



HAL
open science

Parameterized complexity of a parallel machine scheduling problem

Maher Mallem, Claire C. Hanen, Alix Munier Kordon

► **To cite this version:**

Maher Mallem, Claire C. Hanen, Alix Munier Kordon. Parameterized complexity of a parallel machine scheduling problem. International Symposium on Parameterized and Exact Computation (IPEC), Sep 2022, Postdam, Germany. <hal-03834526>

HAL Id: hal-03834526

<https://hal.science/hal-03834526v1>

Submitted on 1 Apr 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons CC BY 4.0 - Attribution - International License

1 Parameterized complexity of a parallel machine 2 scheduling problem

3 **Maher Mallem** ✉

4 Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

5 **Claire Hanen** ✉ 

6 Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

7 Université Paris Nanterre, UPL, 92000 Nanterre, France

8 **Alix Munier-Kordon** ✉ 

9 Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

10 — Abstract —

11 In this paper we consider the parameterized complexity of two versions of a parallel machine
12 scheduling problem with precedence delays, unit processing times and time windows. In the first
13 version - with exact delays - we assume that the delay between two jobs must be exactly respected,
14 whereas in the second version - with minimum delays - the delay between two jobs is a lower bound
15 on the time between them. Two parameters are considered for this analysis: the pathwidth of
16 the interval graph induced by the time windows and the maximum precedence delay value. We
17 prove that our problems are para-NP-complete with respect to any of the two parameters and
18 fixed-parameter tractable parameterized by the pair of parameters.

19 **2012 ACM Subject Classification** Theory of computation → Problems, reductions and completeness

20 **Keywords and phrases** parameterized complexity, scheduling, precedence delays, pathwidth, chains,
21 parallel processors

22 **Digital Object Identifier** 10.4230/LIPIcs.IPEC.2022.1

23 **1** Introduction

24 While scheduling jobs with resources and precedence constraints has a wide range of industrial
25 applications, these problems have been proven to be NP-hard in the strong sense even for very
26 simple settings. For example minimizing the makespan of a schedule of jobs on two parallel
27 processors assuming chain-like precedence constraints is already NP-hard [8]. Scheduling
28 problems are usually denoted by the three field denotation $\alpha|\beta|\gamma$ of Graham [11]. Field α
29 describes the machine setting, field β describes the job relations and properties, and field γ
30 defines the optimization criterion - or is a star \star for a decision problem. For example the
31 NP-hard problem described above is denoted by $P2|chains|C_{\max}$.

32 In this paper we tackle two decision scheduling problems on M parallel processors. We
33 consider a set of n jobs \mathcal{T} . Each job i has a unit processing time and a time interval $[r_i, d_i]$
34 given for its computation. r_i is the release date of job i , and d_i its deadline. Jobs are
35 linked by precedence constraints defined by an acyclic precedence graph G with non-negative
36 precedence delays $\ell_{i,j}$ on each arc (i, j) of G . Two variants of the precedence delays are
37 considered: exact precedence delays and minimum precedence delays, which will be denoted
38 by $\ell_{i,j}^{ex}$ and $\ell_{i,j}^{min}$ in the standard notations.

39 A feasible schedule σ defines for each job $i \in \mathcal{T}$ a starting time $\sigma(i) \in [r_i, d_i]$ so that no
40 more than M jobs are scheduled at the same time, and so that for each arc (i, j) of G :

- 41 ■ in case of exact delays: $\sigma(i) + 1 + \ell_{i,j}^{ex} = \sigma(j)$,
- 42 ■ in case of minimum delays: $\sigma(i) + 1 + \ell_{i,j}^{min} \leq \sigma(j)$.

43 In particular we will explore the parameterized complexity of these problems with chain-
44 like precedence constraints, denoted by $P|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|\star$ and $P|chains(\ell_{i,j}^{min}), p_j =$



© Maher Mallem, Claire Hanen and Alix Munier-Kordon;
licensed under Creative Commons License CC-BY 4.0

17th International Symposium on Parameterized and Exact Computation (IPEC 2022).



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

45 $1, r_j, d_j | \star$.

46 When restricting all processing times to be equal to one, NP-hardness results typically
 47 depend on the precedence relations used in the problem. Ullman [19] first showed that
 48 $P|prec, p_j = 1 | C_{max}$ is strongly NP-hard. This was later improved by Lenstra et al [14] with
 49 a precedence graph of bounded height, then by Garey et al [10] in the case of an opposing
 50 forest. When we have release dates Brucker et al [5] showed that $P|intree, p_j = 1, r_j | C_{max}$
 51 is strongly NP-hard with an intree as the precedence graph.

52 When further restricting the problem to chains with precedence delays, Yu et al [22]
 53 showed that $1|chains(\ell_{i,j}^{min}), p_j = 1 | C_{max}$ and $1|chains(\ell_{i,j}^{ex}), p_j = 1 | C_{max}$ are strongly NP-
 54 hard, even when all chains are of length two (i.e. if we only have coupled tasks). The
 55 problem $1|chains(\ell_{i,j}^{ex}), p_j = 1 | \star$ was also proven NP-hard earlier by Orman in [17]. Note that
 56 apart from the delays themselves, there is nothing more to restrict in order to go around
 57 NP-hardness. Thus within the scope of classical complexity theory it becomes difficult to find
 58 the frontier between polynomial-time solvability and NP-hardness when delays are considered.

59 Parameterized complexity theory gives numerous tools for a refined analysis of such hard
 60 scheduling problems. Given a parameter k and denoting n the input size, a problem is called
 61 *fixed-parameter tractable* (FPT) with respect to parameter k if it can be solved in time
 62 $poly(n) \times f(k)$ with f an arbitrary function [6]. Not only it tells us more about a problem
 63 than if we only showed NP-hardness, it allows us to further classify the NP-hard problems
 64 depending on whether they are FPT parameterized by k or not.

65 When the studied problem is believed to not be FPT, the para-NP class is used as a
 66 parameterized version of NP: a problem is in para-NP with respect to parameter k if there
 67 is a non-deterministic algorithm that solves it in time $poly(n) \times f(k)$ with f an arbitrary
 68 function. In order to prove that a problem is para-NP-hard with respect to k , it is enough to
 69 prove that the un-parameterized problem is NP-hard for some fixed value of the parameter [9].
 70 The W-hierarchy defined in [7] is an additional widely used tool in parameterized complexity.
 71 It is a sequence of intermediate complexity classes between FPT and para-NP, which allows
 72 us to further investigate the time complexity of a parameterized problem.

73 Until now quite few parameterized complexity results have been proved for scheduling
 74 problems. Among the first ones related to our problems, Bodlaender proved in [3] that
 75 minimizing the makespan of unit processing times jobs on parallel machines is W[2]-Hard
 76 considering the number of machines as parameter. When precedence constraints are involved,
 77 several authors studied the parameter defined by the width w of the precedence graph. Van
 78 Bevern et al. [20] proved that the problem $P2|prec, p_j \in \{1, 2\} | C_{max}$ is W[2]-hard with
 79 respect to the width. In the same paper they also proved that if we add the maximum
 80 allowed lag of a job with respect to its earliest start time (ignoring resource constraints) to
 81 the width as parameter then the problem becomes FPT, even for more complex resource
 82 constraints (RCPSP). In [15] it is noted that the parameterized complexity of the problem
 83 with three processors, precedence constraints and unit execution times with respect to width
 84 w is still open.

85 Now considering precedence delays, we can first mention the work of Bessy et al [2]
 86 on coupled tasks with due dates, a compatibility graph and a common deadline on a
 87 single machine. They consider the number of jobs that end before the due date as their
 88 parameter. They establish W[1]-hardness for this problem and propose a FPT algorithm
 89 when the maximum duration of a job (i.e. the processing time of the two tasks plus their
 90 delay) is bounded. Bodlaender et al in [4] studied the two problems we address in this
 91 paper but assuming that each chain C has a time window $[r_C, d_C)$ instead of time windows
 92 for individual jobs like in our case. They considered two parameters: the first one is

the number of chains and the second one is the thickness, i.e. the maximum number of overlapping chain time windows. On one hand they proved that for exact or minimum delays $1|chains(\ell_{i,j}), p_j = 1, r_C, d_C|*$ is $W[1]$ -hard with any of the two parameters, while the parallel machine variant $P|chains(\ell_{i,j}), p_j = 1, r_C, d_C|*$ is $W[2]$ -hard. On the other hand they proved that these problems are in XP (i.e. deterministic $n^{f(k)}$ time) if the delays are unary encoded.

Considering scheduling problems with job time windows, the pathwidth μ has been considered recently as a parameter by several authors [1, 16, 21]. Parameter μ is the pathwidth of the graph induced by jobs time intervals. It can be easily computed since $\mu + 1$ is the maximum number of overlapping job time windows at any given time. In particular Munier proved in [16] that scheduling unit processing times jobs with time windows and precedence constraints is FPT with parameter μ . Note that van Bevern et al. [20] showed that $P2|prec, p_j \in \{1, 2\}|C_{max}$ is $W[2]$ -hard parameterized by w , while this problem is likely to be FPT parameterized by μ according to Hanen and Munier [12]. With μ allowing such problems to be FPT when other parameters do not, it makes it all the more difficult to find hardness results relative to μ .

The two studied parameters in our paper are pathwidth μ and the maximum precedence delay $\ell_{max} = \max_{(i,j)} \text{arc of } G \ell_{i,j}$. We first show in Section 2 that both decision problems $P|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ and $P|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ are para-NP-complete parameterized by either one alone. In the case of pathwidth μ as the parameter we even prove that the single machine variant of both problems are para-NP-complete. Then in section 3 we prove that these problems are FPT parameterized by the couple (μ, ℓ_{max}) .

2 Hardness reductions

In this section we prove para-NP-completeness of our two parallel machine scheduling problems parameterized by pathwidth μ or maximum delay ℓ_{max} alone. We even establish para-NP-completeness of the single-machine variant in the case of parameter μ .

We use the following result from Flum and Grohe's book [9]:

► **Lemma 1** (Flum, Grohe 1998). *If a (nontrivial) problem \mathcal{P} is NP-complete with a fixed value of some parameter k , then the parameterized problem (\mathcal{P}, k) is para-NP-complete.*

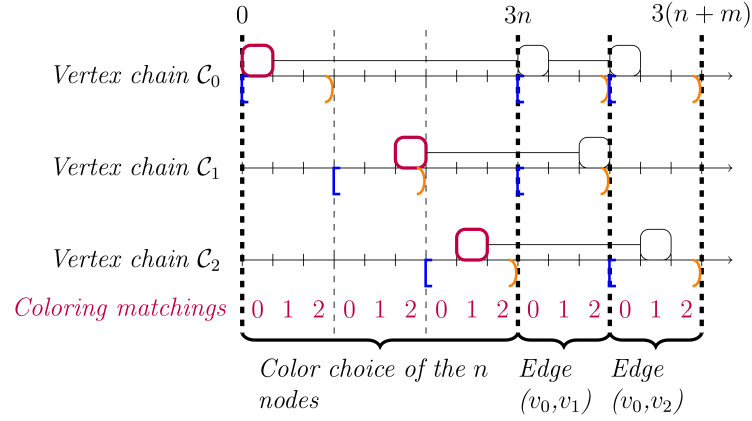
Thus the following scheduling problems will be shown to be NP-complete:

1. $1|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ with $\mu = 1$,
2. $1|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ with $\mu = 2$,
3. $P|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ with $\ell_{max} = 1$,
4. $P|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ with $\ell_{max} = 1$.

These problems are all trivially in NP by guessing the starting time of each job then checking if this leads to a feasible schedule. For the hardness proofs all reductions will start from the (strongly) NP-hard 3-COLORING graph problem [13]. Let $G = (V, E)$ be the input graph. Let v_0, \dots, v_{n-1} be the vertices in V and e_0, \dots, e_{m-1} be the edges in E . Let $n = |V|$ and $m = |E|$. The colors will be named 0, 1 and 2.

