Fixed-parameter tractability of scheduling dependent typed tasks subject to release times and deadlines

Claire C. Hanen, Alix Munier Kordon

Journal of Scheduling manuscript No.
(will be inserted by the editor)

Fixed-Parameter Tractability of Scheduling Dependent Typed Tasks subject to Release Times and Deadlines

Claire Hanen · Alix Munier Kordon

Received: date / Accepted: date

Abstract Scheduling problems involving a set of dependent tasks with release dates and deadlines on a limited number of resources have been intensively studied. However, few parameterized complexity results exist for these problems.

This paper studies the existence of a feasible schedule for a typed task system with precedence constraints and time intervals (r_i, d_i) for each job i. The problem is denoted by $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$. Several parameters are considered: the pathwidth $pw(I)$ of the interval graph I associated to the time intervals (r_i, d_i), the maximum processing time of a task $p_{\max}$ and the maximum slack of a task $s\ell_{\max}$. This paper establishes that the problem is para-NP-complete with respect to any of these parameters. It then provides a fixed-parameter algorithm for the problem parameterized by both parameters $pw(I)$ and $\min(p_{\max}, s\ell_{\max})$. It is based on a dynamic programming approach that builds a levelled graph which longest paths represent all the feasible solutions. Fixed-parameter algorithms for the problems $P|\mathcal{M}_j(type), prec, r_i, d_i|C_{\max}$ and $P|\mathcal{M}_j(type), prec, r_i|L_{\max}$ are then derived using a binary search.

Keywords Fixed-Parameter Tractable Algorithm · Precedence constraints · Typed task system

C.Hanen
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
UPL, Université Paris Nanterre, F-92000 Nanterre, France
Fax: +123-45-678910
E-mail: Claire.Hanen@lip6.fr

A.Munier Kordon
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
E-mail: Alix.Munier@lip6.fr

1 Introduction

Scheduling with resources and precedence constraints has many fields of applications, like computer systems, wireless sensors and industrial production systems. This class of problems has been paid much attention since the early seventies with various settings. A large amount of scientific literature has been published on these problems, see for example the books [7,28] and the survey [11].

Problem definition We consider in this paper a set $\mathcal{T}$ of n tasks with integer processing times p_i, release times r_i and deadlines d_i for $i \in \mathcal{T}$. Tasks are linked by precedence relations given by an acyclic directed graph $G = (\mathcal{T}, A)$. We also consider a class of resource constraints introduced in 1978 by Liu and Liu [30] as typed task system: tasks are processed by machines (or processors) partitioned into k classes $\{C\ell_1, \ldots, C\ell_k\}$. Each class $C\ell_p$ for $p \in \{1, \ldots, k\}$ contains m_p identical processors. Each task $i \in \mathcal{T}$ requires one processor from a fixed class $\tau_i \in \{C\ell_1, \ldots, C\ell_k\}$ during its execution.

A **feasible schedule** σ is a function that defines for each task $i \in \mathcal{T}$ a completion time C_i and a processor π_i such that i is performed on π_i during the time interval $[C_i - p_i, C_i)$ and the following constraints hold:

- For each task $i \in \mathcal{T}$, $r_i + p_i \leq C_i \leq d_i$ (time window constraints);
- For each task $i \in \mathcal{T}$, $\pi_i \in \tau_i$ (typed task constraints);
- For each arc $(i, j) \in A$, $C_i \leq C_j - p_j$ (precedence constraints);
- Each processor performs at most one task at a time (resource constraints).

We first address the decision problem of checking the existence of a feasible schedule. Graham et al. standard notations [18] and their extension developed by Leung and Li [29] are used to describe the problems we consider. A star in the third field denotes a decision problem. The general decision problem tackled in this paper is denoted by $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$ and the related optimization problem by $P|\mathcal{M}_j(type), prec, r_i, d_i|C_{\max}$ where the makespan is defined by $C_{\max} = \max_{i \in \mathcal{T}} C_i$. When $(d_i)_{i \in \mathcal{T}}$ are due-dates instead of deadlines, the completion time of a job is allowed to violate the deadline constraint, and we can measure the maximum lateness as $L_{\max} = \max_{i \in \mathcal{T}} C_i - d_i$. The related optimization problem is denoted by $P|\mathcal{M}_j(type), prec, r_i|L_{\max}$.

The NP-hardness of $P|\mathcal{M}_j(type), prec|C_{\max}$ was proved by Jansen in [23]. Instances of our problem are quite general since they include both the parallel processors problem (only one class of machines) and the job-shop problem (one machine per class and chain like precedence constraints). These problems have been known for a long time to be NP-complete or NP-hard in the strong sense even with simple settings [17]. Many exact or approximation algorithms have been proposed in the literature [7,28].

Parameterized Algorithms and Complexity The algorithmic study of NP-hard optimization problems has been renewed since early 2000, with the development of fixed-parameter tractable algorithms (FPT algorithms in short) and

parameterized complexity analysis [12,14]. A fixed-parameter tractable algorithm solves any instance of size n of the problem with parameter k in a time $\mathcal{O}(f(k) \times \text{poly}(n))$, where f is allowed to be a computable superpolynomial function. Parameters k are chosen to capture some characteristics of the instances that can have small values in practice or that improve the understanding of what makes instances difficult to solve. For example, as detailed below, the width of the precedence graph has been considered by several authors for general scheduling problems with resources constraints. The W hierarchy allows to classify problems for which it is unlikely that a FPT algorithm exists for a given parameter [15]. It is included in the intersection of two parameterized complexity classes of problems para-NP and XP. A problem parameterized by k belongs to XP if it can be solved by an algorithm in time complexity $\mathcal{O}(\text{poly}(n)^{f(k)})$. A problem is in para-NP if there exists a non deterministic algorithm in time complexity $\mathcal{O}(f(k) \times n^{O(1)})$ that solves the problem. The design of FPT algorithms or the study of the parameterized complexity of general resource and precedence constrained scheduling problems is a challenging topic; the key point is to find appropriate parameters.

Related works Several dynamic programming algorithms have been designed in the eighties to solve scheduling problems with precedence and resource limitations. For example, Dolev and Warmuth [13] developed an exact algorithm solving $P|prec, p_i = 1|C_{\max}$ with time complexity $\mathcal{O}(n^{h(G)(m-1)+1})$; $h(G)$ is the length of the longest path of the precedence graph, n the number of tasks and m the number of parallel machines. For the same problem, Möhring [33] proposed an algorithm of time in $\mathcal{O}(m^{w(G)})$, where $w(G)$ is the width of the precedence graph. It can be deduced that $P|prec, p_i = 1|C_{\max}$ parametrized by the tuple $(h(G), m)$ or by the width $w(G)$ belongs to XP.

More recently, an algorithm by Gromicho et al. [19,22] solves the job-shop problem $J||C_{\max}$ in time $\mathcal{O}(p_{\max}^{2n} \times (m+1)^n)$, where $p_{\max}$ is the maximum processing time of an operation, n the number of jobs and m the number of machines. Although this is not an FPT algorithm, it shows that the problem $J||C_{\max}$ parameterized by n and $p_{\max}$ belongs to XP.

Until now, few fixed-parameter tractable algorithms have been provided for scheduling problems. A recent survey of Mnich and van Bevern [31] presents 15 basic open problems. Since then only few of them have been solved.

The maximum processing time $p_{\max}$ is a parameter suggested by previous works on related resource constrained scheduling problems. Mnich and Wiese [32] developed an FPT algorithm with parameter $p_{\max}$ for $P||C_{\max}$. Their approach was extended by Knop et al. [24] for $Q||C_{\max}$ and $R||C_{\max}$. Bessy and Giroudeau [3] proposed an FPT algorithm for a coupled task system with compatibility constraints, assuming bounded processing times, for which the vertex cover of the compatibility graph is the parameter.

Several studies considered the parameter $w(G)$, the width of the precedence graph, for scheduling problems with precedence constraints and parallel processors; they are mostly leading to W[1] or W[2]-completeness results. Günther et al. [20] proved that for two parallel processors and chain like precedence

constraints, $P2|chains, w(G) \leq 3|C_{\max}$ is weakly NP-hard. Bodlaender and Fellows [5] proved that the problem $P|prec, p_i = 1|C_{\max}$ is W[2]-hard parameterized by $w(G)$ and the number of machines. More recently, van Bevern et al. [37] proved that $P2|prec, p_j \in \{1, 2\}|C_{\max}$ is W[2]-hard parameterized by $w(G)$. This discards $w(G)$ and even the tuple of parameters $(w(G), p_{\max})$ to be valid ones of an FPT algorithm for our general problem. However, setting λ as the maximal allowed lag of a task computed from the earliest schedule defined by precedence constraints, van Bevern et al. [37] defined an FPT algorithm for the resource constrained scheduling problem (RCPSP) parameterized by the tuple $(w(G), \lambda)$.

Several recent papers considered new parameters based on the structure of time intervals. Bodlaender and van der Wegen [6] addressed the single machine scheduling of a set of chain like precedence constraints with delays between successive tasks, and each chain has its own interval (release time and deadline). They defined the thickness as the maximum number of overlapping intervals of chains and proved that the problem with minimum delays given in unary is W[2]-hard with respect to the thickness. Munier Kordon [34] developed an FPT algorithm for the decision problem $P|prec, p_i = 1, r_i, d_i|\star$. The parameter is the pathwidth $pw(I)$ of the interval graph I induced by the release times and the deadlines of the tasks corresponding roughly to the maximum number of overlapping intervals of jobs. This new parameter can be seen as a quite natural measure of the parallelism of an instance of this scheduling problem; it can easily be reduced by using preprocessing techniques of deadlines and release dates modifications [2,8,10,16,21,27]. By augmenting this previous FPT decision algorithm with a binary search, Munier Kordon [34] also proved that the two optimization problems $P|prec, p_i = 1, r_i|C_{\max}$ and $P|prec, p_i = 1, r_i|L_{\max}$ are FPT parameterized by $pw(I)$. Here, I is built from the tasks intervals defined using deadlines computed from an upper bound on the criteria ($C_{\max}$ or $L_{\max}$). The upper bound should be chosen as low as possible to avoid trivial high values of the parameter. This work was extended by Tang and Munier Kordon in [36] to $\bar{P}|prec, p_i = 1, c_{ij} = 1, r_i|C_{\max}$ considering communication delays on an unlimited number of processors.

Lastly, in parallel with the first submission of this paper, Baart et al. [1] proposed an FPT algorithm for a single machine scheduling with job intervals, set-up times and rejection and without precedence constraints. They considered two parameters: the maximal slack $s\ell_{\max} = \max_{i \in \mathcal{T}} d_i - p_i - r_i$ and the maximum number of overlapping time windows, which is equal to $pw(I) + 1$. So a challenging question is whether $pw(I)$ could be used as a parameter for typed task systems with non unit processing times.

Our contribution In this paper, we first prove that the problem $P2|r_i, d_i|\star$ parameterized by the pathwidth $pw(I)$ is para-NP-complete, and recall a complexity result established in [26] implying that the problem $P|prec, r_i, d_i|\star$ parameterized by $p_{\max}$ or $s\ell_{\max}$ is also para-NP-complete. It follows that unless FPT =para-NP. no fixed-parameter algorithm exists for the decision problem $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$ parameterized by only one parameter among the

pathwidth $pw(I)$, the maximum processing time $p_{\max}$ or the maximum slack of a task $s\ell_{\max}$.

We show in the rest of the paper that an FPT algorithm exists for this problem parameterized by the tuple $(pw(I), \rho)$ with $\rho = \min(p_{\max}, s\ell_{\max})$. An FPT algorithm parameterized by $(pw(I), \rho)$ can then be derived for the optimization problem $P|\mathcal{M}_j(type), prec, r_i, d_i|C_{\max}$. By considering the problem with due-dates for which the maximal slack is given $P|\mathcal{M}_j(type), prec, r_i, s\ell_{\max}|L_{\max}$, we provide an FPT algorithm parameterized by $(pw(I), \rho)$ where I is the interval graph induced by the set of intervals $\{(r_i, d_i + s\ell_{\max}), i \in \mathcal{T}\}$.

