ELSEVIER Discrete Applied Mathematics 58 (1995) 253–280

DISCRETE APPLIED MATHEMATICS

Preemptive scheduling with variable profile, precedence constraints and due dates

Zhen Liu*[,a], Eric Sanlaville[b]

[a] *INRIA, Centre Sophia Antipolis, 2004 Route des Lucioles, B.P. 93, 06902 Sophia Antipolis, France*
[b] *Laboratoire LITP, Université Pierre et Marie Curie, 4, place Jussieu, 75252 Paris Cedex 05, France*

Received 18 February 1992; revised 9 September 1993

Abstract

This paper is concerned with the problem of scheduling preemptive tasks subject to precedence constraints in order to minimize the maximum lateness and the makespan. The number of available parallel processors is allowed to vary in time. It is shown that when an earliest due date first algorithm provides an optimal nonpreemptive schedule for unit-execution-time (UET) tasks, the preemptive priority scheduling algorithm, referred to as smallest laxity first, provides an optimal preemptive schedule for real-execution-time (RET) tasks. When the objective is to minimize the makespan, we get the same kind of result between highest level first schedules solving nonpreemptive tasks with UET and the longest remaining path first schedule for the corresponding preemptive scheduling problem with RET tasks. These results are applied to four specific profile scheduling problems and new optimality results are obtained.

Keywords: Preemptive Scheduling; List schedule; Priority schedule; Variable profile; Precedence constraints; Due dates; Lateness; Makespan

1. Introduction

We consider the preemptive profile scheduling of partially ordered tasks. The tasks are modeled by a directed acyclic graph where vertices represent tasks and arcs represent precedence relations between the tasks. These tasks are executed, subject to precedence constraints, on some identical parallel processors. The number of processors available to these tasks, referred to as the profile, may vary with time. The executions may be preempted and resumed without any penalty. At any time, a task can be assigned to at most one processor, and a processor can execute at most one task. Each task is associated with a due date. The objective is to minimize the maximum lateness and the makespan (when due dates are not taken into consideration).

*Corresponding author.

SSDI 0166-218X(93)E0151-N

The preemptive and nonpreemptive schedulings of partially ordered tasks on multiprocessor systems have been studied by various authors in the literature (see [9] for an extensive survey). The minimizations of the maximum lateness and the makespan are NP-hard problems in general. Polynomial algorithms exist only in some specific cases of the task graphs or the number of processors. List scheduling algorithms form an important class of nonpreemptive scheduling algorithms. They provide optimal solutions for some particular cases (see [2] for studies of their properties). In preemptive scheduling, there is a corresponding class of algorithms, referred to as the priority scheduling algorithms. For three particular scheduling problems, Lawler [8] constructed preemptive priority algorithms, based on the optimal nonpreemptive list algorithms in the case of unit-execution-time (UET) tasks, and showed that these priority algorithms provide optimal preemptive schedules for real-execution-time (RET) tasks.

The notion of profile scheduling was first introduced by Ullman [16] and later by Garey et al. [7] in the complexity analysis of deterministic scheduling algorithms. Dolev and Warmuth [4–6] carried out various studies on the nonpreemptive profile scheduling with UET tasks. They obtained polynomial algorithms that minimize the makespan for specific profiles (e.g., zigzag profile, bounded profile, etc.) and specific task graphs (e.g., in-forest, out-forest, opposing forest, flat graph, etc.). For the preemptive profile problem, Schmidt [15] presented an algorithm for the special case of independent tasks. Liu and Sanlaville [10] analyzed the stochastic preemptive scheduling problems. Simple optimal schedules were provided for the stochastic minimization of the makespan of interval-order task graphs, in-forests, and uniform out-forests.

In this paper, we investigate the preemptive version of the deterministic profile scheduling problems. We show that if some list scheduling algorithms of the type earliest due date (EDD) are optimal within the class of nonpreemptive schedules for UET tasks, then the preemptive priority scheduling algorithm, referred to as smallest laxity first (SLF), is optimal within the class of preemptive schedules for RET tasks. When the minimization of makespan is under consideration, such a result implies that if list scheduling algorithms of the type highest level first (HLF) are optimal within the class of nonpreemptive schedules for UET tasks, the preemptive priority scheduling algorithm, referred to as longest remaining path (LRP), is optimal within the class of preemptive schedules for RET tasks.

These results are applied to four specific profile scheduling problems and the following new preemptive profile scheduling results are obtained:

- SLF (defined on some modified due dates) minimizes the maximum lateness (with respect to the original due dates) when the task graph is an in-forest and the profile is increasing zigzag,
- LRP minimizes the makespan when the task graph is a union of chains,
- LRP minimizes the makespan when the task graph is an in-forest and the profile is increasing zigzag, or when the task graph is an out-forest and the profile is decreasing zigzag,
- LRP minimizes the makespan when the task graph is arbitrary and the profile is bounded by two.

Our results strengthen the relationship between the optimality of the nonpreemptive list algorithms for UET tasks and the optimality of the preemptive priority algorithms for RET tasks. However, our approach is different from that of Lawler [8]. We establish the optimality of the priority algorithms by using directly the optimality of the corresponding list algorithms, whereas in [8], different proofs were necessary in order to obtain the optimality of the priority algorithms for solving different scheduling problems.

The paper is organized as follows. In Section 2, we define the notation and present some preliminary results on the list and priority schedules and on the graph expansions introduced in the paper. In Section 3, we prove the main results of the paper which relate the optimality of the preemptive SLF (resp. LRP) schedule to those of the nonpreemptive EDD (resp. HLF) schedules. In Section 4, we apply the main theorems to the four profile scheduling problems and we obtain optimal preemptive schedules. When necessary, the nonpreemptive counterparts are first studied. In particular, we extend the optimality of EDD schedules (defined on modified due dates), obtained by Brucker et al. [1] for in-forests with UET tasks and constant profiles, to the increasing zigzag profiles. We also prove that HLF schedules are optimal for the minimization of makespan of chains with UET tasks and arbitrary profiles within the class of nonpreemptive schedules.

2. Preliminaries

2.1. Problem description

There are n tasks to be processed by a multiprocessor system. The executions of these tasks are constrained by some precedence relations between the tasks. A task graph $G = (V, E)$ is used to describe these relations, where $V = \{1, 2, \ldots, n\}$ is the set of vertices representing the tasks, and E is the set of arcs representing the precedence relations between tasks. It is assumed that G is a directed acyclic graph (d.a.g.) and that it contains no transivity arcs. Denote by $\pi(i)$ and $\sigma(i)$ the sets of immediate predecessors and successors of task i, respectively. Let $\pi^*(i)$ and $\sigma^*(i)$ be the sets of (not necessarily immediate) predecessors and successors of i, respectively. A task without successor (resp. predecessor) is called a final (resp. an initial) task. Task $i \in V$ has a processing requirement p_i and a due date d_i.

There are $m \geqslant 1$ identical parallel processors in the system with speed 1 (so that the processing time of a task equals its processing requirement). The number of processors available to the execution of task graph G varies with time. Define $M = \{a_r, m_r\}_{r=1}^{\infty}$ as the profile of the system, where $0 = a_1 < a_2 < \cdots < a_r < \cdots$ are the epochs when the number of available processors changes, and m_r is the number of available processors during $[a_r, a_{r+1})$. Without loss of generality, we assume that $m_r \geqslant 1$ for all $r = 1, 2, \ldots$. Since the processors are identical, we can assume that processor 1 is always available. We will also assume that the profile is not changed infinitely often during any finite time interval. Under these assumptions, there is a finite $\bar{r}$ such that $a_{\bar{r}} > \sum_{i \in V} p_i$. Without loss of generality, we can assume that there is at least one task running at any time unless all the tasks have finished. Thus, we will only consider the truncated sequence $M = \{a_r, m_r\}_{r=1}^{\bar{r}}$. A special case of the variable profile is the constant profile where $m_1 = m$ and $a_2 = \infty$.

A scheduling algorithm decides when an enabled task, i.e. an unassigned unfinished task all of whose predecessors have finished, should be assigned to one of the available processors. At any time, a task can be assigned to at most one processor, and a processor can execute at most one task. A schedule is feasible if these constraints (i.e. the precedence relations, the variable profile, the nonredundancy of the task assignment) are satisfied. Scheduling can be either preemptive, i.e. the execution of a task can be stopped and later resumed on any processor without penalty, or nonpreemptive, i.e. once begun, the execution of a task continues on the same processor until its completion.

Let S be an arbitrary feasible schedule of task graph G under profile M. Let $C_i(S)$ be the completion time of task i under S. The lateness of task i is defined as $L_i(S) = C_i(S) - d_i$, and the maximum lateness of schedule S as $L_S(G, M) = \max_{i \in V} L_i(S)$. Denote by $L_p^*(G, M)$ and $L_{np}^*(G, M)$ the smallest maximum latenesses of task graph G obtained by the preemptive and nonpreemptive schedules, respectively, under profile M. Since nonpreemptive schedules are special cases of preemptive ones, it is trivial that $L_p^*(G, M) \leqslant L_{np}^*(G, M)$.

When the due dates are set to zero, the maximum lateness becomes the makespan. Denote by $C_S(G, M) = \max_{i \in V} C_i(S)$ the makespan of (G, M) obtained by schedule S. Let $C_p^*(G, M)$ and $C_{np}^*(G, M)$ be the smallest makespans of task graph G obtained by the preemptive and nonpreemptive schedules, respectively, under profile M.

The goal of this paper is to find preemptive feasible schedules that minimize the maximum lateness and the makespan. More precisely, we will establish a relation between some optimal nonpreemptive schedules and some optimal preemptive schedules.