2.1 NP-hardness of $1|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ with $\mu = 1$

We build an instance of $1|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ with $\mu = 1$. An example is given in Figure 1. We have n vertex chains \mathcal{C}_i with $\deg(v_i) + 1$ jobs in chain \mathcal{C}_i , $0 \leq i \leq n - 1$ with $\deg(v_i)$ the degree of node v_i in G . We define vertex chain \mathcal{C}_i the following way:



■ **Figure 1** An instance of $1|chains(\ell_{i,j}^{ex}, p_j = 1, r_j, d_j)|\star$ representing a graph coloring. We have $G = (V, E)$ with $V = \{v_0, v_1, v_2\}$ and $E = (\{v_0, v_1\}, \{v_0, v_2\})$. This schedule corresponds to the coloring $(0, 2, 1)$.

136 ► **Definition 2** (Vertex chain C_i). We segment time into $m + 1$ segments: a color choice
 137 segment $[0, 3n)$ and m edge check selection segments of length 3 along $[3n, 3(n + m))$. We
 138 describe the chain from left to right:

- 139 (1) Color choice segment $[0, 3n)$
 140 ■ The first job of chain C_i has time window $[3i, 3(i + 1))$.
 141 a. If v_i appears in no edge of G : end the chain.
 142 b. Else: set $3(n - i) - 1$ as the current exact delay after this job.
- 143 (2) Edge check segment $[3(n + j), 3(n + j + 1))$, $0 \leq j \leq m - 1$
 144 For j in $[0, m - 1]$:
 145 Let edge $e_j = \{v_{i_1}, v_{i_2}\}$, $i_1 < i_2$.
 146 a. Vertex chain C_i with $i \notin \{i_1, i_2\}$
 147 ■ Add 3 to the current exact delay after the currently latest job of chain C_i .
 148 b. Vertex chain C_i with $i = i_1$ or $i = i_2$
 149 ■ Set a job with time window $[3(n + j), 3(n + j + 1))$
 150 i. If e_j is the last edge where v_i appears: end the chain.
 151 ii. Else: set 2 as the current exact delay after this job.

152 ► **Remark 3.** The created instance has pathwidth 1. In the color choice segment: for
 153 $i \in [0, n - 1]$ there is exactly one job to be scheduled in time window $[3i, 3(i + 1))$: the first
 154 job of vertex chain C_i . In the edge check segments: for $j \in [0, m - 1]$ if edge $e_j = \{v_{i_1}, v_{i_2}\}$
 155 then there are exactly two jobs to be scheduled in time window $[3(n + j), 3(n + j + 1))$: one
 156 from vertex chain C_{i_1} and one from vertex chain C_{i_2} . Thus there are indeed at most two
 157 overlapping time windows at any given time.

158 Let $i \in [0, n - 1]$. Vertex chain C_i has three possible starting times in $[3i, 3(i + 1))$ which
 159 corresponds to the three color choices of node v_i . Then this color choice is propagated
 160 to every edge check segment $[3(n + j), 3(n + j + 1))$ where node v_i is a part of edge e_j ,
 161 $j \in [0, m - 1]$. The following lemma ensures that the color choices are faithfully propagated.

162 ► **Lemma 4.** Let $0 \leq i \leq n - 1$. In any feasible schedule, if vertex chain C_i starts at time
 163 $3i + k$ with $k \in \{0, 1, 2\}$, then all jobs J in this chain are scheduled at time $r(J) + k$, where
 164 $r(J)$ is the release date of job J .

165 **Proof.** Suppose we have a feasible schedule where vertex chain \mathcal{C}_i starts at time $3i + k$ with
 166 $k \in \{0, 1, 2\}$.

167 ■ If vertex v_i is part of no edge in the graph:

168 Then by Definition 2 vertex chain \mathcal{C}_i only has one job. It is scheduled at time $3i + k$,
 169 which is indeed the release date of this job plus k .

170 ■ If vertex v_i is part of at least one edge in the graph:

171 Let $j_0 < j_1 < \dots < j_{deg(v_i)-1}$ be the indices of the edges e_j such that $v_i \in e_j$. We prove
 172 by induction on $l \in [0, deg(v_i) - 1]$ that the job of vertex chain \mathcal{C}_i which has time window
 173 $[3(n + j_l), 3(n + j_l + 1))$ is scheduled at time $3(n + j_l) + k$.

174 ■ Consider the job of vertex chain \mathcal{C}_i which has time window $[3(n + j_0), 3(n + j_0 + 1))$.
 175 By Definition 2 this job is the successor of the first job in the chain and the exact delay
 176 between them is $3(n - i) - 1 + 3j_0$. Thus if the first job of the chain is scheduled at time
 177 $3i + k$, then this following job is scheduled at time $(3i + k) + 1 + (3(n - i) - 1 + 3j_0) =$
 178 $3(n + j_0) + k$, which is indeed the release date of this job plus k .

179 ■ Let $l \in [1, deg(v_i) - 1]$. Suppose the job of vertex chain \mathcal{C}_i which has time window
 180 $[3(n + j_{l-1}), 3(n + j_{l-1} + 1))$ is scheduled at time $3(n + j_{l-1}) + k$. Then by Definition 2
 181 there is an exact delay $2 + 3(j_l - j_{l-1} - 1)$ before the next job of the chain. Thus the
 182 next job of the chain is scheduled at time $(3(n + j_{l-1}) + k) + 1 + (2 + 3(j_l - j_{l-1} - 1)) =$
 183 $3(n + j_l) + k$, which is indeed the release date of this job plus k .

184 This proves the lemma for all the jobs of vertex chain \mathcal{C}_i in an edge check segment. ◀

185 Then for each edge $e_j = \{v_{i_1}, v_{i_2}\}, i_1 < i_2$, the color choices of v_{i_1} and v_{i_2} are confronted
 186 in edge check segment $[3(n + j), 3(n + j + 1))$. If both nodes chose the same color then both
 187 jobs in this edge check segment would be scheduled at the same time, which would invalidate
 188 our schedule in this single-machine instance. Conversely if we start from a valid coloring,
 189 then there will never be two jobs scheduled at the same time in an edge check segment. This
 190 is the key ingredient behind the reduction.

191 ► **Proposition 5.** *G is 3-colorable if and only if there exists a feasible schedule for this*
 192 *instance of EXACT DELAYS.*

193 **Proof.** (\implies) Suppose we have $(c_0, \dots, c_{n-1}) \in \{0, 1, 2\}^n$ a 3-coloring of G where vertex v_i
 194 has color c_i . We propose the schedule where for all $0 \leq i \leq n - 1$, chain \mathcal{C}_i starts at time
 195 $3i + c_i$. Then in every edge $e_j \in E$ where vertex v_i appears, we schedule the job of chain \mathcal{C}_i
 196 which is in edge check segment $[3(n + j), 3(n + j + 1))$ at time $3(n + j) + c_i$.

197 We show that the jobs in different chains do not interfere with each other. Since the time
 198 windows do not overlap in the color choice segment, only the edge check segments remain
 199 to be checked. Let $e_j = \{v_{i_1}, v_{i_2}\}$ be an edge in E . By definition of the vertex chains, only
 200 chains \mathcal{C}_{i_1} and \mathcal{C}_{i_2} have a job to be scheduled in time window $[3(n + j), 3(n + j + 1))$. In
 201 our schedule the job of chain \mathcal{C}_{i_1} is scheduled at time $3(n + j) + c_{i_1}$ and the job of chain
 202 \mathcal{C}_{i_2} at time $3(n + j) + c_{i_2}$. Since (c_0, \dots, c_{n-1}) is a 3-coloring and $\{v_{i_1}, v_{i_2}\} \in E$, we have
 203 $c_{i_1} \neq c_{i_2}$. Thus both jobs are scheduled at different times and the jobs in edge check segment
 204 $[3(n + j), 3(n + j + 1))$ do not interfere with each other. Thus the proposed schedule is
 205 feasible.

206 (\impliedby) Suppose we have a feasible schedule. For all $0 \leq i \leq n - 1$, let $s_i \in \{0, 1, 2\}$ be
 207 such that $3i + s_i$ is the starting time of chain \mathcal{C}_i . We show that (s_0, \dots, s_{n-1}) is a 3-coloring
 208 of G . By contradiction suppose there is an edge $e_j = \{v_{i_1}, v_{i_2}\} \in E$ such that $s_{i_1} = s_{i_2}$.
 209 Then by Lemma 4 the jobs of chains i_1 and i_2 that must be scheduled in edge check segment
 210 $[3(n + j), 3(n + j + 1))$ are scheduled at the same time $3(n + j) + s_{i_1}$. Thus the schedule is not
 211 feasible, which leads to a contradiction. Thus (s_0, \dots, s_{n-1}) is indeed a 3-coloring of G . ◀

212 This proves that $1|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ with $\mu = 1$ is NP-hard, which concludes
 213 the para-NP-completeness proof of the corresponding parameterized problem.

214 ► **Theorem 6.** $1|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ is para-NP-complete parameterized by pathwidth
 215 μ .

216 2.2 NP-hardness of $1|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ with $\mu = 2$

217 We build an instance of $1|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ with $\mu = 2$. We begin in a similar way:
 218 for each node we have a vertex chain C'_i with three possible starting times, each corresponding
 219 to a color choice, then we want to propagate this color choice. However now that the delays
 220 are not exact anymore, the color choice cannot be propagated properly as previously. More
 221 constraints are needed in order to deal with the extra flexibility coming from the minimum
 222 delays.

223 One way is to add a closing segment $[3(n+m), 3(2n+m))$ at the end and two gadget
 224 chains $C'_{i,1}, C'_{i,2}$ per node, each composed of two jobs. As shown in Figure 2 the gadget chains
 225 will fill the two gaps at the start and at the end of each vertex chain.

226 ► **Definition 7** (Vertex chain C'_i). *We segment time into $m+2$ segments: a color selection
 227 segment $[0, 3n)$, m edge check selection segments along $[3n, 3(n+m))$ and a closing segment
 228 $[3(n+m), 3(2n+m))$. We describe the chain from left to right:*

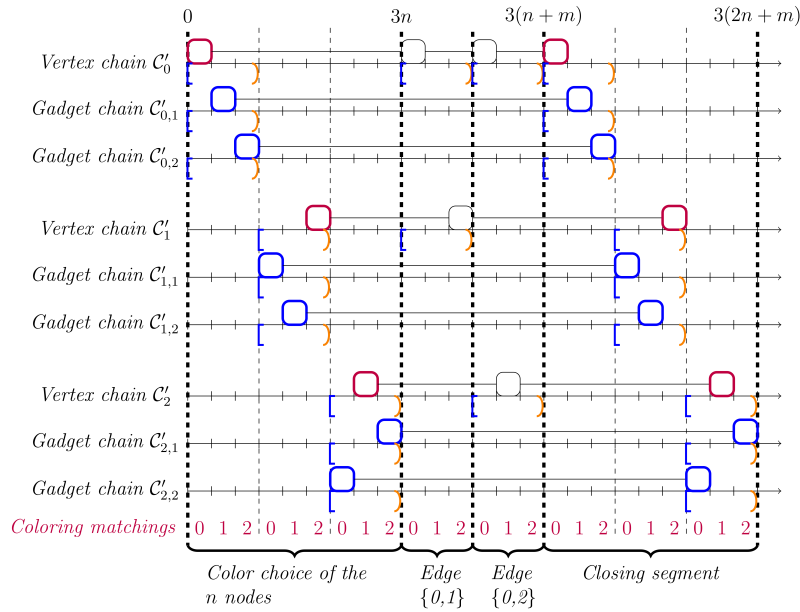
- 229 (1) *Color selection segment $[0, 3n)$*
 230 ■ *The first job of chain C'_i has time window $[3i, 3(i+1))$.*
 231 ■ *Set $3(n-i) - 1$ as the current minimum delay after this job.*
 232 (2) *Edge check segment $[3(n+j), 3(n+j+1))$, $0 \leq j \leq m-1$*
 233 *For j in $[0, m-1]$:*
 234 *Let edge $e_j = \{v_{i_1}, v_{i_2}\}$, $i_1 < i_2$.*
 235 a. *Vertex chain C'_i with $i \notin \{i_1, i_2\}$*
 236 ■ *Add 3 to the current minimum delay after the currently latest job of chain C'_i*
 237 b. *Vertex chain C'_i with $i = i_1$ or $i = i_2$*
 238 ■ *Set a job with time window $[3(n+j), 3(n+j+1))$*
 239 ■ *Set 2 as the current minimum delay after this job*
 240 (3) *Closing segment $[3(n+m), 3(2n+m))$*
 241 ■ *Add $3i$ to the current minimum delay of the currently latest job of chain C'_i .*
 242 ■ *Set a job with time window $[3(n+i+m), 3(n+i+1+m))$ as the last job of vertex
 243 chain C'_i .*