Let us define the strictly increasing sequence u_α, $\alpha \in \mathbb{N}^\star$ of the sorted endpoints of the intervals $\{(r_i, d_i), i \in \mathcal{T}\}$. The FPT algorithm parameterized by $pw(I)$ presented by Munier Kordon in [34] for unit processing time tasks considers any feasible schedule as a sequence of partial schedules of tasks in time intervals $\{[u_\alpha, u_{\alpha+1}], \alpha \in \mathbb{N}^\star\}$. Considering arbitrary processing times, such a decomposition must be refined, due to the presence of jobs that may be in progress at times u_α, $\alpha \in \mathbb{N}^\star$. We show that by adding a second parameter and enumerating the schedules of tasks in progress at each time instant u_α, we can again build a multistage graph $\mathcal{G}$ whose longest paths represent all feasible schedules (if any). We prove that building the graph $\mathcal{G}$ and checking the existence of a longest path is done in time $\mathcal{O}(f(pw(I), \rho) \times n^4)$.

This paper is organised as follows: Section 2 presents additional notations, a small example and complexity results. Section 3 is devoted to the description and the proof of several properties of feasible solutions. The construction of the multistage auxiliary graph $\mathcal{G}$ and the relation between its longest paths and feasible schedules are detailed in Section 4. A brute force algorithm for computing the multistage auxiliary graph $\mathcal{G}$ and its complexity are provided in Section 5. Section 6 is our conclusion.

2 Notations, parameters and complexity issues

This section first presents the parameters we consider and additional notations. We then provide a small example that will be used throughout the paper to illustrate our results. We eventually prove that the problem is para-NP-complete with respect to any of the parameters.

2.1 Notations and example

Consider an instance of the decision problem, with a set of tasks $\mathcal{T}$ and a precedence graph G. For any task $i \in \mathcal{T}$, the set of immediate successors and predecessors of i is respectively denoted by $\Gamma^+(i)$ and $\Gamma^-(i)$. For any tuple of tasks $(i, j) \in \mathcal{T}^2$, we note $i \to j$ if there is a path in G from i to j. We also denote by $\Gamma^{+\star}(i)$ the set of descendants of i, ie. the set of tasks $j \neq i$ such that $i \to j$. Similarly, $\Gamma^{-\star}(i)$ is the set of ancestors of i, ie. the set of tasks $j \neq i$ such that $j \to i$.

We consider throughout this paper the following example: $\mathcal{T}$ is composed by $n = 8$ tasks; the directed acyclic graph $G = (\mathcal{T}, A)$ is pictured in Figure 1. We also suppose that 3 processors are partitioned into 2 classes $C\ell_1$ and $C\ell_2$ with $m_1 = 1$ and $m_2 = 2$. Allocation of tasks, processing times, release dates and deadlines are given by Table 1.

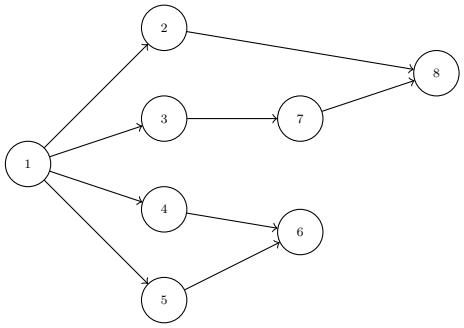

Fig. 1 A precedence graph $G = (\mathcal{T}, A)$ of 8 tasks.

$i \in \mathcal{T}$	1	2	3	4	5	6	7	8
p_i	1	4	2	1	3	5	1	1
τ_i	1	1	1	2	2	2	2	1
r_i	0	1	1	1	1	3	3	7
d_i	1	7	7	3	5	9	9	9

Table 1 Allocation of tasks, processing times, release dates and deadlines associated to the precedence graph presented by Figure 1. Processors are partitioned in two classes with $m_1 = 1$ and $m_2 = 2$.

2.2 Parameters

Several parameters are considered in this paper.

Definition 1 The maximum processing time and the maximum slack are respectively defined by $p_{\max} = \max_{i \in \mathcal{T}} p_i$ and $s\ell_{\max} = \max_{i \in \mathcal{T}} d_i - p_i - r_i$. We will use as a parameter the minimum of these two values: $\rho = \min(p_{\max}, s\ell_{\max})$.

Let us consider the strictly increasing sequence u_α, $\alpha \in \mathbb{N}^\star$ of the sorted endpoints of the intervals $\{(r_i, d_i), i \in \mathcal{T}\}$. Notice that there are at most $2n$ such endpoints. We assume that the first element is $u_1 = 0$; the last element u_κ of the sequences thus satisfies $\kappa \leq 2n$.

Definition 2 For any $\alpha \in \{1, \ldots, \kappa - 1\}$, we define X_α to be the set of tasks whose interval crosses $(u_\alpha, u_{\alpha+1})$:

$$X_\alpha = \{i \in \mathcal{T}, (u_\alpha, u_{\alpha+1}) \cap (r_i, d_i) \neq \emptyset\}.$$

We get then $\mathcal{T} = \bigcup_{\alpha=1}^{\kappa-1} X_\alpha$.

Definition 3 We define the **interval parameter** μ to be the maximum number of overlapping intervals: $\mu = \max_{\alpha \in \{1,\ldots,\kappa-1\}} |X_\alpha|$.

Baart et al. [1] designed this last parameter as the width of an instance, which seems confusing with the usual width $w(G)$ of a precedence graph.

Let us consider now the interval graph $I = (\mathcal{T}, E)$ built as follows:

1. Each vertex $i \in \mathcal{T}$ is associated to the open interval (r_i, d_i);
2. An edge $\{i, j\} \in E$ if $(r_i, d_i) \cap (r_j, d_j) \neq \emptyset$.

The pathwidth pw of a graph was initially introduced by Robertson and Seymour in [35]. In the case of interval graphs, $pw(I) = \max_{\alpha \in \{1,\ldots,\kappa-1\}} |X_\alpha| - 1$ [4], and thus $\mu = pw(I) + 1$.

Figure 2 presents the interval graph $I = (\mathcal{T}, E)$ for the release dates and deadlines of Table 1 and the intervals. We get the time sequence $u_1 = 0$,

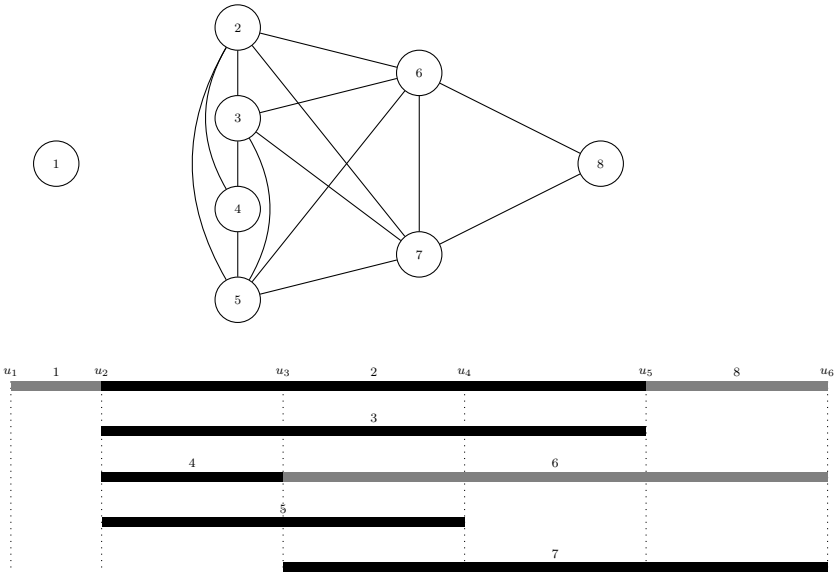

Fig. 2 The interval graph $I = (\mathcal{T}, E)$ associated with the graph $G = (\mathcal{T}, A)$ of Figure 1 and the corresponding intervals (r_i, d_i) for tasks $i \in \mathcal{T}$.

$u_2 = 1$, $u_3 = 3$, $u_4 = 5$, $u_5 = 7$ and $u_6 = 9$, thus $\kappa = 6$. The corresponding sets are $X_1 = \{1\}$, $X_2 = \{2, 3, 4, 5\}$, $X_3 = \{2, 3, 5, 6, 7\}$, $X_4 = \{2, 3, 6, 7\}$ and $X_5 = \{6, 7, 8\}$. Thus the interval parameter is $\mu = \max_{\alpha=1\ldots5} |X_\alpha| = 5$, and the pathwidth $pw(I) = 4$.

2.3 Complexity issues

The decision problem $P2|r_i, d_i|\star$ has been proved to be NP-complete by Lenstra et al. in [25] using a reduction from KNAPSACK. However, the pathwidth of the scheduling problem built by this reduction is not bounded. We prove in the following that another reduction is possible with a pathwidth bounded by 3.

Let us consider the well-known problem PARTITION and its special case PARTITION-SC defined as follows. Garey and Johnson [17] claimed without any argument that PARTITION-SC is NP-complete. Lemma 1 provides a proof of this result.

PARTITION-SC
Input: $n = 2p$ positive integer values $a_1, a_2, \ldots, a_n$ such that, for any value $j \in \{1, \ldots, p\}$, $a_{2j-1} < a_{2j}$.
Question: is there a subset $A \subset \{1, \ldots, n\}$ such that, for any value $j \in \{1, \ldots, p\}$, exactly one value from $\{2j-1, 2j\}$ belongs to A and $\sum_{u \in A} a_u = \sum_{u \in \{1,\ldots n\}-A} a_u$?

PARTITION
Input: p positive integer values $b_1, b_2 \ldots, b_p$.
Question: is there a subset $B \subset \{1, \ldots, p\}$ such that the value $\sum_{j \in B} b_j = \sum_{j \in \{1,\ldots,p\}-B} b_j$?

Lemma 1 *The decision problem* PARTITION-SC *is* NP*-complete.*

Proof PARTITION-SC belongs to NP. Let us consider an instance $\mathcal{I}$ of PARTITION and its associated instance $f(\mathcal{I})$ of PARTITION-SC defined as $a_{2j-1} = 1$ and $a_{2j} = 1 + b_j$ for every $j \in \{1, \ldots p\}$.

Now, let us define the relation between A and B in our reduction : $j \in B$ if and only if $2j \in A$. Then, we observe that for such subsets, $\sum_{u \in A} a_u = \sum_{j \in B} a_{2j} + \sum_{j \in \{1,\ldots,p\}-B} a_{2j-1} = p + \sum_{j \in B} b_j$. Similarly, $\sum_{u \in \{1,\ldots,n\}-A} a_u = \sum_{j \in \{1,\ldots,p\}-B} a_{2j} + \sum_{j \in B} a_{2j-1} = p + \sum_{j \in \{1,\ldots,p\}-B} b_j$.

Let us suppose first that the answer to the instance $\mathcal{I}$ of PARTITION is "yes", and let B be the solution set. Then, $\sum_{j \in B} b_j = \sum_{j \in \{1,\ldots,p\}-B} b_j$. The associated set A thus verifies $\sum_{u \in A} a_u = \sum_{u \in \{1,\ldots,n\}-A} a_u$;

Now, let us suppose that the answer the instance $f(\mathcal{I})$ of PARTITION-SC is "yes", then by definition of A, for any value $j \in \{1, \ldots, p\}$, $2j \in A$ or $2j-1 \in A$, but not both. Thus, B is well defined and since $\sum_{u \in A} a_u = \sum_{u \in \{1,\ldots,n\}-A} a_u$, $\sum_{j \in B} b_j = \sum_{j \in \{1,\ldots,p\}-B} b_j$, which concludes the proof. □

Theorem 1 *The decision problem* $P2|r_i, d_i|\star$ *with a pathwidth bounded by* 3 *is* NP*-complete.*

Proof This scheduling problem clearly belongs to NP. Below, we build a polynomial-time mapping from PARTITION-SC to an instance of $P2|r_i, d_i|\star$ with a pathwidth bounded by 3.

