problem.

Given a solution that provides a \text{YES} answer for the \text{EQUAL SIZE PARTITION}, we construct the following solution \( S \) to our \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) problem. We set \( J_1 = \{ J_j | a_j \in A_1 \} \) for \( i = 1, 2 \). Then, we schedule the jobs in the following order: \( J_{2t+1}, J_1, J_{2t+2}, J_2 \) with no machine idle times. The processing order in each \( J_i \) (\( i = 1, 2 \)) is arbitrary. Since \( |J_1| = |A_1| = t \), all jobs in \( J_{2t+1} \cup J_1 \) share the same due-date of \( \delta = 2.5B \), and all jobs in \( J_{2t+2} \cup J_2 \) share the same due-date of \( \delta = 5.5B \). 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 \( J_1 \) is scheduled during time interval \((2B, 3B)\). Therefore, \( \max_{J_j \in J_1} \{ |L_J| \} = \max \{ |2B + p_{[2]} - 2.5B|, 3B - 2.5B \} \), \( |J_j| \) is the index of the \( j \)-th job in the processing order. The fact that \( p_{[2]} = a_{[2]} < B \) implies that \( \max_{J_j \in J_1} \{ |L_J| \} = 3B - 2.5B = 0.5B \).
- Job \( J_{2t+2} \) is scheduled during time interval \((3B, 5B)\). Thus, \( |L_{2t+2}| = |5B - 5.5B| = 0.5B \).
- Job set \( J_2 \) is scheduled during time interval \((5B, 6B)\). Therefore, \( \max_{J_j \in J_2} \{ |L_J| \} = \max \{ |5B + p_{[t+3]} - 5.5B|, 6B - 5.5B \} \). The fact that \( p_{[t+3]} = a_{[t+3]} < B \) implies that \( \max_{J_j \in J_2} \{ |L_J| \} = 6B - 5.5B = 0.5B \).

It follows that in schedule \( S \), \( \max_{J_j \in J} \{ |L_J| \} = 0.5B \), and we have a \text{YES} answer for the constructed instance of our \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) problem.

Consider next a solution \( S \) that provides a \text{YES} answer for the constructed instance of our \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) 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 \( \delta = 5.5B \), its absolute lateness equals to \( 0.5B + \Delta > 0.5B \), contradicting the fact that \( S \) provides a \text{YES} answer for the \( \{ |gdd|, \max_{J \in \mathcal{J}} \{ |L_J| \} \} \) problem.

Now, let \( J_1 \) be the set of \( t + 1 \) jobs that are scheduled first in \( S \), and let \( J_2 \) be the set of \( t + 1 \) jobs that are scheduled last in \( S \). All jobs in \( J_1 \) share the same due-date of \( \delta = 2.5B \), and all jobs in \( J_2 \) share the same due-date of \( \delta = 5.5B \).

**Lemma 2** Set \( J_1 \) includes exactly a single job out of the pair \( \{ J_{2t+1}, J_{2t+2} \} \).
In this section, we develop 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 $1|\text{gdd}|\max\{|L_j|\}$ problem when the number of different due-dates, $v_d$, is bounded by a constant.

We assume that $\delta_1 < \delta_2 < \cdots < \delta_{v_d}$, and begin with few notations. Let $m_i$ denote the number of positions in the job processing order having a due-date of $\delta_i$ for $i = 1, \ldots, v_d$. Moreover, let $l_r = \sum_{i=1}^{r} m_i$ denote the number of positions having due-date not greater than $\delta_r$ for $r = 1, \ldots, v_d$, with $l_{v_d} = n$ by definition. Given a solution, by $\mathcal{J}_{i}$ we denote the set of $m_i$ jobs assigned to due-date $\delta_i$ ($i = 1, \ldots, v_d$). Our pseudo-polynomial time algorithm exploits the properties in the following two lemmas:

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

**Proof** Consider an optimal solution $S^*$, which includes machine idle time of duration $\Delta$ between the processing of jobs $J_{i+1}$ and $J_{i+1}$ both belong to the same set $\mathcal{J}_i$ (i.e., $i \in \{u-1+1, \ldots, l_i\}$). Define an alternative solution $S'$ out of $S^*$, by starting the processing of jobs $J_{i+1}, \ldots, J_{i+1}$ $\Delta$ time units earlier (i.e., by eliminating the idle time between jobs $J_{i+1}$ and $J_{i+1}$, 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 $J_{i+1}$ has the maximum earliness value among all jobs in $\mathcal{J}_i$, while job $J_{i+1}$ has the maximal tardiness value among all jobs in $\mathcal{J}_i$. As the schedule of job $J_{i+1}$ 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 $J_{i+1}$ is completed $\Delta$ earlier in $S'$ compared 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 $1|\text{gdd}|\max\{|L_j|\}$ 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 $J_{i+1}$, has the maximum earliness value among all jobs in $\mathcal{J}_i$, while the last scheduled job in $\mathcal{J}_i$, which is job $J_{i+1}$, 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 $S^*$

$$\max\{\{L_j\}\} = \max\{E_{i+1}, T_{i+1}\}$$

$$= \max\{\{0, \delta_i - A_i - p_{i+1}\}\}$$

$$\max\{0, A_i + P(\mathcal{J}_i) - \delta_i\}$$

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

Consider now a feasible partition of job set $\mathcal{J}$ into the subsets $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{v_d}$, and let $(i) P_i = \sum_{j \in \mathcal{J}_i} p_j$ represent the total processing time of all jobs assigned to set $\mathcal{J}_i$; and
(ii) and \( a_i = \max_{j \in J_i} p_j \) represent the maximal processing time among all jobs assigned to set \( J_i \). Given the values of \( p_i \) and \( a_i \) for \( i = 1, \ldots, v_d \), we can compute the optimal starting times of job sets \( J_1, J_2, \ldots, J_{v_d} \) 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 \( J_i \) on the single machine \((i = 1, \ldots, v_d)\), 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 as well

\[
S_i \geq S_{i-1} + p_{i-1} \quad \text{for} \quad i \in \{1, \ldots, v_d\} \tag{3}
\]

with \( S_0 = p_0 = 0 \) by definition. Due to Lemmas 4 and 5, and due to the fact that all jobs in \( J_i \) share the same due-date of \( \delta_i \), the maximal earliness value of a job belonging to \( J_i \) is equal to \( \max\{0, \delta_i - S_i - a_i\} \), and the maximal tardiness value of a job belonging to \( J_i \) is equal to \( \max\{0, S_i + p_i - \delta_i\} \). Therefore, we include the following set of constraints as well

\[
Z \geq \delta_i - S_i - a_i \quad \text{for} \quad i \in \{1, \ldots, v_d\}, \tag{4}
\]

and

\[
Z \geq S_i + p_i - \delta_i \quad \text{for} \quad i \in \{1, \ldots, v_d\}. \tag{5}
\]

Thus, given a feasible partition of job set \( J \) into the subsets \( J_1, J_2, \ldots, J_{v_d} \) represented by the vector \((p_1, p_2, \ldots, p_{v_d}, a_1, a_2, \ldots, a_{v_d})\), one can compute the optimal starting time of each of the sets \( J_1, J_2, \ldots, J_{v_d} \), and the minimal objective value in \( O(v_d^{3.5}) \) time.

It follows from Lemma 6 that we can solve our \( 1|\text{gdd}|\) problem by finding all possible \((p_1, p_2, \ldots, p_{v_d}, a_1, a_2, \ldots, a_{v_d})\) vectors, each representing at least a single feasible partition of \( J \) into the sets \( J_1, J_2, \ldots, J_{v_d} \). 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 \((p_1, p_2, \ldots, p_{v_d}, a_1, a_2, \ldots, a_{v_d})\) vectors representing feasible partitions of \( J \) into the sets \( J_1, J_2, \ldots, J_{v_d} \). 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 \((j, k_1, \ldots, k_{v_d}, p_1, \ldots, p_{v_d}, a_1, \ldots, a_{v_d})\) represent a feasible partition of job set \( \{J_1, \ldots, J_j\} \) into the sets \( J_1, J_2, \ldots, J_{v_d} \), where \( k_i \leq m_i \) represents the number of jobs assigned to \( J_i \) \((i \in \{1, \ldots, v_d\})\). Let \( \ell_j \) represent all possible states on job set \( \{J_1, \ldots, J_j\} \).

