Kernelization Algorithms on Single Machine Scheduling with Time Windows

Maher Mallem[1,*], Claire Hanen, Alix Munier-Kordon

[a] *Inria, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668, Lyon cedex 07, 69342, France*
[b] *Université Paris Nanterre, Nanterre, 92000, France*
[c] *Sorbonne Université, CNRS, LIP6, Paris, 75005, France*
[d] *Sorbonne Université, CNRS, LIP6, Paris, 75005, France*

Abstract

Over the past few years parameterized complexity has successfully been considered on scheduling problems, with many fixed-parameter tractable algorithms established on a variety of parameters. Knowing this, it is standard to look for kernels and associated kernelization algorithms in order to preprocess scheduling instances and improve the runtime of parameterized algorithms. However, multiple parameterized scheduling problems still lack kernel characterizations, especially when the number of job types is not bounded by the parameter in any way. In this paper, we study the kernelization of a single machine scheduling problem with precedence constraints and job time windows with respect to three graph parameters: the proper level, the pathwidth and the vertex cover of the compatibility graph induced by the job time window overlaps. While this problem is known to be fixed-parameter tractable with respect to the pathwidth and the proper level, we show that a polynomial kernel is unlikely to be built with these parameters. Nevertheless we provide a polynomial kernel with respect to the vertex cover. Based on this polynomial kernel, we propose kernels of exponential size with respect to the pathwidth and the proper level. This paves the way to kernel ideas for other scheduling problems with known fixed-parameter tractable algorithms in the hopes of improving their runtime in practice.

[*] Corresponding author.
Email addresses: maher.mallem@ens-lyon.fr (Maher Mallem), claire.hanen@lip6.fr (Claire Hanen), alix.munier@lip6.fr (Alix Munier-Kordon)

[1] Part of this research project was conducted when this co-author was a PhD student at LIP6, Sorbonne Université.

Keywords: scheduling, kernel, single machine, fixed-parameter tractability, vertex cover

1. Introduction

Parameterized complexity of scheduling problems has drawn increasing attention in the last ten years [MvB18]. Several fixed-parameter algorithms have been proposed, albeit the most common outcome is hardness at some level in the W-hierarchy, sometimes even above it. In this paper we consider a single machine scheduling problem with job time windows and precedence constraints. This decision problem is denoted by $1|prec, r_j, d_j|\star$ in Graham's notation. It has been proven NP-hard in the strong sense in the 70's [LRKB77], even without precedence constraints.

Single machine scheduling has been studied from a parameterized point of view by several authors with two main tools. Particular structures of linear programs were considered to design fixed-parameter algorithms for scheduling problems where the parameter is closely related to the number of job types [MW15, KKL$^+$19, HKPS21, HMO24]. Within a type of jobs, all jobs share the same properties. In the problem studied in this paper, they would have the same processing time, time window and precedence neighborhood. Several other parameters have been investigated, this time using dynamic programming to obtain fixed-parameter algorithms. For single machine problems, let us mention the pathwidth of the interval graph induced by the time windows [HMMK22], the proper level of the interval graph [MHMK24], or the width of the precedence graph combined with the maximum allowed lag [vBBB$^+$16]. Unfortunately these parameters are not sufficient to derive fixed-parameter algorithms in the presence of precedence delays [BvdW20, HMMK22] or parallel processors [HMK23]. The maximum processing time of a job also leads to a negative parameterized result since the single machine problem is NP-complete with two different fixed values of processing times [EdW14].

A kernelization is a polynomial-time pre-processing algorithm that either solves the problem or reduces an instance I to an equivalent instance I' (the kernel) whose size $g(k(I))$ only depend on the parameter $k(I)$. Now, it is well known that whenever a problem $\mathcal{Q}$ is fixed-parameter tractable with respect to some parameter k, there is a general way to infer a kernelization with respect to k (see e.g. Theorem 1.39, page 29-30 in the book by [FG06]). However, having a custom kernelization generally gives more information on the shape of the set of kernels . One of the challenging questions is the

existence of a polynomial kernel, where the size of the kernel is a polynomial of $k(I)$.

While such kernelizations have already been proposed in the literature for several scheduling problems [BJ22, KK22, JKZ25], they were done in the context of high-multiplicity scheduling with parameters restricting the number of job types. Considering other structural parameters, we need to reduce the instance size - i.e. both the number of jobs and the data values - to a function of the parameter. To our knowledge, no such kernelization has been produced in the literature.

In this paper we aim at finding the minimal parameters for which a polynomial kernel exists on single machine scheduling with precedence relations and job time windows. To this end, we focus on five parameters related to the interval graph induced by the time windows: its pathwidth denoted by pw, its proper level denoted by pl (i.e. the maximum number of time windows which can strictly include a time window on both ends), its vertex integrity vi, its bandwidth bw and its vertex cover number denoted by vc. In Section 2 we give base definitions and relate our parameters. From these relations we derive that our problem is fixed-parameter tractable with respect to all five considered parameters. In Section 3 we show that the problem cannot have a polynomial kernel for all these parameters but vc under usual parameterized complexity assumptions. Then in Section 4 we provide a polynomial kernel with respect to parameter vc. In Section 5 we revisit this polynomial kernel to make it work with a parameter which allows for vertex covers of unbounded size. Finally we summarize our kernelization results in the form of a parameter map, and we propose several directions to expand the parameterized analysis performed in this work.

2. Preliminaries

In this section we define accurately the decision problem and present the parameters that we use. We also sketch some related results.

2.1. Problem definition

Definition 1. *An* **instance** $I = \langle prec, p_j, r_j, d_j \rangle$ **of decision problem** $1|prec, r_j, d_j|\star$ *is defined by:*

1. *a set of n jobs $\mathcal{J}$. Each job i is characterized by a processing time p_i and a time window delimited by a release time r_i and a deadline d_i,*
2. *a set of precedence constraints in the form of a directed acyclic graph $prec = (\mathcal{J}, A)$. An arc (i, j) in A implies that job j cannot start before the end of job i. This relation is also denoted by $i \to j$.*

Definition 2. *A* **schedule** *of an instance defines a starting time* $s_i \in [r_i, d_i - p_i]$ *for each job* i *in* $\mathcal{J}$*. A schedule is called* **feasible** *when at most one job is processed at each time - i.e. for all jobs* $i, j, i \neq j$ *either* $s_j \geq s_i + p_i$ *or* $s_i \geq s_j + p_j$ *- and all precedence constraints are met - i.e. for every arc* $(i, j) \in A$*, we have* $s_i + p_i \leq s_j$*.*

In **decision problem** $1|prec, r_j, d_j|\star$ *we check whether there exists a feasible schedule for a given instance. The* **makespan** *of a schedule is the last completion time of a job* $C_{\max} = \max_{i \in \mathcal{J}} s_i + p_i$*. When due dates* $d'_i \leq d_i$ *are given for each job* $i \in \mathcal{J}$ *in the problem instance, a usual objective function is the* **maximum lateness** $L_{\max} = \max_{i \in \mathcal{J}} s_i + p_i - d'_i$*.*

Without loss of generality we assume that the precedence constraints are consistent with the time windows, i.e. if $(i, j) \in A$ then $r_i + p_i \leq r_j$; $d_i \leq d_j - p_j$ and $r_j < d_i$. Such instances will be called *prec*-**consistent**.

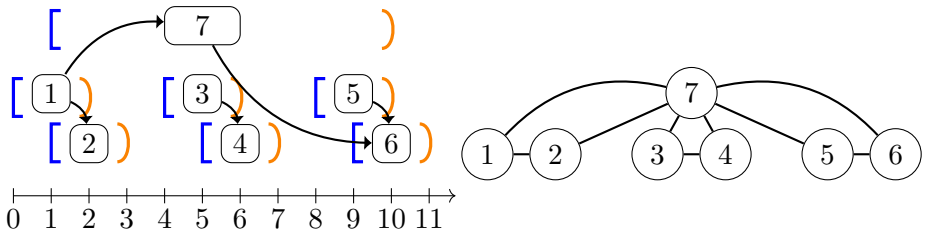
Definition 3.

- *A parameterization* $(\mathcal{Q}, k)$ *of a decision problem* $\mathcal{Q}$ *associates to each instance* I *of* $\mathcal{Q}$ *a nonnegative integer* $k(I)$*.*
- *Given a parameterized decision problem* $(\mathcal{Q}, k)$*, a* **fixed-parameter algorithm** *solves any instance* $(I, k(I))$ *in* $f(k(I)) \cdot |I|^{\mathcal{O}(1)}$ *time, where* f *is a computable function. Problem* $\mathcal{Q}$ *is fixed-parameter tractable with respect to* k *if there exists such an algorithm.*
- *A* **kernelization** *is a polynomial-time algorithm which outputs an instance* I' *of* $\mathcal{Q}$ *with parameter value* $k(I')$ *such that* I' *is equivalent to* I *and the size* $(|I'| + k(I'))$ *of the parameterized instance is bounded by* $g(k(I))$ *for some computable function* g*.*
- *Such an instance* I' *is called a* **kernel***. It is a* **polynomial kernel** *if* g *is a polynomial function.*

Notice that, by using binary search on the objective value, a fixed-parameter algorithm for problem $1|prec, r_j, d_j|\star$ with respect to some parameter provides a fixed-parameter algorithm for the optimization versions of the problem minimizing the makespan or the maximum lateness with respect to the same parameter.

2.2. Parameters based on the compatibility graph

Several parameters have been considered for single machine scheduling problems. In this paper we focus on parameters built from the compatibility graph induced by the jobs time windows.



(a) Time windows and precedence constraints

(b) Compatibility graph G_I

Figure 1: A scheduling instance and its compatibility graph

Definition 4. *The* **compatibility graph** $G_I = (\mathcal{J}, E)$ *of a problem instance is defined as follows. Its set of nodes is the set of jobs. An undirected edge* $\{i, j\}$ *is in* E *if and only if the time windows of jobs* i *and* j *overlap, i.e.* $[r_i, d_i) \cap [r_j, d_j) \neq \emptyset$.