We set $2B = \sum_{u=1}^{n} a_u$. The instance of $P2|r_i, d_i|\star$ is built as follows:

1. We define the set of tasks to be $\mathcal{T} = \{1, \ldots, n\}$; the processing time of each task $i \in \mathcal{T}$ is $p_i = a_i + 2B$; these jobs are to be scheduled on two identical processors;
2. We can assume that the tasks are performed without interruption on each processor in increasing order of indices. We set r_i to be a lower bound on the starting time of i assuming that for each value $j \in \{1, \ldots, p\}$, tasks $2j-1$ and $2j$ are performed by different processors. Observe that in any such schedule, tasks $2j-1$ and $2j$ cannot start before the end of the $j-1$ first odd indices jobs. Thus, we set the release times $r_1 = r_2 = 0$, and for any value $j \in \{2, \ldots, p\}$, $r_{2j-1} = r_{2j} = \sum_{k=1}^{j-1}(a_{2k-1} + 2B)$;
3. Similarly we set the deadlines to be an upper bound of the completion times of tasks in a schedule without any idle slots, and such that on each machine tasks are performed by increasing order. If the tasks are executed without any idle slot, the length of the schedule equals

$$d_{2p} = d_{2p-1} = \frac{\sum_{i=1}^{n} p_i}{2} = \frac{\sum_{i=1}^{n}(a_i + 2B)}{2} = \frac{4pB + 2B}{2} = 2pB + B.$$

   On another hand, executing sequentially tasks $2, 4, 6, \ldots 2p$ on the same processor gives an upper bound of the completion times of the tasks. We then define, for any value $j \in \{1, \ldots, p-1\}$, $d_{2j-1} = d_{2j} = \sum_{k=1}^{j}(a_{2k}+2B)$. Notice that $d_{2(p-1)} = 2(p-1)B + \sum_{k=1}^{p-1} a_{2k} \leq 2pB < d_{2p}$;
4. The question is the existence of a feasible schedule on two processors that fulfills release times and deadlines.

We observe that the pathwidth of the scheduling problem equals 3. Indeed, for each value $j \in \{1, \ldots, p-2\}$, we get $d_{2j} = \sum_{k=1}^{j}(a_{2k} + 2B) = 2jB + \sum_{k=1}^{j} a_{2k}$ and $r_{2j+4} = \sum_{k=1}^{j+1}(a_{2k-1} + 2B) = 2(j+1)B + \sum_{k=1}^{j+1} a_{2k-1}$. Now, $r_{2(j+2)} - d_{2j} = 2B + \sum_{k=1}^{j+1} a_{2k-1} - \sum_{k=1}^{j} a_{2k}$. As $\sum_{k=1}^{j+1} a_{2k-1} - \sum_{k=1}^{j} a_{2k} \geq -2B$, we obtain $r_{2(j+2)} - d_{2j} \geq 0$. Thus, $(r_{2j}, d_{2j}) \cap (r_{2j+4}, d_{2j+4}) = \emptyset$ and the maximum number of overlapping intervals equals 4.

Let us first suppose that the answer to our instance of Partition-SC is "yes", and let A be the solution set. For any integer $i \in A$, we assign the task i to the first processors; otherwise, $i \notin A$ and the task i is assigned to the second processor. On each processor, tasks are then performed without interruption following the increasing indices. Since $\sum_{u \in A} a_u = \sum_{u \in \{1,\ldots,n\}-A} a_u = B$ and by definition of release times and deadlines, this schedule is feasible.

Assume now that a feasible schedule exists for the corresponding instance of $P2|r_i, d_i|\star$. We first show that the size of the intervals $[r_i, d_i]$ of any task i is bounded by $4B$. As for any value $j \in \{1, \ldots, p-1\}$, $p_{2j-1} + p_{2j} > 4B$, jobs $2j-1$ and $2j$ cannot be processed on the same machine. Consider first $j \in \{1, \ldots, p-1\}$; $d_{2j} - r_{2j} = \sum_{k=1}^{j}(a_{2k} + 2B) - \sum_{k=1}^{j-1}(a_{2k-1} + 2B) = 2B + \sum_{k=1}^{j} a_{2k} - \sum_{k=1}^{j-1} a_{2k-1} \leq 4B$. Now for $j = p$, we have $d_{2p} - r_{2p} = 2pB + B - \sum_{k=1}^{p-1}(a_{2k-1} + 2B) = (2p+1)B - \sum_{k=1}^{p-1} a_{2k-1} - 2B(p-1) \leq 3B$. Thus, for any value $j \in \{1, \ldots, p\}$, tasks $2j-1$ and $2j$ are necessarily assigned to different processors, and each processor executes exactly p tasks. Moreover,

since the last deadline d_{2p} is fulfilled, tasks are executed without interruption on each processor.

Thus, by setting A to be the set of indexes $i \in \{1, \ldots, n\}$ such that the task i is assigned to the first processor, we get $\sum_{i \in A} a_i = \sum_{i \in A}(p_i - 2B) = \sum_{i \in A} p_i - 2|A|B$. Now, since $|A| = p$, we get $\sum_{i \in A} a_i = 2pB + B - 2pB = B$ and thus $\sum_{i \in A} a_i = \sum_{i \in \{1,\ldots n\} - A} a_i = B$. The answer to our instance of PARTITION-SC is "yes", which achieves the proof. □

Lenstra and Rinnooy Kan proved in [26] that the problem $P|prec, p_j = 1|C_{\max} \leq 3$ is NP-complete. For any instance of this problem, the maximum processing time equals 1 and the maximum slack is bounded by 2. The following theorem uses the definition of para-NP-completness and the property stated in Theorem 2.14 of Flum and Grohe [15] applied to this last result. It also expresses a corollary of the previous theorem:

Corollary 1 *The scheduling problem $P2|r_i, d_i|\star$ parameterized by the path-width is para-NP-complete. The problem scheduling $P|prec, r_i, d_i|\star$ parameterized by either $p_{\max}$ or $s\ell_{\max}$ is para-NP-complete.*

It follows that it is unlikely that a FPT algorithm can be defined for our problem considering only one of the parameters $pw(I)$, $p_{\max}$ or $s\ell_{\max}$, unless FPT =para-NP.

3 Basic properties of feasible solutions

This section presents some basic properties of feasible schedules. Let us assume that σ is a feasible solution of an instance of our scheduling problem $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$. We suppose that processors are indexed from 1 to m. The schedule σ is then given by the completion times $C_i \in \{r_i + p_i, \ldots, d_i\}$ and the processors $\pi_i \in \tau_i$ associated to any task $i \in \mathcal{T}$.

For any $\alpha \in \{1, \ldots, \kappa - 1\}$, we consider the time interval $[u_\alpha, u_{\alpha+1}]$. A task i crosses the time instant t if $C_i - p_i < t$ and $C_i > t$. Let us consider then the following sequence of task sets:

- For any $\alpha \in \{1, \ldots, \kappa\}$, $B_\alpha = \{i \in \mathcal{T}, C_i - p_i < u_\alpha \text{ and } C_i > u_\alpha\}$ is the set of tasks that cross the instant u_α;
- For any $\alpha \in \{1, \ldots, \kappa - 1\}$, $In_\alpha = \{i \in \mathcal{T}, u_\alpha \leq C_i - p_i \text{ and } C_i \leq u_{\alpha+1}\}$ is the set of tasks that are completely performed during $[u_\alpha, u_{\alpha+1}]$.

Any task $i \in \mathcal{T}$ belongs to at least one of these sets. However, these sets do not form a partition since a task may belong to several successive sets B_α.

For any $\alpha \in \{1, \ldots, \kappa - 1\}$, let us define R_α as the set of tasks that are partially or totally performed during the interval $[u_\alpha, u_{\alpha+1}]$. More formally, $R_\alpha = \{i \in \mathcal{T}, C_i - p_i < u_{\alpha+1} \text{ and } C_i > u_\alpha\}$. We observe that $R_\alpha = B_\alpha \cup In_\alpha \cup B_{\alpha+1}$ (see Figure 3).

For each $\alpha \in \{1, \ldots, \kappa\}$, the schedule of the tasks of B_α is described by a vector of tuples M_α such that, for any value $\ell \in \{1, \ldots, m\}$, $M_\alpha[\ell] = (i, C_i)$ if

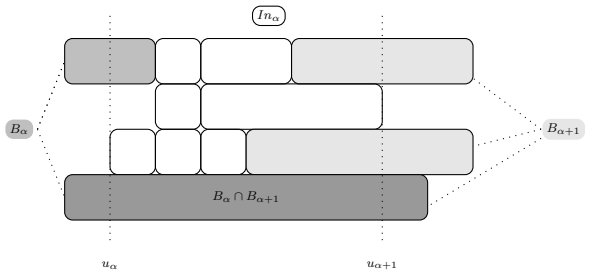

Fig. 3 A set $R_\alpha = B_\alpha \cup In_\alpha \cup B_{\alpha+1}$ for $\alpha \in \{1, \dots \kappa - 1\}$.

the task $i \in B_\alpha$ is mapped to processor ℓ (ie. $\pi_i = \ell$) and is completed at time C_i. Otherwise, if no task crosses u_α on processor ℓ, we set $M_\alpha[\ell] = \bullet$. M_α is called a **border schedule**.

We denote by $M_\bullet$ the empty border schedule, ie. for any $\ell \in \{1, \dots, m\}$, $M_\bullet[\ell] = \bullet$. Observe that $B_\kappa = \emptyset$ and thus $M_\kappa = M_\bullet$.

Let us consider the feasible schedule σ depicted in Figure 4 for the instance described in Figure 1. Table 2 shows the associated sets B_α for $\alpha \in \{1, \dots, 6\}$ and the corresponding border schedules M_α for the feasible schedule presented by Figure 4. For example, for $\alpha = 4$, the tasks 2 and 6 are crossing u_4 and are mapped respectively on processors 1 and 3 with $C_2 = 7$ and $C_6 = 9$, while processor 2 is empty. Thus, $B_4 = \{2, 6\}$ and $M_4 = ((2, 7), \bullet, (6, 9))$. Table 3 presents corresponding sets In_α and R_α for each $\alpha \in \{1, \dots, 5\}$. For example, for $\alpha = 3$, $R_3 = B_3 \cup In_3 \cup B_4 = \{5\} \cup \{7\} \cup \{2, 6\} = \{2, 5, 6, 7\}$.

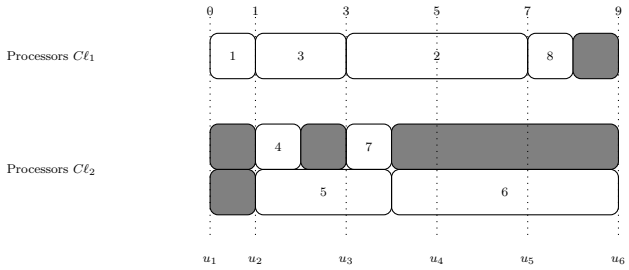

Fig. 4 A feasible schedule σ associated to the example presented by Figures 1 and 2.

α	1	2	3	4	5	6
B_α	$\emptyset$	$\emptyset$	$\{5\}$	$\{2, 6\}$	$\{6\}$	$\emptyset$
M_α	$M_\bullet$	$M_\bullet$	$(\bullet, \bullet, (5, 4))$	$((2, 7), \bullet, (6, 9))$	$(\bullet, \bullet, (6, 9))$	$M_\bullet$

Table 2 Sets B_α for $\alpha \in \{1, \dots, 6\}$ and the corresponding border schedules M_α for the feasible schedule shown by Figure 4.

