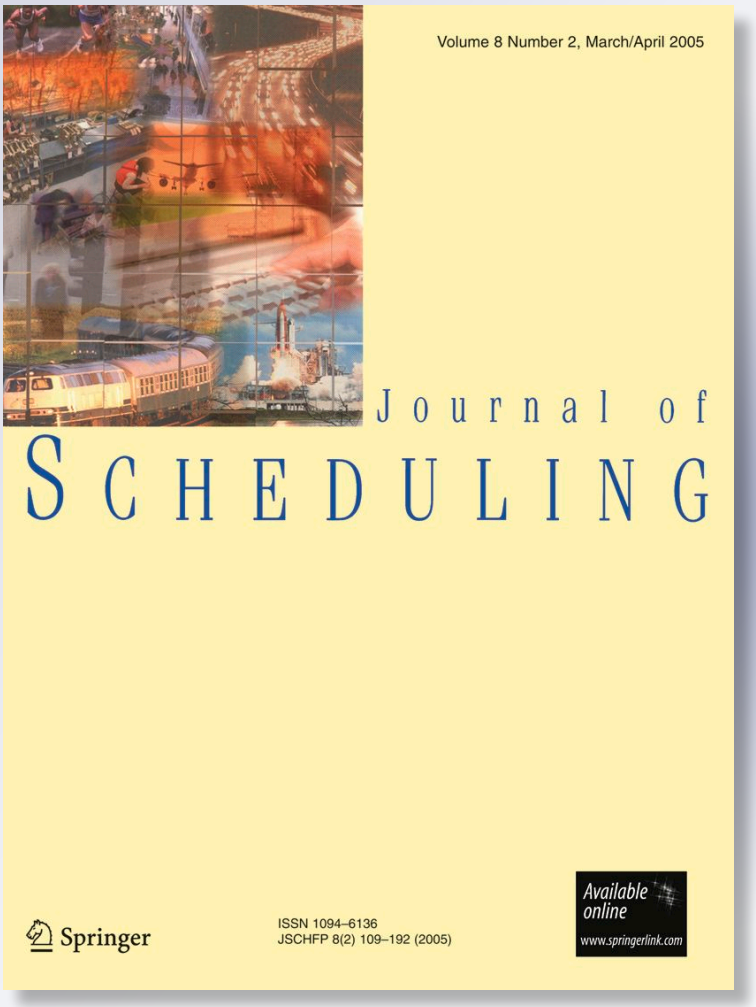
On Eulerian extensions and their application to no-wait flowshop scheduling

Wiebke Höhn, Tobias Jacobs & Nicole Megow

Journal of Scheduling

ISSN 1094-6136
Volume 15
Number 3

J Sched (2012) 15:295-309
DOI 10.1007/s10951-011-0241-1



Springer

J Sched (2012) 15:295–309
DOI 10.1007/s10951-011-0241-1

On Eulerian extensions and their application to no-wait flowshop scheduling

Wiebke Höhn · Tobias Jacobs · Nicole Megow

Published online: 13 July 2011

Abstract We consider a variant of no-wait flowshop scheduling that is motivated by continuous casting in the multi-stage production process in steel manufacturing. The task is to find a feasible schedule with a minimum number of *interruptions*, i.e., continuous idle time intervals on the last production stage. Based on an interpretation as *Eulerian Extension Problems*, we fully settle the complexity status of any particular problem case: We give a very intuitive optimal algorithm for scheduling on two processing stages with one machine in the first stage, and we show that all other problem variants are strongly NP-hard. We also discuss alternative idle time related scheduling models and their justification in the considered steel manufacturing environment. Here, we derive constant factor approximations.

The first author is supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Priority Program "Algorithm Engineering" (1307). The second author is supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD).

W. Höhn (✉)
Technische Universität Berlin, Institut für Mathematik, Straße des 17. Juni 136, 10623 Berlin, Germany
e-mail: hoehn@math.tu-berlin.de

T. Jacobs
National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
e-mail: jacobs@nii.ac.jp

N. Megow
Max-Planck-Institut für Informatik, Campus E1 4, 66123 Saarbrücken, Germany
e-mail: nmegow@mpi-inf.mpg.de

Keywords No-wait flowshop · Complexity · Approximation algorithms · Machine idle times · Continuous casting · Eulerian extensions

1 Introduction

We consider a flowshop scheduling problem that is motivated by a particular application in steel production, the continuous casting process. Scheduling this process can be considered as one of the main challenges in steel production (Harjunkoski and Grossmann 2001). Generally, a steel-making process consists of several production stages that ladles of steel have to pass beginning in a furnace in which the steel is melted and ending with the continuous casting machine where it is solidified to steel slabs. To avoid cooling of the hot melt, no waiting time is allowed between consecutive stages. Depending on the production site, each of these stages may consist of a single or several identical machines. The final stage, the casting machine, plays a special role: the steel must flow continuously into the casting machine. When the continuous flow is broken—this is called a *strand interruption* (or *interruption* for short)—then the casting machine must be stopped for maintenance and extensive cleaning which causes extra cost and a delay in the entire production. Therefore, practitioners call it their objective to find a feasible schedule that minimizes the number of interruptions.

Formulated as a flowshop, we consider a production process where n jobs $J_1, \ldots, J_n$ must pass s production stages $L_1, \ldots, L_s$. Each job J_j consists of s operations each of which is dedicated to a specific stage L_i on which it must process for p_{ij} time units without preemption. Note that we consider operations with zero processing time as infinitely small operations which require a free machine. Each

stage L_i has m_i identical parallel machines available. The jobs pass the production stages $L_1, L_2, \ldots, L_s$ in exactly this order. In a feasible no-wait flowshop schedule, there is no waiting time allowed between the execution of two consecutive operations of the same job.

By the requirements of strand casting, our goal is to process jobs in such a way that the number of interruptions, i.e., the number of continuous idle time intervals on the last production stage L_s, is minimized. We refer to idle time as time where a machine is not processing any job during the actual production process, which means: the time before the first job and after the last job to be processed on the particular machine is not considered as idle time. We denote our objective by $\mathcal{G}$, and we call the problem no-wait flowshop scheduling to minimize the number of interruptions. Following the classical three-field notation (Graham et al. 1979) we denote it by $\mathrm{F}\,|\,\mathrm{nwt}\,|\,\mathcal{G}$ if there is only a single processor available on each stage and by $\mathrm{FF}\,|\,\mathrm{nwt}\,|\,\mathcal{G}$ in the multiprocessor case, which is also called flexible flowshop. In case that the number of stages is fixed to s, we denote the corresponding processor environments by Fs and FFs, respectively. We also compare the objective $\mathcal{G}$ with the standard makespan objective $C_{\max}$, i.e., the completion time of the last job, and consider models to address both.

No-wait flowshop problems (as well as many other sequencing problems) have a natural interpretation as *Eulerian Extension Problems*. This view gives structural insights that lead to very intuitive solution algorithms. A directed multigraph $G = (V, E)$ is called *Eulerian* if it contains a cycle visiting each arc exactly once. A *Eulerian extension* is a set of additional arcs E' for a given (not necessarily connected) multigraph $G = (V, E)$ such that $(V, E \cup E')$ is Eulerian. A *Eulerian Extension Problem* is, generally speaking, the problem of finding a Eulerian extension minimizing the total cost of additional arcs E' according to some cost function.

This problem is very similar to the *Rural Postman Problem* (RPP), see e.g. Orloff (1974), Eiselt et al. (1995), in which we actually determine a Eulerian tour and not only the arcs that guarantee its existence. While there is a one-to-one correspondence between optimal solutions of the two problems, both the runtime complexity and the objective function are slightly different. In the Eulerian Extension Problem only additional arcs contribute to the objective function whereas in the RPP the cost are determined by the entire tour which gives an additional constant. Although the distinction is marginal, we would like to insist on it as a conceptual difference reflected by the name. We show that in the major solvable case of our problem setting the minimum cost Eulerian extension represents implicitly all optimal solutions, whereas a single RPP tour corresponds to just a single optimal solution. This property is not only of theoretical interest, it is also meaningful to practical applications in which often a secondary optimization criterion plays a role. In this case, one may choose accordingly from the set of all optimal solutions regarding the first criterion.

1.1 Related work

Most of the existing literature on no-wait flowshop scheduling addresses the objective of minimizing the makespan. For an extensive survey on various occurrences of no-wait constraints in production environments and previous theoretical work we refer to Hall and Sriskandarajah (1996). The special case of two-stage scheduling $\mathrm{F2}\,|\,\mathrm{nwt}\,|\,C_{\max}$ is well known to be a special case of the so-called Gilmore and Gomory Traveling Salesman Problem (GG-TSP)—one of the first and most famous solvable TSP variants—and can be solved optimally in polynomial time (Gilmore and Gomory 1964; Reddi and Ramamoorthy 1972). The complexity status changes if there is more than one processor on one of the two stages; then the problem becomes strongly NP-hard (Sriskandarajah and Ladet 1986).

The particular problem of scheduling the continuous casting process has been investigated from a practical point of view e.g. in Harjunkoski and Grossmann (2001), Pacciarelli and Pranzo (2004), and Schwindt and Trautmann (2003), where mathematical programming approaches as well as meta-heuristics and simulation are considered. To the best of our knowledge, there is no literature on theoretical investigations on the problem of minimizing the number of interruptions in a no-wait flowshop. The only related theoretical work we are aware of enforces interruption-free scheduling as a hard constraint. In Giaro (2001) and Giaro and Kubale (2004), the authors give complexity and approximation results for openshop and flowshop problems with the objective to minimize the makespan when no interruption is allowed on *any* machine. This restriction is much stronger than what we aim for. In our application, idle times on other stages than the last one, the casting machine, do not incur extra cost. A more restricted variant of the same problem is considered in Wang et al. (2005) with only two production stages and unit processing times on the first stage such that interruptions can occur only on the second stage. Even though this processor environment is close to our setting, we do not see how results could transfer between the makespan minimization problem and our problem of minimizing the number of interruptions.