We initialize our state generation process by including a single state \((0, k_1 = 0, \ldots, k_{v_d} = 0, p_1 = 0, \ldots, p_{v_d} = 0, a_1 = 0, \ldots, a_{v_d} = 0) \in \ell_0 \). Then, for \( j = 1, \ldots, n \), we construct \( \ell_j \) from \( \ell_{j-1} \) as follows: starting from each \((j-1, k_1, \ldots, k_{v_d}, p_1, \ldots, p_{v_d}, a_1, \ldots, a_{v_d}) \in \ell_{j-1} \), we include at most \( v_d \) states in \( \ell_j \) each representing a feasible assignment of \( J_j \) into one of the sets, \( J_1, J_2, \ldots, J_{v_d} \). Accordingly, for \( r = 1, \ldots, v_d \), if \( k_r < m_r \) we assign job \( J_r \) to \( J_r \) and therefore include the state \((j, k_1', \ldots, k_{v_d}', p_1', \ldots, p_{v_d}', a_1', \ldots, a_{v_d}') \in \ell_j \) with

- (i) \( k_i' = k_i \) for \( i \in \{1, \ldots, v_d\} \setminus \{r\} \);
- (ii) \( k_r' = k_r + 1 \);
- (iii) \( p_i' = p_i + p_j \); and
- (iv) \( a_i' = a_i \) for \( i \in \{1, \ldots, v_d\} \setminus \{r\} \).

At the end of the state generation process, set \( \ell_n \) includes the set of all possible state vectors. To conclude the above analysis, the following algorithm can be used to solve the \( 1|\text{gdd}|\) problem:
Algorithm 1 An optimization algorithm for solving the 1 \mid gdd \mid \max \{ |L_j| \} problem.

**Input:** \( n, (\delta_1, \ldots, \delta_{vd}), (m_1, \ldots, m_{vd}), (p_1, \ldots, p_n) \).

**Initialization:**

Set \( L_0 = \{(0, k_1 = 0, \ldots, k_{vd} = 0), P_1 = 0, \ldots, P_{vd} = 0, \alpha_1 = 0, \ldots, \alpha_{vd} = 0\} \)
and \( L_j = \emptyset \) for \( j = 1, \ldots, n \). Set \( Opt = \emptyset \) and \( Opt_value = \infty \).

**Step 1:** Re-index the jobs according to the LPT rule such that \( p_1 \geq p_2 \geq \ldots \geq p_n \).

**Step 2:**

For \( j = 1 \) to \( n \) do

For any \( (j, k_1, \ldots, k_{vd}, P_1, \ldots, P_{vd}, \alpha_1, \ldots, \alpha_{vd}) \in L_{j-1} \) do

For \( r = 1 \) to \( v_d \) do

If \( k_r < m_r \) then

Include state \( (j, k'_1, \ldots, k'_{vd}, P'_1, \ldots, P'_{vd}, \alpha'_1, \ldots, \alpha'_{vd}) \) in \( L_j \), where

\[
  k'_i = k_i, P'_i = P_i \text{ and } \alpha'_i = \alpha_i \text{ for } i \in \{1, \ldots, v_d\} \setminus \{r\}; k'_r = k_r + 1; P'_r = P_r + p_j; \text{ and } \alpha'_r = \alpha_r \text{ if } \alpha_r > 0.
\]

End if

End For

End For

End For

**Step 3:** For any \((n, m_1, \ldots, m_{vd}, P_1, P_2, \ldots, P_{vd}, \alpha_1, \alpha_2, \ldots, \alpha_{vd}) \in L_n\) 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 \( Opt = (n, m_1, \ldots, m_{vd}, P_1, P_2, \ldots, P_{vd}, \alpha_1, \alpha_2, \ldots, \alpha_{vd}) \) and \( Opt_value = Z^* \).