2.2. List and priority schedules

List algorithms are often used in nonpreemptive scheduling. With such algorithms, there is a (static or dynamic) list of tasks. As soon as a processor is available, the enabled task that is closest to the head of the list is assigned to that processor. The *earliest due date* first (EDD) algorithms form a well known subclass of list algorithms, where the tasks are ordered increasingly by their due dates. The schedules generated by EDD algorithms differ in the way that ties are broken. Let $\mathscr{E}(G, M)$ denote the family of schedules obtained by the EDD algorithms for task graph G and profile M.

In accordance with the list of algorithms, (dynamic) priority algorithms are used in the preemptive scheduling. At any time, enabled tasks are assigned to available processors according to a priority list which may change in time and may depend on the partial schedule already constructed. A general description is given below [8, 13]:

- At any time t, enabled tasks are ordered according to their priorities, thus forming subsets $V_1, \ldots, V_k$, where all tasks of V_j have the same priority and higher priority than tasks in V_{j+1}.
- Suppose that tasks in $V_1, \ldots, V_{r-1}$, $r \leqslant k$, are assigned. Let $\tilde{m}_r(t)$ be the number of remaining free processors. If $\tilde{m}_r(t) \geqslant |V_r|$, then one processor is assigned to each of the tasks in V_r, and the algorithm deals with the next subset. Otherwise, the $\tilde{m}_r(t)$ processors are shared by the tasks of V_r so that each task in V_r is executed at speed $v_r = \tilde{m}_r(t)/|V_r|$.
- This assignment remains unchanged until one of the following events occurs:
  (1) A task is completed,
  (2) the priority of one subset V_{r-1} becomes the same as that of V_r,
  (3) the profile changes.
  At such moments the processor assignment is recomputed.

In the above scheme, the processor sharing can be achieved by McNaughton's wrap-around algorithm [11] which is linear in the number of tasks scheduled in each time interval. An example is illustrated in Fig. 1 where three tasks are executed at speed $\frac{2}{3}$ on two processors during a unit length interval.

Note that other processor sharing schemes can be used. However, they will generate the same latenesses of the tasks provided the corresponding task processing speeds are the same in these processor sharing schemes. Therefore, we will not make any difference between them.

Denote by $p_i^S(t)$ the remaining processing requirement of task i at time t in a schedule S. Define the laxity of task i at time t in this schedule to be $l_i^S(t) = d_i - p_i^S(t)$. The dynamic priority algorithm based on the *smallest laxity first rule* is referred to as *SLF*. The schedule it produces for (G, M) is denoted by SLF(G, M).

List and priority schedules are also used for the minimization of makespan. Some simple algorithms are optimal under certain conditions. We will consider the *highest level first* (HLF) and the *longest remaining path* (LRP) schedules.

Let h_i be the height of task i, defined as the length of the longest path between i and a final task in G. This length is computed as the summation of the processing requirements of the tasks (excluding i) in the path. Note that in the case of UET tasks (i.e. $p_1 = \cdots = p_n = 1$), h_i is also referred to as the level of task i, where, by convention, the level of a final task is 0.

In a highest level first (HLF) list algorithm, the tasks are ordered decreasingly by their heights or levels. Let $\mathscr{H}(G, M)$ denote the family of schedules obtained by the

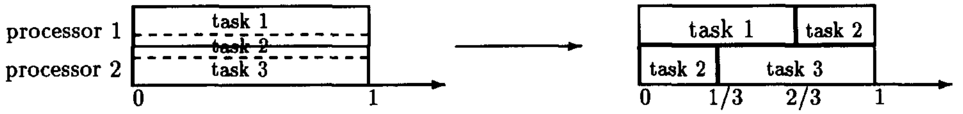

Fig. 1. An example of processor sharing with McNaughton's algorithm.

HLF algorithms for task graph G and profile M. As in the case of EDD schedules, the schedules in $\mathscr{H}(G, M)$ differ in the way that ties are broken.

In the preemptive case, define the length of the remaining longest path at time t in a preemptive schedule S to be $r_i^S(t) = h_i + p_i^S(t)$. In a longest remaining path first (LRP) schedule, the tasks are decreasingly ordered by the lengths of their remaining longest paths.

The following lemma follows from the above definitions.

Lemma 2.1. *For any couple (G, M), if $d_i = -h_i$ for all $i \in V$, then the EDD (resp. SLF) rule coincides with the HLF (resp. LRP) rule.*

2.3. Commensurability and graph expansion

Definition 2.1 (*Commensurability*). The real numbers $x_1, \ldots, x_r \in \mathbb{R}$ are said to be mutually commensurable if there exist $w \in \mathbb{R}$ and r integers $\alpha_1, \ldots, \alpha_r$ such that $x_i = \alpha_i w$ for all $i = 1, \ldots, r$. The real number w is called commensurability factor of $x_1, \ldots, x_r$.

Note that if w is a commensurability factor of $x_1, \ldots, x_r$, then for all $k \in \mathbb{N}_+ \stackrel{\text{def}}{=} \{1, 2, \ldots\}$, w/k is also a commensurability factor of $x_1, \ldots, x_r$.

A task graph G is said to have commensurable timing with commensurability factor w if the processing times and the due dates of the tasks of G are mutually commensurable with commensurability factor w. A special case of commensurable timing of a graph is the UET tasks and integer due dates. The couple (G, M) is said to have commensurable timing with commensurability factor w if the task processing times, the due dates and the profile changing epochs are mutually commensurable with commensurability factor w. In what follows, w always denotes the commensurability factor.

Definition 2.2 (*Graph expansion*). Let $G = (V, E)$ have commensurable timing with commensurability factor w:

$$\forall i \in V, \quad \exists \alpha_i, \beta_i \in \mathbb{Z}: \quad p_i = \alpha_i w, \quad d_i = \beta_i w,$$

where $\mathbb{Z}$ is the set of (positive and negative) integers. For all $k \in \mathbb{N}_+$, we define an operation on G, called the kth expansion of G, as follows: The expanded task graph $G_{w/k} = (V_{w/k}, E_{w/k})$ is obtained by replacing each vertex i in G by a chain of $\alpha_i k$ vertices such that

$$V_{w/k} = \{i_j \mid i \in V, j = 1, \ldots, \alpha_i k\};$$

$$E_{w/k} = \{(i_j, i_{j+1}) \mid i \in V, j = 1, \ldots, \alpha_i k - 1\} \cup \{(i_{\alpha_i k}, i'_1) \mid (i, i') \in E\};$$

$$p_{i_j} = w/k, \quad j = 1, \ldots, \alpha_i k;$$

$$d_{i_j} = d_i - (\alpha_i k - j)w/k, \quad j = 1, \ldots, \alpha_i k.$$

For sake of simplicity, $G_{w/1}$ will be denoted by G_w.

Definition 2.3 (*Schedule expansion and amalgam*). Assume that (G, M) has commensurable timing with commensurability factor w. Let S and S_w be preemptive schedules on (G, M) and (G_w, M), respectively. If $\forall i \in V$, S_w executes one subtask i_j at speed v at time t if and only if S executes the jth portion (with duration w) of task i at speed v at time t, $j = 1, \ldots, \alpha_i$, then S_w is called the w-expansion of schedule S, and S is called the w-amalgam of schedule S_w.

Lemma 2.2. *Assume that (G, M) has commensurable timing with commensurability factor w. Let S and S_w be preemptive schedules on (G, M) and (G_w, M), respectively, such that S_w is the w-expansion of schedule S. Then S_w is a feasible schedule of (G_w, M) if and only if S is a feasible schedule of (G, M). Moreover, if they are both feasible, then*

$$L_S(G, M) = L_{S_w}(G_w, M).$$

Proof. The equivalence of the feasibility of the two schedules is simple to verify and is omitted here. We assume now that they are both feasible. By definition of S_w,

$$C_{i_{\alpha_i}}(S_w) = C_i(S)$$

for all $i \in V$. Hence,

$$\begin{aligned} L_{S_w}(G_w, M) &= \max_{i \in V} \max_{1 \leqslant j \leqslant \alpha_i} (C_{i_j}(S_w) - d_{i_j}) \geqslant \max_{i \in V} (C_{i_{\alpha_i}}(S_w) - d_{i_{\alpha_i}}) \\ &= \max_{i \in V} (C_i(S) - d_i) = L_S(G, M). \end{aligned}$$

On the other hand, for all $1 \leqslant j \leqslant \alpha_i$,

$$C_{i_j}(S_w) - d_{i_j} \leqslant C_i(S) - w(\alpha_i - j) - d_{i_j} = C_i(S) - d_i.$$

Hence the result. $\square$

Lemma 2.3. *Assume that (G, M) has commensurable timing with commensurability factor w. Then*

$$L_p^*(G, M) = L_p^*(G_w, M).$$

Proof. Suppose S^* is an optimal schedule for (G, M). Lemma 2.2 entails that

$$L_p^*(G, M) = L_{S^*}(G, M) = L_{S_w^*}(G_w, M) \geqslant L_p^*(G_w, M).$$

Conversely, let U^* be an optimal schedule for (G_m, M), and S be its w-amalgam. Then Lemma 2.2 entails that

$$L_p^*(G_w, M) = L_{U^*}(G_w, M) = L_S(G, M) \geqslant L_p^*(G, M)$$

so that the lemma is proved. $\square$

Now we specifically consider SLF schedules.

