Coupled-task Scheduling with Time Windows, Bounded Pathwidth and Bounded Slack is para-NP-complete

Maher Mallem[a,1,2,*], Claire Hanen[b,c,3], Alix Munier Kordon[b,4]

[a] *Inria, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668 Lyon cedex 07, 69342, France*
[b] *Sorbonne Université, CNRS, LIP6 F-75005, Paris, France*
[c] *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 two parameters: the pathwidth of the interval graph induced by time windows and the maximum slack of a job within its time window. Three types of precedence delays are studied: minimum, maximum and exact. For all three types of precedence delays we prove that these problems are para-NP-complete parameterized by the slack and the pathwidth, even for unit-time coupled tasks with equal delays.

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

*Corresponding author.
Email addresses: Maher.Mallem@ens-lyon.fr (Maher Mallem), Claire.Hanen@lip6.fr (Claire Hanen), Alix.Munier@lip6.fr (Alix Munier Kordon)
[1]Part of this research project was conducted when this co-author was a PhD student at LIP6, Sorbonne Université.
[2]orcID: 0000-0001-5654-1090
[3]orcID: 0000-0003-2482-5042
[4]orcID: 0000-0002-2170-6366

we just consider decision problems only few problems can be solved in polynomial time - like scheduling unit execution time tasks with time windows [1] or scheduling tasks with precedence constraints and deadlines [2]. On the other hand many sequencing problems with simple settings are NP-hard in the strong sense, especially when time windows are considered [3].

This paper addresses the feasibility check of single machine scheduling with time windows and precedence constraints with delays. We are given a set of tasks $\mathcal{T}$. Each task $j \in \mathcal{T}$ is characterized by a processing time p_j and a time window $[r_j, d_j)$ in which it should be scheduled. A nonnegative delay ℓ_a is attached to each precedence constraint a between two tasks 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 kinds of precedence constraints, sometimes called "generalized precedence constraints" in the literature [4] have many applications - e.g. representing product shelf life in the case of maximum delays or instruction scheduling on pipelined processors in the case of minimum delays [5]. Notice that the complexity of problems with maximum delays is usually less studied than with minimum and exact delays.

In particular we focus on the coupled-task setting. A coupled task is a triplet (a_j, ℓ_j, b_j) where a_j and b_j are two tasks linked by a precedence relation with delay value ℓ_j from the former to the latter, and with no other task linked to a_j or b_j in the precedence graph. In this setting and with no time windows, Yu et al. showed that scheduling unit-time tasks on a single machine is strongly NP-complete for exact and minimum delays [6]. See the recent works by Chen and Zhang [7] and Khatami et al.[8] for a survey of complexity results for coupled tasks with exact delays. Note that coupled-task scheduling on a single machine is essentially equivalent to the two-machine flow shop setting. With minimum delays, Yu notably showed strong NP-completeness with two different values among the delays given in the input, and where both tasks of each coupled task have equal processing time [9]. Since this problem becomes polynomial-time solvable in the special cases of either unit processing times or equal-length delays [9, 10], this highlights the role of both the range of task processing times and the number of delay lengths in the difficulty of the problem.

With this in mind, we propose not only to investigate NP-completeness but also carry out a parameterized complexity analysis of the coupled-task

setting. Several parameters have already been considered for more general scheduling problems, notably the maximum processing time $p_{\max}$ [11] or the width of the precedence graph [12]. For instance van Bevern et al. proved that the problem of scheduling a precedence graph on two machines with task processing times in $\{1, 2\}$ is W[2]-hard with respect to the width [12]. In the same paper, they introduced an additional parameter λ that measures the maximum allowed lag of a task with respect to its earliest start time (ignoring resource constraints). They proved that the Resource-Constrained Project Scheduling Problem (RCPSP) is fixed-parameter tractable parameterized by both the allowed lag and the width. Considering precedence delays the work of Bessy and Giroudeau [13] studies coupled tasks with due dates, a compatibility graph and a common deadline on a single machine. Their consider the number of tasks ending before their due date as parameter. They established W[1]-hardness for this problem and developed a fixed-parameter algorithm when the maximum duration of a task (i.e. the processing time of the two tasks plus their delay) is bounded.

In the presence of time windows, little is known about the parameterized complexity of coupled-task scheduling. Since coupled tasks can be interpreted as chains of precedence relations of length one, one can take inspiration from the existing results on chain-like precedence constraints. In this setting, Bodlaender and van der Wegen addressed single-machine scheduling where the time windows are expressed on the chains instead of the tasks individually [14]. Then the width of the precedence graph corresponds to the number of chains. On top this parameter, they considered the *thickness*, which corresponds to the maximum number of overlapping chain time windows. They proved that the problem with exact delays given in unary is W[1]-complete with respect to the number of chains, W[t]-hard for all $t \geq 1$ with respect to the thickness, and in XP with respect to either parameter. With minimum delays given in unary, the problem is W[1]-hard and in XP with respect to either parameter. Finally, with minimum delays given in binary, they showed XP membership with respect to the number of chains.

When time windows are tied to the tasks individually instead of the chains, two parameters have been introduced in the literature, in line with thickness and allowed lag. The pathwidth μ is the maximum number of overlapping time windows minus one. The slack σ is the maximum slack $(d_j - r_j - p_j)$ of a task j within its time window - i.e. the maximum number of starting times minus one. On parallel identical machines, Munier developed a fixed-parameter algorithm for the decision problem of scheduling unit exe-

cution time tasks with precedence constraints and time windows [15]. With tasks of arbitrary execution time, fixed-parameter algorithms with both couples of parameters "slack and pathwidth" and "maximum processing time and pathwidth" were proposed [16]. On a single processor, scheduling chains of unit execution time tasks with minimum or exact delays was proved to be fixed-parameter tractable if no precedence delays are assumed [17], whereas A Baart et al. [18] proposed a fixed-parameter algorithm with parameters slack and pathwidth for a problem with sequence-dependent setup times and task rejection criteria.

When precedence delays are considered on top of task time windows, unlike in [14], no XP algorithm is currently known with respect to parameter couple "slack and pathwidth" on single machine scheduling with chains of unit-time tasks. In fact, with minimum or exact delays, this problem was recently proved to be para-NP-complete with respect to pathwidth μ, essentially ruling out any XP algorithm for this problem - unless P = NP [19]. The proposed hardness reduction had slack two, which also showed para-NP-completeness.

Now, it must be noted that this reduction heavily relied on the fact that both the chain lengths and the number of delay values were unbounded. In this paper, we show that these properties are not pivotal in yielding the hardness of single machine scheduling with task time windows and precedence delays. We do so by proving para-NP-completeness with respect to parameter couple "slack and pathwidth" even in the case of unit-time coupled tasks with equal-length delays.

The paper is structured as follows. In Section 2 we present our notations, parameters pathwidth μ and slack σ, and the relation between them in the single machine setting. In Section 3 we show that with minimum or maximum delays, considering coupled tasks is equivalent to allowing chains of length lower than or equal to one. Then in Section 4 we describe the gadgets used in all the reductions and we outline the main ideas. In Section 5 (resp. Section 6, Section 7) we use these gadget ideas to show that scheduling coupled tasks with equal precedence delays and unit execution time on a single machine with minimum (resp. maximum, exact) delays is para-NP-hard with respect to the slack, and thus the pathwidth. Finally in Section 8 we draw some perspectives.

2. Problem setting and parameters

In this section we first give some definitions and notations for scheduling with precedence delays and time windows. Then, we define the parameters we consider and establish a relation between two of them, which gives a first insight on the parameterized complexity of our problem. Then we define coupled tasks and state the main result of the paper, that will be proved in the next sections.

2.1. Definitions

In this paper we study the problem denoted by $1|prec(\ell_{i,j}), r_j, d_j|\star$ in the standard notation of Graham [20]. An instance of this problem is characterized by a set $\mathcal{T}$ of n tasks. Each task j in $\mathcal{T}$ has a processing time p_j, a release date r_j and a deadline d_j. These parameters may be also denoted by $p(J), r(J), d(J)$ when the task J has a long name. Moreover we consider an acyclic precedence graph $prec$ representing precedence relations between the tasks in $\mathcal{T}$. Each arc $a = (i, j)$ in $prec$ is valued by a nonnegative integer $\ell_{i,j}$. A schedule s defines for each task j a starting time $s(j)$ such that $r_j \leq s(j) \leq d_j - p_j$ and for any arc $a = (i, j)$ in $prec$:

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 min, max or ex is indicated in the problem definition. For example, $1|prec(\ell_{i,j}^{\max}), r_j, d_j|\star$ describes the problem with only maximum delays.

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

Definition 1. *The* **pathwidth** μ *and the* **slack** σ *are defined as follows:*

- $\mu = \max_{t \geq 0} |\{j \in \mathcal{T}, r_j \leq t < d_j\}| - 1$,
- $\sigma = \max_{j \in \mathcal{T}} d_j - r_j - p_j$.

2.2. Slack vs pathwidth

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

Theorem 1. *[21] If a single machine scheduling instance with time windows is feasible, then* $\mu \leq 2\sigma$.

Proof. Let j be a task. By the definition of slack σ: $r_j \geq d_j - p_j - \sigma$ and $d_j \leq r_j + p_j + \sigma$. Let 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 task 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 $2\sigma + 1$ tasks 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 implication to parameter σ

In previous studies [17] we defined fixed-parameter algorithms on single machine problems with parameter pathwidth. It follows from Theorem 1 that these results transfer to slack instead of pathwidth.

Corollary 1. $(1|prec, r_j, d_j|L_{max}$ *is fixed-parameter tractable parameterized by* σ.

We see that the slack is a powerful parameter that gives a fixed-parameter algorithm for precedence relations without delays. It is thus interesting to address the complexity of the problem with precedence delays parameterized by slack σ.

2.4. Coupled tasks

A coupled task is a set of two tasks $\{j, j'\}$ linked by an arc of the precedence graph with delay ℓ. So a coupled task is usually described by a triplet (j, ℓ, j'). In scheduling problems with coupled tasks we assume that each task has its time window and that no precedence constraint exists between tasks from different coupled tasks. This represents a minimal setting for precedence relations with delays.

Here we consider that all tasks have a unit execution time and all the precedence delays are equal to a single value ℓ. The problem is then denoted by $1|(1,\ell,1),r_j,d_j|\star$. We define $\mathcal{P}_{min}$ (resp. $\mathcal{P}_{max},\mathcal{P}_{ex}$), to be the problem $1|(1,\ell,1),r_j,d_j|\star$ with only minimum (resp. maximum, exact) delays. In this paper we prove that, even with this very simple setting, parameter σ is unlikely to give a FPT algorithm for these three problems.

Theorem 2. *Problems $\mathcal{P}_{min},\mathcal{P}_{max}$ and $\mathcal{P}_{ex}$ are para-NP-complete parameterized by slack σ.*

The proof of the theorem is developed in the next sections. In Section 3 we show that with minimum or maximum delays, considering coupled tasks is equivalent to allowing chains of length lower than or equal to one. Then in Section 4 we describe the gadgets used in all the reductions and we outline the main ideas. Section 5 gives the detailed reduction and proofs for minimum delays. The reductions for maximum and exact delays are then presented in sections 6 and 7 to complete the proof of Theorem 2.

3. Problem relaxation

In this section we show that adding isolated tasks to coupled task instances does not impact the problem difficulty for minimum and maximum delays.We do so while preserving the slack value.

With minimum delays we set:

$$\mathcal{P}'_{min} = 1|chains(\ell^{min}, len \leq 1), \ell_{i,j} = \ell, p_j = 1, r_j, d_j| \star .$$

Lemma 1. *$\mathcal{P}'_{min}$ can be reduced to $\mathcal{P}_{min}$ while keeping the same slack.*

Proof. Given an instance I of problem $\mathcal{P}'_{min}$, we build an instance I' of problem $\mathcal{P}_{min}$ by coupling a new task to every isolated task in I. Let D be the maximum deadline in instance I. We define $L = max(\ell+1, D)$. We put our new tasks in time segment $[L, 2L)$. For each isolated task j in I, we set a coupled task (j,ℓ,j') in I' where $r_{j'} = r_j + L$ and $d_{j'} = d_j + L$. Then I' has indeed the same slack as I.

