Parameterized analysis of single machine scheduling with time windows and precedence delays

Maher Mallem[1][0000−0001−5654−1090], Claire Hanen[1,2][0000−0003−2482−5042], and Alix Munier Kordon[1][0000−0002−2170−6366]

[1] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
{Maher.Mallem,Claire.Hanen,Alix.Munier}@lip6.fr
http://www.lip6.fr
[2] UPL, Université Paris Nanterre, F-92000 Nanterre, France

Abstract. This paper studies the parameterized complexity of single machine scheduling problems with time windows and precedence delays considering three parameters: the pathwidth of the interval graph induced by time windows, the maximum slack of a job within its time window, and the maximum delay value. Three types of precedence delays are studied: minimum, maximum and exact. We first exhibit a relation between the slack and the pathwidth which may induce some parameterized complexity results. Then for all types of precedence delays we prove that these problems are para-NP-hard parameterized by either the slack or the pathwidth even for unit processing time coupled tasks with a limited number of distinct delay values. Finally a FPT algorithm is provided when the maximum delay is combined with the slack.

Keywords: parameterized complexity · scheduling · precedence delays · general precedence

1 Introduction

Since the seventies many studies have been devoted to scheduling problems on a single processor with different settings and optimization criteria. If we just consider decision problems only few problems can be solved in polynomial time - like scheduling unit execution time jobs with time windows [14] or scheduling jobs with precedence constraints and deadlines [11]. On the other hand many sequencing problems with simple settings are NP-hard in the strong sense, especially when time windows are considered [12].

This paper addresses the feasibility check of a single machine problem with time windows and precedence constraints with delays. We are given a set of jobs $\mathcal{T}$. Each job $j \in \mathcal{T}$ is characterized by a processing time p_j and a time window $[r_j, \bar{d}_j)$ in which it should be scheduled. A nonnegative delay ℓ_a is attached to each precedence constraint a between two jobs i and j. Three types of delays are considered. In the case of minimum delays, the difference between the starting

time of j and the completion time of i must not be less than ℓ_a; in the case of maximum delays, this difference must be not greater than ℓ_a; for exact delays this difference must be exactly equal to ℓ_a. Such kind of precedence constraints, sometimes called "generalized precedence constraints" in the literature [6] have many applications. Notice that the complexity of problems with maximum delays is usually less studied than minimum and exact delays. However, even for coupled tasks with unit processing times - where the precedence graph is a set of disjoint arcs scheduling jobs on single machine has been proven NP-hard by Yu [17] for exact and minimum delays. See [5,10] for a survey of complexity results of coupled tasks with exact delays.

Our aim is to carry out a parameterized complexity analysis of our problem. Several parameters have been considered for the parameterized complexity analysis of scheduling problems. We can mention the maximum processing time $p_{\max}$ [15] or the width of the precedence graph [3].

Van Bevern et al. [3] proved that the problem of scheduling a precedence graph on two machines with processing times in $\{1, 2\}$ is W[2]-hard with respect to the width. In the same paper they introduced an additional parameter λ that measures the maximum allowed lag of a job with respect to its earliest start time (ignoring resource constraints). They proved that the Resource-Constrained Project Scheduling Problem (RCPSP) is FPT parameterized by both the allowed lag and the width. Considering precedence delays the work of Bessy et al. [2] studies coupled tasks with due dates, a compatibility graph and a common deadline on a single machine. Their consider the number of jobs ending before their due date as parameter. They established W[1]-hardness for this problem and developed a FPT algorithm when the maximum duration of a job (i.e. the processing time of the two tasks plus their delay) is bounded.

Bodlaender and van der Wegen [4] addressed the single-machine scheduling of a set of chain like precedence constraints with delays where the time windows are expressed on chains and not on individual jobs. Their parameter - called the thickness - corresponds to the maximum number of overlapping chains. They proved that the problem with minimum delays given in unary is W[2]-hard with respect to the thickness. When each job has its time window, two parameters have been introduced in the literature, in line of thickness and allowed lag. The pathwidth μ is the maximum number of overlapping time windows minus one. The slack σ is the maximum slack of a job j in its time window: $\bar{d_j} - (r_j + p_j)$.

Munier Kordon [16] developed a FPT algorithm for the decision problem of scheduling unit execution time jobs with precedence constraints and time windows of jobs parameterized by μ. In [13] scheduling chains of unit execution time jobs with minimum or exact delays on a single processor was proved to be para-NP-hard with parameter μ, whereas it was proved to be FPT if no precedence delays are assumed in [8]. A FPT with the tuple of parameters "slack and pathwidth" (or "maximum processing time and pathwidth") was proposed in [9] for a parallel machine scheduling problem with precedence and time windows. For a single machine problem with job rejection criteria a FPT algorithm with parameters slack and pathwidth has also been proposed [1].

In this paper we consider parameters μ and σ. We also consider the maximum precedence delay $\ell_{\max}$ as a parameter. To the best of our knowledge this last parameter has not been investigated for this problem yet. Our goal is to establish the boundary between the FPT problems and the others as accurately as possible.

Our contribution first establishes in Section 2 a relation between the slack and the pathwidth. This implies that results already known for one of the two parameters can be transferred to the other. Then we show in Section 3 that scheduling coupled tasks with equal precedence delays and unit execution time on a single machine with minimum,maximum or exact delays is para-NP-hard with respect to the slack, and thus the pathwidth. In section 4 we provide a FPT algorithm for the problem with any precedence graph when parameter slack is combined with the maximum precedence delay. Finally in Section 5 we draw some perspectives.

2 Definitions and parameters

2.1 Definitions

In this paper we study the problem denoted by $1|prec(\ell_{i,j}), r_j, \bar{d}_j|\star$ in the standard notation of Graham [7]. An instance of this problem is characterized by a set $\mathcal{T}$ of n jobs. Each job j in $\mathcal{T}$ has a processing time p_j, a release date r_j and a deadline $\bar{d}_j$. These parameters may be also denoted by $p(J), r(J), \bar{d}(J)$ when the job J has a long name. Moreover we consider an acyclic precedence graph G representing precedence relations between the jobs in $\mathcal{T}$. Each arc $a = (i, j)$ in G is valued by a nonnegative integer $\ell_{i,j}$. A schedule s defines for each job j a starting time $s(j)$ so that $r_j \leq s(j) < \bar{d}_j$ and for any arc $a = (i, j)$ in G:

minimum delays: $s(j) \geq s(i) + p_i + \ell_{i,j}$
maximum delays: $s(i) + p_i \leq s(j) \leq s(i) + p_i + \ell_{i,j}$
exact delays: $s(j) = s(i) + p_i + \ell_{i,j}$

When a unique type of delays is considered, a superscript max, min or ex is indicated in the problem definition. For example, $1|prec(\ell^{\max}_{i,j}), r_j, \bar{d}_j|\star$ describes the problem with only maximum delays.

A large part of this paper is devoted to the special case with unit execution time coupled tasks. A coupled task is a set of two unit execution time jobs linked by an arc of the precedence graph. So a coupled task can be described by a triplet (J_1, ℓ, J_2). In scheduling problems with coupled tasks we assume that each job has its time window and that no precedence constraint exists between tasks from different coupled tasks. In Section3, we also consider that all the precedence delays are equal to a single value ℓ. The problem is then denoted by $1|(1, \ell, 1), r_j, \bar{d}_j|\star$.

Considering an instance of $1|prec(\ell_{i,j}), r_j, \bar{d}_j|\star$ we define our three parameters:

Definition 1. *The* **pathwidth** μ*, the* **slack** σ *and the* **maximal delay** ℓ_{max} *are defined as follows:* $\mu = \max_{t \geq 0} |\{j \in \mathcal{T}, r_j \leq t < \bar{d}_j\}| - 1$,
$\sigma = \max_{j \in \mathcal{T}} \bar{d}_j - r_j - p_j$*, and* $\ell_{max} = \max_{(i,j) \in arcs\ of\ G} \ell_{i,j}$.

2.2 Slack vs pathwidth

In this section we show a relation between both parameters in feasible schedules.

Theorem 1. *If a single machine scheduling instance with time windows is feasible, then* $\mu \leq 2\sigma$.

Proof. By the definition of slack σ: $r(J) \geq d(J) - p(J) - \sigma$ and $d(J) \leq r(J) + p(J) + \sigma$.

Let J be a job and t be a time slot. If t is part of interval $[r(J), d(J))$ then $r(J) \leq t < d(J)$. Then by using both inequalities on the definition of slack σ we get: $r(J) > t - p(J) - \sigma$ and $d(J) \leq t + p(J) + \sigma$.

This means that whenever job J is scheduled, $p(J)$ consecutive time slots will be taken by J in time interval $[t - \sigma - p(J) + 1, t + \sigma + p(J))$. Thus at least one time unit will be taken by J in time interval $[t - \sigma, t + \sigma + 1)$. Since we only have one machine, this means that at most $2\sigma + 1$ jobs can have t be a part of their time window $[r(J), d(J))$ in any feasible schedule of this instance. This yields the wanted inequality: $\mu \leq 2\sigma$.

2.3 Direct implications to parameters σ and $\sigma + \ell_{max}$

In previous studies [8,13] we defined FPT algorithms on single machine problems with parameter pathwidth and maximum delays. It follows from Theorem 1 that these results transfer to slack instead of pathwidth.

Corollary 1. $1|prec, r_j, \bar{d}_j|L_{max} \in FPT(\sigma)$ *[8],*
$1|prec(\ell_{i,j}^{min}), p_j = 1, r_j, \bar{d}_j|L_{max} \in FPT(\sigma + \ell_{max})$ *[13].*

3 Hardness reductions with parameters μ and σ

In this section our goal is to characterize the most simple setting for which our problem becomes para-NP-hard with respect to the slack and the pathwidth with minimum (resp. maximum, exact) delays.

In all three delay types our para-NP-hardness results will be proved with a reduction from the 3-COLORING graph problem. Let $G = (V, E)$ with $V = [0, n)$ and $E \subseteq V \times V$. Without loss of generality we suppose that there is no self loop in E. The scheduling instances described below will have fixed pathwidth and slack (namely $\mu = 3$ and $\sigma = 4$).

3.1 With minimum delays

We first establish para-NP-Hardness in the case of coupled tasks with equal minimum delays. The following theorem will be proved in the rest of the section.

Theorem 2. $1|(1, \ell^{min}, 1), r_j, \bar{d}_j|\star$ *is para-*NP*-hard parameterized by pathwidth* μ *and slack* σ*.*

We build I_{min} a scheduling instance of $1|(1, \ell^{min}, 1), r_j, \bar{d}_j|\star$ where all coupled tasks have the same minimum delay $\ell = \beta - 2$, with $\beta = 18n$. We define instance I_{min} then prove that it is feasible if and only if G is 3-colorable.

Let $\gamma = (3 + 6m + 1)\beta$. We segment time into a working space $[0, \gamma)$ and a garage space $[\gamma, 2\gamma)$. The garage space purpose is detailed later in Remark 1. The working space is split into:

- a color choice segment $[0, 3\beta)$ where all n color choices are set. A section of length $\beta = 18n$ is split into $3n$ subsections of length six, one for each couple (node, chosen color) $= (i, k)$. Several sections of length β are needed in order to connect the three possible color choices of a node and ensure that at least one color is chosen for each node - see Figure 1,
- an edge check segment $[3\beta, (3 + 6m)\beta)$. A segment of length 2β is dedicated to each of the $3m$ couples (edge, checked color) $= (e_j, k)$. For each couple we check that both nodes in e_j do not have color k - which would invalidate the 3-coloring. Again more than one section of length β is needed in order to connect both color choices.
  For each couple (e_j, k) we define $b_{j,k} = 3 + 6j + 2k$ in order to describe concisely the segment of length 2β where color k is checked for edge e_j. Then this segment can be written as $[b_{j,k} \cdot \beta, (b_{j,k} + 2) \cdot \beta)$ - see Figure 2.

In order to simulate color choices and edge checks we use five kinds of coupled tasks. The first two deal with node color choices:

- Color choice tasks $(C(i), C'(i))$: task $C(i)$ represents the color choice of node i. Only the position of $C(i)$ determines the color used. $C(i)$ has three possible starting times and each one corresponds to a color choice. As shown in Figure 1 if the starting time of $C(i)$ is:
  - $r(C(i))$ then node i has color 0,
  - $r(C(i)) + 1$ then node i has color 2,
  - $r(C(i)) + 2$ then node i has color 1.
- Color choice propagators $(P_{i,k}(b), P'_{i,k}(b))$: these tasks propagate the color choice of node i for later use in the edge check segment. They have time windows of length two and operate like a switch. With $k \in \{0, 1, 2\}$ if node i has color k then all tasks $P_{i,k}(b)$ and $P'_{i,k}(b)$ must be *pushed to the 'right'* (i.e. scheduled at their release date plus one, see Lemma 3). If color k is not chosen by node i then these tasks are free to be scheduled in either time - but preferably to the 'left', i.e. at their release date.