Several sequencing problems have been solved by interpreting them as Rural Postman Problems with particular cost functions; see e.g. the survey by Eiselt et al. (1995) and the bibliography by Laporte and Osman (1995). For general cost functions RPP is NP-hard (Lenstra and Kan 1976). A very natural and general solution approach has been introduced in Ball and Magazine (1988) in the context of circuit board assembly: in a first step, one balances indegree and outdegree of each vertex, and in a second step,

the strong connectivity is established. The particular algorithm for their cost function applies also to the GG-TSP and more general TSP variants (Gutin and Punnen 2002; Kabadi 2002). It is very intuitive and admits a simple correctness proof, but with the running time $\mathcal{O}(n^2)$ it is slightly slower than the algorithms by Gilmore and Gomory (1964) and Vairaktarakis (2003) which have a worst case computation time $\mathcal{O}(n \log n)$. This is best possible (Rote and Woeginger 1998).

1.2 Our contribution

We fully resolve the complexity status of no-wait flowshop scheduling to minimize the number of interruptions for any machine configuration. The crucial ingredient for both, deriving polynomial time algorithms and showing NP-hardness, is an interpretation of the scheduling problem as a Eulerian Extension Problem with an appropriate cost function.

For two-stage scheduling with single processors, we derive an elegant and fast optimal algorithm. In this case, we obtain an implicit representation of all optimal solutions in time $\mathcal{O}(n \log n)$, from which any particular optimum can be extracted in time $\mathcal{O}(n^2)$. Moreover, we solve optimally the generalized problem FF2 | nwt | $\mathcal{G}$ with a single machine on the first stage, $m_1 = 1$, and m_2 arbitrary. Notice that this result is a sharp contrast to the makespan variant of the same problem which is known to be strongly NP-hard (Sriskandarajah and Ladet 1986).

All other problem variants FFs | nwt | $\mathcal{G}$ for any value of s and any machine configuration $m_1, \ldots, m_s$ are NP-hard. The basis for this result is the proof of NP-hardness for the problem on three stages F3 | nwt | $\mathcal{G}$. Here we establish the connection to what we call a two-dimensional Eulerian Extension Problem with a more general cost function, which could be interpreted as an Extension Problem in a two-dimensional space. To derive these results, we actually give reductions to a decision variant of the problem, i.e., we ask if an interruption-free solution exists. Obviously, NP-completeness for the decision problem and a potential optimal value of zero in the optimization variant imply that no algorithm can yield an approximation guarantee, unless P = NP.

This inapproximability result for most of the machine arrangements is extremely discouraging. We show that a small change in the optimization view point can change this situation. We simply rephrase our objective of minimizing the number of interruptions as maximizing the number of non-interruptions. Clearly, the optimal solution does not change, but the approximability issue does and we give constant approximation algorithms for this objective.

The objective of minimizing the number of strand interruption as requested by practitioners might be a reasonable objective but it completely ignores the makespan of the final schedule. We give examples where the makespans of $\mathcal{G}$-optimal schedules differ by a factor in the order of the number of stages. This raises the quest for a more general objective function that combines both, makespan and number of interruptions. In certain productions, the cost for an interruption can be translated directly into a *minimum delay time* λ; this is the minimum time period for which the last stage processor must be kept idle in case of an interruption. For such scheduling scenarios, we consider the objective of minimizing the makespan $C_{\max}$ subject to minimum delay times. For two-stage single processor scheduling we derive a factor 2 approximation which can be improved for certain choices of minimum delay time λ.

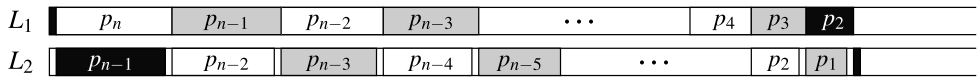
1.3 Organization of the paper

In Sect. 2 we contrast the new objective $\mathcal{G}$ with the classical one $C_{\max}$. We present optimal algorithms for the two-stage problem with a single machine on the first stage in Sect. 3. In Sect. 4 follow the inapproximability results for all other problem variants differentiated by the number of machines on each stage. Finally in Sect. 5 we discuss two alternative models and their justification, and we derive constant factor approximations.

2 Makespan versus interruptions

The no-wait condition in single processor flowshop scheduling generally implies the same job order on each stage. Thus, idle times are uniquely determined by the job order which leads to a natural interpretation as an asymmetric Traveling Salesman Problem (TSP).

This fact has been observed first for the makespan variant by Piehler (1960): Given an instance of Fs | nwt | $C_{\max}$, we add an auxiliary job J_0, having zero processing times on all stages, and consider a complete directed graph on this extended set of jobs. The distance between two jobs is defined as the sum of the processing time of the first job on the last stage and the idle time between the jobs on this stage. Deleting the job J_0 from a tour of this TSP instance, yields a schedule whose makespan equals the cost of the tour. In fact, F2 | nwt | $C_{\max}$ is well known to be a special case of GG-TSP (Gilmore and Gomory 1964; Reddi and Ramamoorthy 1972), one of the most famous solvable subclasses of TSP, where each city i is associated with two numbers A_i and B_i. The cost for traveling from city i to city j is $\int_{B_i}^{A_j} f(x)\,dx$ if $A_j \geq B_i$ and $\int_{A_j}^{B_i} g(x)\,dx$ otherwise, where f, g are integrable functions satisfying $f(x) + g(x) \geq 0$, for any x. Each job J_j of an instance of F2 | nwt | $C_{\max}$ can be interpreted as a city



(a) Schedule with minimum makespan and $n-1$ interruptions.

(b) Schedule without interruptions and maximum makespan.

Fig. 1 Example where an optimal schedule with respect to $C_{\max}$ ($\mathscr{G}$) performs very badly with respect to $\mathscr{G}$ ($C_{\max}$). Instance with n jobs of F2 | nwt | $C_{\max}$ with processing times $p_{11} = p_{2n} = \varepsilon$, $p_{1j} = p_{n-j+2}$ for $j = 2, \ldots, n$, and $p_{2j} = p_{n-j}$ for $j = 1, \ldots, n-1$, where $p_k = p + k\varepsilon$ for some $p > 0$ and sufficiently small $\varepsilon > 0$

with $A_j = p_{1j}$ and $B_j = p_{2j}$, and the cost function is of Gilmore–Gomory type with $f \equiv 1$ and $g \equiv 0$.

Similarly, the $\mathscr{G}$-related flowshop scheduling problem can be formulated as a TSP problem, even though the cost function is not of Gilmore–Gomory type. We define the processing time of J_0 on the last stage to be $\max_{j=1,\ldots,n} \sum_{i=1}^{s-1} p_{ij}$ and 0 on all other stages. Due to the no-wait constraint, the processing times of any two distinct jobs J_i, J_j determine whether or not an interruption occurs when J_i is scheduled directly before J_j. Hence, we can set the distance of the directed arc between the corresponding nodes to 1 if an interruption occurs and 0 otherwise. However, the TSP with distances 0 and 1 is highly intractable; it is easy to see that it is hard to approximate. Therefore, this approach motivated by the makespan variant of the problem does not lead to a straightforward solution for minimizing $\mathscr{G}$.

The different structure of the problems is also visible in the example in Fig. 1. An optimal schedule with respect to one objective may perform very badly for the other, that is, a schedule with minimum makespan may have $n-1$ interruptions, whereas a schedule for the same instance without interruptions may have maximum makespan.

Nevertheless, the problem Fs | nwt | $\mathscr{G}$ can be reduced to Fs | nwt | $C_{\max}$ in the following way. For each $k = 0, \ldots, n-1$ we can verify for an instance I of Fs | nwt | $\mathscr{G}$ if a solution with at most k interruptions exists, using an optimal algorithm for the corresponding makespan minimization problem on a slightly modified instance. Simply add $k+1$ additional jobs to I with processing time 0 on the first $s-1$ stages and $B = \max\{\sum_{i=1}^{s-1} p_{ij} \mid j \in J\}$ on the final stage. A schedule of minimum makespan $(k+1)B + \sum_{j \in J} p_{sj}$ exists if and only if there is a solution for I with at most k interruptions. Exploring $k = 0, \ldots, n-1$ via binary search, we obtain an $\mathcal{O}(C(I) \log n)$ time algorithm for computing an optimal schedule where $C(I)$ is the computation time of the optimal algorithm for the makespan problem. In the single processor two-stage environment, the algorithms for the GG-TSP (Gilmore and Gomory 1964; Vairaktarakis 2003) lead to an optimal algorithm with time complexity $\mathcal{O}(n \log^2 n)$ for minimizing the number of interruptions.

Notice that this relation between the makespan and the interruption minimization problem does not generally imply the same complexity status for both problems.

3 Solving two-stage no-wait flowshop problems

In this section, we present an algorithm for solving the problem F2 | nwt | $\mathscr{G}$ directly in a very natural and intuitive way. Whereas the approach described in the section above yields exactly one optimal solution, the alternative approach leads to a representation of all optimal solutions, which is desirable in particular for bicriteria optimization.

In this approach, we interpret the flowshop problem as a Eulerian Extension Problem where the vertices are labeled with real numbers. More specific, the vertex labels refer to processing times, and each job is represented by an arc between the vertex pair corresponding to its processing times on the first and second machine. In cases of the same processing time occurring more than once, we only add one vertex with the respective label. Additional arcs, we call them *extension arcs*, can be inserted in order to make the graph Eulerian. We will see that the insertion of (u, v) with $u < v$ accounts for an interruption on the second machine. We call such extension arcs *up arcs*, whereas extension arcs (u, v) with $u > v$ are called *down arcs*. The number of up arcs in an optimal Eulerian Extension will correspond to the minimum number of interruptions between jobs, and a corresponding optimal schedule can be read off from a Eulerian tour in this extended graph. Now, we define the $\mathscr{G}$-related One-Dimensional Eulerian Extension Problem ($\mathscr{G}$-1DEE) as follows.

Definition 1 ($\mathscr{G}$-1DEE) Given a finite directed multigraph $G = (V, E)$ where the vertices in V are labeled with distinct real numbers, the problem $\mathscr{G}$-1DEE is to find a Eulerian extension for G minimizing the number of up arcs.

Let indeg(v) and outdeg(v), $v \in V$, denote the indegree and outdegree of v, respectively. The following sufficient condition for a Eulerian graph is well known; see West (2001).