Lemma 2.4. *Assume that* (G, M) *has commensurable timing with commensurability factor* w. *Let* SLF_w *be the expanded schedule issued from* $\mathrm{SLF}(G, M)$. *Then* SLF_w *is identical to the schedule obtained by directly applying the SLF rule to* (G_w, M):

$$\mathrm{SLF}_w \equiv \mathrm{SLF}(G_w, M).$$

Proof. The proof is by induction on the events in SLF_w. Let $0 = t_0 < t_1 < t_2 < \cdots$ be the time epochs when the assignment decisions of SLF_w are made. By definition of the expanded schedule,

$$\forall i \in V, \quad \forall t \in \mathbb{R}^+: \quad p_i^{\mathrm{SLF}}(t) = \sum_{j=1}^{\alpha_i} p_{i_j}^{\mathrm{SLF}_w}(t). \tag{1}$$

Let $\mathrm{SLF}' = \mathrm{SLF}(G_w, M)$. We will prove that

$$p_i^{\mathrm{SLF}}(t_s) = \sum_{j=1}^{\alpha_i} p_{i_j}^{\mathrm{SLF}'}(t_s), \quad i \in V, \quad s = 0, 1, 2, \ldots, \tag{2}$$

which, together with Eq. (1), will imply the assertion of the lemma.

It is trivial that Eq. (2) holds when $s = 0$ as $p_i(0) = p_i = \sum_{j=1}^{\alpha_i} w$. Suppose it is true for some $s \geqslant 0$. Let i be an enabled task at time t_s under SLF. By induction hypothesis, there is a task i_j of G_w which is enabled at t_s under SLF′. Using the induction hypothesis implies $p_i^{\mathrm{SLF}}(t_s) = p_{i_j}^{\mathrm{SLF}'}(t_s) + w(\alpha_i - j)$ so that

$$l_{i_j}^{\mathrm{SLF}'}(t_s) = d_i - w(\alpha_i - j) - p_{i_j}^{\mathrm{SLF}'}(t_s) = d_i - p_i^{\mathrm{SLF}}(t_s).$$

In words, the laxity of i in $\mathrm{SLF}(G, M)$ at time t_s is the same as that of i_j in $\mathrm{SLF}'(G_w, M)$ at time t_s. If i' is enabled at t_s in $\mathrm{SLF}(G, M)$, and $i'_{j'}$ enabled at t_s in SLF′, then i has higher priority than i' in $\mathrm{SLF}(G, M)$ if and only if i_j has higher priority than $i'_{j'}$ in SLF′. Hence i is scheduled at the same speed in $\mathrm{SLF}(G, M)$ as i_j in SLF′ during $[t_s, t_{s+1})$, so the equality remains true at t_{s+1}. By induction, (2) holds for $s = 0, 1, 2, \ldots$. □

Last, we define some closures of graphs and profiles.

Definition 2.4 (*Closure under expansion of graphs*). A class $\mathscr{C}$ of graphs is said to be closed under expansion if the following property is true for any graph $G = (V, E) \in \mathscr{C}$: For any vertex $i \in V$, if G' is the graph obtained from G by replacing vertex i by a chain of two vertices i_1 and i_2 such that

$$\pi(i_1) = \pi(i), \quad \sigma(i_1) = \{i_2\} \quad \text{and} \quad \pi(i_2) = \{i_1\}, \quad \sigma(i_2) = \sigma(i),$$

then G' still belongs to the class $\mathscr{C}$.

It is straightforward that if the class $\mathscr{C}$ is closed under expansion, and if $G \in \mathscr{C}$, then any k-expansion of G still belongs to $\mathscr{C}$.

Definition 2.5 (*Closure under translation of profiles*). A class $\mathscr{M}$ of profiles is said to be closed under translation if for any profile $M = \{a_r, m_r\}_{r=1}^{\infty}$ in $\mathscr{M}$, all the profiles $M' = \{a'_r, m_r\}_{r=1}^{\infty}$ belong to $\mathscr{M}$, provided $\{a'_r\}_{r=1}^{\infty}$ is an increasing sequence of real numbers.

3. Main results

In this section we will establish the tight relation between the optimality of the EDD algorithms in the nonpreemptive case with UET tasks and that of the SLF algorithm in the preemptive case with RET tasks. We will first consider commensurable-execution-time (CET) tasks.

Theorem 3.1. *Assume that* (G, M) *has commensurable timing with commensurability factor* w. *If*

$$\forall k \in \mathbb{N}_+, \quad \exists S \in \mathscr{E}(G_{w/k}, M)\colon \quad L_S(G_{w/k}, M) = L^*_{\mathrm{np}}(G_{w/k}, M),$$

then SLF(G, M) *is an optimal preemptive schedule*:

$$L_{\mathrm{SLF}}(G, M) = L^*_{\mathrm{p}}(G, M).$$

In other words, if for any expansion of G, there is an EDD list algorithm which minimizes the maximum lateness within the class of nonpreemptive schedules, then the priority schedule SLF is optimal within the class of preemptive schedules. It will be seen later on that this theorem is particularly useful for the classes of graphs which are closed under expansion and for which some list algorithms are known to be optimal. The relation between optimal nonpreemptive EDD schedules and preemptive SLF schedules was first observed by Lawler [8] in the analysis of three special cases of preemptive scheduling.

The proof of this theorem is somewhat tedious and is forwarded to Appendix A. The scheme of the proof is similar to that of [13]. Roughly speaking, we first prove that the optimal preemptive solution for (G, M) may be approached arbitrarily close by considering optimal nonpreemptive schedules when the graph G is sufficiently expanded. Secondly, we show that the sequence of nonpreemptive schedules in $\mathscr{E}(G_{w/k}, M)$ converge to the schedule SLF when k goes to infinity. Putting these two points together yields the desired result.

Let us now consider the problem of makespan minimization. Consider first the following lemma.

Lemma 3.1. *For every feasible (preemptive or nonpreemptive) schedule* S *of* (G, M), *if* $d_i = -h_i$ *for all* $i \in V$, *then*

$$L_S(G, M) = C_S(G, M).$$

Proof. Let $\tilde{V} \subseteq V$ be the set of tasks whose latenesses are equal to the maximum lateness:

$$\tilde{V} = \{i \in V \mid L_S(G, M) = C_i(S) - d_i = C_i(S) + h_i\}.$$

Let k be a task in $\tilde{V}$ with the smallest height: $h_k \leqslant h_i$, $\forall i \in \tilde{V}$.

If k is not a final task, then there is $j \in \sigma(k)$ such that $h_k = h_j + p_j$. As $C_j(S) \geqslant C_k(S) + p_j$, it follows that $C_j(S) + h_j \geqslant C_k(S) + h_k = L_S(G, M)$, which implies that $j \in \tilde{V}$. This last fact contradicts the assumption that task k has the smallest height within $\tilde{V}$. Therefore, k is necessarily a final task.

Thus, $h_k = 0$ so that $L_S(G, M) = C_k(S) \leqslant C_S(G, M)$. As trivially

$$L_S(G, M) = \max_{i \in V}(C_i(S) + h_i) \geqslant \max_{i \in V} C_i(S) = C_S(G, M),$$

the lemma is proved. □

Corollary 3.1. *Assume that* (G, M) *has commensurable timing with commensurability factor* w. *If*

$$\forall k \in \mathbb{N}_+, \quad \exists S \in \mathscr{H}(G_{w/k}, M){:} \quad C_S(G_{w/k}, M) = C^*_{\mathrm{np}}(G_{w/k}, M),$$

then LRP(G, M) *is an optimal preemptive schedule*:

$$C_{\mathrm{LRP}}(G, M) = C^*_{\mathrm{p}}(G, M).$$

Proof. Setting the due dates of G in such a way that $d_i = -h_i$ for all $i \in V$, it then follows from Lemma 2.1 that an HLF (resp. LRP) schedule coincides with an EDD (resp. SLF) schedule. Since the maximum lateness and the makespan are identical in such a case (cf. Lemma 3.1), an application of Theorem 3.1 yields the desired result. □

Corollary 3.1 states that if for any expansion of G, there is an HLF list algorithm which minimizes the makespan within the class of nonpreemptive schedules, then the priority schedule LRP is optimal within the class of preemptive schedules.

Remark. Theorem 3.1 and Corollary 3.1 actually hold in a more general framework where the tasks are associated with release dates: a task is executable only after its release date. However, there are few applications with nonzero release dates so that they will not be considered in the paper.

We now get rid of the commensurability assumption and prove the following results for RET tasks.

Theorem 3.2. *Let* $\mathscr{M}$ *be a class of profiles which is closed under translation and* $\mathscr{C}$ *be a class of graphs which is closed under expansion. If for any* $M \in \mathscr{M}$ *with integer profile changing epochs and for any* $G \in \mathscr{C}$ *with UET tasks and integer due dates, there exists an*

EDD schedule minimizing the maximum lateness of G within the class of nonpreemptive policies, then for any $M \in \mathcal{M}$ and any $G \in \mathcal{C}$, the SLF schedule minimizes the maximum lateness of G within the class of preremptive schedules.

Proof. The proof is provided in Appendix B. □

As a consequence (cf. the proof of Corollary 3.1), we obtain the following result:

Corollary 3.2. *Let $\mathcal{M}$ be a class of profiles which is closed under translation and $\mathcal{C}$ a class of graphs which is closed under expansion. If for any $M \in \mathcal{M}$ with integer profile changing epochs and for any $G \in \mathcal{C}$ with UET tasks, there exists an HLF schedule minimizing the makespan of G within the class of nonpreemptive policies, then for any $M \in \mathcal{M}$ and any $G \in \mathcal{C}$, the LRP schedule minimizes the makespan of G within the class of preemptive schedules.*

In the remainder of this paper, we will apply these results to four profile scheduling problems. New optimality results for SLF and LRP schedules are obtained.

4. Applications

4.1. Maximum lateness of in-forests

We first apply our results to the class of *in-forests* $\mathcal{G}_{\text{if}}$ with *increasing zigzag* profiles $\mathcal{M}_{\text{iz}}$. A task graph $G = (V, E)$ is an in-forest, $G \in \mathcal{G}_{\text{if}}$, if $|\sigma(i)| \leqslant 1$ for all $i \in V$, i.e. a task has at most one successor. When G is an in-forest, the final tasks are also referred to as the roots, and the initial tasks as the leaves. A profile M is increasing zigzag, $M \in \mathcal{M}_{\text{iz}}$, if for all $r \in \mathbb{N}_+$, $m_r \geqslant \max_{1 \leqslant u \leqslant r-1} m_u - 1$.