The three remaining coupled task kinds connect the color choices in two ways:

1. for each node connect the start of the three corresponding color choice propagator sequences so that at least one of them is scheduled to the 'right' - i.e. the corresponding color is chosen. See Lemma 2,
2. for each (edge, checked color) couple $(e_j = \{i_1, i_2\}, k)$ connect the color choice propagator sequences corresponding to the (node, chosen color) couples (i_1, k) and (i_2, k) so that both must not be scheduled to the 'right' at the same time - see Lemma 4.

Their respective purpose is detailed below:

- Local connectors $L_{i,k}(b, in)$ and $L_{i,k}(b, bridge)$: the main tasks used to connect color choices. Only the first task in the coupled task is used. For each subsection of length six we set at most two of them: a task $L_{i,k}(b, in)$ located completely inside the subsection and a task $L_{i,k}(b, bridge)$ located both in subsection (i, k) and the next one - either $(i, k+1)$ with $k \in \{0, 1\}$ or $((i+1) \ mod \ n, 0)$ with $k = 2$.
- Redirection tasks $R_{i,k}(b)$: these coupled tasks help to propagate a color choice the other way around. Something that color choice propagator sequences alone cannot perform, since they only propagate in one direction.
- Fill tasks $F_{i,k}(b)$ and $f_{i,k}(b, q)$: these tasks block one half of local connectors and redirection tasks so that they only have two possibilities in any feasible schedule - see Lemma 1. Only the first task in the coupled task is used. On the one hand tasks $F_{i,k}(b)$ have a time window of length three located at the connection between two color choice propagators. Then three tasks must be scheduled in a time window of length three, which blocks the whole window. On the other hand a task $f_{i,k}(b, q)$ has a time window of length one and blocks one time unit by itself.

The indices in the task names indicate the task location. In the case of color choice tasks like $C(i)$ index i indicates the segment $[\beta + 18i, \beta + 18(i + 1))$ in which the task is located. For all other tasks like $P_{i,k}(b)$ index b refers to section $[b \cdot \beta, (b+1) \cdot \beta)$ and couple (i, k) indicates the corresponding subsection of length six $[b \cdot \beta + 18i + 6k, b \cdot \beta + 18i + 6(k + 1))$ within this section of length β. Finally fill tasks $f_{i,k}(b, q)$ have an extra index $q \in [0, 6)$ which indicates the location of the fill task in this subsection of length six.

Remark 1. Note that only the first half of color choice tasks, local connectors and fill tasks will be used. The second half of these coupled tasks will be scheduled to the garage space - with an offset γ - so that it does not interfere with the working space.

The time windows of all tasks in I_{min} are given in Appendix A.1. Checking that both our parameters are bounded in I_{min} is also done there. Now we show that all the task properties announced above are met. First fill tasks must block one half of local connectors and redirection tasks, only leaving them two possibilities in any feasible schedule.

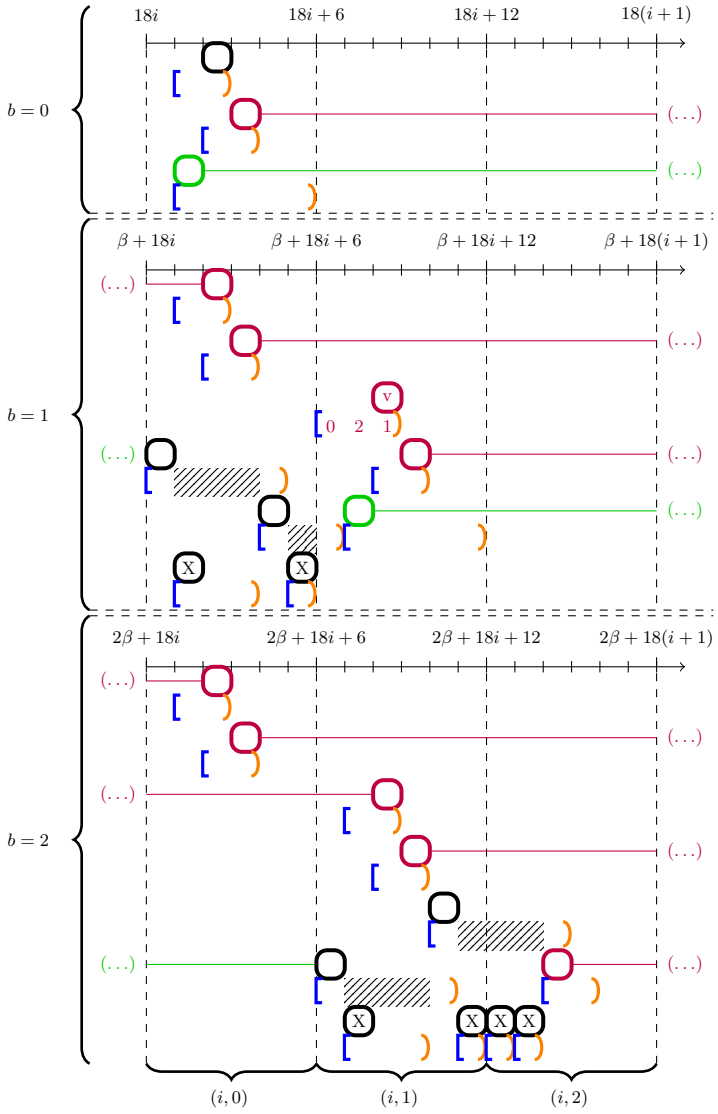

Fig. 1. The part of the color choice segment corresponding to node i in the reduction with minimum delays. In the case of color choice tasks, local connectors and fill tasks only the first task of the coupled task is shown.

Lemma 1. *Let $i \in [0, n), k \in \{0, 1, 2\}$. In any feasible, the following tasks can only be scheduled at either their release date or their deadline minus one:*

- *all local connectors $L_{i,k}(b, in)$ and $L_{i,k}(b, bridge)$,*
- *redirection tasks $R'_{i,k}(b)$ in the color choice segment and $R_{i,k}(b)$ in the edge check segment.*

Proof. See Appendix A.2. □

Second this lemma is directly used to show that each of the three possibilities of color choice $C(i)$ pushes one color choice propagator to the 'right'. See Figure 1 to visualize the local connecting process.

Lemma 2. *Let $i \in [0, n), k \in \{0, 1, 2\}$. In any feasible schedule, if color choice task $C(i)$ is scheduled at time:*

- *$r(C(i))$ then task $P_{i,0}(0)$ must be scheduled at its release date plus one,*
- *$r(C(i)) + 1$ then task $P_{i,2}(2)$ must be scheduled at its release date plus one,*
- *$r(C(i)) + 2$ then task $P_{i,1}(1)$ must be scheduled at its release date plus one.*

Proof. Let s be the starting time of $C(i)$.

Case $s = r(C(i))+2$ is straightforward. $C(i)$ blocks time $r(P_{i,1}(1))$, thus task $P_{i,1}(1)$ must be scheduled at its release date plus one.

If $s = r(C(i)) + 1$ then redirection task $R_{i,1}(1)$ must be scheduled later than its release date. Then with the minimum delay of length $\beta - 2$ starting from this task, the other half $R'_{i,1}(1)$ must also be scheduled later than its release date. By Lemma 1 this only leaves time $d(R'_{i,1}(1)) - 1$. This corresponds to the release date of local connector $L_{i,1}(2, bridge)$, which again by Lemma 1 only leaves time $d(L_{i,1}(2, bridge)) - 1$ to schedule this task. This corresponds to $r(P_{i,2}(2))$, which forces propagator $P_{i,2}(2)$ to be scheduled at its release date plus one.

Finally case $s = r(C(i))$ is proved the same way through tasks $L_{i,0}(1, bridge)$, $R'_{i,0}(0)$, $R_{i,0}(0)$, $L_{i,0}(0, in)$ and two calls from Lemma 1. □

Third when one of these three tasks $P_{i,k}(k)$ is pushed to the 'right', it pushes all other propagators $P_{i,k}(b)$ and $P'_{i,k}(b)$ to the 'right' in order to propagate this color choice to the edge check segment.

Lemma 3. *Let $i \in [0, n), k \in \{0, 1, 2\}$. In any feasible schedule, if $P_{i,k}(k)$ is scheduled at its release date plus one, then for all $b \in [k, 3 + 6m)$ tasks $P_{i,k}(b)$ and $P'_{i,k}(b)$ must be scheduled at their release date plus one.*

Proof. By induction on b. Let $b \in [k, 3+6m-1)$. Suppose that $P_{i,k}(b)$ is scheduled at its release date plus one. Then by the minimum delay of length $\beta - 2$ the other half $P'_{i,k}(b)$ must also be scheduled at its release date plus one. This corresponds to the release date of the next propagator $P_{i,k}(b + 1)$, which in turn must also be scheduled at its release date plus one. □

Finally given an edge $e_j = \{i_1, i_2\}$ and a color k both nodes i_1, i_2 must not choose color k at the same time. According to Lemma 2 we must show that both tasks $P_{i_1,k}(k)$ and $P_{i_2,k}(k)$ cannot be scheduled to their release date plus one in a feasible schedule.

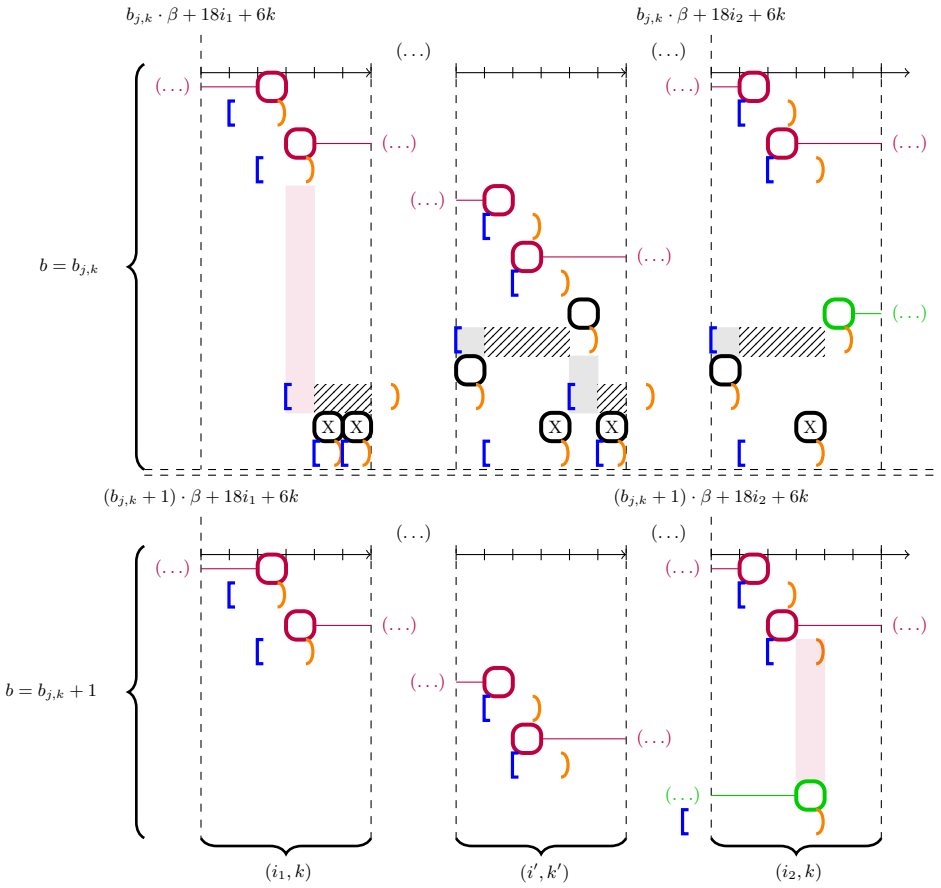

Fig. 2. The part of the edge check segment corresponding to (edge, checked color) couple (e_j, k) in the reduction with minimum delays. The middle part describes the tasks in all subsections (i', k') in between subsections (i_1, k) and (i_2, k) in the same section of length β. In the case of local connectors and fill tasks only the first task of the coupled task is shown.