Lemma 1 *A graph $G = (V, E)$ is Eulerian if and only if it is connected and* indeg$(v) =$ outdeg(v), *for all $v \in V$.*

We call a down arc (u, v) *minimal* if u and v are direct neighbors in a linear ordering of V by non-decreasing labels. Furthermore, we denote an up arc (u, v) as *maximal* if u has the minimum label and v has the maximum label in V. Consequently, there is only one distinct maximal arc.

Lemma 2 *Given a Eulerian extension E' for an instance of $\mathcal{G}$-1DEE with $G = (V, E)$, there is a Eulerian extension E'' satisfying the following properties: E'' contains only minimal down arcs and maximal up arcs, E' and E'' have the same cost, and for each Eulerian cycle in $(V, E \cup E')$ there is one in $(V, E \cup E'')$ where the arcs from E appear in the same order.*

Proof Given an E', we replace every non-maximal up arc (u, v) by the arcs $(u, u'), (u', v'), (v', v)$. Here (u', v') is the maximal up arc, and (u, u') and (v', v) are down arcs. If $u = u'$ or $v = v'$, then the arcs (u, u') and (v', v) are not required, respectively. Obviously, this procedure preserves the total number of up arcs, and keeps the balance of the vertex degrees unchanged.

Now that all up arcs are maximal, we replace non-minimal down arcs by minimal ones: Let (u, v) be a non-minimal arc in the extension E', i.e., there is a vertex w with $u < w < v$ or $u > w > v$. We replace (u, v) by (u, w) and (w, v) to obtain E''. This preserves the balance of indegree and outdegree for each vertex, and still uses only down arcs. Repeatedly performing this kind of operation for each non-minimal arc we obtain a Eulerian extension that contains only minimal arcs without increasing the cost.

Analogously, we can replace each arc of a Eulerian cycle in $E \cup E'$ by the corresponding path of minimal down arcs and maximal up arcs in $E \cup E''$. This preserves the order of arcs from E in the cycle. □

Theorem 1 F2 | nwt | $\mathcal{G}$ *can be reduced to $\mathcal{G}$-1DEE in polynomial time.*

Proof We construct an instance $I' = (V, E)$ of $\mathcal{G}$-1DEE from an F2 | nwt | $\mathcal{G}$ instance I by defining $V = \bigcup_{i=1}^{n} \{p_{1i}, p_{2i}\}$, keeping only one vertex for each distinct processing time. We add a *job arc* (p_{1i}, p_{2i}) to E for each job J_i, $i = 1, \ldots, n$. Furthermore, we add one maximal up arc to E, serving as a dummy job between the first and last job in the tour. In the following we show that an optimal extension E' to I' contains k up arcs if and only if an optimal solution to I causes k interruptions.

Consider a schedule for I. For simplicity, we assume that jobs are scheduled in increasing order of their indices. We add an extension arc (p_{2i}, p_{1i+1}) to E' for $i = 1, \ldots, n-1$. This way we obtain a number of up arcs equal to the number of interruptions caused by the schedule. Let (u, v) be the maximal dummy up arc of the instance I'. Adding the down arcs $(p_{2n}, u), (v, p_{11})$ to E', we obtain an extension admitting the Eulerian cycle $p_{11}, p_{12}, p_{21}, p_{22}, \ldots, p_{2n}, u, v, p_{11}$.

Consider an optimal solution E' to I'. We employ Lemma 2 assuming that all down arcs are minimal and all up arcs are maximal. We convert the set E' into a solution to I by scheduling the jobs in the order in which the corresponding job arcs appear in some Eulerian cycle in $(V, E \cup E')$, breaking the cycle at the dummy arc in E.

Whenever an interruption appears between two consecutive jobs, there is at least one up arc from E' between the corresponding job arcs in the cycle, i.e., the number of interruptions is at most k. We also see that between the traversal of any two maximal up arcs (including the dummy arc from E) there is at least one job arc, because otherwise the Eulerian tour would traverse a cycle of only arcs from E' and that cycle could be removed from the extension. For any up arc from E', consider the job arcs (p_{1i}, p_{2i}) and (p_{1j}, p_{2j}) traversed before and after it in the Eulerian tour. It must hold that $p_{2i} < p_{1j}$, because otherwise the assumption of only minimal down arcs in E' would again imply the existence of a pure extension arc cycle in the tour. This argumentation shows that there are at least k interruptions. □

Let indeg(V') denote the indegree of a subset $V' \subseteq V$, i.e., the number of arcs $(u, v) \in E$ with $u \notin V'$ and $v \in V'$. Let the outdegree outdeg(V') be defined analogously. We state a necessary condition based on indegree and outdegree of vertex subsets.

Lemma 3 *If a graph $G = (V, E)$ is Eulerian, then* indeg(V') = outdeg(V') *for any subset of vertices $V' \subseteq V$.*

Proof We prove the statement by showing a more general result.

$$
\begin{aligned}
&\text{indeg}(V') - \text{outdeg}(V') \\
&= \left(\sum_{v \in V'} \text{indeg}(v) - |\{(u, v) \mid u, v \in V'\}| \right) \\
&\quad - \left(\sum_{v \in V'} \text{outdeg}(v) - |\{(v, u) \mid u, v \in V'\}| \right) \\
&= \sum_{v \in V'} \text{indeg}(v) - \sum_{v \in V'} \text{outdeg}(v)
\end{aligned}
$$

□

Algorithm 1 Algorithm for 1DEE

1: Sort all vertices in V in non-decreasing order of their labels.
2: Let v_i be the ith vertex in the ordering and let $V_i := \{u \in V \mid u \leq v_i\}$. Compute $b_{\max} := \max_{i=1,\ldots,|V|-1}\{0, b(v_i)\}$, where $b(v_i) := \text{indeg}(V_i) - \text{outdeg}(V_i)$.
3: Initialize E' as the set containing $b_{\max}$ copies of the maximal up arc $(v_1, v_{|V|})$.
4: For $i := 1$ to $|V| - 1$, add $b_{\max} - b(v_i)$ minimal down arcs (v_{i+1}, v_i) to E'.
5: If $(V, E \cup E')$ is not strongly connected, add one further maximal up arc $(v_1, v_{|V|})$ and minimal down arcs $\{(v_{i+1}, v_i) \mid 1 \leq i \leq |V| - 1\}$ to E'.

The following simple and intuitive Algorithm 1 for solving $\mathcal{G}$-1DEE follows generally the balance-and-connect approach of Ball and Magazine (1988). First, it determines the minimum number of up arcs that are necessary and sufficient for achieving balanced indegree and outdegree at each vertex. They are inserted together with the suitable set of down arcs. Then, the algorithm checks if the resulting graph is strongly connected. If not, one additional up arc and the respective sequence of down arcs suffice to achieve the desired connectivity.

Lemma 4 *Algorithm 1 chooses in Step 3 and 4 only arcs that are necessary for any feasible Eulerian extension that consist only of minimal down arcs and maximal up arcs. The two steps effectuate that* $\text{indeg}(v) = \text{outdeg}(v)$ *for each* $v \in V$ *in* $(V, E \cup E')$.

Proof Let v_i be the vertex maximizing $b(v_i)$ in Step 2 of the algorithm. Lemma 3 states that there must be $\max\{0, b(v_i)\} = b_{\max}$ arcs from V_i to $V \setminus V_i$ in any feasible Eulerian extension. As these arcs are up arcs, they have to be maximal due to our restriction. They are inserted by the algorithm in Step 3.

After Step 3, we have $\text{indeg}(V_i) - \text{outdeg}(V_i) = b(v_i) - b_{\max} \leq 0$ in the graph $(V, E \cup E')$ for each $i = 1, \ldots, |V| - 1$. So from Lemma 3 follows that for the graph to be Eulerian there must be $b(v_i) - b_{\max}$ additional arcs from $V \setminus V_i$ to V_i. Such arcs are down arcs, and due to our restriction to minimal ones, they must be copies of the arc (v_{i+1}, v_i). Inserting them is exactly what Algorithm 1 does in Step 4.

As soon as Step 4 has been executed, we have $\text{indeg}(V_i) = \text{outdeg}(V_i)$ for $1 \leq i \leq |V|$, and in particular $\text{indeg}(v_1) = \text{outdeg}(v_1)$. Since $\text{indeg}(V_i) - \text{outdeg}(V_i) = \sum_{j=1}^{i} \text{indeg}(v_j) - \text{outdeg}(v_j)$ (see proof of Lemma 3), it follows inductively that $\text{indeg}(v_i) = \text{outdeg}(v_i)$ for all i. □

Theorem 2 *Algorithm 1 solves $\mathcal{G}$-1DEE optimally in time* $\mathcal{O}(|V| \log |V| + |E|)$.

Proof By Lemma 2 it suffices to consider only maximal up arcs and minimal down arcs as extension arcs. Let E_1 be the set of arcs chosen by the algorithm by the end of Step 4. By Lemma 4, E_1 is a subset of any feasible Eulerian Extension for G, and we know that the indegree and outdegree of each node is balanced in $(V, E \cup E_1)$. If this graph is strongly connected, it is Eulerian (Lemma 1), and thus, Algorithm 1 is optimal.

Otherwise, at least one additional extension arc, i.e., either a maximal up arc or a minimal down arc, must be added. Suppose, we add a minimal down arc, say (v_{i+1}, v_i), then we have $\text{indeg}(V_i) - \text{outdeg}(V_i) = 1$ in the resulting graph. From Lemma 3 follows that we need an additional up arc for re-establishing balanced indegree and outdegree for each node. Thus, to establish connectivity in $(V, E \cup E_1)$ it is necessary to add an up arc. It is easy to see that with one additional up arc the set of minimal down arcs inserted in Step 5 of the algorithm is necessary and sufficient to re-establish the balancedness of each node's indegree and outdegree. The resulting graph $(V, E \cup E')$ contains the cycle $v_1, v_{|V|}, v_{|V|-1}, \ldots, v_1$ and is therefore strongly connected.