We show that I is feasible if and only if I' is feasible. I' is the same as I but with extra tasks, so the indirect implication is straightforward. Now suppose I has a feasible schedule S. We propose schedule S' which mimics S on the tasks that were in I, and for all isolated tasks j extra task j' is

scheduled at time $S(j) + L$. We show that S' is valid. Time segment $[0, L)$ is a replica of valid schedule S, so no problem there. Next in time segment $[L, 2L)$ we only have the extra tasks from the isolated tasks in I. As a result it is a copy of time segment $[0, L)$ but with potentially less tasks. So a problem could only come from one of the added delays. But the offset L between a former isolated task and its extra task is larger than delay ℓ, so there is no issue in time segment $[L, 2L)$ either. Thus S' is a valid schedule of I'. □

With maximum delays we will also add extra tasks to be coupled with the isolated tasks. Except we need an extra assumption on the task time windows to complete the reduction. We define $\mathcal{P}'_{max}$ as problem:

$$1|chains(\ell^{max}, len \leq 1), \ell_{i,j} = \ell, p_j = 1, r_j, d_j|\star$$

restricted to instances where for every task j there is an integer k such that time window $[r_j, d_j)$ is included in interval $\Theta_k = [k\ell, (k+1)\ell)$. We then say that task j is *in* Θ_k.

Lemma 2. *$\mathcal{P}'_{max}$ can be reduced to $\mathcal{P}_{max}$ while keeping the same slack.*

Proof. Given an instance I of problem $\mathcal{P}'_{max}$, let D be the maximum deadline in instance I. We define $L = max(\ell + 1, D)$. We build an instance I' of problem $\mathcal{P}_{max}$ with time span $L' = 2L$ and where all coupled tasks will have the same delay $\ell' = 2\ell$. For every integer k in $[0, \frac{L}{\ell})$ we associate interval $\Theta_k = [k\ell, (k+1)\ell)$ in I with interval $\Theta'_k = [k\ell', (k+1)\ell')$ as follows.

Let j be a task of I in Θ_k. So, $r_j = k\ell + \rho_j, d_j = k\ell + \delta_j$. Instance I' also contains task j, with $r'_j = k\ell' + \rho_j, d'_j = k\ell' + \delta_j$, so that its time window is included in $[k\ell', k\ell' + \ell)$. Moreover, if j is an isolated task, we define a new task j' to make (j, ℓ', j') a coupled task in I'. The time window of j' is the one of j with an offset ℓ: $r'_{j'} = r'_j + \ell, d'_{j'} = d'_j + \ell$. It is thus included in the right part $[k\ell' + l, (k+1)\ell')$ of interval Θ'_k.

We show that I is feasible if and only if I' is feasible. Suppose I has a feasible schedule S. We propose schedule S' which mimics S on the tasks that were in I. If $S(j) = k\ell + t, 0 \leq t < \ell$ then we set $S'(j) = k\ell' + t$. And for any isolated task j of I, its associated new task j' is scheduled at time $S'(j) + \ell$.

We show that S' is valid. Consider the machine constraint: as in S' the interval $[k\ell', k\ell' + \ell)$ is a copy of interval $[k\ell, (k+1)\ell)$ in S the machine constraint is still satisfied. The new tasks of Θ'_k can only interfere with other

new tasks. Their schedule is a copy of S' for tasks of Θ'_k so no resource conflict can occur either. Consider now the precedence constraints. Let (i, ℓ, j) be a coupled task of I with i in Θ_k. Notice that, due to maximum delay ℓ, j is either in Θ_k or in Θ_{k+1}. So either $S'(j) - S'(i) - 1 = S(j) - S(i) - 1 \leq \ell \leq \ell'$ or $S'(j) - S'(i) - 1 = S(j) - S(i) - 1 + \ell \leq 2\ell = \ell'$. If now j is an isolated task of I in I' its successor j' is scheduled at $S'(j) + \ell \leq S'(j) + 1 + \ell'$, so that the precedence constraint is met.

Now suppose I' has a feasible schedule S'. We propose schedule S which mimics S' on the tasks that were in I. If $S'(j) = k\ell' + t(j), 0 \leq t(j) < \ell'$ with j a task from I then we set $S(j) = k\ell + t(j)$. We show that S' is valid. Schedule S is a copy of schedule S' in interval $[k\ell', k\ell' + \ell)$. So no resource conflict can occur. If (i, l, j) is a coupled task with $i, j \in \Theta_k$ then $S(j) - S(i) - 1 \leq \ell$. If $i \in \Theta_k, j \in \Theta_{k+1}$ then $S'(j) - S'(i) - 1 = (k+1)\ell' + t(j) - k\ell' - t(i) - 1 = \ell' + t(j) - t(i) - 1 \leq \ell'$, so that $S(j) - S(i) - 1 = \ell + t(j) - t(i) - 1 \leq \ell$. Thus S is a valid schedule of I. □

4. Gadgets and reduction baseline

This section first describes the gadgets and the properties which will be used in all the reductions used in the proof of Theorem 2. Then we outline the structure of the reductions developed in the next sections.

4.1. Switches and redirects

Definition 2. *We say that a task is* **left (resp. right)-shifted** *when it is scheduled at its release time (resp. its deadline minus one).*

Definition 3. *A* **switch** *is a task τ such that in any feasible schedule of the instance, τ is either left-shifted or right-shifted. In other words, τ must be scheduled either at time r_τ - it is then said to be OFF - or at time $d_\tau - 1$ - it is then said to be ON.*

Switches are the base mechanism to propagate information across the created scheduling instances. As it is shown in Figure 1 the most straightforward to make a switch τ is to have exactly $d_\tau - r_\tau - 2$ other tasks which must be performed within interval $[r_\tau + 1, d_\tau - 1)$. In our figures, for each task, we put a dashed area in its time window to represent the set of inaccessible start times for it in any feasible schedule.

Now, notice that we can combine several switches and have a similar behavior as one switch.

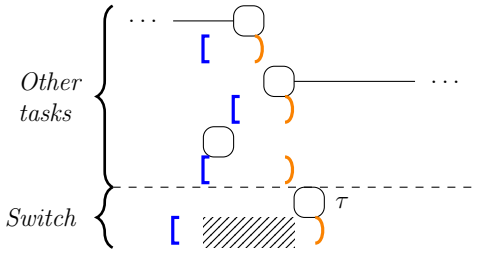


Figure 1: A switch τ in ON position.

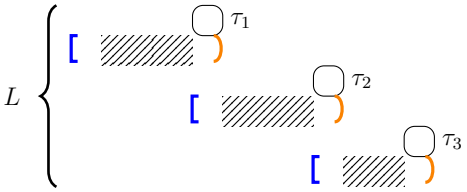


Figure 2: A LongSwitch L with 3 switches, $r(L) = 0$, $d(L) = 12$. Dashed areas indicate task positions which are not available in any feasible schedule.

Definition 4. *A* **LongSwitch** *is a sequence $\tau_1, \ldots, \tau_p$ of switches such that: $\forall j \in \{2, \ldots, p\}, r(\tau_j) = d(\tau_{j-1}) - 1$.*

A LongSwitch example is given in Figure 2. If L is a LongSwitch, we will denote by $r(L)$ the release time of its first switch, and by $d(L)$ the deadline of its last switch. Accordingly to the following lemma, if all switches of a LongSwitch are ON (resp. OFF), we say by extension that the LongSwitch is ON (resp. OFF).

Lemma 3. *In any feasible schedule: (i) if the first switch of a LongSwitch L is ON, then all switches in L must be on ON ; (ii) if the last switch of L is OFF, then all switches in L must be OFF.*

We now define a gadget which will be used to propagate some information in the reverse order as switches.

Definition 5. *A* **left (resp. right) redirect** *is a coupled task (τ, ℓ, τ') such that:*

- *its left half τ (resp. its right half τ') is a switch,*
- $r_{\tau'} - r_\tau = d_{\tau'} - d_\tau = \ell + 1$.

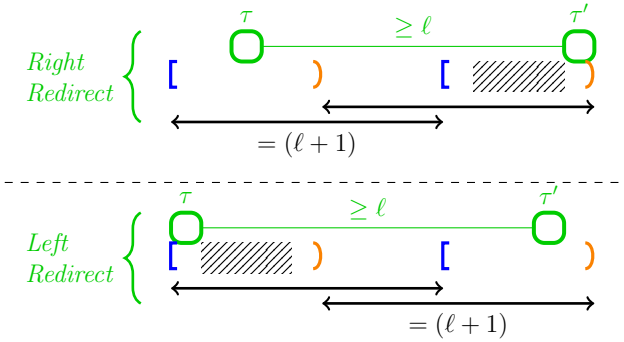


Figure 3: Left and right redirects with minimum delays.

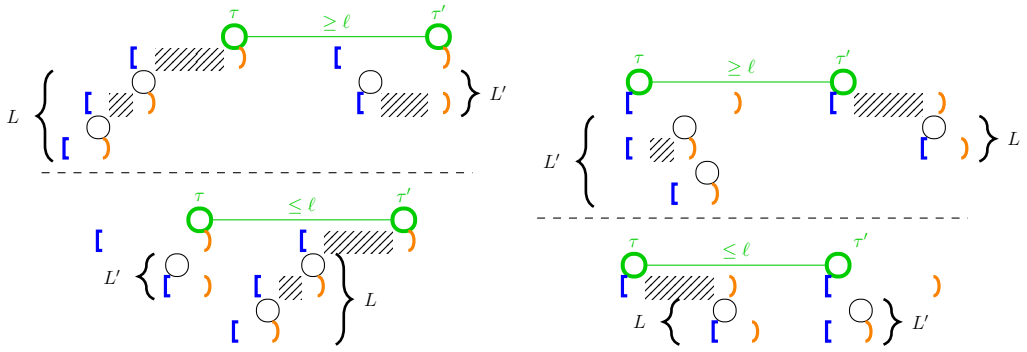
Figure 3 illustrates redirects in the case of minimum delays. We deduce from the definitions of a switch, left and right redirects the two following lemmas:

Lemma 4. *Suppose we have a feasible schedule. If (τ, ℓ, τ') is a redirect with a **minimum** (or exact) delay then:*

- *in the case of a right redirect: if τ is not scheduled at its release date then τ' must be ON. Conversely if τ' is OFF then τ is performed at time $r(\tau)$.*
- *In the case of a left redirect: if τ is ON then τ' must be scheduled at time $d(\tau') - 1$. Conversely if τ' is not scheduled at time $d'(\tau') - 1$ then τ must be OFF.*

Lemma 5. *Suppose we have a feasible schedule. If (τ, ℓ, τ') is a redirect with a **maximum** (or exact) delay then:*

- *in the case of a right redirect: if τ is not scheduled at its deadline minus one then τ' must be OFF. Conversely if τ' is ON then τ is performed at time $d(\tau) - 1$.*
- *In the case of a left redirect: if τ is OFF then τ' must be scheduled at time $r(\tau')$. Conversely if τ' is not scheduled at time $r(\tau')$ then τ must be ON.*



(a) *ON*-to-*OFF* bridges

(b) *OFF*-to-*ON* bridges

Figure 4: Examples of bridges.

4.2. Bridges

Combining switches and redirects allows us to perform some logical operations on the switches status, by the mean of Bridges, an example of which is given in Figure 4:

Definition 6. *A* **bridge** $B = \{L, (\tau, \ell, \tau'), L'\}$ *is a mechanism composed of two LongSwitches* L, L' *and a redirect* (τ, ℓ, τ')*. We distinguish between two types of bridges: ON-to-OFF and OFF-to-ON.*

We have an ON-to-OFF bridge in the following cases:

- (τ, ℓ, τ') *is a left redirect with a minimum (or exact) delay,* $d(L) - 1 = r(\tau)$ *and* $d(L') = d(\tau')$*,*
- (τ, ℓ, τ') *is a right redirect with a maximum (or exact) delay,* $d(L) - 1 = r(\tau')$ *and* $d(L') = d(\tau)$*.*

We have an OFF-to-ON bridge in the following cases:

- (τ, ℓ, τ') *is a right redirect with a minimum (or exact) delay,* $r(L) = d(\tau') - 1$ *and* $r(L') = r(\tau)$*,*
- (τ, ℓ, τ') *is a left redirect with a maximum (or exact) delay,* $r(L) = d(\tau) - 1$ *and* $r(L') = r(\tau')$*.*

Let L, L' be two LongSwitches. The following lemma expresses the logical properties of the bridges.

Lemma 6. *Let $B = \{L, (\tau, \ell, \tau'), L'\}$ be a bridge.*

If B is ON-to-OFF then in any feasible schedule:

$$L \text{ is } ON \implies L' \text{ is } OFF \quad \text{and} \quad L' \text{ is } ON \implies L \text{ is } OFF.$$

Thus LongSwitches L, L' cannot be both ON.

If B is OFF-to-ON then in any feasible schedule:

$$L \text{ is } OFF \implies L' \text{ is } ON \quad \text{and} \quad L' \text{ is } OFF \implies L \text{ is } ON.$$

Thus LongSwitches L, L' cannot be both OFF.