Lemma 4. *Let $e_j = \{i_1, i_2\}$ be an edge in E and $k \in \{0, 1, 2\}$ be a color. In any feasible schedule, either $P_{i_1,k}(k)$ or $P_{i_2,k}(k)$ must be scheduled at its release date.*

Proof. (sketch) Consider a feasible schedule. By contradiction suppose that both tasks $P_{i_1,k}(k)$ and $P_{i_2,k}(k)$ are scheduled at their release date plus one. By Lemma 3 both tasks $P_{i_1,k}(b_{j,k})$ and $P_{i_2,k}(b_{j,k}+1)$ must also be scheduled at their release date plus one. We show that when task $P_{i_1,k}(b_{j,k})$ is scheduled at its release date plus one then time $r(P_{i_2,k}(b_{j,k}+1))+1$ is blocked, and vice versa.

This is proved by induction on subsection (i, k) between (i_1, k) and (i_2, k). See Figure 2 to visualize the local connecting process. The induction step uses

Lemma 1 the same way as in the proof of Lemma 2: the local connectors successively block one scheduling option of the adjacent local connector, which forces it into its second scheduling option. The full proof is available in Appendix A.3. □

So, in any feasible schedule, by Lemmas 2 and 3 each node must choose a color by pushing to the 'right' one of three color choice propagator sequences. Then by Lemma 4 both nodes in an edge cannot choose the same color by pushing to the 'right' their propagator sequence corresponding to the same color. This is enough to simulate a graph 3-coloring.

Proposition 1. *G is 3-colorable if and only if there exists a feasible schedule for I_{min}.*

Proof. See Appendix A.4. □

The reduction is now complete and $1|(1, \ell^{min}, 1), r_j, \bar{d_j}|\star$ has been successfully proved para-NP-hard parameterized by pathwidth μ and slack σ.

3.2 With maximum delays

In this section we now consider maximum delays. We perform a similar reduction to prove the following theorem:

Theorem 3. *$1|(1, \ell^{max}, 1), r(j), \bar{d}(j)|\star$ is para-NP-hard parameterized by pathwidth μ and slack σ.*

The full reduction is available in Appendix B. We use the same idea as in the minimum delays case with two main changes. First if we want to propagate color choices from left to right with maximum delays in place of minimum ones, then we must *pull to the 'left'* instead of pushing to the 'right'. So now a color is chosen when color choices propagators are scheduled at their release date. Plus the time windows between the end of a propagator and the beginning of the next one are swapped, again in order to pull to the 'left' instead of pushing to the 'right'.

Second in the minimum delays reduction the unused coupled task halves were set to a garage space. This second half was set far from the first half so that the minimum delay $\beta - 2$ between them could be rendered useless and ignored. In order to get the same result with maximum delays, we need the garage space to be close to these first halves instead. We do so by intertwining working and garage intervals: a working interval of length $18n$, then a garage interval of length $18n$, and so on. As a result β is doubled to $36n$, γ is set to $18n$ and the rest of the reduction with minimum delays can be reused with minimal changes.

3.3 With exact delays

In this section exact delays are considered. This case is different since problem $1|(1, \ell^{ex}, 1), r(j), \bar{d}(j)|\star$ can be solved in time $\mathcal{O}(n \cdot log(n))$. Sorting coupled tasks

by non-decreasing deadline of their first half and applying the 'earliest due date' (EDD) rule on these first halves works if and only if the instance is feasible. Indeed suppose we have a feasible schedule. In the case of equal-processing time tasks, they are identical in the time length they require and thus can be swapped until we get the task order given by the (EDD) rule. In our problem all the the coupled tasks are identical in their 'processing' time requirement and thus can be swapped the same way whenever the order given by the (EDD) rule is not followed.

However if we take a look at the instance I_{min} from the reduction with minimum delays, replacing all minimum delays with exact delays of the same value almost makes it work. Indeed Lemmas 1 to 4 still hold. Unfortunately the idea to put a coupled task half in the garage space and ignore the delay does not work anymore. There are two ways to fix it with two slight generalizations:

- either add delay value $\gamma - 1$ and use it whenever we need to set a task to the garage space. This corresponds to problem $1|(1, \{\ell_1^{ex}, \ell_2^{ex}\}, 1), r(j), \bar{d}(j)|\star$,
- or delete these second halves and the garage space entirely, and allow chains of length one in our instance. This corresponds to problem $1|chains(\ell^{ex}, len \in \{1, 2\}), p_j = 1, r(j), \bar{d}(j)|\star$.

In both cases the only detail that prevented the reduction from working is now dealt with and the para-NP-hardness proof can be done.

Theorem 4. *Scheduling problems* $1|chains(\ell^{ex}, len \in \{1, 2\}), p_j = 1, r(j), \bar{d}(j)|\star$ *and* $1|(1, \{\ell_1^{ex}, \ell_2^{ex}\}, 1), r(j), \bar{d}(j)|\star$ *are para-*NP*-hard parameterized by pathwidth* μ *and slack* σ*.*

4 A FPT algorithm with parameter $\sigma + \ell_{max}$

We consider here an instance I of the decision problem $1|prec(\ell_{i,j}), r(j), \bar{d}(j)|\star$ assuming a general precedence graph, precedence delays of all three types in the instance and any processing times of jobs. We showed that this problem is para-NP-hard with parameter σ alone. In [4] the authors noted that with exact (or maximum) delays and chain precedence this problem is also NP-hard when $\ell_{max} = 0$ - i.e para-NP-hard parameterized by ℓ_{max} - using a straightforward reduction from 3-PARTITION. In this section we define a FPT dynamic programming algorithm parameterized by $\sigma + \ell_{max}$ for this problem. Notice that this algorithm can be used within a binary search to solve in FPT time the related optimization problems minimizing the makespan $1|prec(\ell_{i,j}), r(j), \bar{d}(j)|C_{\max}$ and the maximum lateness $1|prec(\ell_{i,j}), r(j), \bar{d}(j)|L_{\max}$.

Stage α of the dynamic programming scheme corresponds to the α^{th} task to be scheduled in order from left to right. In a state u of stage α the following items are stored:

(a) the α^{th} task to be scheduled, denoted by $j(u)$;
(b) its completion time $c(u) = C_{j(u)}$;

(c) the set $A(u)$ of scheduled tasks i for which $\bar{d}(i) > c(u)$;
(d) a schedule $S(u)$ of time interval $[c(u) - \ell_{max}, c(u))$, describing all jobs that complete in this interval and their completion times.

Definition 2. *A state u is valid if there exists a feasible schedule of all jobs of $A(u) \cup \{j(u)\} \cup \{i, \bar{d}(i) \leq c(u)\}$ that coincides in its last time interval with $S(u)$. We denote by $\mathcal{V}_\alpha$ the set of valid states of stage α and by N_α the subset of jobs $j(u)$ for which there exists a valid state u of stage α.*

We now define the transitions between consecutive stages, which correspond to the edges of a multi-stage graph G_I. Let u be a valid state of stage $\alpha - 1$. To build a valid successor v of stage α we:

- choose $j(v)$ such that $j(v)$ is unscheduled, has no predecessor, or any predecessor k of $j(v)$ is completed before $c(u)$ in u: $k \in A(u)$ or $\bar{d}(k) \leq c(u)$;
- choose a completion time $c(v)$ for $j(v)$ within the time interval of $j(v)$ satisfying precedence constraints and resource requirements;
- check that a job k with $\bar{d}(k) \leq c(v)$ does not remain unscheduled (otherwise v is discarded);
- then $A(v)$ and $S(v)$ can be deduced from $j(v)$, $A(u)$ and $S(u)$.

So we can build the graph stage by stage, starting from an initial state with an empty schedule. At each stage $\alpha-1$ we consider each state u and generate only valid successors of u in stage α. Once a successor v is generated by appending a task, the algorithm checks whether the state has already been created, and if not it appends v to the list of states of stage α. The instance is feasible if and only if there is a valid state of stage n.

We now analyze the complexity of the graph construction. The detailed analysis can be found in Appendix C. We choose here to describe the core of the proof, which shows that the number of valid states is FPT.

Lemma 5. $\sum_{1 \leq \alpha \leq n} |N_\alpha| \leq [2\mu + \sigma + 1] \cdot n$.

Proof. We show that each task appears in at most $2\mu + 1 + \sigma$ stages. We do so by bounding the indices α where a task j can appear in N_α.

- Lower bound: let Y_j be the set of tasks with a deadline not greater than $r(j)$. Then in any feasible schedule all these tasks must be scheduled before j. This means that α must be greater than $|Y_j|$.
- Upper bound: let i be a task scheduled before j in some feasible schedule. Then i must be completed before the latest starting time of j, which is bounded by $r(j) + \sigma$. Now, if $i \notin Y_j$ then $\bar{d}(i)$ must be greater than $r(j)$. This means that the availability window of task i intersects with time interval $[r(j), r(j) + \sigma)$ of length σ. By Lemma 10 (see Appendix C.1) the number of such task i not included in Y_j is bounded by $2\mu + \sigma$. Thus in any feasible schedule task j must appear in a stage α lower or equal to $|Y_j| + 2\mu + \sigma + 1$.

Hence a task j can appear in at most $2\mu + \sigma + 1$ different stages α. This means that each task is included in at most $2\mu + \sigma + 1$ different sets N_α. This bounds the sum of their sizes by the wanted value. □

Then we use Lemma 5 to bound the total number of states:

Proposition 2. *The number of valid states of stage α satisfies:*

$$|\mathcal{V}_\alpha| \leq |N_\alpha| \cdot f(\sigma, \ell_{max})$$

where $f(\sigma, \ell_{max}) = (\sigma+1) \cdot 2^{2\sigma+1} \cdot (\ell_{max}+1)^{4\sigma+\ell_{max}}$. The total number of states is bounded by $(5\sigma+1) \cdot f(\sigma, \ell_{max}) \cdot n$.

Proof. First we prove the first part of the proposition part by part:

(a) By definition of N_α the number of possible tasks $j(u)$ of a state u of stage α is bounded by $|N_\alpha|$.
(b) For a given $j(u)$, by definition of slack σ there are at most $\sigma+1$ possible completion times for $j(u)$.
(c) If a job i is in $A(u)$, it was scheduled before $j(u)$ but could complete after $c(u)$. Then time slot $c(u)$ is part of the interval of i. By the definition of pathwidth μ there can only be at most $\mu+1$ of such jobs. This bounds the number of possible sets by $2^{\mu+1}$.
(d) A partial schedule of length ℓ_{max} is stored. Thus once $c(u)$ is set (from parts (a) and (b)), by Lemma 10 (see Appendix C.1) there are at most $2\mu+\ell_{max}$ different tasks which can appear in these partial schedules. For each of these tasks there are at most $\ell_{max}+1$ choices: either not be part of the partial schedule or have its completion time at one of the ℓ_{max} times in this interval. Thus the number of such partial schedules is bounded by $(\ell_{max}+1)^{2\mu+\ell_{max}}$.

Now if we add them up with all stages α, we get the same expression, except with $(\sum_{1\leq\alpha\leq n} |N_\alpha|)$ as a factor instead of $|N_\alpha|$. By Lemma 5 this sum is bounded by $(2\mu+\sigma+1) \cdot n$. Thus the total number of states is bounded by $[2\mu+\sigma+1] \cdot (\sigma+1) \cdot 2^{\mu+1} \cdot (\ell_{max}+1)^{2\mu+\ell_{max}} \cdot n$. Finally applying Theorem 1 to the three occurrences of μ in this expression achieves the proof. □

Finally we prove in Lemma 11 (see Appendix C.2) that each state has at most $\mu+1$ successors and that generating valid successors of a valid state is FPT. This lead to our main result. The proof is given in Appendix C.3.

Theorem 5. *The proposed dynamic programming algorithm is correct and is FPT with respect to parameters σ and ℓ_{max}. Its complexity is $\mathcal{O}(\sigma^6(\sigma+\ell_{max}) \cdot (2(\ell_{max}+1))^{\ell_{max}+6\sigma} \cdot n^2)$.*

5 Conclusion

We proved that no matter the delay type, even in the most simple setting, the single machine scheduling problem with time windows and precedence delays is para-NP-hard with respect to slack σ and pathwidth μ. When combining parameters slack σ and maximum precedence delay ℓ_{max} we provided a FPT algorithm. The complexity of scheduling coupled tasks with parameter ℓ_{max} alone or general instances with both parameters μ and ℓ_{max} remains open.

