On the parameterized tractability of the just-in-time flow-shop scheduling problem

Danny Hermelin1 · Dvir Shabtay1 · Nimrod Talmon1

Published online: 14 August 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract
We study the parameterized complexity of a set of flow-shop scheduling problems in which the objective is to maximize the weighted number of just-in-time jobs. Our analysis focuses on the case where the number of due dates is considerably smaller than the number of jobs and thus can be considered as a parameter. We prove that the two-machine problem is W[1]-hard with respect to this parameter, even if all processing times on the second machine are of unit length, while the problem is in XP when parameterized by the number of machines combined with the number of different due dates. We then move on to study the tractability of the problem when combining the number of different due dates with either the number of different weights or the number of different processing times on the first machine. We prove that in both cases the problem is fixed-parameter tractable for the two-machine case and is W[1]-hard for three or more machines.

Keywords Scheduling · Flow-shop · Just-in-time · Parameterized complexity · Fixed-parameter tractability

1 Introduction

The concept of just-in-time (JIT) production has attracted the attention of many industries in the last 50 years and has been widely adopted over the years to improve production efficiency in many industries (see, e.g., White and Prybutok 2001 and Fullerton and McWatters 2001). One of the main concepts in JIT production systems is to make sure that customer orders (jobs to be produced) are completed exactly when required, avoiding unnecessary inventory and late delivery costs. Therefore, in the field of JIT scheduling, the objective to be minimized includes penalties for both early and tardy completion of jobs.

Two main types of such cost functions are described in the literature. In the first, there is a penalty for each job that is proportional to the deviation of its completion time from the required due date. In the second, jobs that are not completed exactly at due date incur a fixed penalty. Scheduling problems with a cost function of the first type are commonly known as earliness-tardiness scheduling problems [see Baker and Scudder (1990) for a survey], while problems with a cost function of the second type are commonly known as JIT scheduling problems [see Shabtay and Steiner (2012) for a survey]. It should be noted that the set of JIT scheduling problems on a single and parallel machines forms a special case of another important set of scheduling problems, the set of fixed interval scheduling problems [see Kovalyov et al. (2007) for a survey].

The JIT scheduling problem is solvable in polynomial time on a single machine (see Lann and Mosheiov 1996) and on various parallel machine systems (see, e.g., Arkin and Silverberg 1987; Carlisle and Lloyd 1995 and Čepek and Sung 2005). However, this is not necessary the case when we move to more sophisticated scheduling systems such as flow-shop, job-shop and open-shop (see Choi and Yoon 2007 and Shabtay and Bensoussan 2012). The NP-hardness proofs of these JIT scheduling problems (that appear in Choi and Yoon 2007 and Shabtay and Bensoussan 2012) heavily depend on the fact that several parameters of the problem (such as the number of different processing times and the number of different due dates) may be arbitrarily large.

Such an assumption, however, may be over-pessimistic in many real-life scheduling problems. For example, in many
cases the manufacturer produces only a predefined set of different products, yielding instances with a limited number of different processing times. As another example, in many production systems the manufacturer limits the number of due dates to only a small number of predefined delivery dates, resulting in instances with only limited number of different due dates. Therefore, it is quite natural to ask whether any of the NP-hard variants of the JIT scheduling problem becomes tractable when some of its natural parameters are comparatively small.

In this paper, we study the parameterized tractability of the JIT scheduling problem in a flow-shop scheduling system. We do so by using the theory of parameterized complexity. The main idea in parameterized complexity is to analyze the tractability of NP-hard problems with respect to (wrt.) various instance parameters that are independent of the total input length. For other papers analyzing scheduling problems with parameterized complexity, see, e.g., Bodlaender and Fellows (1995), Fellows and McCartin (2003), Hermelin et al. (2015), Mnich and Wiese (2015), van Bevern et al. (2015), van Bevern et al. (2016), and Mnich and van Bevern (2018).

Below, we continue the introduction by providing a brief exposition to the theory of parameterized complexity. Then, we formally define the scheduling problem we aim to analyze, survey the known results in the literature that are related to this problem, and present our research objectives.

1.1 Basic concepts in parameterized complexity theory

The main objective in parameterized complexity theory is to analyze the tractability of NP-hard problems wrt. their natural parameters. Given an instance for a problem \( \pi \), a parameter in this sense measures some structural properties of the instance. For example, in a scheduling problem that involves processing times, due dates, and weights, a parameter could be the number of different due dates, the maximal processing time, due dates, and weights, a parameter could be the number of different due dates, the maximal processing time, due dates. Therefore, it is quite natural to ask whether any of the NP-hard variants of the JIT scheduling problem becomes tractable when some of its natural parameters are comparatively small.

Let \( n \) denote its instance size, and \( k \) be a predefined parameter. The following definitions lays at the core of parameterized complexity theory:

**Definition 1** Problem \( \pi \) is fixed-parameter tractable (FPT) wrt. a predefined parameter \( k \) if there is an algorithm that solves any instance of \( \pi \) in \( f(k)n^{O(1)} \) time, for some computable function \( f \) that solely depends on \( k \). An algorithm running in this time is said to be a fixed-parameter algorithm.

**Definition 2** Problem \( \pi \) belongs to the XP class wrt. a predefined parameter \( k \) if there is an algorithm that solves any instance of \( \pi \) in \( n^{f(k)} \) time, for some computable function \( f \) that solely depends on \( k \). A running time of \( n^{f(k)} \) is also known as XP time, and an algorithm running in this time is said to be an XP algorithm.

Note that the advantage of a fixed-parameter algorithm over an XP algorithm is that the exponent on \( n \) stays fixed while \( k \) grows, making this type of algorithm much more efficient in the asymptotic sense. In particular, it is easy to show that FPT, the class of all parameterized problems which admit fixed-parameter algorithms, is contained in XP. There are parameterized analogues of NP-hardness which can be used to show that a problem is presumably not fixed-parameter tractable. Similarly to the reductions used to show NP-hardness of some (nonparameterized) problem, the standard technique for showing parameterized hardness is through a parameterized reduction.

**Definition 3** A parameterized reduction from a problem \( \pi_2 \) to a problem \( \pi_1 \) is an algorithm mapping an instance \( I_2 \) of \( \pi_2 \) with a parameter value \( k_2 \) to an instance \( I_1 \) of \( \pi_1 \) with a parameter value \( k_1 \) in time \( f(k_2)|I_2|^{O(1)} \) such that \( k_1 \leq f(k_2) \) and \( I_1 \) is a yes-instance for \( \pi_1 \) if and only if \( I_2 \) is a yes-instance for \( \pi_2 \).

There are several classes of problems that are conjectured not to be fixed-parameter tractable, with the class of W[1]-hard problems and the class of W[2]-hard problems being the most popular. Indeed, for any \( i \), the fact that a decision problem \( \pi \) is W[\( i \)]-hard wrt. parameter \( k \) implies that \( \pi \) is not fixed-parameter tractable wrt. \( k \) unless FPT=W[\( i \)] (which is widely believed to be unlikely).

For proving that a problem is W[1]-hard, one can provide a parameterized reduction from a known W[1]-hard problem, such as the CLIQUE problem, with the parameter being the size of the clique sought. Similarly, for proving that a problem is W[2]-hard one can provide a parameterized reduction from a known W[2]-hard problem, such as the SET COVER problem, with the parameter being the number of sets in the cover sought. (While W[1] and W[2] are potentially different complexity classes, they both presumably rule out the existence of a fixed-parameter algorithm and for our point of view, similarly to the point of view of most papers studying parameterized complexity, the difference between them is of no special significance.)

For more details about the theory of parameterized complexity, we refer the reader to Cygan et al. (2015), Flum and Grohe (1998), Downey and Fellows (2013) and Niedermeier (2006).

1.2 Problem definition

The aim of the current paper is to study the JIT flow-shop scheduling problem from a parameterized complexity perspective. This problem can be defined as follows. We are given a set of \( n \) independent, non-preemptive jobs,
\( J = \{J_1, J_2, \ldots, J_n\} \), which are available for processing at time zero and are to be scheduled on a set of \( m \) machines \( \mathcal{M} = \{M_1, M_2, \ldots, M_m\} \). The machines are arranged in a flow-shop machine setting. This means that the following restrictions holds: (i) all jobs have to be processed on all machines, first on \( M_1 \), then on \( M_2 \), and so on, up to \( M_m \); (ii) each machine can process only one job at a time; and (iii) each job can be processed only on one of the machines at any moment in time. We assume that apart from those three restrictions, there are no other technological restrictions, such as no-wait restrictions and/or constraints on the size of the buffer between the machines.

For \( i = 1, \ldots, m \) and \( j = 1, \ldots, n \), let \( p_j^{(i)} \) be the processing time of job \( J_j \) on machine \( M_i \). We use \( d_j \) to represent the due date of job \( J_j \) and \( w_j \) to represent the gain (income) of completing job \( J_j \) in a JIT mode (i.e., completing job \( J_j \) on \( M_m \) exactly at time \( d_j \)). We assume that all the \( d_j \), \( w_j \), and \( p_j^{(i)} \) values are positive integers.