The initial sorting step of the algorithm requires time $\mathcal{O}(|V| \log |V|)$. Steps 2, 3, and 4 take $\mathcal{O}(|E|)$ time. The number of *distinct* up arcs added to E' is linear in $|V|$, and therefore in Step 5 it can be tested in time $\mathcal{O}(|V| + |E|)$ whether the graph constructed so far is strongly connected or not. □

We remark that the optimal solution to $\mathcal{G}$-1DEE is always unique, because the objective function value corresponds to the number of up arcs, and—assuming that all up arcs are maximal and all down arcs minimal—the set of down arcs is uniquely determined by this number. As a consequence of the reduction given in the proof of Theorem 1 and Lemma 2, the extension E' computed by Algorithm 1 implicitly represents all optimal schedules for the flowshop problem. According to the runtime analysis in the proof that representation can be computed in time $\mathcal{O}(n \log n)$ because both $|V|$ and $|E|$ are linear in the number of jobs n. However, there may be $\mathcal{O}(n^2)$ many extension arcs E', so computing an Eulerian tour in the graph obtained by Algorithm 1 may require a running time quadratic in the input size of the scheduling instance.

Corollary 1 *Any problem instance of* $\text{F2} \mid \text{nwt} \mid \mathcal{G}$ *can be solved optimally in time* $\mathcal{O}(n^2)$.

The above algorithm can be applied also to the two-stage flowshop problem to minimize the number of interruptions with more than one machine on the second stage.

Theorem 3 *Any problem instance I of* $\text{FF2} \mid \text{nwt} \mid \mathcal{G}$ *with a single processor on the first stage,* $m_1 = 1$, *can be solved optimally in time* $\mathcal{O}(n^2)$.

Proof Given an instance I of problem FF2 | nwt | $\mathcal{G}$ with $m_1 = 1$, we consider an instance I' which equals I restricted to single machines on both stages, i.e., $m'_1 = m'_2 = 1$. We find an optimal solution S' for I' with r' interruptions. This can be done efficiently by Theorem 2 and gives a feasible solution for I. Now, we construct an improved feasible schedule S for instance I with $r = \max\{0, r' - m_2 + 1\}$ interruptions: if there is an interruption in S' then we move the next block of interruption-free processing jobs to an unused machine of the second stage; we repeat until all interruptions are resolved or until all m_2 machines are used in S. This reduces the number of interruptions by $m_2 - 1$ or less if $r' < m_2 - 1$.

The solution S is optimal for I. To see that, assume for the sake of contradiction there is an optimal solution S^* with less interruptions $r^* < r$. Then the corresponding schedule can be transformed into a feasible one for instance I' with $r'' < r'$ interruptions. Run the set of jobs using machine m_i in S^* consecutively for $i = 1, \ldots, r^* - 1$ using only one processor at the second stage. This gives a feasible solution S'' for I' with at most $m_2 - 1$ interruptions more, i.e., $r'' \leq r^* + (m_2 - 1) < r + m_2 - 1$. This contradicts the optimality of schedule S' for I' with $r' \geq r + m_2 - 1$ interruptions. □

4 Complexity of minimizing the number of interruptions

In contrast to the polynomial time solvable problems with two stages in Sect. 3, the problem becomes strongly NP-hard in any other case.

Theorem 4 *The problem* FFs | nwt | $\mathcal{G}$ *is strongly NP-hard for any constant number of stages* $s \geq 3$ *and arbitrary constant numbers of machines. The same is true for* FF2 | nwt | $\mathcal{G}$ *with* $m_1 > 1$.

The proof follows by combining NP-hardness results for four particular problem classes (machine configurations). The problem F3 | nwt | $\mathcal{G}$, which plays a key role, is considered in Sect. 4.1; to show hardness, we utilize a two-dimensional generalization of the $\mathcal{G}$-1DEE. The remaining problem classes are considered in Sect. 4.2.

In the proofs, we actually give reductions to a decision variant of the problem under consideration, i.e., we ask if an interruption-free solution exists. We denote the decision problem by $\mathcal{E}_0$(F3 | nwt | $\mathcal{G}$). Obviously, the NP-completeness of the decision problem implies NP-hardness of the optimization variant. Moreover, it rules out the existence of an approximation algorithm, unless P = NP. We remark here that our proofs can easily be extended to show that the optimization problem remains strongly NP-hard under the assumption that every solution has at least one interruption.

4.1 Hardness for scheduling on three stages

We show that the problem F3 | nwt | $\mathcal{G}$ is strongly NP-hard. To show this result, we consider a natural two-dimensional interpretation of $\mathcal{G}$-1DEE in which each vertex has two labels which can be seen as points in $\mathbb{R}^2$. We define the $\mathcal{G}$-related two-dimensional Eulerian Extension Problem ($\mathcal{G}$-2DEE) as follows.

Definition 2 ($\mathcal{G}$-2DEE) Given a directed multigraph $G = (V, E)$ with vertices $V \subset \mathbb{R}_0^+ \times \mathbb{R}_0^+$, determine whether there exists a Eulerian extension E' for G using only *down arcs* $\{(u, v) \mid u, v \in V,\ u \geq v$ component-wise$\}$.

We show that the problem $\mathcal{G}$-2DEE is strongly NP-complete by reduction from the Three-Dimensional Matching Problem (3DM).

Definition 3 (Three-Dimensional Matching, 3DM) Given a set $U \subseteq M_1 \times M_2 \times M_3$ of triples, where M_1, M_2 and M_3 are pairwise disjoint and have the same number k of elements, decide whether U contains a subset $U' \subseteq U$ with $|U'| = k$ and no two elements of U' agree in any coordinate.

Here we assume w.l.o.g. that any element of $M_1 \cup M_2 \cup M_3$ appears in at least one triple of U. The problem 3DM is well known to be strongly NP-complete (Karp 1972). Our reduction to $\mathcal{G}$-2DEE borrows ideas from the reduction of 3DM to a weighted tour problem given by Röck (1984) in order to show the NP-hardness of the flowshop problem F3 | nwt | $C_{\max}$.

Theorem 5 *The problem* $\mathcal{G}$*-2DEE is strongly* NP*-complete.*

Proof Denote the arcs in a solution E' to $\mathcal{G}$-2DEE as *extension arcs*. Note that in contrast to the one-dimensional case, extension arcs in this setting only contain down arcs. We say that two points $u = (x_u, y_u), v = (x_v, y_v) \in \mathbb{R}_0^+ \times \mathbb{R}_0^+$ are *independent*, if the relation of their coordinates is such that neither (u, v) nor (v, u) could be an extension arc, i.e., if either $x_u < x_v$ and $y_u > y_v$, or $x_u > x_v$ and $y_u < y_v$.

Consider two rectangles $A = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$, $A' = [x'_{\min}, x'_{\max}] \times [y'_{\min}, y'_{\max}]$ in $\mathbb{R}_0^+ \times \mathbb{R}_0^+$. We say that A and A' are *independent* if any two points $u \in A$, $v \in A'$ are independent. Formally, A and A' are independent if and only if either $x_{\max} < x'_{\min}$ and $y_{\min} > y'_{\max}$, or $x_{\min} > x'_{\max}$ and $y_{\max} < y'_{\min}$.

A point $v = (v_x, v_y)$ is *reachable* from another point $u = (u_x, u_y)$ if (u, v) is a down arc, and v is *one-way reachable* from u if it is reachable from u, but u is not reachable from v. Formally, v is reachable from u if $u_x \geq v_x$ and $u_y \geq v_y$. One-way reachability is obtained by additionally demanding that $u \neq v$.

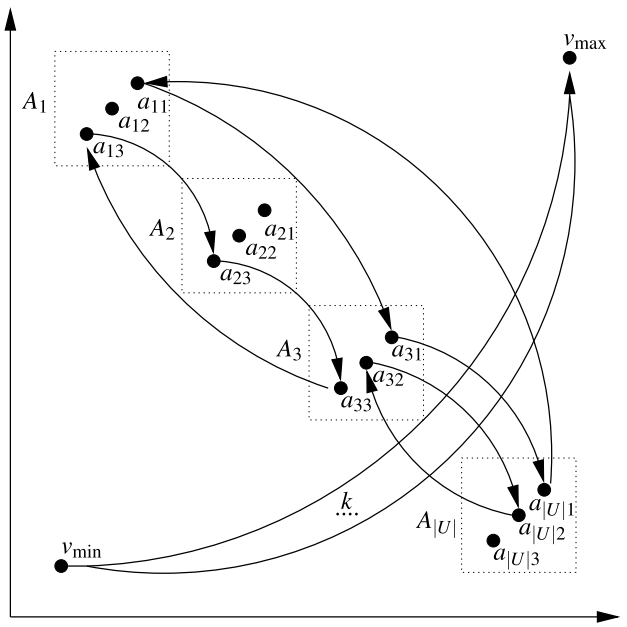


Fig. 2 2DEE representation of a 3DM instance

Given an instance $U \subseteq M_1 \times M_2 \times M_3$ of 3DM with $|M_1| = |M_2| = |M_3| = k$, we construct an equivalent $\mathscr{G}$-2DEE instance (V, E) as follows: Let $\{A_1, \ldots, A_{|U|}\}$ be a collection of pairwise independent rectangles. For $i = 1, \ldots, |U|$, define three points $a_{i1}, a_{i2}, a_{i3} \in A_i$ such that a_{i2} is one-way reachable from a_{i1}, and a_{i3} is one-way reachable from a_{i2}. We define the set of vertices as $V = \bigcup_{i=1,\ldots,|U|}\{a_{i1}, a_{i2}, a_{i3}\} \cup \{v_{\min}, v_{\max}\}$, where $v_{\min}$ is such that it is one-way reachable from any other vertex in V, and $v_{\max}$ is such that any other vertex in V is one-way reachable from it. Formally, the vertex set can be implemented as $v_{\min} = (0, 0)$, $v_{\max} = (|U| + 1, |U| + 1)$, and $a_{ij} = (4i - j, 4(|U| + 1 - i) - j)$ for $1 \leq i \leq |U|$ and $j \in \{1, 2, 3\}$.

For constructing the arc set E, let $U = \{U_1, \ldots, U_{|U|}\}$ be an arbitrary enumeration of the triples in U. For any element $b \in M_j$, $j = 1, 2, 3$, we add arcs to (V, E) that constitute a directed cycle C_b. In an arbitrary order, that cycle includes exactly all vertices a_{ij} where b is the jth component of U_i. Consequently, E contains $3k$ pairwise vertex-disjoint cycles. The construction of E is completed by adding k arcs from $v_{\min}$ to $v_{\max}$; see Fig. 2.