Figure 1 shows an example and its compatibility graph. We define the following structural parameters:

Definition 5. *Let us consider the compatibility graph* G_I *of our instance.*

- *Note that in interval graphs the pathwidth corresponds to the maximum clique size minus one [BM93]. In graph* G_I *a clique corresponds to a set of overlapping time windows. So, the* **pathwidth** *pw can be defined as the maximum number of time windows overlapping the release time of a job :*

$$pw = \max_{i \in \mathcal{J}}(|\{j \in \mathcal{J}, j \neq i \wedge r_j \leq r_i < d_j\}|).$$

- *The* **proper level** *pl is defined as the maximum number of time windows* $[r_j, d_j)$ *which can strictly include a time window* $[r_i, d_i)$ *on both ends. In other words:*

$$pl = \max_{i \in \mathcal{J}}(|\{j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}|).$$

- *The* **vertex cover number** *vc is the minimum size of a subset of jobs* $\mathcal{VC} \subseteq \mathcal{J}$ *such that any edge of the compatibility graph has at least one of its nodes in* $\mathcal{VC}$.

- *The* **vertex integrity** *vi is defined as follows according to [BES87, GHK+22]. If* $S \subseteq \mathcal{J}$ *is a subset of nodes, called a separator, we denote*

by $cc(G_I - S)$ the set of connected components of the graph when nodes of S are removed. We then have:

$$vi = \min_{S \subseteq \mathcal{J}}(|S| + \max_{D \in cc(G_I - S)} |D|).$$

- *The* **bandwidth** *bw of the graph is defined as follows. If we consider a linear arrangement $f : J \to \{1, \ldots, |\mathcal{J}|\}$ of the jobs, for each edge $\{i, j\}$ we measure the distance between assigned labels $|f(j) - f(i)|$ and compute the maximum distance over the edges. The bandwidth bw is the minimum value of this maximum distance over all linear arrangements.*

To illustrate these definitions, we compute the parameters on the compatibility graph of Figure 1b. The release time of job 2 (resp. $4, 6$) is included

pw	*pl*	*vc*	*vi*	*bw*
2	1	4	3	3

Table 1: Parameter values of the example given in Figure 1

in the time window of jobs 1 (resp. $3, 5$) and 7. So, we have $pw = 2$. The proper level pl is equal to 1 since only job 7 has a lower release time and a higher deadline than other jobs - namely jobs 3 and 4. To cover all the edges we need to choose at least one node among each pair $\{1, 2\}, \{3, 4\}, \{5, 6\}$ plus node 7, so $vc = 4$. The vertex integrity can be computed using the separator $\{7\}$ which leaves three connected components of size 2 each. Hence the vertex integrity is $vi = 3$. Finally, the bandwidth of this example is obtained by giving job 7 an index in the middle of the linear arrangement $f(7) = 4$ since it is linked to all other jobs. The distance is then $bw = 3$.

As with other common graph parameters it is NP-complete to compute the pathwidth and the vertex cover number for general graphs [Kas79]. However, the underlying interval structure of the compatibility graph eases the computation of these parameters. Pathwidth pw can be computed in $\mathcal{O}(n \cdot \log(n))$ time by sorting the release times and deadlines and then by going over them in order while updating the number of overlapping time windows accordingly. Proper level can be computed with a straightforward algorithm in $\mathcal{O}(n^2)$ time. Furthermore, a vertex cover of size vc can be computed in linear time when restricted to interval graphs [MRR92]. So parameter vc can be computed in linear time. Similarly [KKM97] showed that vi could be computed in $\mathcal{O}(n^3)$ time on interval graphs.

2.3. Related results

[BG19] considered the vertex cover number in the context of coupled-task scheduling with a compatibility graph $G_c = (\mathcal{J}, E)$ which, unlike graph G_I, was not necessarily an interval graph. In G_c there was an edge between two coupled tasks in G_c if and only if they could be interleaved. Via a kernelization, they showed that computing the minimum makespan is fixed-parameter tractable parameterized by $vc(G_c)$ plus the maximum length of a coupled task.

We recall the fixed-parameter tractability results in the literature with respect to parameters pathwidth pw and proper level pl.

Lemma 6. *[HMMK22, MHMK24]* $1|prec, r_j, d_j|\star$ *has a fixed-parameter algorithm with respect to pathwidth* pw *(resp. proper level* pl*) running in time* $\mathcal{O}((pw+1)^2 4^{pw} \cdot n + n \cdot \log(n))$ *(resp. time* $\mathcal{O}((pl+1)^2 4^{pl} \cdot n + n \cdot \log(n))$*).*

From the literature we also know that the parameters are linked as shown below:

$$pl \leq pw \leq vi - 1 \leq vc \leq n - 1. \tag{1}$$

The relation between pl and pw was established by [MHMK24]. The relation between vi, pw and vc was established by [GHK$^+$22].

Notice that bandwidth bw and vc are unrelated but we know from [KS96] that:

$$pw \leq bw \leq n - 1. \tag{2}$$

This means that any problem in FPT with respect to parameter pw is also in FPT with respect to vc, bw and vi. We can thus derive the following result from [HMMK22].

Corollary 7. $1|prec, r_j, d_j|\star$ *is fixed-parameter tractable parameterized by either* vc*,* bw *or* vi*.*

So we know that kernelizations can be defined for all parameters pl, pw, vi, bw and vc. One of the challenging questions is the existence of a polynomial kernel for these parameters and, in case of a negative answer, to determine the minimal parameter for which a polynomial kernel exists.

3. Non-existence of Polynomial Kernels

Even though fixed-parameter algorithms exist for parameters proper level, pathwidth, vertex integrity and bandwidth, our problem does not admit a polynomial kernel for any of these parameters under usual parameterized complexity assumptions. In this section we prove the following theorem:

Theorem 8. *Unless $NP \subseteq coNP/poly$ there is no polynomial kernel for $1|r_j, d_j|\star$ with respect to any of the four parameters pathwidth pw, proper level pl, vertex integrity vi or bandwidth bw.*

We use the cross-composition technique developed by [BJK14] to show Theorem 8 for vertex integrity vi, pathwidth pw and proper level pl. Intuitively a cross-composition corresponds to reducing a disjunction of instances of a decision problem $\mathcal{D}$ to a single instance of a parameterized problem $(\mathcal{Q}, k)$. The existence of such a reduction when $\mathcal{D}$ is an NP-complete problem implies that there is no polynomial kernel for $(\mathcal{Q}, k)$.

We consider here a restricted version of cross-composition where the equivalence relation mentioned in the original definition by [BJK14] decides whether a string is a valid instance of a decision problem or not. For the sake of consistency, we express the original definition in the parameterized decision problem vocabulary.

Definition 9. *Let $\mathcal{D}$ be a decision problem, and $(\mathcal{Q}, k)$ be a parametrized decision problem. We say that $\mathcal{D}$* **cross-composes** *into $(\mathcal{Q}, k)$ if there is an algorithm (called a* **cross-composition***) which, given t instances $I_1, I_2, \ldots, I_t$ of $\mathcal{D}$ computes an instance $(I^*, k(I^*))$ of $(\mathcal{Q}, k)$ in time polynomial in $(\Sigma_{i=1}^{t}|I_i|)$ such that:*

1. *deciding whether a given I is an instance of $\mathcal{D}$ can be done in time $\mathcal{O}(|I|^{\mathcal{O}(1)})$,*
2. *$(I^*, k(I^*))$ is a feasible instance of $(\mathcal{Q}, k) \iff \exists i \in \{1, \ldots, t\}$, I_i is a feasible instance of $\mathcal{D}$,*
3. *$k(I^*)$ is bounded by a polynomial in $\max_{1 \leq i \leq t}(|I_i| + \log(t))$.*

3.1. Cross-composition for Parameters Vertex Integrity, Pathwidth and Proper level

We consider as decision problem $\mathcal{D}$ the PARTITION problem, which is defined as follows:

Definition 10. *An instance of* PARTITION *is defined by a set A of n integers $a(1), \ldots, a(n)$ and an integer B such that $\sum_{i=1}^{n} a(i) = 2B$. The instance is feasible if there exists a subset $X \subset A$ such that $\sum_{a(i) \in X} a(i) = B$.*

Consider t instances of the PARTITION problem. We denote n_k the number of integers of the instance I_k, by $a_k(i)$ the i^{th} integer of instance I_k, and by B_k its threshold. From these t instances we build an instance $\mathcal{J}(I_1, \ldots, I_t)$ of $1|r_j, d_j|\star$ parameterized by pw - i.e. our single machine problem with an

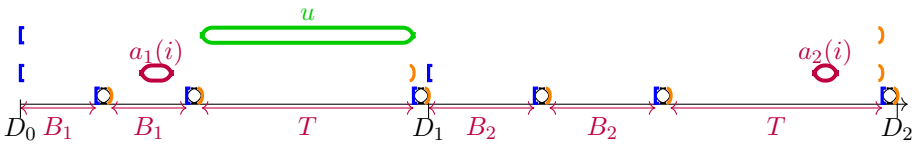


Figure 2: Cross-composition with respect to vertex integrity vi from two instances I_1, I_2 of Partition

empty precedence relation. We first set $T = 2 \cdot (\max_{1 \leq i \leq t} B_i)$, and for all $1 \leq k \leq t$ we set $D_k = (T+1) \cdot k + 2 \cdot (\sum_{i=1}^{k} (B_i + 1))$. We notably have: $D_k - D_{k-1} = 2B_k + T + 3$.