For a JIT scheduling problem, a partition of the set \( J \) into two disjointed subsets \( E \) and \( T \) is considered to be a feasible partition (or a feasible schedule), if it is possible to schedule jobs belonging to the set \( E \) such that they are all completed in JIT mode. We also say that the set \( E \) of a feasible partition is a feasible set of JIT jobs. Without loss of generality, we assume that jobs belonging to set \( T \) of a feasible partition are rejected (i.e., not processed at all). The objective is to find a feasible partition with a maximal weighted number of jobs in the set \( E \), that is, to maximize \( \sum_{J_j \in E} w_j \).

Following the classical three-field notation of Graham et al. (1979), we denote our JIT flow-shop scheduling problem by \( F_m || \sum_{J_j \in E} w_j \) problem, where \( F_m \) in the first field indicates that the scheduling is done in a flow-shop scheduling system with \( m \) machines and the objective to be maximized appears in the last field. Note that the middle (second) field is empty, while in general it is used to specify specific processing characteristics and constraints. By \( F_m || E \), we refer to the special case where weights are identical, and the problem is simply to maximize the number of JIT jobs.

### 1.3 Related work

The first to study the \( F_m || \sum_{J_j \in E} w_j \) problem were Choi and Yoon (2007). By a reduction from the partition problem, they proved that the two-machine case is NP-hard. They also show that the unweighted version of the problem is solvable in polynomial time (specifically, \( O(n^4) \) time) for two machines and that it is strongly NP-hard for three machines. Shabtay and Bensoussan (2012) provided a \( O(n^2 \sum_{i=1}^{m} p_j^{(1)}) \) pseudo-polynomial time algorithm for the solution of the \( F_2 || \sum_{J_j \in E} w_j \) problem. In the same paper, it is also shown how the pseudo-polynomial time algorithm can be converted into a fully polynomial time approximation scheme (FPTAS), which provides \((1 + \varepsilon)\) approximation in \( O((n^4/\varepsilon) \log (n/\varepsilon)) \) time. The latter result was later improved by Elalouf et al. (2013), who accelerated the running time of the FPTAS by a factor of \( n \log (n/\varepsilon) \).

Shabtay (2012) provided an \( O(n^3) \) time algorithm for solving the \( F_2 || |E| \) problem, improving an earlier result of Choi and Yoon (2007) by a factor of \( n \). Shabtay (2012) also show that (i) the problem on \( m \) machines with machine-independent processing times is weakly NP-hard in general and is solvable in \( O(n^4) \) time if all weights are identical; (ii) if the processing times are job-independent (but machine-dependent), then the \( m \) machine problem is solvable in \( O(n^3) \) time for arbitrary weights and in \( O(n \log n) \) time if all weights are identical; and (iii) the \( m \) machine problem is solvable in \( O(nm^2) \) time if there is a no-wait restriction, i.e., if jobs are not allowed to wait between machines.

### 1.4 Research objective and roadmap

Our main research objective is to study the tractability of the hard variations of the \( F_m || \sum_{J_j \in E} w_j \) problem wrt. the number of different due dates, \#\( d \), of a given instance. In Sect. 2, we provide some important properties of feasible and optimal schedules. In Sect. 3, we prove that the two-machine problem is \( \text{W}[1]-hard \) wrt. this parameter, even if all processing times in the second machine are of unit size. We also prove that the more general \( m \)-machine problem belongs to the \( \text{XP class when parameterized by the number of machines combined with the number of different due dates.} \)

We further continue to consider combining the number \#\( d \) of different due dates (as a parameter) with two other parameters: The number of different processing times on the first machine (\#\( p^1 \)) and the number of different weights (\#\( w \)). In Sect. 4, we prove, for both of these combined cases, that the two-machine problem is fixed-parameter tractable; in Sect. 5, we prove that the corresponding three-machine problem is \( \text{W}[1]-hard \). A summary and future research section concludes our paper (our results are also summarized in Table 1).

### Table 1 Summary of our results

| \( F_2 || \sum_{J_j \in E} w_j \) | \( F_3 || \sum_{J_j \in E} w_j \) |
|-----------------|-----------------|
| \#\( d \) | XP (Corollary 2) | XP (Corollary 2) |
| \#\( d \) + \#\( w \) | FPT (Corollary 3) | \( \text{W}[1]-hard \) (Theorem 5) |
| \#\( d \) + \#\( p^1 \) | FPT (Theorem 3) | \( \text{W}[1]-hard \) (Theorem 4) |
| \#\( d \) + \#\( p^2 \) | \( \text{W}[1]-hard \) (Theorem 1) |

The number of different due dates is denoted by \#\( d \), the number of different processing times in the first (second) machine is denoted by \#\( p^1 \) (\#\( p^2 \)), respectively, and the number of different weights is denoted by \#\( w \).
2 General properties

Let $\sigma^E$ be a permutation of the jobs in set $E$, representing the order in which the jobs in $E$ are processed on machine $M_i$, for $i = 1, \ldots, m$. A job schedule is called a permutation schedule if $\sigma_1^E = \sigma_2^E = \cdots = \sigma_m^E$. The following two lemmas are due to Choi and Yoon (2007).

**Lemma 1** If $E$ is a feasible set of JIT jobs for the $F_m|| \sum_{j \in E} w_j$ problem, then there exists a feasible schedule of the jobs in $E$ in which $\sigma_1^E = \sigma_2^E$.

**Lemma 2** In any feasible schedule for the $F_m|| \sum_{j \in E} w_j$ problem, $\sigma_m^E$ follows the earliest due date (EDD) order, i.e., in $\sigma_m^E$ the jobs are ordered in a non-decreasing order of due dates.

It follows from Lemmas 1 and 2 that there exists an optimal permutation schedule for the $F_2|| \sum_{j \in E} w_j$ problem in which both $\sigma_1^E$ and $\sigma_2^E$ follow the EDD rule. When $m \geq 3$, however, the optimal schedule is not necessarily a permutation schedule; indeed, Choi and Yoon (2007) provided a counterexample showing that there exists a set of instances of the $F_3|| \sum_{j \in E} w_j$ problem in which the optimal permutation in $M_1$ and $M_2$ does not follow the EDD rule.

**Observation 1** If $E$ is a feasible set of JIT jobs, then there exists a feasible schedule of the jobs in $E$ in which jobs in machines $M_1, \ldots, M_{m-1}$ are scheduled as soon as possible, i.e., for $i = 1, \ldots, m-1$ and $j = 1, \ldots, |E|$: job $J_{j[1]}^{(i)}$ starts on $M_i$ at the time which is the maximum between (i) the completion of job $J_{j[1]-1}^{(i)}$ on $M_i$ and (ii) the completion of job $J_{j[1]}^{(i)}$ on $M_{i-1}$.

3 The complexity of JIT flow-shop wrt. $\#d$

In this section, we consider the parameterized complexity of the JIT flow-shop problem with respect to a single parameter, namely the number $\#d$ of different due dates.

3.1 The $W[1]$-hardness of the two-machine case

Consider the following definition of the $k$SUM problem.

**Definition ($k$SUM)** Given a set of $h$ positive integers $X = \{x_1, \ldots, x_h\}$, a positive integer $k$ ($1 \leq k < h$), and a positive integer $B$, determine whether there exists a set $S \subseteq X$ (possibly with repetitions) such that $|S| = k$ and $\sum_{x_i \in S} x_i = B$. Without loss of generality, we assume that $B < k \cdot \max_{x_i \in X} x_i$ and that $\max_{x_i \in X} x_i > 1$. Otherwise, the solution for the corresponding instance can be obtained in polynomial time.

It is known that $k$SUM is $W[1]$-hard wrt. $k$ (see Downey and Fellows 1992). Below we prove that the $F_2|| \sum_{j \in E} w_j$ problem is $W[1]$-hard wrt. $\#d$ by providing a parameterized reduction from the $k$SUM problem.

**Theorem 1** The $F_2|| \sum_{j \in E} w_j$ problem is $W[1]$-hard when parameterized by the number $\#d$ of different due dates even if all jobs have unit processing time on the second machine.

**Proof** Given an instance $\{(x_1, \ldots, x_k), (B, k)\}$ of $k$SUM, we construct an instance of the decision version of the $F_2|| \sum_{j \in E} w_j$ problem as follows. The set $E$ includes $n = kh + 1$ jobs. The set of first $kh$ jobs is the union of $k$ sets $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_k$, where each set $\mathcal{J}_i$ includes $h$ jobs corresponding to the $h$ elements in the $k$SUM instance. Let $J_{ij}$ be the $j$th job in the set $\mathcal{J}_i$. Moreover, for any job $J_{ij}$, let $d_{ij}, w_{ij}, p_{ij}^{(1)}$, and $p_{ij}^{(2)}$ be the due date, the weight, and the processing times on machines $M_1$ and $M_2$, respectively. For any job $J_{ij}$, set

- $d_{ij} = iZ$, where $Z = k \cdot \max_{x_i \in X} x_i$;
- $w_{ij} = x_j$;
- $p_{ij}^{(1)} = x_j$; and
- $p_{ij}^{(2)} = 1$.

