Journal of Electronic Systems

DLINE JOURNALS

Print ISSN: 2278 – 652X
Online ISSN: 2278 – 6538

JES 2025: 15 (2)
https://doi.org/10.6025/jes/2025/15/2/81-96

Solving the Parameterized Complexity of Machines

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

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

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

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

ABSTRACT

We present new computational hardness findings for Interval Scheduling on Unrelated Machines. This is a traditional scheduling challenge inspired by just-in-time or lean manufacturing practices, where the objective is to finish tasks precisely by their deadlines. We have n tasks and m machines available. Each task is accompanied by a deadline, a weight, and a processing time that may vary across different machines. The aim is to devise a schedule that maximizes the total weight of tasks completed precisely by their deadlines. It is essential to note that this creates a distinct processing time interval for each task on each machine. Interval Scheduling on Unrelated Machines has strong connections to coloring interval graphs and has been extensively examined over the years. However, as highlighted by Mnich and van Bevern [Computers & Operations Research, 2018], the parameterized complexity regarding the number m of machines remains unresolved. We address this by demonstrating that Interval Scheduling on Unrelated Machines is W[1]-hard when the parameter is the number m of machines. To achieve this, we establish W[1]-hardness with respect to m in the specific scenario of having parallel machines with designated machine sets for tasks. This resolves Open Problem from Mnich and van Bevern's compilation of unresolved questions in the parameterized complexity of scheduling.

Keywords: Parameterized Complexity, Interval Scheduling, W[1]-hardness, Unrelated Machines

Received: 19 October 2024, Revised 8 January 2025, Accepted 29 January 2025

1. Introduction

In scheduling problems, we wish to assign jobs to machines in order to maximize a certain optimization objective while respecting certain constraints. In many traditional scheduling settings, jobs can be scheduled to start at any point in time and then need a given *processing time* to be completed. However, in a typical *interval scheduling problem*, each job can be processed only in a fixed time interval, or sometimes in a set of time intervals, that may vary from machine to machine [16, 17]. Many different variations of interval scheduling have been considered and investigated [1, 2, 4, 5, 6, 7, 27].

In interval scheduling in its most basic form, we are given a set of *n jobs* and a set of *m* identical parallel *machines* that each can process one job at a time. Each job has a *processing time*, a *deadline*, and a *weight*, and shall be processed such that it finishes exactly at its deadline. This uniquely defines an interval for each job in which it can be processed. A *schedule* assigns a subset of the jobs to machines. Unassigned jobs are rejected. We call a schedule *feasible* if no two jobs with overlapping processing time intervals are assigned to the same machine. The goal is to find a feasible schedule that maximizes the weighted number of scheduled jobs.1

This setting corresponds to the concept of *just-in-time (JIT)* or *lean manufacturing* that revolutionized industrial production processes in the 1980s and 1990s [18, 23, 26, 30, 31]. Herein, the main goal is to provide and receive goods precisely when they are needed in order to reduce storage costs and wastage. The first implementation of this manufacturing paradigm is attributed to the Japanese automobile company Toyota and is sometimes also called Toyota Production System (TPS) [23, 26]. Naturally, just-in-time and interval scheduling in many different variants has received much attention from the research community since the late 1980s until today [1, 3, 4, 5, 6, 7, 9, 13, 19, 20, 25, 27, 28].

The basic form of interval scheduling as described above is known to be solvable in polynomial time [2, 7, 8, 9, 13]. It is closely related to the classical problems of finding maximum independent sets in interval graphs and coloring interval graphs [8, 11, 24, 32]. The jobs of an interval scheduling instance naturally define an interval graph with vertex weights. For example, if there is only one machine, then interval scheduling is equivalent to finding a maximum weight independent set in an interval graph. Coloring an interval graph or, more specifically, computing its chromatic number is equivalent to determining the minimum number of machines necessary to schedule all jobs. In our work, we investigate several natural generalizations and variants of interval scheduling and answer some longstanding open questions about their (parameterized) computational complexity.

The first problem we consider in this paper is Interval Scheduling on Eligible Machines, a natural generalization of the basic interval scheduling problem we introduced earlier. Here, each job additionally has a set of eligible machines and each job can only be assigned to a machine in this set in a feasible schedule. Arkin and Silverberg [2] proved in 1987 that Interval Scheduling on Eligible Machines is strongly NP-hard and can be solved in $O(mnm+1)$ time. In terms of parameterized complexity, Arkin and Silverberg [2] showed that Interval Scheduling on Eligible Machines is in XP when parameterized by the number m of machines. However, they left open whether

[1]In the standard three field notation for scheduling problems of Graham [12] this problem is sometimes denoted by $P \mid p_j = d_j - r_j \mid \Sigma_j w_j U_j$, or $P \mid\mid \Sigma_j w_j E_j$, or $P \mid\mid$ JIT. We give a more formal definition in Section 2.

Interval Scheduling on Eligible Machines also admits an FPT-algorithm for parameter m. Mnich and van Bevern [22] included this question as Open Problem 8 in their 2018 list of 15 open problems in the parameterized complexity of scheduling. We answer this question negatively in our first main contribution of this paper.

Theorem 1. *Interval Scheduling on Eligible Machines is strongly*[2] *W[1]-hard when parameterized by the number m of machines.*

A natural and well-studied generalization of Interval Scheduling on Eligible Machines is Interval Scheduling on Unrelated Machines. In the latter, the processing time of each job can be machine-dependent whereas the deadline stays the same on all machines. Furthermore, each job is eligible on all machines. This definition stems from the just-in-time motivation, where each job should be finished exactly at its deadline but on different machines it may take different times to complete the job. We mention in passing that if both processing times and deadlines can be machine-dependent, the problem becomes NP-hard on two machines [17, 27]. Sung and Vlach [28] showed that Interval Scheduling on Unrelated Machines can also be solved in $O(mnm+1)$ time, generalizing the result of Arkin and Silverberg [2]. Mnich and van Bevern [22] asked in Open Problem 8 for an FPTalgorithm for Interval Scheduling on Eligible Machines parameterized by the number m of machines as a first step towards finding an FPT-algorithm for Interval Scheduling on Unrelated Machines parameterized by m. However, Theorem 1 naturally implies that Interval Scheduling on Unrelated Machines presumably also does not admit an FPT-algorithm for the number m of machines as a parameter.