244 ► **Definition 8** (Gadget chains $C'_{i,1}, C'_{i,2}$). *For both gadget chains $C'_{i,1}, C'_{i,2}$ relative to vertex v_i ,
 245 the first job must be scheduled in time window $[3i, 3(i+1))$, the second one in time window
 246 $[3(n+m+i), 3(n+m+i+1))$, and there is a minimum delay $3(n+m) - 1$ between them.*

247 ► **Remark 9.** The created instance has indeed pathwidth 2: gadget chains add two more time
 248 windows at the beginning and the end of each vertex chain, so there are at most three time
 249 windows overlapping at any time.

250 A full example is given in Figure 2. For our proof the goal is to show that adding these
 251 gadget chains is enough to get an analogue result to Lemma 4. Note that if we only use the
 252 definition of the chains like in the reduction of Section 2.1, we only get this weaker result:

253 ► **Lemma 10.** *Let $0 \leq i \leq n-1$. In any feasible schedule, if a chain starts at time $3i+k$
 254 with $k \in \{0, 1, 2\}$, then all jobs J in this chain are scheduled at time $r(J) + k$ or later, where
 255 $r(J)$ is the release date of job J .*



■ **Figure 2** An instance of $1|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ representing a graph coloring. We have $G = (V, E)$ with $V = \{v_0, v_1, v_2\}$ and $E = (\{v_0, v_1\}, \{v_0, v_2\})$. This schedule corresponds to the coloring $(0, 2, 1)$.

256 **Proof.** For gadget chains $C'_{i,1}, C'_{i,2}$ this comes from the minimum delay $3(n+m) - 1$ between
 257 their two jobs. For vertex chain C'_i : suppose we have a feasible schedule where vertex chain
 258 C'_i starts at time $3i + k$ with $k \in \{0, 1, 2\}$.

- 259 ■ If vertex v_i is part of no edge in the graph:
 260 Then by Definition 7 vertex chain C_i only has two jobs and there is a minimum delay
 261 $3(n-i) - 1 + 3m + 3i = 3(n+m) - 1$ between them. Thus if the first job is scheduled at time
 262 $3i + k$, then the job in the closing segment is scheduled at time $(3i + k) + 1 + (3(n+m) - 1) =$
 263 $3(n+m+i) + k$, which is indeed the release date of this job plus k .
- 264 ■ If vertex v_i is part of at least one edge in the graph:
 265 Let $j_0 < j_1 < \dots < j_{deg(v_i)-1}$ be the indices of the edges e_j such that $v_i \in e_j$. We prove
 266 the lemma for all the jobs of vertex chain C_i in an edge check segment by induction on
 267 $l \in [0, deg(v_i) - 1]$ the same way as in Lemma 4. Then only the job of the chain in the
 268 closing segment is left. By Definition 7 there is a minimum delay $2 + 3(m-1 - j_{deg(v_i)-1}) + 3i$
 269 between this job and the one in edge check segment $[3(n + j_{deg(v_i)-1}), 3(n + j_{deg(v_i)-1} + 1))$.
 270 Since we know from the induction that the latter job is scheduled at time $3(n + j_{deg(v_i)-1}) +$
 271 k or later, this means that the job of vertex chain C_i in the closing segment is scheduled
 272 at time $(3(n + j_{deg(v_i)-1}) + k) + 1 + (2 + 3(m-1 - j_{deg(v_i)-1}) + 3i) = 3(n+m+i) + k$
 273 or later, which is indeed the release date of this job plus k .

274 ◀

275 By taking into account the constraints added by the gadget chains at the beginning and
 276 at the end of each vertex chain, we are able to prove the needed key property.

277 ▶ **Lemma 11.** *In any feasible schedule, if a vertex chain C_i starts at time $3i + k$ with*
 278 *$k \in \{0, 1, 2\}$, then all jobs J in vertex chain C_i which are in an edge check segment have to*
 279 *be scheduled at time $r(J) + k$, where $r(J)$ is the release date of job J .*

280 **Proof.** Consider a feasible schedule. Let $0 \leq i \leq n - 1$. Three jobs are scheduled in time
 281 window $[3i, 3(i + 1))$: the first job of vertex chain \mathcal{C}_i and the first job of the two gadget chains
 282 relative to it. Let $3i + k$ (resp. $3i + k'_1, 3i + k'_2$) be the starting time of the first job of vertex
 283 chain \mathcal{C}_i (resp. gadget chains $\mathcal{C}'_{i,1}, \mathcal{C}'_{i,2}$). We have k, k'_1 and k'_2 in $\{0, 1, 2\}$ and since we have
 284 a feasible schedule the three values are different from each other. Thus one chain starts at
 285 time $3i + 2$. By Lemma 10 and the time window $[3(n + m + i), 3(n + m + i + 1))$ of the last
 286 job, this means that this last job must be exactly scheduled at time $3(n + m + i) + 2$. Now
 287 consider the chain which starts at time $3i + 1$. By Lemma 10 its last job must be scheduled
 288 at time $3(n + m + i) + 1$ or $3(n + m + i) + 2$. However the later time position is already
 289 taken by the chain starting at time $3i + 2$, which means that this last job must be exactly
 290 scheduled at time $3(n + m + i) + 1$. Finally by the same reasoning we get that the chain
 291 starting at time $3i$ must have its last job scheduled at time $3(n + m + i)$.

292 This means that whatever the starting time $3i + k$ is for vertex chain \mathcal{C}_i , its last job must
 293 be scheduled at time $3(n + m + i) + k$. So all the delays in the chain must be equal to their
 294 minimum and thus by Lemma 10 all the jobs J in the vertex chain must be scheduled at
 295 time $r(J) + k$. ◀

296 Now in any feasible schedule we proved that we have the same guarantee on the position
 297 of the edge check jobs as we had with Lemma 4 in the exact delay case. Thus we are able to
 298 propagate the color choices accurately and complete the reduction the same way.

299 ▶ **Proposition 12.** *G is 3-colorable if and only if there exists a feasible schedule for this*
 300 *instance of MIN DELAYS.*

301 **Proof.** (\implies) Suppose we have $(c_0, \dots, c_{n-1}) \in \{0, 1, 2\}^n$ a 3-coloring of G where vertex v_i
 302 has color c_i . We propose a schedule where for all $0 \leq i \leq n - 1$, vertex chain \mathcal{C}'_i starts at time
 303 $3i + c_i$ and gadget chains $\mathcal{C}'_{i,1}, \mathcal{C}'_{i,2}$ start in the two remaining time positions $3i + k_1, 3i + k_2$
 304 in $[3i, 3(i + 1))$ (with $k_1 \neq k_2$). Plus we require all delays to match their minimum value.

305 Then, according to Definition 7 and going from left to right as we did in the proof of
 306 Lemma 10, we know that in every edge $e_j \in E$ where node v_i appears, the job of vertex chain
 307 \mathcal{C}'_i which is in edge check segment $[3(n + j), 3(n + j + 1))$ is scheduled at time $3(n + j) + c_i$,
 308 and the last job of \mathcal{C}'_i is scheduled at time $3(n + m + i) + c_i$. Plus from Definition 8 we
 309 know that the last job of gadget chain $\mathcal{C}'_{i,1}$ (resp. $\mathcal{C}'_{i,2}$) is scheduled $3(n + m + i) + k_1$ (resp.
 310 $3(n + m + i) + k_2$).

311 We show that the jobs in different chains do not interfere with each other. For the color
 312 choice segment we know that vertex chain \mathcal{C}'_i and gadget chains $\mathcal{C}'_{i,1}, \mathcal{C}'_{i,2}$ start respectively at
 313 times $3i + c_i, 3i + k_1$, and $3i + k_2$ with c_i, k_1 and k_2 in $\{0, 1, 2\}$ and different from each other.
 314 For the closing segment we determined that vertex chain \mathcal{C}'_i and gadget chains $\mathcal{C}'_{i,1}, \mathcal{C}'_{i,2}$ end
 315 respectively at times $3(n + m + i) + c_i, 3(n + m + i) + k_1$, and $3(n + m + i) + k_2$, again with c_i, k_1
 316 and k_2 in $\{0, 1, 2\}$ and different from each other. Thus only the edge check segments remain
 317 to be checked. Let $e_j = \{v_{i_1}, v_{i_2}\}$ be an edge in E . By definition of the vertex chains, only
 318 vertex chains \mathcal{C}'_{i_1} and \mathcal{C}'_{i_2} have a job to be scheduled in time window $[3(n + j), 3(n + j + 1))$.
 319 In our schedule the job of chain \mathcal{C}'_{i_1} is scheduled at time $3(n + j) + c_{i_1}$ and the job of chain
 320 \mathcal{C}'_{i_2} at time $3(n + j) + c_{i_2}$. Since (c_0, \dots, c_{n-1}) is a 3-coloring and $\{v_{i_1}, v_{i_2}\} \in E$, we have
 321 $c_{i_1} \neq c_{i_2}$. Thus both jobs are scheduled at different times and the jobs in edge check segment
 322 $[3(n + j), 3(n + j + 1))$ do not interfere with each other. Thus the proposed schedule is
 323 feasible.

324 (\impliedby) Suppose we have a feasible schedule. We reuse the same coloring as in the proof
 325 of Proposition 5: for all $0 \leq i \leq n - 1$, let $s_i \in \{0, 1, 2\}$ be such that $3i + s_i$ is the starting
 326 time of chain \mathcal{C}'_i . We show that (s_0, \dots, s_{n-1}) is a 3-coloring of G . Considering any edge

327 $e_j = \{v_{i_1}, v_{i_2}\} \in E$, Lemma 11 ensures that the job of vertex chain \mathcal{C}_{i_1} (resp. \mathcal{C}_{i_2}) in edge
 328 check segment $[3(n+j), 3(n+j+1))$ is scheduled at time $3(n+j) + s_{i_1}$ (resp. $3(n+j) + s_{i_2}$),
 329 with s_{i_1} (resp. s_{i_2}) the starting time of vertex chain \mathcal{C}_{i_1} (resp. \mathcal{C}_{i_2}). Since this is a feasible
 330 schedule we have: $3(n+j) + s_{i_1} \neq 3(n+j) + s_{i_2}$, which means: $s_{i_1} \neq s_{i_2}$. Thus the two
 331 nodes of edge e_j have indeed different colors. \blacktriangleleft

332 This proves that $1|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ with $\mu = 2$ is NP-hard, which concludes
 333 the para-NP-completeness proof of the corresponding parameterized problem.

334 **► Theorem 13.** $1|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ is para-NP-complete parameterized by path-
 335 width μ .