For job $J_{kh+1}$, we set

- $d_{kh+1} = (k + 1)Z + 1$;
- $w_{kh+1} = k(Z + 1)^2$;
- $p_{kh+1}^{(1)} = (k + 1)Z - B$; and
- $p_{kh+1}^{(2)} = 1$.

In our decision version of the $F_2|| \sum_{j \in E} w_j$ problem, we ask whether there is a feasible schedule (partition) with $\sum_{j \in E} w_j \geq kZ + B + k^2(Z + 1)^2$. This finishes the description of the parameterized reduction (a tabular representation of jobs created by the reduction is given in Table 2).

Below we argue for the correctness of the reduction. We begin by noticing that, since all jobs in the set $\mathcal{J}_i$ ($i = 1, \ldots, k$) share the same due date, namely $iZ$, it follows that, in any feasible schedule, at most one of them can be scheduled in a JIT mode. Therefore, it holds that

$$|E| \leq k + 1.$$  (1)

Let us now prove that, if we have a yes-instance for the $k$SUM, then there exists a feasible schedule for the constructed instance of the $F_2|| \sum_{j \in E} w_j$ problem with $\sum_{j \in E} w_j \geq kZ + B + k^2(Z + 1)^2$. The fact that we
have a yes-instance of kSUM implies that there exists a subset of k elements \( S = \{x_1, x_2, \ldots, x_k\} \subseteq X \) such that \( \sum_{j=1}^{k} x_j = B \), where \( [j] \) is the index of the \( j \)th element in \( S \). We then construct the following feasible schedule for the constructed instance of our scheduling problem. We set \( E = \{J_{1,[1]}, J_{2,[2]}, \ldots, J_{k,[k]}\} \cup \{J_{kh+1}\} \) and \( T = \mathcal{J} \setminus E \). For \( i = 1, \ldots, k \), we then schedule job \( J_{i,[i]} \) during the time interval \( (\sum_{j=1}^{i-1} x_{[j]}, \sum_{j=1}^{i} x_{[j]} ) \) on \( M_1 \), and during the time interval \( (iZ - 1, iZ) \) on \( M_2 \) (since \( \sum_{j=1}^{k} x_{[j]} = B < Z \), it follows that there is no overlap between the processing operations of the same job on both machines). Moreover, we schedule job \( J_{kh+1} \) during the time interval \( (B, (k+1)Z) \) on \( M_1 \) (which is just after the completion of job \( J_{k,[k]} \) on that machine) and during the time interval \( ((k+1)Z, (k+1)Z+1] \) on \( M_2 \). The constructed schedule is illustrated in Fig. 1. The constructed schedule is feasible since (i) there is no overlap between the processing operations of different jobs on the same machine; (ii) there is no overlap between the processing operations of the same job on different machines; and (iii) all jobs in the set \( E \) are scheduled in a JIT mode. Further, for the constructed schedule we have that

\[
\sum_{j \in E} w_j = \sum_{i=1}^{k} (Z + x_{[i]}) + k^2(Z + 1)^2
\]

\[
= kZ + B + k^2(Z + 1)^2;
\]

thus, the constructed instance of our scheduling problem is a yes-instance.

Now we prove that, if we have a feasible partition \( \tau = E \cup T \) with \( \sum_{j \in E} w_j \geq kZ + B + k^2(Z + 1)^2 \) for the constructed instance of the \( F_2||\sum_{j \in E} w_j \) problem, then we have a yes-instance of the kSUM problem. The fact that at most a single job from each \( J_i \) can be scheduled in a JIT mode in any feasible schedule implies that the total weight of the subset of JIT jobs among all jobs in \( \bigcup_{i=1}^{k} J_i \) is at most \( kZ + Z = Z(k + 1) \). Therefore, \( J_{kh+1} \) is included in the set \( E \) (as otherwise, we have that \( \sum_{j \in E} w_j \leq Z(k + 1) < kZ + B + k^2(Z + 1)^2 \), contradicting our assumption that \( \sum_{j \in E} w_j \geq kZ + B + k^2(Z + 1)^2 \). Let \( E' = E \setminus \{J_{kh+1}\} \). Since \( \sum_{j \in E} w_j \geq kZ + B + k^2(Z + 1)^2 \) and \( J_{kh+1} \in E \), we conclude that

\[
\sum_{j \in E'} w_j = \sum_{j \in E} w_j - w_{kh+1} \geq kZ + B.
\]

Let us now prove that \( |E'| = k \). The fact that \( |E'| \leq k \) follows from Eq. (1) above and the fact that \( |E'| = |E| - 1 \). Consider now a JIT set \( E' \) with \( |E'| < k \). Then, we have that

\[
\sum_{j \in E'} w_{ij} = \sum_{j \in E'} (Z + x_j) = |E'|Z
\]

\[
+ \sum_{j \in E'} x_j \leq (k-1)Z + Z \leq kZ,
\]

which contradicts the relation in Eq. (3). Thus, \( |E'| = k \). Based on Eq. (3), we can now conclude that

\[
\sum_{j \in E'} x_j \geq B.
\]
The fact that \( J_{k h + 1} \in E \) implies that this job is scheduled during the time interval \( ((k + 1) Z, (k + 1) Z + 1) \) on \( M_2 \), which in turn implies that it starts not later than in time point \( B \) on \( M_1 \). The fact that \( p_{(1)}^{(k + 1)} = (k + 1) Z - B \geq k Z \) and that any job in \( E' \) has a due date which is not greater than \( k Z \) implies that all jobs in \( E' \) shall be scheduled prior to job \( J_{k h + 1} \) on \( M_1 \). Thus, we have that

\[
\sum_{j_i \in E'} p_i = \sum_{j_i \in E'} x_j \leq B. \tag{6}
\]

Based on Eqs. (5) and (6), we conclude that \( \sum_i \in E' x_i = B \). Now, let \( y_{ij} = 1 \) if \( J \in E' \) and \( y_{ij} = 0 \) otherwise. Moreover, let \( y_{ij} = \sum_{i=1}^k y_{ij} \) for \( j = 1, \ldots, h \). We construct a solution \( S \) for the \( k \)SUM problem by including \( y_{ij} \) copies of \( x_j \) in \( S \). The fact that \( |S| = k \) and \( \sum_{x \in S} x_j = B \) implies that the instance of \( k \)SUM is a yes-instance. \( \square \)

### 3.2 An XP algorithm for a parameterized number of machines combined with the number of different due dates

In this subsection, we complement the result of Theorem 1 by providing an XP algorithm for the \( F_m || \sum J_i \in E w_j \) problem wrt. \#d combined with \( m \). We start by providing an XP algorithm for \( m = 3 \) and then show how to extend the result for a parameterized number of machines.

We say that job \( J \) is of type \( i \) if its due date is \( d_i \) for \( i = 1, \ldots, \#d \). Importantly, the fact that all jobs of the same type share the same due date implies that at most one of them can be scheduled in a JIT mode. Given an instance of the \( F_m || \sum J_i \in E w_j \) problem, we partition the set \( J \) into sets \( J_1, J_2, \ldots, J_{\#d} \), where all jobs in \( J_i \) are of the same type. We construct the sets in such a manner that at the end of the construction the sets are ordered according to the EDD rule such that \( d_1 \leq d_2 \leq \cdots \leq d_\#d \).

Let \( \Omega \) be the set of all subsets of jobs that do not contain two or more jobs of the same type. Without loss of generality, we assume that in any of the elements in \( \Omega \) the jobs are ordered according to the EDD rule (the ordering does not take time given that the sets \( J_1, J_2, \ldots, J_{\#d} \) are ordered according to the EDD rule.) Note that (i) any feasible set of JIT jobs is an element of \( \Omega \); and that (ii) \( |\Omega| = O(n^{\#d}) \).

Next, we explain how we can construct an XP algorithm that can determine whether there exists a feasible schedule where all jobs in some \( \omega \in \Omega \) are scheduled in a JIT mode.

Given a set \( \omega \), let \( d' = (d' \leq \#d) \) be the number of jobs in \( \omega \). We renumber the jobs in \( \omega \) in a non-decreasing order of their due dates such that the first job, \( J_1 \), is the job with the earliest due date among all jobs in \( \omega \); the second job, \( J_2 \), is the job with the second earliest due date among all jobs in \( \omega \); and so on until the last job, \( J_{d'} \) that has the latest due date among all jobs in \( \omega \). Note that this step requires only \( O(d') \) time as the jobs are already ordered according to the EDD rule.