**Step 4:** Track back the optimal assignment of jobs into the job sets \( J_1, J_2, \ldots, J_{vd} \) out of \( Opt \).

---

**Theorem 2** Algorithm 1 solves the 1 \mid gdd \mid \max \{ |L_j| \} problem in pseudo-polynomial time when the value of \( v_d \) 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 \( P_i \) \((i \in \{1, \ldots, v_d\})\) is upper bounded by \( P_{\Sigma} = \sum_{j \in J} P_j \), and that (ii) there are \( O(n) \) different possible values for each \( \alpha_i \), implies that there are at most \( O((n P_{\Sigma})^{v_d}) \) states in each \( L_j \). In Step 2, we construct at most \( v_d \) states in \( L_j \) out of \( n \) states in \( L_{j-1} \), each of which requires \( O(v_d) \) time. Thus, each iteration \( j \in \{1, \ldots, n\} \) of Step 2 requires \( O((v_d)^2(n P_{\Sigma})^{v_d}) \) time, and the overall complexity of Step 2 is \( O((v_d)^2 n^{v_d+1}(P_{\Sigma})^{v_d}) \). In Step 3, for each state in \( L_n \), we solve an LP formulation, which requires \( O(v_d^3) \) time (see Lemma 6). The fact that we have \( O((n P_{\Sigma})^{v_d}) \) states in each \( L_j \), implies that Step 3 can be done in \( O(v_d^3(n P_{\Sigma})^{v_d}) \) time. In Step 4 we track back the optimal assignment of jobs into the job sets \( J_1, J_2, \ldots, J_{vd} \) out of \( Opt \). This can easily be done in a linear time, and therefore the time complexity of Algorithm 1 is \( O((v_d)^2 \max\{n, (v_d)^{1.5}\}(n P_{\Sigma})^{v_d}) \).

If the value of \( v_d \) is upper bounded by a constant, this time complexity reduces to \( O(n^{v_d+1}(P_{\Sigma})^{v_d}) \), which is pseudo-polynomial. \( \square \)

---

2.3 An FPT algorithm for the combined parameter \((v_d, \nu_p)\)

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

**Theorem 3** The 1 \mid gdd \mid \max_{j \in J} \{ |L_j| \} problem is FPT wrt. the combined parameter \((v_d, \nu_p)\).

The proof of Theorem 3 is done by providing a Mixed Integer Linear Programming (MILP) formulation for the 1 \mid gdd \mid \max_{j \in J} \{ |L_j| \} problem with \( O(v_d \nu_p) \) 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 \( v_d \): \((\delta_1, \delta_2, \ldots, \delta_{vd})\) and \((m_1, m_2, \ldots, m_{vd})\). The first includes the \( v_d \) distinct due-dates in the instance. In the second, \( m_i \) represents the number of positions in the sequence having a due-date of \( \delta_i \). We also represent the vector of \( n \) processing times in a modified manner by two vectors of size \( \nu_p \): \((p_1, p_2, \ldots, p_{\nu_p})\) and \((n_1, n_2, \ldots, n_{\nu_p})\), where the first includes the \( \nu_p \) 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 \( \delta_1 < \delta_2 < \ldots < \delta_{vd} \), and that the processing
times are numbered according to the SPT rule, such that $p_1 < p_2 < \cdots < p_{v_p}$.

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

$$\sum_{i=1}^{v_d} x_{ij} = n_j \text{ for } j = 1, \ldots, v_p, \quad (6)$$

and

$$\sum_{j=1}^{v_p} x_{ij} = m_i \text{ for } i = 1, \ldots, v_d \quad (7)$$

Now, let $S_i$ ($i = 1, \ldots, v_d$) 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 $S_i + \sum_{j=1}^{v_p} p_j x_{ij}$. Therefore, we include the following set of constraints that ensures that processing time intervals do not overlap:

$$S_{i+1} \geq S_i + \sum_{j=1}^{v_p} p_j x_{ij} \text{ for } i = 0, \ldots, v_d - 1, \quad (8)$$

with $S_0 = 0$ by definition.