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), number: 2 Publisher: Elsevier
2. Bessy, S., Giroudeau, R.: Parameterized complexity of a coupled-task scheduling problem. Journal of Scheduling **22**(3), 305–313 (Jun 2019)
3. van Bevern, R., Bredereck, R., Bulteau, L., Komusiewicz, C., Talmon, N., Woeginger, G.J.: Precedence-constrained scheduling problems parameterized by partial order width. In: Kochetov, Y., Khachay, M., Beresnev, V., Nurminski, E., Pardalos, P. (eds.) Discrete Optimization and Operations Research. pp. 105–120. Springer International Publishing, Cham (2016)
4. Bodlaender, H.L., van der Wegen, M.: Parameterized complexity of scheduling chains of jobs with delays. In: 15th International Symposium on Parameterized and Exact Computation (IPEC). p. (2020)
5. Chen, B., Zhang, X.: Scheduling coupled tasks with exact delays for minimum total job completion time. J. Sched. **24**(2), 209–221 (2021)
6. Dorndorf, U., Pesch, E., Phan-Huy, T.: A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. Management Science **46**, 1365–1384 (2000)
7. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of discrete mathematics, vol. 5, pp. 287–326. Elsevier (1979)
8. Hanen, C., Mallem, M., Munier-Kordon, A.: 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). p. (2022)
9. Hanen, C., Munier-Kordon, A.: Fixed-Parameter tractability of scheduling dependent typed tasks subject to release times and deadlines. Submitted (2021)
10. Khatami, M., Salehipour, A., Cheng, T.: Coupled task scheduling with exact delays: Literature review and models. European Journal of Operational Research **282**(1), 19–39 (2020)
11. Lawler, E.: Optimal sequencing of a single machine subject to precedence constraints. Management Sci. **19**, 544–546 (1973)
12. Lenstra, J., Rinnooy Kan, A., Brucker, P.: Complexity of machine scheduling problems. Ann. of Discrete Math. **1**, 343–362 (1977)
13. Mallem, M., Hanen, C., Munier-Kordon, A.: Parameterized complexity of a parallel machine scheduling problem. In: 17th International Symposium on Parameterized and Exact Computation (IPEC). p. (2022)
14. Martel, C.U.: Preemptive scheduling with release times, deadlines, and due times. J. ACM **29**(3), 812–829 (1982)
15. Mnich, M., Wiese, A.: Scheduling and fixed-parameter tractability. Mathematical Programming **154**(1-2), 533–562 (Dec 2015)
16. 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)
17. Yu, W., Hoogeveen, H., Lenstra, J.K.: Minimizing Makespan in a Two-Machine Flow Shop with Delays and Unit-Time Operations is NP-Hard. Journal of Scheduling **7**(5), 333–348 (Sep 2004)

Appendix A Omitted details in Section 3.1

A.1 Full definition of scheduling instance I_{min}

Definition 3. *Scheduling instance I_{min} contains the following tasks:*

- *Color choice tasks $(C(i), C'(i))$ with $i \in [0, n)$*
  - *$C(i)$ with release date $\beta + 18i + 6$ and a time window of length three.*
    $C'(i)$ has the same time window with an offset γ.
- *Color choice propagators $(P_{i,k}(b), P'_{i,k}(b))$ with $i \in [0, n), k \in \{0, 1, 2\}$ and $b \in [k, 3 + 6m)$*
  - *$P_{i,k}(b)$ with release date $b \cdot \beta + 18i + 6k + 2$ and a time window of length two.*
    $P'_{i,k}(b)$ has the same time window with an offset $\beta - 1$.
- *Other tasks in color choice segment $[0, 3\beta)$*
  (see Figure 1)
  For each i in $[0, n)$:
  - *two redirection tasks: $R_{i,0}(0)$ (resp. $R_{i,1}(1)$) with release date $18i + 1$ (resp. $\beta + 18i + 1$) and a time window of length five.*
    $R'_{i,0}(0)$ (resp. $R'_{i,1}(1)$) has the same time window with an offset $\beta - 1$.
  - *two fill tasks with a time window of length three: $F_{i,0}(1)$ (resp. $F_{i,1}(2)$) with release date $\beta + 18i + 1$ (resp. $2\beta + 18i + 7$).*
    $F'_{i,0}(1)$ (resp. $F'_{i,1}(2)$) has the same time window with an offset γ.
  - *three local connectors: $L_{i,0}(0, in)$ (resp. $L_{i,0}(1, bridge)$, $L_{i,1}(2, bridge)$) with release date $18i + 1$ (resp. $\beta + 18i + 4$, $2\beta + 18i + 10$) and a time window of length two (resp. three, five).*
    $L'_{i,0}(0, in)$ (resp. $L'_{i,0}(1, bridge)$, $L'_{i,1}(2, bridge)$) has the same time window with an offset γ.
  - *four fill tasks with a time window of length one: $f_{i,0}(1, 5)$ (resp. $f_{i,1}(2, 5)$, $f_{i,2}(2, 0)$, $f_{i,2}(2, 1)$) with release date $\beta + 18i + 5$ (resp. $2\beta + 18i + 11$, $2\beta + 18i + 12$, $2\beta + 18i + 13$)*
    $f'_{i,0}(1, 5)$ (resp. $f'_{i,1}(2, 5)$, $f'_{i,2}(2, 0)$, $f'_{i,2}(2, 1)$) has the same time window with an offset γ.
- *Other tasks in edge check segment $[3\beta, (3 + 6m)\beta)$*
  (see Figure 2)
  For each couple (e_j, k) in $E \times \{0, 1, 2\}$ with $e_j = \{i_1, i_2\}, i_1 < i_2$:
  - *In subsection (i_1, k):*
    - *one local connector $L_{i_1,k}(b_{j,k}, bridge)$ with release date $b_{j,k} \cdot \beta + 18i_1 + 6k + 3$ and a time window of length four.*
      $L'_{i_1,k}(b_{j,k}, bridge)$ has the same time window with an offset γ.
    - *two fill tasks with a time window of length one: $f_{i_1,k}(b_{j,k}, 4)$ (resp. $f_{i_1,k}(b_{j,k}, 5)$) with release date $b_{j,k} \cdot \beta + 18i_1 + 6k + 4$ (resp. $b_{j,k} \cdot \beta + 18i_1 + 6k + 5$).*
      $f'_{i_1,k}(b_{j,k}, 4)$ (resp. $f'_{i_1,k}(b_{j,k}, 5)$) has the same time window with an offset γ.
  - *In the $3(i_2 - i_1) - 1$ subsections (i', k') between (i_1, k) and (i_2, k):*

  - *two local connectors:* $L_{i',k'}(b_{j,k}, in)$ *(resp.* $L_{i',k'}(b_{j,k}, bridge)$*) with release date* $b_{j,k} \cdot \beta + 18i' + 6k'$ *(resp.* $b_{j,k} \cdot \beta + 18i' + 6k' + 4$*) and a time window of length five (resp. three).*
    $L'_{i',k'}(b_{j,k}, in)$ *(resp.* $L'_{i',k'}(b_{j,k}, bridge)$*) has the same time window with an offset* γ*.*
  - *one fill task with a time window of length three:* $F_{i',k'}(b_{j,k})$ *with release date* $b_{j,k} \cdot \beta + 18i' + 6k' + 1$*.*
    $F'_{i',k'}(b_{j,k})$ *has the same time window with an offset* γ*.*
  - *one fill task with a time window of length one:* $f_{i',k'}(b_{j,k}, 5)$ *with release date* $b_{j,k} \cdot \beta + 18i' + 6k' + 5$*.*
    $f'_{i',k'}(b_{j,k}, 5)$ *has the same time window with an offset* γ*.*
- *In subsection* (i_2, k)*:*
  - *a redirection task* $R_{i_2,k}(b_{j,k})$ *with release date* $b_{j,k} \cdot \beta + 18i_2 + 6k$ *and a time window of length five.*
    $R'_{i_2,k}(b_{j,k})$ *has the same time window with an offset* $\beta - 1$*.*
  - *one fill task with a time window of length three:* $F_{i_2,k}(b_{j,k})$ *with release date* $b_{j,k} \cdot \beta + 18i_2 + 6k + 1$
    $F'_{i_2,k}(b_{j,k})$ *has the same time window with an offset* γ*.*

Remark 2. Note that all tasks have a time window of length five of less, so we have slack $\sigma = 4$. Plus at any time at most four time windows intersect. It happens at time units of the form $b \cdot \beta + 18i + 6k + 2$ with two color choice propagators, a fill task of time window length three and either a redirection task or a local connector. Thus pathwidth $\mu = 3$.

A.2 Proof of Lemma 1

Proof. We check one task at a time following Definition 3.

- Tasks in color choice segment $[0, 3\beta)$
  (see Figure 1)
  For each i in $[0, n)$:
  - Redirection task $R'_{i,0}(0)$
    It has release date $\beta + 18i$ and a time window of length five. The three tasks $F_{i,0}(1)$, $P'_{i,0}(0)$ and $P_{i,0}(1)$ must be scheduled within interval $[\beta + 18i + 1, \beta + 18i + 4)$ so only time slots $\beta + 18i$ and $\beta + 18i + 4$ are left.
  - Redirection task $R'_{i,1}(1)$
    It has release date resp. $2\beta + 18i$ and a time window of length five. The three tasks $F_{i,1}(2)$, $P'_{i,1}(1)$ and $P_{i,1}(2)$ must be scheduled within interval $[2\beta + 18i + 1, 2\beta + 18i + 4)$ so only time slots $2\beta + 18i$ and $2\beta + 18i + 4$ are left.
  - Local connector $L_{i,0}(0, in)$
    It has release date $18i + 1$ and already a time window of length two, so nothing to be done here.
  - Local connector $L_{i,0}(1, bridge)$
    It has release date $\beta + 18i + 4$ and a time window of length three. One task $f'_{i,0}(1, 5)$ must be scheduled at time $\beta + 18i + 5$ so only time slots $\beta + 18i + 4$ and $\beta + 18i + 6$ are left.

- Local connector $L_{i,1}(2, bridge)$
  It has release date $2\beta+18i+10$ and a time window of length five. Three fill tasks $f_{i,1}(2,5)$, $f_{i,2}(2,0)$ and $f_{i,2}(2,1)$) must be scheduled at time slots $2\beta+18i+11$, $2\beta+18i+12$ and $2\beta+18i+13$ so only time slots $2\beta+18i+10$ and $2\beta+18i+14$ are left.

– Tasks in edge check segment $[3\beta, (3+6m)\beta)$
(see Figure 2)
For each couple (e_j, k) in $E \times \{0,1,2\}$ with $e_j = \{i_1, i_2\}, i_1 < i_2$:

- In subsection (i_1, k):
  * Local connector $L_{i_1,k}(b_{j,k}, bridge)$
    It has release date $b_{j,k}\cdot\beta+18i_1+6k+3$ and a time window of length four. The two tasks $f_{i_1,k}(b_{j,k},4)$ and $f_{i_1,k}(b_{j,k},5)$ must be scheduled at time slots $b_{j,k}\cdot\beta+18i_1+6k+4$ and $b_{j,k}\cdot\beta+18i_1+6k+5$ so only time slots $b_{j,k}\cdot\beta+18i_1+6k+3$ and $b_{j,k}\cdot\beta+18i_1+6k+6$ are left.
- In the $3(i_2-i_1)-1$ subsections (i',k') between (i_1,k) and (i_2,k):
  * Local connector $L_{i',k'}(b_{j,k}, in)$
    It has release date $b_{j,k}\cdot\beta+18i'+6k'$ and a time window of length five. Fill task $F_{i',k'}(b_{j,k})$ and two color choice propagators must be scheduled within interval $[b_{j,k}\cdot\beta+18i'+6k'+1, b_{j,k}\cdot\beta+18i'+6k'+4)$ so only time slots $b_{j,k}\cdot\beta+18i'+6k'$ and $b_{j,k}\cdot\beta+18i'+6k'+4$ are left.
  * Local connector $L_{i',k'}(b_{j,k}, bridge)$
    It has release date $b_{j,k}\cdot\beta+18i'+6k'+4$ and a time window of length three. Task $f_{i',k'}(b_{j,k},5)$ must be scheduled at time $b_{j,k}\cdot\beta+18i'+6k'+5$ so only time slots $b_{j,k}\cdot\beta+18i'+6k'+4$ and $b_{j,k}\cdot\beta+18i'+6(k'+1)$ are left.
- In subsection (i_2,k):
  * Redirection task $R_{i_2,k}(b_{j,k})$
    It has release date $b_{j,k}\cdot\beta+18i_2+6k$ and a time window of length five. Task $R'_{i_2,k}(b_{j,k})$ and two color choice propagators must be scheduled within interval $[b_{j,k}\cdot\beta+18i_2+6k+1, b_{j,k}\cdot\beta+18i_2+6k+4)$ so only time slots $b_{j,k}\cdot\beta+18i_2+6k$ and $b_{j,k}\cdot\beta+18i_2+6k+4$ are left. □

A.3 Full proof of Lemma 4

Proof. Consider a feasible schedule. By contradiction suppose that both tasks $P_{i_1,k}(k)$ and $P_{i_2,k}(k)$ are scheduled at their release date plus one. By Lemma 3 both tasks $P_{i_1,k}(b_{j,k})$ and $P_{i_2,k}(b_{j,k}+1)$ must also be scheduled at their release date plus one. We show that when task $P_{i_1,k}(b_{j,k})$ is scheduled at its release date plus one then time $r(P_{i_2,k}(b_{j,k}+1))+1$ is blocked.