Corollary 2. *Interval Scheduling on Unrelated Machines is strongly W[1]-hard when parameterized by the number m of machines.* We point out that all known hardness reductions for Interval Scheduling on Unrelated Machines require job weights, raising the question whether the weights play an integral role in the computational complexity of the problem. Unweighted Interval Scheduling on Unrelated Machines is the natural special case of Interval Scheduling on Unrelated Machines where all jobs have weight one. Sung an Vlach [28] asked in 2005 to resolve the computational complexity status of Unweighted Interval Scheduling on Unrelated Machines. We give an answer to this in our second main contribution.

Theorem 3. *Unweighted Interval Scheduling on Unrelated Machines is NP complete.*

We remark that our reduction for Theorem 3 does not imply hardness for the unweighted version of Interval Scheduling on Eligible Machines. We leave this open for future research. An additional immediate question that we leave open for future research is whether Unweighted Interval Scheduling on Unrelated Machines admits an FPT-algorithm for the number m of machines as a parameter.

With Theorem 1, Corollary 2, and Theorem 3 we answer fundamental longstanding open questions concerning the (parameterized) computational complexity of natural interval scheduling problems. For Interval Scheduling on Eligible Machines and Interval Scheduling on Unrelated Machines, our results together with the XP-containment results from Arkin and Silverberg [2] and Sung and Vlach [28], respectively, essentially resolve their parameterized complexity classification for the number m of machines as a parameter. We point out that all considered problem variants are known to be fixed-parameter tractable when parameterized by the number n of jobs. This can be shown with a simple reduction to Multicolored Independent Set on Interval Graphs

[2]A parameterized problem is *strongly* W[1]-hard if it remains W[1]-hard when all numbers are encoded unarily.

parameterized by the number of colors, which is known to be fixed-parameter tractable [4, 5]. Hence, we make an important further step towards fully understanding the parameterized complexity of several basic and natural interval scheduling problems. We remark that our results also imply that Multicolored Independent Set on Interval Graphs is W[1]-hard when parameterized by the maximum number of vertices of any color.

The rest of the paper is organized as follows: we give formal definitions of all problems in Section 2. We prove Theorem 1 and Corollary 2 in Section 3 and we prove Theorem 3 in Section 4. We conclude with future research directions in Section 5.

2. Problem Setting

The first problem we consider is Interval Scheduling on Eligible Machines. Here, we have a set of n jobs $\{j_1, j_2., j_n\}$ and a set of m machines $\{i_1, i_2, \ldots, i_m\}$ that each can process one job at a time. Each job j has a *processing time* p_j, a *deadline* d_j, a *weight* wj, and a set of *eligible machines* $M_j \subseteq \{i_1, i_2, \ldots, i_m\}$. Job j can be processed in exactly *one* fixed time interval $(d_j - p_j, d_j]$, specified by its processing time and deadline, that is the same on each of its eligible machines. A schedule is a mapping from jobs to machines. More formally, a *schedule* is a function $\sigma : \{j_1, j_2, \ldots, j_n\} \rightarrow \{i_1, i_2, \ldots, i_m, \perp\}$. If for job j we have $\sigma(j) = i$ (with $i \neq \perp$), then job j is scheduled to be processed on machine i. If for job j we have $\sigma(j) = \perp$, then job j is not scheduled, that is, it is not assigned to any machine. We say that two jobs j, j' are *in conflict* on a machine i if $(d_j - p_j, d_j] \cap (d_{j'} - p_{j'}, d_{j'}] \neq \emptyset$, that is, the processing time intervals corresponding to jobs j and j' on machine i overlap. A schedule σ is *feasible* if there is no pair of jobs j, j' with $\sigma(j) = \sigma(j') = i \neq \perp$ that is in conflict on machine i and each job is mapped to one of its eligible machines. The goal is to find a feasible schedule that maximizes the weighted number of scheduled jobs $W = \Sigma_j |\sigma_{(j) \neq \perp} w_j$. In the standard three field notation for scheduling problems of Graham [12] Interval Scheduling on Eligible Machines is sometimes denoted by $P \mid M_j, p_j = d_j - r_j \mid \Sigma_j w_j U_j$, or $P \mid M_j \mid \Sigma_j w_j E_j$, or $P \mid M_j \mid$ JIT.

The second problem we consider is Interval Scheduling on Unrelated Machines. Here, for each job j the processing time $p_{i,j}$ can depend on machine i whereas the deadline d_j is the same on all machines. Hence, the processing time interval of job j on machine i is $(d_j - p_{i,j}, d_j]$. Moreover, the all jobs are eligible on all machines, that is, $M_j = \{i_1, i_2, \ldots, i_m\}$ for all jobs j. In the standard three field notation for scheduling problems of Graham [12] Interval Scheduling on Unrelated Machines is sometimes denoted by $R|p_j = d_j - r_j |\Sigma_j w_j U_j$, or $R||\Sigma_j w_j E_j$, or $R||$JIT.

Finally, the third problem we consider is Unweighted Interval Scheduling on Unrelated Machines, the unweighted version of Interval Scheduling on Unrelated Machines. Here, we have that $w_j = 1$ for all jobs j. In the standard three field notation for scheduling problems of Graham [12] Unweighted Interval Scheduling on Unrelated Machines is sometimes denoted by $R| p_j = d_j - r_j |\Sigma_j U_j$, or $R||\Sigma_j E_j$, or $R| w_j = 1|$ JIT.

3. W[1]-Hardness of Interval Scheduling on Eligible Machines

In this section, we prove Theorem 1 from which Corollary 2 follows directly. To prove Theorem 1, we present a parameterized polynomial-time reduction from Multicolored Clique parameterized by the number of colors to Interval Scheduling on Eligible machines parameterized by the number m of machines. In Multicolored Clique, we are given a k-partite graph $G = (V1 \uplus V2 \uplus \ldots \uplus V_k, E)$, we are asked whether G contains a clique of size k. The k vertex parts $V_1, V_2, \ldots, V_k$ are called *colors*. Multicolored Clique parameterized by k is known to be W[1]-hard [10].

Given an instance of Multicolored Clique, we construct an instance of Interval Scheduling on Eligible Machines as follows.