We first check if the condition that \( d_j - p_j^{(3)} \leq d_j \) holds for \( j = 1, \ldots, d' \), where \( d_0 = 0 \) by definition. If not, then there is no feasible schedule with \( E = \omega \). Otherwise, based on Lemma 1, we have that there are \( d' \) possible permutations to schedule jobs in \( \omega \) on machines \( M_1 \) and \( M_2 \). (As according to this lemma, there exists an optimal schedule where the processing permutation on \( M_1 \) is identical to that on \( M_2 \).) For each such possible permutation \( \sigma \), we schedule jobs on \( M_1 \) and \( M_2 \) according to Observation 1, and therefore obtain (in \( O(\#d) \) time) the completion time \( C_j^{(2)} \) of each job \( J \) on \( M_2 \).

Finally, we check whether job \( J_1 \) is indeed ready to be scheduled during the time interval \( (d_j - p_j^{(3)}, d_j) \) on \( M_2 \) for \( j = 1, \ldots, d' \); that is if \( C_j^{(2)} \leq d_j - p_j^{(3)} \), for \( j = 1, \ldots, d' \). If this last condition holds, then there is a feasible schedule with \( E = \omega \). By checking, in a similar fashion, the feasibility of each subset in \( \Omega \), we can find (in XP time) an optimum schedule. To summarize the above discussion, we can use Algorithm 1 below to solve the \( F_3 || \sum J_i \in E w_j \) problem in XP time wrt. \#d.

**Theorem 2** The \( F_3 || \sum J_i \in E w_j \) problem is solvable in \( O(n^{\#d} \#d! \#d!) \) time, and thus belongs to XP wrt. the parameter \#d.

**Proof** The fact that Algorithm 1 solves the \( F_3 || \sum J_i \in E w_j \) problem follows from the discussion that appears above. Step 1 can be done in \( O(n \log n) \) time. Consider now Step 2: the fact that the \( n \) jobs are partitioned into \#d sets \( J_1, J_2, \ldots, J_{\#d} \) implies that at least one of these sets has at most \( n/\#d \) jobs. The fact that set \( J_i \) includes all jobs having the same due date of \( d_i \), for \( i = 1, \ldots, \#d \), implies that set \( \Omega \) includes \( O(n^{\#d} \#d!) \) subsets of \( J \). As each subset of jobs includes \#d jobs, listing out set \( \Omega \) requires \( O(n^{\#d}) \) time. In Step 3, we repeat Steps 3.1–3.4 \( O(\#d) \) times (once for each \( \omega \in \Omega \)). The facts that (i) Steps 3.1 and 3.2 require \( O(\#d) \) time; (ii) in Step 3.3, we construct the set of all \( d'! = O(\#d!) \) permutations of the \( d' \) jobs in \( \omega \); and (iv) in Step 3.4, we perform a set of \( O(\#d) \) operations on each of those \( O(\#d!) \) permutations, implies that the most time-consuming operation within Steps 3.1–3.4 is Step 3.4 that requires \( O(\#d!) \) time. The fact that in Step 3, we repeat Steps 3.1–3.4 for each subset \( \omega \in \Omega \), and there are \( O(n^{\#d}) \) such subsets in \( \Omega \) implies that Step 3 requires \( O(n^{\#d} \#d!) \) time, and the theorem follows. \( \square \)

We note that the result in Theorem 2 can be extended to show that the more general \( F_m || \sum J_i \in E w_j \) problem is solvable in XP time for parameterized number of machines combined with the number of different due dates. The only change required in Algorithm 1 is in Step 3 where we
Lemma 2 There exists an optimal schedule in which job per-
formance of the number of machines combined with the number of different due dates.

We next show that the result in Theorem 2 can be improved by presenting a faster algorithm for the $F_3||\sum_{j \in E} w_j$ problem, running in time $O(n^{\Omega d}d^2\Omega d^m)$ time and thus belongs to XP when parameterized by the number of machines combined with the number of different due dates.

Definition 5 Consider a feasible schedule $S(E)$ for the $F_3||\sum_{j \in E} w_j$ problem on a given feasible set of JIT jobs $E$. We say that a schedule $S(E')$ on job set $E' = E \cup E''$ is a feasible extension of $S(E)$ if (i) in schedule $S(E')$ all jobs in $E$ are scheduled exactly as in $S(E)$; (ii) all jobs in $E$ are scheduled before the jobs in $E''$ on both $M_1$ and $M_2$ (note that according to Lemma 2 in any feasible schedule the jobs are scheduled on $M_3$ according to the EDD rule); and (iii) $S(E')$ is a feasible schedule.

Definition 6 Consider two feasible schedules $S_1(E)$ and $S_2(E)$ for the $F_3||\sum_{j \in E} w_j$ problem on a given feasible set of JIT jobs $E$. We say that $S_1(E)$ is dominated by $S_2(E)$ if for any feasible schedule $S_1(E')$ which is an extension of $S_1(E)$, there exists a feasible schedule $S_2(E')$, which is an extension of $S_2(E)$.

Lemma 3 Let $E$ be a feasible set of JIT jobs for the $F_3||\sum_{j \in E} w_j$ problem, then there exists a feasible schedule, $S(E)$, on job set $E$ that dominates all other feasible schedules on job set $E$ and satisfies the following two properties: (i) it has no idle times on $M_1$; and (ii) it has the minimum completion time on $M_2$ among all feasible schedules on job set $E$.

Proof We divide the proof into two parts. In the first part, we prove that if $E$ is a feasible set of JIT jobs for the $F_3||\sum_{j \in E} w_j$ problem, then there exists a feasible schedule, $S(E)$, on job set $E$ that satisfies properties (i) and (ii) as stated in the lemma. In the second part, we prove that schedule $S(E)$ dominates all other feasible schedules on job set $E$.  

need to enumerate not only all possible permutations on the first two machines, but also the permutations on machines $M_3$, $M_4$, ..., $M_{n-1}$ (recall that according to Lemma 1 and Lemma 2 there exists an optimal schedule in which job permutations on the first two machines are identical; further, in an optimal schedule the job permutation on the last machine follows the EDD rule). The fact that there are $(\#d)^{m-2}$ different combinations of such job permutations on machines, and for each we have to calculate the completion time of each of the $O(\#d)$ jobs on each of the $m$ machines using Observation 1 which requires $O(m\#d)$ time leads to the following corollary:

Corollary 1 The $F_m||\sum_{j \in E} w_j$ problem is solvable in $O(mn^{\Omega d}d^2\Omega d^m)$ time and thus belongs to XP when parameterized by the number of machines combined with the number of different due dates.
The fact that E is a feasible set of JIT jobs implies that there is a non-empty set of feasible schedules, E_{\text{min}}, on job set E, that has the minimum completion time on M_2 among all other feasible schedules on job set E. Now, by contradiction assume that all schedules in E_{\text{min}} include idle times on M_1. We can take any of them, say S’(E) ∈ E_{\text{min}} and construct a schedule S(E) out of it by eliminating the idle times on M_1. This is done simply by shifting any job that has an idle time before its processing on M_1 to the left until the idle time is eliminated, without changing the processing order of the jobs, and keeping the schedule on the other two machines unchanged. The fact that any job in S(E) is completed on M_1 no later than in S’(E) implies that in S(E) it is feasible to schedule the jobs on M_2 and M_3 as in S’(E). Thus, S(E) is a feasible schedule satisfying conditions (i) and (ii).

Consider now a feasible schedule S’(E) on job set E that is not dominated by S(E). The way we defined schedule S(E) implies that S(E) is completed on both M_1 and M_2 no later than the completion time of S’(E) on those machines. Moreover, the completion of S and S’ on M_3 is identical (and is equal to the due date of the job with the latest due date among all jobs in E). Therefore, if S’(E’) on job set E’ = E ∪ E” is a feasible extension of S’(E), then we can construct a feasible schedule S’(E’) on job set E’ = E ∪ E” which is an extension of S(E) by using the same schedule of the jobs in E” as in schedule S’(E’). Thus, S’(E) is dominated by S(E) and we have a contradiction.

Our improved algorithm starts in a similar manner as Algorithm 1. Specifically, it starts by constructing the set Ω in Step 2 and by preforming Steps 3.1 and 3.2. However, for each ω ∈ Ω, instead of trying all d’! possible permutations to schedule jobs in ω in Steps 3.3 and 3.4, it finds all non-dominated schedules for all feasible subsets of ω using Lemma 3 and dynamic programming as follows.