First we prove by induction on subsection (i',k') between (i_1,k) and (i_2,k) that in segment $[b_{j,k}\cdot\beta, (b_{j,k}+1)\cdot\beta)$ all local connectors from subsection (i',k') must be scheduled at their deadline minus one. See Figure 2 to visualize the local connecting process.

- Initialization: $r(P_{i_1,k}(b_{j,k})) + 1$ corresponds to the release date of local connector $L_{i_1,k}(b_{j,k}, bridge)$. Thus by Lemma 1 only its deadline minus one is left.
- Induction step: let (i', k') be a subsection between (i_1, k) and (i_2, k). Suppose that the result holds in the previous subsection. Then by the induction hypothesis the bridge local connector from that subsection must be scheduled at its deadline minus one. The time slot is $b_{j,k} \cdot \beta + 18i' + 6k'$ which corresponds to the release date of local connector $L_{i',k'}(b_{j,k}, in)$. Thus by Lemma 1 only its deadline minus one is left. The time slot is $b_{j,k} \cdot \beta + 18i' + 6k' + 4$ which corresponds to the release date of local connector $L_{i',k'}(b_{j,k}, bridge)$. Thus by Lemma 1 only its deadline minus one is left.

This concludes the induction and shows that the bridge local connector between (i_2, k) and its previous subsection must be scheduled at its deadline minus one. The time slot is $b_{j,k} \cdot \beta + 18i_2 + 6k$ which corresponds to the release date of redirection task $R_{i_2,k}(b_{j,k})$. Thus by Lemma 1 this only leaves its deadline minus one. Then by the minimum delay of length $\beta - 2$ starting from $R_{i_2,k}(b_{j,k})$, redirection task $R'_{i_2,k}(b_{j,k})$ must also be scheduled at its deadline minus one. The time slot is $(b_{j,k}+1)\cdot\beta+18i_2+6k+3$ which corresponds to $r(P_{i_2,k}(b_{j,k}+1))+1$.

Thus both tasks $R'_{i_2,k}(b_{j,k})$ and $P_{i_2,k}(b_{j,k} + 1)$ must be scheduled at time $r(P_{i_2,k}(b_{j,k} + 1)) + 1$. However this is a single machine instance, which leads to a contradiction. This means that in any feasible schedule either $P_{i_1,k}(k)$ or $P_{i_2,k}(k)$ is not scheduled at their release date plus one. □

A.4 Proof of Proposition 1

Proof.

($\Longleftarrow$) Suppose there exists a feasible schedule for instance I_{min}. Let s be the starting times of such a schedule. We propose a 3-coloring candidate $(c(i))_{0\leq i<n}$ based on the starting times of color choice tasks $C(i)$. For i in $[0, n)$ if $s(C(i))$ is equal to:

- $r(C(i))$ then we set $c(i) = 0$,
- $r(C(i)) + 1$ then we set $c(i) = 2$,
- $r(C(i)) + 2$ then we set $c(i) = 1$.

We pretend that $(c(i))_{0\leq i<n}$ is a valid 3-coloring of graph G. By contradiction suppose that it is not a valid 3-coloring. Then there is an edge $e_j = \{i_1, i_2\}$ such that $c(i_1) = c(i_2) = k$ with k a color in $\{0, 1, 2\}$. Say we have $i_1 < i_2$ without loss of generality. Then by Lemma 2 both color choice propagators $P_{i_1,k}(k)$ and $P_{i_2,k}(k)$ must be scheduled at their release date plus one. This contradicts with Lemma 4 where it was proved that this should not happen in a feasible schedule.

($\Longrightarrow$) Suppose there exists a 3-coloring $(c(i))_{0\leq i<n}$ for graph G. We propose the following schedule s based on this 3-coloring:

- For $i \in [0, n)$ color choice task $C(i)$ is scheduled at time:
  - $\beta + 18i + 6$ if $c(i) = 0$,

  - $\beta + 18i + 8$ if $c(i) = 1$,
  - $\beta + 18i + 7$ if $c(i) = 2$,

  The other end $C'(i)$ is scheduled γ time units later.
- Let $i \in [0, n), k \in \{0, 1, 2\}$. For all $b \in [k, 3+6m)$ color choice propagators $P_{i,k}(b)$ and $P'_{i,k}(b)$ are scheduled at:
  - their release date plus one if $c(i) = k$,
  - their release date otherwise.
- Other tasks in color choice segment $[0, 3\beta)$
  For each i in $[0, n)$:
  - Local connector $L_{i,0}(1, bridge)$ (resp. fill task $F_{i,0}(1)$, redirection task $R_{i,0}(0)$) is scheduled at its release date if $c(i) = 0$ and at its deadline minus one otherwise.
    The other end $L'_{i,0}(1, bridge)$ (resp. $F'_{i,0}(1)$, $R'_{i,0}(0)$) is scheduled γ (resp. γ, $\beta - 1$) time units later.
  - Local connector $L_{i,0}(0, in)$ is scheduled at its deadline minus one if $c(i) = 0$ and at its release date otherwise.
    The other end $L'_{i,0}(0, in)$ is scheduled γ time units later.
  - Fill task $f_{i,0}(1, 5)$ is scheduled at its only available time slot.
    The other end $f'_{i,0}(1, 5)$ is scheduled γ time units later.
  - Fill task $F_{i,1}(2)$ is scheduled at its release date if $c(i) = 1$ and at its deadline minus one otherwise.
    The other end $F'_{i,1}(2)$ is scheduled γ time units later.
  - Redirection task $R_{i,1}(1)$ (resp. local connector $L_{i,1}(2, bridge)$) is scheduled at its deadline minus one if $c(i) = 2$ and at its release date otherwise.
    The other end $R'_{i,1}(1)$ (resp. $L'_{i,1}(2, bridge)$) is scheduled $\beta - 1$ (resp. γ) time units later.
  - Fill task $f_{i,1}(2, 5)$ (resp. $f_{i,2}(2, 0)$, $f_{i,2}(2, 1)$) is scheduled at its only available time slot.
    The other end $f'_{i,1}(2, 5)$ (resp. $f'_{i,2}(2, 0)$, $f'_{i,2}(2, 1)$) is scheduled γ time units later.
- Other tasks in edge check segment $[3\beta, (3 + 6m)\beta)$
  For each couple (e_j, k) in $E \times \{0, 1, 2\}$ with $e_j = \{i_1, i_2\}, i_1 < i_2$:
  - In subsection (i_1, k):
    - Local connector $L_{i_1,k}(b_{j,k}, bridge)$ is scheduled at its deadline minus one if $c(i_1) = k$ and at its release date otherwise.
      The other end $L'_{i_1,k}(b_{j,k}, bridge)$ is scheduled γ time units later.
    - Fill task $f_{i_1,k}(b_{j,k}, 4)$ (resp. $f_{i_1,k}(b_{j,k}, 5)$) is scheduled at its only available time slot.
      The other end $f'_{i_1,k}(b_{j,k}, 4)$ (resp. $f'_{i_1,k}(b_{j,k}, 5)$) is scheduled γ time units later.
  - In the $3(i_2 - i_1) - 1$ subsections (i', k') between (i_1, k) and (i_2, k):
    - Local connector $L_{i',k'}(b_{j,k}, in)$ (resp. fill task $F_{i',k'}(b_{j,k})$, local connector $L_{i',k'}(b_{j,k}, bridge)$) is scheduled at its deadline minus one if $c(i_1) = k$ and at its release date otherwise.
      The other end $L'_{i',k'}(b_{j,k}, in)$ (resp. $F'_{i',k'}(b_{j,k})$, $L'_{i',k'}(b_{j,k}, bridge)$) is scheduled γ time units later.

· Fill task $f_{i',k'}(b_{j,k}, 5)$ is scheduled at its only available time slot. The other end $f'_{i',k'}(b_{j,k}, 5)$ is scheduled γ time units later.

∗ In subsection (i_2, k):

· Redirection task $R_{i_2,k}(b_{j,k})$ (resp. $F_{i_2,k}(b_{j,k})$) is scheduled at its deadline minus one if $c(i_1) = k$ and at its release date otherwise. The other end $R'_{i_2,k}(b_{j,k})$ (resp. $F'_{i_2,k}(b_{j,k})$) is scheduled $\beta - 1$ (resp. γ) time units later.

We show that schedule s is feasible. Note that color choice propagators do not interfere with each other. The time windows of propagators related to different couples (i, k) are disjoint, and the propagators related to a common couple (i, k) are all scheduled the same way in schedule s: at their release date plus one if $c(i) = k$, and at their release date otherwise. Each propagator has its own release date, so there is no conflict between them.

Moreover the garage space is a replica of the working space with an offset γ but with strictly less tasks - no color choice propagators and redirection tasks. So only the working space needs to be considered. We check all subsections of length six where a task $C(i)$, $R_{i,k}(b)$, $R'_{i,k}(b)$, $L_{i,k}(b, in)$, $L_{i,k}(b, bridge)$, $F_{i,k}(b)$ or $f_{i,k}(b, q)$ appears.

First we consider color choice segment $[0, 3\beta)$ (see Figure 1). Let $i \in [0, n)$.

- Subsection $[18i, 18i + 6)$
  - ∗ Case $c(i) = 0$
    Redirection task $R_{i,0}(0)$ is scheduled at its release date $18i+1$ while local connector $L_{i,0}(0, in)$ (resp. propagator $P_{i,0}(0)$) is scheduled at time $18i + 2$ (resp. $18i + 3$). No conflicts here.
  - ∗ Case $c(i) \neq 0$
    Redirection task $R_{i,0}(0)$ is scheduled at its deadline minus one $18i+5$ while local connector $L_{i,0}(0, in)$ (resp. propagator $P_{i,0}(0)$) is scheduled at time $18i + 1$ (resp. $18i + 2$). No conflicts here either.
- Subsection $[\beta + 18i, \beta + 18i + 6)$
  - ∗ Case $c(i) = 0$
    According to Definition 3 we have:
    · $s(R'_{i,0}(0)) = \beta + 18i$,
    · $s(F_{i,0}(1)) = \beta + 18i + 1$,
    · $s(P'_{i,0}(0)) = \beta + 18i + 2$,
    · $s(P_{i,0}(1)) = \beta + 18i + 3$,
    · $s(L_{i,0}(1, bridge)) = \beta + 18i + 4$,
    · $s(f_{i,0}(1, 5)) = \beta + 18i + 5$.
  - ∗ Case $c(i) \neq 0$
    According to Definition 3 we have:
    · $s(P'_{i,0}(0)) = \beta + 18i + 1$,
    · $s(P_{i,0}(1)) = \beta + 18i + 2$,
    · $s(F_{i,0}(1)) = \beta + 18i + 3$,
    · $s(R'_{i,0}(0)) = \beta + 18i + 4$,
    · $s(f_{i,0}(1, 5)) = \beta + 18i + 5$.
    · $s(L_{i,0}(1, bridge)) = \beta + 18i + 6$,

  No conflicts in either case.