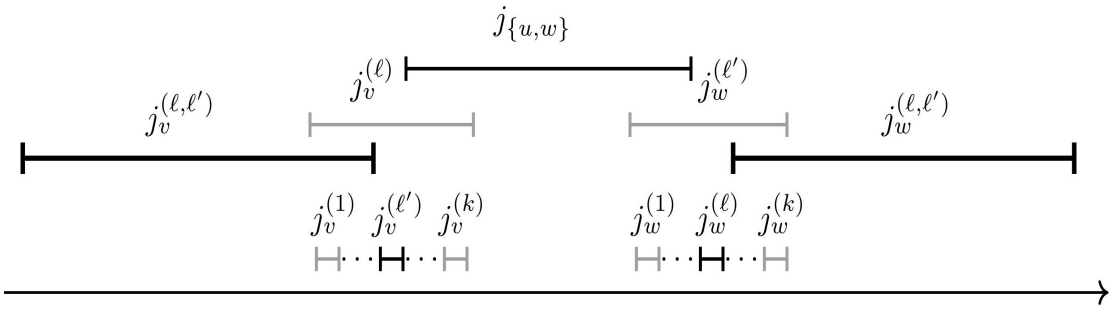

Figure 1. Illustration of the edge selection machine for color combination ℓ,-ℓ' with ℓ<–ℓ'. Depictedare intervals of jobs relating to $v \in V_\ell$, $w \in V_{\ell'}$, and $e = \{v, w\} \in E$. Gray intervals correspond to jobs that are not eligible on the machine

Construction 1. Let $G = (V1 \uplus V2 \uplus \ldots \uplus V_k, E)$ be a k-partite graph with n_G vertices. Assume we have some total ordering $<_\pi$ over $V := V1 \uplus V2 \uplus \ldots \uplus V_k$ such that for all $v \in V_\ell$ and $w \in V_{\ell'}$ we have that if $\ell < \ell'$ then $v <_\pi w$. Let $\pi(v)$ denote the ordinal position of $v \in V$ in the ordering $<_\pi$.

In the following, we describe the jobs and specify their processing times, deadlines and weights. Then we describe the machines and the eligible machine sets for the jobs. In order to describe the weights more easily, we introduce the following three values: $c1 = n_G + 1$, $c_2 = (k-1)n_G c_1 + n_G + 1$, and $c_3 = (kn_G + k^2 n_G)\, n_G c_2 + 1$. We create the following jobs:

• For each vertex $v \in V$, we create k *vertex jobs* $j_v^{(1)}, j_v^{(2)}, \ldots, j_v^{(k)}$, where one of the vertex jobs corresponds to the color of v and the k -1 other vertex jobs correspond to the other k -1 colors.

Let $v \in V_\ell$. The processing time of $j_v^{(\ell)}$ (the vertex job corresponding to the same color as v) is $k + 2$, the deadline of j $j_v^{(\ell)}$ is $(k + 2)\ \pi(v) + 1$, and the weight of $j_v^{(\ell)}$ is one.

The processing time of $j_v^{(\ell)}$ with $\ell \neq \ell'$ (vertex jobs corresponding to a different color than v) is one, the deadline of $j_v^{(\ell)}$ with $\ell \neq \ell'$ is $(k + 2)\ \pi(v) - \ell'$, and the weight of $j_v^{(\ell')}$ with $\ell' \neq \ell$ is c_1.

• For each edge $e = \{v, w\} \in E$ with $v \in V_\ell$, $w \in V_{\ell'}$, and $\ell < \ell'$, we create one *edge job* j_e with processing time $(k + 2)(\pi(w) - \pi(v)) - \ell + \ell'$, deadline $(k + 2)\ \pi(w) - \ell - 1$, and weight $c_2(\pi(w) - \pi(v)) + c_3$.

• For each color combination ℓ, ℓ' with $\ell < \ell'$ we create $|V_\ell| + |V_{\ell'}|$ *color combination jobs*, one for each $v \in V\ell$ and one for each $w \in V_{\ell'}$.

Let $v \in V_\ell$, we create a job $j_v^{(\ell,\ell')}$ with processing time $(k + 2)\ \pi(v) - \ell' - 2$, deadline $(k + 2)\ \pi(v) - \ell' - 1$, and weight $c_2 \pi(v)$.

Let $w \in V_{\ell'}$, we create a job $j_w^{(\ell,\ell')}$ with processing time $(k + 2)(n_G - \pi(w)) + \ell + 2$, deadline $(k + 2)n_G + 2$, and weight $c_2(n_G - \pi(w))$.

We create $m = \binom{k}{2}+1$ machines $i_1, i_2, \ldots, i_{\binom{k}{2}+1}$. We call the first $\binom{k}{2}$ machines *edge selection machines* (one machine for each color combination) and we call the remaining machine *validation machine*.

Consider color combination ℓ, ℓ' with $\ell < \ell'$ and let i be the corresponding edge selection machine.

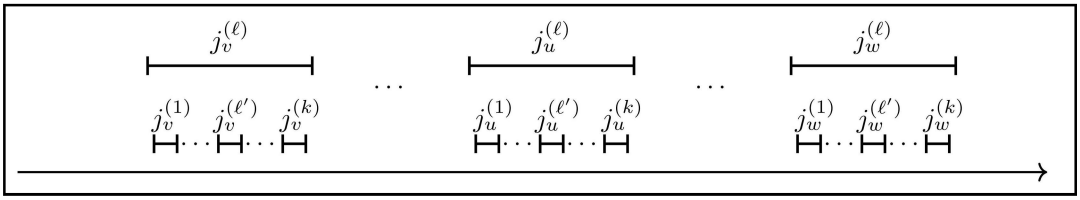

Figure 2. Illustration of the validation machine. Depicted are intervals of jobs corresponding to vertices $v, u, w \in V_\ell$ with $v <_\pi u <_\pi w$

- For each vertex $v \in V_\ell$ we add machine i to the set of eligible machines of job $j_v^{(\ell')}$ and of job $j_v^{(\ell,\ell')}$.
- For each vertex $w \in V - \ell'$ we add machine i to the set of eligible machines of job $j_w^{(\ell)}$ and of job $j_w^{(\ell,\ell')}$.
- For each edge $e = \{v, w\} \in E$ with $v \in V_\ell$ and $w \in V_{\ell'}$ we add machine i to the set of eligible machines of job j_e.

We give an illustration of the edge selection machines in Figure 1. Finally, consider the validation machine $i_{\binom{k}{2}+1}$. We add the validation machine to the set of eligible machines of all vertex jobs. We give an illustration of the validation machine in Figure 2.