All jobs associated to instance I_k have their time window between D_{k-1} and D_k. First to each Partition instance I_k are associated three unit-time fill jobs $f_{k,i}$ $(1 \leq i \leq 3)$. These jobs have a time window of length one and delimit three zones in interval $[D_{k-1}, D_k)$: two zones of length B_k and one of length T. See Figure 3 for an example with $k = 2$. Below is a more detailed description of the fill job time windows:

	$f_{k,1}$	$f_{k,2}$	$f_{k,3}$
r_i	$D_{k-1} + B_k$	$D_{k-1} + 2B_k + 1$	$D_k - 1$
d_i	$D_{k-1} + B_k + 1$	$D_{k-1} + 2B_k + 2$	D_k

Then for each integer $a_k(i)$ in Partition instance I_k we set a job of processing time $a_k(i)$, release time D_{k-1} and deadline $D_k - 1$. Finally we define a job u, which we call the *selector job*, with processing time T and time window $r_u = 0$, $d_u = D_t$. Due to its processing time it can only be scheduled in the third zone of an interval $[D_{k-1}, D_k)$ for some k. This forces all jobs associated to Partition instance I_k to be scheduled in the two remaining zones of length B_k in this interval.

Lemma 11. Partition *cross-composes into problem* $1|r_j, d_j|\star$ *parameterized by any of the three parameters* pw, pl *or* vi.

Proof. Proof. Let $\mathcal{J}(I_1, \ldots, I_t)$ be the instance of $1|r_j, d_j|\star$ build from t instances of Partition $I_1, \ldots, I_t$ as shown previously. The polynomiality of the construction is straightforward.

By considering the selector job as separator $S = \{u\}$, removing u leads to t connected components of G_I, one for each instance I_k, each with n_k+2 jobs. Hence $vi \leq 3+(\max_{1 \leq k \leq t} n_k)$, which is polynomial in $\max_{1 \leq i \leq t}(|I_i|+\log(t))$. Following Equation 1 we have $pl \leq pw \leq vi$. So this cross-composition holds for proper level pl and pathwidth pw too.

We finally have to verify that the instance $\mathcal{J}(I_1, \ldots, I_t)$ is feasible if and only if one of the PARTITION instance is feasible.

($\Longleftarrow$) Assume that the PARTITION instance I_k is feasible. Let X_k be the subset of values such that $\sum_{a_k(i) \in X_k} a_k(i) = B_k$. We build a feasible schedule by scheduling job u at time $D_k - T - 1$, the tasks of X_k in sequence in the interval $[D_{k-1}, D_{k-1} + B_k)$, and the other tasks $a_k(j)$ are performed in sequence in interval $[D_{k-1} + B_k + 1, D_{k-1} + 2B_k + 1)$. The jobs $a_l(i)$ of the other instance are all performed in any order in the interval $[D_{l-1} + 2B_l + 2, D_l - 1)$. Job u does not interfere with any other job, so the schedule is feasible.

($\Longrightarrow$) Let us observe that in any feasible schedule, there is a k such that job u is performed at time $D_k - 1 - T$ between $f_{k,2}$ and $f_{k,3}$. Indeed, the size of the interval between two other fill jobs is always strictly less than T. Now the other jobs $a_k(i)$ cannot be scheduled in interval $[D_k - 1 - T, D_k)$ that perform u and a fill job. So they can only be executed before $f_{k,1}$ or between $f_{k,1}$ and $f_{k,2}$. If all jobs are executed this defines a partition of this instance: X_k is the set of jobs performed before $f_{k,1}$. □

It is well known that the PARTITION problem is NP-hard (see Section 3.1.5 in the book by [GJ79]). From Theorem 9 of [BJK14] we immediately deduce the validity of Theorem 8 for parameters vi, pw, pl.

3.2. Cross-composition for Parameter Bandwidth

For the bandwidth parameter, in the previous reduction the selector job u is linked to all other jobs in G_I. So its bandwidth is at least $(\sum_{k=1}^{t} n_k + 3)/2$, which does not satisfy the cross-composition condition. To overcome this difficulty, we build a new cross-composition from PARTITION as follows. Consider t instances of the PARTITION problem with the same notations as previously. Again we set $T = 2(\max_{1 \leq i \leq t} B_i)$. And: $D'_0 = 0$; for all $1 \leq k < t$: $D'_k - D'_{k-1} = 2B_k + 2T + 4$; $D'_t - D'_{t-1} = 2B_t + T + 3$.

First, for every $k < t$, four unit-time fill jobs $f_{k,i}$ $(1 \leq i \leq 4)$ are associated to PARTITION instance I_k. These jobs have a time window of length one and delimit four zones in interval $[D'_{k-1}, D'_k)$: two zones of length B_k then two zones of length T. Only three fill jobs are associated to the last instance I_t, delimiting three zones: two of length B_k then one of length T.

Below are described the fill job time windows:

	$f_{k,1}$	$f_{k,2}$	$f_{k,3}$	$f_{k,4}$	$f_{t,1}$	$f_{t,2}$	$f_{t,3}$
r_i	$D'_{k-1} + B_k$	$D'_{k-1} + 2B_k + 1$	$D'_k - T - 2$	$D'_k - 1$	$D'_{t-1} + B_t$	$D'_{t-1} + 2B_t + 1$	$D'_t - 1$
d_i	$D'_{k-1} + B_k + 1$	$D'_{k-1} + 2B_k + 2$	$D'_k - T - 1$	D'_k	$D'_{t-1} + B_t + 1$	$D'_{t-1} + 2B_t + 2$	D'_t

Then for each integer $a_k(i)$ in PARTITION instance I_k we set a job of processing time $a_k(i)$, release time D'_{k-1} and deadline $D'_k - T - 2$. Fi-

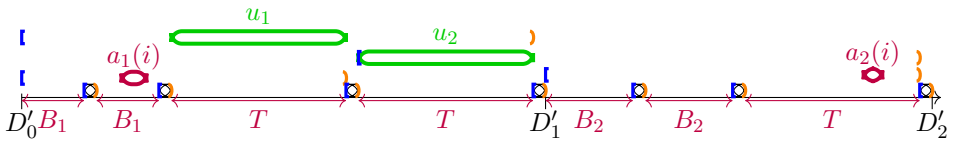


Figure 3: Cross-composition with respect to bandwidth bw from two instances I_1, I_2 of Partition

nally we define for each instance I_k a selector job u_k, with time window $[\max(0, D'_{k-1} - T - 1), D'_k - 1)$. So the time windows of u_{k-1} and u_k overlap in interval $[D'_{k-1} - T - 1, D'_k)$

Due to its processing time a selector job u_k can only be scheduled either in the third or fourth (if it exists) of interval $[D'_{k-1}, D'_k)$ or in the fourth zone of interval $[D'_{k-2}, D'_{k-1})$ (if $k \geq 2$). Like in the previous reduction if a selector is in the third zone of an interval $[D'_{k-1}, D'_k)$, this forces all jobs associated to Partition instance I_k to be scheduled in the two remaining zones of length B_k in this interval.

Now, there are $t - 1$ fourth zones and t selector jobs, so that in any feasible schedule at least one selector job should be scheduled in a third zone. Conversely, if I_k is feasible, then we schedule u_k in the third zone of interval $[D'_{k-1}, D'_k)$ and the corresponding jobs $a_k(i)$ in the two first zones, as they are partitioned. The selector jobs u_l with $k < l \leq t$ are scheduled in their left fourth zone $[D'_{l-1} - T, D'_{l-1})$ whereas selector jobs u_l with $l < k$ are scheduled in their right fourth zone $[D'_l - T, D'_l)$. The jobs $a_l(i)$ are then scheduled in the third zone within $2B_l$ time units. We deduce that Theorem 8 holds for parameter bandwidth bw.

4. A Polynomial Kernel with respect to the Vertex Cover Number

In this section we propose a polynomial kernelization of $1|prec, r_j, d_j|\star$ with respect to parameter vc. We first consider a reduction rule which reduces the number of jobs down to $\mathcal{O}(vc^2)$. Then we reduce the integer values of the instance to a function of vc.

4.1. Reducing the number of jobs

Let I be an instance of $1|prec, r_j, d_j|\star$. Let $\mathcal{VC}$ be a vertex cover of G_I with minimum size vc. Let $(u_\alpha)_{\alpha \geq 0}$ be the increasing sequence of the different release time/deadline values of the jobs from $\mathcal{VC}$. First, note that the jobs outside the vertex cover have limited time window possibilities:

Proposition 12. *Given a vertex cover $\mathcal{VC}$ of graph G_I, the jobs in $\mathcal{J} - \mathcal{VC}$ form a sequence $(w_j)_{1 \leq j \leq |\mathcal{J}-\mathcal{VC}|}$ such that $d_{w_j} \leq r_{w_{j+1}}$ for all $1 \leq j < |\mathcal{J} - \mathcal{VC}|$.*

Proof. Proof. By the definition of $\mathcal{VC}$ all job pairs in $\mathcal{J} - \mathcal{VC}$ have no time window intersection. So we get sequence $(w_j)_{1 \leq j \leq |\mathcal{J}-\mathcal{VC}|}$ by ordering the jobs in $\mathcal{J} - \mathcal{VC}$ by increasing release times. □

Corollary 13. *In any prec-consistent instance of $1|prec, r_j, d_j|\star$, the number of precedence relations is in $\mathcal{O}(vc^2)$.*

Proof. Proof. By *prec*-consistency if there is a precedence relation between jobs i and j, then their time windows must intersect without one strictly including the other on both ends. Thus, given a vertex cover $\mathcal{VC}$ of graph G_I, by Proposition 12 jobs i and j cannot be both in $\mathcal{J} - \mathcal{VC}$, and each job in $\mathcal{VC}$ has at most one ancestor and one descendant in $\mathcal{J} - \mathcal{VC}$. □

Definition 14. *Let w_j, w_{j+1} be two consecutive jobs in $\mathcal{J} - \mathcal{VC}$. Couple (w_j, w_{j+1}) is called a* **gap***. Jobs w_j, w_{j+1} are called the delimiters of this gap. Given a schedule $(s_i)_{i \in \mathcal{J}}$, the* **capacity** *of gap (w_j, w_{j+1}) is defined as $(s_{w_{j+1}} - s_{w_j} - p_{w_j})$. The* **maximum capacity** *of gap (w_j, w_{j+1}) is defined as $(d_{w_{j+1}} - r_{w_j} - p_{w_j} - p_{w_{j+1}})$ and is obtained by left-shifting w_j and right-shifting w_{j+1} as much as possible. A gap g_j is an* **inner gap** *if there is some index α such that $u_\alpha \leq r_{w_j}$ and $d_{w_{j+1}} \leq u_{\alpha+1}$. The other gaps are called* **border gaps**.