In the following, we assume that E' is a solution to the $\mathscr{G}$-2DEE problem instance (V, E). A Eulerian tour $(V, E \cup E')$ traverses $(v_{\min}, v_{\max})$ exactly k times. The following two points state crucial properties of such a tour.

1. Let $P = v_{\max}, \ldots, v_{\min}$ be a path in $(V, E \cup E')$ that is part of a Eulerian tour and does not include the arc $(v_{\min}, v_{\max})$. Then all arcs in $P \cap E$ are contained in not more than three different cycles C_{b_1}, C_{b_2} and C_{b_3}.
2. Any Eulerian tour in $(V, E \cup E')$ includes each cycle C_b as a contiguous sub-tour.

For showing the first property, consider some arc in P that belongs to a cycle C_b with $b \in M_j$. There are only three possibilities to continue P after the sink a_{ij} of the arc has been reached. If P continues using another arc from E, then, due to the vertex-disjointness of the cycles, that arc also belongs to C_b. If P continues with an arc from E', then, due to the independence and reachability properties of V, that next arc will be either $(a_{ij}, v_{\min})$ or $(a_{ij}, a_{ij'})$ with $j' > j$. In the former case, P ends. In the latter case, P can enter a new cycle $C_{b'}$ with $b' \in M_{j'}$, but j' is strictly larger than j. In other words, each time P enters a new cycle C_b, $b \in M_j$, the index j strictly increases. As $j \leq 3$, the first property follows.

Since there are k arcs $(v_{\min}, v_{\max})$ and $3k$ cycles C_b, it follows from the first property that each cycle can be entered at most once, because otherwise some cycles would remain not entered at all. Thus, in a Eulerian tour each cycle must be completely traversed as soon at it is entered, yielding the second property.

Thus, a Eulerian tour leaves each cycle C_b at the same vertex it entered it. It follows that between any two consecutive traversals of $(v_{\min}, v_{\max})$, three cycles $C_{b_1}, C_{b_2}, C_{b_3}$ are traversed, each time starting and ending inside the same rectangle A_i. So the Eulerian extension of (V, E) has to have the form $E' = \bigcup_{h=1,\ldots,k}\{(v_{\max}, a_{i_h1}), (a_{i_h1}, a_{i_h2}), (a_{i_h2}, a_{i_h3}), (a_{i_h3}, v_{\min})\}$, where $i_1, \ldots, i_k$ are such that each cycle C_b can be traversed. This is the case if and only if no two a_{i_hj} and $a_{i_{h'}j}$ belong to the same cycle, which is equivalent to $U_{i_1}, \ldots, U_{i_k}$ constituting a matching. □

Now we are ready to show NP-completeness for $\mathrm{F3}\,|\,\mathrm{nwt}\,|\,\mathscr{G}$ respective its decision variant $\mathscr{E}_0(\mathrm{F3}\,|\,\mathrm{nwt}\,|\,\mathscr{G})$.

Theorem 6 *It is NP-complete to decide whether there is an interruption-free schedule for an instance of* $\mathrm{F3}\,|\,\mathrm{nwt}\,|\,\mathscr{G}$.

Proof We show the NP-completeness of the decision problem $\mathscr{E}_0(\mathrm{F3}\,|\,\mathrm{nwt}\,|\,\mathscr{G})$ by reduction from $\mathscr{G}$-2DEE. Our proof has the following structure: We first give an interpretation of our scheduling problem as an extension problem. Then we fix a set of properties that are only satisfied by $\mathscr{G}$-2DEE instances representing a scheduling problem instance in that way. Finally, we prove that any $\mathscr{G}$-2DEE instance can be transformed into an equivalent instance satisfying these properties.

Consider a set of jobs $J_1, \ldots, J_n$. A schedule without interruptions corresponds to a permutation $J_{\sigma(1)}, \ldots, J_{\sigma(n)}$ of the jobs, where for $1 \leq i < n$ job $J_{\sigma(i+1)}$ can be scheduled after $J_{\sigma(i)}$ without causing an idle time on the third machine. This is the case if and only if $p_{3\sigma(i)} \geq p_{2\sigma(i+1)}$ and $p_{3\sigma(i)} + p_{2\sigma(i)} \geq p_{2\sigma(i+1)} + p_{1\sigma(i+1)}$.

In terms of our extension problem, this means that the point $(p_{2\sigma(i+1)}, p_{2\sigma(i+1)} + p_{1\sigma(i+1)})$ is reachable from the point $(p_{3\sigma(i)}, p_{3\sigma(i)} + p_{2\sigma(i)})$. We associate every job J_j with an arc from $(p_{2j}, p_{2j} + p_{1j})$ to $(p_{3j}, p_{3j} + p_{2j})$. In the consequence, there is an interruption-free schedule of $J_1, \ldots, J_n$ if and only if the induced graph (V_0, E_0) admits an extension E' such that there is a path traversing each arc exactly once. This is the case if and only if there is a Eulerian extension of the graph $(V, E) = (V_0 \cup \{v_{\min}, v_{\max}\}, E_0 \cup \{(v_{\min}, v_{\max})\})$, where $v_{\min}$ ($v_{\max}$) is such that it is smaller (greater) than all other elements of V in both coordinates.

We call a graph $G^* = (V^*, E^*)$ with $V^* \subset \mathbb{R}_0^+ \times \mathbb{R}_0^+$ *legal*, if it represents a scheduling instance in the way we have just described. Respectively, an arc is called legal if it represents some job from a scheduling instance. It is not hard to observe that for an arc to be legal it suffices that it has the form $((x, y), (x', x + x'))$ with $x \leq y$. In other words, the source and sink vertex of a legal arc are above the bisectrix, and for a given source there is only one degree of freedom for the choice of the sink. For a graph to be legal it suffices that it is induced by $\hat{E} \cup \{(v_{\min}, v_{\max})\}$, where $\hat{E}$ is a set of legal arcs, and $v_{\min}$ and $v_{\max}$ are like defined in the preceding paragraph.

We complete our reduction by describing in Lemma 5 how to *legalize* an arbitrary instance $G = (V, E)$ of $\mathcal{G}$-2DEE, that is, to transform it into a legal instance $G^* = (V^*, E^*)$, where G admits a Eulerian extension if and only if G^* does. □

Lemma 5 *Each instance of $\mathcal{G}$-2DEE can be legalized in polynomial time.*

Proof First, we ensure that every vertex is above the bisectrix and no vertex has coordinates $(0, 0)$. This is achieved by vertically shifting the whole graph by $x_{\max} = \max\{x \mid (x, y) \in V\} + 1$. Technically, we obtain the graph $G_1 = (V_1, E_1)$ by adding $x_{\max}$ to the second coordinate of every vertex. As this does not change the reachability relation between any pair of vertices, G and G_1 are equivalent in the sense of $\mathcal{G}$-2DEE.

In the second transformation step, we eliminate all illegal arcs. Let arc $(u, v) = ((u_x, u_y), (v_x, v_y)) \in E_1$ be illegal. Let $y_{\max} = \max\{y \mid (x, y) \in V_1\} + 1$. We enhance V_1 by the vertices $w_1 = (y_{\max} - u_x, y_{\max})$, $w_2 = (0, y_{\max})$, $w_3 = (y_{\max}, y_{\max})$ and $w_4 = (v_y - v_x, y_{\max})$. Then, we replace (u, v) with the arcs (u, w_1), (w_2, w_3) and (w_4, v); see Fig. 3.

Straightforward observation shows that all new arcs are legal and any possible tour in a Eulerian extension must traverse the path u, w_1, w_2, w_3, w_4, v. Thus, the modified graph admits a Eulerian extension if and only if G_1 does.

We obtain the graph $G_2 = (V_2, E_2)$ by iteratively eliminating every illegal arc from G_1. Note that each such elimination causes $y_{\max}$ to increase by one. Still G_2 does not necessarily represent a flowshop scheduling instance, as there has to be an arc $(v_{\min}, v_{\max})$. We take care of this requirement in the last step of transformation. Let $u = (u_x, u_y) \in V_2$ be an arbitrary vertex. We insert $v_{\min} = (0, 0)$ and $v_{\max} = (y_{\max}, y_{\max})$ into V_2, where $y_{\max} = \max\{y \mid (x, y) \in V_2\} + 1$. Furthermore, we insert w_1 and w_2 defined like in the transformation from G_1 to G_2, and $w_4 = (u_y - u_x, y_{\max})$. Note that w_3 has already been inserted as $v_{\max}$. Then, the arcs (u, w_1), $(w_2, v_{\min})$, $(v_{\min}, v_{\max})$ and (w_4, u) are added to E_2.

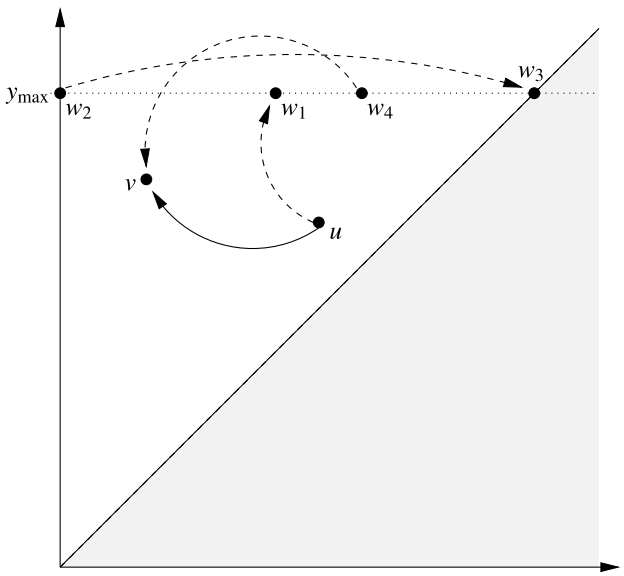


Fig. 3 The arc (u, v) is replaced with the legal arcs (u, w_1), (w_2, w_3), and (w_4, v)

As the new vertices are above the bisectrix, the new arcs are legal, and the resulting graph $G^* = (V^*, E^*)$ contains an arc from $v_{\min}$ to $v_{\max}$, it represents an instance of F3 | nwt | $\mathcal{G}$. Any Eulerian tour traverses the cycle $u, w_1, w_2, v_{\min}, v_{\max}, w_4, u$ as a closed sub-tour. Hence, G_2 and G^* are equivalent $\mathcal{G}$-2DEE instances. □

4.2 More hardness results

First we consider flexible flowshops with two stages. We show by reduction from 3-PARTITION that minimizing the number of interruptions is strongly NP-hard if the first stage contains two machines and the second stage only one.