336 2.3 NP-hardness of $P|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ with $\ell_{max} = 1$

337 We build an instance of $P|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ with $\ell_{max} = 1$. We still have n vertex
 338 chains, one per node v_i in graph G , and we want to represent and check a coloring of G in
 339 a similar way to Section 2.1: choose the color of the nodes with the starting time of the
 340 vertex chains, then propagate these choices and check that two nodes of an edge do not have
 341 the same color. However with the change of parameter we must worry about the maximum
 342 delay value instead of the overlapping of time windows. Here we want to propagate the color
 343 choices while keeping the delays small.

344 We propose to add extra intermediate jobs that we call *propagators* at every other time
 345 position. This way the color choices can be propagated along the odd time positions with
 346 exact delays of length 1 while the even time positions are kept for edge checking. We set
 347 $M = n$ as the number of machines in order to make room for these propagators. We define
 348 vertex chain \mathcal{C}_i the following way:

349 **► Definition 14** (Vertex chain \mathcal{C}_i with $\ell_{max} = 1$). We define \mathcal{C}_i as a chain of $3(n+m-i) +$
 350 $deg(v_i)$ jobs. These jobs will fulfill two roles:

- 351 \blacksquare Propagators $O_{j,k}^i$: $O_{i,0}^i$ will give the color choice of node v_i . The other $3(n+m-i) - 1$
 352 jobs $O_{j,k}^i$ ($i \leq j \leq n+m-1$, $0 \leq k \leq 2$) will propagate this color choice along the
 353 whole chain while keeping the maximum delay value at 1. Job $O_{j,k}^i$ will have time window
 354 $[6j+2k, 6(j+1)+2k)$.
- 355 \blacksquare Edge jobs J_j^i : the $deg(v_i)$ jobs J_{n+j}^i will represent the color choice of node v_i in every
 356 edge e_j where node v_i is in ($0 \leq j \leq m-1$). Job J_{n+j}^i will have time window $[6(n+j), 6(n+j+1))$.
 357

358 We segment time into $m+1$ segments: a color choice segment $[0, 6n)$ and m edge check
 359 segments along $[6n, 6(n+m))$. We describe the chain from left to right:

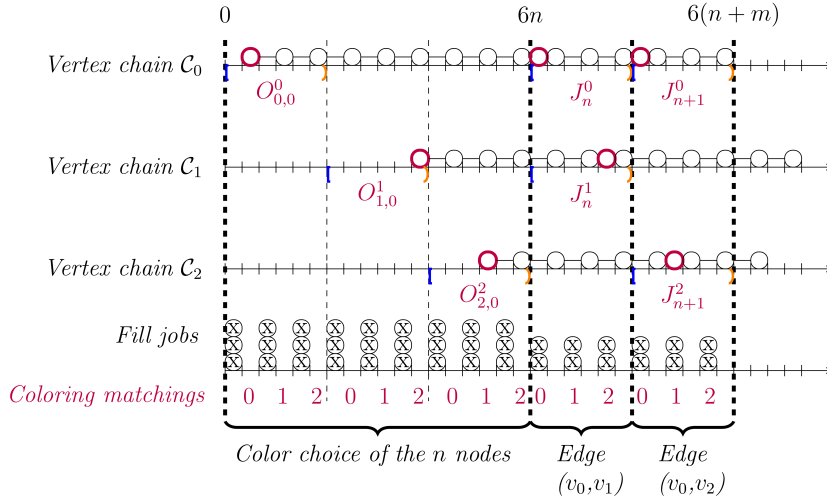
360 (1) Color choice segment $[0, 6n)$

- 361 \blacksquare Set the first job $O_{i,0}^i$ of \mathcal{C}_i in time window $[6i, 6(i+1))$.
- 362 \blacksquare Add a unit-time exact delay then a job, and do this $3(n-i) - 1$ times in a row. These
 363 jobs are named $O_{i,1}^i, O_{i,2}^i, O_{i+1,0}^i, \dots, O_{n,2}^i$.

364 (2) Edge check segment $[6(n+j), 6(n+j+1))$, $0 \leq j \leq m-1$

365 Let edge $e_j = \{v_{i_1}, v_{i_2}\}$, $i_1 < i_2$. This segment will check if the vertices v_{i_1} and v_{i_2} have
 366 different colors.

- 367 a. Vertex chain \mathcal{C}_i with $i \notin \{i_1, i_2\}$: add a unit-time exact delay then job $O_{n+j,0}^i$ then a
 368 unit-time exact delay then job $O_{n+j,1}^i$ then a unit-time exact delay then job $O_{n+j,2}^i$.
- 369 b. Vertex chain \mathcal{C}_i with $i = i_1$ or $i = i_2$: add job J_{n+j}^i then an exact delay of length zero
 370 then job $O_{n+j,0}^i$ then a unit-time exact delay then job $O_{n+j,1}^i$ then a unit-time exact
 371 delay then job $O_{n+j,2}^i$.



■ **Figure 3** An instance of $P|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ representing a graph coloring. We have $G = (V, E)$ with $V = \{v_0, v_1, v_2\}$ and $E = (\{v_0, v_1\}, \{v_0, v_2\})$. There are $M = n = 3$ machines and this schedule corresponds to the coloring $(0, 2, 1)$.

372 Note that time intervals of length 6 are used instead of length 3. This way three odd
 373 starting times are available for each vertex chain. However *fill jobs* must be added at every
 374 even time position of the color choice segment: M of them so that only these odd starting
 375 times are actually allowed. Plus now that we are in a parallel-machine framework instead of
 376 a single-machine one, more *fill jobs* are needed at the even time positions of the edge check
 377 segments: $M - 1$ of them so that one edge job at any of these positions is allowed but two
 378 would invalidate the schedule.

379 ► **Definition 15** (Fill jobs). *Fill jobs are chains of one job with a time window of length 1.*
 380 M fill jobs are set at every even time position in color choice segment $[0, 6n)$ and $M - 1$ fill
 381 jobs are set at every even time position in time segment $[6n, 6(n + m))$.

382 An example is given in Figure 3. With the addition of fill jobs the reduction can now
 383 be proved correct in a similar way to Section 2.1: determine the exact positions of all jobs
 384 in a vertex chain given its starting time, then show that a schedule is feasible if and only
 385 if whenever two vertex chains have an edge job in the same edge check segment they must
 386 start at different times.

387 ► **Proposition 16.** G is 3-colorable if and only if there exists a feasible schedule for this
 388 instance of $P|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$.

389 First we determine the positions of all jobs in a vertex chain given its starting time. In
 390 particular we confirm that in any feasible schedule propagators are always scheduled at odd
 391 time positions and edge jobs at even time positions:

392 ► **Lemma 17.** Let $0 \leq i \leq n - 1$. In any feasible schedule, if vertex chain \mathcal{C}_i starts at time
 393 $6i + 2l + 1$ with $l \in \{0, 1, 2\}$, then all the jobs $O_{j,k}^i$ (resp. J_j^i) in this chain are scheduled at
 394 time $r(O_{j,k}^i) + 2l + 1$ (resp. $r(J_j^i) + 2l$), where $r(J)$ is the release date of job J .

395 **Proof.** Suppose we have a feasible schedule where vertex chain \mathcal{C}_i starts at time $6i + 2l + 1$
 396 with $l \in \{0, 1, 2\}$.

- 397 ■ Propagators $O_{j,k}^i$: by Definition 14 there is always either a unit-time exact delay or a
 398 job J_j^i between two consecutive jobs $O_{j,k}^i$ in vertex chain \mathcal{C}_i . Thus if the first job $O_{i,0}^i$
 399 is scheduled at time $6i+2l+1 = r(O_{i,0}^i) + 2l+1$, then we know that the next propagator $O_{i,1}^i$
 400 is scheduled at time $(6i+2l+1)+2 = r(O_{i,1}^i) + 2l+1$, and so on. By induction on the couple
 401 (j, k) with $i \leq j \leq n+m-1$ and $0 \leq k \leq 2$, we get that all the jobs $O_{j,k}^i$ in vertex chain
 402 \mathcal{C}_i are scheduled at time $[6i+2l+1] + 2 \times (3(j-i) + k) = 6j+2k+2l+1 = r(O_{j,k}^i) + 2l+1$.
- 403 ■ Edge jobs J_j^i : let $j_0 < j_1 < \dots < j_{deg(v_i)-1}$ be the indices of the edges e_j such that
 404 $v_i \in e_j$ (if there are any). Let $p \in [0..deg(v_i) - 1]$. According to Definition 14 job $O_{j_p,0}^i$
 405 has the same time window $[6(n+j_p), 6(n+j_p+1))$ as job $J_{j_p}^i$ and it is scheduled right
 406 before it. Therefore according to our previous point about propagators $O_{j,k}^i$, job $J_{j_p}^i$ is
 407 scheduled at time $[r(O_{j_p,0}^i) + 2l+1] - 1 = r(J_{j_p}^i) + 2l$.
- 408 ◀

409 Now we are able to prove Proposition 16:

410 **Proof.** (\implies) Suppose we have $(c_0, \dots, c_{n-1}) \in \{0, 1, 2\}^n$ a 3-coloring of G where vertex v_i
 411 has color c_i . We propose the schedule σ where for all $0 \leq i \leq n-1$, chain \mathcal{C}_i starts at time
 412 $6i+2c_i+1$. Then by Definition 14 and abiding by exact delays, in every edge $e_j \in E$ where
 413 vertex v_i is in, job J_j^i is scheduled at time $r(J_j^i) + 2c_i = 6(n+j) + 2c_i$.

414 We show that there are never more than $M = n$ jobs scheduled at any time position.
 415 Since there is no fill job at an odd time position and two jobs of the same chain cannot
 416 be scheduled at the time, there are at most n jobs scheduled at every odd time. Thus
 417 only the even time positions remain to be checked. All the chains \mathcal{C}_i start at an odd time
 418 $6i+2c_i+1$, so by Definition 14 and abiding by exact delays every job $O_{j,k}^i$ is scheduled
 419 at time $r(O_{j,k}^i) + 2c_i+1$, which is odd since all the release dates are even according to
 420 Definition 14. Plus it means that only fill jobs are scheduled at the even time positions of
 421 color choice segment $[0, 6n)$. Thus only the even time positions of the edge check segments in
 422 $[6n, 6(n+m))$ remain to be checked. There are $M-1$ fill jobs scheduled at all of them. Let
 423 $j \in [0..m-1]$ and $e_j = \{v_{i_1}, v_{i_2}\}, i_1 < i_2$. In edge check segment $[6(n+j), 6(n+j+1))$ there
 424 are exactly two non-fill jobs to be scheduled: $J_{n+j}^{i_1}$ and $J_{n+j}^{i_2}$. As mentioned in the previous
 425 paragraph they are respectively scheduled at time $6(n+j) + 2c_{i_1}$ and $6(n+j) + 2c_{i_2}$. Since
 426 $(c_0, \dots, c_{n-1}) \in \{0, 1, 2\}^n$ is a 3-coloring and $\{v_{i_1}, v_{i_2}\} = e_j \in E$, we have $c_{i_1} \neq c_{i_2}$ and thus
 427 $\sigma(J_{n+j}^{i_1}) \neq \sigma(J_{n+j}^{i_2})$. Therefore at most M jobs are scheduled at any time position.