By Proposition 12 the existence of a feasible schedule relies on the feasibility of inserting jobs of $\mathcal{VC}$ into gaps defined by the sequence of jobs of $\mathcal{J} - \mathcal{VC}$. Considering some interval $[u_\alpha, u_{\alpha+1})$ - i.e. where no release time/deadline from the vertex cover is interfering - this is reminiscent of the multiple knapsack problem, for which several polynomial kernels were proposed by [GRR19]. Indeed, in a multiple knapsack instance with k items, any solution can be converted to one where only the k knapsacks of highest capacity are used. In our problem the gaps between consecutive jobs of $\mathcal{J} - \mathcal{VC}$ could be interpreted as knapsacks, except their capacities can be adjusted from both ends by changing when their delimiters are scheduled. And doing so impacts the capacity of the neighboring gaps. Still we manage to prove that, in order to find a feasible schedule, we only need to keep a bounded number of inner gaps in each interval $[u_\alpha, u_{\alpha+1})$.

Definition 15. *We denote vc_α the number of jobs of vertex cover $\mathcal{VC}$ such that $[u_\alpha, u_{\alpha+1})$ is included in their time window: $vc_\alpha = |\{j \in \mathcal{VC}, r_j \leq u_\alpha \text{ and } u_{\alpha+1} \leq d_j\}|$.*

Lemma 16. *If the instance I is feasible then there is a schedule where every job i in vertex cover $\mathcal{VC}$ is scheduled either in a border gap or in one of the $3vc_\alpha$ inner gaps with highest maximum capacity of some interval $[u_\alpha, u_{\alpha+1})$.*

Proof. Proof. Suppose we have a feasible schedule τ for instance I and consider an interval $[u_\alpha, u_{\alpha+1})$. If there are less than $3vc_\alpha$ inner gaps, then nothing needs to be done. Assume there are more than $3vc_\alpha$ inner gaps. The key idea of the proof is that when a gap (w_j, w_{j+1}) is at maximum capacity, it only impacts the capacity of neighboring gaps (w_{j-1}, w_j) and (w_{j+1}, w_{j+2}). So, by considering $3vc_\alpha$ gaps, we can always have vc_α gaps among them at maximum capacity.

Let $g_1, \ldots, g_{3vc_\alpha}$ be the $3vc_\alpha$ inner gaps with the highest maximum capacity. Let τ be a feasible schedule on instance I. Then in interval $[u_\alpha, u_{\alpha+1})$ at most vc_α inner gaps of $[u_\alpha, u_{\alpha+1})$ are filled by jobs from vertex cover $\mathcal{VC}$. Let $g'_1, \ldots, g'_k$ be these gaps ($k \leq vc_\alpha$). We show that the jobs scheduled in these gaps can be scheduled in gaps $g_1, \ldots, g_{3vc_\alpha}$ instead while keeping feasibility. We do so by mapping each gap g'_i to some gap g_j with larger or equal maximum capacity.

If a gap g'_i is already one of the $3vc_\alpha$ gaps with the highest maximum capacity then we assign it to itself. Then for all other gaps g'_i we assign a gap g_j which is a neighbor of no other assigned gap $g_{j'}$. This allows us to use gap g_j at its maximum capacity, which is necessarily higher than or equal to the capacity used by schedule τ in gap g'_i. Since each gap has at most two neighbors, at any iteration of this process there can only be at most $3(k-1) < 3vc_\alpha$ gaps $g_1, \ldots, g_{3vc_\alpha}$ which are either already assigned or a neighbor of an already assigned gap. So for each gap g'_i used in schedule τ a suitable gap g_j can be found in $\mathcal{O}(vc_\alpha)$ time. Plus *prec*-consistency ensures that within every interval $[u_\alpha, u_{\alpha+1})$ there is no precedence relation between a job from the vertex cover and a job from an inner gap in $[u_\alpha, u_{\alpha+1})$. So the jobs in τ can be moved to their assigned gaps without invalidating any precedence constraint. Thus the schedules which only use the (at most) $3vc_\alpha$ inner gaps with highest maximum capacity in each interval $[u_\alpha, u_{\alpha+1})$ are dominant. □

We now aim for a reduction rule which reduces the set of jobs by only keeping useful gaps. To this end, we first define a basic transformation of the instance, called a Left Shift, which will be used in the next reduction rule.

Definition 17. *Let I be an instance of our problem and let (w_i, w_{i+1}) be an inner gap. We call* **Left Shift** *the following modified instance $I' = LS(I, i)$,*

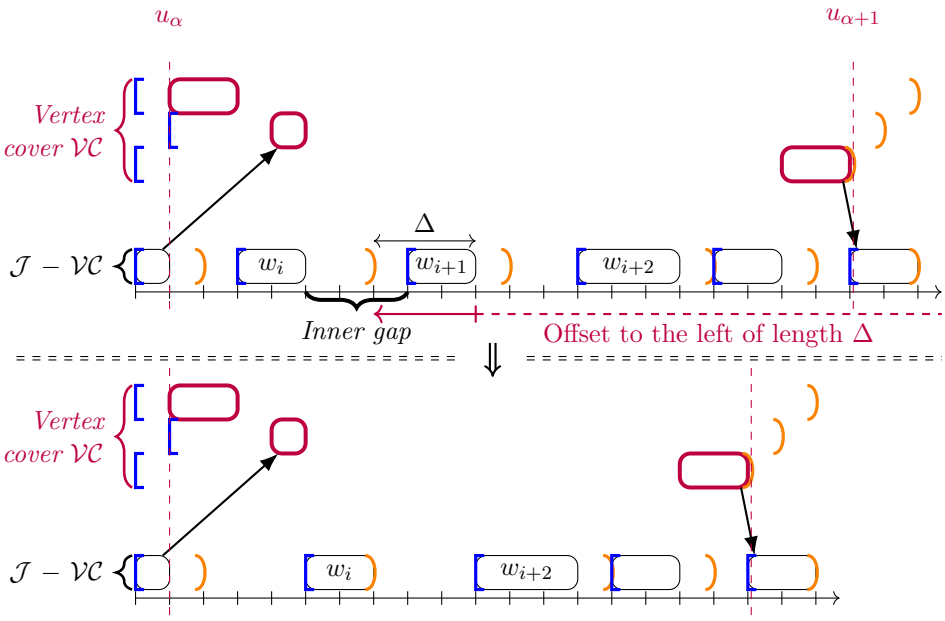


Figure 4: Job removal process $LS(I, i)$ for inner gap w_i, w_{i+1}

in which:

- *the release time of job w_i is changed so that this job has no flexibility anymore: $r'_{w_i} = d_{w_i} - p_i$,*
- *job w_{i+1} is removed,*
- *an offset $-\Delta = -(p_{w_{i+1}} + r_{w_{i+1}} - d_{w_i})$ is applied to any release time and deadline which is greater than $d_{w_{i+1}}$: for all j in $\mathcal{J}$, if $r_j > d_{w_{i+1}}$ then $r'_j = r_j - \Delta$; if $d_j > d_{w_{i+1}}$ then $d'_j = d_j - \Delta$.*

Figure 4 illustrates $LS(I, i)$. We observe that after applying $LS(I, i)$, the new gap between w_i and w_{i+2} has minimum length $(r_{w_{j+2}} - r_{w_{i+1}} + p_{W_{i+1}})$ and the same capacity as the previous gap between w_{i+1} and w_{i+2}. All the following gaps are just shifted and still have the same capacity and flexibility. Plus an inner gap remains an inner gap in the modified instance.

We now state the following lemma, which describes a condition under which a Left Shift is a valid reduction.

Lemma 18. *Let I be an instance, let $[u_\alpha, u_{\alpha+1})$ be one of its interval and let w_i, w_{i+1} an inner gap of this interval. If:*

(I is feasible) $\implies$ *(there exists a feasible schedule for which no job of $\mathcal{VC}$ is scheduled in gap $\{w_i, w_{i+1}\}$),*

then $LS(I, i)$ is valid.

Proof. Proof. First, notice that due to the *prec*-consistency assumption, there are no precedence relations between the jobs of the vertex cover that can be scheduled in $[u_\alpha, u_{\alpha+1})$ and the jobs of the inner gap w_i, w_{i+1}. Now let $\tau = (s_j)_{j\in\mathcal{J}}$ be a feasible schedule for which no job of the vertex cover is scheduled in gap w_i, w_{i+1}. We can thus assume without loss of generality that in such a schedule, w_i is right-shifted so that it starts at time $d_{w_i} - p_i$; w_{i+1} is left-shifted so that it starts at time $r_{w_{i+1}}$. Let $\Delta = (p_{w_{i+1}} + r_{w_{i+1}} - d_{w_i})$. Then, in schedule τ, the only job scheduled between times d_{w_i} and $d_{w_i} + \Delta$ is w_{i+1}. Let us now define schedule $\tau' = (s'_j)_{j\in\mathcal{J}}$:

$$s'_j = s_j - \Delta \text{ if } s_j \geq s_{w_{i+1}} + p_{w_{i+1}} \tag{3}$$

Then τ' satisfies the time window constraints of instance $LS(I, i)$. Indeed, for jobs w_k with $k > i + 2$ and jobs j of the vertex cover such that $r_j \geq u_{\alpha+1}$, both the time window and the job starting time are left-shifted by Δ. Now take any job j of the vertex cover with a release time $r_j \leq u_\alpha$. If it is scheduled after gap (w_k, w_{k+1}) in τ then only its starting time and its deadline are left-shifted by Δ. Then it is still scheduled before job w_i, so the release time constraint is satisfied. Otherwise, only its deadline is potentially left-shifted by Δ. But this job is still scheduled before w_j, so the deadline constraint is guaranteed to be satisfied.