Definition 4 (3-PARTITION) Given a set A of $3m$ elements from $\mathbb{N}^+$ with $B/4 < a < B/2$ for all $a \in A$, where $B := \frac{1}{m}\sum_{a \in A} a$, decide whether A can be partitioned into m disjoint sets $A_1, \ldots, A_m$ with $\sum_{a \in A_i} a = B$ for $i = 1, \ldots, m$.

It is well known that 3-PARTITION is NP-complete in the strong sense; see Garey and Johnson (1979).

Theorem 7 *The problem* $\mathscr{E}_0(\mathrm{FF2}\,|\,\mathrm{nwt}\,|\,\mathscr{G})$ *with* $m_1 = 2$ *and* $m_2 = 1$ *is strongly NP-complete.*

Proof Given a $3m$-element instance A of 3-PARTITION, we construct an instance I of $\mathscr{E}_0(\mathrm{FF2}\,|\,\mathrm{nwt}\,|\,\mathscr{G})$ with $m_1 = 2$ and $m_2 = 1$. For any $a \in A$ we choose a job in I with processing times 1 and a on the first and second stage, respectively. To partition these jobs, we add $m + 1$ auxiliary jobs having processing times B on the first stage and 0 on the second stage.

We show that in a schedule without interruptions, there is no point in time at which two auxiliary jobs are processed in parallel on the first stage. First notice that two auxiliary jobs running fully in parallel must cause an interruption (i) with any job scheduled before them, because no job in I has processing time B on the second stage, and (ii) with any job scheduled after them, because no job in I has processing time 0 on the first stage. Now, assume that there are auxiliary jobs J_1 and J_2 with start times $S_1 < S_2 < S_1 + B$, which, as a consequence, fully block the first stage during $[S_2, S_1 + B)$. Since all jobs in I have positive processing times on the first stage, no job can start at $S_1 + B$ on the second stage, and thus, an idle time after J_1 is unavoidable.

Therefore, we may assume that in any schedule without interruptions all auxiliary jobs are processed on the same machine in the first stage. This induces m gaps of length at least B between the auxiliary jobs on the second stage. These gaps can be filled with the remaining jobs if and only if A is a yes-instance. □

Now, we generalize this NP-completeness result and combine it with the single-machine case in Theorem 6. The following two lemmata show that $\mathscr{E}_0(\mathrm{FF}s\,|\,\mathrm{nwt}\,|\,\mathscr{G})$ with any constant number of stages $s \geq 2$ and constant numbers of machines, except $s = 2$ and $m_1 = 1$, is also strongly NP-complete.

Lemma 6 *Consider* $\mathscr{E}_0(\mathrm{FF}s\,|\,\mathrm{nwt}\,|\,\mathscr{G})$ *with a constant number of stages* $s \geq 2$. *Any variant of this problem with constant numbers of machines* $m_1, \ldots, m_s$ *where* $m_s = 1$ *can be reduced to the problem variant with constant numbers of machines* m'_i, $i = 1, \ldots, s$, *satisfying* $m'_k = m_k + 1$ *for some* $k < s$ *and* $m'_i = m_i$ *for* $i \neq k$.

Proof We consider $\mathscr{E}_0(\mathrm{FF}s\,|\,\mathrm{nwt}\,|\,\mathscr{G})$ with a constant number of stages $s \geq 2$ and constant numbers of machines. Given an instance I of a problem variant with numbers of machines m_i, $i = 1, \ldots, s$, where $m_s = 1$, we build an instance I' of a problem variant with numbers of machines m'_i as specified in the lemma. In addition to the original jobs J of I, we choose auxiliary jobs for I' which force the jobs from J to use only m_k machines on stage L_k. For blocking this stage, we add $m'_k + 1$ jobs with processing time $p = \max_{j \in J} \sum_{i=1}^{s-1} p_{ij} + \sum_{j \in J} p_{sj}$ on stage L_k, and 0 elsewhere. We denote these auxiliary jobs by A. For an arbitrary auxiliary job $J_{\mathrm{mod}} \in A$, we modify the processing time on the last stage to $q = p - \sum_{j \in J} p_{sj}$.

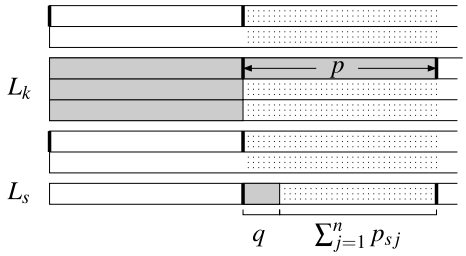


Fig. 4 Arrangement of auxiliary jobs in a schedule without interruptions (Lemma 6). Original jobs have to be processed in dotted slots

Given a schedule without interruptions for I', we consider the sub-schedule of auxiliary jobs. Observe that the idle times on the last stage in this sub-schedule sum up to at most $\sum_{j \in J} p_{sj}$, since otherwise they could not be filled with the jobs J in the total schedule. We examine the arrangement of jobs from A. By its cardinality, there is a machine in stage L_k that processes a subset $\tilde{A} \subseteq A$ of at least two jobs. Due to the largest possible length of idle times on the only machine of the last stage, $\tilde{A}$ contains exactly two jobs, the first of them being w.l.o.g. J_{mod}. Hence, the jobs from $\tilde{A}$ generate an idle time of length $\sum_{j \in J} p_{sj}$ on the last stage. Since no further idle times are allowed, the remaining jobs from A must be scheduled such that they exactly meet the previous jobs on the last stage. Thus, we can assume that the auxiliary jobs are arranged as in Fig. 4.

By definition of p and q there exists a schedule without interruptions for the instance I' if and only there exists one for the instance I. □

Lemma 7 *Consider* $\mathscr{E}_0(\mathrm{FF}s\,|\,\mathrm{nwt}\,|\,\mathscr{G})$ *with a constant number of stages* $s \geq 2$. *Then any variant of this problem with constant numbers of machines* $m_1, \ldots, m_s$ *can be reduced to any other problem variant with constant numbers of machines* m'_i, $i = 1, \ldots, s$, *satisfying* $m'_s > m_s$ *and* $m'_i = m_i$ *for* $i = 1, \ldots, s - 1$.

Proof We reduce the problem $\mathscr{E}_0(\mathrm{FF}s\,|\,\mathrm{nwt}\,|\,\mathscr{G})$ with constant numbers of machines $m_1, \ldots, m_s$ to another variant of this problem with machine numbers m'_i, $i = 1, \ldots, s$, satisfying $m'_s > m_s$ and $m'_i = m_i$ elsewhere. Given an instance I of the former problem with set of jobs J, we build an instance I' of the latter problem by adding auxiliary jobs to the instance I: We choose m_s jobs with processing time $q = \max_{j \in J} \sum_{i=1}^{s} p_{ij}$ on the last stage, and 0 on all other stages. Furthermore, we add $m'_1 m'_s$ jobs with processing time $p > (m_s + 1)q + \sum_{j \in J} p_{sj}$ on stage L_1, and 0 elsewhere. We denote these jobs with J^q_{aux} and J^p_{aux}, respectively.

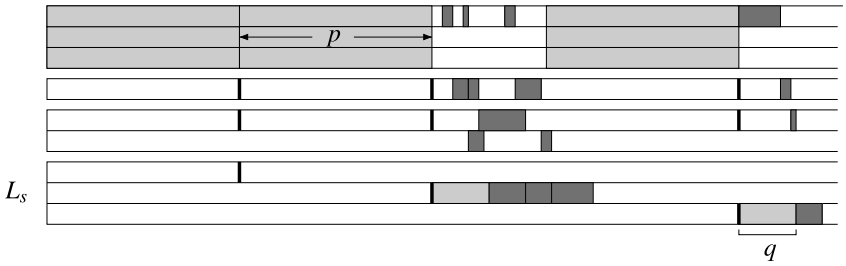


Fig. 5 Schedule without interruptions for I'. (Lemma 7)

If there exists an interruption-free schedule S for I, then the schedule in Fig. 5 shows how to arrange the jobs from I' without idle times on the last stages: The jobs from J^p_{aux} are scheduled in m'_s blocks of m'_1 jobs which are processed in parallel on the first stage and on the same machine in the last stage, choosing separate machines on the last stage for each block. With sufficiently long idle times between the blocks, we can process on m_s of the last stage machines additionally one job from J^q_{aux} and then a sub-schedule that corresponds to a last stage machine in S.

On the other hand, given a schedule for I' without interruptions, we can construct a schedule without interruptions for I. In the following we call jobs with processing time 0 on the stages $L_1, \ldots, L_{s-1}$ *zero-jobs*. We claim that in any interruption-free schedule for I' every machine on the last stage processes first one or more jobs from J^p_{aux}, second a zero-job and third an interruption-free sequence of jobs from $J \cup J^q_{\text{aux}}$ (which may also include jobs from J^p_{aux}).

We consider a schedule for I' that does not contain an interruption. Then, the last stage idle times in the sub-schedule induced by the jobs J^p_{aux} do not equal or exceed p, by definition of p. Thus, there is no pair of jobs from J^p_{aux} that is processed on the same machine on the first and last stage. In particular, every first stage machine processes exactly m'_s jobs from J^p_{aux}. Hence, given a machine on the first and last stage, there is a unique job from J^p_{aux} which is processed on both machines. As a consequence, the m'_1 jobs from J^p_{aux}, which are processed on a particular machine M on the last stage, use every machine in the first stage for processing exactly one job.