This finishes our construction of the Interval Scheduling on Eligible Machines instance. We first show that given a clique of size k in G, we can create a feasible schedule for the constructed instance such that the total weight of scheduled jobs attains at least a certain value.

$$W \geq \binom{k}{2} c_3 + \binom{k}{2} n_G c_2 + (k-1) n_G c_1 + k$$

Proof. Let $G = (V_1 \uplus V_2 \uplus \ldots \uplus V_k, E)$ be an instance of Multicolored Clique and consider the corresponding Interval Scheduling on Eligible Machines instance specified by Construction 1. Assume there is a clique X of size k in G. Then we schedule the following jobs.

For color combination ℓ, ℓ' with $\ell < \ell'$ let $\{v\} = X \cap V_\ell$ and $\{w\} = X \cap V_{\ell'}$. Since X is a clique in G, we know that $e = \{v, w\} \in E$. On edge selection machine i corresponding to color combination ℓ, ℓ' we schedule the following jobs: j_e, $j_v^{(\ell,\ell')}$, $j_w^{(\ell,\ell')}$, $j_v^{(\ell,\ell')}$, and $j_w^{(\ell,\ell')}$. By construction of the instance, the intervals of the jobs are non-intersecting on the machine i, and machines i is in the eligible set of the four jobs. Hence, scheduling these jobs on machine i yields a feasible schedule. Furthermore, it accounts for weight $c_3 + n_G c_2 + 2c_1$ of scheduled jobs per color combination.

Summing over all color combinations, we obtain weight $\binom{k}{2} c_3 + \binom{k}{2} n_G c_2 + k(k-1) c_1$.

Note that for each $\{v\} = X \cap V_\ell$ we have scheduled all vertex jobs $j_v^{(\ell')}$ with $\ell \neq \ell'$.

On the validation machine $i_{\binom{k}{2}+1}$ we schedule the following jobs. Let $\{v\} = X) \cap V_\ell$, then we schedule vertex job $j_v^{(\ell)}$. Note that this job is only in conflict with vertex jobs $j_v^{(\ell')}$ with $\ell \neq \ell'$, which are scheduled on the edge selection machines. Furthermore, we schedule all jobs $j_w^{(\ell')}$ with $v \neq w \in V_\ell$ and $\ell \neq \ell'$. By construction, all these jobs can be scheduled on machine $i\binom{k}{2}+1$ without conflicts and all these jobs have machine $i\binom{k}{2}+1$ in their set of eligible machines.

For all colors, this accounts for weight $(n_G - k)(k-1)c_1 + k$ of scheduled jobs. Clearly, we have that the constructed schedule is feasible. Furthermore, it is straight forward to check that the total weight of scheduled jobs in this constructed schedule is W.

Before we show a similar statement for the opposite direction, we make an observation about feasible schedules in Interval Scheduling on Eligible Machines instances from Construction 1. We show that we can assume that any feasible schedule where the total weight of scheduled jobs is at least $\binom{k}{2}c_3 + \binom{k}{2}n_G c_2 + (k-1)n_G c_1 + k$ schedules exactly one edge job on each edge selection machine.

Observation 5. *Let I be an instance of Interval Scheduling on Eligible Machines resulting from applying Construction 1 to some k-partite graph G. Let $\tilde{A}$ be a feasible schedule such that for the total weight W of scheduled jobs we have*

$$W \geq \binom{k}{2}c_3 + \binom{k}{2}n_G c_2 + (k-1)n_G c_1 + k$$

Then exactly $\binom{k}{2}$ edge jobs are scheduled, one on each edge selection machine.

Proof. We first show that no feasible schedule with total weight $W \geq \binom{k}{2}c_3 + \binom{k}{2}n_G c_2 + (k-1)n_G c_1 + k$ of scheduled jobs can schedule more than $\binom{k}{2}$ edge jobs. Let $\ell, -\ell'$ with $\ell < \ell'$ be a color combination. On the edge selection machine corresponding to color combination ℓ, ℓ' we have that all edge jobs corresponding to edges that do *not* connect vertices of colors ℓ and ℓ' are not eligible. Furthermore, all edge jobs corresponding to edges that connect vertices of colors ℓ and ℓ' are pairwise in conflict. It follows that on each edge selection machine, at most one edge job can be scheduled. Hence, any feasible schedule can schedule at most $\binom{k}{2}$ edge jobs, one on each edge selection machine.

We next show that any feasible schedule with $W \geq \binom{k}{2}c_3 + \binom{k}{2}n_G c_2 + (k-1)n_G c_1 + k$ needs to schedule at least $\binom{k}{2}$ edge jobs. Assume we have a feasible schedule that schedules(strictly) less than $\binom{k}{2}$ edge jobs. Note that each edge job has weight at least c_3. Furthermore,there are at most $k_{nG} + k^2_{nG}$ jobs that are not edge jobs and those jobs each have weightat most $n_G c_2$. Let σ be a feasible schedule that does not schedule all edge jobs and let W be the total weight of all jobs scheduled by . Let W^* denote the sum of weights of all jobs that are not edge jobs. Then we have

$$W < (\binom{k}{2} - 1)c_3 + W^\star < \binom{k}{2}c_3$$

Hence, the observation follows.

Now we are ready to show how to construct a clique of size k from a feasible schedule where the total weight of W scheduled jobs is at least $\binom{k}{2} c_3 + \binom{k}{2} n_G c_2 + (k\text{-}1) n_G c_1 + k$.

Lemma 6. *Let G be an instance of Multicolored Clique. Let I be the Interval Scheduling on Eligible Machines instance computed from G as specified by Construction 1. If there is a feasible schedule* σ *for I such that for the total weight W of scheduled jobs we have*

$$W \geq \binom{k}{2} c_3 + \binom{k}{2} n_G c_2 + (k-1) n_G c_1 + k,$$

then G contains a clique of size k.

Proof. Let $G = (V1 \uplus V2 \uplus \ldots \uplus V_k, E)$ be an instance of Multicolored Clique and consider the corresponding Interval Scheduling on Eligible Machines instance specified by Construction 1. Assume we have a feasible schedule σ such that the for the total weight W of scheduled jobs we have $W \geq \binom{k}{2} c_3 + \binom{k}{2} n_G c_2 + (k\text{-}1) n_G c_1 + k$ We construct a clique of size k in G as follows.

By Observation 5 we know that σ schedules one edge job on each edge selection machine. We can also observe that on the validation machine, only vertex jobs are eligible and can be scheduled. Note that the sum of weights of all vertex jobs is $(k\text{-}1) n_G c_1 + n_G$, which is strictly smaller than c_2.