Additionally, in any ON/OFF admissible configuration for L, L', there is a schedule of the tasks in redirect (τ, ℓ, τ') in which both are scheduled either at their release time or at their deadline minus one.

Proof. Let B be an ON-to-OFF bridge (see Figure 4a). Assume that L is ON.

- If we have minimum or exact delays, task τ of the left redirect cannot be scheduled at its release date, so that by Lemma 4 τ' is ON and both tasks τ, τ' are scheduled at their deadline minus one. As then $d(L') = d(\tau')$, L' must be OFF.

- If we have maximum delays, then τ' cannot be scheduled at its release date, so it is ON, and by Lemma 5 τ is scheduled at its deadline minus one,. So both tasks of the redirect are right-shifted, an the last switch of L' cannot be ON, and thus L' is OFF.

Conversely, assume that L' is ON.

- If we have minimum or exact delays, τ' cannot be scheduled at its deadline, so that by Lemma 4, τ is OFF and thus the last switch of L is OFF, so L is OFF. The schedule where both tasks of the redirect are left-shifted is consistent with the configuration.

- If we have maximum delays, then τ cannot be scheduled at its deadline minus one, so that by Lemma 5, τ' is OFF, which implies that the last switch of L is OFF, and thus L is OFF. Similarly, the schedule where both tasks of the redirect are left-shifted is consistent with the configuration.

Now in the configuration where both L and L' are OFF, the redirect can be either left or right-shifted.

Consider now an OFF-to-ON bridge (see Figure 4b). If L is OFF then in the case of minimum or exact delays (resp. maximum) , τ' is OFF (resp. τ is OFF) so that by Lemma 4 and 5, τ (resp. τ') is scheduled at its release time. And this implies that L' is ON. The schedule where τ, τ' are scheduled at their release time is consistent with this configuration.

If L' is OFF then in the case of minimum or exact delays (resp. maximum), τ (resp. τ') cannot be scheduled at its release time, so that by Lemma 4 and 5, τ' (resp. τ) is ON, which implies that L is ON. The schedule where both τ, τ' are scheduled at their deadline minus one is consistent with this configuration.

If both LongSwitches are ON, then any left-shifted of right-shifted configuration of the redirect is consistent.

□

A variant of bridges, which we call half-bridges, can be obtained by removing one of the LongSwitches. The motivation is to make the redirect interact with a task which is not necessarily a switch.

Definition 7. *A* **half-bridge** $B = \{(\tau, \ell, \tau'), L\}$ *is a mechanism composed of a LongSwitch* L *and a redirect* (τ, ℓ, τ')*, Two cases may occur:*

- (τ, ℓ, τ') *is a right redirect with a minimum (or exact) delay,* $r(L) = d(\tau') - 1$,
- (τ, ℓ, τ') *is a left redirect with a maximum (or exact) delay,* $r(L) = d(\tau) - 1$.

We can state the following lemma.

Lemma 7. *If* $B = \{(\tau, \ell, \tau'), L\}$ *is a half-bridge, then in any feasible schedule, in the case of minimum or exact delays (resp. maximum delays) if a task* j *is scheduled at time* $r(\tau)$ *(resp.* $r(\tau')$*) then* L *is* ON.

Proof. If j is scheduled at time $r(\tau)$ (resp. $r(\tau')$) the corresponding task of the redirect cannot be scheduled at the same time, so according to Lemma 4 and 5, its sibling τ' (resp τ) is ON, which implies that the first switch of LongSwitch L cannot be OFF, so L is ON. □

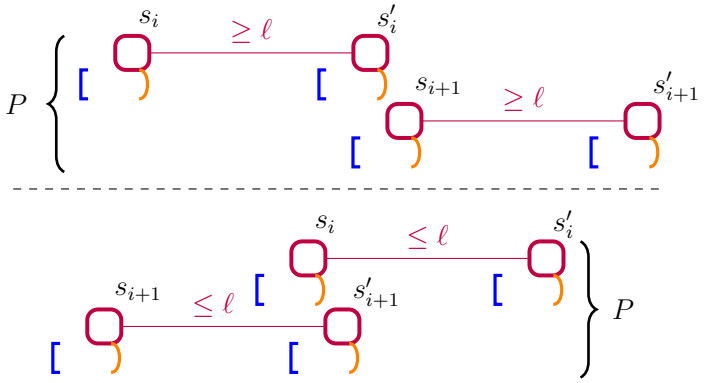


Figure 5: Propagation sequences with minimum and maximum delays.

4.3. Propagation sequences

Definition 8. *A* **propagation sequence** P *is a sequence of coupled tasks* $(s_i, \ell, s'_i)_{1 \leq i \leq k}$ *such that for all* $1 \leq i \leq k$*:*

1. s_i, s'_i *are switches with time window of length* 2*:* $d(s_i) - r(s_i) = d(s'_i) - r(s'_i) = 2$,
2. $r(s'_i) = r(s_i) + \ell + 1$,
3. *if* $i \leq k - 1$*:*
   (a) *for minimum or exact delays:* $r(s_{i+1}) = r(s_i) + \ell + 2$,
   (b) *for maximum delays:* $r(s_{i+1}) = r(s_i) - \ell$.

We define $r(P)$ *the release time of propagation sequence* P *as the release time of its first task* $r(s_1)$ *(resp.* $r(s'_1)$*) in the in the case of minimum or exact delays (resp. maximum delays).*

We say that propagation sequence P *is ON if all its tasks are scheduled at their deadline minus one. Conversely, we say that* P *is* **not** *ON if at least of its tasks is scheduled at its release date.*

Notice that a propagation sequence P is entirely defined by its length k, delay ℓ and release time $r(P)$. Propagation sequences have the following logical property which will be used in all our reductions:

Lemma 8. *Let* $P = (s_i, \ell, s'_i)_{1 \leq i \leq k}$ *be a propagation sequence in some feasible schedule. With minimum or exact (resp. maximum) delays, if* s_1 *(resp.* s'_1*) is scheduled at its deadline minus one, then* P *is ON.*

Proof. Observe that the coupled tasks (s_i, ℓ, s'_i) of a propagation sequence are both left and right redirects. So in case of minimum or exact delays, according to Lemma 4, if s_i is right-shifted, so is s'_i, which implies that s_{i+1} is right-shifted. In case of maximum delays, according to Lemma 5, if s'_i is right-shifted so is s_i, and thus s'_{i+1} is right-shifted. □

So no matter if we have minimum, exact or maximum delays, in any feasible schedule a propagation sequence will propagate whether its first tasks is right-shifted to all other.

Bridge mechanisms can be used to interact with propagation sequences by embedding one of the propagation sequence switches into a LongSwitch of the bridge. Typically we will use a bridge to either make a propagation sequence ON, or to prevent two propagation sequences from being ON at the same time.

4.4. Reduction baseline

For 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 $V = \{0, \ldots, n-1\}$ and $E = (e_j)_{0 \leq j < m} \subseteq V \times V$. Without loss of generality we suppose that there is no self loop in E.

When building our scheduling instance, we segment time into n color choice zones plus m edge check zones. In each color choice zone, the color of some node i is chosen in $\{0, 1, 2\}$. And in each edge check zone, for some edge $e_j = \{i_1, i_2\}, i_1 < i_2$ we verify that nodes i_1 and i_2 did not choose the same color.

We set $\beta = 18n$. Starting from time zero, we consider the $3n$ couples (i, k) of graph node and color choice in lexicographic order, and associate an interval of length 6 to each one of them. By repeating this process over and over, we associate each couple (i, k) to intervals of the form $[x\beta + 18i + 6k, x\beta + 18i + 6(k+1))$ with x some nonnegative integer.

Definition 9. *Given $i \in \{0, \ldots, n-1\}$ and $k \in \{0, 1, 2\}$, we say that an interval is a subsection of* **type** (i, k) *when it is of the form $[x\beta + 18i + 6k, x\beta + 18i + 6(k+1))$ for some nonnegative integer x.*

Then whether node i has color k will be encoded in subsections of type (i, k). And, given an edge $e_j = \{i_1, i_2\}, i_1 < i_2$, we will verify that nodes i_1

and i_2 does not choose the same color by relating a task from a subsection of type (i_1, k) with a task from a subsection of type (i_2, k).

In terms of tasks we will have:

- n color choice tasks $\mathcal{C}_i$ with $i \in \{0, \ldots, n-1\}$: task $\mathcal{C}_i$ represents the color choice of node i. $\mathcal{C}_i$ has three possible starting times and each one corresponds to a color choice.

- $3n$ color propagation sequences $P_{i,k}$ with $i \in \{0, \ldots, n-1\}, k \in \{0, 1, 2\}$: if color k is chosen for node i, then propagation sequence $P_{i,k}$ will be ON and it will propagate this piece of information in the edge check zones.

- In the color choice zones: $2n$ OFF-to-ON bridges (or half-bridges) $B_{i,k}$ with $i \in \{0, \ldots, n-1\}$, $k \in \{0, 2\}$. In the color choice zone associated to node $i \in \{0, \ldots, n-1\}$: bridge $B_{i,0}$ (resp. half-bridge $B_{i,2}$) will guarantee that propagation sequence $P_{i,0}$ (resp. $P_{i,2}$) is ON if color choice task $\mathcal{C}_i$ has the position corresponding to color choice 0 (resp. 2).

- In the edge check zones: $3m$ ON-to-OFF bridges $B_{i,k}$ with $i \in \{0, \ldots, m-1\}$, $k \in \{0, 1, 2\}$. In the edge check zone associated to edge $e_j = \{i_1, i_2\}$, $j \in \{0, \ldots, m-1\}$: for each $k \in \{0, 1, 2\}$ bridge $B_{n+j,k}$ will guarantee that propagation sequences $P_{i_1,k}$ and $P_{i_2,k}$ cannot be ON at the same time.

- Some fill tasks - marked with an "X" in the reduction figures- to set the LongSwitches of the bridges.

5. Reduction for minimum delays

In this section, we build I_{min} a scheduling instance of $\mathcal{P}'_{min}$ where all coupled tasks have the same minimum delay $\ell = \beta - 2$ with $\beta = 18n$. Then, we prove that it is feasible if and only if G is 3-colorable.

5.1. Timeline

Color choice zones are set within time interval $[0, 3\beta)$, while edge check zones are set within time interval $[3\beta, (3+6m)\beta)$. Interval $[0, 3\beta)$ is split into three sections of length β called X_0, X_1, X_2. Each node $i \in \{0, \ldots, n-1\}$ defines a time interval of length 18 in each zone, so that intervals associated with node i have length β :

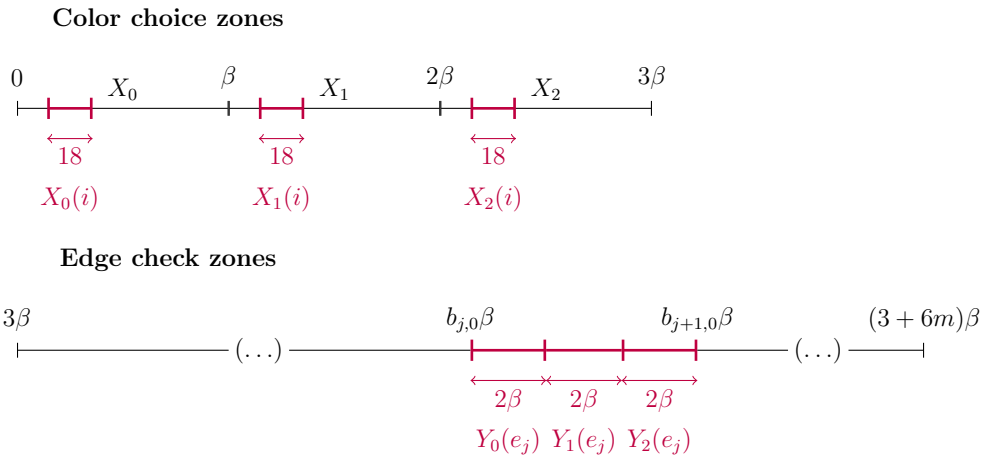


Figure 6: Timeline of the reduction for minimum delays, with $b_{j,k} = 3 + 6j + 2k$, for $k \in \{0, 1, 2\}$. We highlight color choice zone i and edge check zone e_j, which are described in greater detail in Figure 7 and Figure 8 respectively.

$X_0(i) = [18i, 18(i + 1))$, $X_1(i) = [18i + \beta, 18(i + 1) + \beta)$ and $X_2(i) = [18i + 2\beta, 18(i + 1) + 2\beta)$.