α	1	2	3	4	5
In_α	$\{1\}$	$\{3,4\}$	$\{7\}$	$\emptyset$	$\{8\}$
R_α	$\{1\}$	$\{3,4,5\}$	$\{2,5,6,7\}$	$\{2,6\}$	$\{6,8\}$

Table 3 Sets In_α and R_α with $\alpha \in \{1,\ldots,5\}$ for the feasible schedule presented by Figure 4.

Lemma 2 *For each value $\alpha \in \{1,\ldots,\kappa-1\}$, $|R_\alpha| \leq \mu$.*

Proof Any task from R_α is partially or totally performed during $(u_\alpha, u_{\alpha+1})$, thus it belongs to X_α and $R_\alpha \subseteq X_\alpha$. By definition of the interval parameter, $\mu \geq |X_\alpha|$, so the inequality holds. □

Definition 4 A set of tasks $Y \subseteq \mathcal{T}$ is **consistent** with respect to the precedence graph G if, for any tuple $(i,j) \in Y^2$ with $i \neq j$ and $i \to j$, any task k on a path from i to j belongs to Y.

Definition 5 A set $Y \subseteq \mathcal{T}$ is **strictly consistent** if, for any $i \in Y$, $\Gamma^+(i) \subseteq Y$.

Lemma 3 *For any $\alpha \in \{1,\ldots,\kappa-1\}$, R_α is consistent.*

Proof Assume by contradiction that R_α is not consistent. Then, there exists a tuple $(i,j) \in R_\alpha^2$ with a task $k \in \mathcal{T} - R_\alpha$ such that $i \to k$ and $k \to j$. Thus, there exists $\gamma \in \{1,\ldots,\kappa-1\}$ with $\gamma \neq \alpha$ and $k \in R_\gamma$.

- If $\gamma < \alpha$, then $C_k - p_k < u_{\gamma+1} \leq u_\alpha$. Since $C_i > u_\alpha$, the precedence constraint due to the path $i \to k$ does not hold;
- Assume now that $\gamma > \alpha$, then $C_k > u_\gamma \geq u_{\alpha+1}$. Since $C_j - p_j < u_\alpha$, the precedence constraint due to the path $k \to j$ does not hold.

Thus, the lemma holds. □

Lemma 4 *For any value $\alpha \in \{1,\ldots,\kappa-1\}$, $A_\alpha = \cup_{\beta=\alpha}^{\kappa-1} R_\beta$ is strictly consistent.*

Proof By contradiction, assume that there exists $\alpha \in \{1,\ldots,\kappa-1\}$ such that $A_\alpha = \cup_{\beta=\alpha}^{\kappa-1} R_\beta$ is not strictly consistent. Consider $i \in A_\alpha$ with $\Gamma^+(i) \not\subseteq A_\alpha$. Thus, there exists a task $j \in \Gamma^+(i)$ with $j \notin A_\alpha$. Then, there exists $\gamma \in \{1,\ldots,\alpha-1\}$ with $j \in R_\gamma$ and $C_j - p_j < u_{\gamma+1} \leq u_\alpha < C_i$. The schedule σ does not fulfill the precedence constraint $(i,j) \in A$, a contradiction. □

Lemma 5 *For any $\alpha \in \{1,\ldots,\kappa\}$, we define $Z_\alpha = \{i \in \mathcal{T}, r_i \geq u_\alpha\}$. If σ is a feasible schedule, $A_\alpha = \cup_{\beta=\alpha}^{\kappa-1} R_\beta$ can be partitioned such that $A_\alpha = Z_\alpha \cup p(A_\alpha)$ with $p(A_\alpha) \subseteq X_\alpha \cap X_{\alpha-1}$.*

Proof Any element $i \in Z_\alpha$ verifies $r_i \geq u_\alpha$. Since σ is feasible, $C_i - p_i \geq r_i$, so $C_i - p_i \geq u_\alpha$ and thus $i \in A_\alpha$. We conclude that $Z_\alpha \subseteq A_\alpha$.

Now, consider $i \in p(A_\alpha) = A_\alpha - Z_\alpha$. Since $i \in A_\alpha$, there exists $\gamma \in \{\alpha,\ldots,\kappa-1\}$ with $i \in R_\gamma$. Thus, $C_i > u_\gamma$ and then $d_i > u_\gamma \geq u_\alpha$. Now, $i \notin Z_\alpha$, thus $r_i < u_\alpha$. It follows that $r_i < u_\alpha < d_i$, thus $i \in X_\alpha \cap X_{\alpha-1}$, which concludes the proof. □

For our example shown by Figure 1 and Table 1, we get $Z_1 = \mathcal{T}$, $Z_2 = \{2,3,4,5,6,7,8\}$, $Z_3 = \{6,7,8\}$, $Z_4 = \{8\}$, $Z_5 = \{8\}$ and $Z_6 = \emptyset$. Now, let us consider the feasible schedule σ depicted in Figure 4. For $\alpha = 3$, $A_3 = \cup_{\beta=3}^{5} R_\beta = \{2,5,6,7,8\} = Z_3 \cup \{2,5\}$ and $X_3 \cap X_2 - Z_3 = \{2,3,5\} - \{6,7,8\} = \{2,3,5\}$ with $\{2,5\} \subseteq \{2,3,5\}$.

4 Description and validity of the multistage auxiliary graph $\mathcal{G}$

We prove in this section that the properties shown above allow us to build a multistage auxiliary graph $\mathcal{G}$ whose paths from the source to the sink describe all feasible schedules. Section 4.1 presents the construction of $\mathcal{G}$. Section 4.2 proves that paths from the source to the sink of $\mathcal{G}$ represent feasible schedules.

4.1 Description of the multistage auxiliary graph $\mathcal{G}$

Nodes of $\mathcal{G}$

The set of nodes $\mathcal{V}$ of $\mathcal{G}$ is partitioned into κ sets $\mathcal{V}_1, \ldots, \mathcal{V}_\kappa$. For any value $\alpha \in \{1, \ldots, \kappa\}$, each node $v \in \mathcal{V}_\alpha$ is a tuple (V, M) where V is a set of tasks and M is a border schedule.

Each node $v = (V, M) \in \mathcal{V}_\alpha$ is associated with the existence of a feasible partial schedule σ of tasks of $V \cup Z_\alpha$ such that all of them are completed after time u_α (ie. $C_i > u_\alpha$). Moreover, σ matches with the border schedule M, ie. if $(i, C_i) \in M$ then $\sigma(i) + p_i = C_i$ and all tasks in $V \cup Z_\alpha$ that are not scheduled in M start not earlier than u_α. Thus, we obtain the following conditions:

1. $\mathcal{V}_1 = \mathcal{V}_\kappa = \{(\emptyset, M_\bullet)\}$;
2. For any integer α with $1 < \alpha < \kappa$, a node $v = (V, M) \in \mathcal{V}_\alpha$ satisfies:
   Inclusion: V is a subset of tasks from $X_\alpha \cap X_{\alpha-1}$;
   Consistency: $V \cup Z_\alpha$ is strictly consistent;
   Border feasibility: M is a feasible border schedule composed by at most m tasks from V such that their execution crosses u_α and fulfills resource constraints, release dates and deadlines. Moreover, tasks from M are independent (ie. there is no precedence constraint between them).

Arcs of $\mathcal{G}$

Let us consider two vertices $v = (V, M)$ and $v' = (V', M')$, with $v \in \mathcal{V}_\alpha$ and $v' \in \mathcal{V}_{\alpha+1}$ for $\alpha \in \{1, \ldots, \kappa - 1\}$.

If the arc (v, v') belongs to $\mathcal{A}$, there exists a feasible partial schedule of tasks from $V \cup Z_\alpha$ that satisfies the requirements of both nodes. The two border schedules M and M' match together. Moreover, the set of jobs that are supposed to start after u_α and to complete not later than $u_{\alpha+1}$ must be scheduled in the interval $[u_\alpha, u_{\alpha+1})$.

More formally, there is an arc (v, v') in $\mathcal{A}$ if the following properties hold:

Inclusion: The set of tasks $V' \cup Z_{\alpha+1} \subseteq V \cup Z_\alpha$;

Compatibility of the border schedules: Let B (resp. B') be the set of tasks scheduled by M (resp. M'). For each $\ell \in \{1, \ldots, m\}$, the following properties hold:

- If $M[\ell] = (i, C_i)$, two cases must be considered:
  1. If $C_i > u_{\alpha+1}$, then $i \in B'$ with $M'[\ell] = (i, C_i)$;
  2. Else, $i \notin V'$. Then, $M'[\ell] = \bullet$ or there exists a task $j \in B'$ with $M'[\ell] = (j, C_j)$ with $C_i \leq C_j - p_j$.
- Else, $M[\ell] = \bullet$. Two cases may occur:
  1. There exists a task $j \in B'$ with $M'[\ell] = (j, C_j)$ and $C_j - p_j \geq u_\alpha$;
  2. Otherwise, $M'[\ell] = \bullet$.

Lastly, if $i \to j$ with $i \in B$ and $j \in B'$, then $C_i \leq C_j - p_j$.

Existence of a local schedule: Consider the set of tasks $In = (V \cup Z_\alpha) - (V' \cup Z_{\alpha+1}) - B$ corresponding to the set of tasks that must be completely executed during the interval $[u_\alpha, u_{\alpha+1}]$.

- The set $R = In \cup B \cup B'$ must be consistent;
- There exists a feasible schedule of tasks from In satisfying precedence and resource constraints. This schedule must fulfill the constraints induced by the two border schedules M and M' of tasks from respectively B and B', so that a schedule of R exists.

Figure 5 shows the multistage auxiliary graph associated to the example presented by Figure 1 restricted to the nodes that belong to a longest path. For example, let us consider the node v_{42} corresponding to the tuple $(V, M) = (\{3, 6, 7\}, (\bullet, (6, 9), \bullet))$. It is associated with the set of tasks $V \cup Z_4 = \{3, 6, 7, 8\}$. Similarly, the node v_{52} corresponds to $(V', M') = (\{6, 7\}, (\bullet, (6, 9), \bullet))$ and is associated with the set of tasks $V' \cup Z_5 = \{6, 7, 8\}$.

We can check that arc (v_{42}, v_{52}) exists. Indeed, $V' \cup Z_5 \subseteq V \cup Z_4$, thus the inclusion constraint holds. Moreover, for the task 6, $M[2] = (6, 9) = M'[2]$, $C_6 = 9$ and $C_6 - p_6 = 9 - 5 = 4$ and then $C_6 - p_6 < u_4 < u_5 < 9$, thus the two border schedules are compatible. Lastly, $In = \{3\}$ and $R = \{3, 6\}$. Setting $C_3 = u_5 = 7$ leads to a feasible schedule of the two tasks 3 and 6, thus a local schedule exists.

Lemma 6 *For any arc $(v, v') \in \mathcal{A}$ with $v = (V, M) \in \mathcal{V}_\alpha$ and $v' = (V', M') \in \mathcal{V}_{\alpha+1}$ for $\alpha \in \{1, \ldots, \kappa - 1\}$, we get $V \cup Z_\alpha - (V' \cup Z_{\alpha+1}) \subseteq X_\alpha$.*

Proof Let us consider a task $i \in V \cup Z_\alpha - (V' \cup Z_{\alpha+1})$.

- If $i \in V$, then by the definition of V, $i \in X_\alpha \cap X_{\alpha-1} \subseteq X_\alpha$;
- Assume now that $i \in Z_\alpha$, then $r_i \geq u_\alpha$. Moreover, since $i \notin Z_{\alpha+1}$, $r_i < u_{\alpha+1}$, thus $r_i = u_\alpha$. As $d_i > r_i$, we get $d_i \geq u_{\alpha+1}$ and $i \in X_\alpha$.