Assume that edge job j_e is scheduled on the edge selection machine i corresponding to color combination ℓ, ℓ' with $\ell < \ell'$. Then by construction, $e = \{v, w\} \in E$ with $v \in V_\ell$ and $w \in V_{\ell'}$. Now by construction of the instance, two color combination jobs can be scheduled on machine i, one for a vertex of color ℓ and one for a vertex of color ℓ'. In order to obtain a weight of scheduled jobs of at least $c3 + n_G c_2$ it is necessary that jobs $j_v^{(\ell, \ell')}$ and $j_w^{(\ell, \ell')}$ arescheduled (note that the weights of j_e, $j_v^{(\ell, \ell')}$, and $j_w^{(\ell, \ell')}$ sum up to exactly $c_3 + n_G c_2$). Any other selection of color combination jobs to schedule either results in a weight that is lower by at least $c2$ or in an infeasible schedule. Now, by construction, the only further jobs that can be scheduled are $j_v^{(\ell, \ell')}$ and $j_w^{(\ell, \ell')}$. It follows that the maximum weight achievable on any edge selection machine is $c_3 + n_g c_2 + 2_{c1}$. Since $\binom{k}{2}$ 2 $c_1 < c_2$, it follows that for each edge selection machine corresponding to color combination ℓ, ℓ' with $\ell < \ell'$ we have the following: one edge job j_e for $e = \{v, w\}$ with $v \in V_\ell$ and $w \in V_{\ell'}$ is scheduled and the two color combination jobs $j_v^{(\ell, \ell')}$ and $j_w^{\ell, \ell'}$ are scheduled.We can conclude that the jobs scheduled on all edge selection machines have a total weight of at least $\binom{k}{2} c_3 + \binom{k}{2} n_G c_2$. Hence, there are additional jobs scheduled that have a total weight of $(k\text{-}1) n_G c_1 + k$ on the validation machine.

Since no additional edge jobs or color combination jobs can be scheduled, we have that all $(k\text{-}1) n_G$ vertex jobs $j_v(\ell')$ with $v \in V_\ell$ and $\ell = \ell'$(having weight $c_1 > n_G$) are scheduled. Furthermore, at least k vertex jobs $j_v \ell, \ell'$with $v \in V_\ell$(having weight one) are scheduled.

Let X be the set of vertices in G such that if $v \in X$ and $v \in V_\ell$,the job $j_v^{(\ell,)}$ is scheduled. We claim that X is a clique of size at least k in G.

By construction we have that $|X| \geq k$, assume for contradiction that X is not a cliquein G. Then there are two vertices $v, w \in X$ such that $e = \{v, w\} \notin E$. Let $v \in V_\ell$ and $w \in V_{\ell'}$. Then, in particular, vertex jobs $j_v^{(\ell)}$ and $j_w^{(\ell')}$ cannot be scheduled on the validation machine, since they are in conflict with vertex jobs $j_v^{(\ell)}$ and $j_w^{(\ell')}$, respectively. However, as observed above, vertex jobs $j_v^{(\ell')}$ and $j_w^{(\ell)}$ can only be scheduled on the edge selection machine corresponding to color combination ℓ, ℓ' if there is edge job j_e with $e = \{v, w\}$ scheduled on that machine, a contradiction to the assumption that $e = \{v, w\} \notin E$.

Finally, we have all ingredients to prove Theorem 1.

Proof of Theorem 1. To prove Theorem 1, we show that Construction 1 is parameterized polynomial-time reduction from Multicolored Clique parameterized by the number of colors to Interval Scheduling on Eligible Machines parameterized by the number m of machines. First, it is easy to observe that given an instance of Multicolored Clique, the Interval Scheduling on Eligible Machines instance specified by Construction 1 can be computed in polynomial time. Furthermore, if k is the number of colors in the Multicolored Clique instance, then the number of machines in the constructed Interval Scheduling on Eligible Machines instance is $m=\binom{k}{2}+1$. Lastly, observe that all weights in the constructed Interval Scheduling on Eligible Machines instance are in $n_G^{O(1)}$ (where n_G is the number of vertices in the Multicolored Clique instance).

The correctness of the reduction follows from Lemmas 4 and 6. Since Multicolored Clique parameterized by the number of colors is W[1]-hard [10], we have that Theorem 1 follows.

4. NP-Hardness of Unweighted Interval Scheduling on Unrelated Machines

In this section we prove Theorem 3. The containment of Unweighted Interval Scheduling on Unrelated Machines in NP is easy to see, hence we focus on proving NP-hardness. To this end, we present a polynomial-time many-one reduction from Exact (3,4)-SAT to Unweighted Interval Scheduling on Unrelated Machines. In Exact (3, 4)-SAT we are given a Boolean formula ϕ in conjunctive normal form where every clause has exactly three literals and every variable appears in exactly four clauses, and are asked whether ϕ has a satisfying assignment. Exact (3,4)-SAT is known to be NP-hard [29].

Given such a formula ϕ, we construct an instance I of Unweighted Interval Scheduling on Unrelated Machines as follows.

Construction 2. Let ϕ be a Boolean formula in conjunctive normal form where every clause has exactly three literals and every variable appears in exactly four clauses. Let α be the number of variables in ϕ and let β be the number of clauses in ϕ. We construct an instance I of Unweighted Interval Scheduling on Unrelated Machines as follows.

We first describe the jobs, then we define an ordering of the jobs and use it to specify their deadlines. Lastly, we describe the processing times of the jobs on the different machines. We create the following jobs.

- For every variable x, we create two *variable jobs*: x^T and x^F.

• For every clause c, we create three *clause jobs*: c_1, c_2, and c_3.

• We create $2\alpha + 2\beta$ *dummy jobs*.

We next define an ordering π of the jobs, which we will use to define the deadlines of the jobs. To this end, we partition the jobs into the following sets.

• Let $T = \{x^T \mid x \text{ is a variable in } \phi\}$.

• Let $F = \{x^F \mid x \text{ is a variable in } \phi\}$.

• Let $P = \{c_\ell \mid \text{the } \ell \text{ th literal of clause } c \text{ of } \phi \text{ is non-negated}\}$.

• Let $N = \{c_\ell \mid \text{the } \ell \text{ th literal of clause } c \text{ of } \phi \text{ is negated}\}$.

• Let D be the set of dummy jobs.