428 (\impliedby) Suppose we have a feasible schedule. For all $0 \leq i \leq n-1$, let $s_i \in \{0, 1, 2\}$ be
 429 such that $6i+2s_i+1$ is the starting time of chain \mathcal{C}_i (recall that it can only be an odd time
 430 because of the fill jobs defined in Definition 15). We show that (s_0, \dots, s_{n-1}) is a 3-coloring
 431 of G . By contradiction suppose there is an edge $e_j = \{v_{i_1}, v_{i_2}\} \in E$ such that $s_{i_1} = s_{i_2}$. Then
 432 according to Lemma 17 jobs $J_{n+j}^{i_1}$ and $J_{n+j}^{i_2}$ are scheduled at the same time $6(n+j) + 2s_{i_1}$.
 433 Thus taking into account the $M-1$ fill jobs at this time position, there are $M+1$ jobs
 434 scheduled at the same time position, which is greater than the number of machines M . Thus
 435 the schedule is not feasible, which leads to a contradiction. Thus (s_0, \dots, s_{n-1}) is indeed a
 436 3-coloring of G . ◀

437 This proves that $P|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ with $\ell_{max} = 1$ is NP-hard, which concludes
 438 the para-NP-completeness proof of the corresponding parameterized problem.

439 ► **Theorem 18.** $P|chains(\ell_{i,j}^{ex}), p_j = 1, r_j, d_j|*$ is para-NP-complete when parameterized by
 440 maximum delay ℓ_{max} .

441 **2.4 NP-hardness of $P|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ with $\ell_{max} = 1$**

442 We build an instance of $P|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ with $\ell_{max} = 1$. The general
 443 idea is to combine the two previously proposed extensions of the basic reduction from
 444 Section 2.1: gadget chains from Section 2.2 that dealt with the extra flexibility coming from
 445 using minimum delays instead of exact ones, and propagators from Section 2.3 that helped
 446 to keep the maximum delay value equal to one. However getting the final product proved to
 447 be significantly more technical than with the previous reductions, as propagators and gadget
 448 chains could interfere with each other. The reduction will not be detailed in this section: the
 449 full description and proof is available in Appendix A.1, as well as an example in Figure 7.
 450 Instead we give insight into the major roadblock that we faced and eventually managed to
 451 overcome: interference between propagators and gadget chains.

452 Again the goal was to prove that *in any feasible schedule the edge jobs accurately represent*
 453 *the color choice*. As we have minimum delays we thought about using gadget chains like
 454 the reduction in Section 2.2. This required to replicate a situation equivalent to the one
 455 displayed in Figure 2 while propagators from other chains were around and could potentially
 456 trade places. As we needed propagators at every other time position to keep the maximum
 457 delay value at 1, it was not possible to completely isolate each triplet of chains as we did in
 458 Section 2.2 when pathwidth μ was the parameter. So when a triplet of chains was considered
 459 by the lemma it had to be guaranteed that the propagators from other chains were fixed and
 460 thus could not trade places.

461 This was made possible by setting the closing time windows of the chain triplets in the
 462 reverse order of the color choice time windows. Then, as shown in Figure 7, chains $C'_0, C'_{0,1}$,
 463 and $C'_{0,2}$ start at the first part of the color choice segment and end at the last part of the
 464 closing segment. Then chains $C'_{1,1}, C'_{1,2}$ start at the second part of the color choice
 465 segment and end at the second to last part of the closing segment, and so on. This way only
 466 propagators of chains $C'_j, C'_{j,1}, C'_{j,2}$ with $j < i$ might interfere. These jobs would be fixed by
 467 induction hypothesis and as such could not trade places.

468 Now that such a property was proved, the correspondence between valid graph 3-colorings
 469 and feasible schedules could be established the same way as in our previous reductions.

470 ► **Proposition 19.** *G is 3-colorable if and only if there exists a feasible schedule for this*
 471 *instance of $P|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$.*

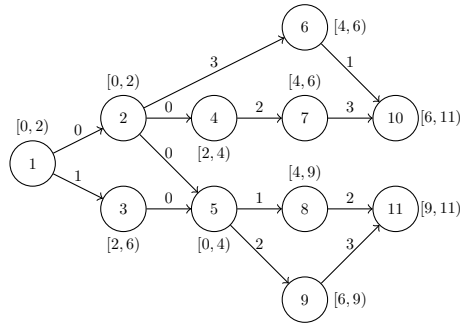
472 This proves that $P|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ with $\ell_{max} = 1$ is NP-hard, which
 473 concludes the para-NP-completeness proof of the corresponding parameterized problem.

474 ► **Theorem 20.** *$P|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ is para-NP-complete when parameterized by*
 475 *maximum delay ℓ_{max} .*

476 **3 A FPT algorithm with two parameters**

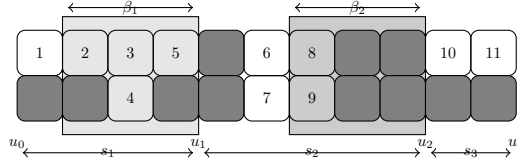
477 In this section we prove that the problem with precedence delays (exact or minimum)
 478 $P|prec(\ell_{i,j}), p_i = 1, r_i, d_i|*$ is fixed-parameter tractable with the couple of parameters
 479 (ℓ_{max}, μ) . For the sake of readability we detail the minimum delays case.

480 Let us consider the sorted list $x_k, k \in \{0, \dots, K\}$ of the release times and deadlines in non
 481 decreasing order. We define a sub-sequence $u_\alpha, \alpha \in \{0, \dots, \kappa - 1\}$ of this sequence so that two
 482 consecutive terms - except the last one - are separated by at least ℓ_{max} . So we set $u_0 = x_0$,
 483 then $u_{\alpha+1} = x_k$ with k the minimum value in $\{1, \dots, K\}$ such that $u_{\alpha+1} - u_\alpha \geq \ell_{max}$. Lastly
 484 we set $u_\kappa = x_K$.



■ **Figure 4** A precedence graph with minimum delays. Each precedence arc $e = (i, j)$ is labeled by the minimum delay ℓ_{ij} . Each node i is labelled by its time window $[r_i, d_i)$.

485 Let us consider the instance with minimum delays described in Figure 4.
 486 For this instance we get the sequence $x_0 = 0, x_1 = 2, x_2 = 4, x_3 = 6, x_5 = 9$ and $x_6 = 11$
 487 with $K = 6$. The associated pathwidth $\mu = 3$ is reached in the interval $[4, 6)$ crossed by
 488 intervals of jobs 3, 6, 7, 8. We also have $\ell_{max} = 3$, so we get: $u_0 = x_0 = 0, u_1 = x_2 = 4,$
 489 $u_2 = x_5 = 9$ and $u_3 = x_6 = 11$.
 490 We set $X_0 = \emptyset$ and for any $\alpha \in \{1, \dots, \kappa\}$ we define $X_\alpha = \{i \in \mathcal{T}, [r_i, d_i) \cap [u_{\alpha-1}, u_\alpha) \neq \emptyset\}$
 491 the set of jobs that could be scheduled in interval $[u_{\alpha-1}, u_\alpha)$. The idea of sequence (u_0, \dots, u_κ)
 492 is that the number of jobs in each X_α is bounded by $(\mu + 1) \times \ell_{max}$ (see Lemma 22). We
 493 also define $Z_\alpha, \alpha \in \{0, \dots, \kappa\}$ the set of jobs with a deadline not greater than u_α , *i.e.* $Z_\alpha =$
 494 $\{i \in \mathcal{T}, d_i \leq u_\alpha\}$. In our example we have: $Z_0 = \emptyset, Z_1 = \{1, 2, 4, 5\}, Z_2 = Z_1 \cup \{3, 6, 7, 8, 9\}$
 495 and $Z_3 = \mathcal{T}$.
 496 We define a dynamic programming scheme for our problem. The stages of the scheme are
 497 $\{0, \dots, \kappa\}$. For each stage $\alpha \in \{0, \dots, \kappa\}$ we denote N_α the set of states of stage α . A state
 498 $s \in N_\alpha$ represents the minimum information from a feasible schedule spanning in $[0, u_\alpha)$ that
 499 is necessary to extend this schedule in interval $[u_\alpha, u_\kappa)$.
 500 Hence a state $s \in N_\alpha$ with $\alpha \in \{1, \dots, \kappa - 1\}$ is a tuple $s = (\beta, Y)$, where:
 501 ■ $Y \subseteq X_\alpha - Z_\alpha$ is a subset of jobs such that $Y \cup Z_\alpha$ represents the set of jobs scheduled in
 502 $[0, u_\alpha)$.
 503 ■ β is a complete schedule (*i.e.* a set of jobs with their starting time) of the last ℓ_{max} time
 504 units before time u_α - *i.e.* in interval $[u_\alpha - \ell_{max}, u_\alpha)$. We denote $J(\beta)$ the set of jobs
 505 scheduled in β . β is called a *border schedule*. Only such a schedule can influence the
 506 earliest starting times of the jobs not scheduled yet.
 507 For $\alpha = \kappa$ we set $N_\kappa = \{s_\kappa\}$ with $s_\kappa = (\bullet, \emptyset)$ where \bullet is an empty schedule. Moreover
 508 $X_\kappa - Z_\kappa = \emptyset$ and so N_κ can be reduced to only one element. Similarly, we set $N_0 = \{s_0\}$
 509 with $s_0 = (\bullet, \emptyset)$ since $X_0 = \emptyset$ and no job may be executed in interval $[u_0, u_0) = \emptyset$.
 510 As an example let us consider a feasible schedule σ pictured by Figure 5 for $m = 2$
 511 identical machines associated to the instance given by Figure 4. The associates states are:
 512 $s_0 = (\bullet, \emptyset), s_1 = (\beta_1, \{3\}), s_2 = (\beta_2, \emptyset)$ and $s_3 = (\bullet, \emptyset)$.
 513 Now assume that $s = (\beta, Y) \in N_\alpha$ with $\alpha \in \{0, \dots, \kappa\}$. The boolean function
 514 ExistSched(s) is set to *true* if and only if there exists a (partial) feasible schedule of jobs
 515 from $Y \cup Z_\alpha$ in time interval $[u_0, u_\alpha)$ that ends with schedule β .
 516 We can now establish the recurrence equation for this function:
 517 1. ExistSched(s_0) = *true*; indeed, $s_0 = (\bullet, \emptyset)$ and $Z_0 = \emptyset$, thus no job has to be scheduled.
 518 2. Let us now consider $\alpha \in \{1, \dots, \kappa\}$. If ExistSched(s) = *true* then there exists a feasible
 519 schedule in $[0, u_\alpha)$ which can be decomposed into a feasible schedule in $[0, u_{\alpha-1})$ associated



■ **Figure 5** A feasible schedule σ associated with the example given in Figure 4 for $m = 2$ machines

520 with a state $s' \in N_{\alpha-1}$ and a schedule in the interval $[u_{\alpha-1}, u_\alpha)$ consistent with s and s' .

521 The existence of such a schedule is denoted by the function $\text{Sched}(s, s')$.

522 We now bound the complexity of computing $\text{Sched}(s, s')$ from a tuple of states $(s', s) \in$
 523 $N_{\alpha-1} \times N_\alpha$.

524 Let $s = (\beta, Y)$ and $s' = (\beta', Y')$. Then boolean $\text{Sched}(s', s)$ is true if and only if there
 525 exists a schedule of $Y \cup Z_\alpha - Y' - Z_{\alpha-1}$ in the interval $[u_{\alpha-1}, u_\alpha)$ that is consistent with the
 526 border schedule β' and ends with the border schedule β .

527 ► **Lemma 21.** For any $\alpha \in \{1, \dots, \kappa\}$ and $(s', s) \in N_{\alpha-1} \times N_\alpha$, the time complexity of
 528 $\text{Sched}(s', s)$ is $\mathcal{O}(\mu^2 \times \ell_{max}^2 \times ((\mu + 1) \times \ell_{max})!)$.

529 To prove this lemma, two more technical lemmas are needed. These lemmas bound the
 530 total number of schedulable jobs in time interval $[u_{\alpha-1}, u_\alpha)$ for $\alpha \in \{1, \dots, \kappa\}$:

531 ► **Lemma 22.** $\forall \alpha \in \{0, \dots, \kappa\}, |X_\alpha| \leq (\mu + 1) \times \ell_{max}$.