□

4.2 Relation between longest paths of $\mathcal{G}$ and feasible schedules

We will prove in this subsection that the paths of length κ of $\mathcal{G}$ correspond to feasible schedules.

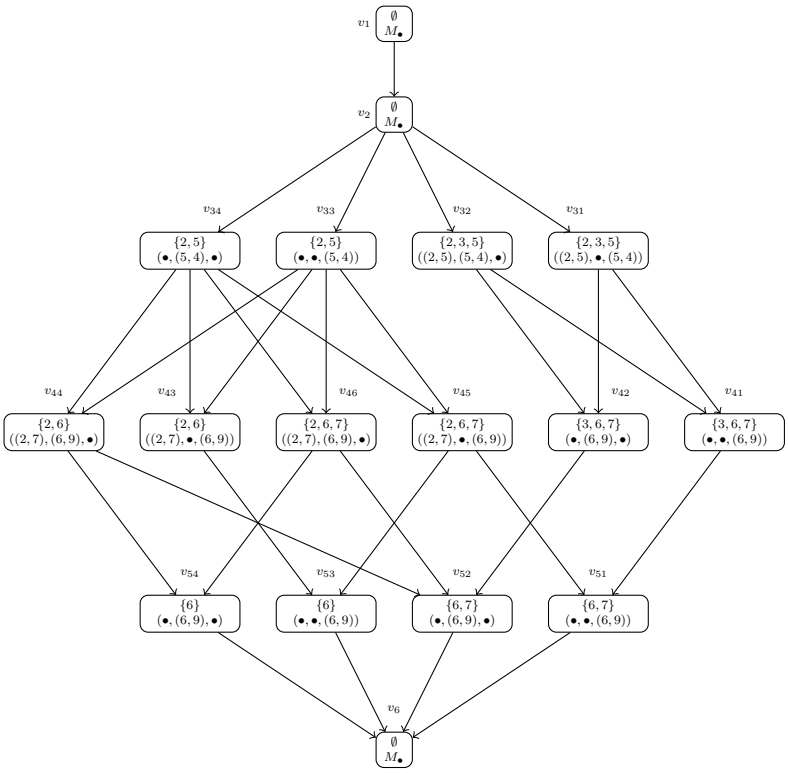

Fig. 5 The multistage auxiliary graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ associated with the example presented by Figure 1 and Figure 2. We get $\mathcal{V} = \cup_{\alpha=1}^{6}\mathcal{V}_\alpha$ with $\mathcal{V}_1 = \{v_1\}$, $\mathcal{V}_2 = \{v_2\}$, $\mathcal{V}_3 = \{v_{31}, v_{32}, v_{33}, v_{34}\}$, $\mathcal{V}_4 = \{v_{41}, v_{42}, v_{43}, v_{44}, v_{45}, v_{46}\}$, $\mathcal{V}_5 = \{v_{51}, v_{52}, v_{53}, v_{54}\}$ and $\mathcal{V}_6 = \{v_6\}$.

Let us first assume that σ is a feasible schedule. We note $V_\kappa = \emptyset$ and $V_\alpha = \bigcup_{\beta=\alpha}^{\kappa-1} R_\beta - Z_\alpha$ for $\alpha \in \{1, \ldots, \kappa-1\}$, according to the definitions of Section 3. For any $\alpha \in \{1, \ldots, \kappa\}$, we set $v_\alpha = (V_\alpha, M_\alpha)$.

Lemma 7 *For each value* $\alpha \in \{1, \ldots, \kappa\}$, $v_\alpha \in \mathcal{V}_\alpha$.

Proof By definition of $v_\kappa = (\emptyset, M_\bullet)$, $v_\kappa \in \mathcal{V}_\kappa$. Now, let us consider v_α for each $\alpha \in \{1, \ldots, \kappa-1\}$.

Inclusion: By Lemma 5, $V_\alpha = p(\bigcup_{\beta=\alpha}^{\kappa-1} R_\beta) \subseteq X_\alpha \cap X_{\alpha-1}$;
Consistency: By Lemma 4, $V_\alpha \cup Z_\alpha$ is strictly consistent;
Feasibility of Border: Lastly, by definition, $B_\alpha \subseteq R_\alpha$, thus $B_\alpha \subseteq V_\alpha \cup Z_\alpha$.

Tasks from M_α are exactly those from B_α, thus the last property is fulfilled.

We conclude that $v_\alpha \in \mathcal{V}_\alpha$ for each $\alpha \in \{1, \ldots, \kappa\}$. □

Lemma 8 *For each* $\alpha \in \{1, \ldots, \kappa-1\}$, $(v_\alpha, v_{\alpha+1}) \in \mathcal{A}$.

Proof By Lemma 7, nodes v_α with $\alpha \in \{1, \ldots, \kappa\}$ belong to $\mathcal{V}$.

Inclusion: Clearly, $V_\kappa \cup Z_\kappa = \emptyset \subseteq V_{\kappa-1} \cup Z_{\kappa-1}$. Now, for any $\alpha \in \{1, \ldots, \kappa-2\}$, $V_{\alpha+1} \cup Z_{\alpha+1} = \bigcup_{\beta=\alpha+1}^{\kappa-1} R_\beta \subseteq \bigcup_{\beta=\alpha}^{\kappa-1} R_\beta = V_\alpha \cup Z_\alpha$;

Compatibility of border schedules: Since σ is a feasible schedule, successive vectors M_α and $M_{\alpha+1}$ are consistent with respect to resource, precedence constraints and duration of tasks;

Existence of a local schedule: Now, for any $\alpha \in \{1, \ldots, \kappa-1\}$, $(V_\alpha \cup Z_\alpha) - (V_{\alpha+1} \cup Z_{\alpha+1}) - B_\alpha = In_\alpha$. By Lemma 3, $R_\alpha = In_\alpha \cup B_\alpha \cup B_{\alpha+1}$ is consistent. Moreover, tasks from In_α are schedulable between the two schedules M_α and $M_{\alpha+1}$, which concludes the proof.

□

The outcome of these two lemmas is that any feasible schedule is associated to a path of $\mathcal{G}$ from a node of $\mathcal{V}_1$ to a node of $\mathcal{V}_\kappa$. For example, let us consider the schedule presented in Figure 4. We observe that the associated path in the graph of Figure 5 is $(v_1, v_2, v_{33}, v_{43}, v_{53}, v_6)$.

Indeed, for the first node, we observe that all the tasks are performed during $[u_1, u_6] = [0, 9]$ and that $Z_1 = \mathcal{T}$. Moreover, no task crosses time u_1, the corresponding node is then $(\emptyset, M_\bullet) = v_1$. Now, for example, for $\alpha = 3$, the set of tasks performed partially or completely during the interval $[u_3, u_6] = [3, 9]$ is $\{2, 5, 6, 7, 8\} = Z_3 \cup \{2, 5\}$. Task 5 crosses u_3 and completes at time 4; the corresponding node is then $(\{2, 5\}, (\bullet, \bullet, (5, 4))) = v_{33}$. Lastly, for $\alpha = 6$, no task is performed strictly after u_6, thus the associated node is $v_6 = (\emptyset, M_\bullet)$.

Conversely, we show that any path of $\mathcal{G}$ from a node of $\mathcal{V}_1$ to a node of $\mathcal{V}_\kappa$ can be associated to a feasible schedule σ. Let $v(1), \ldots, v(\kappa)$ be such a path. For each $\alpha \in \{1, \ldots, \kappa\}$, we set $v(\alpha) = (V(\alpha), M(\alpha))$. Let us also consider the following sets of tasks:

- For each $\alpha \in \{1, \ldots, \kappa\}$, let $B(\alpha) = \{i \in \mathcal{T}, \exists \ell \in \{1, \ldots, m\}, M(\alpha)[\ell] = (i, C_i)\}$ be the set of tasks scheduled in $M(\alpha)$;
- For each $\alpha \in \{1, \ldots, \kappa-1\}$, we also set $In(\alpha) = (V(\alpha) \cup Z_\alpha) - (V(\alpha+1) \cup Z_{\alpha+1}) - B(\alpha)$;
- Lastly, for each $\alpha \in \{1, \ldots, \kappa-1\}$, $R(\alpha) = In(\alpha) \cup B(\alpha) \cup B(\alpha+1)$. Let us recall that according to the definition of the arcs $\mathcal{A}$, there exists a schedule $\sigma^{(\alpha)}$ of $R(\alpha)$, that meets the requirements of the precedence, release times, deadlines and resource constraints restricted to $R(\alpha)$. Moreover, this schedule is compatible with the border schedules of $M(\alpha)$ and $M(\alpha+1)$.

Let us build a schedule σ as follows:

1. For each $\alpha \in \{1, \ldots, \kappa\}$, tasks from $B(\alpha)$ are scheduled following $M(\alpha)$. By definition, for each $\alpha \in \{1, \ldots, \kappa-1\}$, $M(\alpha)$ and $M(\alpha+1)$ are compatible and satisfy resource constraints, precedence constraints and tasks processing times;
2. For each $\alpha \in \{1, \ldots, \kappa-1\}$, tasks from $In(\alpha)$ are scheduled according to the feasible schedule $\sigma^{(\alpha)}$. This partial schedule exists since $(v(\alpha), v(\alpha+1)) \in \mathcal{A}$.

We first establish that each task is scheduled by σ.

Lemma 9 *For each task $i \in \mathcal{T}$, there exists $\alpha \in \{1, \dots \kappa - 1\}$ such that $i \in In(\alpha) \cup B(\alpha)$.*

Proof By the definition of a path of $\mathcal{G}$,

$$V(\kappa) \cup Z_\kappa = \emptyset \subseteq V(\kappa - 1) \cup Z_{\kappa-1} \subseteq \dots V(1) \cup Z_1 = \mathcal{T}.$$

Thus, for each task i, there exists $\alpha \in \{1, \dots \kappa - 1\}$ such that $i \in V(\alpha) \cup Z_\alpha$ and $i \notin V(\alpha+1) \cup Z_{\alpha+1}$, then $i \in (V(\alpha) \cup Z_\alpha) - (V(\alpha+1) \cup Z_{\alpha+1}) = In(\alpha) \cup B(\alpha)$, which proves the lemma. □

We prove in the next lemma that σ is a feasible schedule.

Lemma 10 *Each path of $\mathcal{G}$ from a node of $\mathcal{V}_1$ to a node of $\mathcal{V}_\kappa$ is associated with a feasible schedule.*

Proof Let σ be the schedule built previously from the sequence $v(\alpha)$. By Lemma 9, for each task $i \in \mathcal{T}$ there exists $\alpha_i \in \{1, \dots, \kappa - 1\}$ such that $i \in In(\alpha_i) \cup B(\alpha_i)$. The schedule of i is thus defined by the arc $(v(\alpha_i), v(\alpha_i+1))$.

Now, for any $\alpha \in \{1, \dots, \kappa - 1\}$, tasks from $In(\alpha) \cup B(\alpha)$ are completely or partially performed in the interval $[u_\alpha, u_{\alpha+1}]$. Moreover, by Lemma 6, $In(\alpha) \cup B(\alpha) = (V(\alpha) \cup Z_\alpha) - (V(\alpha + 1) \cup Z_{\alpha+1}) \subseteq X_\alpha$, and thus σ meets the release dates and deadlines.