For a given in-forest $G \in \mathcal{G}_{\text{if}}$ with processing times $p_1, \ldots, p_n$ and due dates $d_1, \ldots, d_n$, we define an in-forest $G' \in \mathcal{G}_{\text{if}}$ such that G' has the same set of tasks, the same precedence constraints and the same processing times. The due dates in G' are modified as follows:

$$d_i' = \begin{cases} d_i, & i \text{ is a root;} \\ \min(d_i, d'_{\sigma(i)} - p_{\sigma(i)}), & \text{otherwise,} \end{cases}$$

where, with a harmless abuse of notation, $\sigma(i)$ denotes the successor of $i \in G$ when i is not a root of G. Such modification on the due dates has no effect on the maximization lateness.

Lemma 4.1. *Let $G \in \mathcal{G}_{\text{if}}$ be an in-forest. Then for any feasible (preemptive or nonpreemptive) schedule S,*

$$L_S(G, M) = L_S(G', M) \stackrel{\text{def}}{=} \max_{i \in V} (C_i(S) - d_i').$$

Proof. Consider a given schedule S on (G, M). It is clear that $d_i \geqslant d_i'$ for all $i \in V$ so that

$$L_S(G, M) \leqslant L_S(G', M).$$

Let k be a task such that $C_k(S) - d_k' = L_S(G', M)$ and that for all $j \in \sigma^*(k)$, $C_j(S) - d_j' < L_S(G', M)$. If $d_k > d_k'$, then k has a successor $s \in \sigma(k)$ such that $d_k' = d_s' - p_s < d_k$. Thus,

$$C_s(S) - d_s' \geqslant C_k(S) + p_s - d_s' = C_k(S) - d_k' = L_S(G', M).$$

This contradicts the assumption that $C_j(S) - d_j' < L_S(G', M)$ for all $j \in \sigma^*(k)$. Therefore, we have necessarily $d_k = d_k'$ so that

$$L_S(G, M) \geqslant C_k(S) - d_k = C_k(S) - d_k' = L_S(G', M).$$

The proof is thus completed. □

In [1], it was shown that in the case of contant profile and UET tasks, the EDD schedules defined on the modified due dates $\mathscr{E}(G', M)$ are optimal for the minimization of the maximum lateness of in-forests within the class of nonpreemptive schedules. Such a result is extended to the increasing zigzag profiles below.

Lemma 4.2 (Extension of Theorem 2 of Brucker et al. [1]). *Let $G \in \mathscr{G}_{\text{if}}$ be an in-forest, and $M \in \mathscr{M}_{\text{iz}}$ be an increasing zigzag profile. Assume that the tasks are UET and that the profile changing epochs are integer. Then any EDD schedule $S \in \mathscr{E}(G', M)$ defined on the modified due dates minimizes the maximum lateness within the class of nonpreemptive schedules*:

$$L_S(G, M) = L^*_{\text{np}}(G, M).$$

Proof. See Appendix C. □

In order to get optimal preemptive schedules, we have to show that the due dates modified before and after an expansion are the same. Assume that $G \in \mathscr{G}_{\text{if}}$ has commensurable timing with commensurability factor w. Denote by

- $(G')_{w/k}$: the kth expansion of G' which has the modified due dates of G,
- $(G_{w/k})'$: the task graph with the modified due dates of the kth expansion $G_{w/k}$ of G.

Lemma 4.3. *Assume that $G \in \mathscr{G}_{\text{if}}$ has commensurable timing with commensurability factor w. Then for all $k \in \mathbb{N}_+$, $(G')_{w/k}$ and $(G_{w/k})'$ are isomorphic in the sense that their topologies are isomorphic and that the corresponding tasks have the same processing times and the same due dates.*

Proof. Clearly, $(G')_{w/k}$ and $(G_{w/k})'$ have the same set of tasks and the same structure as $G_{w/k}$, and the processing times of all the tasks are equal to w/k. Let the tasks in $G_{w/k}$ be indexed in such a way that $i_1, i_2, \ldots, i_{\alpha_i k}$ are the subtasks of $i \in G$, where α_i is an integer, and i_j is the predecessor of i_{j+1}, $1 \leqslant j \leqslant \alpha_i k - 1$. Let d'_{i_j} be the due date of task i_j in $(G')_{w/k}$ and $(d_{i_j})'$ the due date of task i_j in $(G_{w/k})'$. We will show by induction that for all $i \in G$,

$$d'_{i_j} = (d_{i_j})', \quad 1 \leqslant j \leqslant \alpha_i k. \tag{3}$$

First, for all the roots $i \in G$, we have that

$$d'_{i_{\alpha_i k}} = d_i = (d_{i_{\alpha_i k}})',$$

so that

$$(d_{i_j})' = d_i - (\alpha_i k - j) w/k = d'_{i_j}, \quad 1 \leqslant j \leqslant \alpha_i k.$$

Consider now task i such that (3) holds for $s \in \sigma(i)$. It then follows

$$\begin{aligned}(d_{i_{\alpha_i k}})' &= \min(d_i, (d_{s_1})' - w/k)\\ &= \min(d_i, d'_{s_1} - w/k)\\ &= \min(d_i, d'_s - (\alpha_s k - 1) w/k - w/k)\\ &= \min(d_i, d'_s - p_s)\\ &= d'_i = d'_{i_{\alpha_i k}}.\end{aligned}$$

Assume now that for some $2 \leqslant j \leqslant \alpha_i k$, $(d_{i_j})' = d'_{i_j}$. Then

$$\begin{aligned}(d_{i_{j-1}})' &= \min(d_{i_{j-1}}, (d_{i_j})' - w/k)\\ &= \min(d_i - (\alpha_i k - j + 1) w/k, d'_{i_j} - w/k)\\ &= \min(d_i, d'_i) - (\alpha_i k - j + 1) w/k\\ &= d'_i - (\alpha_i k - j + 1) w/k = d'_{i_j}.\end{aligned}$$

Therefore, by induction, we have shown that for all $1 \leqslant j \leqslant \alpha_i k$, $(d_{i_j})' = d'_{i_j}$, so that (3) holds for task i. Hence, (3) holds for all $i \in G$. $\square$

Theorem 4.1. *Let $G \in \mathscr{G}_{\mathrm{if}}$ be an in-forest, and $M \in \mathscr{M}_{\mathrm{iz}}$ an increasing zigzag profile. Then the SLF′ schedule defined on the modified due dates minimizes the maximum lateness within the class of preemptive schedules*:

$$L_{\mathrm{SLF}'}(G, M) = L^*_{\mathrm{p}}(G, M).$$

Proof. Assume first that (G, M) has commensurable timing. Consider the task graph G' with the modified due dates of G. Since the class of in-forests $\mathscr{G}_{\mathrm{if}}$ is closed under expansion, an application of Lemma 4.2 yields that for all $k \in \mathbb{N}_+$, and all $S \in \mathscr{E}((G_{w/k})', M)$,

$$L_S((G_{w/k})', M) = L^*_{\mathrm{np}}((G_{w/k})', M).$$

Owing to Lemma 4.3, $(G_{w/k})'$ and $G'_{w/k}$ are isomorphic so that $\mathscr{E}((G_{w/k})', M) = \mathscr{E}(G'_{w/k}, M)$ and that for all $k \in \mathbb{N}_+$, and all $S \in \mathscr{E}(G'_{w/k}, M)$,

$$L_S(G'_{w/k}, M) = L^*_{\text{np}}(G'_{w/k}, M).$$

Applying now Theorem 3.1 implies that

$$L_{\text{SLF}}(G', M) = L^*_{\text{p}}(G', M).$$

The above relation together with Lemma 4.1 entail

$$L_{\text{SLF}'}(G, M) = L_{\text{SLF}}(G', M) = L^*_{\text{p}}(G', M) = L^*_{\text{p}}(G, M).$$

Note that $\mathscr{G}_{\text{if}}$ is closed under expansion, and that $\mathscr{M}_{\text{iz}}$ is closed under translation. Using the same argument as in the proof of Theorem 3.2, we can conclude that the above relation holds without the commensurability assumption. □

Note that due to the modification of due dates, we cannot directly apply Theorem 3.2 in order to get the assertion of Theorem 4.1.

Remark. Theorem 4.1 extends Theorem 7.3 of Lawler [8] to the case of increasing zigzag variable profile. It is possible to apply Theorem 3.1 to the case of an arbitrary task graph and constant profile with two processors. In such a case, a new proof of Theorem 8.3 of Lawler [8] can be obtained in the case of parallel processors.

4.2. Makespan of chains

Consider the makespan minimization problem. We first analyze the simplest case of the task graphs: the chains. Let $\mathscr{G}_{\text{ch}}$ be the class of unions of chains. A task graph $G \in \mathscr{G}_{\text{ch}}$ is a union of chains if for all $i \in V$, $|\sigma(i)| \leqslant 1$ and $|\pi(i)| \leqslant 1$.

Lemma 4.4. *Let $G \in \mathscr{G}_{\text{ch}}$ be a union of chains with UET tasks, and M be a profile that changes only at integer time epochs. Then every HLF schedule $S \in \mathscr{H}(G, M)$ minimizes the makespan within the class of nonpreemptive schedules*:

$$C_{\text{HLF}}(G, M) = C^*_{\text{np}}(G, M).$$

Proof. Observe first that all HLF schedules have the same makespan, for the task graph is a union of chains. In order to prove the lemma, we only need to show that there is at least one optimal HLF schedule.