Let ζ be a subset of ω. Moreover, given a job schedule, define the completion time of a set of jobs on a given machine as the completion of the entire set of jobs on the machine. Now, let q^{(i)}(ζ) be the completion time of set ζ on M_i in a non-dominated feasible schedule (we set q^{(i)}(ζ) = ∞ if ζ is not a feasible subset of JIT jobs). It follows from Lemma 3 that q^{(1)}(ζ) = ∑_{J_j ∈ ζ} p_{j}^{(1)}, and that q^{(2)}(ζ) equals the minimum completion time on M_2 among all feasible schedules on job set ζ. In order to find q^{(2)}(ζ), we use the following recursion, which is similar to the recursion of Held and Karp (1962) designed to solve the Traveling Salesman Problem (TSP):

\[
q^{(2)}(ζ) = \min_{J_j ∈ ζ \backslash [J_j]} \left\{ C = \max[q^{(1)}(ζ), q^{(2)}(ζ \backslash [J_j])] + p_j^{(2)} \right\} \quad \text{if } C \leq d_j - p_j^{(3)} \quad \text{otherwise}.
\]

Lemma 4 The value of q^{(2)}(ζ) which is calculated by the recursion in Eq. (7), with the initial condition that q^{(2)}(∅) = 0, represents the minimum possible completion time of set ζ on M_2 for any feasible set of JIT jobs (if ζ is not a feasible set of JIT jobs then the value of q^{(2)}(ζ) is set to ∞).

Proof We prove the lemma using induction on l, the cardinality of ζ. We first prove that the lemma holds for l = 1. Assume, without loss of generality, that ζ = {J_j}. The recursion in Eq. (7) states that

\[
q^{(2)}(ζ) = \begin{cases} C = \max[p_j^{(1)}, 0] + p_j^{(2)} = p_j^{(1)} + p_j^{(2)} & \text{if } C \leq d_j - p_j^{(3)} \quad \text{otherwise}. \end{cases}
\]

Since job J_j is the only job in ζ, and as we cannot start any job on M_1 before its completion on M_{l−1} for i = 2, 3, it follows that the earliest completion time of job J_j on M_2 is at time p_j^{(1)} + p_j^{(2)}. Set ζ = {J_j} is feasible in this case only if we can schedule job J_j in a JIT mode on M_3. Therefore, the condition that p_j^{(1)} + p_j^{(2)} ≤ d_j - p_j^{(3)} has to hold. Otherwise, {J_j} is not a feasible set of JIT jobs.

Consider set ζ of cardinality l and assume (the induction assumption) that the lemma holds for any set ζ’ = ζ \backslash [J_j] of cardinality l − 1 (J_j ∈ ζ); that is, for any set ζ’ of cardinality l − 1 the recursion in Eq. (7) provides the minimum completion time of all jobs in ζ’ on M_2. We next prove that the lemma holds for set ζ as well. For any J_j ∈ ζ, if J_j is the last to be scheduled on M_2, then we cannot start processing J_j on M_2 before (i) the completion of ζ on M_1 at time q^{(1)}(ζ) = ∑_{J_j ∈ ζ} p_j^{(1)}; and (ii) the completion of set ζ’ on M_2. Due to the induction assumption, q^{(2)}(ζ’) = q^{(2)}(ζ \backslash [J_j]) is the earliest time to complete set ζ’ on M_2. Therefore, if J_j is the last job to be scheduled on M_2 among all jobs in ζ, then the earliest completion time of set ζ on M_2 is at time C = max[q^{(1)}(ζ), q^{(2)}(ζ \backslash [J_j])] + p_j^{(2)}. Moreover, such a schedule is feasible only if the condition that C ≤ d_j - p_j^{(3)} holds (as otherwise we cannot complete job J_j in a JIT mode). The lemma now holds from the fact that any job J_j ∈ ζ can be the last to be scheduled on M_2.

To see whether ω is a feasible subset of JIT jobs, we compute the value of q^{(2)}(ζ) using the recursion in Eq. (7) for any subset ζ ⊆ ω in an increasing order of the subset sizes with the initial condition that q^{(2)}(∅) = 0. Lemma 4 guarantees the correctness of this recursion. Then, ω is a feasible set of JIT jobs if and only if q^{(2)}(ω) < ∞.

 Springer
reduced to \#d^2 \#p^d$, by replacing Steps 3.3 and 3.4 in Algorithm 1 with the implementation of the recursion in Eq. (7). Thus, we have the following corollary:

**Corollary 2** The $\sum_{j \in E} w_j$ problem is solvable in $O(n^\#d \#d^2 \#p^d)$ time.

4 Two machines and combined parameters

Theorem 1 shows the (fixed-parameter) intractability of the $\sum_{j \in E} w_j$ problem when we combine \#d with the number of different processing times on the second machine ($\#p^{(2)}$). Below, we analyze the tractability of the $\sum_{j \in E} w_j$ problem when we combine \#d with either the number of different processing times on the first machine ($\#p^{(1)}$) or with the number of different weights ($\#w$).

Since we focus on the two-machine case, according to Lemmas 1 and 2, there exists an optimal schedule with all jobs in the set $E$ being scheduled according to the EDD rule on both machines. Suppose that all $n$ jobs are numbered according to the EDD rule such that $d_1 \leq d_2 \leq \ldots \leq d_n$. In Lemma 5, we present an elimination property that will be used later to design optimization algorithms for the $\sum_{j \in E} w_j$ problem. The following two definitions are necessary to present the lemma.

**Definition 7** Consider an instance for the $\sum_{j \in E} w_j$ problem and let $E = \{J_1, \ldots, J_h\}$ be a feasible subset of JIT jobs for this problem with $J_j$ being the index of the $j$th job in the set $E$ ($l \leq j \leq h$). We say that $E' = \{J_1, \ldots, J_{h-1}\} \cup \{J_{h+1}, J_{h+2}, \ldots, J_{h+k}\}$ is a feasible extension of $E$ with $\sum_{j \in E} w_j \leq \sum_{j \in E'} w_j$ if $E'$ is a feasible set of JIT jobs.

**Definition 8** Consider an instance for the $\sum_{j \in E} w_j$ problem. We say that a feasible subset of JIT jobs $E_1$ is dominated by an alternative feasible subset of JIT jobs $E_2$ if for any feasible JIT set $E_1'$ which is a feasible extension of $E_1$, there exists a feasible JIT set $E_2'$ which is an extension of $E_2$ with $\sum_{j \in E_1} w_j \leq \sum_{j \in E_2} w_j$.

Given the two definitions above, the following lemma was proven by Shabtay (2012) for the $\sum_{j \in E} w_j$ problem:

**Lemma 5** Let $E_1 = \{J_1, \ldots, J_l\}$ and $E_2 = \{J_1, \ldots, J_k\}$ be two feasible subsets of JIT jobs. $E_1$ is dominated by $E_2$ if the following three conditions hold: (i) $\sum_{j \in E_1} w_j \leq \sum_{j \in E_2} w_j$; (ii) $\sum_{j \in E_1} p_{j}^{(1)} \leq \sum_{j \in E_2} p_{j}^{(1)}$; and (iii) $d_{[k]} \geq d_{[k]}$.

Consider first the case of the combined parameters \#d + $\#p^{(1)}$. We say that two jobs $J_i$ and $J_j$ are of the same type if both $d_i = d_j$ and $p_{j}^{(1)} = p_{j}^{(1)}$. Now, let $k$ be the number of different job types ($k \leq \#d \cdot \#p^{(1)}$). Note that (i) at most one job of each type can be scheduled in a JIT mode; and (ii) jobs of the same type are differentiated by their weight and by their processing time on the second machine.

Given an instance of the $\sum_{j \in E} w_j$ problem, we partition the set $E$ into sets, $J_1, J_2, \ldots, J_k$, where all jobs in $J_i$ ($i = 1, \ldots, k$) are of the same type. Let $J_{ij}$ be the $j$th job in $J_i$, and let $d_i$ and $p_{i}^{(1)}$, respectively, be the due date and the processing time on $M_1$ of all jobs in $J_i$. We construct the sets in such a manner that at the end of the construction the sets are ordered according to the EDD rule such that $d_1 \leq d_2 \leq \ldots \leq d_k$.

Given a job schedule, we say that job set $J_i$ is a JIT job set if one of jobs in $J_i$ is scheduled in JIT mode. Accordingly, a subset of job types is considered to be a feasible subset of job types if there exists a feasible schedule which includes a single job of each of these types in the set of JIT jobs (set $E$).

Let $\Omega$ be the set that includes all possible (feasible or not) subsets of job types having different due dates. Note that the set of all feasible subsets of job types is a subset of $\Omega$. Accordingly, to find an optimal set of JIT jobs, it is enough to find, for each $\omega \in \Omega$, the best feasible schedule whose JIT set includes exactly a single job out of each of the job types in $\omega$.

Consider now a specific subset of job sets $\omega \in \Omega$. Below we explain how we can determine, in polynomial time, the best feasible schedule (if such a feasible schedule exists) out of all feasible schedules whose JIT set includes exactly the job types in $\omega$. Given set $\omega$, let $d'$ be the number of job sets in $\omega$ ($d' \leq \#d$). We keep the job sets in $\omega$ ordered according to the EDD rule such that $\omega = \{J_{[1]}, J_{[2]}, \ldots, J_{[d']}, J_{[d+1]}, J_{[d+2]}, \ldots, J_{[d+k]}\}$, where $[i]$ is the index of the $i$th set in $\omega$ and $[i] < [i+1]$ for $i = 1, \ldots, d' - 1$.