Conversely, let us consider a schedule τ' of the modified instance $LS(I, i)$. We can build a feasible schedule τ of the original instance. Let j be a job. If $s'_j \leq r'_{w_i}$ then we set $s_j = s'_j$. If it was the removed job, we set $s_{w_{i+1}} = r_{w_{i+1}}$. Finally if $s'_j \geq d'_{w_i}$ then we set $s_j = s_j + \Delta$. Schedule τ satisfies the time window constraints and the precedence constraints of the original instance. □

We now define our first reduction rule.

Reduction rule 19. *For each α: if there are more than $3vc_\alpha$ inner gaps in $[u_\alpha, u_{\alpha+1})$, select the $3vc_\alpha$ ones with highest maximum capacity. We denote by S the set of selected gaps to which we add the border gaps. While there is an inner gap w_i, w_{i+1} not in S, update the instance: $I \leftarrow LS(I, i)$.*

Lemma 20. *Reduction rule 19 is valid. The number of jobs w_j from $\mathcal{J} - \mathcal{VC}$ can be reduced down to $\mathcal{O}(vc_\alpha)$ in each interval $[u_\alpha, u_{\alpha+1})$ and thus to $\mathcal{O}(vc^2)$*

in the whole instance. The time complexity is $\mathcal{O}(vc^2 \cdot n + n\log(n))$. The size of the resulting encoded instance is $\mathcal{O}(vc^2 \log(D))$, where $D = \max_{j\in\mathcal{J}} d_j$.

Proof. Proof. Consider an interval $[u_\alpha, u_{\alpha+1})$ of a feasible instance I. Let X_α be the set of inner gaps in $[u_\alpha, u_{\alpha+1})$ which are not among the $3vc_\alpha$ of highest maximum capacity in this interval. By Lemma 16 there exists a schedule such that no job of $\mathcal{VC}$ is scheduled in a gap of X. And Lemma 18 indicates that for any gap (w_i, w_{i+1}) in X_α, then Left Shift $LS(I, i)$ is valid and leads to an instance with one less gap - and thus one less job - and unchanged maximum capacity for all other gaps. So Left Shift operations can be iterated on each gap of X_α until it is empty. The instance remains with at most $3vc_\alpha$ inner gaps in interval $[u_\alpha, u_{\alpha+1})$. Since the remaining gaps are then consecutive, at most $3vc_\alpha + 3$ jobs of $\mathcal{J} - \mathcal{VC}$ remain in interval $[u_\alpha, u_{\alpha+1})$. We can then iterate on each α and the reduction keeps the equivalence between intermediate instances. Once Reduction rule 19 cannot be applied anymore, as there are at most $2vc - 1$ possible values of α, the resulting instance has $\mathcal{O}(vc^2)$ jobs. Plus, by Corollary 13, it has $\mathcal{O}(vc^2)$ precedence relations.

Let us determine the time complexity of the proposed kernelization algorithm. During a preprocessing phase we first ensure that the input instance is *prec*-consistent in $\mathcal{O}(n^2)$ time. Then a vertex cover of minimum size vc is computed in $\mathcal{O}(|G_I|) = \mathcal{O}(n^2)$ time [MRR92]. After that, the corresponding sequence $(u_\alpha)_\alpha$ of interval bounds is computed in $\mathcal{O}(vc\log(vc))$ time. Finally we sort the gaps in each interval $[u_\alpha, u_{\alpha+1})$ in $\mathcal{O}(n\log(n))$ time. Removing all but $\mathcal{O}(vc^2)$ jobs from $\mathcal{J} - \mathcal{VC}$ can be done by applying a Left Shift $\mathcal{O}(vc^2)$ times. The time complexity of a Left Shift is in $\mathcal{O}(n)$. Hence the overall time complexity is in $\mathcal{O}(vc^2 \cdot n + n\log(n))$. □

4.2. Reducing the Time Values

In this section we give an additional reduction rule which reduces the size of the kernel of jobs down to $\mathcal{O}(vc^8)$. Similarly to several previous approaches on knapsack and scheduling problems [EKMR17, GRR19, KZ20], we intend to use the framework by [FT87] in order to bound the processing time, release time and deadline values by a function of vc.

First we consider the following binary integer program with positional variables which was proved equivalent to the associated scheduling instance by [LQ92]:

$$x_{i,j} = \begin{cases} 1 & \text{if job } j \text{ is the } i^{th} \text{ job in the schedule,} \\ 0 & \text{otherwise.} \end{cases}$$

The linear program is defined as follows:

$$BIP(I): \begin{cases} \text{One job per position} & \sum_{j=1}^{n} x_{i,j} = 1 \text{ for all } 1 \leq i \leq n. \\ \text{One position per job} & \sum_{i=1}^{n} x_{i,j} = 1 \text{ for all } 1 \leq j \leq n. \\ \text{Precedence constraints} & \sum_{i=1}^{i_0} (x_{i,j_1} - x_{i,j_2}) \geq 0 \text{ for all } 1 \leq i_0 \leq n, (j_1, j_2) \in A. \\ \text{Time window checks} & \sum_{j=1}^{n} r_j \cdot x_{i_1,j} + \sum_{i=i_1}^{i_2} \sum_{j=1}^{n} p_j \cdot x_{i,j} \leq \sum_{j=1}^{n} d_j \cdot x_{i_2,j} \\ & \text{for all } 1 \leq i_1 \leq i_2 \leq n. \end{cases}$$

The linear constraint associated with a precedence constraint (j_1, j_2) in A checks whether, for each position i_0, if j_2 is scheduled in an earlier position than i_0, then j_1 is too. For each tuple of positions $i_1 \leq i_2$, the constraints representing time windows check that the time value given by the release time of the job in position i_1 plus the sum of the processing times of all jobs in consecutive positions i_1 to i_2 does not exceed the deadline of the job in position i_2. The constraints rely on the dominance of **active** schedules - i.e. with no unnecessary idle time - for makespan minimization. So such a schedule can be decomposed into "blocks" of jobs which start at the release time of the first job in each block. The constraints consider all the $n(n-1)/2$ block possibilities.

Lemma 21. *[LQ92] An instance I of $1|prec, r_j, d_j|\star$ is feasible if and only if binary integer program $BIP(I)$ has a solution.*

Although there are $\Theta(n^2)$ variables in some constraints of $BIP(I)$, given a solution of the program, we show that at most $n+2$ variables can be equal to one in each constraint.

Lemma 22. *In any solution of $BIP(I)$ at most $n+2$ variables are equal to one in each constraint.*

Proof. Proof. At most one variable can be equal to one in the first two sets of constraints (one job per position and one position per job). Thus, for the precedence constraints, at most two variables might be equal to one in the sum. Now, considering the time window constraint, it sums up with release time coefficient the variables of jobs in position i_1 (so only 1 variable equals one), similarly the third term of the sum considers the deadlines of jobs in position i_2, so only one of them equals one, whereas the intermediate sum considers all positions between i_1 and i_2, so at most n variables equal one. This gives our $(n+2)$ bound. □

Knowing this, we now consider the framework by [FT87]:

Lemma 23. *[FT87] Let $\mathbf{w} = (w_1, \ldots, w_d)$ be a vector of rational numbers and let N be a positive integer. Then in time $\mathcal{O}(d^8 \cdot (d + \log(N)))$ one can build a vector $\mathbf{w'} = (w'_1, \ldots, w'_d)$ of positive integers such that:*

- $w'_i \leq 2^{4d^3} \cdot N^{d(d+2)}$ *for all* $1 \leq i \leq d$,
- *for every* $\mathbf{x} = (x_1, \ldots, x_d)$ *vector of integers such that* $\sum_{i=1}^{d} |x_i| \leq N - 1$*:*

$$\sum_{i=1}^{d} w_i x_i \geq 0 \text{ if and only if } \sum_{i=1}^{d} w'_i x_i \geq 0.$$

Let I be an instance of $1|prec, r_j, d_j|\star$ and let $\kappa(I)$ be the instance obtained with Lemma 20. Let $\tilde{n}$ be the number of its jobs. We know that $\tilde{n} \in \mathcal{O}(vc^2)$. By Lemma 21 we can consider the equivalent program $BIP(\kappa(I))$. We wish to modify the values of the release times, deadlines and processing times using Lemma 23 to make the instance size polynomial in the number of jobs and thus polynomial in parameter vc.

To this end, we set $d = 3\tilde{n}$ and consider vector $\mathbf{w} = (p_1, r_1, d_1, \ldots, p_{\tilde{n}}, r_{\tilde{n}})$. Let us set $N = \tilde{n}+3$. Notice that $d, \tilde{n}$ and N are all in $\mathcal{O}(vc^2)$. Now Lemma 22 states that any feasible solution of $BIP(\kappa(I))$ satisfies that the sum of variables involved in a time window constraint is at most $N - 1$. We can thus apply Lemma 23 to get a vector $\mathbf{w}' = (p'_1, r'_1, d'_1, \ldots, p'_{\tilde{n}}, r'_{\tilde{n}}, d'_{\tilde{n}})$, which represents an instance I' equivalent to $\kappa(I)$ according to Lemma 21. The time values in I' are all bounded by a function of vc. This gives us the following reduction rule:

Reduction rule 24. *Let I be an instance I and $\kappa(I)$ be the instance obtained after applying iteratively reduction 19.Let $\tilde{n}$ be the number of jobs in $\kappa(I)$. Then we set vector $\mathbf{w} = (p_1, r_1, d_1, \ldots, p_{\tilde{n}}, r_{\tilde{n}}, d_{\tilde{n}})$ from the time values of $\kappa(I)$ and apply Lemma 23 with $d = 3\tilde{n}$ and $N = \tilde{n} + 3$.*

Applying our reduction rules leads to a polynomial kernel with respect to vc.