Let $b_i(S)$ denote the time instant when task i is assigned for execution under a nonpreemptive schedule S (so that $C_i(S) = b_i(S) + 1$). Consider an optimal schedule S^* which yields the minimal makespan $C^*_{\text{np}}(G, M)$. If S^* is not a HLF type schedule, then there are at least two tasks i and j such that i is at a higher level than j and that i is assigned after j, viz. $h_i > h_j$ and $b_i(S^*) > b_j(S^*)$.

Let $\sigma^*(i) = \{i_1, i_2, \ldots, i_{h_i}\}$ and $\sigma^*(j) = \{j_1, j_2, \ldots, j_{h_j}\}$ be the sets of (not necessarily immediate) successors of i and j, respectively. Let $i_0 = i$ and $j_0 = j$. Assume that $i_u \in \pi(i_{u+1})$, $0 \leqslant u \leqslant h_i - 1$, and that $j_v \in \pi(j_{v+1})$, $0 \leqslant v \leqslant h_j - 1$. Let k be the largest integer in $\{0, 1, \ldots, h_j\}$ such that for all $0 \leqslant u \leqslant k$, $b_{i_u}(S^*) > b_{j_u}(S^*)$.

Construct a schedule S which differs from S^* only in the assignments of tasks $i_0, i_1, \ldots, i_k$ and $j_0, j_1, \ldots, j_k$. In S, the assignments of S^* for tasks i_u and j_u, $0 \leqslant u \leqslant k$, are interchanged. It is easy to see that S is a feasible schedule and has the same makespan as S^*. Further, schedule S has at least one less non-HLF decision. If S is still not HLF, then we repeat this interchange procedure on S to reduce the number of non-HLF decisions. After at most n^2 steps of interchange, we will finally obtain an HLF schedule which has the optimal makespan $C^*_{\mathrm{np}}(G, M)$. □

Remark. Lemma 4.4 still holds when the chains have nonzero release dates. The interchange argument of the proof remains valid.

Theorem 4.2. *Let $G \in \mathscr{G}_{\mathrm{ch}}$ be a union of chains. Then the LRP schedule minimizes the makespan within the class of preemptive schedules:*

$$C_{\mathrm{LRP}}(G, M) = C^*_{\mathrm{p}}(G, M).$$

Proof. It is clear that the class of unions of chains $\mathscr{G}_{\mathrm{ch}}$ is closed under expansion. An application of Corollary 3.2 and Lemma 4.4 implies the result. □

Note that in the preemptive case, scheduling problems for a union of disjoint chains and for a set of independent tasks are equivalent.

4.3. Makespan of forests

We next study the minimization of the makespan of forests. Apart from the class of in-forests $\mathscr{G}_{\mathrm{if}}$, we will also consider the class of *out-forests* $\mathscr{G}_{\mathrm{of}}$. A task graph $G = (V, E)$ is an out-forest, $G \in \mathscr{G}_{\mathrm{of}}$, if $|\pi(i)| \leqslant 1$ for all $i \in V$, i.e. a task has at most one predecessor. Clearly, the class of out-forests $\mathscr{G}_{\mathrm{of}}$ is closed under expansion. When G is an out-forest, the initial tasks are also referred to as the roots, and the final tasks as the leaves.

The out-forests will be scheduled with the class of *decreasing zigzag* profiles $\mathscr{M}_{\mathrm{dz}}$. A profile M is decreasing zigzag, $M \in \mathscr{M}_{\mathrm{dz}}$, if for all $r \in \mathbb{N}_+$, $m_r \leqslant \min_{1 \leqslant u \leqslant r-1} m_u + 1$.

Consider first the nonpreemptive schedules. The following lemma is due to Dolev and Warmuth [5].

Lemma 4.5 (Theorems 5.1 and 5.2 of Dolev and Warmuth [5]). *Let $(G, M) \in (\mathscr{G}_{\mathrm{if}}, \mathscr{M}_{\mathrm{iz}}) \cup (\mathscr{G}_{\mathrm{of}}, \mathscr{M}_{\mathrm{dz}})$ be either an in-forest with increasing zigzag profile or an out-forest with decreasing zigzag profile. Assume that all the tasks have UET and that* *the profile changes at integer time epochs. Then every HLF schedule $S \in \mathscr{H}(G, M)$ minimizes the makespan within the class of nonpreemptive schedules:*

$$C_{\mathrm{HLF}}(G, M) = C^*_{\mathrm{np}}(G, M).$$

Theorem 4.3. *Let $(G, M) \in (\mathscr{G}_{\mathrm{if}}, \mathscr{M}_{\mathrm{iz}}) \cup (\mathscr{G}_{\mathrm{of}}, \mathscr{M}_{\mathrm{dz}})$ be either an in-forest with increasing zigzag profile or an out-forest with decreasing zigzag profile. The LRP schedule minimizes the makespan within the class of preemptive schedules:*

$$C_{\mathrm{LRP}}(G, M) = C^*_{\mathrm{p}}(G, M).$$

Proof. Note that both the classes of in-forests and out-forests are closed under expansion. Similarly, the classes of profiles $\mathscr{M}_{\mathrm{iz}}$ and $\mathscr{M}_{\mathrm{dz}}$ are closed under translation. Thus, by applying Corollary 3.2 and Lemma 4.5 we obtain the desired assertion. □

Remark. Theorem 4.3 extends a result of Muntz and Coffman [13] to the zigzag variable profiles. The optimality of an LRP schedule for the makespan minimization of in-forests with increasing zigzag profiles may also be obtained from Theorem 4.1.

4.4. Makespan of arbitrary task graphs

Consider now the makespan minimization of arbitrary task graphs. In [3], Coffman and Graham proved that if there are constantly two processors and if the tasks have UET, then some special HLF schedules, referred to as the CG schedules in our paper, minimize the makespan within the class of nonpreemptive schedules. In the CG schedules, the list of tasks is determined by a lexicographical order on the sets of successors. Such a result still holds when the profile is bounded by two, i.e. $m_r \leqslant 2$ for all $r \in \mathbb{N}_+$.

Lemma 4.6 (Extension of Theorem 1 of Coffman and Graham [3]). *Let G be an arbitrary task graph with UET tasks, and M be a profile which is bounded by two and changes at integer time epochs. Then every CG schedule minimizes the makespan within the class of nonpreemptive schedules:*

$$C_{\mathrm{CG}}(G, M) = C^*_{\mathrm{np}}(G, M).$$

Proof. The result can be shown by mimicking the proof of [3]. The generalization is straightforward by simply adding dummy tasks when a machine is unavailable during some time period. See [14] for details. □

Theorem 4.4. *Let G be an arbitrary task graph and M be a profile bounded by two. The LRP schedule minimizes the makespan within the class of preemptive schedules:*

$$C_{\mathrm{LRP}}(G, M) = C^*_{\mathrm{p}}(G, M).$$

Proof. Since the CG schedules form a subclass of HLF schedules, an application of Corollary 3.2 yields the desired result. □

Remark. Theorem 4.4 extends a result of Muntz and Coffman [12] to variable profiles bounded by two.

Appendix A. Proof of Theorem 3.1

The scheme of our proof is similar to that of [13]. We first establish some intermediary results.

Lemma A.1. *Assume that* (G, M) *has commensurable timing with commensurability factor* w. *Then there is a constant* Δ *such that*

$$\forall k \in \mathbb{N}_+: \quad L_{\mathrm{p}}^*(G, M) \leqslant L_{\mathrm{np}}^*(G_{w/k}, M) \leqslant L_{\mathrm{p}}^*(G, M) + \Delta/k.$$

Proof. The first inequality comes from the fact

$$L_{\mathrm{p}}^*(G, M) = L_{\mathrm{p}}^*(G_{w/k}, M) \leqslant L_{\mathrm{np}}^*(G_{w/k}, M),$$

where we used Lemma 2.3 for the first equality. We now prove the second inequality.

Let S^* be an optimal preemptive schedule, and $C_1, \ldots, C_n$ be the completion times of the tasks of G. Without loss of generality, we assume that the tasks of G are labeled in such a way that $C_1 \leqslant \cdots \leqslant C_n$. Let $C_0 = 0$.

For $j = 1, \ldots, n$, let V_j denote the set of tasks that are assigned for execution in the time interval $\theta_j = [C_{j-1}, C_j)$. Note that all the tasks in V_j are enabled at time C_{j-1} so that there is no precedence relation between these tasks. Clearly, $|V_j| \leqslant n$.

For a given time interval θ_j, we define the *assigned pieces* (i, μ, t^1, t^2), $i \in V_j$, $1 \leqslant \mu \leqslant m$, such that task i is continuously executed by processor μ during $[t^1, t^2)$, where $C_{j-1} \leqslant t^1 < t^2 \leqslant C_j$. Furthermore, there is $\varepsilon > 0$ such that the task i is assigned to processor μ neither during $[t^1 - \varepsilon, t^1)$, nor during $[t^2, t^2 + \varepsilon)$. The quantity $t^2 - t^1$ is referred to as the length of the assigned pieces.

By hypothesis, the number of profile changes during each interval θ_j is bounded by $\bar{r}$. It is possible to transform S^* so that the number of total assigned pieces in each time interval is bounded. Such a transformation can be obtained by for instance considering the tasks in V_j one by one. A task i is executed by an available processor continuously until the profile changes or the total amount of executions of i in S^* during θ_j is reached. Under such a transformation, the number of preemptions in each time interval is bounded by $mn\bar{r}$. Hence, suppose the number of total assigned pieces in S^* is bounded by B in each time interval θ_j, $j = 1, 2, \ldots, n$.

We want to construct a nonpreemptive (possibly idling) schedule S_k for $G_{w/k}$ such that the relation

$$L_{S_k}(G_{w/k}, M) \leqslant L_{S^*}(G, M) + \Delta/k$$

holds for some constant Δ independent of k. For this purpose, we first construct an intermediate (possibly nonfeasible) schedule S'. This schedule is constructed by modifying S^* so that each of its assigned pieces has a length multiple of w/k.