• Now we define π as a total ordering of the jobs such that

$$D \prec N \prec F \prec P \prec T,$$

and the jobs within the sets are ordered in an arbitrary but fixed way. Let $\pi(j)$ denote the ordinal position of job j in π. For each job j, we set

$$d_j = \pi(j).$$

We next describe the machines, more specifically, the processing times of all jobs on each of the machines. We first introduce α *variable selection machines*, one for each variable in ϕ.

Let x be a variable in ϕ, then we introduce a machine where the processing time of variable.

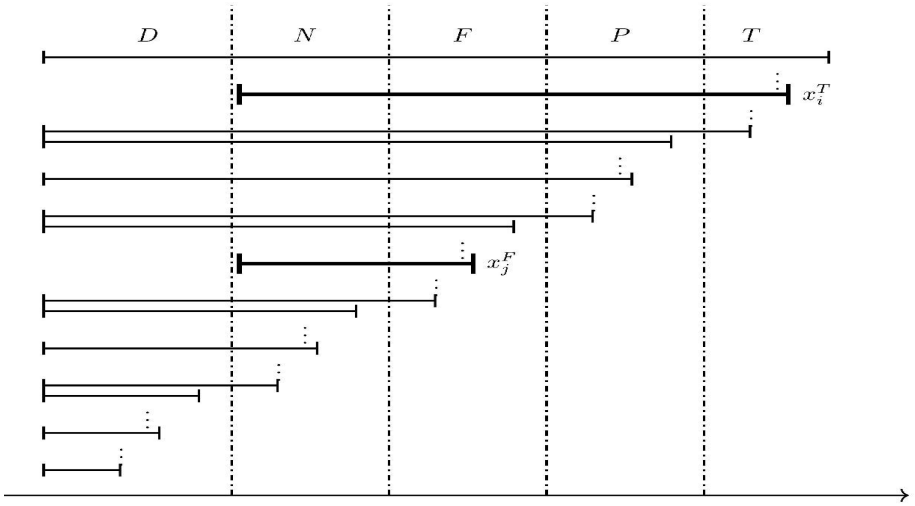

Figure 3. Illustration of the job intervals on the variable selection machine for variable x. On this machine only one of jobs x^T and x^F (bold) can be scheduled alongside with one dummy job

job x^T is $\pi(x^T) - 2\alpha - 2\beta$ and the processing time of variable job x^F is $\pi(x^F) - 2\alpha - 2\beta$. We set the processing times of all other jobs to their respective deadlines. We give an illustration of the variable selection machines in Figure 3.

Next, we introduce 2β *clause selection machines*, two for each clause in ϕ. Let c be a clause in ϕ, then we introduce two machines where the processing times of c_1, c_2, and c_3 are $\pi(c_1) - 2\alpha - 2\beta$, $\pi(c_2) - 2\alpha - 2\beta$ and $\pi(c_3) - 2\alpha - 2\beta$, respectively. We set the processing times of all other jobs to their respective deadlines. We give an illustration of the clause selection machines in Figure 4.

Furthermore, we have α *validation machines*, one for each variable in ϕ. Let x be a variable in ϕ, then we introduce a machine where

- The processing time of x^T is $\alpha(x^T) - \pi(x^F) + 1$,
- The processing time of x^F is $\alpha(x^F) - 2\alpha - 2\beta$,
- If x appears in the ℓth literal of clause c, then the processing time of c_ℓ is one, and
- Processing times of all other jobs are set to their respective deadlines.

We give an illustration of the validation machines in Figure 5.

This finishes our construction of the Unweighted Interval Scheduling on Unrelated Machines instance. We first show that given a satisfying assignment for ϕ, we can

Lemma 7. *Let ϕ be an instance of Exact (3,4)-SAT. Let I be the Unweighted Interval Scheduling on Unrelated Machines instance computed from ϕ as specified by Construction 2. If ϕ is satisfiable, then there is a feasible schedule π for I such that all jobs are scheduled.*

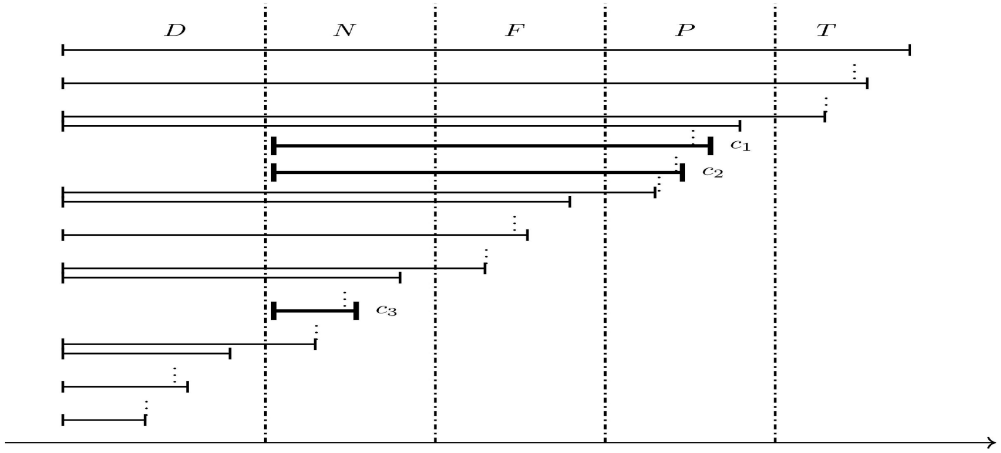

Figure 4. Illustration of the job intervals on the clause selection machine for clause c. On this machine only one of the jobs c_1, c_2 and c_3(bold) can be scheduled alongside with one dummy job

Proof. Let ϕ be an instance of Exact (3, 4)-SAT and consider the corresponding Unweighted Interval Scheduling on Unrelated Machines instance specified by Construction 2. Assume there is a satisfying assignment for ϕ. Then we schedule the jobs as follows.

We first describe on which machine we schedule each variable job. Let x be a variable in ϕ. If x is set to true in the satisfying assignment, we schedule variable job x^T on the variable selection machine corresponding to x and we schedule variable job x^F on the validation machine corresponding to x. Otherwise, we schedule variable job x^F on the variable selection machine corresponding to x and we schedule variable job x^T on the validation machine corresponding to x.

Next, we describe on which machine to schedule each clause job. Let c be a clause in ϕ. Let clause c be satisfied by its ℓ^{th} literal (if multiple literals satisfy the clause, pick one of them arbitrarily). Let x be the variable appearing in the $-$™th literal of c. We schedule clause job c_ℓ on the validation machine corresponding to x. We schedule clause jobs $c_{\ell'}$ with $\ell' \in \{1, 2, 3\} \setminus \{\ell\}$ on the two clause selection machines corresponding to c, respectively.