Theorem 25. *With respect to parameter vc problem $1|prec, r_j, d_j|\star$ has a kernelization that can be computed in time $\mathcal{O}(vc^2 \cdot n + n\log(n) + (vc^2)^8 \cdot (vc^2 + \log(vc^2))) = \mathcal{O}(n^{18})$ and outputs a polynomial kernel of size $\mathcal{O}(vc^8)$ with $\mathcal{O}(vc^2)$ jobs.*

Proof. Proof. Let I be a *prec*-consistent instance of problem $1|prec, r_j, d_j|\star$. First we build instance $\kappa(I) = \langle prec, p_j, r_j, d_j\rangle$ obtained in $\mathcal{O}(vc^2 \cdot n + n\log(n))$ time by Lemma 20. Let I' be the instance build from $\kappa(I)$ by the reduction rule 24.

We know that the new coefficients are not greater than $2^{4d^3} \cdot N^{d(d+2)}$. Substituting d by $3\tilde{n}$ and N by $\tilde{n}+3$ yields vector $\mathbf{w}' = (p'_1, r'_1, d'_1, \ldots, p'_{\tilde{n}}, r'_{\tilde{n}}, d'_{\tilde{n}})$

with components not greater than $2^{\mathcal{O}(\tilde{n}^3)}$. Since $\tilde{n} = \mathcal{O}(vc^2)$, the size of each encoded component is $\mathcal{O}(vc^6)$. And since we have $\mathcal{O}(vc^2)$ jobs in I' we get the announced space bound for it.

Similarly, the time complexity $\mathcal{O}(d^8 \cdot (d + \log(N)))$ from Lemma 23 yields time complexity $\mathcal{O}((vc^2)^8 \cdot (vc^2 + \log(vc^2)))$. Use these values to define $I' = \langle prec, p'_j, r'_j, d'_j \rangle$ instance of $1|prec, r_j, d_j|\star$. We show that $\kappa(I)$ is feasible if and only if I' is feasible. Lemma 22 ensures that bound $N = \tilde{n} + 3$ of all the positive values of variables in each inequality was enough to make integer programs $BIP(\kappa(I))$ and $BIP(I')$ equivalent according to Lemma 23. Plus by Lemma 21 we know that $\kappa(I)$ (resp. I') is feasible if and only if $BIP(\kappa(I))$ (resp. $BIP(I')$) has a solution. So I' is indeed equivalent to $\kappa(I)$, which is itself equivalent to I according to Lemma 20.

□

Considering that the processing times are rather bounded in many practical scheduling problems, instead of Reduction rule 24 we can use another reduction to get an instance with size polynomial in vc and $\log(p_{\max})$, where $p_{\max}$ is the maximum processing time with a lower time complexity. Let I be an instance of the problem, and let $(v_\beta)_{\beta \geq 0}$ be the sorted sequence of release times and deadlines of jobs in $\mathcal{J}$. Notice that in each interval $[v_\beta, v_{\beta+1})$ at most $vc + 1$ jobs could be performed: the jobs of $\mathcal{VC}$ and the only job w_j (if any) whose time window crosses this interval. As such we propose the following reduction rule:

Reduction rule 26. *If $v_{\beta+1} - v_\beta > (vc+1) \cdot p_{\max}$, then set $t = v_\beta + (vc+1) \cdot p_{\max}$. Apply an offset of $-(v_{\beta+1} - t)$ to all release times and deadlines $\geq v_{\beta+1}$.*

The validity of Reduction rule 26 is left to the reader. After iterative application, the maximum deadline is in $\mathcal{O}(vc^3 \cdot p_{\max})$. Thus, when combined with Lemma 16, a kernel of $\mathcal{O}(vc^2 \cdot [\log(vc) + \log(p_{\max})])$ size can be obtained in $\mathcal{O}(n^2 \cdot \log(n))$ time.

5. Refining the Polynomial Kernel with respect to the Vertex Cover Number

In this section we revisit the approach used in the previous section to derive a polynomial kernelization with respect to a parameter which allows for vertex covers of unbounded size.

5.1. Bounding the Number of Time Window Bounds in the Vertex Cover

Looking back at the polynomial kernel with respect to vc from Section 4, since $vc_\alpha \leq pw$, using parameter pw instead seems to almost work. The only catch is that pathwidth pw is unrelated to the number of intervals $[u_\alpha, u_{\alpha+1})$. As such, we propose to combine pathwidth pw with the following parameter:

Definition 27.
- *Given a vertex cover $\mathcal{VC}$ of G_I, we define $\eta(\mathcal{VC})$ as the number of different release time and deadline values among the jobs in $\mathcal{VC}$.*
- *We define parameter $\eta_{\min}$ as the minimum value of $\eta(\mathcal{VC})$ among all vertex covers $\mathcal{VC}$ of G_I.*

Among investigated parameters on single machine scheduling problems, the number of different release time and/or deadline values has been considered several times when maximizing the weighted number of tardy jobs [HKPS21, KMM24]. With parameter $\eta_{\min}$ we only take into account the time bounds of the tasks in a given vertex cover of the compatibility graph.

Furthermore, note that with parameter $\eta_{\min}$ we consider all vertex covers and not just the ones of minimum size. In the example given in Figure 5, $\eta_{\min}$ is obtained e.g. by taking all black jobs and green jobs as the vertex cover. However, a vertex of minimum size necessarily includes all purple jobs, which nearly triples the number of different release time and deadline values in the vertex cover - for instance when taking all purple jobs and all black jobs as the vertex cover.

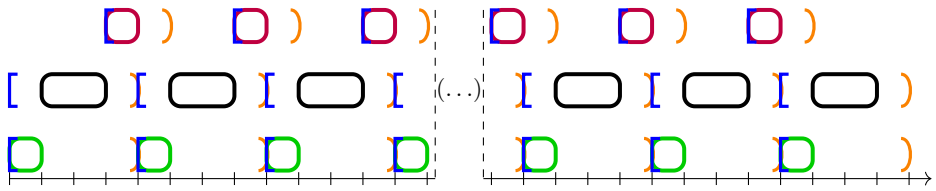


Figure 5: An example where $\eta_{\min}$ is only attained by vertex covers of size greater than vc

To the best of our knowledge, it is unknown whether finding a vertex cover $\mathcal{VC}$ such that $\eta(\mathcal{VC}) = \eta_{\min}$ can be done in polynomial time. Luckily, we can show that there are vertex covers of minimum size vc, the number of time window bounds of which is within a constant factor of $\eta_{\min}$.

Lemma 28. *Let $\mathcal{VC}$ be a vertex cover of G_I of minimum size vc such that for all jobs i in $\mathcal{VC}$, there is no job j in $\mathcal{J}$ such that $r_j < r_i$ and $d_i < d_j$. Then: $\eta(\mathcal{VC}) \leq 3\eta_{\min} + 2$.*

Proof. Proof. Let $\mathcal{VC}$ be such a vertex cover. Let $(u_i)_{1\leq i\leq \eta(\mathcal{VC})}$ be the increasing sequence of its different release time and deadline values. Let $\mathcal{VC}'$ be a vertex cover such that $\eta(\mathcal{VC}') = \eta_{\min}$ and let $(u'_i)_{1\leq i\leq \eta_{\min}}$ be the increasing sequence of its different release time and deadline values. We set $u'_0 = -\infty$ and $u'_{\eta_{\min}+1} = +\infty$.

By contradiction suppose that $\eta(\mathcal{VC}) \geq 3\eta_{\min} + 3$. Then, by the pigeonhole principle, there are indices α in $[1, \eta(\mathcal{VC}) - 2]$ and β in $[0, \eta_{\min}]$ such that $u'_\beta < u_\alpha$ and $u_{\alpha+2} < u'_{\beta+1}$. Note that there must be two values among $u_\alpha, u_{\alpha+1}$ and $u_{\alpha+2}$ which are either both release times or both deadlines of jobs in $\mathcal{VC}$. Thus there are at least two jobs j_1, j_2 in $\mathcal{VC}$ such that either $u'_\beta < r_{j_1}, r_{j_2} < u'_{\beta+1}$ or $u'_\beta < d_{j_1}, d_{j_2} < u'_{\beta+1}$ (or both).

Suppose the time windows of j_1 and j_2 did not intersect. Then one of them would be included in open interval $(u'_\beta, u'_{\beta+1})$ - say the time window of j_1 without loss of generality. By the additional property given to vertex cover $\mathcal{VC}$, there would be no job j in $\mathcal{J}$ - and thus in $\mathcal{VC}'$ - such that $r_j \leq u'_\beta$ and $u'_{\beta+1} \leq d_j$. So $\mathcal{VC}'$ would be a vertex cover with no job time window intersecting with $[r_{j_1}, d_{j_1})$. Thus j_1 would be an isolated job, which could be removed from $\mathcal{VC}$ to obtain a vertex cover of size $vc - 1$, contradicting the minimality of vc.