Let c be the first completion time of these jobs, and let $J \in J^p_{\text{aux}}$ be a job completing at that time. Since idle times on the last stage must be compensable with the jobs from I', none of the jobs from J^p_{aux} on M completes after $d = c + m_s q + \sum_{j \in J} p_{sj}$. Thus, these jobs block the first stage during $[d - p, c)$. This yields two characteristics for schedules without interruptions: First, J can be followed on machine M only by zero-jobs. And second, by

$$\left|[d-p,c)\right| = p - m_s q + \sum_{j \in J} p_{sj} > q = \max_{j \in J} \sum_{i=1}^{s} p_{ij},$$

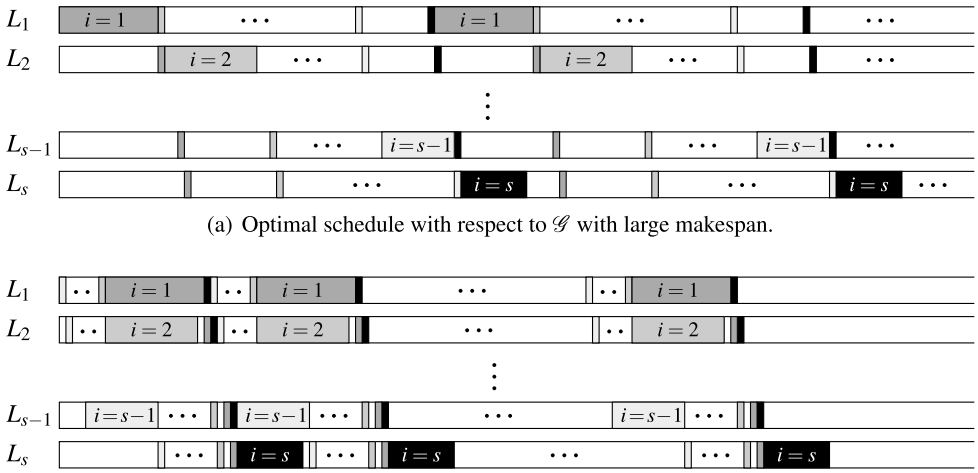
no job can be processed before J on M. Since the jobs from J^p_{aux} have processing time 0 on the last stage, all further jobs from $J \cup J^q_{\text{aux}}$ on M follow the zero-job without idle time. This completes the proof of the claim.

Given a schedule for I' with the properties claimed above, we show how to construct an interruption-free schedule for the instance I. Consider all machines on the last stage that process a job from $J \cup J^q_{\text{aux}}$. If there is a machine that processes more than one job from J^q_{aux}, then we move the sequence of jobs, starting with the second job from J^q_{aux} (ignoring the jobs from J^p_{aux}), to a machine on the last stage, which processes no job from J^q_{aux}. Since the jobs from J^q_{aux} are zero-jobs, this does not create an interruption on the new machine. If there occur conflicts with other jobs, then we delay sub-schedules corresponding to some machines on the last stage. In case that a machine processes only jobs from J, but no job from J^q_{aux}, then by our claim, the first job is a zero-job. Hence, we can move the sequence of jobs from J to another machine, which already processes a job from J^q_{aux}. Thus, we may assume that there are at most m_s machines on the last stage that process jobs from J and each of these machines processes one job from J^q_{aux}. If the job from J^q_{aux} is not the first one in the sequence of jobs from $J \cup J^q_{\text{aux}}$, then we can move the subsequence starting with it to the beginning. Since the first job in the sequence and the job from J^q_{aux} are zero-jobs, this creates no interruption. Finally, we obtain a schedule for I' that uses only m_s machines on the last stage, and in which the jobs of J are processed without interruption. This yields an interruption-free schedule for the instance I. □

Theorem 4 is proven by combining the results of this subsection with Theorem 6.

5 Alternative scheduling models

The scheduling model used in the previous sections bears two problems: (i) it is inapproximable for almost every arrangement of machines, as we have shown in Sect. 4, and (ii) it may lead to schedules with unacceptable makespan, as we demonstrate in Sect. 5.2. Therefore, it is natural to consider other scheduling models that aim for a small number of interruptions.



(a) Optimal schedule with respect to $\mathcal{G}$ with large makespan.

(b) Optimal schedule with respect to $\mathcal{G}$ with small makespan.

Fig. 6 Instance of Fs | nwt | $\mathcal{G}$ with $s \geq 2$ machines. For all $i = 1, \ldots, s$ and some $k \in \mathbb{N}$, choose k copies of a job with processing time $p + (i-1)\varepsilon$ on machine i and ε elsewhere for $p \gg \varepsilon > 0$. The makespans of the schedules in 6(a) and 6(b) differ asymptotically about a factor of s

5.1 Maximizing the number of non-interruptions

We consider the maximization variant corresponding to the problem of minimizing $\mathcal{G}$. That is, instead of counting the number of interruptions, we count how often jobs run on the last stage without incurring idle time after them, which we call a *non-interruption*. Since a schedule maximizes the number of non-interruptions if and only if it minimizes the number of interruptions, the optimal algorithm for F2 | nwt | $\mathcal{G}$ from Sect. 3 and the NP-hardness results from Sect. 4 transfer immediately. However, the situation is completely different concerning approximability. In fact, we derive a constant factor approximation for the single-machine maximization problem for three or more stages, whereas the minimization variants of those problems are inapproximable.

Theorem 8 *The no-wait flowshop problem of maximizing the number of non-interruptions can be approximated with factor* $3/4$ *in the single-machine case.*

Proof We simply reduce our problem to a maximum asymmetric Traveling Salesperson Problem (maxATSP): Given a directed complete graph, find a TSP tour of maximum weight (Bläser 2004). We interpret every job as a vertex in an instance of maxATSP and choose weight 1 for an arc between two nodes if the corresponding jobs can be processed without interruption, and 0 otherwise. Additionally, we choose an auxiliary vertex whose adjacent arcs all have weight 0. Clearly, a tour in this graph, interrupted at the auxiliary vertex, yields a schedule with as many non-interruptions as the cost of the tour. Thus, the 3/4-approximation algorithm for maxATSP with cost 0 and 1 given by Bläser (2004) solves the problem of maximizing the number of non-interruptions with the same performance guarantee. □

5.2 A model combining $\mathcal{G}$ and $C_{\max}$

Even though practitioners ask for minimizing the number of interruptions, this objective does not allow any control over the makespan $C_{\max}$ of a schedule, and hence, may lead to solutions that are unreasonable in practice. Figure 6 demonstrates such an example in which two schedules that are optimal with respect to $\mathcal{G}$ have makespans that differ by a factor in the order of the number of stages. This is in practice an unrealistic scenario, which justifies a new model, which combines both the minimization of $\mathcal{G}$ and $C_{\max}$.

We propose a model in which we aim for minimizing the makespan under the restriction that a certain minimum idle time length $\lambda \geq 0$ is satisfied. From a practical point of view, λ can be seen as unavoidable cleaning time in case of a strand interruption. We denote this job characteristic by λ in the second component of the three-field scheduling notation.

Depending on the choice of λ, the optimization focus is set more on minimizing the pure makespan or the number of interruptions. If $\lambda = 0$, we have the traditional makespan minimization problem. Thus, it follows from the strong NP-hardness of F3 | nwt | $C_{\max}$ (Röck 1984) that the generalized problem FFs | nwt, λ | $C_{\max}$ is also NP-hard in the strong sense for any constant number of stages $s \geq 3$ and all constant numbers of machines. We can simply reduce the decision problem corresponding to F3 | nwt | $C_{\max}$ with respect to a makespan T to any problem with more machines or stages by blocking the additional machines with auxiliary jobs with processing time T on one stage and zero elsewhere. Analogously, the strong NP-hardness of the two-stage problem FF2 | nwt, λ | $C_{\max}$ with one of the stages containing more than one machine follows from Sriskandarajah and Ladet (1986), where it is shown that FF2 | nwt | $C_{\max}$ with one single-machine stage and one two-machine stage is strongly NP-hard. Thus, only the complexity of the single processor two-stage case of our generalized problem, F2 | nwt, λ | $C_{\max}$, remains open.

In contrast to this analogy to the makespan minimization problem, a large minimal interruption length λ puts more emphasis on the minimization of the number of interruptions. For an extreme case we make the following observation.

Proposition 1 *For $\lambda > \max_{j\in J} \sum_{i=1}^{s-1} p_{ij}$, every optimal solution of an instance I of* FFs | nwt, $\lambda \mid C_{\max}$ *with $m_s = 1$ also minimizes $\mathscr{G}$.*

Proof Given a makespan optimal schedule S^* for I, we assume that there exists a schedule S with less interruptions than S^*. By the choice of λ, idle times on the last stage have either length 0 or λ, and thus,

$$\begin{aligned} C_{\max}(S) &\le \max_{j\in J} \sum_{i=1}^{s-1} p_{ij} + \sum_{j\in J} p_{sj} + \lambda \mathscr{G}(S) \\ &< \sum_{j\in J} p_{sj} + \lambda\big(\mathscr{G}(S)+1\big) \\ &\le \sum_{j\in J} p_{sj} + \lambda \mathscr{G}(S^*) \le C_{\max}(S^*), \end{aligned}$$

which contradicts the optimality of S^*. □

Note that this is not true for $m_s > 1$, since an unavoidable interruption on one machine allows the other machines to have an interruption at the same time without increasing the makespan.

In the following, we investigate approximation algorithms for minimizing $C_{\max}$ in the model with minimum interruption length. By the above considerations, it is a natural approach to use algorithms for minimizing the classical makespan or the number of interruptions to construct approximation algorithms for our generalized problem. In the single-machine case every schedule not satisfying the minimal interruption length of λ can be made feasible by just enlarging the idle times to the appropriate length. Hence, we can simply transform schedules from the model without minimal interruption length to feasible schedules in our model. In case of parallel machines this is not necessarily possible, since jobs may overtake each other on stages with more than one machine.

First, we consider an approximation algorithm which is based on the exact algorithm for F2 | nwt | $\mathscr{G}$ from Sect. 3.

Theorem 9 *The problem* F2 | nwt, $\lambda \mid C_{\max}$ *can be approximated with factor* 2 *in polynomial time.*

Proof We interpret a given instance I of F2 | nwt, $\lambda \mid C_{\max}$, as an instance I' of F2 | nwt | $\mathscr{G}$. We solve this problem optimally in polynomial time (Sect. 3) and obtain an optimal schedule S' for I'. We show that the schedule S, which is obtained by enlarging the interruption lengths of S' to at least λ, satisfies the claimed approximation factor.