In each interval $X_k(i)$ the propagation of the information that the color of node i is k is initiated.

The edge check zone $[3\beta, 3\beta + 6m\beta)$ is split into m intervals of length 6β, one for each edge check zone. So for each edge e_j (indexed from 0 to $m - 1$), the interval $[3\beta + 6\beta j, 3\beta + 6\beta(j + 1))$ is devoted to checking a conflict for edge e_j. To this purpose, this interval is divided into three consecutive zones of length 2β, corresponding to the three colors. Let $b_{j,k} = 3 + 6j + 2k$, with $k \in \{0, 1, 2\}$. So in interval $Y_k(e_j) = [b_{j,k}\beta, b_{j,k+1}\beta)$ of length 2β, the existence of a color conflict for color k on edge e_j is checked. Figure 6 illustrates this timeline.

5.2. Color choice zones

Let us consider time interval $[0, 3\beta)$, the color choice section of the timeline. Recall that for each $k \in \{0, 1, 2\}$, $X_k(i) = [18i + k \cdot \beta, 18(i + 1) + k \cdot \beta)$ (see Figure 6).

In each interval $X_k(i)$ is initiated the propagation of the information that the color of node i is k by the mean of a propagation sequence $P_{i,k}$ with its release time equal $r(P_{i,k}) = k\beta + 18i + 6k + 2$. It is composed of $3 + 6m - k$

coupled tasks, and thus propagates its information until the end of the edge check zone.

Note that propagation sequence $P_{i,k}$ only appears in subsections of type (i, k). So we can deduce the following property:

Lemma 9. *In instance I_{min} two tasks of two different propagation sequences cannot be performed at the same time.*

Proof. All tasks of propagation sequence $P_{i,k}$ appear in intervals of the form $[x\beta + 18i + 6k + 1, x\beta + 18i + 6k + 4)$. Thus we immediately get the result from the value $\ell = \beta - 2$ and from the definition of a propagation sequence. □

In interval $X_1(i)$ the color choice of node i is set via task $\mathcal{C}_i$. It has release time $\beta + 18i + 6$ and a time window of length three. The successive positions of this task correspond to colors $0, 2, 1$. The other tasks in color choice zone i - see Figure 7 - link the color choice tasks to the relevant propagation sequences:

- Bridge $B_{i,0}$ is a OFF-to-ON bridge. Its left LongSwitch is composed of a switch with release time $18i + 1$ and a time window of length two, chained with the first switch of propagation sequence $P_{i,0}$. Its redirect has release time $18i + 1$ and a time window of length five. Its right LongSwitch is composed of a switch with release time $\beta + 18i + 4$ and a time window of length three.

- $B_{i,2}$ is a half-bridge with release time $\beta + 18i + 7$. Its LongSwitch is composed of a switch with release time $2\beta + 18i + 10$ and a time window of length five, chained with the first switch of the propagation sequence $P_{i,2}$. Its redirect has time window of length 5 with release time $\beta + 18i + 7$.

- Six fill tasks ensure that the switches of the bridges are well formed. The fill task with release time $\beta+18i+5$ (resp. $2\beta+18i+7$) and a time window of length 3, along with the two first switches of propagation sequence $P_{i,0}$ (resp.$P_{i,1}$), ensure that the right task of the redirect of Bridge $B_{i,0}$ (resp. $B_{i,2}$) is a switch. The fill task with release time $\beta + 18i + 5$ and a time window of length one ensures that the right LongSwitch of $B_{i,0}$ is a switch. The three last fill tasks with a time window of length 1 and release times $2\beta + 18i + 11$, $2\beta + 18i + 12$ and

$2\beta + 18i + 13$ ensure that the right LongSwitch of half-bridge $B_{i,2}$ is a switch.

Lemma 10. *In any feasible schedule of instance I_{min}, if task $\mathcal{C}_i$ has a position corresponding to color choice k, then propagation sequence $P_{i,k}$ is ON.*

Proof. If color choice task $\mathcal{C}_i$ is scheduled at its deadline minus one (i.e. at time $\beta + 18i + 8$), corresponding to color choice 1, then propagation sequence $P_{i,1}$ cannot start at its release time, so by Lemma 8 it is ON. If task $\mathcal{C}_i$ is scheduled at its release time, then the right LongSwitch of bridge $B_{i,0}$ cannot start at its deadline. So it is OFF which, according to Lemma 6, implies that propagation sequence $P_{i,0}$ cannot start at its release time. So again by Lemma 8 it is ON. Finally, if $\mathcal{C}_i$ is scheduled at time $\beta + 18i + 7$ then it activates half-bridge $B_{i,2}$ which, according to Lemma 7, prevents propagation sequence $P_{i,2}$ from starting at its release time. So by Lemma 8 the latter is ON. □

5.3. Edge check zones

Consider now time interval $[3\beta, 3\beta + 6m\beta)$, the Edge check section of the timeline. For an edge e_j and a color $k \in \{0, 1, 2\}$, interval $Y_k(e_j) = [b_{j,k}\beta, b_{j,k+1}\beta)$ with $b_{j,k} = 3 + 6j + 2k$, will be used to check the existence of a color conflict for color k on edge e_j (see Figure 6).

Now let us zoom on interval $Y_k(e_j)$ (see Figure 8). Let us assume that $e_j = \{i_1, i_2\}$ with $i_1 < i_2$. To ensure that nodes i_1, i_2 cannot both have color k, within zone $Y_k(e_j)$ we use an ON-to-OFF bridge $B_{n+j,k}$ to relate one switch of propagation sequence $P_{i_1,k}$ with one switch of propagation sequence $P_{i_2,k}$: the former will be part of its left LongSwitch while the latter will be its right LongSwitch. We also use $6(i_2 - i_1) + 1$ fill tasks to set the bridge. ON-to-OFF bridge $B_{n+j,k}$ is defined as follows :

- Its redirect has release time $b_{j,k}\beta + 18i_2 + 6k$ and a time window of length five.
- A fill task with release time $b_{j,k}\beta + 18i_2 + 6k + 1$ and a time window of length three which, together with propagation sequence $P_{i_2,k}$, ensure that left task of the redirect is a switch.
- Its right LongSwitch is the switch of propagation sequence $P_{i_2,k}$ with release time $(b_{j,k} + 1)\beta + 18i_2 + 6k + 2$.

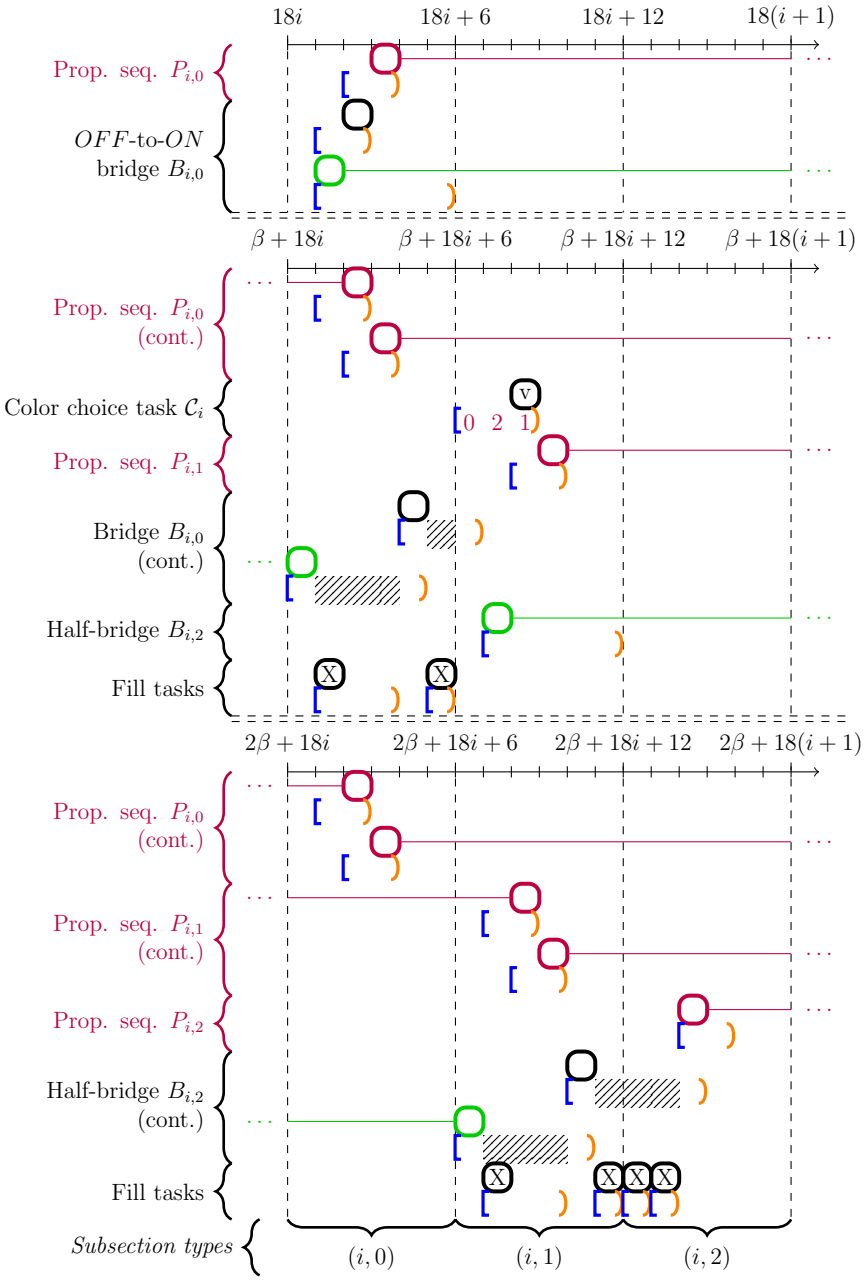


Figure 7: Color choice zone i in instance I_{min}.

- Its left LongSwitch is composed of a sequence of switches:
  - Its first switch is the switch of propagation sequence $P_{i_1,k}$ with release time $b_{j,k}\beta + 18i_1 + 6k + 2$.
  - This switch is chained with another switch with release time $b_{j,k}\beta + 18i_1 + 6k + 3$ and a time window of length four. Two fill tasks with release time $b_{j,k} \cdot \beta + 18i_1 + 6k + 4$, $b_{j,k} \cdot \beta + 18i_1 + 6k + 5$ and a time window of length one ensure that it is a switch.
  - Then for each subsection of type (i', k') such that $b_{j,k}\beta + 18i' + 6k' \in [b_{j,k}\beta + 18i_1 + 6(k+1), b_{j,k}\beta + 18i_2 + 6k)$, we define two switches chained with the previous ones:
    - One has release time $b_{j,k}\beta + 18i' + 6k'$ and a time window of length five. A fill task with release time $b_{j,k} \cdot \beta + 18i' + 6k' + 1$ and a time window of length three, together with propagation sequence $P_{i',k'}$, ensure that the three central positions of the switch are used by other tasks.
    - The other has release time $b_{j,k}\beta + 18i' + 6k' + 4$ and a time window of length three. A fill task with release time $b_{j,k} \cdot \beta + 18i' + 6k' + 5$ and a time window of length one ensures that it is indeed a switch.

Lemma 11. *Let $e_j = \{i_1, i_2\}$ be an edge and $k \in \{0, 1, 2\}$ be a color. Then, in any feasible schedule of instance I_{min}, $P_{i_1,k}$ and $P_{i_2,k}$ cannot be both ON.*

Proof. As bridge $B_{n+j,k}$ is an ON-to-OFF Bridge, by Lemma 6, if $P_{i_1,k}$ is ON then $P_{i_2,k}$ is OFF. □

5.4. Proof of the reduction

Remark 1. *Note that all tasks have a time window of length five of less, so we have slack $\sigma = 4$.*

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

Proof.