Finally if the time windows of j_1 and j_2 intersected, then every vertex cover would include at least one of these jobs. However, both have a time window bound which does not appear in sequence $(u'_i)_{1\leq i\leq \eta_{\min}}$. So none of the two would be in $\mathcal{VC}'$, which would contradict that the latter is a vertex cover. □

Such a vertex cover can be found in polynomial time. First, a vertex cover $\mathcal{VC}_0$ of minimum size vc can be found in linear time. If there are i in $\mathcal{VC}_0$ and j in $\mathcal{J}$ such that $r_j < r_i$ and $d_i < d_j$, then replacing i with j would still result in a vertex cover. By taking j such that there is no job j' with both a lower release time and a higher deadline, after a number of iterations at most linear in vc we get the desired additional property.

Under P $\neq$ NP, parameter $\eta_{\min}$ alone does not allow for fixed-parameter algorithms for our problem. Indeed, by representing a single instance of the NP-complete Partition problem introduced in Definition 10 in a similar manner as in Section 3, we have $\eta_{\min} = 2$. In other words, our problem is para-NP-complete parameterized by $\eta_{\min}$.

While pw and $\eta_{\min}$ are bounded by vc and $2vc$ respectively, we show that vc is bounded by a function of pw and $\eta_{\min}$.

Lemma 29. $vc \leq (pw + 1) \cdot (\eta_{\min} - 1)$.

Proof. Proof. Let $\mathcal{VC}$ be a vertex cover of minimum size vc and let $\mathcal{VC}'$ be a vertex cover with a number of different release time and deadline values equal to $\eta_{\min}$. Then: $vc = |\mathcal{VC}| \leq |\mathcal{VC}'|$. Now, sort the different release time and deadline values associated to $\mathcal{VC}'$ into an increasing sequence $(u'_\alpha)_{1 \leq \alpha \leq \eta_{\min}}$. Then given each segment $[u'_\alpha, u'_{\alpha+1})$, the number of jobs from $\mathcal{VC}'$, the time window of which intersects with this segment, is bounded by $pw + 1$. Since the time window of every job in $\mathcal{VC}'$ intersects with at least one of the segments $[u_\alpha, u'_{\alpha+1})$, we have:

$|\mathcal{VC}'| \leq \sum_{\alpha=1}^{\eta_{\min}-1}(pw + 1) \leq (pw + 1) \cdot (\eta_{\min} - 1)$. □

This bound is tight and attained when, for every $0 \leq \alpha \leq \eta_{\min}-1$, exactly $pw + 1$ jobs from the vertex cover have time window $[u_\alpha, u_{\alpha+1})$. Lemma 29 implies that the vertex cover number vc is equivalent to parameter $(pw + \eta_{\min})$ in terms of fixed-parameter tractability and the existence of polynomial kernels. In essence, we decomposed vc into a "width" parameter (pw), restricting the number of jobs to consider at any given time, and a "length" parameter $(\eta_{\min})$, bounding the number of time intervals to examine.

5.2. A Polynomial Kernel with respect to the Proper Level plus the Minimum Number of Time Window Bounds in the Vertex Cover

Following the last remark from the previous subsection, we show that going from pathwidth pw to proper level pl as the "width" parameter still leads to a polynomial kernel. Observe first that unlike with $(pl + \eta_{\min})$, vc is not bounded by a function of parameter $(pl + \eta_{\min})$. Indeed, if we have n independent tasks with the same time window, then: $pl = 0, \eta_{\min} = 2$ and $vc = n - 1$.

Recall that after applying Reduction Rule 19 in the polynomial kernel with respect to vc, the number of inner gaps in each interval $[u_\alpha, u_{\alpha+1})$ was linearly bounded by the number vc_α of jobs from the vertex cover, the time window of which intersected with interval $[u_\alpha, u_{\alpha+1})$. Note that the number of such inner gaps can be linearly bounded by proper level pl instead.

Lemma 30. *For every interval $[u_\alpha, u_{\alpha+1})$ with at least two inner gaps: $vc_\alpha \leq pl$.*

Proof. Proof. If there are at least two inner gaps in an interval $[u_\alpha, u_{\alpha+1})$, then there is a job j such that $u_\alpha < r_j$ and $d_j < u_{\alpha+1}$. So all the vc_α jobs from the vertex cover, the time window of which intersects with interval $[u_\alpha, u_{\alpha+1})$, have a lower release time than r_j and a higher deadline than d_j. Thus $vc_\alpha \leq pl$. □

So Reduction Rule 19 can be applied and, according to Lemma 20, it leads to a reduction of the number of jobs of $\mathcal{J}-\mathcal{VC}$ in this interval to $\mathcal{O}(pl)$. Once the reduction rule does not apply anymore, the number of remaining jobs in $\mathcal{J}-\mathcal{VC}$ is in $\mathcal{O}(pl\cdot\eta_{\min})$.

Additionally, we propose a reduction rule which will reduce the number of jobs in $\mathcal{VC}$. Let u_α, u_β be two time window bounds of $\mathcal{VC}$ with $\alpha<\beta$. We consider the subset $\mathcal{VC}_{\alpha,\beta}$ of the vertex cover such that:

$$\mathcal{VC}_{\alpha,\beta}=\{j\in\mathcal{VC},(r_j=u_\alpha)\wedge(d_j=u_\beta)\}. \tag{4}$$

Reduction rule 31. *For each interval $[u_\alpha,u_\beta)$, if $|\mathcal{VC}_{\alpha,\beta}|>pl$ then merge all the jobs of $\mathcal{VC}_{\alpha,\beta}$ into a single job with processing time equal to the sum of their processing times.*

Lemma 32. *Reduction rule 31 is valid.*

Proof. Proof. If $|\mathcal{VC}_{\alpha,\beta}|>pl$ then no job has its time window strictly enclosed in interval $[u_\alpha,u_\beta)$. This means that if the time window of job j overlap with $[u_\alpha,u_\beta)$ then either $r_j\leq u_\alpha$ or $d_j\geq u_\beta$. Consider a feasible schedule. Without loss of generality we can reorder the tasks in interval $[u_\alpha,u_\beta)$ so that we execute first the jobs with $d_j<u_\beta$ in non decreasing order of deadlines, then all the jobs of $\mathcal{VC}_{\alpha,\beta}$ in any order and then the jobs with $r_j>u_\alpha$ in non decreasing order of r_j. No precedence constraint nor time window can be violated, considering the *prec*-consistency of the instance. This means that, without loss of generality, we can merge all jobs of $\mathcal{VC}_{\alpha,\beta}$ into a single job with processing time equal to the sum of processing times of jobs in $\mathcal{VC}_{\alpha,\beta}$. □

Lemma 33. *Given a vertex cover $\mathcal{VC}$ of minimum size vc such that $\eta(\mathcal{VC})=\mathcal{O}(\eta_{\min})$, applying Reduction rules 19 and 31 iteratively reduces the number of jobs to $\mathcal{O}(pl\cdot\eta_{\min})$.*

Proof. Proof. Because Reduction rule 19 does not apply anymore, by Lemmas 20 and 30 there are $\mathcal{O}(pl\cdot\eta_{\min})$ jobs in $\mathcal{J}-\mathcal{VC}$. Now we show that there are $\mathcal{O}(pl\cdot\eta_{\min})$ jobs in $\mathcal{VC}$ as well. To this end, a job j in vertex cover $\mathcal{VC}$ is called a **$\mathcal{VC}$-summit** if there is no job i in $\mathcal{VC}$ such that $r_j<r_i$ and $d_i<d_j$. We partition $\mathcal{VC}$ into $S\cup T$ where S is the subset of the $\mathcal{VC}$-summits.

We claim that there are $\mathcal{O}(\eta_{\min})$ couples (α,β) such that $\mathcal{VC}_{\alpha,\beta}\cap S$ is nonempty. Given a couple (α,β) we define its **index average** as rational number $(\alpha+\beta)/2$. By contradiction suppose that there are two different couples (α,β), (α',β') with the same index average r and with an nonempty intersection with S. Let $j\in\mathcal{VC}_{\alpha,\beta}\cap S$ and $j'\in\mathcal{VC}_{\alpha',\beta'}\cap S$. Clearly any

couple with index average r is of the form $(r-x, r+x)$ for some positive real x. Thus either ($\alpha < \alpha'$ and $\beta' < \beta$) or ($\alpha' < \alpha$ and $\beta < \beta'$). But then this would contradict that j and j' are $\mathcal{VC}$-summits. Since the index average of every couple is in $\{k/2, 3 \leq k \leq 2\eta(\mathcal{VC}) - 1\}$, there are at most $(2\eta(\mathcal{VC}) - 3)$ couples (α, β) such that $\mathcal{VC}_{\alpha,\beta} \cap S$ is nonempty - i.e. $\mathcal{O}(\eta_{\min})$ such couples.

Knowing this, and because Reduction rule 31 does not apply anymore, we deduce that $|S| = \mathcal{O}(pl \cdot \eta_{\min})$. Now, take a job j in T. It is not a $\mathcal{VC}$-summit, thus there is a job i in $\mathcal{VC}$ such that $r_j < r_i$ and $d_i < d_j$. Either i is a $\mathcal{VC}$-summit or we repeat this reasoning until we come across a $\mathcal{VC}$-summit s such that $r_j < r_s$ and $d_s < d_j$. Let α, β be such that $r_s = u_\alpha$ and $d_s = u_\beta$. Then, by the definition of proper level pl, there cannot be more than pl tasks j in T such that $r_j < u_\alpha$ and $u_\beta < d_j$. Hence, by the previous paragraph there are $\mathcal{O}(pl \cdot \eta_{\min})$ jobs in T. To conclude, we showed that there are $\mathcal{O}(pl \cdot \eta_{\min})$ jobs in $\mathcal{VC}$. □

We deduce the existence of a polynomial kernel with respect to $(pl+\eta_{\min})$.

Theorem 34. *Problem* $1|prec, r_j, d_j|\star$ *has a polynomial kernelization with respect to parameter* $(pl + \eta_{\min})$.