Lastly, we prove that σ satisfies the precedence constraints. Consider a precedence relation $(i, j) \in A$. Let α_i be the maximum integer in $\{1, \dots, \kappa - 1\}$ such that $i \in In(\alpha_i) \cup B(\alpha_i)$. Since $V(\alpha_i) \cup Z_{\alpha_i}$ is strictly consistent, $j \in V(\alpha_i) \cup Z_{\alpha_i}$, and thus there exists $\alpha' \geq \alpha_i$ with $j \in In(\alpha') \cup B(\alpha')$. Let α_j be the minimum value in $\{\alpha_i, \dots, \kappa - 1\}$ such that $j \in In(\alpha_j) \cup B(\alpha_j)$. Two cases must be considered:

- If $\alpha_i = \alpha_j$ or ($\alpha_j = \alpha_i + 1$ and $j \in B(\alpha_i + 1)$), then there exists a feasible local schedule of tasks from $R(\alpha_i) = In(\alpha_i) \cup B(\alpha_i) \cup B(\alpha_i + 1)$, that results from the arc $(v(\alpha_i), v(\alpha_i + 1)) \in \mathcal{A}$. Thus the precedence constraint $i \to j$ is satisfied;
- Otherwise, $\alpha_j > \alpha_i$ and since $j \notin B(\alpha_i + 1)$, j is scheduled in the local schedule $\sigma^{(\alpha_j)}$ at time $t = C_j - p_j \geq u_{\alpha_j} \geq u_{\alpha_i+1}$. Now, by the definition of α_i, task i is completed in the local schedule $\sigma^{(\alpha_i)}$ at time $t' = C_i \leq u_{\alpha_i+1}$, thus $t' \leq t$ and the precedence constraint $i \to j$ is fulfilled.

□

Let us consider the path $(v_1, v_2, v_{31}, v_{42}, v_{52}, v_6)$ of the multistage auxiliary graph $\mathcal{G}$ presented by Figure 5. A feasible schedule associated to this path is depicted in Figure 6. For example, tasks associated with nodes v_{31} and v_{42} are respectively $V_{31} \cup Z_3 = \{2, 3, 5, 6, 7, 8\}$ and $V_{42} \cup Z_4 = \{3, 6, 7, 8\}$. Node v_{31} defines the border schedule of tasks 2 and 5, while v_{42} fixes 6 with $C_2 = 5$, $C_5 = 4$ and $C_6 = 9$. The schedule of these three tasks fulfills precedence and resource constraints. Moreover, since $V_{31} \cup Z_3 - V_{42} \cup Z_4 = \{2, 5\}$, no task is scheduled inside the interval $[u_3, u_4]$.

The next theorem is a simple outcome of Lemmas 8 and 10.

Theorem 2 *There exists a feasible schedule if and only if there exists a path in $\mathcal{V}$ from a node of $\mathcal{V}_1$ to a node of $\mathcal{V}_\kappa$.*

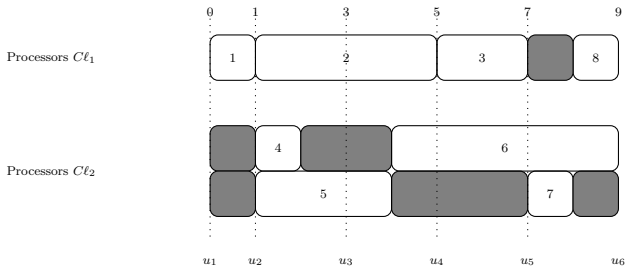

Fig. 6 A feasible schedule σ' associated to path $(v_1, v_2, v_{31}, v_{42}, v_{52}, v_6)$ of Figure 5.

5 Algorithm and complexity for checking the existence of a feasible schedule

We first present in this section a brute force algorithm to check the existence of a feasible schedule. Its complexity is then assessed.

5.1 Algorithm for checking the existence of a feasible schedule

Algorithm 1 is deduced from Theorem 2 to check the existence of a feasible schedule for an instance of $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$. The idea is simply to build the graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ and to check the existence of a path from $\mathcal{V}_1$ to $\mathcal{V}_\kappa$. The nodes are built by level from $\mathcal{V}_\kappa$ to $\mathcal{V}_1$, and thus the algorithm follows a Breadth-first search.

At line 4, nodes of $\mathcal{V}_\alpha$ are enumerated according to the nodes definition. Lines $5-8$ enumerate all the possible arcs in $\mathcal{V}_\alpha \times \mathcal{V}_{\alpha+1}$ and test their feasibility with Algorithm 2 before adding them to $\mathcal{A}$. Then, following Theorem 2, vertices of $\mathcal{V}_\alpha$ without successor are removed at lines $10 - 12$ as they cannot belong to a path from $\mathcal{V}_1$ to $\mathcal{V}_\kappa$. Moreover, the algorithm returns false if there exists $\alpha \in \{1, \ldots, \kappa - 1\}$ with $\mathcal{V}_\alpha = \emptyset$. In the opposite case, $\mathcal{V}_1 \neq \emptyset$ and the condition stated in Theorem 2 holds. The algorithm then returns true.

5.2 Complexity of Algorithm 1

The aim of this subsection is to evaluate the time complexity of Algorithm 1. We start with the complexity of checking the consistency and the strict consistency of a subset of dependent tasks. Then, we analyze the initialization phase of Algorithm 1, followed by the construction of nodes and arcs of the multi-stage auxiliary graph $\mathcal{G}$. The last subsection concludes with the evaluation of the overall complexity of Algorithm 1.

Algorithm 1 Checking the existence of a feasible schedule

Require: An instance $\mathcal{I}$ of $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$
Ensure: True iff $\mathcal{I}$ is feasible
Build the sequence u_α, the sets X_α and Z_α, for $\alpha \in \{1, \ldots, \kappa\}$
$\mathcal{V}_\kappa \leftarrow \{(\emptyset, M_\bullet)\}$, $\mathcal{V} \leftarrow \mathcal{V}_\kappa$, $\mathcal{A} \leftarrow \emptyset$ and $\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{A})$
for $\alpha \in (\kappa - 1, \kappa - 2, \ldots, 1)$ **do**
  Enumerate all the nodes of $\mathcal{V}_\alpha$ following nodes conditions
  for all $(v, v') \in \mathcal{V}_\alpha \times \mathcal{V}_{\alpha+1}$ **do**
    if $Existence\_arc(v, v')$ **then**
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{(v, v')\}$
    end if
  end for
  for all $v \in \mathcal{V}_\alpha$ with no successor in $\mathcal{G}$ **do**
    $\mathcal{V}_\alpha \leftarrow \mathcal{V}_\alpha - \{v\}$
  end for
  if $\mathcal{V}_\alpha = \emptyset$ **then**
    return False
  end if
  $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{V}_\alpha$
end for
return True

5.2.1 Complexity for consistency/strict consistency

Lemma 11 *Let $G = (\mathcal{T}, A)$ be a directed acyclic graph with $n = |\mathcal{T}|$. Consider a subset $X \subseteq \mathcal{T}$. Checking the consistency of X can be done in time $\mathcal{O}(n^3)$ while checking its strict consistency can be achieved in time $\mathcal{O}(|A|)$.*

Proof Checking that X is consistent can be done by testing, for each tuple $(i, j) \in X^2$ with $i \neq j$ if $\Gamma^{+\star}(i) \cap \Gamma^{-\star}(j) \subseteq X$. The computation of the sets $\Gamma^{-\star}(i)$ and $\Gamma^{+\star}(i)$ for any task $i \in \mathcal{T}$ can be done once in time $\mathcal{O}(n + |A|)$ using a depth-first search algorithm. The overall time complexity for checking consistency of X is then in $\mathcal{O}(n^3)$.

Set X is strictly consistent if for any task $x \in X$, $\Gamma^+(x) \subseteq X$. Checking this for a given task x can be done in time $\mathcal{O}(|\Gamma^+(x)|)$. The overall time complexity is thus bounded by $\sum_{x \in X} |\Gamma^+(x)| \leq \sum_{x \in \mathcal{T}} |\Gamma^+(x)| = |A|$; the algorithm is thus in time complexity $\mathcal{O}(|A|)$. □

5.2.2 Initialization phase

Lemma 12 *Computing the sequence u_α and the sets X_α and Z_α for $\alpha \in \{1, \ldots, \kappa\}$ can be done in time complexity $\mathcal{O}(n^2)$.*

Proof The sequence u_α can be easily built from the sorted values r_i and d_i, $i \in \{1, \ldots, n\}$ in time $\mathcal{O}(n \log_2 n)$.

All the sets X_α for $\alpha \in \{1, \ldots, \kappa\}$ can be built in time $\mathcal{O}(n^2)$: we first find for each task $i \in \mathcal{T}$ the values α_i and β_i such that $u_{\alpha_i} = r_i$ and $u_{\beta_i} = d_i$. Each of these searches can be done in time $\mathcal{O}(\log_2 n)$. Adding each task $i \in \mathcal{T}$ to sets $X_{\alpha_i}, \ldots, X_{\beta_i}$ can be done in time $\mathcal{O}(n)$ thus the overall time to compute all theses sets is $\mathcal{O}(n^2)$.

Lastly, Z_α can be built in $\mathcal{O}(n^2)$ by adding each task $i \in \mathcal{T}$ to the sets $Z_1, \ldots, Z_{\alpha_i}$. □

5.2.3 Computation of nodes of $\mathcal{G}$

Let us now consider an instance of our scheduling problem, and a class of machines $C\ell_p$ for $p \in \{1, \ldots, k\}$. Following Definition 3, in any feasible schedule and at each time unit t, there are at most μ tasks $i \in \mathcal{T}$ processed at time t and such that $\tau_i = p$. So if $m_p > \mu$ we can remove the useless $m_p - \mu$ processors. Thus, we can assume without loss of generality that $\forall p \in \{1, \ldots, k\}$, $m_p \leq \mu$. Lemma 13 follows.

Lemma 13 *For each integer value $\alpha \in \{1, \ldots, \kappa\}$, $|\mathcal{V}_\alpha| \leq f(\mu, \rho)$ with $\mu = max_{\alpha \in \{1,\ldots,\kappa-1\}} |X_\alpha|$, $\rho = \min(p_{\max}, s\ell_{\max})$ and $f(\mu, \rho) = \rho^{\mu^2} \times (\mu+1)^{\mu^2} \times 2^\mu$.*

Proof Let us consider a vertex $v = (V, M) \in \mathcal{V}_\alpha$ for $\alpha \in \{1, \ldots, \kappa\}$. Since $V \subset X_\alpha \cap X_{\alpha-1}$, the number of all the possible subsets V is bounded by $2^{|X_\alpha|}$, and thus 2^μ.

A class of machines $C\ell_p$ with $p \in \{1, \ldots, k\}$ is said to be active for V if at least one task $i \in V$ fulfills $\tau_i = C\ell_p$. We denote by $C\ell(V)$ the set of active classes of V. As $|V| \leq |X_\alpha|$, there are at most $|X_\alpha|$ active machine classes, thus $|C\ell(V)| \leq |X_\alpha| \leq \mu$.

Asssume that V is fixed. Let $C\ell_p \in C\ell(V)$ and let $\nu_\alpha(p, V)$ be the number of border schedules M_p of tasks from V assigned to a machine of $C\ell_p$. The number of mappings of tasks from V to machines of $C\ell_p$ is upper bounded by $(m_p + 1)^{|V|}$. Moreover, for each task allocated to a given machine, at most $\rho + 1$ possible completion times for this task are to be considered in a border schedule. So, $\nu_\alpha(p, V) \leq \rho^{m_p} \times (m_p+1)^{|V|}$. Moreover, as noticed before, $m_p \leq \mu$ and thus $\nu_\alpha(p, V) \leq \rho^\mu \times (\mu + 1)^\mu$.

