Complexity of parallel scheduling unit-time jobs with in-tree precedence constraints while minimizing the mean flow time

Tiayu Wang, Odile Bellenguez-Morineau

Noname manuscript No.
(will be inserted by the editor)

Complexity of parallel scheduling unit-time jobs with in-tree precedence constraints while minimizing the mean flow time

Tianyu Wang · Odile Bellenguez-Morineau

Received: date / Accepted: date

Abstract This paper deals with a particular scheduling problem. We consider unit-time jobs and in-tree precedence constraints while minimizing the mean flow time. This problem is observed as $P|p_j = 1, \text{in-tree}|\sum C_j$ with the use of the 3-filed notation. To the best of our knowledge, its complexity is still open. Through a reduction from 3-PARTITION, we show that this problem is $\mathcal{NP}$-complete.

Keywords parallel scheduling · in-tree · precedence constraints · complexity theory

Acknowledgments

This work was supported by the China Scholarship Council [grant numbers 201404490037].

1 Introduction

We consider the following scheduling problem: a set of n unit-time jobs ($p_j = 1$) has to be done by m identical parallel machines. The jobs are submitted to precedence constraints. Prot and Bellenguez-Morineau (2018) revealed the fact that the complexity of a problem is likely to be different for specific type of precedence graphs. In this paper, we focus on the problem with *in-trees* (or *in-forest* in some literature) precedence graph. This particular graph suggests that each job has no more than one successor. We use C_j to denote the completion time of a job j. The problem involves finding an optimal schedule, respecting the precedence constraints and minimizing the total completion time $\sum C_j$. This criterion is termed as *mean flow time*(noted as MFT in the following text). When the number of machines m is not fixed, this problem is noted as $P|p_j = 1, \text{in-tree}|\sum C_j$ in the 3-field notation of Graham et al. (1979).

This problem was among the *minimal open* problems according to Sigrid Knust (2009). To the best of our knowledge, it is still open (Prot and Bellenguez-Morineau, 2018). Herein, we aim at providing a proof of its $\mathcal{NP}$-completeness.

Organization of this paper is as hereunder: in the subsequent section, we present the state of the art. In section 3, we prove that the problem $P|p_j = 1, \text{in-tree}|\sum C_j$ is $\mathcal{NP}$-complete. In subsection 3.1, we present how we reduce the 3-PARTITION problem to the decision version of the scheduling problem. We prove the $\mathcal{NP}$-completeness in detail in subsection 3.2 and 3.3. Finally, section 4 provides our conclusion.

2 State of the art

A study involving in-tree precedence constraints in a parallel machine scheduling problem has been performed: Hu (1961) focus on the makespan and suggests that the problem is polynomially solvable using the HLF(highest level first) strategy. Nevertheless, Garey et al. (1983) prove that the problem is $\mathcal{NP}$-complete when the precedence graph is an opposing forest, which corresponds to a set of in-trees and out-trees.

HLF can also be used to minimize the MFT if the precedence graph is an out-tree. Furthermore, a polynomial algorithm is put forward by Brucker et al. (2001) for the solution of this problem in case of allowance of preemption.

Nevertheless, Huo and Leung (2006) show HLF cannot optimally solve the in-tree version. In addition, Baptiste et al. (2004) proved that it can be solved in $O(n^m)$ time, i.e. if the number of machines m is a fixed parameter, the problem is polynomially solvable.

Finally, Garey et al. (1983) prove that the problem is $\mathcal{NP}$-complete for a profile scheduling. Accordingly, the number of available machines varies along the time rather than parallel machines. However, the complexity of the given parallel scheduling problem $P|p_j = 1, \text{in-tree}|\sum C_j$ is still open.