Consider now a non-dominated feasible set of JIT jobs $E$ defined on job sets $J_{[1]}, J_{[2]}, \ldots, J_{[d-1]}$ ($d \leq d'$). By definition and Observation 1, we have that the completion time of the last scheduled job in $E$ is (i) $\sum_{i=1}^{l-1} p_{i}^{(1)}$ on $M_1$; and (ii) $d_{[l-1]}$ on $M_2$. Now, $E' := E \cup \{J_{[j]}\}$ ($J_{[j]} \in J_{[j]}$) is a feasible extension of $E$ only if

$$\max \left\{ \sum_{i=1}^{l-1} p_{i}^{(1)} + d_{[l-1]} + p_{[j]}^{(2)} \right\} \leq d_{[j]}.$$

(9)

Let $N(\{I\})$ be the set of all jobs of type $[I]$ that satisfy the condition in Eq. (9), i.e., let $N(\{I\}) = \{J_{[j]} \in J_{[j]} : \max \{\sum_{i=1}^{l-1} p_{i}^{(1)} + d_{[l-1]} + p_{[j]}^{(2)} \leq d_{[j]}\} \}$ If $N(\{I\}) = \emptyset$, then we have no feasible solution with all job types in $\omega$ being JIT. Otherwise, based on Lemma 5, among all feasible extensions of $E$ to include a job from $N(\{I\})$, the one that includes the job with the largest weight among all jobs in $N(\{I\})$ dominates all others. Based on this observation and the above analysis, we can use Algorithm 2 below to solve the $\sum_{j \in E} w_j$ problem with respect to the combined parameter \#d + $\#p^{(1)}$. 

 Springer
Algorithm 2: A fixed-parameter algorithm for the $F_2|| \sum_{j \in E} w_j$ problem wrt. $\#d + \# p^{(1)}$.

**Input:**
- $p^{(1)}_j$, $p^{(2)}_j$, $d_j$ and $w_j$ for $j = 1, 2, \ldots, n$.

**Initialization:**
- Set $\text{Opt} = 0$, $d_0 = 0$, and $E^* = \emptyset$.

**Step 1:**
Partition the set $\mathcal{J}$ into job types $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_k$, in such a manner that (i) all jobs in $\mathcal{J}_i$ have the same due date $d_i$, and the same processing time on $M_1$, $p^{(1)}_i$; and (ii) the job types are ordered according to the EDD rule such that $d_1 \leq d_2 \leq \cdots \leq d_k$.

**Step 2:**
Construct the set $\Omega$, which is a set that includes all possible subsets of job types having different due dates.

**Step 3:**
while $\Omega \neq \emptyset$

1. Select a specific set $\omega \in \Omega$. Set $\mathcal{J} = \mathcal{J} \setminus \{\omega\}$.
2. Let $i = \arg\max_{j \in [1]} (|P(j)|)$. If $N(i) = \emptyset$ return to Step 3.
3. Set $j^* = \arg\min_{i \in [1]} (|P(i)|)$.
4. Set $\mathcal{J} = \mathcal{J} \setminus \{j^*\}$.
5. Set $E^* = E(\omega) + w_{i^*}$.
6. Goto Step 3.

**Output:**
The optimal set of JIT jobs $E^*$. The jobs in $E^*$ are scheduled on $M_1$ and $M_2$ according to the EDD rule. On $M_1$ one after the other with no idle time, and on $M_2$ in a JIT mode.

To calculate the number of subsets in $\Omega$, we note that for each of the $\#d$ possible due dates, there are two possible cases. The first is that no job type having this due date is included in a specific subset $\omega \in \Omega$, and the second is that one of the job types having this due date is included in $\omega$. The fact that there are at most $\# p^{(1)}$ job types with the same due date implies that for each of the $\#d$ due dates there are at most $\# p^{(1)} + 1$ options to select (or not) a job type to be included in $\omega$. Thus, $|\Omega| = O((\# p^{(1)} + 1)^{\#d})$, and accordingly Step 2 requires $O((\# p^{(1)} + 1)^{\#d} + \#d)$ time. In Step 3, for each $\omega \in \Omega$, our algorithm finds the non-dominated feasible schedule (if such exists) out of all feasible schedules where exactly the job sets in $\omega$ are JIT jobs. The fact that $|\Omega| = O((\# p^{(1)} + 1)^{\#d})$ and that the most time-consuming operation within Step 3 is Step 3.2, which requires $O(n)$ time (due to the fact that computing all $N(i)$ sets requires such time, independent of $\#d$) completes our proof.

Consider now the case of the combined parameters $\#d + \# w$ and let $k = \#d + \# w$. Below we show how to modify Algorithm 2 to solve the $F_2|| \sum_{j \in E} w_j$ problem for the combined parameters $\#d$ and $\# w$ in $O(n \log n + n \cdot (\#w + 1)^{\#d})$ time. The first change is in Step 1 where we define $\mathcal{J}_i$ as the set of jobs sharing the same due date $d_i$, and the same weight $w_i$.

**Theorem 3** Algorithm 2 solves the $F_2|| \sum_{j \in E} w_j$ problem in $O(n \log n + n \cdot (\# p^{(1)} + 1)^{\#d})$ time. Thus, the problem is fixed-parameter tractable when parameterized by combining the number $\#d$ of different due dates with the number $\# p^{(1)}$ of different processing times on the first machine.

**Proof** The fact that the algorithm provides the optimal set of JIT jobs follows from our discussion in this section. Step 1 constructs the set of jobs where each set includes all jobs of the same type and the sets are indexed according to the EDD rule. This can be done in $O(n \log n)$ time. In Step 2, we construct a set $\Omega$ which includes all possible subsets of job types having different due dates. As each set $\omega \in \Omega$ includes $O(\#d)$ elements, listing out set $\Omega$ requires $O(|\Omega| \cdot \#d)$ time.
Based on Theorem 3 and the above analysis, we have the following corollary.

**Corollary 3** A slight modification of Algorithm 2 can solve the $F_2|| \sum_{j \in E} w_j$ problem in $O(n \log n + n \cdot (\#w + 1)^n)$ time.

### 5 Three machines and combined parameters

In Sect. 4, we proved that the JIT flow-shop scheduling problem on two machines is FPT with respect to both $\#d + \#p^{(1)}$ and $\#d + \#w$. Now we show that these two cases become W[1]-hard when we consider the JIT flow-shop scheduling problem on three machines.

**Lemma 6** The $F_2|| \sum_{j \in E} w_j$ problem polynomially reduces to the $F_3|| \sum_{j \in E} w_j$ problem with unit processing times on the $M_3$.

**Proof** Given an instance $I = \{p_j^{(1)}, p_j^{(2)}, w_j, d_j : 1 \leq j \leq n\}$ of the $F_2|| \sum_{j \in E} w_j$ problem, we construct the following instance $\tilde{I} = \{\tilde{p}_j^{(1)}, \tilde{p}_j^{(2)}, \tilde{p}_j^{(3)}, \tilde{w}_j, \tilde{d}_j : 1 \leq j \leq n\}$ of the $F_3|| \sum_{j \in E} w_j$ problem. For $j = 1, \ldots, n$, we set (i) $\tilde{p}_j^{(1)} = 1$ (which follows the lemma statement); (ii) $\tilde{p}_j^{(2)} = p_j^{(1)}$; (iii) $\tilde{p}_j^{(3)} = p_j^{(2)}$; (iv) $\tilde{w}_j = w_j$; and (v) $\tilde{d}_j = d_j + 1$.

It follows that there exists a feasible schedule for the constructed instance of the $F_3|| \sum_{j \in E} w_j$ problem with all jobs in the set $E$ being scheduled in JIT mode iff the same set of jobs is a feasible set of jobs for the corresponding instance of the $F_2|| \sum_{j \in E} w_j$ problem.  □

We conclude the following, based on Theorem 1 and Lemma 6. (Note that we use $p_j^{(1)}$, $p_j^{(2)} \geq 1$ for $j = 1, \ldots, n$ in the proof of Lemma 6 as is used in the instance used to prove Theorem 1.)

**Theorem 4** The $F_3|| \sum_{j \in E} w_j$ problem is W[1]-hard when parameterized by $\#d$, even if all jobs have unit processing times on $M_1$ and on $M_3$.

The next theorem shows that the $F_3|| |E|$ problem is W[1]-hard as well wrt. $\#d$.

**Theorem 5** The $F_3|| |E|$ problem is W[1]-hard when parameterized by $\#d$, even if all jobs have unit processing times on $M_3$.