Let us consider the first time interval $\theta_1 = [0, C_1)$. Schedule S' is constructed from S^* by cutting a small portion of execution time from each assigned piece so that its length becomes a multiple of w/k. At time C_1, an assigned piece of length w/k is added sequentially on processor 1 (which is assumed to be always available) for each of the assigned pieces. To be more precise, let there be $B_1 \leqslant B$ assigned pieces in θ_1: $(v_i, \mu_i, t_i^1, t_i^2)$, $1 \leqslant i \leqslant B_1$, $v_i \in V$. In S', each assigned piece $(v_i, \mu_i, t_i^1, t_i^2)$, of S^* is replaced by two pieces $(v_i, \mu_i, t_i^1, t_i^1 + \lfloor (t_i^2 - t_i^1)k/w \rfloor w/k)$ and $(v_i, 1, C_1 + (i-1)w/k, C_1 + iw/k)$. Processor μ_i is idle during the time interval $[t_i^1 + \lfloor (t_i^2 - t_i^1)k/w \rfloor w/k, t_i^2)$. Let

$$C_1' \stackrel{\text{def}}{=} C_1 + B_1 w/k < C_1 + (B+m)w/k.$$

Clearly, in the time interval $[0, C_1')$, the lengths of all the assigned pieces of S' are integer multiples of w/k, and S' finishes all the amounts of executions of the tasks in S^* during the time interval $[0, C_1)$.

Assume that for some $j \geqslant 1$, in the time interval $[C_{j-1}, C_j')$, the lengths of all the assigned pieces of S' are integer multiples of w/k, and that S' finishes all the amounts of executions of the tasks in S^* during the time interval $[C_{j-1}, C_j)$.

Consider now the assigned pieces of S^* in time interval θ_{j+1}. There are two cases: $C_j \leqslant C_j' \leqslant C_{j+1}$ or $C_j' > C_{j+1}$.

Assume first $C_j \leqslant C_j' \leqslant C_{j+1}$. Let there be $B_{j+1} \leqslant B$ assigned pieces in θ_{j+1}. We split the $\bar{m}$ aassigned pieces (v, μ, t^1, t^2) of S^* such that $t' < C_j' < t^2$ into two pieces (v, μ, t^1, C_j') and (v, μ, C_j', t^2). Let the $B_{j+1} + \bar{m}$ assigned pieces $(v_i, \mu_i, t_i^1, t_i^2)$, $1 \leqslant i \leqslant B_{i+1} + \bar{m}$, be ordered in such a way that $t_i^1 \geqslant C_j'$ holds for all $1 \leqslant i \leqslant B' \leqslant B_{j+1} + \bar{m}$, and that $t_i^2 \leqslant C_j'$ holds for all $B' < i \leqslant B_{j+1} + \bar{m}$. Construct S' as follows:

- For the assigned pieces $(v_i, \mu_i, t_i^1, t_i^2)$ of S^* with $i \leqslant B'$, we apply the same procedure as in the case $j = 1$, viz. the assigned piece $(v_i, \mu_i, t_i^1, t_i^2)$ of S^* is replaced by two pieces $(v_i, \mu_i, t_i^1, t_i^1 + \lfloor (t_i^2 - t_i^1)k/w \rfloor w/k)$ and $(v_i, 1, C_{j+1} + (i-1)w/k, C_{j+1} + iw/k)$ in S'.
- For the assigned pieces $(v_i, \mu_i, t_i^1, t_i^2)$ of S^* with $i > B'$, we slightly increase their lengths so that they become integer multiples of w/k and then sequentially assign them at processor 1 at time $C_{j+1} + B'w/k$, i.e. the assigned piece $(v_i, \mu_i, t_i^1, t_i^2)$ of S^* is replaced by $(v_i, 1, t_i', t_i'')$ in S', where

$$t_i' = C_{j+1} + B'w/k + \sum_{u=B'+1}^{i-1} \lfloor (t_u^2 - t_u^1)k/w + 1 \rfloor w/k,$$

$$t_i'' = t_i' + \lfloor (t_i^2 - t_i^1)k/w + 1 \rfloor w/k.$$

Let $C_{j+1}' = t''_{B_{j+1}+\bar{m}}$. It is easily verified that in the time interval $[C_j', C_{j+1}')$ the lengths of all the assigned pieces of S' are integer multiples of w/k, and that S' finishes all the amounts of executions of the tasks in S^* during the time interval $[C_j, C_{j+1})$. Note that

$$\begin{aligned} t''_{B_{j+1}+\bar{m}} &\leqslant C_{j+1} + B'w/k + m(C'_j - C_j) + (B_{j+1} + \bar{m} - B')w/k \\ &\leqslant C_{j+1} + m(C'_j - C_j) + (B + m)w/k. \end{aligned}$$

Therefore,

$$C'_{j+1} - C_{j+1} \leqslant m(C'_j - C_j) + (B + m)w/k.$$

Assume now $C'_j > C_{j+1}$. Let there be $B_{j+1} \leqslant B$ assigned pieces in θ_{j+1}: $(v_i, \mu_i, t_i^1, t_i^2)$, $1 \leqslant i \leqslant B_{j+1}$. We slightly increase their lengths so that they become integer multiples of w/k and then sequentially assign them at processor 1 at time C'_j, i.e. the assigned piece $(v_i, \mu_i, t_i^1, t_i^2)$ of S^* is replaced by $(v_i, 1, t'_i, t''_i)$ in S', where

$$t'_i = C'_j + \sum_{u=1}^{i-1} \lfloor (t_u^2 - t_u^1)k/w + 1 \rfloor w/k,$$

$$t''_i = t'_i + \lfloor (t_i^2 - t_i^1)k/w + 1 \rfloor w/k.$$

Let $C'_{j+1} = t''_{B_{j+1}}$. As in the previous case, it is easily verified that in the time interval $[C'_j, C'_{j+1})$ the lengths of all the assigned pieces of S' are integer multiples of w/k and that S' finishes all the amounts of executions of the tasks in S^* during the time interval $[C_j, C_{j+1})$. Since

$$\begin{aligned} C'_{j+1} &= t''_{B_{j+1}} \\ &\leqslant C'_j + B_{j+1} w/k + m(C_{j+1} - C_j) \\ &\leqslant C'_j + (B + m)w/k + C_{j+1} + (m-1)C_{j+1} - mC_j \\ &\leqslant C'_j + (B + m)w/k + C_{j+1} + (m-1)C'_j - mC_j, \end{aligned}$$

we obtain that

$$C'_{j+1} - C_{j+1} \leqslant m(C'_j - C_j) + (B + m)w/k.$$

This construction is continued until $j = n$ so that a complete schedule S' is generated. By induction, the lengths of all the assigned pieces of S' are integer multiples of w/k, and for all $i \in V$, the total execution time of task i is at least p_i. Furthermore, an easy computation yields

$$\forall\, 1 \leqslant j \leqslant n, \quad C'_j - C_j \leqslant \frac{m^j - 1}{m - 1}(B + m)w/k \leqslant \frac{\Delta}{k}, \tag{4}$$

where

$$\Delta = \frac{m^n - 1}{m - 1}(B + m)w. \tag{5}$$

The schedule S_k is now defined for the couple $(G_{w/k}, M)$ as follows: task i_j is running at time t if and only if the jth portion of length w/k of task i is running at time t under schedule S', where $1 \leqslant j \leqslant \alpha_i k$, and i_j is the jth task in the chain that replaces task i in the expansion of G to obtain $G_{w/k}$. It is easy to see that for all $i \in V$,

$$C_{i_{\alpha_i k}}(S_k) \leqslant C_i' \leqslant C_i + \Delta/k.$$

As S_k can be considered as the expansion of a schedule S'' obtained by restricting S' to the first $\alpha_i k$ portions of length w/k of each task $i \in V$, an application of Lemma 2.2 implies that

$$\begin{aligned} L_{S_k}(G_{w/k}, M) = L_{S''}(G, M) &= \max_{i \in V}(C_{i_{\alpha_i k}}(S_k) - d_i) \leqslant \max_{i \in V}(C_i - d_i) + \Delta/k \\ &= L_{\mathrm{p}}^*(G, M) + \Delta/k. \end{aligned}$$

Therefore,

$$L_{\mathrm{np}}^*(G_{w/k}, M) \leqslant L_{S_k}(G_{w/k}, M) \leqslant L_{\mathrm{p}}^*(G, M) + \Delta/k, \tag{6}$$

which completes the proof. □

Recall that in a priority scheduling algorithm, there are three types of events: (1) a task is completed; (2) the priority of one subset becomes the same as another; (3) the profile changes. Let t_1 be the time epoch when the first event occurs in a priority schedule S applied to (G, M). Denote by $G_{t_1}(S)$ the remaining graph of G at time t_1 under the schedule S, where the processing times of the tasks of $G_{t_1}(S)$ are the remaining processing times of tasks in G at time t_1 under the schedule S. The due dates remain unchanged. Similarly, denote by $M_{t_1}(S) = \{a_r^1(S), \mathrm{m}_r^1(S)\}_{r=1}^{\bar{r}}$ the remaining profile, where

$$a_1^1(S) = 0, \quad a_r^1(S) = a_{r + \mathbf{1}(t_1 = a_2)} - t_1, \quad r \geqslant 2,$$

$$m_r^1(S) = m_{r + \mathbf{1}(t_1 = a_2)}, \quad r \geqslant 1,$$

where $\mathbf{1}(\bullet)$ is the indicator function: $\mathbf{1}(t_1 = a_2) = 1$ if $t_1 = a_2$, and $\mathbf{1}(t_1 = a_2) = 0$ otherwise.