Let $Z$ be a continuous decision variable representing the maximum absolute lateness, and let $\alpha_i$ be a continuous decision variable representing the largest processing time among all processing times of the jobs in $\mathcal{J}_i$ ($i \in \{1, \ldots, v_d\}$). 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

$$Z \geq \delta_i - S_i - \alpha_i \text{ for } i = 1, \ldots, v_d. \quad (9)$$

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

$$Z \geq S_i + \sum_{j=1}^{v_p} p_j x_{ij} - \delta_i \text{ for } i = 1, \ldots, v_d. \quad (10)$$

To ensure that $\alpha_i$ 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 ($i \in \{1, \ldots, v_d\}, j \in \{1, \ldots, v_p\}$). Then, we include the following set of constraints:

$$y_{ij} \leq x_{ij} \text{ for } i = 1, \ldots, v_d \text{ and } j = 1, \ldots, v_p; \quad (11)$$

and also the following set of constraints:

$$\alpha_i \leq p_j + \sum_{l=j+1}^{v_p} \nu_l y_{il} \text{ for } i = 1, \ldots, v_d \text{ and } j = 1, \ldots, v_p. \quad (12)$$

Now, consider the problem $P$ 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 $P$

$$Z = \max_{i=1, \ldots, v_d} \{\delta_i - S_i - \alpha_i, S_i + \sum_{j=1}^{v_p} p_j x_{ij} - \delta_i\}. \quad (13)$$

which is exactly the value of the maximal absolute lateness of any solution that satisfy the properties in Lemmas 4 and 5 if indeed $\alpha_i$ 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 $\alpha_i$ for $i = 1, .., n$. Therefore, there exists an optimal solution for problem $P$, where $\alpha_i$ 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 $y_{ij}$ 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 $\mathcal{J}_i$ ($q \in \{1, \ldots, v_p\}$). It follows that $y_{iq} = 1$, and that $y_{ij} = 0$ for $j = q + 1, \ldots, v_p$. Therefore, $p_j + \sum_{l=j+1}^{v_p} \nu_l y_{il} > p_q$ for any $j \in \{1, \ldots, q - 1\}, p_q + \sum_{l=q+1}^{v_p} \nu_l y_{il} = p_q$ and $p_j + \sum_{l=j+1}^{v_p} \nu_l y_{il} > p_q$ for any $j \in \{q + 1, \ldots, v_p\}$. 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 $\alpha_i$ for $i = 1, .., n$, there exists an optimal solution for $P$ in which $\alpha_i = p_q$.

The fact that solving $P$ provides an optimal solution for the $1|\text{gdd}|\max_{J \in \mathcal{J}}\{|L_j|\}$ problem, and that $P$ includes $O(v_d v_p)$ integer variables, implies that we can use Lenstra’s algorithm (1983) algorithm to solve the $1|\text{gdd}|\max_{J \in \mathcal{J}}\{|L_j|\}$ problem wrt. the combined parameter $(v_d, v_p)$, 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 $1|\text{gdd}||\mathcal{E}|$ 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 $\mathcal{J}' = \{J_j \in \mathcal{J} \mid p_j \leq \delta\}$ 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 \(1|\text{gdd}|\sum_{j \in E} w_j\) problem is solvable in \(O(n)\) time when \(v_d = 1\).

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

### 3.1 \(\mathcal{NP}\)-hardness for the two distinct due-dates case

In this subsection, we show that the \(1|\text{gdd}||E|\) is \(\mathcal{NP}\)-hard even when \(v_d = 2\). 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 \(A = \{a_1, a_2, \ldots, a_t\}\) with \(\sum_{j=1}^{t} a_j = 2B\) and \(0 < a_j < B\). Can \(A\) be partitioned into two subsets \(A_1\) and \(A_2\) such that \(\sum_{j \in A_1} a_j = B\) for \(i = 1, 2\)?

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

**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 \(J = \{J_1, \ldots, J_n\}\) of \(n = t\) jobs. The processing times are

\[
p_j = a_j \quad (14)
\]

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

\[
\delta_j = \begin{cases} 
B & \text{for } j = 1, \ldots, t-1 \\
2B & \text{for } j = t 
\end{cases} \quad (15)
\]

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

Consider a solution that provides a YES answer for the PARTITION problem, we construct the following solution \(S\) to our \(1|\text{gdd}||E|\) problem. We set \(J_i = \{J_j | a_j \in A_i\}\) for \(i = 1, 2\). Then, we schedule all jobs in \(J_1\) before any job in \(J_2\) with no machine idle times. The processing order within each \(J_i\) \((i = 1, 2)\) is arbitrary. As \(\sum_{j \in J_i} p_j = \sum_{j \in A_i} a_j = B\), the completion time of the last job in \(J_1\) is exactly at time \(B\). Moreover, the fact that \(|J_1| < n\), implies that the last job to be scheduled in \(J_1\) is assigned with a due-date of \(B\) (see eq. (15)). Therefore, the last scheduled job in \(J_1\) is completed in a JIT mode. Moreover, as we schedule all jobs with no idle times, the last job to be scheduled in \(J_2\) 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 \(|E| = 2\), and therefore \(S\) provides a YES answer for the constructed instance of our \(1|\text{gdd}||E|\) problem.