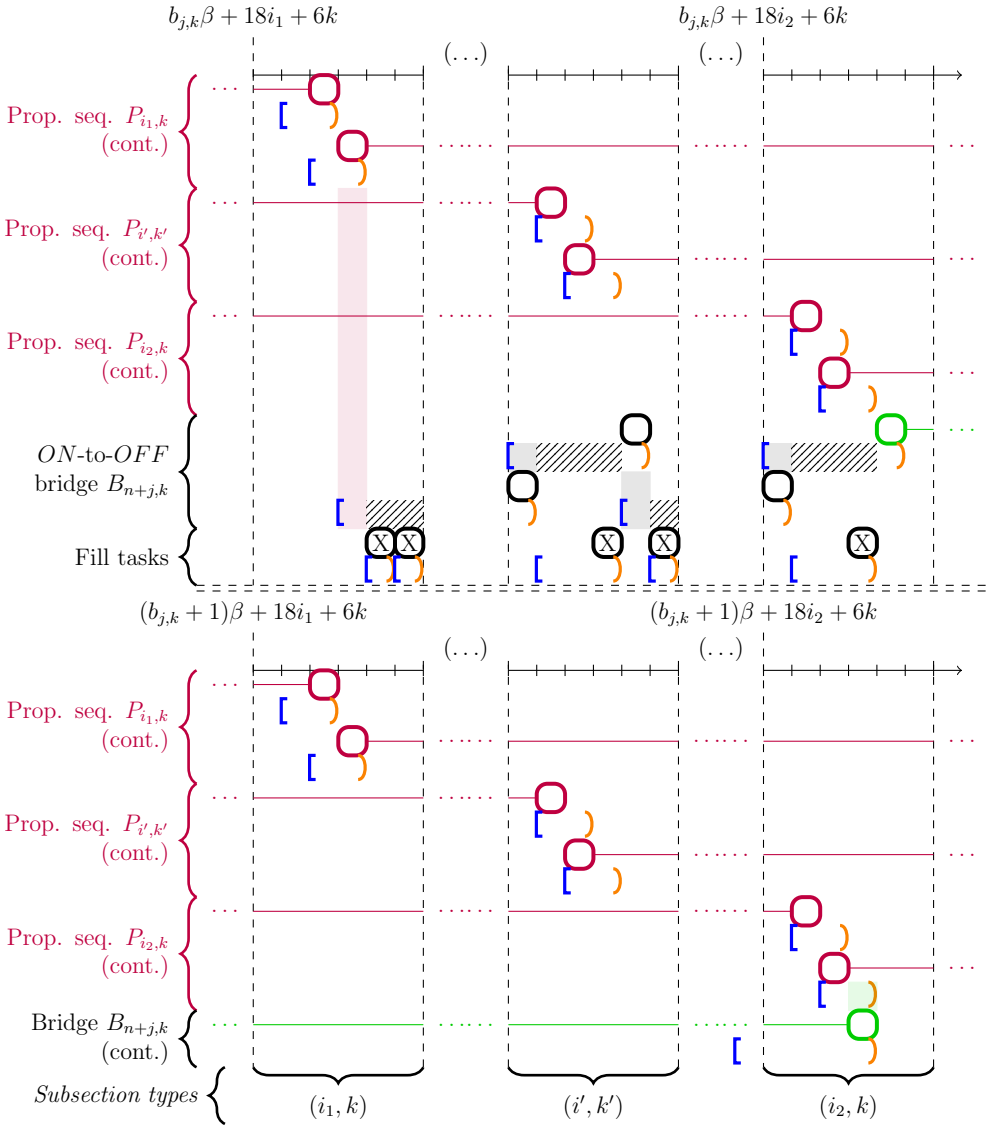


Figure 8: Edge check zone e_j in instance I_{min}. (i', k') represents any subsection type between (i_1, k) and (i_2, k).

($\Longleftarrow$) Suppose there exists a feasible schedule s for instance I_{min}. We propose a 3-coloring $(c(i))_{0\leq i<n}$ based on the starting times of color choice tasks $\mathcal{C}_i$. For i in $[0, n)$, we set the color of node i in G:

$$c(i) = \begin{cases} 0 & if \quad s(\mathcal{C}_i) = r_{\mathcal{C}_i} \\ 2 & if \quad s(\mathcal{C}_i) = r_{\mathcal{C}_i} + 1 \\ 1 & if \quad s(\mathcal{C}_i) = r_{\mathcal{C}_i} + 2 \end{cases}$$

By contradiction suppose that $(c(i))_{0\leq i<n}$ 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 10 both color choice switches $P_{i_1,k}$ and $P_{i_2,k}$ are ON in feasible schedule s, which contradicts Lemma 11.

($\Longrightarrow$) Suppose there exists a valid 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 $\mathcal{C}_i$ is scheduled at time $r(\mathcal{C}_i)$ if $c(i) = 0$, $r(\mathcal{C}_i) + 1$ if $c(i) = 2$, $r(\mathcal{C}_i) + 2$ if $c(i) = 1$,
- For $i \in [0, n)$ and $k \in \{0, 1, 2\}$ propagation sequence $P_{i,k}$ and all LongSwitches which include a task of $P_{i,k}$ are ON if $c(i) = k$ and OFF otherwise. This will guarantee that the LongSwitches of bridges and half-bridges will have an ON/OFF position consistent with Lemma 6.
- Bridges and half-bridges are then set to their consistent configuration according to Lemma 6, with any task either left-shifted or right-shifted. So they cannot induce a conflict with any propagation sequence - obviously for propagation sequences connected to the bridge, but also for tasks of propagation sequences that defines the switches of the bridges - their time window does not include the first and the last position of the bridge.
- Fill tasks with a time window of length three are scheduled at their release time if the overlapping propagation sequence is ON and at their deadline minus one if the overlapping propagation sequence is OFF. This implies that fill tasks cannot be scheduled at the same time as another task.

We deduce that schedule s is feasible for instance I_{min}.

□

From these results and Lemma 1 we deduce the following part of Theorem 2:

Lemma 12. *$\mathcal{P}_{min}$ is para-NP-hard parameterized by slack σ.*

6. Reduction for maximum delays

In this section we use similar mechanisms to build I_{max}, a scheduling instance of $\mathcal{P}'_{max}$, where all coupled tasks have the same maximum delay $\ell = \beta$ with $\beta = 18n$. We define instance I_{max}.

6.1. Timeline

Unlike the minimum delay case, color choice information is propagated from right to left. As such edge check zones are set within time interval $[\beta, (6m+1)\beta)$ while color choice zones are within time interval $[(6m+1)\beta, (6m+4)\beta)$. (Note that time interval $[0, \beta)$ will only host the left task of the last coupled task in every propagation sequence.)

For any vertex $i \in \{0, \ldots, n-1\}$ and color $k \in \{0, 1, 2\}$, we set $X_k(i)$ to be the interval where the propagation that node i has color k is initiated:

$$X_k(i) = [(6m+3-k)\beta + 18i, (6m+3-k)\beta + 18(i+1))$$

The edge check section of the timeline $[\beta, (6m+1)\beta)$ is split into m intervals of length 6β, one for each edge check zone. So for each edge e_j (indexed from 0 to $m-1$), the interval $[(1+6j)\beta j, (1+6(j+1))\beta)$ is devoted to checking a conflict for edge e_j. To this purpose, this interval is divided into three consecutive zones of length 2β, corresponding to the three colors. Let $b_{j,k} = 1 + 6j + 2k$, with $k \in \{0, 1, 2\}$. So in interval $Y_k(e_j) = [b_{j,k}\beta, b_{j,k+1}\beta)$ the existence of a color conflict for color k on edge e_j is checked.

6.2. Color choice zones

In each interval $X_k(i)$ is initiated the propagation of the information that the color of node i is k by the mean of a propagation sequence $P_{i,k}$ with its release time equal to $r(P_{i,k}) = (6m+3-k)\beta + 18i + 6k + 2$. It is composed of $3 + 6m - k$ coupled tasks, and thus propagates its information until the beginning of the edge check zone.

As in the reduction with minimum delays, propagation sequence $P_{i,k}$ only appears in subsections of type (i, k). So we can deduce the following property:

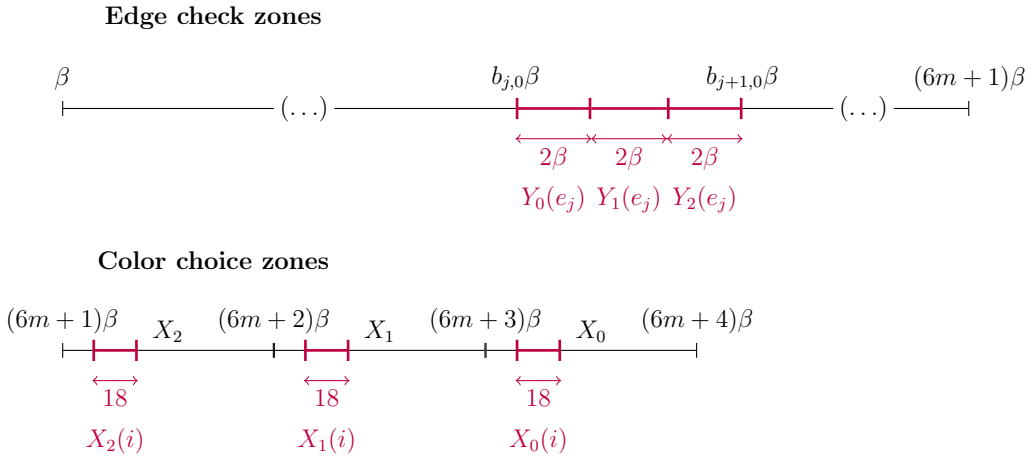


Figure 9: Timeline of the reduction for maximum delays, where $b_{j,k} = 1 + 6j + 2k$, for $k \in \{0, 1, 2\}$. We highlight color choice zone i and edge check zone e_j, which are described in greater detail in Figure 10 and Figure 11 respectively.

Lemma 13. *In instance I_{max} two tasks of two different propagation sequences cannot be performed at the same time.*

Proof. All tasks of propagation sequence $P_{i,k}$ necessarily appear in intervals of the form $[x\beta + 18i + 6k + 2, x\beta + 18i + 6k + 5)$. Thus we immediately get the result from the value $\ell = \beta$ and from the definition of a propagation sequence. □

In interval $X_1(i)$ the color choice of node i is set via task $\mathcal{C}_i$. It has release time $(6m + 2)\beta + 18i + 7$ and a time window of length three. The successive positions of this task correspond to colors $0, 2, 1$. The other tasks in color choice zone i - see Figure 10 - link the color choice tasks to the relevant propagation sequences:

- Bridge $B_{i,0}$ is a OFF-to-ON bridge. Its left LongSwitch is composed of a switch with release time $(6m + 2)\beta + 18i + 5$ and a time window of length three. Its redirect has release time $(6m + 2)\beta + 18i + 1$ and a time window of length five. Its right LongSwitch is composed of a switch with release time $(6m + 3)\beta + 18i + 2$ and a time window of length two, chained with the first switch of propagation sequence $P_{i,0}$.
- $B_{i,2}$ is a half-bridge with release time $(6m+1)\beta+18i+7$. Its LongSwitch is composed of a switch with release time $(6m + 1)\beta + 18i + 11$ and a

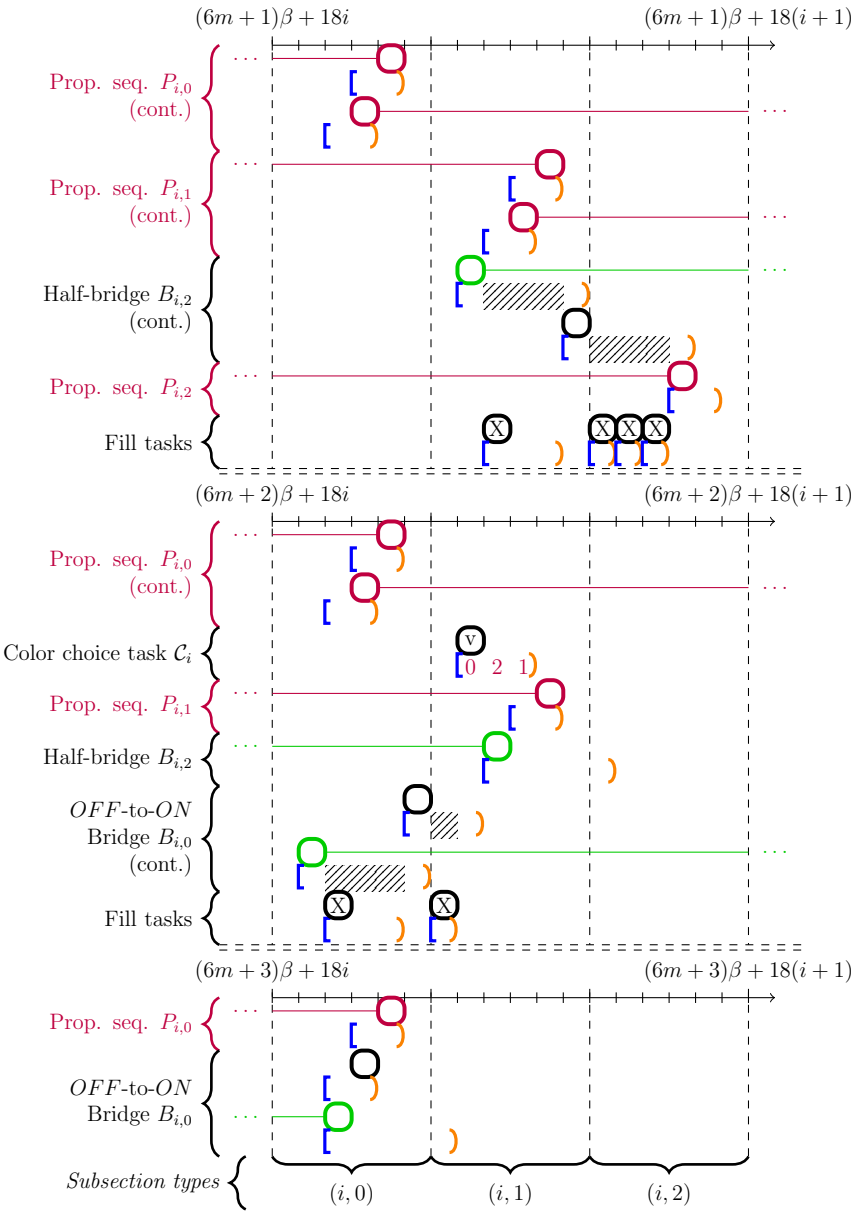


Figure 10: Color choice zone i in instance I_{max}.

time window of length five, chained with the first switch of propagation sequence $P_{i,2}$. Its redirect has release time $(6m+1)\beta + 18i + 7$ and a time window of length five.