532 **Proof.** We simply observe that if $u_{\alpha-1} = x_k$ and $u_\alpha = x_{k'}$ then by construction $k' - k \leq \ell_{max}$.
 533 Thus the inequality holds by the definition of μ . ◀

534 ► **Lemma 23.** For any $\alpha \in \{1, \dots, \kappa - 1\}$, $|N_\alpha| \leq 2^{(\mu+1) \times \ell_{max}} \times (\ell_{max} + 1)^{(\mu+1) \times \ell_{max}}$.

535 **Proof.** The total number of schedules from a set $V = J(\beta)$ is bounded by $(\ell_{max} + 1)^{|V|}$.
 536 Thus, by Lemma 22, it is bounded by $(\ell_{max} + 1)^{(\mu+1) \times \ell_{max}}$. And because the number of sets
 537 $V \subseteq X_\alpha$ is bounded by $2^{|X_\alpha|} \leq 2^{(\mu+1) \times \ell_{max}}$, the lemma holds. ◀

538 Now we are able to prove Lemma 21:

539 **Proof.** The problem is to schedule jobs from $S = Y \cup Z_\alpha - (Y' \cup Z_{\alpha-1})$ in the interval
 540 $[u_{\alpha-1}, u_\alpha)$ so that the schedule is consistent with the two border schedules β' and β . This
 541 can be done in several steps:

- 542 1. adjusting the release times of jobs of S with respect to the border schedule β' : if j is a
 543 successor of $i \in J(\beta')$ then $r_j = \max(r_j, \beta'(i) + 1 + \ell_{i,j})$, and propagate to precedence
 544 constraints in S .
- 545 2. adjusting the deadlines of jobs of S with respect to the border schedule β : if i is a
 546 predecessor of $j \in J(\beta)$ then $d_i = \min(d_i, \beta(j) - \ell_{i,j})$, and propagate to precedence
 547 constraints in S .
- 548 3. if a contradiction is detected at this step (a job j for which $r_j \geq d_j$), $\text{Sched}(s', s) = \text{false}$.
- 549 4. Otherwise we can enumerate all active schedules (i.e. schedules in which no job can be
 550 scheduled earlier provided the other jobs are not delayed) of $S - J(\beta)$ and verify that
 551 one of them spans in $[u_{\alpha-1}, u_\alpha - \ell_{max})$.

552 The time complexity of the two first steps is $\mathcal{O}(|S|^2)$. For the last step it is known that
 553 any active schedule can be generated by list scheduling using a permutation of jobs [18]. Thus
 554 the enumeration of active schedules can be done by a brute force algorithm that enumerates

555 all permutations of jobs and then performs a list scheduling algorithm to check whether the
556 schedule spans in the interval $[u_{\alpha-1}, u_{\alpha} - \ell_{\max})$.

557 At most m jobs are executed at each instant, and the number of iterations is bounded
558 by $|S|$. For each iteration, we must check that all the precedence constraints (with exact or
559 minimum delays) are fulfilled, and thus one execution of this priority list has a complexity
560 bounded by $\mathcal{O}(|S|^2)$.

561 The total number of permutations is $|S - J(\beta)|!$. Thus the overall complexity is bounded
562 by $\mathcal{O}(|S|^2 \times |S - J(\beta)|!)$. And since $S \subseteq X_{\alpha}$, by Lemma 22 we get that $|S| \leq (\mu + 1) \times \ell_{\max}$
563 and the lemma holds. ◀

564 Finally we formalize the recurrence equation that yields a FPT algorithm when we have
565 minimum delays: if $s \in N_{\alpha}$,

$$566 \text{ExistSched}(s) = \bigvee_{s' \in N_{\alpha-1}} \text{Sched}(s', s) \wedge \text{ExistSched}(s') \quad (1)$$

567

568 ▶ **Theorem 24.** *The answer to an instance \mathcal{I} of $P|prec(\ell_{ij}), p_j = 1, r_j, d_j|*$ (with minimum or*
569 *exact delays) is "yes" if and only if $\text{ExistSched}(s_{\kappa})$ is true. Moreover the time complexity of the*
570 *computation of $\text{ExistSched}(s_{\kappa})$ is $\mathcal{O}(n \times (2\ell_{\max} + 2)^{2(\mu+1) \times \ell_{\max}} \times \mu^2 \times \ell_{\max}^2 \times ((\mu+1) \times \ell_{\max})!)$.*

571 **Proof.** If $\text{ExistSched}(s_{\kappa}) = \text{true}$, then a sequence of states $s_0, s_1, \dots, s_{\kappa}$ with $s_{\alpha} \in N_{\alpha}$ for
572 $\alpha \in \{0, \dots, \kappa\}$ and $\text{Sched}(s_{\alpha-1}, s_{\alpha}) = \text{true}$ for all $\alpha \in \{1, \dots, \kappa\}$ can be built. And conversely
573 such a sequence induces a feasible schedule.

574 Now, the number of calls of the function Sched necessary to compute the recurrence
575 equation (1) is proportional to $\sum_{\alpha=1}^{\kappa} |N_{\alpha-1}| \times |N_{\alpha}|$. By Lemma 23, this value is bounded by
576 $\kappa \times 2^{2(\mu+1) \times \ell_{\max}} \times (\ell_{\max} + 1)^{2(\mu+1) \times \ell_{\max}}$. Since $\kappa \leq 2n$, by Lemma 21 we get the theorem.

577 Notice that for exact delays, the computation of $\text{Sched}(s_{\alpha-1}, s_{\alpha})$ is slightly different since
578 the border schedule of $s_{\alpha-1}$ induces the schedule of all successors of these jobs. Similarly
579 the border of s_{α} induces starting times for predecessors of these jobs, so the first step is to
580 verify the consistency of the job starting times induced by the two border schedules. This
581 can be done in $\mathcal{O}((\mu \times \ell_{\max})^2)$. The remaining enumeration concerns the schedule of jobs
582 without predecessors that start after $u_{\alpha-1}$. In the worst case the number of these jobs is still
583 $\mathcal{O}(\mu \times \ell_{\max})$ so that the complexity is the same as in the min delays case. ◀

584 4 Conclusion

585 In this paper we analyzed the parameterized complexity of two scheduling problems with
586 precedence delays, unit processing times and job time windows with respect to two parameters:
587 the pathwidth μ and the maximum precedence delay ℓ_{\max} . To the best of our knowledge this
588 is the first hardness result with pathwidth μ as a parameter and unit processing times, and
589 also the first time that ℓ_{\max} is considered as a parameter. Our work raises an open problem
590 with parameter ℓ_{\max} , namely the single-machine problem $1|chains(\ell_{i,j}), p_j = 1, r_i, d_i|*$ for
591 which the hardness reductions we developed do not apply. Further work is also underway to
592 extend our FPT algorithm to more general problems, for instance with any processing times
593 jobs or more complex resource constraints.

594 References

- 595 1 Robert Baart, Mathijs de Weerd, and Lei He. Single-machine scheduling with release times,
596 deadlines, setup times, and rejection. *European Journal of Operational Research*, 291(2):629–
597 639, 2021. Number: 2 Publisher: Elsevier.

- 598 2 S. Bessy and R. Giroudeau. Parameterized complexity of a coupled-task scheduling problem.
599 *Journal of Scheduling*, 22(3):305–313, June 2019. doi:10.1007/s10951-018-0581-1.
- 600 3 Hans L Bodlaender and Michael R Fellows. W[2]-hardness of precedence constrained k-
601 processor scheduling. *Operations Research Letters*, 18(2):93–97, 1995. Number: 2 Publisher:
602 Elsevier.
- 603 4 Hans L Bodlaender and Marieke van der Wegen. Parameterized complexity of scheduling
604 chains of jobs with delays. In *15th International Symposium on Parameterized and Exact
605 Computation (IPEC)*, 2020.
- 606 5 P. Brucker, M.R. Garey, and D.S. Johnson. Scheduling equal-length tasks under treelike
607 precedence constraints to minimize maximum lateness. *Math. Oper. Res.*, 2(3):275–284, 1977.
- 608 6 R. Downey and M. Fellows. *Parameterized complexity*. Springer, 1999.
- 609 7 Rodney G Downey, Michael R Fellows, and Kenneth W Regan. Descriptive complexity and
610 the W hierarchy. In *Proof Complexity and Feasible Arithmetics*, pages 119–134, 1996.
- 611 8 Jianzhong Du, Joseph YT Leung, and Gilbert H Young. Scheduling chain-structured tasks to
612 minimize makespan and mean flow time. *Information and Computation*, 92(2):219–236, 1991.
613 Number: 2 Publisher: Elsevier.
- 614 9 J. Flum and M. Grohe. *Parameterized complexity theory*. Springer, 1998.
- 615 10 M.R. Garey, D.S. Johnson, R.E. Tarjan, and M. Yannakakis. Scheduling opposing forests.
616 *SIAM Journal on Algebraic Discrete Methods*, 4(1):72–93, 1983.
- 617 11 Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan.
618 Optimization and approximation in deterministic sequencing and scheduling: a survey. In
619 *Annals of discrete mathematics*, volume 5, pages 287–326. Elsevier, 1979.
- 620 12 Claire Hanen and Alix Munier-Kordon. Fixed-Parameter tractability of scheduling dependent
621 typed tasks subject to release times and deadlines. *Submitted*, 2021.
- 622 13 Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer
623 computations*, pages 85–103. Springer, 1972.
- 624 14 J.K. Lenstra and A.H.G. Rinnooy Kan. Complexity of scheduling under precedence constraints.
625 *Oper. Res.*, 26(1):22–35, 1978.
- 626 15 Matthias Mnich and René van Bevern. Parameterized complexity of machine schedul-
627 ing: 15 open problems. *Computers & Operations Research*, 100:254–261, December
628 2018. URL: <https://www.sciencedirect.com/science/article/pii/S0305054818302041>,
629 doi:10.1016/j.cor.2018.07.020.
- 630 16 Alix Munier Kordon. A fixed-parameter algorithm for scheduling unit dependent tasks on
631 parallel machines with time windows. *Discrete Applied Mathematics*, 290:1–6, 2021.
- 632 17 Alex J Orman and Chris N Potts. On the complexity of coupled-task scheduling. *Discrete
633 Applied Mathematics*, 72(1-2):141–154, 1997.
- 634 18 Linus Schrage. Solving resource-constrained network problems by implicit enumeration—nonpreemptive case. *Operations Research*, 18(2):263–278, 1970.
- 635 19 J. D. Ullman. NP-complete scheduling problems. *Journal of Computer and System sciences*,
636 1975. URL: <https://core.ac.uk/reader/82723490>.
- 637 20 René van Bevern, Robert Brederick, Laurent Bulteau, Christian Komusiewicz, Nimrod Talmon,
638 and Gerhard J. Woeginger. Precedence-constrained scheduling problems parameterized by
639 partial order width. In Yury Kochetov, Michael Khachay, Vladimir Beresnev, Evgeni Nurminski,
640 and Panos Pardalos, editors, *Discrete Optimization and Operations Research*, pages 105–120,
641 Cham, 2016. Springer International Publishing.
- 642 21 René van Bevern, Andrey Melnikov, Pavel V. Smirnov, and Oxana Yu. Tsidulko. On data
643 reduction for dynamic vector bin packing. *CoRR*, abs/2205.08769, 2022. arXiv:2205.08769,
644 doi:10.48550/arXiv.2205.08769.
- 645 22 Wenci Yu, Han Hoogeveen, and Jan Karel Lenstra. Minimizing Makespan in a Two-Machine
646 Flow Shop with Delays and Unit-Time Operations is NP-Hard. *Journal of Scheduling*, 7(5):333–
647 348, September 2004. doi:10.1023/B:JOSH.0000036858.59787.c2.