Consider next a solution \(S\) that provides a YES answer for the constructed instance of our \(1|\text{gdd}||E|\) 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 \(|E| = 2\) in \(S\) implies that there is job which completes exactly at time \(B\) in schedule \(S\). Let \(J_1\) be the set of jobs that are completed no later than time \(B\) in \(S\), and let \(J_2\) be the set of all other jobs. As there are no machine idle time in \(S\), we have that \(\sum_{j \in J_1} p_j = B\) for \(i = 1, 2\). Therefore, by setting \(A_i = \{a_j | J_j \in J_i\}\) for \(i = 1, 2\), we have a solution that provides a YES answer for the PARTITION problem. \(\square\)

### 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 \(r \leq v_d\)). There are \(O(n^{v_d})\) 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 \(1|\text{gdd}|\sum_{j \in E} w_j\) 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 \(1|\text{gdd}|\sum_{j \in E} w_j\) 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 \(A_i\) being the set of \(n_i\) jobs scheduled in positions \(\{l_i-1, \ldots, l_i\}\) for \(i = 1, \ldots, r + 1\). Note that all jobs in each \(A_i\) \((i = 1, \ldots, r)\) share the same due-date of \(\delta_{l_i}\). Accordingly, only a single job in each \(A_i\) \((i = 1, \ldots, r)\) can be completed at the common due-date, \(\delta_{l_i}\). It follows that \(\sum_{j \in A_i} p_j \leq \delta_{l_i} - \delta_{l_i-1}\) for \(i = 1, \ldots, r\), as otherwise we cannot complete the last job in \(A_i\) at time \(\delta_{l_i}\), given that the last job in \(A_{i-1}\) 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 \(A_i\), and with \(\sum_{j \in A_i} p_j \leq \delta_{l_i} - \delta_{l_i-1}\) for \(i = 1, \ldots, r\), it is optimal to schedule all jobs in \(A_i\) during
time interval \( (\delta_i - \sum_{j \in A_i} p_j, \delta_i] \) with the last job, which is the only one in \( A_i \) being scheduled in a JIT mode, is the one of maximal weight among all jobs in \( A_i \).

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 \( F_j(x_1, \ldots, x_r, q_1, \ldots, q_r, q_{r+1}) \) be the maximal gain that can be obtained out of all feasible partial schedule on job set \( J_j = \{ J_1, \ldots, J_r \} \) with \( q_r \) jobs the assigned to set \( A_i \) and \( x_i = \sum_{j \in A_i} p_j \) for \( i = 1, \ldots, r+1 \). It follows from the feasibility of the partial schedule that the following conditions holds:

**Condition 1** \( x_i \leq \delta_i - \delta_{i-1} \) for \( i = 1, \ldots, r \), as otherwise we cannot complete the last job in \( A_i \) at time \( \delta_i \), given that the last job in \( A_{i-1} \) is completed at time \( \delta_{i-1} \).

**Condition 2** \( x_{r+1} = \sum_{i=1}^{j} p_i - \sum_{i=1}^{r} x_i \), and \( q_{r+1} = j - \sum_{i=1}^{r} q_i \) if any job has to be assigned to one of the \( A_i \) set (\( i = 1, \ldots, r+1 \)).