Lastly, notice that the number of dummy jobs equals the number of machines. For each dummy job we arbitrarily choose a distinct machine and schedule it on this machine.

In the constructed schedule, we clearly schedule each job. We next show that the schedule is feasible.

Notice that on each variable selection machine and each clause selection machine we schedule exactly two jobs, one dummy job and one variable job of the variable corresponding to the variable selection machine or, respectively, one clause job of the clause corresponding to the clause selection machine. Since the processing time intervals of the variable jobs of variables corresponding to the variable selection machines start at

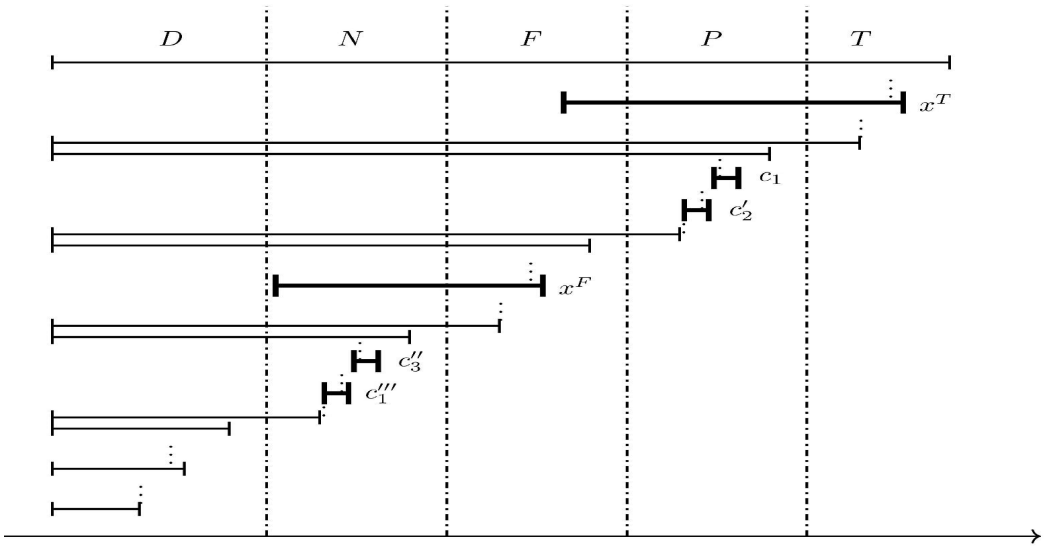

Figure 5. Illustration of the job intervals on the validation machine for variable x for the case that x appears in clauses c and c' non-negated in positions one and two, respectively, and that x appears in clauses c'' and c''' negated in positions three and one, respectively. Processing time intervals of jobs that do not conflict with dummy jobs on this machine are depicted in bold

$2\alpha + 2\beta$ and the deadline of each dummy job is at most $2\alpha+2\beta$, the schedules for the variable selection machines are feasible. Analogously, the schedules for the clause selection machines are feasible.

It remains to show that the schedules for the validation machines are feasible. Notice that by construction, the variable jobs and clause jobs that are potentially scheduled on a validation machine cannot conflict with any dummy job. Furthermore, the variable jobs that are potentially scheduled on a validation machine cannot conflict with each other. We have the same for the clause jobs.

Hence, the only way to obtain an infeasible schedule is if a variable job and a clause job are in conflict. Assume the variable x^T is scheduled (the case of variable job x^F is analogous) and clause job c_ℓ is scheduled and the two jobs are in conflict. Note that this implies that we are dealing with the validation machine for variable x. By construction of the schedule, this means that variable x is set to false in the satisfying assignment. However, the jobs c_ℓ and x^T are in conflict on the validation machine for x (and the job of c_ℓ is not in conflict with the dummy jobs) if x appears non-negated in the ℓ^{th} literal of clause c. Furthermore, by construction of the schedule, we have that clause c is satisfied by its ℓ^{th} literal. This is a contradiction to x being set to false in the satisfying assignment.

Now we show how to construct a satisfying assignment from a feasible schedule where all jobs are scheduled.

Lemma 8. *Let ϕ be an instance of Exact (3,4)-SAT. Let I be the Unweighted Interval Scheduling on Unrelated Machines instance computed from ϕ as specified by Construction 2. If there is a feasible schedule σ for I such that all jobs are scheduled, then ϕ is satisfiable.*

Proof. Let ϕ be an instance of Exact (3,4) - SAT and consider the corresponding Unweighted Interval Scheduling on Unrelated Machines instance specified by Construction 2. Assume we have a feasible schedule for the constructed instance such that all jobs are scheduled. We construct a satisfying assignment for ϕ as follows.

First, observe that by construction, the at most one dummy job can be scheduled on each machine. Since the number of dummy jobs equals the number of machines, we have that on each machine exactly one dummy job is scheduled. This means that on each machine no non-dummy jobs that conflict with a dummy job (that is, jobs with processing time equal to their deadline) can be scheduled.

We can further observe that on the variable selection machine of variable x, apart from a dummy job, only the variable job x^T or the variable job x^F can be scheduled. Since the two jobs conflict, they cannot both be scheduled. We assume w.l.o.g. that exactly one of the two jobs is scheduled. If the variable job x^T is scheduled, we set variable x to true, otherwise we set variable x to false. In the remainder, we show that this yields a satisfying assignment for ϕ.

Assume for contradiction that ϕ is not satisfied by the constructed assignment. Then there is a clause c in ϕ such that none of its literals are satisfied. Consider the three clause jobs c_1, c_2, and c_3 associated with the three literals in clause c. Each of these three jobs can only be scheduled (without creating a conflict with a dummy job) on the clause selection machines corresponding to c, and the validation machine corresponding to the variable appearing in the respective literal of the clause c. Since we only have two clause selection machines, at least one of the clause jobs c_1, c_2, and c_3 has to be scheduled on a validation machine. Assume c_1 is scheduled on a

validation machine (the case of c_2 and c_3 is symmetric). Let x be the variable appearing in the first literal of c. Assume the variable job x^T is scheduled on the variable selection machine corresponding to x (the case where the variable job x^F is scheduled is symmetric). Then the variable job x^F has to be scheduled on the validation machine corresponding to x, since on all other machines it is in conflict with all dummy jobs. However, by construction of the validation machines, the clause job $c1$ and the variable job x^F can only both be scheduled on the validation machine corresponding to x if setting x to true satisfies the first literal of c, a contradiction to the assumption that c is not satisfied.