**Proof** We prove the theorem by providing a parameterized reduction from the $k$SUM problem (see Definition 4). Given an instance $\{(x_1, \ldots, x_h), k, B\}$ of $k$SUM, we construct an instance of the decision version of the $F_3|| |E|$ problem as follows. The set $J$ includes $n = kh + 2$ jobs. The set of first $kh$ jobs is the union of $k$ sets $J_1, J_2, \ldots, J_k$, where each set $J_i$ ($i \in \{1, \ldots, k\}$) includes $h$ jobs. Let $J_{ij}$ be the $j$th job in the set $J_i$. Moreover, for any job $J_{ij}$, let $d_{ij}, p_{ij}^{(1)}, p_{ij}^{(2)}$, and $p_{ij}^{(3)}$ be the due date, and the processing times on machines $M_1, M_2, M_3$, respectively. We then set:

- $d_{ij} = kZ + i + 1$ for $i = 1, \ldots, k$ and $j = 1, \ldots, h$, where $Z = k \cdot \max_{x_i \in X} x_i; d_{kh+1} = B + 2; \text{ and } d_{kh+2} = 2kZ + 2$.
- $p_{ij}^{(1)} = x_j$ for $i = 1, \ldots, k$ and $j = 1, \ldots, h; p_{kh+1}^{(1)} = 1; \text{ and } p_{kh+2}^{(1)} = kZ - B$.
- $p_{ij}^{(2)} = Z - x_j$ for $i = 1, \ldots, k$ and $j = 1, \ldots, h; \text{ and } p_{kh+1}^{(2)} = B; \text{ and } p_{kh+2}^{(2)} = kZ$.
- $p_{ij}^{(3)} = 1$ for $i = 1, \ldots, k$ and $j = 1, \ldots, h; \text{ and } p_{kh+1}^{(3)} = p_{kh+2}^{(3)} = 1$.

In our decision version of the $F_3|| |E|$ problem, we ask whether there is a feasible schedule (partition) with $|E| \geq k + 2$. This finishes the description of the parameterized reduction. (A tabular representation of jobs created by the reduction is given in Table 3.) Note that in the reduction indeed all jobs have unit processing times on $M_3$. This matches the theorem statement.

Notice that, since all jobs in $J_i$ ($i = 1, \ldots, k$) share the same due date of $kZ + i + 1$, in any feasible schedule at most one of these jobs can be scheduled in a JIT mode. Therefore, we have that

$$|E| \leq k + 2.$$  \hfill (10)

Let us begin by proving that, if we have a yes-instance of the $k$SUM, then there exists a feasible schedule for the constructed instance of the $F_3|| |E|$ problem with $|E| \geq$
$k + 2$. The fact that we have a yes-instance of the $k$SUM problem implies that there exists a subset of $k$ elements, $S = \{x_1, x_2, \ldots, x_k\} \subseteq X$, such that $\sum_{j=1}^{k} x_j = B$, where $[j]$ is the index of the $j$th element in $S$.

We construct the following schedule of the constructed instance of our scheduling problem. We set $E = \{J_{1,[1]}, J_{2,[2]}, \ldots, J_{k,[k]}\} \cup \{J_{kh+1}, J_{kh+2}\}$ and $T = J \setminus E$. Moreover, we maintain the same processing order on the entire set of machines; specifically, $\sigma_1 = \sigma_2 = \sigma_3 = \{J_{kh+1}, J_{1,[1]}, J_{2,[2]}, \ldots, J_{k,[k]}, J_{kh+2}\}$. Following this processing order, we schedule job $J_{kh+1}$ during the time interval $(0, 1]$ on $M_1$; during the time interval $(1, B + 1]$, and during the time interval $(B + 1, B + 2]$ on $M_3$. Then, for $i = 1, \ldots, k$, we schedule job $J_{i,[i]}$ during the time interval $(1 + \sum_{j=1}^{i-1} x_j, 1 + \sum_{j=1}^{i} x_j]$ on $M_1$; during the time interval $(B + 1 + (i - 1)Z - \sum_{j=1}^{i-1} x_j, B + 1 + iZ - \sum_{j=1}^{i} x_j]$ on $M_2$; and during the time interval $(kZ + i, kZ + i + 1]$ on $M_3$. Finally, we schedule job $J_{kh+2}$ during the time interval $(B + 1, kZ + 1]$ on $M_1$; during the time interval $(kZ + 1, 2kZ + 1]$ on $M_2$; and during the time interval $(2kZ + 1, 2kZ + 2]$ on $M_3$. The constructed schedule is illustrated in Fig. 2.

We next prove that the constructed schedule is feasible (with all jobs in the set $E$ being scheduled in a JIT mode). To this end, we shall prove that (a) there is no overlap between processing operations on each one of the three machines; and (b) there is no overlap between processing operations of each of the jobs on the different machines.

Let us first prove (a). The fact that job $J_{1,[1]}$ starts its processing on $M_1$ right after the completion of job $J_{kh+1}$ at time 1; that job $J_{i+1,[i+1]}$ starts its processing on $M_1$ right after the completion of job $J_{i,[i]}$ at time $1 + \sum_{j=1}^{i} x_j$ for $i = 1, \ldots, k - 1$; and that job $J_{kh+2}$ starts its processing on $M_1$ right after the completion of job $J_{k,[k]}$ at time $1 + \sum_{j=1}^{k} x_j = 1 + B$, implies that there is no overlap of processing operations on $M_1$. The fact that job $J_{1,[1]}$ starts its processing on $M_2$ right after the completion of job $J_{kh+1}$ at time $B + 1$; that job $J_{i+1,[i+1]}$ starts its processing on $M_2$ right after the completion of job $J_{i,[i]}$ at time $B + 1 + iZ - \sum_{j=1}^{i} x_j$ for $i = 1, \ldots, k - 1$; and that job $J_{kh+2}$ starts its processing on $M_2$ right after the completion of job $J_{k,[k]}$ at time $B + 1 + kZ - \sum_{j=1}^{k} x_j = kZ + 1$, implies that there is no overlap of processing operations on $M_2$. Then, the fact that there is no overlap between processing operations on $M_3$ follows from the fact that (i) the start time of job $J_{1,[1]}$ on $M_3$ is at time $kZ + 1$ which is not earlier than the completion time of $J_{kh+1}$ on $M_3$ at time $B + 2$ (as $Z = k \cdot \max_{x_i \in X} x_i > B = \sum_{x_i \in S} x_i$); (ii) job $J_{i+1,[i+1]}$ starts its processing on $M_3$ right after the completion of job $J_{i,[i]}$ at time $Zk + i + 1$ for $i = 1, \ldots, k - 1$; and (iii) job $J_{kh+2}$ starts its processing on $M_3$ at time $2kZ + 1$ which is later than the completion time of $J_{k,[k]}$ on $M_3$ at time $kZ + k + 1$ (as $Z = k \cdot \max_{x_i \in X} x_i \geq 2k > k$).

We move on to prove (b). This claim follows from the fact that (i) the processing of job $J_{kh+1}$ starts at $M_t$ exactly after its completion on $M_{i-1}$ for $i = 2, 3$; (ii) the processing of each job in the set $\{J_{1,[1]}, J_{2,[2]}, \ldots, J_{k,[k]}\}$ starts at $M_t$ only after the entire set is completed on $M_{i-1}$ for $i = 2, 3$; and (iii) the processing of job $J_{kh+2}$ starts at $M_t$ exactly after its completion on $M_{i-1}$ for $i = 2, 3$.

The fact that the schedule is feasible and all jobs in the set $E$ are completed in a JIT mode implies that $|E| = k + 2$. Thus, we have a yes answer for the constructed scheduling instance.

Now we prove that, if we have a feasible partition $r = E \cup T$ for the constructed instance of the $F_3||E|$ problem with $|E| \geq k + 2$, then there exists a solution for the corresponding instance of the $k$SUM problem that yields a yes answer. From Eq. (10), we can conclude that $|E| = k + 2$ in $r$. Let $E' = E \setminus \{J_{kh+1}, J_{kh+2}\}$. The fact that at most a single job for each $J_t$ can be scheduled in a JIT mode in any feasible schedule implies that the number of JIT jobs among all jobs in $\cup_{t=1}^{k} J_t$
is at most $k$, i.e., $|E'| \leq k$. Accordingly, we have that (i) $|E'| = k$; and that (ii) both $J_{kh+1}$ and $J_{kh+2}$ are included in the set $E$. (As otherwise, we would have that $|E| < k+2$, contradicting our assumption that $|E| \geq k+2$.