6. Conclusion

In this paper we have provided a first overview of the existence of polynomial kernels on single machine scheduling with precedence constraints and job time windows. Figure 6 summarizes these kernelization results in the form of a parameter map. This gives a first insight on reduction rules which could be used to reduce the size of an instance in practice, especially when the number of job types is unbounded. This also confirms the apparent difficulty of scheduling problems, which need restrictive parameters to derive polynomial kernels even in the single machine setting.

Regarding the compatibility graph, the family of *distance to* X parameters, where X is a graph class, has yet to be considered. Such parameters ask for the minimum number of vertices to be removed in order to reach the requested graph class. In this setting the vertex cover number can be interpreted as *distance to independent set*. This opens up several options to attempt an approach similar to ours on a less restrictive parameter. For instance the next investigated parameter could be *distance to disjoint paths*, for which it is unclear whether a result similar to Lemma 16 could be obtained. Indeed, consider N unit-time jobs $j_1, \ldots, j_N$ where job j_k has time window $[k, k+2)$. This defines a path in the compatibility graph, while

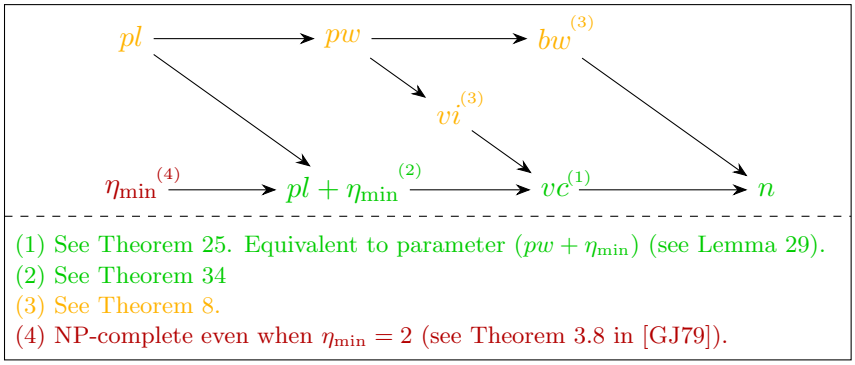


Figure 6: Polynomial kernel status and relations between parameters of $1|prec, r_j, d_j|\star$. A parameter is green when there is a polynomial kernel, orange when there is an exponential size kernel and no polynomial kernel (unless NP $\subseteq$ coNP/poly) and red when there is no kernelization algorithm (unless P = NP)

right-shifting j_1 would impact the scheduling of all other jobs j_k. Thus an approach different from the proof of Lemma 16 would be needed to bound the number of jobs by a polynomial function of the parameter.

On another note, our study could be extended to parameters which take into account the job processing times on top of the time windows, or which are based on the precedence graph. For example the slack of an instance is the maximum difference between the processing time of a job and the length of its time window. This parameter linearly bounds pathwidth pw in the single machine setting (see e.g. Lemma 5.3 in [vBNS17]). So problem $1|prec, r_j, d_j|\star$ is fixed-parameter tractable with respect to the slack, and the latter is among the next candidate parameters for future polynomial kernels. It is especially interesting to note that allowing for unbounded slack is a key ingredient in both cross-compositions proposed in this paper.

Finally, we wonder whether our approach could be adapted to the identical parallel machine setting. In this setting with precedence constraints and job time windows, it is known that the problem restricted to unit-time jobs is fixed-parameter tractable with respect to pw [MK21]. So a similar study could be conducted on this problem. With arbitrary processing times, the problem becomes para-NP-complete with respect to pw [HMK23], whereas its parameterized complexity with respect to vc is open. The latter could be answered while looking for a (potentially polynomial) kernelization algorithm.

References

[BES87] C. A. Barefoot, Roger Entringer, and Henda Swart. Vulnerability in graphs-a comparative survey. *Journal of Combinatorial Mathematics and Combinatorial Computing (JCMCC)*, 1(38):13–22, 1987.

[BG19] S. Bessy and R. Giroudeau. Parameterized complexity of a coupled-task scheduling problem. *Journal of Scheduling*, 22(3):305–313, June 2019.

[BJ22] Hauke Brinkop and Klaus Jansen. High multiplicity scheduling on uniform machines in fpt-time. *CoRR*, abs/2203.01741, 2022.

[BJK14] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014.

[BM93] Hans L Bodlaender and Rolf H Möhring. The pathwidth and treewidth of cographs. *SIAM Journal on Discrete Mathematics*, 6(2):181–188, 1993.

[BvdW20] Hans L. Bodlaender and Marieke van der Wegen. Parameterized Complexity of Scheduling Chains of Jobs with Delays. In *15th Int. Symposium on Parameterized and Exact Computation (IPEC 2020)*, LIPIcs, pages 4:1–4:15, 2020.

[EdW14] Jan Elffers and Mathijs de Weerdt. Scheduling with two non-unit task lengths is np-complete. Technical report, KTH Royal Institute of Technology, 2014.

[EKMR17] Michael Etscheid, Stefan Kratsch, Matthias Mnich, and Heiko Röglin. Polynomial kernels for weighted problems. *Journal of Computer and System Sciences*, 84:1–10, 2017.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[FT87] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7:49–65, 1987.

[GHK+22] Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, and Yota Otachi. Exploring the gap between treedepth and vertex cover through vertex integrity. *Theoretical Computer Science*, 918:60–76, 2022.

[GJ79] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[GRR19] Frank Gurski, Carolin Rehs, and Jochen Rethmann. Knapsack problems: A parameterized point of view. *Theoretical Computer Science*, 775:93–108, 2019.

[HKPS21] Danny Hermelin, Shlomo Karhi, Michael Pinedo, and Dvir Shabtay. New algorithms for minimizing the weighted number of tardy jobs on a single machine. *Annals of Operations Research*, 298(1):271–287, 2021.

[HMK23] Claire Hanen and Alix Munier Kordon. Fixed-parameter tractability of scheduling dependent typed tasks subject to release times and deadlines. *Journal of Scheduling*, pages 1–15, 2023.

[HMMK22] Claire Hanen, Maher Mallem, and Alix Munier-Kordon. Parameterized complexity of single-machine scheduling with precedence, release dates and deadlines. In *15th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP)*, pages 131–134, 2022.

[HMO24] Danny Hermelin, Matthias Mnich, and Simon Omlor. Serial batching to minimize the weighted number of tardy jobs. *Journal of Scheduling*, 27(6):545–556, 2024.

[JKZ25] Klaus Jansen, Kai Kahler, and Esther Zwanger. Exact and approximate high-multiplicity scheduling on identical machines. In Irene Finocchi and Loukas Georgiadis, editors, *Algorithms and Complexity - 14th International Conference, CIAC 2025*, pages 1–17. Springer Nature Switzerland, 2025.

[Kas79] T. Kashiwabara. NP-completeness of the problem of finding a minimal-cliquenumber interval graph containing a given graph as a subgraph. *Proc. 1979 Int. Symp. Circuit Syst*, pages 657–660, 1979.

[KK22] Dusan Knop and Martin Koutecký. Scheduling kernels via configuration LP. In *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, LIPIcs, pages 73:1–73:15, 2022.

[KKL+19] Dusan Knop, Martin Koutecký, Asaf Levin, Matthias Mnich, and Shmuel Onn. Multitype integer monoid optimization and applications. *CoRR*, abs/1909.07326, 2019.

[KKM97] Dieter Kratsch, Ton Kloks, and Haiko Muller. Measuring the vulnerability for classes of intersection graphs. *Discrete Applied Mathematics*, 77(3):259–270, 1997.

[KMM24] Matthias Kaul, Matthias Mnich, and Hendrik Molter. Single-Machine Scheduling to Minimize the Number of Tardy Jobs with Release Dates. In *19th Int. Symposium on Parameterized and Exact Computation (IPEC 2024)*, LIPIcs, pages 19:1–19:15, 2024.

[KS96] Haim Kaplan and Ron Shamir. Pathwidth, bandwidth, and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25(3):540–561, 1996.

[KZ20] Martin Koutecký and Johannes Zink. Complexity of scheduling few types of jobs on related and unrelated machines. In *31st Int. Symposium on Algorithms and Computation, ISAAC 2020*, LIPIcs, pages 18:1–18:17, 2020.

[LQ92] Jean B. Lasserre and Maurice Queyranne. Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling. In *Proc. of the 2nd Integer Programming and Combinatorial Optimization Conf.*, pages 136–149, 1992.

[LRKB77] J.K. Lenstra, A.H.G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Ann. of Discrete Math.*, 1:343–362, 1977.

[MHMK24] Maher Mallem, Claire Hanen, and Alix Munier-Kordon. A new structural parameter on single machine scheduling with release dates and deadlines. In *Combinatorial Optimization*, pages 205–219. Springer Nature Switzerland, 2024.

[MK21] Alix Munier Kordon. A fixed-parameter algorithm for scheduling unit dependent tasks on parallel machines with time windows. *Discrete Applied Mathematics*, 290:1–6, 2021.

[MRR92] Madhav V Marathe, R Ravi, and C Pandu Rangan. Generalized vertex covering in interval graphs. *Discrete Applied Mathematics*, 39(1):87–93, 1992.

[MvB18] Matthias Mnich and René van Bevern. Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research*, 100:254–261, December 2018.

[MW15] Matthias Mnich and Andreas Wiese. Scheduling and fixed-parameter tractability. *Mathematical Programming*, 154(1-2):533–562, December 2015.

[vBBB+16] René van Bevern, Robert Bredereck, Laurent Bulteau, Christian Komusiewicz, Nimrod Talmon, and Gerhard J. Woeginger. Precedence-constrained scheduling problems parameterized by partial order width. In *Discrete Optimization and Operations Research*, pages 105–120. Springer Int. Publishing, 2016.

[vBNS17] René van Bevern, Rolf Niedermeier, and Ondřej Suchỳ. 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, 2017.