- Six fill tasks ensure that the switches of the bridges are well formed. The fill task with release time $(6m+2)\beta + 18i + 2$ (resp. release time $(6m+1)\beta + 18i + 8$) and a time window of length three, along with the second and third switches of propagation sequence $P_{i,0}$ (resp.$P_{i,1}$), ensure that the right task of the redirect of bridge $B_{i,0}$ (resp. half-bridge $B_{i,2}$) is a switch. The three fill tasks with release times $(6m+1)\beta+18i+12$, $(6m+1)\beta+18i+13$ and $(6m+1)\beta+18i+14$, and a time window of length one, ensure that the right LongSwitch of half-bridge $B_{i,2}$ is a switch. The final fill task with release time $(6m+2)\beta+18i+6$ and a time window of length one ensures that the right LongSwitch of bridge $B_{i,0}$ is a switch.

Lemma 14. *In any feasible schedule of instance I_{max}, if color choice task $\mathcal{C}_i$ is scheduled at the position corresponding to color choice k, then propagation sequence $P_{i,k}$ is ON.*

Proof. If color choice task $\mathcal{C}_i$ is scheduled at its deadline minus one (i.e. at time $(6m+2)\beta + 18i + 9$), corresponding to color choice 1, then propagation sequence $P_{i,1}$ cannot start at its release time, so by Lemma 8 it is ON. If task $\mathcal{C}_i$ is scheduled at its release time, then the right LongSwitch of bridge $B_{i,0}$ cannot start at its deadline. So it is OFF which, according to Lemma 6, implies that propagation sequence $P_{i,0}$ cannot start at its release time. So again by Lemma 8 it is ON. Finally, if $\mathcal{C}_i$ is scheduled at time $(6m+2)\beta+18i+8$ then it activates half-bridge $B_{i,2}$ which, according to Lemma 7, prevents propagation sequence $P_{i,2}$ from starting at its release time. So by Lemma 8 the latter is ON. □

6.3. Edge check zones

Now let us zoom on interval $Y_k(e_j)$ of the edge check section, devoted to check whether there is a conflict with color k on edge e_j. Figure 10 illustrates the mechanisms used in this interval. Assuming $e_j = \{i_1, i_2\}$ with $i_1 < i_2$, we use an ON-to-OFF bridge $B_{n+j,k}$ to relate one switch of propagation sequence $P_{i_1,k}$ with one switch of propagation sequence $P_{i_2,k}$: the former will be part of its left LongSwitch while the latter will be its right LongSwitch.

We also use $6(i_2 - i_1) + 1$ fill tasks to set the bridge. ON-to-OFF bridge $B_{n+j,k}$ is defined as follows :

- Its redirect has release time $b_{j,k}\beta + 18i_2 + 6k$ and a time window of length five.
- A fill task with release time $(b_{j,k}+1)\beta+18i_2+6k+2$ and a time window of length three which, together with propagation sequence $P_{i_2,k}$, ensure that right task of the redirect is a switch.
- Its left LongSwitch is the switch of propagation sequence $P_{i_2,k}$ with release time $(b_{j,k} + 1)\beta + 18i_2 + 6k + 2$.
- Its right LongSwitch is composed of a sequence of switches:
  - Its first switch is the switch of propagation sequence $P_{i_1,k}$ with release time $b_{j,k}\beta + 18i_1 + 6k + 3$.
  - This switch is chained with another switch with release time $b_{j,k}\beta + 18i_1 + 6k + 4$ and a time window of length four. Two fill tasks with release time $b_{j,k} \cdot \beta+18i_1+6k+5$, $b_{j,k} \cdot \beta+18i_1+6(k+1)$ and a time window of length one ensure that it is a switch.
  - Then for each subsection of type (i', k') such that $b_{j,k}\beta+18i'+6k' \in [b_{j,k}\beta + 18i_1 + 6(k + 1), b_{j,k}\beta + 18i_2 + 6k)$, we define two switches chained with the previous ones:
    - One has release time $b_{j,k}\beta+18i'+6k'+1$ and a time window of length five. A fill task with release time $b_{j,k} \cdot \beta + 18i' + 6k' + 2$ and a time window of length three, together with propagation sequence $P_{i',k'}$, ensure that the three central positions of the switch are used by other tasks.
    - The other has release time $b_{j,k}\beta + 18i' + 6k' + 5$ and a time window of length three. A fill task with release time $b_{j,k} \cdot \beta + 18i' + 6(k' + 1)$ and a time window of length one ensures that it is indeed a switch.

Lemma 15. *Let $e_j = \{i_1, i_2\}$ be an edge and $k \in \{0, 1, 2\}$ be a color. Then, in any feasible schedule of instance I_{max}, $P_{i_1,k}$ and $P_{i_2,k}$ cannot be both ON.*

Proof. As bridge $B_{n+j,k}$ is an ON-to-OFF Bridge, by Lemma 6, if $P_{i_1,k}$ is ON then $P_{i_2,k}$ is OFF. □

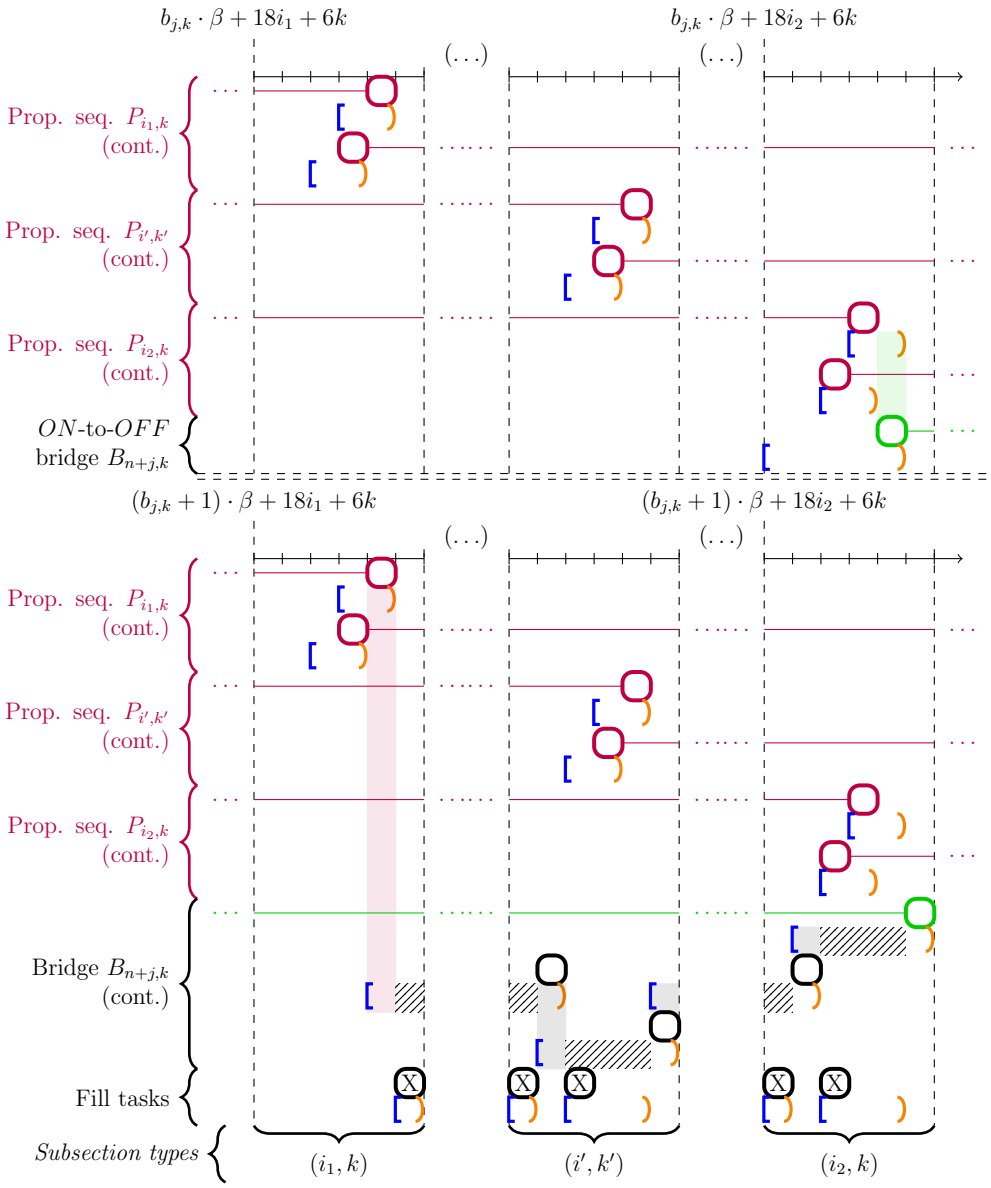


Figure 11: Edge check zone e_j in instance I_{max}. (i', k') represents any subsection type between (i_1, k) and (i_2, k).

6.4. Proof of the reduction

Remark 2. *Note that all tasks have a time window of length five of less, so we have slack $\sigma = 4$.*

Proposition 2. *G is 3-colorable if and only if there exists a feasible schedule for I_{max}*

Proof. The sketch of this proof is very similar to the proof of Proposition 1.

($\Longleftarrow$) Given a feasible schedule for instance I_{max}, we define a 3-coloring by following the starting times of the color choice tasks $\mathcal{C}_i$. By Lemma 14 the corresponding propagation sequences are ON. So, if the proposed 3-coloring were invalid, it would contradict with Lemma 15.

($\Longrightarrow$) Give a valid 3-coloring of G, we define a schedule for instance I_{max} by following the node colors. From them, we directly infer the starting times of the color choice tasks $\mathcal{C}_i$. Then the corresponding propagation sequences $P_{i,k}$ are set to ON while all the other propagation sequences are set to OFF. LongSwitches which include a task of a propagation sequence $P_{i,k}$ are set the same way as $P_{i,k}$. Following this, bridges and half-bridges are scheduled accordingly to Lemma 6, so that all tasks of the bridges are either left or right shifted, inducing no conflict with the overlapping propagation sequences nor fill task. Finally fill tasks with a time window of length three are scheduled at their release time if the overlapping propagation sequence is ON and at their deadline minus one if the overlapping propagation sequence is OFF. The resulting schedule is feasible for instance I_{max}. □

From these results and Lemma 2 we deduce the following part of Theorem 2:

Lemma 16. *$\mathcal{P}_{max}$ is para-*NP*-hard parameterized by slack σ.*

7. Reduction for exact delays

In this section we build I_{ex} a scheduling instance of $\mathcal{P}_{ex}$ where all coupled tasks have the same exact delay $\ell = \beta - 2$ with $\beta = 18n$. We define instance I_{ex} then prove that it is feasible if and only if G is 3-colorable.