**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 \( A_i \) 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:

\[
F_j(x_1, \ldots, x_{r+1}, q_1, \ldots, q_{r+1}) = \max_{i=1, \ldots, r+1} \left\{ \begin{aligned}
& F_{j-1}(x_1, \ldots, x_i - p_j, \ldots, x_{r+1}, q_1, \ldots, q_i) & \text{if } i < r + 1 \\
& F_{j-1}(x_1, \ldots, x_i - p_j, -1, \ldots, q_{r+1}) + w_j & \text{and } q_i = 1 \\
& q_i - 1, \ldots, q_{r+1} & \text{otherwise}
\end{aligned} \right.
\]

with the initial condition that

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

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

\[
F^*(l) = \max \{ F_n(x_1, \ldots, x_{r+1}, n_1, \ldots, n_{r+1}) \mid x_i \leq \delta_i - \delta_{i-1} \text{ for } i = 1, \ldots, r \text{ and } \\
\sum_{i=1}^{r+1} x_i = \sum_{i=1}^{n} p_i \}
\]

To conclude, we can use the following algorithm to solve our \( 1 \mid gdd \mid \sum_{j \in E} w_j \) problem:

**Algorithm 2** An optimization algorithm for solving the \( 1 \mid gdd \mid \sum_{j \in E} w_j \) problem.

Initialization Set \( F^* = 0 \).

Step 1 Determine \( L \) 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) \( \delta_i - 1 < \delta_i \) for \( i = 1, \ldots, r \) with \( l_0 = 0 \) by definition.

Step 2

For any \( l \in L \) 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 \( x_i \leq \delta_i - \delta_{i-1} \), that \( j \geq \sum_{i=1}^{r} q_i \), that \( \sum_{i=1}^{j} p_i \geq \sum_{i=1}^{r} x_i \), and that \( q_i \leq n_i \) and \( q_i + (n - j + 1) \geq n_i \) for \( i = 1, \ldots, r \) do:

Compute \( x_{r+1} = \sum_{i=1}^{j} p_i - \sum_{i=1}^{r} x_i \), and \( q_{r+1} = j - \sum_{i=1}^{r} q_i \).

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 \( A_i \) 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 \( i = 1, \ldots, r \) schedule the jobs in each \( A_i \) during time interval \( (\delta_i - \sum_{j \in A_i} p_j, \delta_i] \) where the last schedule job in each interval \( (\delta_i - \sum_{j \in A_i} p_j, \delta_i] \) is the job of maximal weight (smallest index) among all jobs in \( A_i \). Schedule the jobs in \( A_{r+1} \) during time interval \( (\delta_i, \delta_i + \sum_{j \in A_i} p_j) \).
Theorem 6 Algorithm 2 solves the 1 |gdd| \( \sum_{i \in E} w_j \) problem in \( O(v_d n^{2v_d+1} \prod_{i=1}^{v_d} \delta_i) \).

Proof The fact that Algorithm 2 provides the optimal solution follows from the discussion in this section. Step 1 requires \( O(n^{v_d}) \) 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 \( r = O(n^d) \), that \( \delta_i - \delta_i-1 = O(\delta_i) \) and that \( q_i \leq n_i = O(n) \), implies that we compute \( O(n^{v_d+1} \prod_{i=1}^{v_d} \delta_i) \) 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 \( O(n^{v_d+1} \prod_{i=1}^{v_d} \delta_i) \) 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 \( O(v_d) \) 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 \( O(v_d n^{2v_d+1} \prod_{i=1}^{v_d} \delta_i) \). According to the \( F^*(l) \) by (18) can be done in \( O(v_d n^{2v_d+1} \prod_{i=1}^{v_d} \delta_i) \) time, applying Step 2 for a given \( l \) vector requires \( O(v_d n^{2v_d+1} \prod_{i=1}^{v_d} \delta_i) \) time. The fact that we repeat Step 2, for any \( l \in \mathcal{L} \), and that \( |\mathcal{L}| = O(n^{v_d}) \), implies that Step 2 requires \( O(v_d n^{2v_d+1} \prod_{i=1}^{v_d} \delta_i) \) time. This time complexity reduces to \( O(n^{v_d+1} \prod_{i=1}^{v_d} \delta_i) \) when the value of \( v_d \) 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. \( \square \)

3.3 An FPT algorithm for the 1 |gdd| |\( \mathcal{E} \)| problem wrt. the combined parameter (\( v_d, \nu_p \))

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

Theorem 7 The 1 |gdd| |\( \mathcal{E} \)| problem is FPT wrt. the combined parameter (\( v_d, \nu_p \)).