Finally, we have all ingredients to prove Theorem 3.

Proof of Theorem 3. To prove Theorem 3, we show that Construction 2 is polynomial-time many-one reduction from Exact (3, 4)-SAT to Unweighted Interval Scheduling on Unrelated Machines. First, it is easy to observe that given an instance of Exact (3, 4)-SAT, the Unweighted Interval Scheduling on Unrelated Machines instance specified by Construction 2 can be computed in polynomial time. The correctness of the reduction follows from Lemmas 7 and 8. Since Exact (3, 4)-SAT is NP-hard [29], we have that Theorem 3 follows.

5. Conclusion

We proved that Interval Scheduling on Eligible Machines and its generalization Interval Scheduling on Unrelated Machines are W[1]-hard when parameterized by the number m of machines, and that Unweighted Interval Scheduling on Unrelated Machines is NP-complete, answering two open questions by Mnich and van Bevern [22] and Sung and Vlach [28], respectively. With this, we contribute to the understanding of the (parameterized) computational complexity of basic and natural interval scheduling problems.

Our results leave two main open questions. Our NP-hardness proof for Unweighted Interval Scheduling on Unrelated Machines does not imply NP-hardness of Unweighted Interval Scheduling on Eligible Machines, which leaves the following question.

References

[1] Angelelli, Enrico., Bianchessi, Nicola., Filippi, Carlo. (2014). Optimal interval scheduling with a resource constraint. *Computers & Operations Research*, 51, 268–281.

[2] Arkin, Esther M., Silverberg, Ellen B. (1987). Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18(1), 1–8.

[3] Baker, Kenneth R., Scudder, Gary D. (1990). Sequencing with earliness and tardiness penalties: A review. *Operations Research*, 38(1), 22–36.

[4] Bentert, Matthias., van Bevern, René., Niedermeier, Rolf. (2019). Inductive k-independent graphs and c-colorable subgraphs in scheduling: A review. *Journal of Scheduling*, 22(1), 3–20.

[5] van Bevern, René., Mnich, Matthias., Niedermeier, Rolf., Weller, Mathias. (2015). Interval scheduling and colorful independent sets. *Journal of Scheduling*, 18(5), 449–469.

[6] van Bevern, René., Niedermeier, Rolf., Suchý, Ondøej. (2017). A parameterized complexity view on non-preemptively scheduling interval-constrained jobs: Few machines, small looseness, and small slack. *Journal of Scheduling*, 20(3), 255–265.

[7] Bouzina, Khalid I., Emmons, Hamilton. (1996). Interval scheduling on identical machines. *Journal of Global Optimization*, 9(3), 379–393.

[8] Carlisle, Martin C., Lloyd, Errol L. (1995). On the k-coloring of intervals. *Discrete Applied Mathematics*, 59(3), 225–235.

[9] Èepek, Ondøej., Sung, Shao Chin. (2005). A quadratic time algorithm to maximize the number of just-in-time jobs on identical parallel machines. *Computers & Operations Research*, 32(12), 3265–3271.

[10] Fellows, Michael R., Hermelin, Danny., Rosamond, Frances., Vialette, Stéphane. (2009). On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1), 53–61.

[11] Gavril, Fanica. (1974). The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1), 47–56.

[12] Graham, Ronald L. (1969). Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2), 416–429.

[13] Hiraishi, Kunihiko., Levner, Eugene., Vlach, Milan. (2002). Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs. *Computers & Operations Research*, 29(7), 841–848.

[14] Impagliazzo, Russell., Paturi, Ramamohan. (2001). On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2), 367–375.

[15] Impagliazzo, Russell., Paturi, Ramamohan., Zane, Francis. (2001). Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4), 512–530.

[16] Kolen, Antoon W.J., Lenstra, Jan Karel., Papadimitriou, Christos H., Spieksma, Frits C.R. (2007). Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5), 530–543.

[17] Kovalyov, Mikhail Y., Ng, Chi To., Cheng, T.C. Edwin. (2007). Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research*, 178(2), 331–342.

[18] Krafcik, John F. (1988). Triumph of the lean production system. *Sloan Management Review*, 30(1), 41–52.

[19] Lann, Avital., Mosheiov, Gur. (1996). Single machine scheduling to minimize the number of early and tardy jobs. *Computers & Operations Research*, 23(8), 769–781.

[20] Leyvand, Yaron., Shabtay, Dvir., Steiner, George., Yedidsion, Liron. (2010). Just-in-time scheduling

with controllable processing times on parallel machines. *Journal of Combinatorial Optimization*, 19(3), 347–368.

[21] Lokshtanov, Daniel., Marx, Dániel., Saurabh, Saket. (2013). Lower bounds based on the exponential time hypothesis. *Bulletin of EATCS*, 3(105).

[22] Mnich, Matthias., van Bevern, René. (2018). Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research*, 100, 254–261.

[23] Ohno, Taiichi., Bodek, Norman. (1988). *Toyota production system: Beyond large-scale production*. Productivity Press.

[24] Rose, Donald J., Tarjan, Robert Endre., Lueker, George S. (1976). Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2), 266–283.

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

[26] Shingo, Shigeo., Dillon, Andrew P. (1985). *A revolution in manufacturing: The SMED system*. Productivity Press.

[27] Spieksma, Frits C.R. (1999). On the approximability of an interval scheduling problem. *Journal of Scheduling*, 2(5), 215–227.

[28] Sung, Shao Chin., Vlach, Milan. (2005). Maximizing weighted number of just-in-time jobs on unrelated parallel machines. *Journal of Scheduling*, 8(5), 453–460.

[29] Tovey, Craig A. (1984). A simplified NP-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1), 85–89.

[30] Womack, James P., Jones, Daniel T. (1997). Lean thinking – Banish waste and create wealth in your corporation. *Journal of the Operational Research Society*, 48(11), 1148–1148.

[31] Womack, James P., Jones, Daniel T., Roos, Daniel. (1990). *The machine that changed the world: The story of lean production – Toyota's secret weapon in the global car wars that is now revolutionizing world industry*. Simon and Schuster.

[32] Yannakakis, Mihalis., Gavril, Fanica. (1987). The maximum k-colorable subgraph problem for chordal graphs. *Information Processing Letters*, 24(2), 133–137.