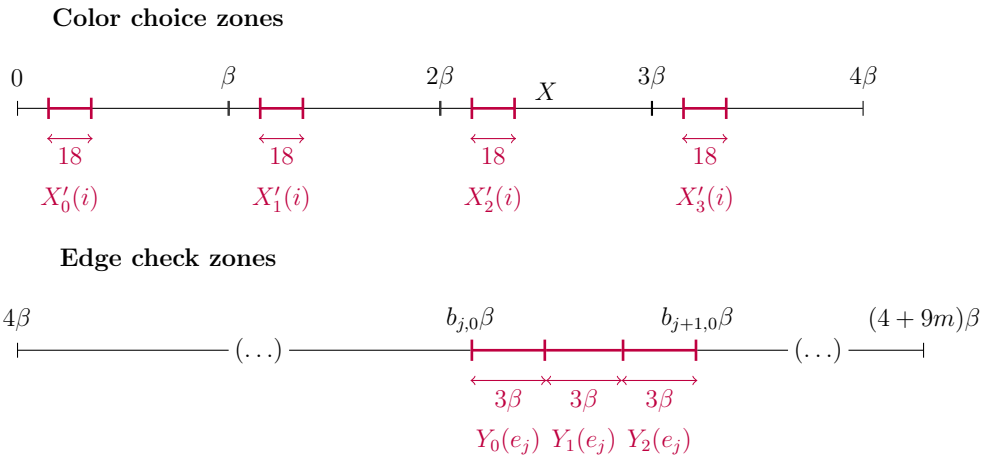


Figure 12: Timeline of the reduction for exact delays. We highlight color choice zone i and edge check zone e_j, which are described in greater detail in Figure 13 and Figure 14 respectively. Recall that $b_{j,0} = (4 + 9j)$ in this reduction.

7.1. Timeline

This time no problem relaxation was proved, so all tasks must be coupled. As such we need a bit more room in each zone to set extra tasks without hindering the base mechanisms. Color choice zones are set within time interval $[0, 4\beta)$ while edge check zones are set within time interval $[4\beta, (4+9m+1)\beta)$. For any vertex $i \in \{0, \ldots, n-1\}$ and any index $j \in \{0, 1, 2, 3\}$, we define interval:

$$X'_j(i) = [j\beta + 18i, j\beta + 18(i+1)).$$

In interval $X'_2(i)$ we initiate the color choice of node i and its propagation. The other intervals $X'_j(i)$ will be used to set the mechanisms that ensure the propagation of this information to the rest of the timeline.

In the edge check section, as in previous reductions, given j in $[0, m)$ and k in $\{0, 1, 2\}$ we define $b_{j,k} = 4 + 9j + 3k$ in order to describe concisely the part of the $(j+1)^{th}$ edge check zone where color k is checked for edge e_j. Then this segment can be written as $Y_k(e_j) = [b_{j,k} \cdot \beta, (b_{j,k} + 3) \cdot \beta)$.

7.2. Color choice zones

Let us zoom on the intervals $X'_0(i), X'_1(i), X'_2(i), X'_3(i)$ depicted in Figure 13. For each color $k \in \{0, 1, 2\}$ we set a propagation sequence $P_{i,k}$ with its release time equal to $r(P_{i,k}) = 2\beta + 18i + 6k + 2$, so that it starts in

section $X'_2(i)$. It is composed of $9m+2$ coupled tasks, and thus propagates its information until the end of the edge check zone, from left to right. The other three intervals $X'_j(i)$ associated to node i are used to set bridge $B_{i,0}$, half-bridge $B_{i,2}$, fill tasks, color choice task $\mathcal{C}_i$ and the tasks coupled to all these tasks.

As in the reduction with minimum delays, propagation sequence $P_{i,k}$ only appears in subsections of type (i,k). So we can deduce the following property:

Lemma 17. *In instance I_{ex} two tasks of two different propagation sequences cannot be performed at the same time.*

Proof. All tasks of propagation sequence $P_{i,k}$ necessarily appear in intervals of the form $[x\beta+18i+6k+2, x\beta+18i+6k+5)$. Thus we immediately get the result from the value $\ell=\beta$ and from the definition of a propagation sequence. □

In interval $X'_2(i)$ the color choice of node i is set via task $\mathcal{C}_i$. It has release time $2\beta+18i+6$ and a time window of length three. The successive positions of this task correspond to colors $0,2,1$. The other tasks in color choice zone i - see Figure 13 - link the color choice tasks to the relevant propagation sequences:

- Bridge $B_{i,0}$ is a OFF-to-ON bridge. Its left LongSwitch is composed of the first switch of propagation sequence $P_{i,0}$. Its redirect has release time $\beta+18i+3$ and a time window of length three. Its right LongSwitch is composed of a switch with release time $\beta+18i+5$ and a time window of length three.

- $B_{i,2}$ is a half-bridge with release time $\beta+18i+8$. Its LongSwitch is composed of a switch with release time $\beta+18i+12$ and a time window of length four, the coupled task of which is chained with the first switch of propagation sequence $P_{i,2}$. It has a left redirect with release time $\beta+18i+8$ and time windows of length five.

- Seven fill tasks ensure that the switches of the bridges in interval $X'_1(i)$ are well formed. The fill task with release time $\beta+18i+4$ (resp.$\beta+18i+6$) and a time window of length one ensures that the left task of the redirect (resp. the right LongSwitch) of Bridge $B_{i,0}$ is a switch. The fill tasks with release time $\beta+18i+9$, $\beta+18i+10$ and $\beta+18i+11$ (resp. $\beta+18i+13$ and $\beta+18i+14$) and a time window of length

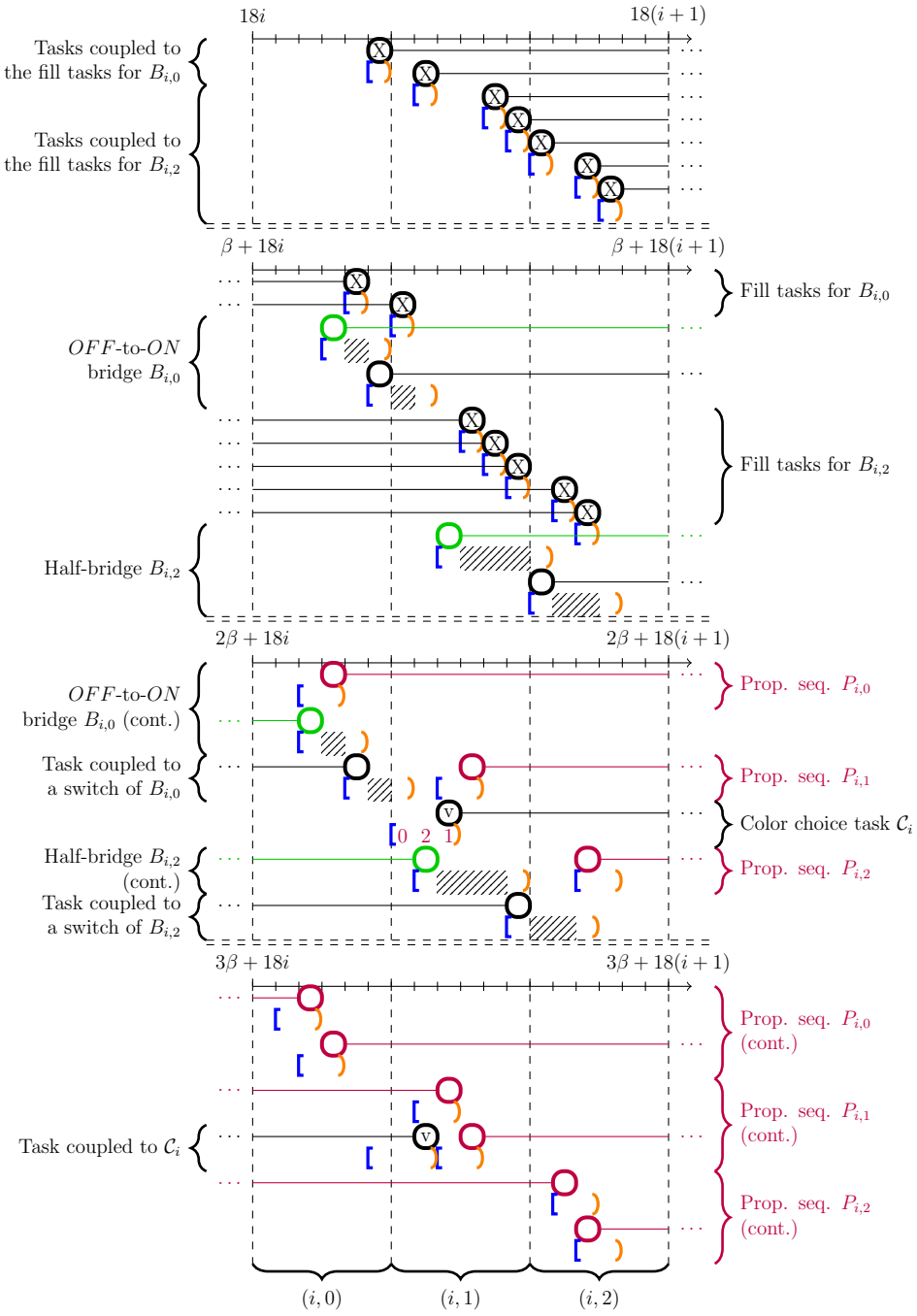


Figure 13: Color choice zone associated to node i in instance I_{ex}.

1 ensure that the left task of the redirect (resp. the LongSwitch) of half-bridge $B_{i,2}$ is a switch.

Lemma 18. *In any feasible schedule of instance I_{ex}, if color choice task $\mathcal{C}_i$ is scheduled at the position corresponding to color choice k, then propagation sequence $P_{i,k}$ is ON.*

Proof. If color choice task $\mathcal{C}_i$ is scheduled at its deadline minus one (i.e. at time $2\beta + 18i + 8$), corresponding to color choice 1, then propagation sequence $P_{i,1}$ cannot start at its release time, so by Lemma 8 it is ON. If task $\mathcal{C}_i$ is scheduled at its release time, then both the right LongSwitch of bridge $B_{i,0}$ and the task coupled to it cannot start at their deadline. So this LongSwitch is OFF which, according to Lemma 6, implies that propagation sequence $P_{i,0}$ cannot start at its release time. So again by Lemma 8 the latter is ON. Finally, if $\mathcal{C}_i$ is scheduled at time $2\beta + 18i + 7$ then it activates half-bridge $B_{i,2}$ which, according to Lemma 7, forces the task coupled to its LongSwitch to be scheduled at its deadline minus one. This prevents propagation sequence $P_{i,2}$ from starting at its release time. So by Lemma 8 the latter is ON. □

7.3. Edge check zones

Notice that unlike with minimum or maximum delays, given a task τ, it can become a switch by setting constraints not necessarily on τ itself, but on its coupled task instead. This will come into play in the edge check zones.

Given an edge e_j and a color $k \in \{0, 1, 2\}$, we describe the content of segment $Y_k(e_j) = [b_{j,k}\beta, (b_{j,k} + 3)\beta)$ where $b_{j,k} = 4 + 9j + 3k$ - see Figure 14. Assuming $e_j = \{i_1, i_2\}$ with $i_1 < i_2$, we use an ON-to-OFF bridge $B_{n+j,k}$ to relate one switch of propagation sequence $P_{i_1,k}$ with one switch of propagation sequence $P_{i_2,k}$: the former will be connected to its left LongSwitch while the latter will be its right LongSwitch. We also use $6(i_2 - i_1) + 1$ fill tasks to set the bridge. Bridge $B_{n+j,k}$ is defined as follows :

- Its redirect has release time $(b_{j,k}+1)\beta+18i_2+6k-1$ and a time window of length six.
- two fill tasks with release time $(b_{j,k} + 1)\beta + 18i_2 + 6k$ - one with a time window of length four, the other with a time window of length two - which, together with propagation sequence $P_{i_2,k}$, ensure that left task of the redirect is a switch.

- Its right LongSwitch is the switch of propagation sequence $P_{i_2,k}$ with release time $(b_{j,k}+1)\beta + 18i_2 + 6k + 2$.
- Its left LongSwitch is composed of a sequence of switches:
  - Its first switch is the switch of propagation sequence $P_{i_1,k}$ with release time $(b_{j,k}+2)\beta + 18i_1 + 6k + 2$.
  - This switch is chained with another switch with release time $(b_{j,k}+2)\beta + 18i_1 + 6k + 3$ and a time window of length three. A fill task with release time $(b_{j,k}+2)\cdot\beta + 18i_1 + 6k + 4$ and a time window of length one ensures that it is a switch.
  - Then for each subsection of type (i', k') such that $(b_{j,k}+2)\beta + 18i' + 6k' \in [(b_{j,k}+2)\beta + 18i_1 + 6(k+1), (b_{j,k}+2)\beta + 18i_2 + 6k)$, we define up to two switches chained with the previous ones:
    - one switch with release time $(b_{j,k}+2)\beta + 18i' + 6k' - 1$ and a time window of length six. A fill task with release time $(b_{j,k}+2)\beta + 18i' + 6k' + 2$ and a time window of length four, together with its coupled task and propagation sequence $P_{i',k'}$, ensure that it is a switch - see Lemma 19.
    - If we are not right before the subsection of type (i_2, k): another switch with release time $(b_{j,k}+2)\beta + 18i' + 6k' + 4$ and a time window of length two.