A Appendix

A.1 Description of the reduction in Section 2.4 and proof of Proposition 19

► **Definition 25** (Vertex chain \mathcal{C}'_i with $\ell_{max} = 1$). We define \mathcal{C}'_i as a chain of $3(2n + m - 2i - 1) + \deg(v_i) + 1$ jobs. These jobs will fulfill two roles:

- Propagators $O_{j,k}^i$: job $O_{i,0}^i$ will give the color choice of node v_i . The other $3(2n + m - 2i)$ jobs $O_{j,k}^i$ ($i \leq j \leq 2n + m - i - 1$, $0 \leq k \leq 2$) and $O_{2n+m-i-1,0}^i$ will propagate this color choice along the whole chain while keeping the maximum delay value at 1. Job $O_{j,k}^i$ will have time window $[6j + 2k, 6j + 2k + 3)$.
- Edge jobs J_j^i : The $\deg(v_i)$ jobs J_{n+j}^i will represent the color choice of node v_i in every edge e_j where node v_i is in ($0 \leq j \leq m - 1$). Job J_{n+j}^i will have time window $[6(n + j) + 1, 6(n + j) + 4)$.

In order to define vertex chain \mathcal{C}'_i we segment time into $m + 2$ segments: a color choice segment $[0, 6n)$, m edge check segments along $[6n, 6(n + m))$ and a closing segment $[6(n + m), 6(n + 2m))$. We describe the chain from left to right:

(1) Color choice segment $[0, 6n)$

- Set the first job $O_{i,0}^i$ of \mathcal{C}'_i in time window $[6i, 6(i + 1))$. Then add a unit-time minimum delay.
- Add a job then a unit-time minimum delay, and do this $3(n - i) - 1$ times in a row. These jobs are named $O_{i,1}^i, O_{i,2}^i, O_{i+1,0}^i, \dots, O_{n,2}^i$.

(2) Edge check segment $[6(n + j), 6(n + j + 1))$, $0 \leq j \leq m - 1$

Let edge $e_j = \{v_{i_1}, v_{i_2}\}$, $i_1 < i_2$. This segment will check if the vertices v_{i_1} and v_{i_2} have different colors.

a. Vertex chain \mathcal{C}'_i with $i \notin \{i_1, i_2\}$

- Add job $O_{n+j,0}^i$ then a unit-time minimum delay then job $O_{n+j,1}^i$ then a unit-time minimum delay then job $O_{n+j,2}^i$ then a unit-time minimum delay.

b. Vertex chain \mathcal{C}'_i with $i = i_1$ or $i = i_2$

- Add job $O_{n+j,0}^i$ then a minimum delay of length zero then job J_{n+j}^i then a minimum delay of length zero then job $O_{n+j,1}^i$ then a unit-time minimum delay then job $O_{n+j,2}^i$ then a unit-time minimum delay.

(3) Closing segment $[6(n + m), 6(n + 2m))$

- Add a job then a unit-time minimum delay, and do this $3(n - i - 1)$ times in a row. These jobs are named $O_{n+m,0}^i, O_{n+m,1}^i, O_{n+m,2}^i, O_{n+m+1,0}^i, \dots, O_{2n+m-(i+2),2}^i$.
- Add the last job $O_{2n+m-(i+1),0}^i$ of \mathcal{C}'_i with time window $[6(2n + m - (i + 1)), 6(2n + m - (i + 1)) + 3)$.

► **Definition 26** (Gadget chain $\mathcal{C}'_{i,1}$ (resp. $\mathcal{C}'_{i,2}$)).

- Set the first job $O_{i,0}^{i,1}$ (resp. $O_{i,0}^{i,2}$) in time window $[6i, 6i + 3)$.
- Add a unit-time minimum delay then a job, and do this $3(2n + m - 2i - 1) - 1$ times in a row. These jobs are named $O_{i,1}^{i,1}, O_{i,2}^{i,1}, O_{i+1,0}^{i,1}, \dots, O_{2n+m-(i+1),0}^{i,1}$ (resp. $O_{i,1}^{i,2}, O_{i,2}^{i,2}, O_{i+1,0}^{i,2}, \dots, O_{2n+m-(i+1),0}^{i,2}$).

► **Definition 27** (Fill jobs).

(1) Color choice segment $[0, 6n)$

Let $i \in [0, n - 1]$. In time segment $[6i, 6(i + 1))$:

- At time $6i$: set $M - 1 - 2i$ fill jobs.

693 ■ At time $6i + 1$: set $M - 1 - i$ fill jobs.

694 ■ At time $6i + 2$: set $M - 2 - 2i$ fill jobs.

695 (2) Edge check segments $[6n, 6(n + m))$

696 Let $j \in [0, m - 1]$. In time segment $[6(n + j), 6(n + j + 1))$:

697 ■ At time $6(n + j) + 1$: set $M - n - 1$ fill jobs.

698 ■ At time $6(n + j) + 3$: set $M - n - 1$ fill jobs.

699 (3) Closing segment $[6(n + m), 6(2n + m))$

700 Let $i \in [0, n - 1]$. In time segment $[6(n + 2m - (i + 1)), 6(n + 2m - i))$:

701 ■ At time $6(n + 2m - (i + 1))$: set $M - 2 - 2i$ fill jobs.

702 ■ At time $6(n + 2m - (i + 1)) + 1$: set $M - 1 - i$ fill jobs.

703 ■ At time $6(n + 2m - (i + 1)) + 2$: set $M - 1 - 2i$ fill jobs.

704 ► **Lemma 28.** Let $0 \leq i \leq n - 1$. In any feasible schedule, if a chain starts at time $6i + k$
 705 with $k \in \{0, 1, 2\}$, then all jobs J in this chain are scheduled at time $r(J) + k$ or later, where
 706 $r(J)$ is the release date of job J .

707 **Proof.** First the result is proved for vertex chains \mathcal{C}'_i . Suppose we have a feasible schedule
 708 where vertex chain \mathcal{C}'_i starts at time $6i + l$ with $l \in \{0, 1, 2\}$.

709 ■ Propagators $O^i_{j,k}$: by Definition 25 there is always either a unit-time minimum delay or a
 710 job J^i_j between two consecutive jobs $O^i_{j,k}$ in vertex chain \mathcal{C}_i . Thus if the first job $O^i_{i,0}$ is
 711 scheduled at time $6i + l = r(O^i_{i,0}) + l$ (or later), then we know that the next propagator
 712 $O^i_{i,1}$ is scheduled at time $(6i + l) + 2 = r(O^i_{i,1}) + l$ or later, and so on. By induction on the
 713 couple (j, k) with $i \leq j \leq n + m - 1$ and $0 \leq k \leq 2$, we get that all the jobs $O^i_{j,k}$ in vertex
 714 chain \mathcal{C}_i are scheduled at time $(6i + l) + 2 \times (3(j - i) + k) = 6j + 2k + l = r(O^i_{j,k}) + l$ or
 715 later.

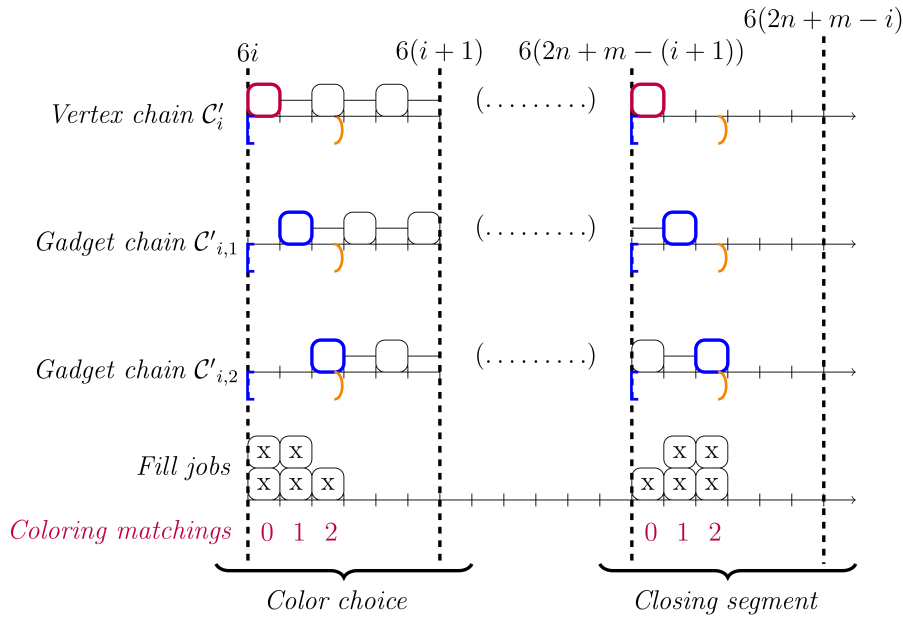
716 ■ Edge jobs J^i_j : let $j_0 < j_1 < \dots < j_{deg(v_i)-1}$ be the indices of the edges e_j such that
 717 $v_i \in e_j$ (if there are any). Let $p \in [0, deg(v_i) - 1]$. According to Definition 25 job $J^i_{j_p}$
 718 is scheduled right before job $O^i_{j_p,0}$ with a minimum delay of length zero between them.
 719 Therefore according to our previous point about jobs $O^i_{j,k}$, job $J^i_{j_p}$ is scheduled at time
 720 $(r(O^i_{j_p,0}) + l) + 1 = r(J^i_{j_p}) + l$ or later.

721 Gadget chain $\mathcal{C}'_{i,1}$ (resp. $\mathcal{C}'_{i,2}$) only features propagators. By Definition 26 there is always
 722 a unit-time minimum delay between two consecutive jobs $O^{i,1}_{j,k}$ (resp. $O^{i,2}_{j,k}$), so the result can
 723 be proven the same way as in the first item of the proof for vertex chains. ◀

724 ► **Lemma 29.** Let $0 \leq i \leq n - 1$. In any feasible schedule, if a chain starts at time $6i + k$
 725 with $k \in \{0, 1, 2\}$, then all jobs J in this chain have to be scheduled at time $r(J) + k$, where
 726 $r(J)$ is the release date of job J .

727 **Proof.** We prove by induction on $i \in [0, n - 1]$ that for all $0 \leq j \leq i$ exactly one chain starts
 728 at each time $6j, 6j + 1, 6j + 2$ and all jobs J in a chain $\mathcal{C}'_j, \mathcal{C}'_{j,1}, \mathcal{C}'_{j,2}$ that starts at time
 729 $6j + k$ have to be scheduled at time $r(J) + k$.

730 ■ According to Definition 27 on time windows $[0, 6)$ and $[6(2n + m - 1), 6(2n + m))$, the
 731 chain triplet $\mathcal{C}'_0, \mathcal{C}'_{0,1}, \mathcal{C}'_{0,2}$ is in the situation described in Figure 6. Thus at least one
 732 chain must start at time 2 which means by Lemma 28 that this chain has to end at
 733 time $6(2n + m - 1) + 2$. Then time $6(2n + m - 1) + 2$ is blocked, so by the same lemma
 734 another chain cannot start at time 2. Thus the two other chains have to start at the two
 735 remaining time positions 0 and 1, one per chain. By Lemma 28 the chain that starts at
 736 time 1 ends at time $6(2n + m - 1) + 1$ (or later but the only other time position possible
 737 $6(2n + m - 1) + 2$ is already blocked). So time position $6(2n + m - 1) + 1$ is now blocked,
 738 which forces the chain that starts at time 0 to end at time $6(2n + m - 1)$.



■ **Figure 6** A toy situation with three machines and the three chains related to a node in the $P|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ reduction. In any feasible schedule featuring these chains, for $k \in \{0, 1, 2\}$ exactly one chain starts at time $6i + k$, and then this chain has to end at time $6(n + 2m - (i + 1)) + k$.