Lemma A.2. *Assume that (G, M) has commensurable timing with commensurability factor w. Let $t_1 > 0$ be the first time epoch when an event occurs in the SLF schedule. Then there exists an integer $\gamma_1 \in \mathbb{N}_+$ such that:*

(i) *t_1 is an integer multiple of $w_1 = w/\gamma_1$;*

(ii) *the remaining couple $(G_{t_1}(\mathrm{SLF}), M_{t_1}(\mathrm{SLF}))$ has commensurable timing with commensurability factor w_1;*

(iii) *for all $k \in \mathbb{N}_+$, and for all EDD schedule $S \in \mathscr{E}(G_{w_1/k}, M)$, the remaining task graph of $G_{w_1/k}$ at t_1 in schedule S, denoted by $G' = (G_{w_1/k})_{t_1}(S)$, is isomorphic to $(G_{t_1}(\mathrm{SLF}))_{w_1/k}$ in the sense that both d.a.g.s are isomorphic, and that the corresponding tasks have the same processing times and the same due dates.*

Proof. Let $V_1, V_2, \ldots, V_u$ be the subsets of the tasks that are assigned for execution under the SLF schedule. Assume that laxities of the tasks in the same subset are identical and that the tasks in V_i have (strictly) smaller laxity than those in V_{i+1}, $i = 1, \ldots, u-1$.

If $|V_1| + |V_2| + \cdots + |V_u| \leqslant m_1$ (recall that m_1 is the number of available processors at time 0), then all these tasks are executed at speed 1. In this case, we define $\gamma_1 = 1$. Otherwise, $|V_1| + |V_2| + \cdots + |V_u| > m_1$. Let $a = m_1 - (|V_1| + |V_2| + \cdots + |V_{u-1}|)$ and $b = |V_u|$. According to the definition of priority schedules, we have $1 \leqslant a < b$. In this case, the tasks in $V_1, V_2, \ldots, V_{u-1}$ are executed at speed 1 and those in V_u at speed a/b. Define $\gamma_1 = ab(b-a)$.

(i) Let $q_i \in \{0, 1, a/b\}$ be the speed of task $i \in V$, where $q_i = 0$ if task i is not assigned for execution. There are three possible cases:

- If at time t_1, an event of type 1 occurs, then there is a task i (with $q_i > 0$) which completes at time t_1. Thus $t_1 = p_i/q_i$.
- If an event of type 2 occurs at time t_1, then at least two tasks i and j which had different laxities at time 0 become of the same priority at time t_1:

$$d_i - (p_i - q_i t_1) = d_j - (p_j - q_j t_1).$$

Note that this is possible only if $(q_i - q_j)[(d_i - p_i) - (d_j - p_j)] < 0$, i.e. the speed of the task with smaller laxity at time 0 is (strictly) greater than that of the task with larger laxity at time 0. Therefore,

$$t_1 = \frac{1}{q_i - q_j}[(d_j - p_j) - (d_i - p_i)].$$

- If now the event is of type 3, then $t_1 = a_2$.

Suppose $|V_1| + |V_2| + \cdots + |V_u| \leqslant m_1$. Due to the fact that (G, M) has commensurable timing with commensurability factor w, we obtain that in all these three cases, t_1 is an integer multiple of $w_1 \stackrel{\text{def}}{=} w/\gamma_1 = w$. Otherwise, t_1 is an integer multiplier of bw_1.

(ii) By definition, γ_1 is an integer multiple of the speed q_i, $i \in V$. Using the fact that (G, M) has commensurability factor w (which implies the commensurability of factor w/γ_1), one readily gets that all the remaining processing times in the task graph $G_{t_1}(\mathrm{SLF})$ are integer multiples of w_1. Using further the fact that t_1 is an integer multiple of w_1 implies that all the profile changing epochs of $M_{t_1}(\mathrm{SLF})$ are integer multiples of w_1. Therefore, the remaining couple $(G_{t_1}(\mathrm{SLF}), M_{t_1}(\mathrm{SLF}))$ has commensurable timing with commensurability factor w_1.

(iii) Since G has commensurability factor w_1, we assume that $p_i = \alpha_i w_1$, where α_i is an integer, $i \in V$. Consider the expansion $G_{w_1/k} = (V_{w_1/k}, E_{w_1/k})$, where the subtasks of $i \in V$ are indexed by $i_1, i_2, \ldots, i_{\alpha_i k}$. For all $i \in V$, let $T_i = \{i_1, i_2, \ldots, i_{\alpha_i k}\}$.

Owing to Lemma 2.4, we have that the expanded graph of $G_{t_1}(\mathrm{SLF})$, denoted by $(G_{t_1}(\mathrm{SLF}))_{w_1/k}$, is identical to the remaining graph $(G_{w_1/k})_{t_1}(\mathrm{SLF})$ of the expanded graph $G_{w_1/k}$ at time t_1 under the SLF schedule. Therefore, we only have to show the isomorphism between G' and the remaining graph $(G_{w_1/k})_{t_1}(\mathrm{SLF})$ of $G_{w_1/k}$ at time t_1 under the schedule SLF.

For all $i \in V$, let $T_i(S) \subseteq T_i$ (resp. $T_i(\mathrm{SLF}) \subseteq T_i$) be the set of tasks of T_i that remain at time t_1 in the EDD schedule S (resp. SLF schedule). As the remaining processing times in $G_{t_1}(\mathrm{SLF})$ are integer multiples of w_1, it suffices to prove that for all $i \in V$, $T_i(S) = T_i(\mathrm{SLF})$, or simply $|T_i(S)| = |T_i(\mathrm{SLF})|$.

Consider the subsets of tasks of G that are assigned for execution under SLF: $V_1, V_2, \ldots, V_u$. Assume that $|V_1| + |V_2| + \cdots + |V_u| > m_1$ so that $\gamma_1 = ab(b-a)$. (The case $|V_1| + |V_2| + \cdots + |V_u| = m_1$ can be shown analogously and is thus omitted). Hence in the SLF schedule, the tasks in $V_1, V_2, \ldots, V_{u-1}$ are executed at speed 1 and those in V_u at speed a/b. It is shown in (i) that t_1 is an integer multiple of bw_1 say $t_1 = \eta b w_1$, where η is an integer. It then follows that

$$|T_i(\mathrm{SLF})| = \begin{cases} (p_i - t_1)k/w_1 = (\alpha_i - \eta b)k, & i \in V_1 \cup \cdots \cup V_{u-1}, \\ (p_i - t_1 a/b)k/w_1 = (\alpha_i - \eta a)k, & i \in V_u, \\ p_i k/w_1 = \alpha_i k, & i \in V - (V_1 \cup \cdots \cup V_u). \end{cases} \tag{7}$$

Consider now the nonpreemptive EDD schedule S for $G_{w_1/k}$. Since all the tasks in $G_{w_1/k}$ have the same processing time (w_1/k), a task i_j has smaller laxity than task $i'_{j'}$ if and only if i_j has smaller due date than task $i'_{j'}$. Furthermore, for all $1 \leqslant j \leqslant \eta b k = t_1 k / w_1$,

- task i_j has the same due date as i'_j for all $i, i' \in V_f$, $1 \leqslant f \leqslant u$;
- task i_j has (strictly) smaller due date than i'_j for all $i \in V_f$, $i' \in V_{f'}$, $1 \leqslant f < f' \leqslant u$;
- task i_j has (strictly) smaller due date than $i'_{j'}$ for all $i \in V_1 \cup \cdots \cup V_u$, $i' \in V - (V_1 \cup \cdots \cup V_u)$, $j' \geqslant 1$.

Therefore, in nonpreemptive EDD schedule S for $G_{w_1/k}$, during each of the first $\eta b k = t_1 k/w_1$ time intervals of length w_1/k, one task from each of the sets T_i, $i \in V_1 \cup \cdots \cup V_{u-1}$, and a enabled tasks having the smallest due dates among those in $\bigcup_{i \in V_u} T_i$ are chosen for execution. Since there are in total $\eta b k$ intervals of length w_1/k in $[0, t_1)$, the EDD schedule S finishes $\eta b k$ tasks from each of the sets T_i, $i \in V_1 \cup \cdots \cup V_{u-1}$, and finishes $\eta a k$ tasks from each of the sets T_i, $i \in V_u$. Thus,

$$|T_i(S)| = \begin{cases} \alpha_i k - \eta b k, & i \in V_1 \cup \cdots \cup V_{u-1}, \\ \alpha_i k - \eta a k, & i \in V_u, \\ \alpha_i k, & i \in V - (V_1 \cup \cdots \cup V_u). \end{cases} \tag{8}$$

The Eqs. (7) and (8) readily imply assertion (iii). $\square$

Informally, Lemma A.2 means that if G is sufficiently "sliced", all EDD schedules behave analogously to $\mathrm{SLF}(G, M)$ during $[0, t_1)$. Using this property, we establish the following lemma.

Lemma A.3. *Assume that (G, M) has commensurable timing with commensurability factor w. Then there exists an integer γ such that for all $k \in \mathbb{N}_+$, and all EDD schedule $S_k \in \mathscr{E}(G_{w_0/k}, M)$, where $w_0 = w/\gamma$,*

$$L_{S_k}(G_{w_0}, M) = L_{\mathrm{SLF}}(G, M).$$

Proof. Observe first that in the schedule SLF(G, M), the number of events is bounded by $2n + \bar{r} - 1$. Indeed, the number of tasks, the number of subsets of tasks having the same priorities, and the number of profile changes are bounded by $n + n + \bar{r} - 1$. Whenever an event occurs in SLF(G, M), either the number of tasks is decreased by one (event type 1), or the number of subsets is decreased by one (event type 2, in which case at least two subsets merge), or the number of profile changes is decreased by one (event type 3). This bound is tight. Consider for example three independent tasks to be scheduled on one machine, with $p_1 = 4$, $d_1 = 6$, $p_2 = 2$, $d_2 = 5$ and $p_3 = 1$, $d_3 = 3.5$.