Suppose that the jobs are indexed in the order of the sequence S. To express the makespan of S, we use a binary variable δ_j indicating if there is an interruption after job J_j. Then

$$\begin{aligned} C_{\max}(S) &= p_{11} + \sum_{j=1}^{n} p_{2j} + \sum_{j=1}^{n-1} \delta_j \max\{\lambda,\, p_{1,j+1} - p_{2j}\} \\ &\le p_{11} + \sum_{j=1}^{n} p_{2j} + \lambda \mathscr{G}(S') + \sum_{j=1}^{n-1} \delta_j (p_{1,j+1} - p_{2j}) \\ &\le \sum_{j=1}^{n} p_{2j} + \lambda \mathscr{G}(S') + \sum_{j=1}^{n} p_{1j}. \end{aligned}$$

The lower bounds $\sum_{j=1}^{n} p_{2j} + \lambda \mathscr{G}(S')$ and $\sum_{j=1}^{n} p_{1j}$ on the optimal makespan conclude the proof. □

A simple example with two jobs and processing times $p_{11} = p_{22} = \varepsilon$, $p_{12} = p + \varepsilon$, and $p_{21} = p$ for some $p \gg \varepsilon > 0$ shows that the approximation factor is in fact tight.

We continue with investigating solution methods that are based on algorithms for the classical makespan minimization problem. When λ is part of the input or a large constant, this approach generally fails. Already simple examples show that schedules with optimal makespan may have the maximum number of interruptions whereas there exist idle time free schedules; see Fig. 1 in Sect. 2. On the other hand, when λ is bounded by some constant multiple of the average processing time per stage, then approximation algorithms for the classical makespan problem lead to reasonable algorithms for the problem with minimal interruption length.

Proposition 2 *Any k-approximation algorithm for the problem* Fs | nwt | $C_{\max}$ *yields a $(k+r)$-approximation algorithm for* Fs | nwt, $\lambda \mid C_{\max}$, *where $\lambda \le r\,(\sum_{i,j} p_{ij})/(s\,n)$.*

Proof We construct a schedule S for an instance of the problem Fs | nwt, $\lambda \mid C_{\max}$ as follows: first, we compute a schedule S' applying a k-approximation algorithm for the classical makespan problem, and then we enlarge the idle time slots on the last stage in S' to their required length. Then

$$C_{\max}(S) \le C_{\max}(S') + (n-1)\lambda \le C_{\max}(S') + r \sum_{i,j} p_{ij}/s.$$

The optimal makespan for the classical makespan version of the problem and $\sum_{i,j} p_{ij}/s$ are both lower bounds on the optimal makespan with minimal interruption length. This concludes the proof. □

Using an optimal algorithm for $F2\,|\,nwt\,|\,C_{max}$ (see Gilmore and Gomory 1964), Proposition 2 yields a $(1+r)$-approximation for the generalized model. For the larger problem class $Fs\,|\,nwt\,|\,C_{max}$ the polynomial time approximation scheme by Sviridenko and Woeginger (2000), leads to an $(1+r+\varepsilon)$-approximation algorithms for $Fs\,|\,nwt,\lambda\,|\,C_{max}$.

Combining both algorithms in Theorem 9 and Proposition 2, that is, returning simply the better result, yields an approximation algorithm for the problem $F2\,|\,nwt,\lambda\,|\,C_{max}$ with factor $\min\{2, \lambda sn/(\sum_{i,j} p_{ij}) + 1\}$.

6 Conclusions

We considered a new, practically relevant flowshop scheduling problem and gave a full analysis of its complexity status. We interpreted the flowshop problem as a Eulerian Extension Problem (or Rural Postman Problem) to obtain polynomial time algorithms but also to derive NP-hardness results. This concept of interpreting sequencing problems as Eulerian Extension Problems does not only lead to elegant algorithms, it also allows an implicit representation of all optimal solutions. We believe that this technique influences also other algorithmic frameworks for related problems and may lead also to approximate solutions.

Our algorithmic results for $FF2\,|\,nwt\,|\,\mathscr{G}$ essentially require that no stage can be skipped by a job. Otherwise the problem becomes NP-hard even in the single-machine case with two stages, which can be shown by similar arguments as in the makespan minimization version (Sahni and Cho 1979).

However, our results generalize to the case where the no-wait constraint is replaced by (possibly negative) waiting times between completing a job on one stage and starting it on the next one. In the reduction from the variant with two single-machine stages to G-1DEE, we now add job arcs from Δ_{start} to Δ_{compl}, where Δ_{start} is the (possibly negative) difference between the starting times of a job on the two stages, and Δ_{compl} is the difference between the completion times. Note that this construction generalizes all algorithmic results in Sect. 3, and moreover, it can even handle the case where the waiting times are individual for each job. However, it is still necessary to require that the job permutation is equal on each stage, i.e., jobs are not allowed to overtake each others. Otherwise the problem is NP-hard, since one can simulate machine skipping of a job by adding a small processing time on the skipped stage and making the delay of the job so large that its processing time slot on the skipped stage can never collide with any other job.

The view as a Eulerian Extension Problem raises interesting potential for multi-criteria optimization. As a first step toward multi-criteria optimization, consider both objective functions, $\mathscr{G}$ and C_{max}. Already simple examples (even on two stages) show that schedules with optimal makespan may have the maximum number of interruptions whereas there exist idle time free schedules. For the two-stage variants of these problems, our technique provides us an implicit representation of all optimal solutions from which one could choose accordingly. Using insights from the proof of Theorem 9, this implies that an algorithm that solves the interruption problem optimally, yields a schedule of makespan no greater than twice the minimum makespan, giving a trivial $(1,2)$-approximation for the bicriteria objective $(\mathscr{G}, C_{max})$. Moreover, it follows from the NP-completeness of $FF2\,|\,nwt, m_2 \geq 2\,|\,C_{max}$ (Sriskandarajah and Ladet 1986) that approximating such problems with a $(c,1)$-guarantee is NP-hard for any constant $c \geq 1$.

References

Ball, M. O., & Magazine, M. J. (1988). Sequencing of insertions in printed circuit board assembly. *Operations Research*, *36*(2), 192–201.

Bläser, M. (2004). A 3/4-approximation algorithm for maximum asymmetric TSP with weights zero and one. In *Lecture Notes in Computer Science: Vol. 3122. Proceedings of the 7th international workshop on approximation algorithms for combinatorial optimization problems (APPROX)* (pp. 61–71). Berlin: Springer.

Eiselt, H. A., Gendreau, M., & Laporte, G. (1995). Arc routing problems II. The rural postman problem. *Operations Research*, *43*(3), 399–414.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability*. New York: Freeman.

Giaro, K. (2001). NP-hardness of compact scheduling in simplified open shop and flowshop. *European Journal of Operational Research*, *130*, 90–98.

Giaro, K., & Kubale, M. (2004). Compact scheduling of zero-one time operations in multi-stage systems. *Discrete Applied Mathematics*, *145*, 95–103.

Gilmore, P. C., & Gomory, R. E. (1964). Sequencing a one state-variable machine: a solvable case of the traveling salesman problem. *Operations Research*, *12*, 655–679.

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.

Gutin, G., & Punnen, A. P. (2002). *The traveling salesman problem and its variations*. Berlin: Springer.

Hall, N. G., & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, *44*(3), 510–525.

Harjunkoski, I., & Grossmann, I. (2001). A decomposition approach for the scheduling of a steel plant production. *Computers & Chemical Engineering*, *25*, 1647–1660.

Kabadi, S. N. (2002). New polynomially solvable classes and a new heuristic for the traveling salesman problem and its generalization. *Discrete Applied Mathematics*, *119*, 149–167.

Karp, R. M. (1972). Reducibility among combinatorial problems. In R. Miller & J. Thatcher (Eds.), *Complexity of computer computations* (pp. 85–103). New York: Plenum Press.

Laporte, G., & Osman, I. H. (1995). Routing problems: a bibliography. *Annals of Operation Research*, *61*, 227–262.

Lenstra, J. K., & Kan, A. H. G. R. (1976). On general routing problems. *Networks*, *6*(3), 273–280.

Orloff, C. (1974). A fundamental problem in vehicle routing. *Networks*, *4*, 35–64.

Pacciarelli, D., & Pranzo, M. (2004). Production scheduling in a steelmaking-continuous casting plant. *Computers & Chemical Engineering*, *28*(12), 2823–2835.

Piehler, J. (1960). Ein Beitrag zum Reihenfolgenproblem. *Unternehmensforschung*, *4*, 138–142.

Reddi, S. S., & Ramamoorthy, C. V. (1972). On the flow-shop sequencing problem with no wait in process. *Operational Research Quarterly*, *23*(3), 323–331.

Röck, H. (1984). The three-machine no-wait flow shop is NP-complete. *Journal of the Association for Computing Machinery*, *31*(2), 336–345.

Rote, G., & Woeginger, G. J. (1998). Time complexity and linear-time approximation of the ancient two-machine flow shop. *Journal of Scheduling*, *1*(3), 149–155.

Sahni, S., & Cho, Y. (1979). Complexity of scheduling shops with no-wait in process. *Mathematics of Operations Research*, *4*, 448–457.

Schwindt, C., & Trautmann, N. (2003). Scheduling the production of rolling ingots: industrial context, model, and solution method. *International Transactions in Operational Research*, *10*(6), 547–563.

Sriskandarajah, C., & Ladet, P. (1986). Some no-wait shops scheduling problems: complexity aspect. *European Journal of Operational Research*, *24*(3), 424–438.

Sviridenko, M., & Woeginger, G. J. (2000). Approximability and inapproximability results for no-wait shop scheduling. In *41st Annual symposium on foundations of computer science (FOCS)* (pp. 116–125).

Vairaktarakis, G. L. (2003). Simple algorithms for Gilmore–Gomory's traveling salesman and related problems. *Journal of Scheduling*, *6*(6), 499–520.

Wang, Z., Xing, W., & Bai, F. (2005). No-wait flexible flowshop scheduling with no-idle machines. *Operations Research Letters*, *33*, 609–614.

West, D. B. (2001). *Introduction to graph theory* (2nd edn.). Englewood Cliffs: Prentice-Hall.