The fact that $J_{kh+1} \in E$ implies that it has to be scheduled during the time interval $(0, 1]$ on $M_1$; during the time interval $(1, B+1]$ on $M_2$; and during the time interval $(B+1, B+2]$ on $M_3$. Next we prove that all the $k$ jobs in $E'$ shall be scheduled before $J_{kh+2}$ on (i) $M_2$ and (ii) $M_1$. Let us first prove (i). By contradiction, assume that one of jobs in $E'$ (say job $J_{ij}$) is scheduled after $J_{kh+2}$ on $M_2$. The fact that we have to schedule $J_{kh+1}$ during the time interval $(1, B+1]$ on $M_2$ and that $p_{kh+2}^{(2)} = kZ$ implies that job $J_{ij}$ will start its processing on $M_2$ not earlier than on time $kZ + B + 1$ and thus will be completed later than its due date on $M_3$, contradicting its inclusion in $E'$. Let us now prove (ii). By contradiction, assume that one of jobs in $E'$ (say job $J_{ij}$) is scheduled after $J_{kh+2}$ on $M_1$. The fact that we have to schedule $J_{kh+1}$ during the time interval $(0, 1]$ on $M_1$ and that $p_{kh+2}^{(1)} = kZ - B$ implies that job $J_{ij}$ will complete its processing on $M_1$ not earlier than on time $kZ - B + x_j + 1$ and thus will be completed on $M_2$ not earlier than on time $(k + 1)Z - B + 1$. Due to (i) above, this further implies that $J_{kh+2}$ will be completed on $M_2$ not earlier than at time $(2k + 1)Z - B + 1$ and on $M_3$ not earlier than at time $(2k + 1)Z - B + 2 > 2kZ + 2 = d_{kh+2}$, contradicting the fact that $J_{kh+2}$ is an early job.

The fact that $J_{kh+2} \in E$ implies that $J_{kh+2}$ has to start not later than on time $B + 1$ on $M_1$ and not later than on time $kZ + 1$ on $M_2$. Based on (i) and (ii) and the fact that $J_{kh+1}$ has to be scheduled first on each of the three machines, we can conclude that $J_{kh+2}$ cannot start its processing on $M_2$ before (a) the completion of all jobs in $E$ on $M_1$; and (b) before the completion of the jobs in $\{J_{kh+1}\} \cup E'$ on $M_2$. Based on (a), $J_{kh+2}$ cannot start its processing on $M_2$ before $p_{kh+1}^{(1)} + \sum_{i \in E'} p_{ij} ^{(1)} + p_{kh+2}^{(2)} = 1 + \sum_{i \in E'} x_j + kZ - B$. Based on (b), $J_{kh+2}$ cannot start its processing on $M_2$ before $p_{kh+1}^{(1)} + p_{kh+1}^{(2)} + \sum_{i \in E'} p_{ij} ^{(1)} + p_{ij} ^{(2)} = 1 + B + \sum_{i \in E'} (Z - x_j) = 1 + B + kZ - \sum_{i \in E'} x_j$. Therefore, $J_{kh+2}$ cannot start its processing on $M_2$ earlier than $1 + kZ + \max\{\sum_{i \in E'} x_j - B, B - \sum_{i \in E'} x_j\}$. This together with the fact that $J_{kh+2}$ has to start not later than on time $B + 1$ on $M_1$ and not later than on time $kZ + 1$ on $M_2$ implies that $1 + kZ + \max\{\sum_{i \in E'} x_j - B, B - \sum_{i \in E'} x_j\} \leq kZ + 1$, which further implies that $\sum_{i \in E'} x_j = B$.

Now we construct a solution $S$ for the $k$SUM problem as follows. Let $y_{ij} = 1$ if $i \in E'$ and $y_{ij} = 0$ otherwise. Moreover, let $y_j = \sum_{i=1}^{k} y_{ij}$ for $j = 1, \ldots, h$, then by including $y_j$ copies of $x_j$ in $S$, we have a solution for the $k$SUM with $|S| = k$ and $\sum_{x_i \in S} x_i = B$. \hfill \Box

6 Summary and future research

In this paper, we provide a parameterized analysis of the NP-hard JIT flow-shop scheduling problem on two and three machines. The main parameter being studied is the number of different due dates. We prove that the problem is intractable in the parameterized sense, even when the scheduling is done on two machines, and belongs to XP when the scheduling is done on a parameterized number of machines.

Then, we show that when combining the number of different due dates with either the number of different processing times on the first machine or the number of different weights, the problem becomes fixed-parameter tractable when the scheduling is done on two machines. It remains $W[1]$-hard, however, when the scheduling is done on three machines. Our results are summarized in Table I.

One immediate direction for future research is to focus on other parameters, and to study the parameterized complexity of the JIT flow-shop problem with respect to those parameters. Finally, one can move to other machine environments such as unrelated machines, job-shop, and open-shop.

Acknowledgements This research was partially supported by Grant No. 2016049 from the United States-Israel Binational Science Foundation (BSF). The first author's work was supported by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA Grant 631163.11, and by the Israel Science Foundation (Grant 551145/14). We would also like to thank the reviewers for their insightful comments on this paper, comments which improved the paper dramatically.

References

Arkin, E. M., & Silverberg, E. L. (1987). Scheduling jobs with fixed start and finish times. Discrete Applied Mathematics, 18, 1–8.
Baker, K. R., & Scudder, G. D. (1990). Sequencing with earliness and tardiness penalties: A review. Operations Research, 38, 22–36.
Bodlaender, H. L., & Fellows, M. R. (1995). $W[2]$-hardness of precedence constrained $k$-processor scheduling. Operations Research Letters, 18(2), 93–97.
Carlisle, M. C., & Lloyd, E. L. (1995). On the $k$-coloring of intervals. Discrete Applied Mathematics, 59, 225–235.
Čepek, O., & Sung, S. C. (2005). A quadratic time algorithm to maximize the number of just-in-time jobs on identical parallel machines. Computers and Operations Research, 32, 3265–3271.
Choi, B. C., & Yoon, S. J. (2007). Maximizing the weighted number of just-in-time jobs in flow-shop scheduling. Journal of Scheduling, 10, 237–243.
Cyan, M., Fomin, F. V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., & et al. (2015). Parameterized algorithms. New York: Springer.
Downey, R. G., & Fellows, M. R. (1992). Fixed-parameter intractability. In Proceedings of the 7th annual structure in complexity theory conference (COCO ’92) (pp. 36–49).
Downey, R. G., & Fellows, M. R. (2013). Fundamentals of parameterized complexity. New York: Springer.
Elalouf, A., Levern, E., & Tang, H. (2013). An improved FPTAS for maximizing the weighted number of just-in-time jobs in a two-
article discusses the machine flow shop problem. Journal of Scheduling, 16(4), 429–435.

Fellows, M. R., & McCartin, C. (2003). On the parametric complexity of schedules to minimize tardy tasks. Theoretical Computer Science, 298(2), 317–324.

Flum, J., & Grohe, M. (1998). Parameterized complexity theory. New York: Springer.

Fullerton, R. R., & McWatters, C. S. (2001). The production performance benefits from JIT implementation. Journal of Operations Management, 29, 81–96.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics, 3, 287–326.

Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied Mathematics, 10(1), 196–210.

Hermelin, D., Kubitz, J. M., Shabtay, D., Talmon, N., & Woeginger, G. (2015). Scheduling two competing agents when one agent has significantly fewer jobs. In Proceedings of the 10th international symposium on parameterized and exact computation (IPEC ’15).

Kovalyov, M. Y., Ng, C. T., & Cheng, T. C. E. (2007). Fixed interval scheduling: Models, applications, computational complexity and algorithms. European Journal of Operational Research, 178, 331–342.

Lann, A. Y., & Mosheiov, G. (1996). Single machine scheduling to minimize the number of early and tardy jobs. Computers and Operations Research, 23, 769–781.

Mnich, M., & van Bevern, R. (2018). Parameterized complexity of machine scheduling: 15 open problems. Computers and Operations Research, 100, 254–261.

Mnich, M., & Wiese, A. (2015). Scheduling meets fixed-parameter tractability. Mathematical Programming, 154(1), 533–562.

Niedermeier, R. (2006). Invitation to Fixed-Parameter Algorithms. Oxford: Oxford University Press.

Shabtay, D. (2012). The just-in-time scheduling problem in a flow-shop scheduling system. European Journal of Operational Research, 216(3), 521–532.

Shabtay, D., & Bensoussan, Y. (2012). Maximizing the weighted number of just-in-time jobs in several two-machine scheduling systems. Journal of Scheduling, 15(1), 39–47.

Shabtay, D., & Steiner, G. (2012). Scheduling to maximize the number of just-in-time jobs: A survey. In R. Z. Rios-Mercado & Y. A. Rios-Solis (Eds.), Just-In-Time systems. New York: Springer.

van Bevern, R., Mnich, M., Niedermeier, R., & Weller, M. (2015). Interval scheduling and colorful independent sets. Journal of Scheduling, 18(5), 449–469.

van Bevern, R., Niedermeier, R., & Suchý, O. (2016). A parameterized complexity view on non-preemptively scheduling interval-constrained jobs: few machines, small looseness, and small slack. Journal of Scheduling, 20(3), 255–264.

White, R. E., & Prybutok, V. (2001). The relationship between JIT practices and type of production system. Omega, 29, 113–124.

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