Observe that $\nu_\alpha(p, V) = 0$ if $C\ell_p$ is not active, ie. $C\ell_p \notin C\ell(V)$. Thus, for a fixed set V, the total number of associated border schedules is bounded by

$$\prod_{C\ell_p \in C\ell(V)} \nu_\alpha(p, V) \leq \rho^{\mu \times |C\ell(V)|} \times (\mu + 1)^{\mu \times |C\ell(V)|}.$$

Now, $|C\ell(V)| \leq |V| \leq \mu$ and thus $\prod_{C\ell_p \in C\ell(V)} \nu_\alpha(p, V) \leq \rho^{\mu^2} \times (\mu+1)^{\mu^2}$. Then, the number of nodes from $\mathcal{V}_\alpha$ verifies:

$$|\mathcal{V}_\alpha| \leq \sum_{V \subseteq X_\alpha} \prod_{C\ell_p \in C\ell(V)} \nu_\alpha(p, V) \leq \sum_{V \subseteq X_\alpha} \rho^{\mu^2} \times (\mu + 1)^{\mu^2}.$$

Thus, $|\mathcal{V}_\alpha| \leq 2^{|X_\alpha|} \times \rho^{\mu^2} \times (\mu + 1)^{\mu^2} \leq 2^\mu \times \rho^{\mu^2} \times (\mu + 1)^{\mu^2} = f(\mu, \rho)$ which concludes the proof. □

Lemma 14 *The computation of the set $\mathcal{V}$ is in time $\mathcal{O}(n^4 \times f(\mu, \rho))$ with $f(\mu, \rho) = \rho^{\mu^2} \times (\mu + 1)^{\mu^2} \times 2^\mu$.*

Proof By Lemma 11, testing the consistency of $V \cup Z_\alpha$ for any $\alpha \in \{1, \ldots, \kappa\}$ is in time $\mathcal{O}(n^3)$. By Lemma 13, $|\mathcal{V}_\alpha| \leq f(\mu, \rho)$, thus the construction of $\mathcal{V}_\alpha$ (line 4) is in time $\mathcal{O}(n^3 \times f(\mu, \rho))$. Testing the existence of successors for all vertices in $\mathcal{V}_\alpha$ (lines $10 - 12$) is similarly in time $\mathcal{O}(f(\mu, \rho))$. Now, as $\kappa \leq 2n$, we conclude that the construction of $\mathcal{V}$ is in time $\mathcal{O}(n^4 \times f(\mu, \rho))$, thus the lemma holds. □

5.2.4 Computation of the arcs of $\mathcal{G}$

Algorithm 2 summarizes the steps to check the existence of the arc (v, v') with $v \in \mathcal{V}_\alpha$ and $v' \in \mathcal{V}_{\alpha+1}$. One key point of the construction of an arc (v, v') of $\mathcal{G}$ is to check the existence of a feasible schedule of the tasks in an interval $(u_\alpha, u_{\alpha+1})$, described in lines 11-19 of Algorithm 2.

Algorithm 2 *Existence_arc*(v, v')

Require: Two nodes $v \in \mathcal{V}_\alpha = (V, M), v' \in \mathcal{V}_{\alpha+1} = (V', M')$. B (resp. B') is the set of tasks associated to M (resp. M').
Ensure: True if the arc (v, v') exists in $\mathcal{G}$
1: **if** $V' \cup Z_{\alpha+1} \not\subseteq V \cup Z_\alpha$ **then**
2: **return** False
3: **end if**
4: **if** B or B' is not consistent **then**
5: **return** False
6: **end if**
7: $In = (V \cup Z_\alpha) - (V' \cup Z_{\alpha+1}) - B$
8: **if** In is not consistent **then**
9: **return** False
10: **end if**
11: $G_{In,B,B'} = (In \cup B \cup B', A)$
12: Build the set $\mathcal{TO}$ of all topological ordering of In following the order $\prec$ expressed by $G_{In,B,B'}$
13: Build the set Map of all possible mappings of In
14: **for all** $(to, map) \in \mathcal{TO} \times Map$ **do**
15: Build the schedule σ associated to the tuple (to, map)
16: **if** σ is feasible **then**
17: **return** True
18: **end if**
19: **end for**
20: **return** False

Lemma 15 explains and analyzes the complexity of lines 11-19 of Algorithm 2. This complexity relies on the size of the induced subsets of tasks.

Lemma 15 *Let $v = (V, M) \in \mathcal{V}_\alpha$ and $v' = (V', M') \in \mathcal{V}_{\alpha+1}$ be two nodes of $\mathcal{G}$. Let also B (resp. B') the set of tasks scheduled by M (resp. M'), $In = (V \cup Z_\alpha) - (V' \cup Z_{\alpha+1}) - B$, and $R = In \cup B \cup B'$. The time complexity of lines 11-19 of Algorithm 2 that check the existence of a feasible schedule of tasks of set In in the interval $[u_\alpha, u_{\alpha+1}]$ and consistent with the constraints induced by M and M' is in time $\mathcal{O}(|R|^2 \times \mu^{|In|} \times |In|!)$.*

Proof The problem is here to schedule tasks from In considering a fixed schedule $M \cup M'$ of tasks from $B \cup B'$.

For any tuple $(i,j) \in In$, we note $i \prec j$ if $i \to j$. A topological ordering $\prec^\star$ (ie. a total order of tasks such that, if $i \prec j$ then $i \prec^\star j$) can be built in time $\mathcal{O}(n + |A|)$. The value n is here the number of tasks, and $|A|$ is the number of arcs of the precedence relations.

We note that a topological ordering $\prec^\star$ and a mapping (ie. an allocation of all tasks of In to the processors) are sufficient to represent any feasible schedule of tasks from In. Indeed, the tasks are simply executed as soon as possible and the topological ordering is used to resolve the conflicts due to the resource limitation.

Lines 11-19 of Algorithm 2 are based on a naive enumeration of feasible schedules from all the possible topological orderings and mappings. It returns true if and only if a feasible schedule is found.

For any value $p \in \{1, \ldots, k\}$, we denote by $In(p) = \{i \in In, \tau_i = C\ell_p\}$ the set of tasks from In that must be performed on a machine from $C\ell_p$. If $In(p) \neq \emptyset$, the number of feasible mappings of tasks from $In(p)$ is bounded by $m_p^{|In(p)|}$. Since $m_p \leq \mu$, $m_p^{|In(p)|} \leq \mu^{|In(p)|}$ and the number N of feasible mappings of tasks from In is bounded by:

$$N \leq \prod_{p=1}^{k} \mu^{|In(p)|} \leq \mu^{\sum_{p=1}^{k} |In(p)|} = \mu^{|In|}.$$

The total number of topological orderings of In is bounded by $|In|!$, thus $|\mathcal{TO} \times Map| \leq \mu^{|In|} \times |In|!$.

Now, the construction of a schedule from a tuple (to, map) at line 4 can be done by executing tasks following the topological order, the mapping and the precedence constraints, which requires $\mathcal{O}(|R|^2)$ operations. Algorithm 2 is thus in time $\mathcal{O}(|R|^2 \times \mu^{|In|} \times |In|!)$. □

We are ready to analyze the whole time complexity of Algorithm 2.

Lemma 16 *The time complexity of Algorithm 2 is* $\mathcal{O}(\mu^{(\mu+2)} \times \mu! + n^3)$.

Proof Suppose that $v = (V, M) \in \mathcal{V}_\alpha$ and $v' = (V', M') \in \mathcal{V}_{\alpha+1}$. We denote by B (resp. B') the set of tasks scheduled by M (resp. M'). Moreover $In = (V \cup Z_\alpha) - (V' \cup Z_{\alpha+1}) - B$ and $R = In \cup B \cup B'$. We have to compute the following steps to check if $(v, v') \in \mathcal{A}$: inclusion, consistency of borders, consistency of In, and existence of a schedule of In.

Inclusion (lines 1-3): Checking that $V' \cup Z_{\alpha+1} \subseteq V \cup Z_\alpha$ can be done in time $\mathcal{O}(n)$;

Consistency of borders (lines 4-6): As $|B| \leq \mu$ and $|B'| \leq \mu$, checking the border consistency may be done in time $\mathcal{O}(\mu^2)$;

Consistency of In (lines 8-10): The consistency of $In = (V \cup Z_\alpha) - (V' \cup Z_{\alpha+1}) - B$ is checked in $\mathcal{O}(n^3)$ according to Lemma 11;

Existence of a schedule (lines 11-19): Following Lemma 15, checking the existence of a schedule can be done in time $\mathcal{O}((|R|^2 \times \mu^{|In|} \times |In|!))$. Now, as $|In| \leq \mu$, and $|R_\alpha| \leq \mu$ following Lemma 2, the existence of a schedule is bounded in time by $\mathcal{O}(\mu^{(\mu+2)} \times \mu!)$ for one arc.

The overall complexity for each tuple $(v, v') \in \mathcal{V}_\alpha \times \mathcal{V}_{\alpha+1}$ is thus in time $\mathcal{O}(\mu^{(\mu+2)} \times \mu! + n^3)$. □

5.2.5 Overall complexity of Algorithm 1

Next Theorem states the time complexity of Algorithm 1.

Theorem 3 *The time complexity of Algorithm 1 is* $\mathcal{O}(f(\mu,\rho)^2 \times (n^4 + n \times \mu^{\mu+2} \times \mu!))$ *with* $\rho = \min(p_{\max}, s\ell_{\max})$, $\mu = pw(I) + 1$ *and* $f(\mu,\rho) = \rho^{\mu^2} \times (\mu+1)^{\mu^2} \times 2^\mu$.

Proof Let us consider all the steps of Algorithm 1 for the construction of the graph $\mathcal{G}$ for an instance of the scheduling problem. The initialization phase can be done in time $\mathcal{O}(n^2)$ following Lemma 12. The construction of the set of nodes $\mathcal{V}$ can be done in time $\mathcal{O}(n^4 \times f(\mu,\rho))$ following Lemma 14. Consider now the construction of the set of arcs. By Lemma 13 and since $\kappa \leq 2n$, the number of arcs can be measured as follows:

$$|\mathcal{A}| \leq \sum_{\alpha=1}^{\kappa-1} (|\mathcal{V}_\alpha| \times |\mathcal{V}_{\alpha+1}|) \leq \sum_{\alpha=1}^{\kappa-1} f(\mu,\rho)^2 \leq \kappa \times f(\mu,\rho)^2 \leq 2n \times f(\mu,\rho)^2.$$

Then for each couple $(v, v') \in \mathcal{V}_\alpha \times \mathcal{V}_{\alpha+1}$ Algorithm 1 calls Algorithm 2 once; by Lemma 16, we get that building $\mathcal{A}$ is in time $\mathcal{O}(n \times f(\mu,\rho)^2 \times (\mu^{(\mu+2)} \times \mu! + n^3))$. The theorem follows since the complexity of pruning nodes without successors only depends on the number of nodes. □

Corollary 2 $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$ *is fixed-parameter tractable by the tuple* $(pw(I), \rho)$ *where* $pw(I)$ *is the pathwidth of the interval graph built from the intervals* $\{(r_i, d_i), i \in \mathcal{T}\}$ *and* ρ *the minimum between the maximal processing time* $p_{\max}$ *and the maximal slack* $s\ell_{\max}$.

Proof By Theorem 2, an instance of $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$ has a solution if and only if there exists a path in $\mathcal{G}$ from a node of $\mathcal{V}_1$ to a node of $\mathcal{V}_\kappa$. By Theorem 3, the complexity for building $\mathcal{G}$ has a FPT form with respect to the parameters (exponential function of the parameters times a polynomial on n), thus the corollary holds. □

5.2.6 Related optimization problems

Let us first consider the optimization problem $P|\mathcal{M}_j(type), prec, r_i, d_i|C_{\max}$ associated to our decision problem. If we consider any upper bound $\overline{C} \leq \max_{i\in\mathcal{T}} d_i$ of the makespan, deadlines may be adjusted in polynomial time with respect to $\overline{C}$ taking into account precedence and resource constraints (see. as

example [2,9,10]). These new time intervals of tasks are included in the original ones, thus the associated interval graph $I(\overline{C})$ will satisfy $pw(I(\overline{C})) \leq pw(I)$. By using Algorithm 1, we can thus check the existence of a schedule with makespan not greater than $\overline{C}$. This step can be embedded in a binary search on $\overline{C}$ to get the optimal makespan in time $\mathcal{O}(\log(s\ell_{\max})(f(\mu,\rho)^2 \times (n^4 + n \times \mu^{\mu+2} \times \mu!))$, with $f(\mu,\rho) = \rho^{\mu^2} \times (\mu+1)^{\mu^2} \times 2^{\mu}$.