Tianyu Wang
Institut Mines-Télécom Atlantique, LS2N, UMR CNRS 6004, 4 rue Alfred Kastler, B.P. 20722 F-44307 Nantes Cedex 3. France
Tel. +33 7 82 99 22 36
E-mail: tianyu.wang@emn.fr

Odile Bellenguez-Morineau
Institut Mines-Télécom Atlantique, LS2N, UMR CNRS 6004, 4 rue Alfred Kastler, B.P. 20722 F-44307 Nantes Cedex 3. France
E-mail: odile.bellenguez@imt.atlantique.fr

3 Proof of $\mathcal{NP}$-completeness

In a bid to prove $\mathcal{NP}$-completeness of $P|p_j = 1, \text{in-tree}|\sum C_j$, we employed a reduction from a known $\mathcal{NP}$-complete problem: 3-PARTITION. An instance π_1 of the 3-PARTITION problem (noted Π_1) is defined as hereunder (Garey and Johnson, 2002):

Definition 1. *Let α be a set of $3q$ elements and $\mathfrak{B}$ an integer, we have $\forall a \in \alpha$, $\exists \mathfrak{w}_a \in Z^+$, $\frac{\mathfrak{B}}{4} < \mathfrak{w}_a < \frac{\mathfrak{B}}{2}$ and $\sum_{a\in\alpha} \mathfrak{w}_a = q\mathfrak{B}$. The question is: is there a partition of α, which is $\alpha_1, \alpha_2, \ldots, \alpha_q$, such that $\forall i \leq q$, $\sum_{a\in\alpha_i} \mathfrak{w}_a = \mathfrak{B}$?*

As Π_1 is $\mathcal{NP}$-complete in the strong sense, the theorem provided hereunder allows us to employ a pseudo-polynomial transformation(Garey and Johnson, 2002, p. 101):

Theorem 1. *If Π is $\mathcal{NP}$-complete in the strong sense, $\Pi' \in \mathcal{NP}$ and there exists a pseudo-polynomial transformation from Π to Π', then Π' is $\mathcal{NP}$-complete in the strong sense.*

Firstly, we define Π_2 as $P|p_j = 1, in-tree|\sum C_j \leq MFT^*$, which represents the decision version of our scheduling problem. The decision question deals with *whether there exists a feasible schedule such that $\sum C_j \leq MFT^*$ or not*, where MFT^* is a given integer. In the subsequent part, we demonstrate how to reduce any instance π_1 of Π_1 to an instance π_2 of Π_2.

3.1 Transformation

In order to begin our transformation, we define some constants:

- $B = q^5\mathfrak{B}$
- $\forall a \in \alpha$, $w_a = q^5\mathfrak{w}_a$
- $K = B^5$
- $m = 3q + 1 + B$, which will be used as the number of machines
- $MFT^* = \sum_{i=1}^{q} i(3(q+1-i)+1) + \sum_{i=q+1}^{K^3} i + B\sum_{i=1}^{q} i + \sum_{a\in\alpha}\sum_{i=1}^{Kw_a} i + KB\sum_{i=1}^{q} i$
- $\forall k \leq q+1$, $m_k = B + 3k - 3$

Notice that an optimal partition for π_1, such that $\sum_{a\in\alpha_i} \mathfrak{w}_a = \mathfrak{B}$, is totally equivalent to a partition such that $\sum_{a\in\alpha_i} w_a = B$, since the problem does not change when multiplying $\mathfrak{w}_a$ by an integer q^5.

Thereafter, we create the following jobs:

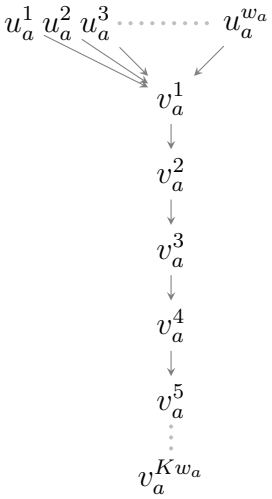

Fig. 1: in-tree of u-jobs and v-jobs

- For each $a \in \alpha$, we build two sets of jobs, which include u-jobs and v-jobs: $u_a^1, u_a^2, \ldots, u_a^{w_a}$, and $v_a^1, v_a^2, \ldots, v_a^{Kw_a}$. The u-jobs precede directly v_a^1, i.e. $\forall i \leq w_a$, $u_a^i \prec v_a^1$. Subsequent to that, we set $v_a^i \prec v_a^{i+1}$, $\forall i < Kw_a$, and they form a chain. The corresponding tree of the u-jobs and v-jobs can be observed in the Figure 1. They form $3q$ in-trees
- In addition, we consider a set of d-jobs, which are defined as hereunder:

$$
\begin{aligned}
&d_1^1, \ldots, d_1^{m-m_1}\\
&d_2^1, \ldots, d_2^{m-m_2}\\
&\ldots\\
&d_q^1, d_q^2, d_q^3, d_q^4\\
&d_{q+1}\\
&d_{q+2}\\
&\ldots\\
&d_{K^3}
\end{aligned}
$$

Those d-jobs also design a tree. In that tree, we fix for any $k \leq q+1$, $d_{k-1}^1 \prec d_k^1$, $d_{k-1}^2 \prec d_k^1$ and $d_{k-1}^3 \prec d_k^1$. $\forall i \in \{4, 5, \ldots, m - m_{k-1}\}$, then, we set: $d_{k-1}^i \prec d_k^{i-3}$. Thereafter, for $\forall k \in \{q+2, q+3, \ldots, K^3\}$, $d_{k-1} \prec d_k$, accordingly forming a chain. The corresponding tree is presented in the Figure 2.

Thus, we transformed π_1 to an instance π_2 of the scheduling problem Π_2. The construction is made in polynomial time of $\mathfrak{B}$ and q (pseudo-polynomial). In accordance to the Theorem 1, we can establish the $\mathcal{NP}$-completeness of Π_2 through the provision of theorem provided as hereunder:

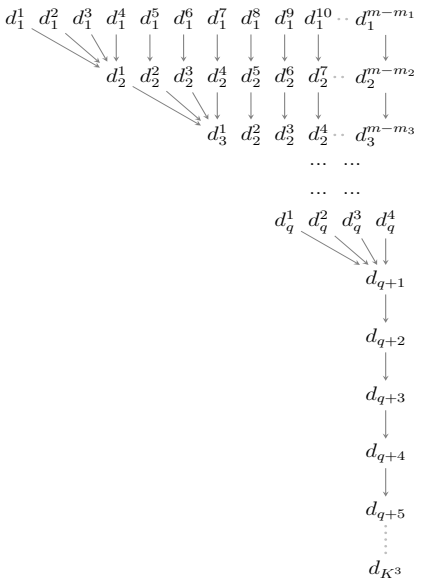

Fig. 2: in-tree of d-jobs

Theorem 2. *There exists a partition for $\pi_1 \in \Pi_1$, if and only if there exists a feasible schedule for π_2 such that $\sum C_j \leq MFT^*$.*

3.2 Proof of Theorem 2: $\Rightarrow$

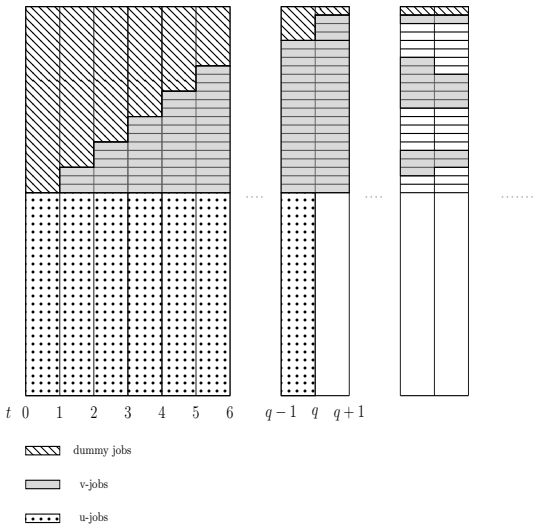

Fig. 3: Gantt graph of σ^*

We firstly show that if there exists a partition $\alpha_1, \alpha_2, \ldots, \alpha_q$ for π_1, such that $\sum_{a \in \alpha_i} w_a = B$, then, we can build a feasible schedule σ^* for π_2 and $\sum C_j \leq MFT^*$. w.l.o.g: moreover, we assume that, for each subset α_k in the partition for $k = 1, \ldots, q$, the three elements in it are a_1^k, a_2^k and a_3^k.