- Subsection $[\beta + 18i + 6, \beta + 18i + 12)$
  - Case $c(i) = 0$
    According to Definition 3 we have:
    - $s(L_{i,0}(1, bridge)) = \beta + 18i + 4$,
    - $s(C(i)) = \beta + 18i + 6$,
    - $s(R_{i,1}(1)) = \beta + 18i + 7$,
    - $s(P_{i,1}(1)) = \beta + 18i + 8$.
  - Case $c(i) = 2$
    According to Definition 3 we have:
    - $s(L_{i,0}(1, bridge)) = \beta + 18i + 6$,
    - $s(C(i)) = \beta + 18i + 7$,
    - $s(P_{i,1}(1)) = \beta + 18i + 8$,
    - $s(R_{i,1}(1)) = \beta + 18i + 11$.
  - Case $c(i) = 1$
    According to Definition 3 we have:
    - $s(L_{i,0}(1, bridge)) = \beta + 18i + 6$,
    - $s(R_{i,1}(1)) = \beta + 18i + 7$,
    - $s(C(i)) = \beta + 18i + 8$,
    - $s(P_{i,1}(1)) = \beta + 18i + 9$.

  No conflicts in either case.
- Subsection $[2\beta + 18i + 6, 2\beta + 18i + 12)$
  - Case $c(i) = 0$
    According to Definition 3 we have:
    - $s(R'_{i,1}(1)) = 2\beta + 18i + 6$,
    - $s(P'_{i,1}(1)) = 2\beta + 18i + 7$,
    - $s(P_{i,1}(2)) = 2\beta + 18i + 8$,
    - $s(F_{i,1}(2)) = 2\beta + 18i + 9$,
    - $s(L_{i,1}(2, bridge)) = 2\beta + 18i + 10$,
    - $s(f_{i,1}(2, 5)) = 2\beta + 18i + 11$.
  - Case $c(i) = 1$
    The three following tasks trade places compared to the previous case:
    - $s(F_{i,1}(2)) = 2\beta + 18i + 7$,
    - $s(P'_{i,1}(1)) = 2\beta + 18i + 8$,
    - $s(P_{i,1}(2)) = 2\beta + 18i + 9$.
  - Case $c(i) = 2$
    According to Definition 3 we have:
    - $s(F_{i,1}(2)) = 2\beta + 18i + 7$,
    - $s(P'_{i,1}(1)) = 2\beta + 18i + 8$,
    - $s(P_{i,1}(2)) = 2\beta + 18i + 9$,
    - $s(R'_{i,1}(1)) = 2\beta + 18i + 10$,
    - $s(f_{i,1}(2, 5)) = 2\beta + 18i + 11$,
    - $s(L_{i,1}(2, bridge)) = 2\beta + 18i + 14$.

  No conflicts in either case.
- Subsection $[2\beta + 18i + 12, 2\beta + 18(i + 1))$
  - Case $c(i) = 2$
    According to Definition 3 we have:

- $s(f_{i,2}(2,0)) = 2\beta + 18i + 12$,
- $s(f_{i,2}(2,1)) = 2\beta + 18i + 13$,
- $s(L_{i,1}(2, bridge)) = 2\beta + 18i + 14$,
- $s(P_{i,2}(2)) = 2\beta + 18i + 15$.

∗ Case $c(i) \neq 2$

According to Definition 3 we have:

- $s(L_{i,1}(2, bridge)) = 2\beta + 18i + 10$,
- $s(f_{i,2}(2,0)) = 2\beta + 18i + 12$,
- $s(f_{i,2}(2,1)) = 2\beta + 18i + 13$,
- $s(P_{i,2}(2)) = 2\beta + 18i + 14$.

No conflicts in either case.

Thus no conflict emerges in the color choice segment. Now we consider edge check segment $[3\beta, (3+6m)\beta)$ (see Figure 2). Let $e_j = \{i_1, i_2\}$ in E with $i_1 < i_2$.

- Subsection $[b_{j,k} \cdot \beta + 18i_1 + 6k, b_{j,k} \cdot \beta + 18i_1 + 6k + 6)$

∗ Case $c(i_1) = k$

According to Definition 3 we have:

- $s(P'_{i_1,k}(b_{j,k} - 1)) = b_{j,k} \cdot \beta + 18i_1 + 6k + 2$,
- $s(P_{i_1,k}(b_{j,k})) = b_{j,k} \cdot \beta + 18i_1 + 6k + 3$,
- $s(f_{i_1,k}(b_{j,k}, 4)) = b_{j,k} \cdot \beta + 18i_1 + 6k + 4$,
- $s(f_{i_1,k}(b_{j,k}, 5)) = b_{j,k} \cdot \beta + 18i_1 + 6k + 5$,
- $s(L_{i_1,k}(b_{j,k}, bridge)) = b_{j,k} \cdot \beta + 18i_1 + 6k + 6$.

∗ Case $c(i_1) \neq k$

According to Definition 3 we have:

- $s(P'_{i_1,k}(b_{j,k} - 1)) = b_{j,k} \cdot \beta + 18i_1 + 6k + 1$,
- $s(P_{i_1,k}(b_{j,k})) = b_{j,k} \cdot \beta + 18i_1 + 6k + 2$,
- $s(L_{i_1,k}(b_{j,k}, bridge)) = b_{j,k} \cdot \beta + 18i_1 + 6k + 3$,
- $s(f_{i_1,k}(b_{j,k}, 4)) = b_{j,k} \cdot \beta + 18i_1 + 6k + 4$,
- $s(f_{i_1,k}(b_{j,k}, 5)) = b_{j,k} \cdot \beta + 18i_1 + 6k + 5$.

No conflicts in either case.

- Subsection $[b_{j,k} \cdot \beta + 18i' + 6k', b_{j,k} \cdot \beta + 18i' + 6k' + 6)$ with (i', k') in between (i_1, k) and (i_2, k)

The scheduling of tasks $P'_{i',k'}(b_{j,k} - 1))$, $P_{i',k'}(b_{j,k}))$ and $F_{i',k'}(b_{j,k})$ only depend on whether $c(i') = k'$ or not. If not, then these tasks are scheduled in this order within time segment $[b_{j,k} \cdot \beta + 18i' + 6k' + 1, b_{j,k} \cdot \beta + 18i' + 6k' + 4)$. If we do have $c(i') = k'$ then $F_{i',k'}(b_{j,k})$ is scheduled first followed by $P'_{i',k'}(b_{j,k} - 1))$ then $P_{i',k'}(b_{j,k}))$.

Now we take a look at the four remaining tasks (which are never scheduled in this previous time interval of length three). Let (i'_{prev}, k'_{prev}) be the couple right before (i', k') (either $(i' - 1, 2)$ if $k = 0$ or $(i', k' - 1)$ otherwise).

∗ Case $c(i_1) = k$

According to Definition 3 we have:

- $s(L_{i'_{prev},k'_{prev}}(b_{j,k}, bridge)) = b_{j,k} \cdot \beta + 18i' + 6k'$,
- $s(L_{i',k'}(b_{j,k}, in)) = b_{j,k} \cdot \beta + 18i' + 6k' + 4$,
- $s(f_{i',k'}(b_{j,k}, 5)) = b_{j,k} \cdot \beta + 18i' + 6k' + 5$,

  - $s(L_{i',k'}(b_{j,k}, bridge)) = b_{j,k} \cdot \beta + 18i' + 6k' + 6$.
* Case $c(i_1) \neq k$
  According to Definition 3 we have:
  - $s(L_{i'_{prev},k'_{prev}}(b_{j,k}, bridge)) < b_{j,k} \cdot \beta + 18i' + 6k'$,
  - $s(L_{i',k'}(b_{j,k}, in)) = b_{j,k} \cdot \beta + 18i' + 6k'$,
  - $s(L_{i',k'}(b_{j,k}, bridge)) = b_{j,k} \cdot \beta + 18i' + 6k' + 4$,
  - $s(f_{i',k'}(b_{j,k}, 5)) = b_{j,k} \cdot \beta + 18i' + 6k' + 5$.

  No conflicts in either case.

- Subsection $[b_{j,k} \cdot \beta + 18i_2 + 6k, b_{j,k} \cdot \beta + 18i_2 + 6k + 6)$
  Again the scheduling of tasks $P'_{i_2,k}(b_{j,k} - 1))$, $P_{i_2,k}(b_{j,k}))$ and $F_{i_2,k}(b_{j,k})$ only depend on whether $c(i_2) = k'$ or not. If not, then these tasks are scheduled in this order within time segment $[b_{j,k} \cdot \beta + 18i_2 + 6k + 1, b_{j,k} \cdot \beta + 18i_2 + 6k + 4)$. If we do have $c(i_2) = k'$ then $F_{i_2,k}(b_{j,k})$ is scheduled first followed by $P'_{i_2,k}(b_{j,k} - 1))$ then $P_{i_2,k}(b_{j,k}))$.
  Now we take a look at the two remaining tasks (which are never scheduled in this previous time interval of length three). Let $(i_{2,prev}, k_{prev})$ be the couple right before (i_2, k) (either $(i_2 - 1, 2)$ if $k = 0$ or $(i_2, k - 1)$ otherwise).
  * Case $c(i_1) = k$
    According to Definition 3 we have:
    - $s(L_{i_{2,prev},k_{prev}}(b_{j,k}, bridge)) = b_{j,k} \cdot \beta + 18i_2 + 6k$,
    - $s(R_{i_2,k}(b_{j,k})) = b_{j,k} \cdot \beta + 18i_2 + 6k + 4$.
  * Case $c(i_1) \neq k$
    According to Definition 3 we have:
    - $s(L_{i_{2,prev},k_{prev}}(b_{j,k}, bridge)) < b_{j,k} \cdot \beta + 18i_2 + 6k$,
    - $s(R_{i_2,k}(b_{j,k})) = b_{j,k} \cdot \beta + 18i_2 + 6k$.

  No conflicts in either case.

- Subsection $[(b_{j,k} + 1) \cdot \beta + 18i_2 + 6k, (b_{j,k} + 1) \cdot \beta + 18i_2 + 6k + 6)$
  * Case $c(i_1) = k$
    $(c(i))_{0 \leq i < n}$ **is a 3-coloring.** Since $c(i_1) = k$ and $e_j = \{i_1, i_2\}$ is an edge in E, this means that $c(i_2) \neq k$. Then according to Definition 3 we have:
    - $s(P'_{i_2,k}(b_{j,k})) = (b_{j,k} \cdot \beta + 18i_2 + 6k + 1$,
    - $s(P_{i_2,k}(b_{j,k} + 1)) = b_{j,k} \cdot \beta + 18i_2 + 6k + 2$,
    - $s(R'_{i_2,k}(b_{j,k}, in)) = b_{j,k} \cdot \beta + 18i_2 + 6k + 3$.
  * Case $c(i_1) \neq k$
    Task $R'_{i_2,k}(b_{j,k}, in)$ is scheduled at its release date which is located before this subsection. So only propagators $P'_{i_2,k}(b_{j,k})$ and $P_{i_2,k}(b_{j,k}+1)$ are left. And we already established that they do not interfere with each other.

  No conflicts in either case.

Thus no conflict emerges in the edge check segment either. The only segment left is $[(3 + 6m)\beta, (3 + 6m + 1)\beta)$. The only tasks there are propagators $P'_{i,k}(3 + 6m)$ with $(i, k) \in [0, n) \times \{0, 1, 2\}$. Their times windows are disjoint so no conflict emerges in this segment either. All tasks in I_{min} were scheduled with no conflict anywhere. Hence schedule s is feasible. □

Appendix B Full reduction from Section 3.2

Let $\beta = 36n$ and $\gamma = 18n$. We define a scheduling instance I_{max} of $1|(1, \ell^{max}, 1), r_j, \bar{d_j}|\star$ with a maximum delay $\ell = \beta$ in all coupled tasks.