For optimization problems whose instances do not include deadlines, the pathwidth parameter should depend on an upper bound of the completion times of tasks. This upper bound should be chosen not too large, otherwise the parameter could be trivially large. Then, like in [37] we can consider that the maximal slack is part of the instance, with $s\ell_{\max}$ defined as an upper bound on $\max_{i\in\mathcal{T}}(C_i - r_i - p_i)$ for any schedule. Then, we can fix the deadline of any task $i \in \mathcal{T}$ to $\overline{d}_i = r_i + p_i + s\ell_{\max}$ and the pathwidth parameter can then be computed with respect to the intervals set $\{(r_i, \overline{d}_i), i \in \mathcal{T}\}$. The previous binary search applies and $P|\mathcal{M}_j(type), prec, r_i, s\ell_{\max}|C_{\max}$ is thus fixed parameter tractable for the tuple of parameters $(pw(I), \rho)$.

This idea can also be used to solve the optimization problem with maximum lateness criteria $P|\mathcal{M}_j(type), prec, r_i|L_{\max}$ for which due dates $(d_i)_{i\in\mathcal{T}}$ are given. If the maximal slack is given, the parameter $pw(I)$ can then be computed as above from the deadlines $(\overline{d}_i)_{i\in\mathcal{T}}$ induced by $s\ell_{\max}$.

Notice that $s\ell_{\max}$ is also an upper bound on the maximum lateness, since for any task $i \in \mathcal{T}$, $C_i - r_i - p_i \leq s\ell_{\max}$; so that $C_i - r_i - p_i \leq d_i - r_i - p_i + s\ell_{\max}$ and thus $C_i \leq d_i + s\ell_{\max}$. A binary search on an upper bound $\overline{L}$ of the lateness starting with $\overline{L} = s\ell_{\max}$ and repeatedly using Algorithm 1 will then lead to a similar time complexity, setting at each step the deadlines to $d_i(\overline{L}) = \min(d_i + \overline{L}, \overline{d}_i)$ for any task $i \in \mathcal{T}$.

Next corollaries are a simple outcome of Corollary 2 combined with the discussion above:

Corollary 3 *The problem with deadlines $P|\mathcal{M}_j(type), prec, r_i, d_i|C_{\max}$ is fixed-parameter tractable by the tuple $(pw(I), \rho)$ where $pw(I)$ is the pathwidth of the interval graph built from the intervals $\{(r_i, d_i), i \in \mathcal{T}\}$ and ρ the minimum between the maximal processing time $p_{\max}$ and the maximal slack $s\ell_{\max}$.*

Corollary 4 *The two optimization problems where the maximal slack is given $P|\mathcal{M}_j(type), prec, r_i, s\ell_{\max}|C_{\max}$ and $P|\mathcal{M}_j(type), prec, r_i, s\ell_{\max}|L_{\max}$ are also fixed-parameter tractable for the tuple $(pw(I), \rho)$, where $pw(I)$ is the pathwidth of the interval graph built from the intervals $\{(r_i, r_i + p_i + s\ell_{\max}), i \in \mathcal{T}\}$.*

6 Conclusion and perspectives

We proved that the scheduling decision problem $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$ is fixed-parameter tractable by the pathwidth $pw(I)$ of the interval graph I induced by tasks intervals, and the minimum value ρ between the maximum processing time of a task $p_{\max}$ and the maximum slack $s\ell_{\max}$. We also

showed that if $\mathsf{P} \neq \mathsf{NP}$, there is no FPT algorithm for the decision problem $P|\mathcal{M}_j(type), prec, r_i, d_i|\star$ parameterized by only one parameter among the previous ones.

By augmenting our FPT algorithm by a binary search, we deduced that the optimization problem $P|\mathcal{M}_j(type), prec, r_i, d_i|C_{\max}$ is also fixed-parameter tractable by $(pw(I), \rho)$. By computing $pw(I)$ from an upper bound on the slack, we prove that the two optimization problems $P|\mathcal{M}_j(type), prec, r_i, s\ell_{\max}|C_{\max}$ and $P|\mathcal{M}_j(type), prec, r_i, s\ell_{\max}|L_{\max}$ are fixed-parameter tractable by the tuple $(pw(I), \rho)$. We thus provide the first fixed-parameter algorithms for these important basic scheduling problems combining precedence constraints and time windows and prove that the tuple $(pw(I), \rho)$ is a good parameter to capture the parallelism of a scheduling problem.

From a practical point of view, dedicated optimizations should be considered to improve the performances of this new exact algorithm. A partial exploration of the multistage auxiliary graph and dominance properties should be developed to reduce dramatically the execution time of the algorithm. Another interesting feature is that the pathwidth $pw(I)$ and the slack can be reduced by preprocessing techniques, that have been used for a long time for problems with unit processing times [8,16,21,27] or for shop problems in branch-and-bound algorithms [2,9,10]. The investigation of such techniques and their efficiency to reduce the parameters should be further carried out.

Lastly, the pathwidth $pw(I)$ of the associated interval graph is clearly a promising new parameter for studying other classes of precedence constraints, as for example communication delays [36] or more general criteria [1]. This parameter gives a new insight in this field, that may lead to many complexity and algorithmic results.

Acknowledgments

We thank Sebastien Bonduelle for pointing out several errors and typos. We are also very grateful to the reviewers for their helpful recommendations.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Baart, R., de Weerdt, M., He, L.: Single-machine scheduling with release times, deadlines, setup times, and rejection. European Journal of Operational Research **291**(2), 629–639 (2021)
2. Baptiste, P., Le Pape, C., Nuijten, W.: Satisfiability tests and timebound adjustments for cumulative scheduling problems. Annals of Operations Research **92**(0), 305–333 (1999)

3. Bessy, S., Giroudeau, R.: Parameterized complexity of a coupled-task scheduling problem. J. Scheduling **22**(3), 305–313 (2019)
4. Bodlaender, H.L.: A tourist guide through treewidth. Acta Cybern. **11**, 1–21 (1992)
5. Bodlaender, H.L., Fellows, M.R.: W[2]-hardness of precedence constrained k-processor scheduling. Oper. Res. Lett. **18**(2), 93–97 (1995)
6. Bodlaender, H.L., van der Wegen, M.: Parameterized complexity of scheduling chains of jobs with delays. arXiv preprint arXiv:2007.09023 (2020)
7. Brucker, P.: Scheduling algorithms (4. ed.). Springer, Berlin (2004)
8. Carlier, A., Hanen, C., Munier Kordon, A.: The equivalence of two classical list scheduling algorithms for dependent typed tasks with release dates, due dates and precedence delays. Journal of Scheduling **20**(3), 303–311 (2017)
9. Carlier, J., Peridy, L., Pinson, E., Rivreau, D.: Elimination rules for the job-shop. In: J.Y.T. Leung (ed.) Handbook of scheduling : algorithms, models, and performance analysis., pp. 1–19. Chapman & Hall, CRC (2004)
10. Carlier, J., Pinson, E.: Jackson's pseudo-preemptive schedule and cumulative scheduling problems. Discrete Applied Mathematics **145**(1), 80–94 (2004)
11. Chen, B., Potts, C.N., Woeginger, G.J.: A Review of Machine Scheduling: Complexity, Algorithms and Approximability, pp. 1493–1641. Springer US, Boston, MA (1998)
12. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms, 1st edn. Springer Cham, Switzerland (2015)
13. Dolev, D., Warmuth, M.K.: Scheduling precedence graphs of bounded height. J. Algorithms **5**(1), 48–59 (1984)
14. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity, 1st edn. Springer-Verlag, London (2013)
15. Flum, J., Grohe, M.: Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series). Springer-Verlag, Berlin, Heidelberg (2006)
16. Garey, M.R., Johnson, D.S.: Two-processor scheduling with start-time and deadlines. SIAM Journal on Computing **6**, 416–426 (1977)
17. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)
18. Graham, R., Lawler, E., Lenstra, J., Kan, A.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: P. Hammer, E. Johnson, B. Korte (eds.) Discrete Optimization II, *Annals of Discrete Mathematics*, vol. 5, pp. 287 – 326. Elsevier (1979)
19. Gromicho, J.A., Van Hoorn, J.J., Saldanha-da Gama, F., Timmer, G.T.: Solving the job-shop scheduling problem optimally by dynamic programming. Computers & Operations Research **39**(12), 2968 – 2977 (2012)
20. Günther, E., König, F.G., Megow, N.: Scheduling and packing malleable and parallel tasks with precedence constraints of bounded width. Journal of Combinatorial Optimization **27**(1), 164–181 (2014)
21. Hanen, C., Zinder, Y.: The worst-case analysis of the Garey-Johnson algorithm. Journal of Scheduling **12**(4), 389–400 (2009)
22. van Hoorn, J.J., Nogueira, A., Ojea, I., Gromicho, J.A.S.: An corrigendum on the paper: Solving the job-shop scheduling problem optimally by dynamic programming. Comput. Oper. Res. **78**, 381 (2017)
23. Jansen, K.: Analysis of scheduling problems with typed task systems. Discrete Applied Mathematics **52**(3), 223 – 232 (1994)
24. Knop, D., Koutecký, M.: Scheduling meets n-fold integer programming. Journal of Scheduling **21**(5), 493–503 (2018)
25. Lenstra, J., Rinnooy Kan, A., Brucker, P.: Complexity of machine scheduling problems. In: P. Hammer, E. Johnson, B. Korte, G. Nemhauser (eds.) Studies in Integer Programming, *Annals of Discrete Mathematics*, vol. 1, pp. 343–362. Elsevier, The Netherlands (1977)
26. Lenstra, J.K., Rinnooy Kan, A.H.G.: Complexity of scheduling under precedence constraints. Operations Research **26**(1), 22–35 (1978)
27. Leung, A., Palem, K.V., Pnueli, A.: Scheduling time-constrained instructions on pipelined processors. ACM Trans. Program. Lang. Syst. **23**, 73–103 (2001)

28. Leung, J., Kelly, L., Anderson, J.H.: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press, Inc., Boca Raton, FL, USA (2004)
29. Leung, J., Li, C.L.: Scheduling with processing set restrictions: A survey. International Journal of Production Economics **116**, 251–262 (2008)
30. Liu, J.W., Liu, C.L.: Performance analysis of multiprocessor systems containing functionally dedicated processors. Acta Informatica **10**(1), 95–104 (1978)
31. Mnich, M., van Bevern, R.: Parameterized complexity of machine scheduling: 15 open problems. Computers and Operations Research **100**, 254 – 261 (2018)
32. Mnich, M., Wiese, A.: Scheduling and fixed-parameter tractability. Mathematical Programming **154**, 533–562 (2015)
33. Möhring, R.H.: Computationally Tractable Classes of Ordered Sets, pp. 105–193. Springer Netherlands, Dordrecht (1989)
34. Munier Kordon, A.: A fixed-parameter algorithm for scheduling unit dependent tasks on parallel machines with time windows. Discrete Applied Mathematics **290**, 1–6 (2021)
35. Robertson, N., Seymour, P.: Graph minors. i. excluding a forest. Journal of Combinatorial Theory, Series B **35**(1), 39 – 61 (1983)
36. Tang, N., Munier Kordon, A.: A fixed-parameter algorithm for scheduling unit dependent tasks with unit communication delays. In: European Conference on Parallel Processing, pp. 105–119. Springer (2021)
37. Van Bevern, R., Bredereck, R., Bulteau, L., Komusiewicz, C., Talmon, N., Woeginger, G.J.: Precedence-constrained scheduling problems parameterized by partial order width. In: Y. Kochetov, M. Khachay, V. Beresnev, E. Nurminski, P. Pardalos (eds.) Discrete Optimization and Operations Research, pp. 105–120. Springer International Publishing, Cham (2016)