739 ■ Let $i \in [0, n - 1)$. Assume the induction hypothesis to be true for chain triplets of index
 740 j with $0 \leq j \leq i - 1$. By Definition 25 we know that only these chain triplets have
 741 propagators that might interfere in time windows $[6i, 6(i + 1))$ and $[6(2n + m - (i + 1)), 6(2n + m - i))$. By induction hypothesis we know that these propagators are fixed,
 742 and we deduce that the number of fixed jobs (propagators from other chains plus fill jobs)
 743 at the relevant time positions is the following:
 744

- 745 (1) Color choice segment, part $[6i, 6(i + 1))$:
- 746 ■ At time $6i$: $M - 1 - 2i$ fill jobs and $2i$ propagators from other chains which add up
 747 to $M - 1$ jobs.
 - 748 ■ At time $6i + 1$: set $M - 1 - i$ fill jobs and i propagators from other chains which
 749 add up to $M - 1$ jobs.
 - 750 ■ At time $6i + 2$: set $M - 2 - 2i$ fill jobs and $2i$ propagators from other chains which
 751 add up to $M - 2$ jobs.
- 752 (2) Closing segment, part $[6(n + 2m - (i + 1)), 6(n + 2m - i))$:
- 753 ■ At time $6(n + 2m - (i + 1))$: set $M - 2 - 2i$ fill jobs and $2i$ propagators from other
 754 chains which add up to $M - 2$ jobs.
 - 755 ■ At time $6(n + 2m - (i + 1)) + 1$: set $M - 1 - i$ fill jobs and i propagators from other
 756 chains which add up to $M - 1$ jobs.
 - 757 ■ At time $6(n + 2m - (i + 1)) + 2$: set $M - 1 - 2i$ fill jobs and $2i$ propagators from
 758 other chains which add up to $M - 1$ jobs.

759 Thus we are again in the situation described in Figure 6 and we can prove the result for
 760 the triplet $C'_i, C'_{i,1}, C'_{i,2}$ the same way as in the initialization.

761 This concludes the proof of the lemma for all chains. ◀

762 Now we are able to prove Proposition 19:

763 **Proof.** (\implies) Suppose we have $(c_0, \dots, c_{n-1}) \in \{0, 1, 2\}^n$ a 3-coloring of G where vertex v_i
 764 has color c_i . We propose a schedule σ where for all $0 \leq i \leq n-1$, chain \mathcal{C}'_i starts at time
 765 $6i + c_i$ and gadget chains $\mathcal{C}'_{i,1}, \mathcal{C}'_{i,2}$ start in the two remaining time positions $6i + l_1, 6i + l_2$ in
 766 $[3i, 3(i+1))$ (with $l_1 \neq l_2$). Plus we require all delays to match their minimum value.

767 Then, according to Definition 25, Definition 26 and going from left to right as we did
 768 in the proof of Lemma 28, we know that propagators $O_{j,k}^i, O_{j,k}^{i,1}$ and $O_{j,k}^{i,2}$ are respectively
 769 scheduled at times $6j + 2k + c_i, 6j + 2k + l_1$ and $6j + 2k + l_2$. In the same way we know that
 770 in every edge $e_j \in E$ where node v_i appears, edge job J_{n+j}^i of vertex chain \mathcal{C}'_i is scheduled at
 771 time $6(n+j) + 1 + c_i$. Thus for all jobs J in our proposed schedule, if its chain starts at
 772 time $6i + l$ with $l \in \{0, 1, 2\}$ then it is scheduled at time $r(J) + l$.

773 We show that there are never more than $M = 2n + 1$ jobs scheduled at any time position.
 774 According to the previous paragraph we can infer that for every chain triplet two propagators
 775 are scheduled at every even time position and one propagator at every odd time position in
 776 time segment $[6i, 6(2n + m - (i+1)) + 3)$. Recall that only the chain triplets of index $j \leq i$
 777 are present in the two time segments $[0, 6n), 6(n + 2m - (i+1))$ related to node v_i . With
 778 Definition 27 we count the number of propagators plus the number of fill jobs at every time
 779 position and show that it is always no more than $M - 1$:

780 **(1) Color choice segment $[0, 6n)$**

781 Let $i \in [0, n-1]$. In time segment $[6i, 6(i+1))$:

- 782 – At time $6i$: $M - 1 - 2i$ fill jobs and $2i$ propagators which add up to $M - 1$ jobs.
- 783 – At time $6i + 1$: set $M - 1 - i$ fill jobs and i propagators which add up to $M - 1$ jobs.
- 784 – At time $6i + 2$: set $M - 2 - 2i$ fill jobs and $2i$ propagators which add up to $M - 2$ jobs.
- 785 – At times $6i + 3, 6i + 4, 6i + 5$: respectively $i, 2i, i$ propagators.

786 **(2) Edge check segments $[6n, 6(n+m))$**

787 Let $j \in [0, m-1]$. In time segment $[6(n+j), 6(n+j+1))$:

- 788 – At time $6(n+j) + 1$: $M - n - 1$ fill jobs and n propagators which add up to $M - 1$
 789 jobs.
- 790 – At time $6(n+j) + 3$: $M - n - 1$ fill jobs and n propagators which add up to $M - 1$
 791 jobs.
- 792 – At times $6(n+j), 6(n+j) + 2, 6(n+j) + 4, 6(n+j) + 5$: respectively $2n, 2n, 2n, i$
 793 propagators.

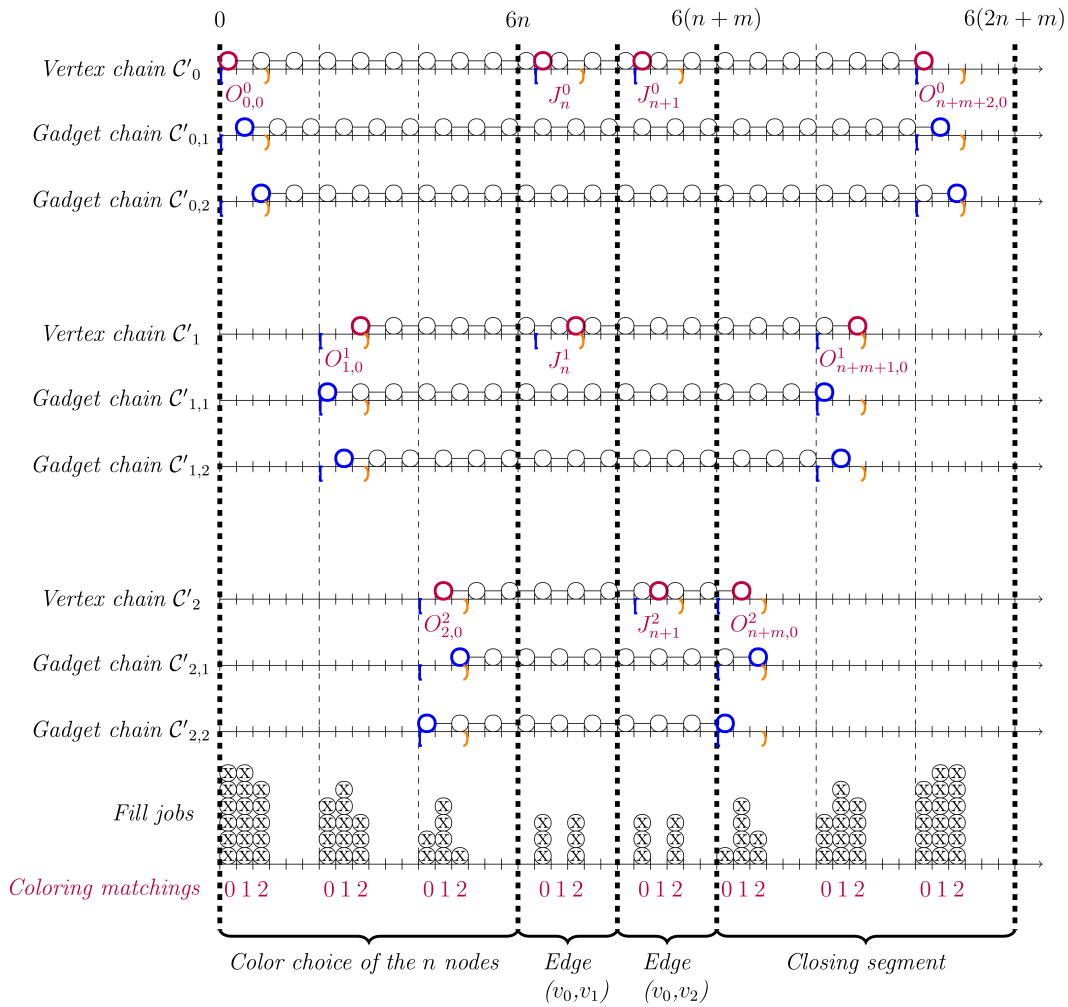
794 **(3) Closing segment $[6(n+m), 6(2n+m))$**

795 Let $i \in [0, n-1]$. In time segment $[6(n+2m - (i+1)), 6(n+2m - i))$:

- 796 – At time $6(n+2m - (i+1))$: set $M - 2 - 2i$ fill jobs and $2i$ propagators which add up
 797 to $M - 2$ jobs.
- 798 – At time $6(n+2m - (i+1)) + 1$: set $M - 1 - i$ fill jobs and i propagators which add
 799 up to $M - 1$ jobs.
- 800 – At time $6(n+2m - (i+1)) + 2$: set $M - 1 - 2i$ fill jobs and $2i$ propagators which
 801 add up to $M - 1$ jobs.
- 802 – At times $6(n+2m - (i+1)) + 3, 6(n+2m - (i+1)) + 4, 6(n+2m - (i+1)) + 5$:
 803 respectively $i, 2i, i$, propagators.

804 Thus only the two edge jobs $J_{n+j}^{i_1}, J_{n+j}^{i_2}$ from an edge $e_j = \{v_{i_1}, v_{i_2}\}$ could invalidate the
 805 schedule if both jobs were scheduled at the same time. This would mean that $6(n+j) + 1 + c_{i_1} =$
 806 $6(n+j) + 1 + c_{i_2}$ and thus $c_{i_1} = c_{i_2}$, which is impossible since we started from a valid
 807 3-coloring. Therefore at most $2n + 1 = M$ jobs are scheduled at any time position.

808 (\Leftarrow) Suppose we have a feasible schedule. For all $0 \leq i \leq n - 1$, let $s_i \in \{0, 1, 2\}$ be
 809 such that $6i + s_i$ is the starting time of chain C'_i (recall that it can only be an odd time
 810 because of the fill jobs defined in Definition 15). We show that (s_0, \dots, s_{n-1}) is a 3-coloring
 811 of G . By contradiction suppose there is an edge $e_j = \{v_{i_1}, v_{i_2}\} \in E$ such that $s_{i_1} = s_{i_2}$. Then
 812 according to Lemma 29 jobs $J_{n+j}^{i_1}$ and $J_{n+j}^{i_2}$ are scheduled at the same time $6(n+j) + s_{i_1}$.
 813 However according to Definition 27 and Lemma 29 fill jobs and propagators add up to $M - 1$
 814 in all three positions $6(n+j) + 1, 6(n+j) + 2, 6(n+j) + 3$.
 815 Thus adding both edge jobs there are $M + 1$ jobs scheduled at one of these three time
 816 positions, which would make the schedule not feasible. This leads to a contradiction. Thus
 817 (s_0, \dots, s_{n-1}) is indeed a 3-coloring of G . \blacktriangleleft



■ **Figure 7** An instance of $P|chains(\ell_{i,j}^{min}), p_j = 1, r_j, d_j|*$ with $M = 2n + 1 = 7$ machines representing a graph coloring. We have $G = (V, E)$ with $V = \{v_0, v_1, v_2\}$ and $E = (\{v_0, v_1\}, \{v_0, v_2\})$. This schedule corresponds to the coloring $(0, 2, 1)$.