We make use of notation MFT_d^* (resp. MFT_v^* and MFT_u^*) for representing the mean flow time of d-jobs(resp. v-jobs and u-jobs) in σ^*. In the subsequent part, we show how jobs are scheduled in σ^* together with calculating the MFT.

Firstly, d-jobs are executed as soon as they are liberated. Formally, $\forall i \leq q, \forall j \leq m - m_i$, $C_{d_i^j} = i$; $\forall i \leq K^3$, $C_{d_i} = i$. This enforces a profile, wherein, the k^{th} time-slot for $k \leq q+1$ has m_k available machines for both u-jobs and v-jobs.

Accordingly, we consider MFT for the d-jobs:

$$MFT_d^* = \sum_{i=1}^{q} \sum_{j=1}^{m-m^i} C_{d_i^j} + \sum_{i=q+1}^{K^3} C_{d_i} \tag{1}$$

as $\forall i \leq K^3, \forall j \leq m - m_i$, $C_{d_i^j} = i$, the first part is equivalent to:

$$\sum_{i=1}^{q} \sum_{j=1}^{m-m_i} C_{d_i^j} = \sum_{i=1}^{q} \sum_{j=1}^{m-m_i} i = \sum_{i=1}^{q} i(m - m_i)$$

and as $m - m_i = 3(q + 1 - i) + 1$, we have:

$$\sum_{i=1}^{q} \sum_{j=1}^{m-m_i} C_{d_i^j} = \sum_{i=1}^{q} i(3(q + 1 - i) + 1)$$

Then, for $i > q$, we also have $C_{d_i} = i$, so the second part of (1) is:

$$\sum_{i=q+1}^{K^3} C_{d_i} = \sum_{i=q+1}^{K^3} i$$

Accordingly, we can obtain the MFT of all d-jobs:

$$MFT_d^* = \sum_{i=1}^{q} i(3(q + 1 - i) + 1) + \sum_{i=q+1}^{K^3} i \tag{2}$$

In the second step, all the u-jobs corresponding to the subset α_k are executed in the same time-slot, which is the k^{th} time-slot. They need exactly B machines by definition of 3-PARTITION. Formally, $\forall k \leq q$, $\forall a \in \alpha_k$, $\forall i \in \{1, 2, 3\}$, and $\forall j \leq w_{a_i^k}$, $C_{u^j_{a_i^k}} = k$. Accordingly, there are exactly B u-jobs executed at each time slot, i.e. $|\{C_u = t\}| = B, \forall t \leq q$. Thus, we are able to deduce that all the u-jobs are accomplished before $t = q$, and we have:

$$MFT_u^* = \sum_{t=1}^{q} t|\{u|C_u = t\}| = B \sum_{i=1}^{q} i \tag{3}$$

Thereafter, every v-job is executed without any delay in accordance with the precedence constraints. Formally, $\forall a \in \alpha$, $C_{v_a^1} = C_{u_a^1} + 1$ and $\forall i \in \{2, 3, \ldots, Kw_a\}$, $C_{v_a^i} = C_{v_a^{i-1}} + 1$.