Lemma 19. *For every edge e_j and color k, $B_{n+j,k}$ is an ON-to-OFF bridge in instance I_{ex}.*

Proof. The only nontrivial switches in $B_{n+j,k}$ are the tasks with release time $(b_{j,k}+2)\beta + 18i' + 6k' - 1$ and time window length six, with (i', k') subsection type between (i_1, k) and (i_2, k). Let τ' be one of them, τ be its (left) coupled task, f' be the associated fill task with a time window of length four and f be its (left) coupled task. By contradiction, given some feasible schedule of I_{ex}, suppose that τ' starts at time $(b_{j,k}+2)\beta + 18i' + 6k' + x$ with $x \in [0, 3]$.

- If $x \in \{0, 1\}$ then the first two positions of task f are blocked. Since f' is coupled to it and $r(f') = r(f) + \ell + 1$, this also blocks the first two positions of f. Plus this forces the two tasks from propagation sequence $P_{i_2,k}$ which overlap the time window of f' to be ON, which blocks the last two remaining positions of f' and leads to a contradiction.

- If $x \in \{2, 3\}$ we obtain a contradiction similarly, starting from the last two positions of f' this time.

So $B_{n+j,k}$ is an ON-to-OFF bridge. □

Lemma 20. *Let $e_j = \{i_1, i_2\}$ be an edge and $k \in \{0, 1, 2\}$ be a color. Then, in any feasible schedule of instance I_{ex}, $P_{i_1,k}$ and $P_{i_2,k}$ cannot be both ON.*

Proof. By Lemma 19 $B_{n+j,k}$ is an ON-to-OFF Bridge. So, by Lemma 6, if $P_{i_1,k}$ is ON then $P_{i_2,k}$ is OFF. □

7.4. Proof of the reduction

Remark 3. *Note that all tasks have a time window of length six of less, so we have slack $\sigma = 5$.*

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

Proof. The sketch of this proof is very similar to the proof of Proposition 1.

($\Longleftarrow$) Given a feasible schedule for instance I_{ex}, we define a 3-coloring by following the starting times of the color choice tasks $\mathcal{C}_i$. By Lemma 18 the corresponding propagation sequences are ON. So, if the proposed 3-coloring were invalid, it would contradict with Lemma 20.

($\Longrightarrow$) Give a valid 3-coloring of G, we define a schedule for instance I_{ex} by following the node colors. From them, we directly infer the starting times of the color choice tasks $\mathcal{C}_i$. Then the corresponding propagation sequences $P_{i,k}$ are set to ON while all the other propagation sequences are set to OFF. LongSwitches which include a task of a propagation sequence $P_{i,k}$ are set the same way as $P_{i,k}$. Following this, bridges and half-bridges are scheduled accordingly to Lemma 6, so that all tasks of the bridges are either left or right shifted, inducing no conflict with the overlapping propagation sequences nor fill task. Finally, in the edge check zones, fill tasks with a time window of length four are scheduled at their release time if the overlapping propagation sequence is ON and at their deadline minus one if the overlapping propagation sequence is OFF. The resulting schedule is feasible for instance I_{ex}. □

From these results we deduce the following part of Theorem 2:

Lemma 21. *$\mathcal{P}_{ex}$ is para-NP-hard parameterized by slack σ.*

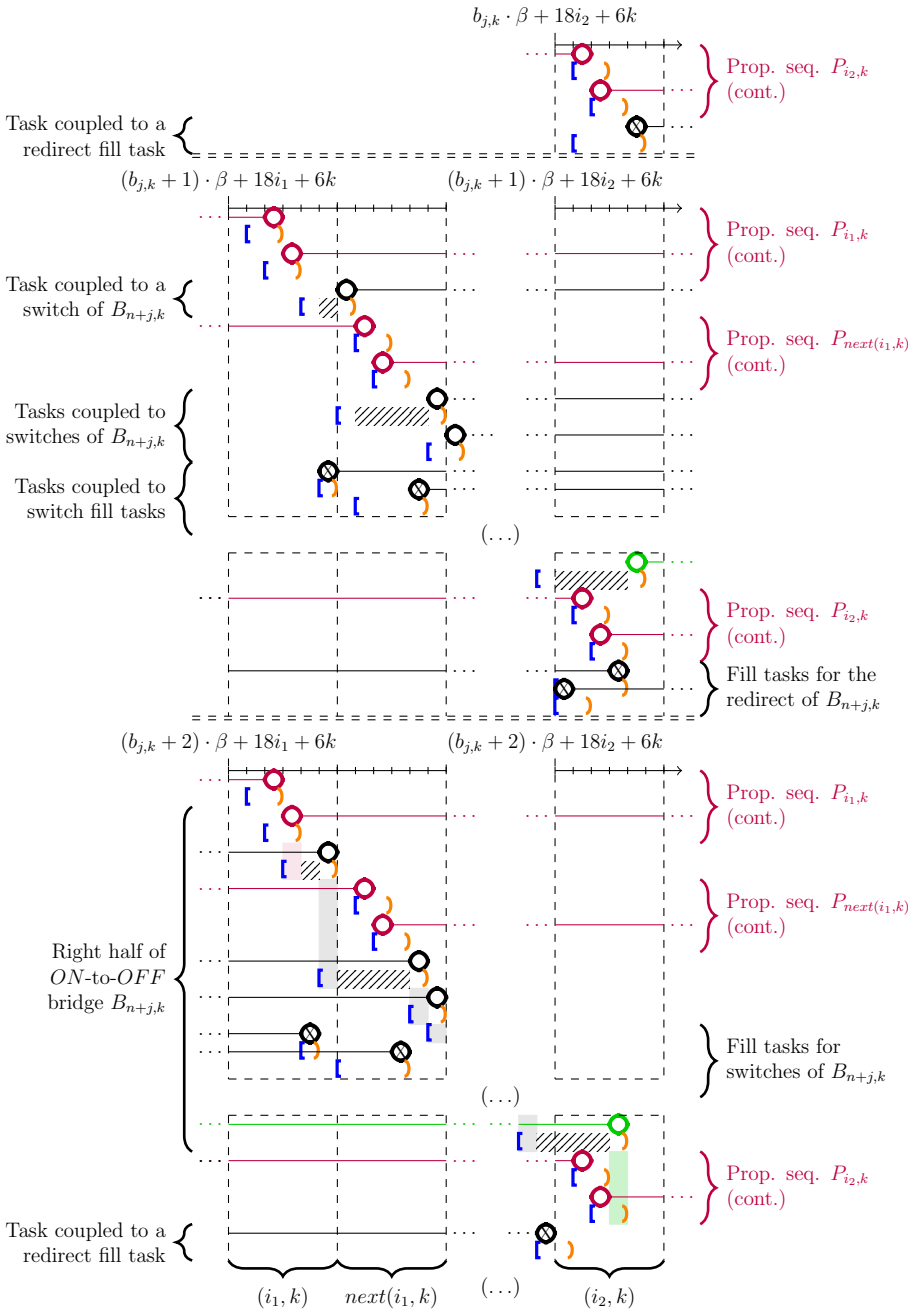


Figure 14: Segment $Y_k(e_j)$ in edge check zone e_j of instance I_{ex}. Given a couple (i',k'), function *next* returns $(i'+1,0)$ if $k'=2$ and $(i',k'+1)$ otherwise.

8. Conclusion

In this paper, for all three precedence delay types, we proved that scheduling coupled tasks with time windows on a single machine is para-NP-complete with respect to parameter couple "pathwidth and slack", even with unit-time tasks and equal-length delays. This shows that, unlike the setting without time windows, the range of processing times, the chain lengths and the number of delay values do not significantly change the difficulty of the problem. In other words, it remains NP-complete even in the most simple variant of these three properties combined. Leaving aside the number of tasks, this leaves the value of the delays as the only remaining unbounded property. essentially identifying it as the main reason behind the difficulty of single-machine scheduling with time windows and precedence delays. As such, it seems natural to have the maximum delay value among the next parameters to consider.

Additionally, one could consider parameters which are specific to the coupled-task setting. For instance Bessy and Giroudeau [13] added a compatibility graph which limits which coupled tasks can interleave each other - i.e. whenever the tasks of a given coupled task can be processed in between the execution of the two tasks of another coupled task. With exact delays, they showed that coupled-task scheduling on a single machine is fixed-parameter tractable parameterized by the maximum coupled task length plus the vertex cover number of this graph as the parameter. It would be interesting to explore whether this result still holds when time windows are added to the problem.

Finally, we claim that parameterized complexity analysis provides a new insight on the difficulty of scheduling problems. However one of the challenging question is to based upon these positive and negative results to design more efficient exact and approximated algorithms.

References

[1] C. U. Martel, Preemptive scheduling with release times, deadlines, and due times, J. ACM 29 (3) (1982) 812–829.

[2] E. Lawler, Optimal sequencing of a single machine subject to precedence constraints, Management Sci. 19 (1973) 544–546.

[3] J. Lenstra, A. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Ann. of Discrete Math. 1 (1977) 343–362.

[4] U. Dorndorf, E. Pesch, T. Phan-Huy, A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints, Management Science 46 (2000) 1365–1384.

[5] Bruno, Jones, K. So, Deterministic scheduling with pipelined processors, IEEE Transactions on Computers C-29 (4) (1980) 308–316, conference Name: IEEE Transactions on Computers.

[6] W. Yu, H. Hoogeveen, J. K. Lenstra, Minimizing Makespan in a Two-Machine Flow Shop with Delays and Unit-Time Operations is NP-Hard, Journal of Scheduling 7 (5) (2004) 333–348.

[7] B. Chen, X. Zhang, Scheduling coupled tasks with exact delays for minimum total job completion time, J. Sched. 24 (2) (2021) 209–221.

[8] M. Khatami, A. Salehipour, T. Cheng, Coupled task scheduling with exact delays: Literature review and models, European Journal of Operational Research 282 (1) (2020) 19–39.

[9] W. Yu, The two-machine flow shop problem with delays and the one-machine total tardiness problem, Ph.D. thesis, Eindhoven University of Technology, publisher: Technische Universiteit Eindhoven (1996).

[10] S. M. Johnson, Optimal two-and three-stage production schedules with setup times included, Naval research logistics quartThe Two-Machine Flow Shop Problem with Delays and the One-Machine Total Tardiness Problemerly 1 (1) (1954) 61–68.

[11] M. Mnich, A. Wiese, Scheduling and fixed-parameter tractability, Mathematical Programming 154 (1-2) (2015) 533–562.

[12] R. van Bevern, R. Bredereck, L. Bulteau, C. Komusiewicz, N. Talmon, G. J. Woeginger, 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, Springer International Publishing, Cham, 2016, pp. 105–120.

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

[14] H. L. Bodlaender, M. van der Wegen, Parameterized complexity of scheduling chains of jobs with delays, in: 15th International Symposium on Parameterized and Exact Computation (IPEC), 2020, pp. 1–15.

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

[16] C. Hanen, A. Munier-Kordon, Fixed-Parameter tractability of scheduling dependent typed tasks subject to release times and deadlines, Journal of Scheduling 27 (2) (2024) 119–133.

[17] C. Hanen, M. Mallem, A. Munier-Kordon, Parameterized complexity of single-machine scheduling with precedence, release dates and deadlines, in: 15th Workshop on Models and Algorithms for Planning and Scheduling Problems (MAPSP), 2022, pp. 131–134.

[18] R. Baart, M. de Weerdt, L. He, Single-machine scheduling with release times, deadlines, setup times, and rejection, European Journal of Operational Research 291 (2) (2021) 629–639, number: 2 Publisher: Elsevier.

[19] M. Mallem, C. Hanen, A. Munier-Kordon, Parameterized complexity of a parallel machine scheduling problem, in: 17th International Symposium on Parameterized and Exact Computation (IPEC), 2022, pp. 1–21.

[20] R. L. Graham, E. L. Lawler, J. K. Lenstra, A. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, in: Annals of discrete mathematics, Vol. 5, Elsevier, 1979, pp. 287–326.

[21] R. van Bevern, R. Niedermeier, O. Suchỳ, A parameterized complexity view on non-preemptively scheduling interval-constrained jobs: few machines, small looseness, and small slack, Journal of Scheduling 20 (2017) 255–265. doi:https://doi.org/10.1007/s10951-016-0478-9.