Definition 4. *Scheduling instance I_{max} contains the following tasks:*

- *Color choice tasks $(C(i), C'(i))$ with $i \in [0, n)$*
  - *$C(i)$ with release date $\beta + 18i + 9$ and a time window of length three.*
    $C'(i)$ has the same time window with an offset γ.
- *Color choice propagators $(P_{i,k}(b), P'_{i,k}(b))$ with $i \in [0, n), k \in \{0, 1, 2\}$ and $b \in [2 - k, 3 + 6m)$*
  - *$P_{i,k}(b)$ with release date $b \cdot \beta + 18i + 6k + 2$ and a time window of length two.*
    $P'_{i,k}(b)$ has the same time window with an offset $\beta + 1$.
- *Other tasks in color choice segment $[0, 3\beta)$*
  (see Figure 3)
  For each i in $[0, n)$:
  - *two redirection tasks: $R_{i,2}(0)$ (resp. $R_{i,1}(1)$) with release date $18i + 12$ (resp. $\beta + 18i + 6$) and a time window of length five.*
    $R'_{i,2}(0)$ (resp. $R'_{i,1}(1)$) has the same time window with an offset $\beta + 1$.
  - *two fill tasks with a time window of length three: $F_{i,2}(1)$ (resp. $F_{i,1}(2)$) with release date $\beta + 18i + 14$ (resp. $2\beta + 18i + 8$).*
    $F'_{i,2}(1)$ (resp. $F'_{i,1}(2)$) has the same time window with an offset γ.
  - *three local connectors: $L_{i,2}(0, in)$ (resp. $L_{i,1}(1, bridge)$, $L_{i,0}(2, bridge)$) with release date $18i + 15$ (resp. $\beta + 18i + 11$, $2\beta + 18i + 3$) and a time window of length two (resp. three, five).*
    $L'_{i,2}(0, in)$ (resp. $L'_{i,1}(1, bridge)$, $L'_{i,0}(2, bridge)$) has the same time window with an offset γ.
  - *four fill tasks with a time window of length one: $f_{i,2}(1, 5)$ (resp. $f_{i,0}(2, 4)$, $f_{i,0}(2, 5)$, $f_{i,1}(2, 0)$) with release date $\beta + 18i + 12$ (resp. $2\beta + 18i + 4$, $2\beta + 18i + 5$, $2\beta + 18i + 6$)*
    $f'_{i,2}(1, 5)$ (resp. $f'_{i,0}(2, 4)$, $f'_{i,0}(2, 5)$, $f'_{i,1}(2, 0)$) has the same time window with an offset γ.
- *Other tasks in edge check segment $[3\beta, (3 + 6m)\beta)$*
  (see Figure 4)
  For each couple (e_j, k) in $E \times \{0, 1, 2\}$ with $e_j = \{i_1, i_2\}, i_1 < i_2$:
  - *In subsection (i_1, k):*
    - *a redirection task $R_{i_1,k}(b_{j,k})$ with release date $b_{j,k} \cdot \beta + 18i_1 + 6k + 1$ and a time window of length five.*
      $R'_{i_1,k}(b_{j,k})$ has the same time window with an offset $\beta + 1$.
    - *one fill task with a time window of length three: $F_{i_1,k}(b_{j,k})$ with release date $b_{j,k} \cdot \beta + 18i_1 + 6k + 2$.*
      $F'_{i_1,k}(b_{j,k})$ has the same time window with an offset γ.
    - *one local connector $L_{i_1,k}(b_{j,k}, bridge)$ with release date $b_{j,k} \cdot \beta + 18i_1 + 6k + 5$ and a time window of length three.*
      $L'_{i_1,k}(b_{j,k}, bridge)$ has the same time window with an offset γ.

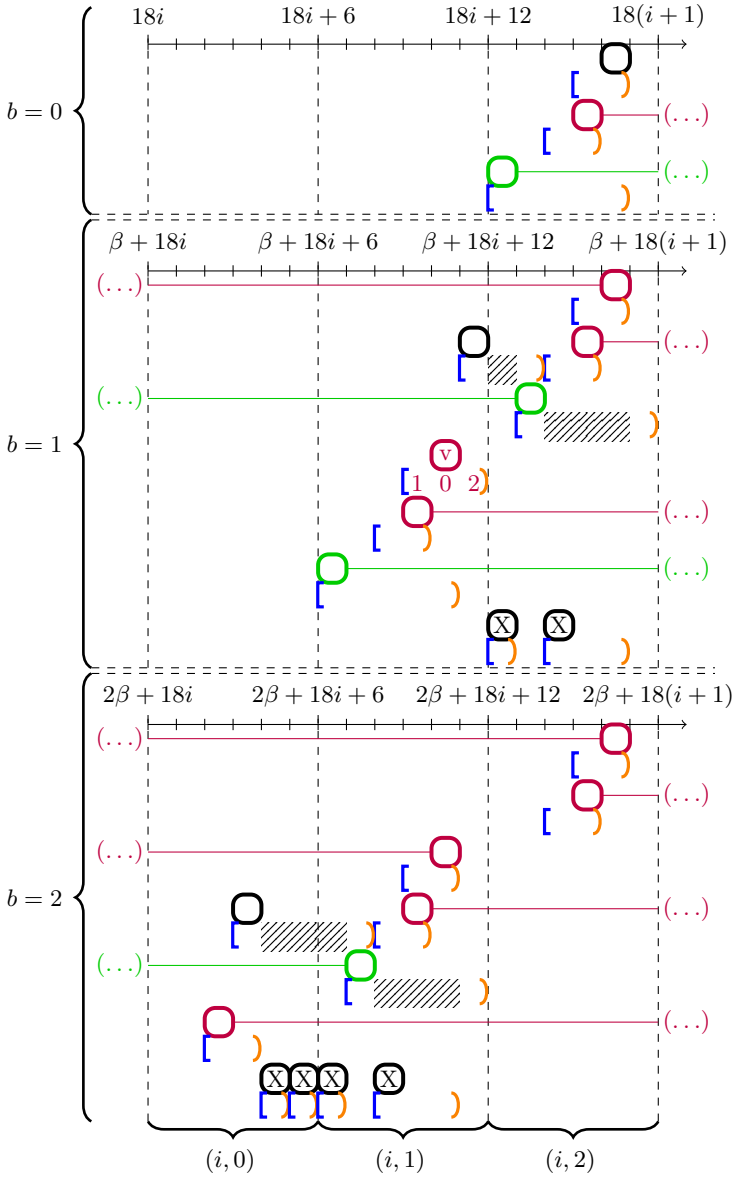

Fig. 3. The part of the color choice segment corresponding to node i in the reduction with maximum delays. In the case of color choice tasks, local connectors and fill tasks only the first task of the coupled task is shown.

- *In the* $3(i_2 - i_1) - 1$ *subsections* (i', k') *between* (i_1, k) *and* (i_2, k)*:*
  In all these subsections except the one right before (i_2, k)*:*

  - *one fill task with a time window of length one:* $f_{i',k'}(b_{j,k}, 0)$ *with release date* $b_{j,k} \cdot \beta + 18i' + 6k'$.
    $f'_{i',k'}(b_{j,k}, 0)$ *has the same time window with an offset* γ.
  - *two local connectors:* $L_{i',k'}(b_{j,k}, in)$ *(resp.* $L_{i',k'}(b_{j,k}, bridge)$*) with release date* $b_{j,k} \cdot \beta + 18i' + 6k' + 1$ *(resp.* $b_{j,k} \cdot \beta + 18i' + 6k' + 5$*) and a time window of length five (resp. three).*
    $L'_{i',k'}(b_{j,k}, in)$ *(resp.* $L'_{i',k'}(b_{j,k}, bridge)$*) has the same time window with an offset* γ.
  - *one fill task with a time window of length three:* $F_{i',k'}(b_{j,k})$ *with release date* $b_{j,k} \cdot \beta + 18i' + 6k' + 2$.
    $F'_{i',k'}(b_{j,k})$ *has the same time window with an offset* γ.

  In the subsection right before (i_2, k)*:*
  - *the same tasks with the same time windows are set, except the bridge local connector which has a time window of length four instead of three.*

- *In subsection* (i_2, k)*:*
  - *two fill tasks with a time window of length one:* $f_{i_2,k}(b_{j,k}, 0)$ *(resp.* $f_{i_2,k}(b_{j,k}, 1)$*) with release date* $b_{j,k} \cdot \beta + 18i_2 + 6k$ *(resp.* $b_{j,k} \cdot \beta + 18i_2 + 6k + 1$*).*
    $f'_{i_2,k}(b_{j,k}, 0)$ *(resp.* $f'_{i_2,k}(b_{j,k}, 1)$*) has the same time window with an offset* γ.

Remark 3. Note that all tasks have a time window of length five of less, so we have slack $\sigma = 4$. Plus at any time at most four time windows intersect. It happens at time units of the form $b \cdot \beta + 18i + 6k + 3$ with two color choice propagators, a fill task of time window length three and either a redirection task or a local connector. Thus pathwidth $\mu = 3$.

Lemma 6. *Let* $i \in [0, n), k \in \{0, 1, 2\}$*. In any feasible, the following tasks can only be scheduled at either their release date or their deadline minus one:*

- *all local connectors* $L_{i,k}(b, in)$ *and* $L_{i,k}(b, bridge)$*,*
- *redirection tasks* $R'_{i,k}(b)$ *in the color choice segment and* $R_{i,k}(b)$ *in the edge check segment.*

Proof. (sketch) We check one task at a time the same way as in the proof of Lemma 1. □

Lemma 7. *Let* $i \in [0, n), k \in \{0, 1, 2\}$*. In any feasible schedule, if color choice task* $C(i)$ *is scheduled at time:*

- $r(C(i))$ *then task* $P_{i,1}(1)$ *must be scheduled at its release date,*
- $r(C(i)) + 1$ *then task* $P_{i,0}(2)$ *must be scheduled at its release date,*
- $r(C(i)) + 2$ *then task* $P_{i,2}(0)$ *must be scheduled at its release date.*

Proof. This is similar to the proof of Lemma 2. Let s be the starting time of $C(i)$.

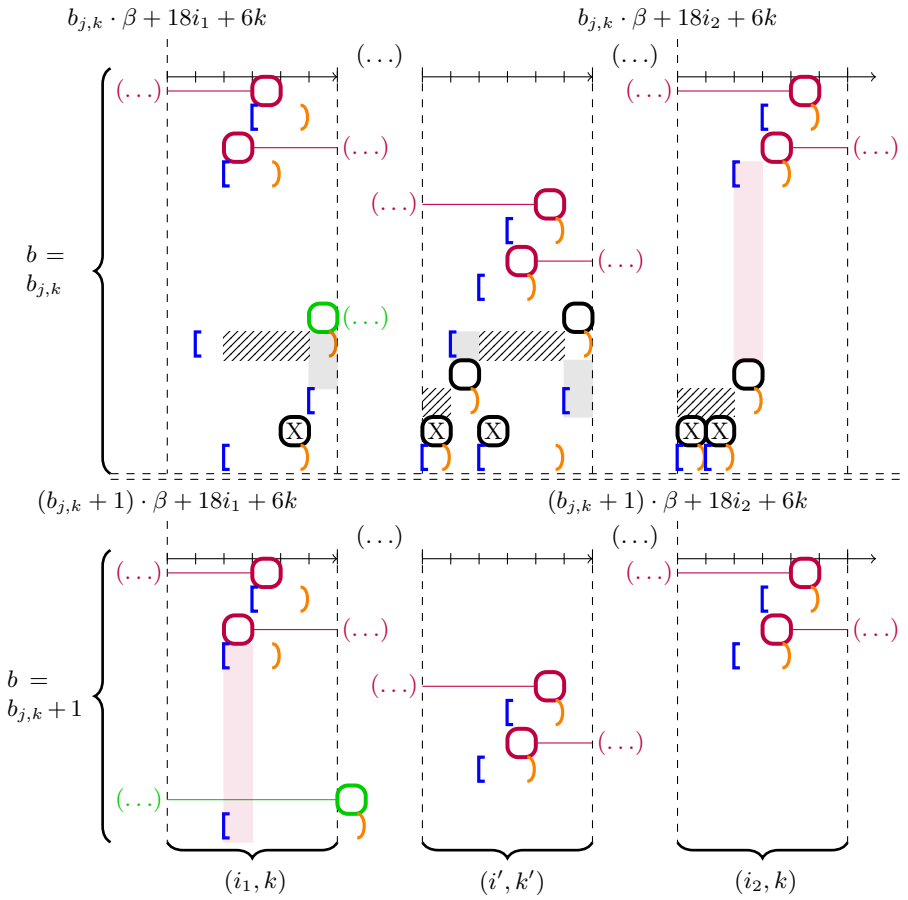

Fig. 4. The part of the edge check segment corresponding to (edge, checked color) couple (e_j, k) in the reduction with minimum delays. The middle part describes the tasks in all subsections (i', k') in between subsections (i_1, k) and (i_2, k) in the same section of length β. In the case of local connectors and fill tasks only the first task of the coupled task is shown.

Case $s = r(C(i))$ is straightforward. $C(i)$ blocks time $r(P_{i,1}(1))+1$ thus task $P_{i,1}(1)$ must be scheduled at its release date.