So, for each v-chain corresponding to the element $a \in \alpha$, we have $C_{v_a^i} = C_{v_a^{i-1}} + 1$, $\forall i \in \{2, 3, \ldots, Kw_a\}$, that is $C_{v_a^i} = C_{v_a^1} + i - 1$. Let u_a be any u-job preceding v_a^1, as $C_{v_a^1} = C_{u_a} + 1$, we have $C_{v_a^i} = C_{u_a} + i$. Now, we calculate the MFT for this given chain:

$$\sum_{i=1}^{Kw_a} C_{v_a^i} = \sum_{i=1}^{Kw_a} C_{u_a} + \sum_{i=1}^{Kw_a} i = Kw_a C_{u_a} + \sum_{i=1}^{Kw_a} i$$

Then, we deduce the MFT for all these v-chains, i.e. all the v-jobs:

$$MFT_v^* = \sum_{\forall a \in \alpha} \sum_{i=1}^{Kw_a} C_{v_a^i} = K \sum_{\forall a \in \alpha} w_a C_{u_a} + \sum_{\forall a \in \alpha} \sum_{i=1}^{Kw_a} i$$

We are aware of the fact that, for each subset α_k, we have $C_{u_{a_j^i}} = i$, where $j \in \{1, 2, 3\}$, and $i \leq q$. So, we are able to deduce that the first part is equivalent to:

$$K \sum_{\forall a \in \alpha} w_a \sum_{j \leq w_a} C_{u_a^j} = K \sum_{i=1}^{q} \sum_{\forall a \in \alpha_i} w_a \sum_{j \leq w_a} C_{u_a^j} = K \sum_{i=1}^{q} i \sum_{\forall a \in \alpha_i} w_a = K \sum_{i=1}^{q} iB$$

So,

$$MFT_v^* = K \sum_{i=1}^{q} iB + \sum_{\forall a \in \alpha} \sum_{i=1}^{Kw_a} i \tag{4}$$

So, in k^{th} time slot of σ^*, $k < q+1$, there are exactly $m - m_k$ d-jobs, B v-jobs and $3k-3$ v-jobs. By definition of m_k, there are exactly m jobs in progress. After $t = q$, there are only one d-job and at most $3q$ v-jobs executed in parallel. Thus, there are no more than m jobs. So, this schedule respects the availability of machines.

Then, we deduce that the total MFT of σ^*, which is a feasible schedule for π_2, is:

$$\sum C_j = MFT_v^* + MFT_u^* + MFT_d^* = MFT^*$$

3.3 Proof of Theorem 2: $\Leftarrow$

In this section, we prove that if π_2 has a feasible schedule, then π_1 has a partition. A schedule without unforced idle time is termed as *non-delay* schedule by Weiss and Pinedo (2012). The authors also prove that there always exists an optimal non-delay schedule. Let σ be a feasible non-delay schedule. We prove in the following part that σ takes the same shape as σ^* and followed by building a partition from σ.

For that purpose, we at first, define the MFT of u-jobs, v-jobs and d-jobs in σ as MFT_u, MFT_v and MFT_d. Subsequent to that, define the following gaps:

$$\Delta_u = MFT_u - MFT_u^*$$

$$\Delta_v = MFT_v - MFT_v^*$$

$$\Delta_d = MFT_d - MFT_d^*$$

In order to respect the MFT^*, we understand that:

$$\Delta_d + \Delta_u + \Delta_v \leq 0 \tag{5}$$

In addition to this, we would like to throw emphasis on the following observations:

Observation 1. *When a chain of n_c jobs is right-shifted by one time slot, then its MFT will be increased by n_c.*

Observation 2. *The sizes of different sets of jobs exhibit the following relation:*

$$K^3 \gg Kw_a \gg B \gg q^5$$

First of all, let us consider the d-jobs. We show that d-jobs must have the same profile as in σ^*, which is the following lemma:

Lemma 1. *In σ, the d-jobs are scheduled as soon as they are liberated (as in σ^*), i.e. $\Delta_d = 0$.*

Proof. By contradiction: assume that there is at least one delayed d-job. σ is a non-delay schedule; accordingly, this d-jobs can be delayed only by insertion of u-jobs or v-jobs.