The proof of Theorem 7 is based on breaking down the 1 |gdd| |\( \mathcal{E} \)| problem into a set of \( O(2^{v_d}) \) subproblems and providing a MILP formulation with \( O(v_d \nu_p) \) 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 \( v_d \): \((\delta_1, \delta_2, \ldots, \delta_{v_d})\) and \((m_1, m_2, \ldots, m_{v_d})\). The first includes the \( v_d \) distinct due-dates in the instance. In the second, \( m_1 \) represents the number of positions in the sequence having a due-date not greater than \( \delta_1 \). We order the due-dates in the first vector according to the EDD rule, such that \( \delta_1 < \delta_2 < \cdots < \delta_{v_d} \). By definition, we also have that \( m_1 < m_2 < \cdots < m_{v_d} \) with \( m_{v_d} = n \). We also represent the vector of \( n \) processing times in a modified manner by two vectors of size \( \nu_p \): \((p_1, p_2, \ldots, p_{\nu_p})\) and \((n_1, n_2, \ldots, n_{\nu_p})\), where the first includes the \( \nu_p \) 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 \( \delta_i \) is active if there exists a job completed exactly at time \( \delta_i \). Now, let \( \Delta \) be a set that includes all subsets of set \((\delta_1, \delta_2, \ldots, \delta_{v_d})\). Note that \(|\Delta| = O(2^{v_d})\). We partition our 1 |gdd| |\( \mathcal{E} \)| problem into a set of \( O(2^{\nu_p}) \) 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 \( \delta \) being active. Let \( P(\delta) \) be the subproblem that corresponds to vector \( \delta \). 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 \( O(v_d \nu_p) \) integer variables. Let \( \delta = (\delta_1, \delta_2, \ldots, \delta_{v_d}) \) (note that \( k \leq v_d \)). We define \( x_i \) as a positive integer variable that represents the position in the sequence of the job completed at \( \delta_1 \) for \( i = 1, \ldots, k \). As there are exactly \( m_i \) positions with due-date not larger than \( \delta_i \), we include the following set of constraints for \( i = 1, \ldots, k \):

\[
\sum_{j=1}^{v_p} y_{ij} = x_i - x_{i-1},
\]

with \( x_i = 0 \) by definition. To make sure that \( \delta_1 \) is indeed an active due-date, we also include the following set of constraints for \( i = 1, \ldots, k \):

\[
\sum_{j=1}^{v_p} p_j y_{ij} \leq \delta_1 - \delta_{i-1}.
\]

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 \( j = 1, \ldots, v_p \):

\[
\sum_{i=1}^{k} y_{ij} \leq n_j.
\]

As \( k \leq v_d \), the above formulation includes \( O(v_d + v_d \nu_p) = O(v_d \nu_p) \) 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, \( v_d \), 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 \( 1|\text{gdd}|\max\{|L_j|\} \), focuses on minimizing the maximal absolute lateness, and the second, denoted by \( 1|\text{gdd}|\Sigma J_j \in \mathcal{E} w_j \), 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 \( \nu_d \). We show that both problems are solvable in polynomial time when \( \nu_d = 1 \). Moreover, we show that problems \( 1|\text{gdd}|\max\{|L_j|\} \) and \( 1|\text{gdd}|\Sigma J_j \in \mathcal{E} w_j \) are \( \mathcal{NP} \)-hard when \( \nu_d = 2 \). We compliment these two results by showing that problems \( 1|\text{gdd}|\max\{|L_j|\} \) and \( 1|\text{gdd}|\Sigma J_j \in \mathcal{E} w_j \) are solvable in pseudo-polynomial time when the value of \( \nu_d \) is bounded by a constant.

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

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 \( \nu_d \), hoping to provide efficient algorithms to solve practical instances with limited \( \nu_d \) values.

Acknowledgements The first author was supported by the Israel Science Foundation (Grant No.250519), 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.

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

References

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 \( \mathcal{NP} \)-hard. Mathematics of Operations Research, 15, 483–495.

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

Garey, M. R., Tanjan, R. E., & Wigfong, 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 \( \mathcal{NP} \)-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., & Srikantharajah, 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 University Press.

Srikantharajah, 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.

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