If $s = r(C(i))+1$ then redirection task $R_{i,1}(1)$ must be scheduled earlier than its deadline minus one. Then with the maximum delay of length β starting from this task, the other end $R'_{i,1}(1)$ must also be scheduled earlier than its deadline minus one. By Lemma 6 this only leaves time $r(R'_{i,1}(1))$. This corresponds to the deadline minus one of local connector $L_{i,0}(2, bridge)$, which again by Lemma 6 only leaves time $r(L_{i,0}(2, bridge))$ to schedule this task. This corresponds to $r(P_{i,0}(2))+1$, which forces propagator $P_{i,0}(2)$ to be scheduled at its release date.

Finally case $s = r(C(i))+2$ is proved the same way through tasks $L_{i,1}(1, bridge)$, $R'_{i,2}(0)$, $R_{i,2}(0)$, $L_{i,2}(0, in)$ and two calls from Lemma 1. □

Lemma 8. *Let $i \in [0, n), k \in \{0, 1, 2\}$. In any feasible schedule, if $P_{i,k}(k)$ is scheduled at its release date, then for all $b \in [k, 3 + 6m)$ tasks $P_{i,k}(b)$ and $P'_{i,k}(b)$ must be scheduled at their release date.*

Proof. This is similar to the proof of Lemma 3. We prove this lemma by induction on b. Let $b \in [2 - k, 3 + 6m - 1)$. Suppose that $P_{i,k}(b)$ is scheduled at its release date. Then by the maximum delay of length β the other end $P'_{i,k}(b)$ must also be scheduled at its release date. This corresponds to the release date plus one of the next propagator $P_{i,k}(b + 1)$, which in turn must also be scheduled at its release date. □

Lemma 9. *Let $e_j = \{i_1, i_2\}$ be an edge in E and $k \in \{0, 1, 2\}$ be a color. In any feasible schedule, either $P_{i_1,k}(k)$ or $P_{i_2,k}(k)$ must be scheduled at its release date plus one.*

Proof. (sketch) This is similar to the proof of Lemma 4. Consider a feasible schedule. By contradiction suppose that both tasks $P_{i_1,k}(k)$ and $P_{i_2,k}(k)$ are scheduled at their release date. By Lemma 8 both tasks $P_{i_1,k}(b_{j,k})$ and $P_{i_2,k}(b_{j,k} + 1)$ must also be scheduled at their release date. We show that when task $P_{i_1,k}(b_{j,k})$ is scheduled at its release date then time $r(P_{i_2,k}(b_{j,k} + 1))$ is blocked, and vice versa.

This is proved by induction on subsection (i, k) between (i_1, k) and (i_2, k). See Figure 4 to visualize the local connecting process. The induction step uses Lemma 6 the same way as in the proof of Lemma 7: the local connectors successively block one scheduling option of the adjacent local connector, which forces it into its second scheduling option. □

Proposition 3. *G is 3-colorable if and only if there exists a feasible schedule for I_{max}.*

Proof. (sketch) This is proved in the same fashion as Proposition 1. □

Appendix C Omitted details in Section 4

Let us now analyze the complexity of the construction of the graph, by bounding the number of nodes and arcs and the construction of successors of a state. We first assume that a preprocessing is done in $\mathcal{O}(n \cdot log(n))$ that sorts the release times and deadlines by nondecreasing order and computes for each date the list of tasks that meet each release time of deadline. We also assume that for each deadline t, the set of at most $\mu + 1$ tasks that crosses $[t - 1, t)$ has been pre-processed in $\mathcal{O}(\mu \cdot n)$.

C.1 Number of tasks crossing a time interval

We first prove the following lemma which bounds the number of tasks that are relevant to a time interval in a feasible instance.

Lemma 10. *Consider a feasible instance of* $1|prec(\ell_{i,j}), r(j), \bar{d}(j)|\star$. *Given a time interval of length* T, *the number of tasks whose availability window intersect with this interval is bounded by* $2\mu + T$.

Proof. Let $[t, t+T)$ be this interval. If $T = 1$ then the result holds by the definition of pathwidth μ. Now suppose $T \geq 2$. Given a task j whose availability window $[r(j), \bar{d}(j))$ intersects $[t, t+T)$, at least one time unit in $[t, t+T)$ must be included in $[r(j), \bar{d}(j))$.

- If time t is included in $[r(j), \bar{d}(j))$:
  by the definition of pathwidth μ the number of such tasks is bounded by $\mu + 1$.
- If time $t + T - 1$ is included in $[r(j), \bar{d}(j))$:
  by the definition of pathwidth μ the number of such tasks is bounded by $\mu + 1$.
- If a time $t' \in (t, t+T-1)$ is included in $[r(j), \bar{d}(j))$:
  if time t or time $t + T - 1$ is also included in $[r(j), \bar{d}(j))$ then the task was already counted in one of the previous cases. If not then the whole availability window of task j must be included in time interval $[t+1, t+T-2]$. By the pigeonhole principle the number of such tasks is bounded by the length of the interval, which is $T - 2$.

In total the number of such tasks is bounded by $(\mu+1)+(\mu+1)+(T-2) = 2\mu + T$. □

C.2 Successor generation in details

We show that the outdegree of any node in G_I is bounded and establish the complexity of generating successors of a state.

Lemma 11. *If* u *is a valid state of stage* $\alpha - 1$.

- *There are at most* $\mu + 1$ *jobs that can be chosen as* $j(v)$ *for a valid successor* v *of* u. *The set of these jobs can be computed in* $\mathcal{O}(\mu \cdot log(\mu))$.
- *There are at most* $\sigma + 1$ *possible values of* $c(v)$ *once* $j(v)$ *is chosen.*
- *Checking whether the choice* $(j(v), c(v))$ *is consistent with time windows and precedence constraints induced by* u *and that* v *is a valid state can be done in* $\mathcal{O}((\ell_{max} + \sigma)\log(\sigma))$.
- *Comparing* v *to already created states can be done in* $\mathcal{O}(log(|\mathcal{V}_\alpha|))$

So, the generation of all valid successors of a valid state u *of stage* $\alpha - 1$ *is in* $\mathcal{O}\left(log(|\mathcal{V}_\alpha|) \cdot \sigma^2(\ell_{max} + \sigma) \cdot \log(\sigma)\right)$.

Proof. Let us consider the least deadline $\bar{d}(\hat{j})$ of an unscheduled job $\hat{j}$. To compute $\bar{d}(\hat{j})$ it is sufficient to search within the sorted deadlines of jobs starting from the one next to $c(u)$ (which can be stored with u). We can search sequentially the next deadlines from $c(u)$ and for each such task k decide whether it is unscheduled (checking if $r(k) \geq c(u)$ (it is unscheduled) or if it is in $A(u)$). Checking if a job is in $A(u)$ can be done in $\mathcal{O}(log(\mu))$ with appropriate data structure. At most $\mu + 1$ jobs will be checked before finding an unscheduled job if any (since all scheduled jobs have a time window crossing $c(u)$).

Once $\hat{j}$ is found, If another candidate i was chosen instead to extend u, then it would be completed before $\hat{j}$ in a valid schedule. Thus $r(i) < \bar{d}(\hat{j})$. Plus by the definition of $\hat{j}$ we also have $\bar{d}(\hat{j}) \leq \bar{d}(i)$. This means that time $\bar{d}(\hat{j}) - 1$ is included in the availability window of all candidates i. By the definition of pathwidth μ this bounds the number of candidates to extend u by $\mu + 1$. Our pre-processing then gives the set of candidates.

Assume we have chosen a $j(v)$ among these candidates There are then at most $\sigma + 1$ possible completion times for $j(v)$. Let us choose $c(v)$ among these possibilities. Notice that we can scan the possible interval by storing at each step the next and previous deadline of $c(v)$. We now have to check several more points.

1. $j(v)$ is unscheduled. As mentioned previously this costs $\mathcal{O}(\log(\mu))$.
2. $\bar{d}(j(v)) - p(j(v)) \geq \max(r(j(v)), c(v))$.
3. All jobs i with $\bar{d}(i) \leq c(v)$ have been scheduled: we just have to check the deadlines from $c(v)$ by decreasing order to $c(u)$, and check whether all such jobs i are in $A(u)$. At most $\mu+1$ such jobs have been already scheduled since then $c(u)$ is in their time window, so the check will take at most $\mathcal{O}(\mu log(\mu))$.
4. We must check that all predecessors of $j(v)$ have been scheduled. If a predecessor k of $j(v)$ was unscheduled then according to the previous check it would satisfy $\bar{d}(k) > c(v)$. We can assume that deadlines and release times are consistent with the precedence constraints with delays. So we would have $r(k) \leq r(j(v)) < c(v) < \bar{d}(k) \leq \bar{d}(j(v))$. Scanning the deadlines from $c(v)$ to $\bar{d}(j(v))$ we can check whether the corresponding jobs are predecessors (with minimal and exact delays) of $j(v)$ and belong to $A(v)$. according to Lemma 10 there are at most $2(\mu + 1) + \sigma$ such tasks. The time complexity of this check is $\mathcal{O}((2(\mu + 1) + \sigma)log(\mu))$.
5. We must satisfy precedence constraints on $j(v)$. if k is a predecessor with maximal or exact delay then it should be scheduled in $S(u)$, otherwise the constraints cannot be satisfied. So we can check for each job in $S(u)$ if it is a predecessor with maximal or exact delay of $j(v)$ and compare to the total number of those predecessors. If k is a predecessor with minimum delay of $j(v)$ which is not scheduled in $S(u)$, it completes before $c(u) - \ell_{max}$. Then k does not impact the starting time of $j(v)$. So we can consider the list of jobs k in $S(u)$, check (in $\mathcal{O}(1)$) whether k is a predecessor of j and compute $s_k = C_k + p(k) + l_{k,j(v)}$. If the delay is exact we should check whether $c(v) = s_k$, if the delay is minimum we should have $c(v) \geq s_k$ and if the delay is maximum $c(v) \leq s_k$. All this can be checked in $\mathcal{O}(\ell_{max} + 1)$.

6. Finally, checking whether a state v has already been created suppose that all states in $\mathcal{V}_\alpha$ are stored in an appropriate data structure, using an encoding of the state. It should be done in $\mathcal{O}(log(|\mathcal{V}_\alpha|))$.

Then bounding μ with 2σ using Theorem 1 achieves the proof. □

C.3 Proof of Theorem 5

Proof. Correctness is ensured by the definition of valid states and by Lemma 11. Now considering the time complexity, from Proposition 2 we know that $|\mathcal{V}_\alpha| \leq |N_\alpha| f(\sigma, \ell_{max})$. Now the algorithm generates successors of all sates of a stage. We can then, according to Lemma 11, state that the complexity of the dynamic programming is in

$$\mathcal{O}\left(\sigma^2 \cdot (\ell_{max} + \sigma) \cdot log(\sigma) \cdot \sum_{\alpha=1}^{n} |\mathcal{V}_{\alpha-1}| log(|\mathcal{V}_\alpha|)\right).$$

But $log(|\mathcal{V}_\alpha|) \leq \log(|N_\alpha|) + log(f(\sigma, \ell_{max}))$. We deduce that

$$\sum_{\alpha=1}^{n} |\mathcal{V}_{\alpha-1}| log(|\mathcal{V}_\alpha|) \leq f(\sigma, \ell_{max}) \left(\sum_{\alpha=1}^{n} |N_{\alpha-1}| log(|N_\alpha|) + log(f(\sigma, \ell_{max})) \sum_{\alpha=1}^{n} |N_{\alpha-1}|\right)$$

Now, $\sum_{\alpha=1}^{n} |N_{\alpha-1}| log(|N_\alpha|) \leq (\sum_{\alpha=1}^{n} |N_\alpha|)^2$ And we know from Proposition 2 that this is bounded by $(5\sigma+1)^2 \cdot n^2$. So we get the overall complexity by setting $h(\sigma, \ell_{max}) =$

$$\left(\sigma^2(\ell_{max} + \sigma) log(\sigma)\right) f(\sigma, \ell_{max})(((5\sigma + 1)^2 + (5\sigma + 1) log(f(\sigma, \ell_{max}))).$$

Now $log(f(\sigma, \ell_{max}))$ is in $\mathcal{O}(\sigma \cdot \ell_{max})$, so that

$$h(\sigma, \ell_{max}) \leq \Delta \cdot \sigma^6(\sigma + \ell_{max}) \cdot (2(\ell_{max} + 1))^{\ell_{max}+6\sigma}$$

where Δ is a constant. Notice that the bounding done here is quite raw, since the aim is not to have the most accurate complexity but only to prove that the algorithm is FPT. □