Notice that the d-jobs can take no more than $m - m_1$ machines in parallel because of the precedence constraints. At any moment, there are at least $m_1 = B$ available machines for both the v-jobs and u-jobs. Moreover, in a non-delay schedule, the makespan(C_{max}) is not larger than the number of jobs. As the total number of v-jobs and u-jobs is $KB + qB$, the last v-job or u-job finishes before $KB + qB$. Consequently, the delay must happen before $KB + qB$.

As the length of the d-chain is K^3, at least $K^3 - KB + qB$ jobs are right-shifted. Following the Observation 1, we have

$$\Delta_d \geq K^3 - KB + qB$$

Remind that $MFT_u^* \ll K^2$, $MFT_v^* \ll K^2$, so we have

$$\Delta_d \gg MFT_u^* + MFT_v^*$$

In accordance with the definition, we have $\Delta_u > -MFT_u^*$ and $\Delta_v > -MFT_v^*$, then

$$\begin{aligned}&\Delta_d + \Delta_v + \Delta_u\\ &> \Delta_d - MFT_u^* - MFT_v^* \gg 0\end{aligned}$$

This is a contradiction to (5). So, our assumption is false, there is no delay during the execution of d-jobs. □

Since $\Delta_d = 0$, (5) becomes

$$\Delta_u + \Delta_v \leq 0 \tag{6}$$

Here we conclude that the profile of available machines in σ is the same as in σ^* to schedule u-jobs and v-jobs. Now, let us deal with u-jobs.

Definition 2. *We define the u-jobs in the same component a* ***u-set*** *and define U_t as the number of u-jobs in u-sets completed at t.*

Lemma 2. *At time $t^* \leq q$, the completed u-sets contain at most t^*B u-jobs, i.e. $\sum_{t=1}^{t^*} U_t \leq t^*B$.*

Proof. The number of jobs finished at $t^* \leq q$ is at most t^*m which is:

$$t^*m = t^*B + t^*(3q+1) \leq t^*B + 3q^3$$

As we are aware of the fact, the number of u-jobs in a u-set is w_a, which is integer times q^5, so, the number of u-jobs in completed u-sets is at most t^*B. □

Now, let us move on to the v-jobs.

Definition 3. *We define V_t as the number of v-jobs in the chains starting at the time t for $t < q$ in σ, and V_q as the number of v-jobs in the chains starting at or after $t = q$.*

Lemma 3. *In σ, at any time $t \leq q$, the total length of the v-chains beginning at t in σ is exactly KB, as in σ^*, i.e. $V_t = KB$.*

Proof. At first, we prove that $\sum_{i=1}^{q} iV_i \geq KB\sum_{i=1}^{q} i$.

According to the precedence constraints, we have a relation between V_t and U_t:

$$\sum_{t=1}^{t^*} V_t \leq K\sum_{t=1}^{t^*} U_t \qquad \forall t^* < q$$

So, from Lemma 2, we can deduce:

$$\sum_{t=1}^{t^*} V_t \leq t^*BK \qquad \forall t^* < q$$

which is:

$$\begin{aligned}&V_1 \leq KB\\ &V_1 + V_2 \leq 2KB\\ &V_1 + V_2 + V_3 \leq 3KB\\ &\cdots\\ &\cdots\\ &V_1 + V_2 + V_3 + \cdots + V_{q-1} \leq (q-1)KB\end{aligned} \tag{7}$$

If we sum them up, we get:

$$(q-1)V_1 + (q-2)V_2 + \cdots + V_{q-1} \leq KB\frac{q^2-q}{2} \tag{8}$$

By definition, we understand that $\sum_{t=1}^{q} V_t$ corresponds to the norm of whole set of v-jobs and it is equal to KqB:

$$\sum_{t=1}^{q} V_t = KqB \tag{9}$$

Then, by multiplying (9) by q, we get:

$$qV_1 + qV_2 + \ldots qV_q = KBq^2 \tag{10}$$

With (10)-(8), we get

$$V_1 + 2V_2 + \cdots + (q-1)V_{q-1} + qV_q \geq KB\frac{q^2+q}{2}$$

which is exactly

$$\sum_{i=1}^{q} iV_i \geq KB\sum_{i=1}^{q} i$$

It requires observation that the two terms are equal if and only if every decomposition of group (7) is written with an equality, which indicates that $V_1 = V_2 = \cdots = V_q = KB$.

Now, we demonstrate that $\sum_{i=1}^{q} iV_i = KB\sum_{i=1}^{q} i$. Suppose by absurd that $\sum_{i=1}^{q} iV_i > KB\sum_{i=1}^{q} i$. By definition, V_i is the total length of some v-chains, which implies that V_i is integer times K. So, we have:

$$\sum_{i=1}^{q} iV_i \geq KB\sum_{i=1}^{q} i + K \tag{11}$$

In the best case of σ, all the v-jobs in one chain are executed without any delay, so, we have:

$$MFT_v \geq \sum_{a\in\alpha}\sum_{i=1}^{Kw_a} i + \sum_{i=1}^{q} iV_i \tag{12}$$

Reminding the expression of MFT_v^* in (4), we can deduce that:

$$\Delta_v \geq \sum_{i=1}^{q} iV_i - KB\sum_{i=1}^{q} i \tag{13}$$

By (13) and (11), we have$\Delta_v \geq K$. Then, by (6), we have $\Delta_u \leq -K$. This contradicts the fact that $\Delta_u \geq -MFT_u^* \gg -K$.

Finally, we can conclude that $\Delta_v = 0$ and $V_t = KB$, $\forall t \leq q$. This suggests that, at any t, the length of v-chains started is exactly KB. □

The Lemma above allows us to build a partition from V_i by selecting the elements corresponding to its 3 v-chains of KB jobs. Thus, we accomplished the proof for the Theorem 2.

4 Conclusion

In this paper, we put forward a proof of $\mathcal{NP}$-completeness for $P|p_j = 1, \text{in-tree}|\sum C_j$. Based on that, we refine the knowledge about the complexity of parallel scheduling problems subjected to precedence constraints in specific graphs.

Nevertheless, some open problems still exist requiring investigation. For example, when preemption is allowed, the complexity of the problem $P|p_j = 1, \text{in-tree}, pmtn|\sum C_j$ is unknown. We conjecture that, with the same reduction as in subsection 3.1, the Theorem 2 still holds even if preemption is allowed, and the problem shall be $\mathcal{NP}$-complete as well.

Another possible research avenue involves exploration of how to solve the problem when m is a fixed number. The complexity of the algorithm proposed by Baptiste et al. (2004) is $O(n^m)$, but it can be certainly improved.

References

Philippe Baptiste, Peter Brucker, Sigrid Knust, and Vadim G Timkovsky. Ten notes on equal-processing-time scheduling. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2 (2):111–127, 2004.

Peter Brucker, Johann L Hurink, and Sigrid Knust. A polynomial algorithm for $P|p_j = 1, r_j, outtree|\sum C_j$. *Math. Methods Oper. Res.*, 2001.

Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

MR Garey, DS Johnson, RE Tarjan, and Yannakakis M. Scheduling opposing forests. *SIAM Journal on Algebraic Discrete Methods*, 4(1):72–93, 1983.

Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a surveyhaha. *Annals of discrete mathematics*, 5:287–326, 1979.

Te C Hu. Parallel sequencing and assembly line problems. *Operations research*, 9(6):841–848, 1961.

Yumei Huo and Joseph Y-T Leung. Minimizing mean flow time for UET tasks. *ACM Transactions on Algorithms (TALG)*, 2(2):244–262, 2006.

Damien Prot and Odile Bellenguez-Morineau. A survey on how the structure of precedence constraints may change the complexity class of scheduling problems. *Journal of Scheduling*, 21(1):3–16, 2018.

Peter Brucker Sigrid Knust. Complexity results for scheduling problems, 2009. http://www2.informatik.uni-osnabrueck.de/knust/class/.

Gideon Weiss and Michael Pinedo. Scheduling: Theory, algorithms, and systems, 2012.