Let N and $t_1, t_2, \ldots, t_N$ be the number and the time epochs, respectively, of events in the SLF schedule for (G, M). Let $G^{(j)} = G_{t_j}(\mathrm{SLF})$ and $M^{(j)} = M_{t_j}(\mathrm{SLF})$ be the remaining graph and the remaining profile at time t_j, $1 \leqslant j \leqslant N$, where $G^{(N)} = \emptyset$.

It is readily shown by induction using Lemma A.2 that there are integers $\gamma_1, \gamma_2, \ldots, \gamma_N$ such that t_j is an integer multiple of $w/(\gamma_1 \cdots \gamma_j)$, $j = 1, 2, \ldots, N$. Let

$$\gamma = \gamma_1 \gamma_2 \cdots \gamma_N, \qquad w_0 = w/\gamma.$$

Thus, for all $k \in \mathbb{N}_+$, and all EDD schedules $S_k \in \mathscr{E}(G_{w_0/k}, M)$, an induction on $j = 1, 2, \ldots, N$ using Lemma A.2 implies that $(G_{w_0/k})_{t_j}(S_k)$ is isomorphic to $(G^{(j)})_{w_0/k}$ for all $j = 1, 2, \ldots, N$. Thus,

$$L_{S_k}(G_{w_0/k}, M) = L_{\mathrm{SLF}}(G_{w_0/k}, M) = L_{\mathrm{SLF}}(G, M),$$

where we used Lemma 2.4 for the last inequality. $\square$

We are now in a position to prove the main theorem.

Proof of Theorem 3.1. Let (G, M) satisfy the conditions of the theorem. Applying Lemma A.3 to the couple (G_w, M) implies that there is an integer γ such that for all $k \in \mathbb{N}_+$, and all EDD schedules $S_k \in \mathscr{E}(G_{w_0/k}, M)$, where $w_0 = w/\gamma$,

$$L_{S_k}(G_{w_0/k}, M) = L_{\mathrm{SLF}}(G, M).$$

Let S_k^* be the EDD schedule which is optimal within the class of nonpreemptive schedules for the minimization of maximum lateness of $(G_{w_0/k}, M)$. Then, according to Lemma A.1

$$L_{\mathrm{p}}^*(G, M) \leqslant L_{\mathrm{np}}^*(G_{w/(\gamma k)}, M) = L_{S_k^*}(G_{w_0/k}, M) = L_{\mathrm{SLF}}(G, M) \leqslant L_{\mathrm{p}}^*(G, M) + \Delta/(\gamma k).$$

Letting k tend to infinity immediately entails the desired result. $\square$

Appendix B. Proof of Theorem 3.2

Let $M = \{a_r, m_r\}_{r=1}^{\infty}$ be an arbitrary profile in $\mathscr{M}$ and $G = (V, E)$ be an arbitrary task graph in $\mathscr{C}$, such that both have arbitrary real timing. Let ε be an arbitrary strictly positive real number.

Consider the schedule obtained by the SLF policy applied to (G, M). Let $0 = t_0 < t_1 < t_2 < \cdots < t_k$ be the time epochs when the assignment decisions of SLF are made, where $t_k = C_{\mathrm{SLF}}(G, M)$. Note that k is finite, and $k \leqslant 2n + \bar{r} - 1$ (cf. the proof of Lemma A.3 in Appendix A, where the commensurability hypothesis plays no role).

Let $G_e = (V_e, E_e)$ be the expansion of G defined as follows: each task i of G is replaced by a chain of subtasks $i_1, i_2, \ldots, i_{j_i}$ such that

- i_h is the predecessor of i_{h+1}, $1 \leqslant h \leqslant j_i - 1$;
- each subtask i_h is executed in one and only one time interval $[t_u, t_{u+1})$ in the SLF(G, M) schedule, and is the only subtask of i in this interval, $1 \leqslant i \leqslant n$, $1 \leqslant h \leqslant j_i$, $0 \leqslant u \leqslant k - 1$;
- the processing time p_{i_h} is the total amount of work that subtask i_h receives in the SLF(G, M) schedule, $1 \leqslant i \leqslant n$, $1 \leqslant h \leqslant j_i$;
- the due date d_{i_h} is defined by $d_{i_h} = d_i - \sum_{u=h+1}^{j_i} p_{i_u}$, $1 \leqslant i \leqslant n$, $1 \leqslant h \leqslant j_i$.

Clearly $G_e \in \mathscr{C}$. Moreover, one can easily verify that the SLF policy applied to (G_e, M) yields the same schedule: SLF$(G_e, M) \equiv$ SLF(G, M).

For any task i in G_e, let h_i be the integer such that in the schedule SLF(G_e, M), task i is executed in the time interval $[t_{h_i}, t_{h_i+1})$. Let also $v_i \leqslant 1$ be the speed at which task $i \in G_e$ is executed in the schedule SLF(G_e, M). Note that v_i is a rational whose denominator is not greater than n, so that v_i is an integer multiple of $1/n!$. Note also that for any two tasks i and j in G_e, if $h_i < h_j$, or if $h_i = h_j$ and $v_i > v_j$, then, according to the definition of SLF policy, task j is unenabled or has a strictly greater laxity than task i during the time interval $[t_{h_i}, t_{h_i+1})$.

Define the task group $G' = (V', E')$ to be such that G' has the same set of tasks and precedence relations as G_e: $V' = V_e$, $E' = E_e$. The task graphs G' and G_e differ only in the processing times and the due dates. Let $p_i(G_e)$ and $d_i(G_e)$ (resp. $p_i(G')$ and $d_i(G')$), be the processing time and due date of task i in G_e (resp. in G'). Then the processing times and the due dates of G' are defined as follows:

$$p_i(G') = v_i \left\lceil \frac{t_{h_i+1} - t_{h_i}}{\varepsilon} \right\rceil \varepsilon, \tag{9}$$

$$d_i(G') = \left\lceil \frac{d_i(G_e)}{\varepsilon} \right\rceil \varepsilon + 2h_i\varepsilon + \lceil 1 - v_i \rceil \varepsilon, \tag{10}$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x. By definition,

$$p_i(G_e) \leqslant p_i(G') < p_i(G_e) + \varepsilon,$$

$$d_i(G_e) + 2h_i\varepsilon \leqslant d_i(G') < d_i(G_e) + 2(h_i + 1)\varepsilon.$$

Moreover, all the processing times and the due dates of G' are integer multiples of $\varepsilon/n!$.

Consider the sequence $t'_0 < t'_1 < \cdots < t'_k$, where:

$$t'_0 = t_0,$$

$$t'_{j+1} = t'_j + \lceil (t_{j+1} - t_j)/\varepsilon \rceil \varepsilon, \quad 0 \leqslant j \leqslant k - 1. \tag{11}$$

Let $\hat{r}$ be the largest integer such that $a_{\hat{r}} \leqslant t_k$. For $1 \leqslant r \leqslant \hat{r}$, let k_r be the index such that $t_{k_r} = a_r$. Define profile $M' = \{a'_r, m_r\}_{r=1}^{\infty}$ as follows:

$$a'_r = t'_{k_r}, \qquad 1 \leqslant r \leqslant \tilde{r}, \tag{12}$$

$$a'_r = \hat{r}\varepsilon + \lceil a_r/\varepsilon \rceil \varepsilon, \qquad r > \hat{r}. \tag{13}$$

It is clear that $M' \in \mathscr{M}$, and that the profile changing epochs are integer multiples of ε.

Define a schedule ρ of (G', M') as follows: a task i of G' is executed at speed v_i in the time interval $[t'_{h_i}, t'_{h_i+1})$ if and only if task i of G_e is executed at speed v_i in the time interval $[t_{h_i}, t_{h_i+1})$ under $\mathrm{SLF}(M, G_e)$. One can verify from (9) and (12) that any task i of G' finishes exactly at time t'_{h_i+1}. Therefore, the schedule ρ is feasible as $\mathrm{SLF}(M, G_e)$ is feasible. Note also that under $\mathrm{SLF}(M, G_e)$, during any time interval $[t_i, t_{i+1})$, at most one subset of tasks which have the same laxity is executed at speed strictly smaller than one. Thus, due to the definition of the due dates of G' (cf. (10)), for any two tasks i and j in G', if $h_i < h_j$, or if $h_i = h_j$ and $v_i > v_j$, task j is unenabled or has a greater laxity than task i during the time interval $[t'_{h_i}, t'_{h_i+1})$. Therefore, the schedule ρ thus constructed is a SLF schedule: $\rho(G', M') = \mathrm{SLF}(G', M')$.

Since (G', M') has commensurable timing (with commensurability factor $\varepsilon/n!$), Theorem 3.1 implies that $\mathrm{SLF}(G', M')$ is an optimal preemptive schedule of (G', M'):

$$L_{\mathrm{SLF}}(G', M') = L^*_{\mathrm{p}}(G', M').$$

Using the facts that for any task $i \in G_e$, $d_i(G') < d_i(G_e) + 2(k+1)\varepsilon$, and that $t'_{h_i+1} \geqslant t_{h_i+1}$, we get

$$C_\rho(i) - d_i(G') > t'_{h_i+1} - (d_i(G_e) + 2(k+1)\varepsilon) \geqslant C_{\mathrm{SLF}(G_e,M)}(i) - d_i(G_e) - 2(k+1)\varepsilon,$$

so that

$$\begin{aligned} L_{\mathrm{SLF}}(G, M) = L_{\mathrm{SLF}}(G_e, M) &< L_{\mathrm{SLF}}(G', M') + 2(k+1)\varepsilon \\ &= L^*_{\mathrm{p}}(G', M') + 2(k+1)\varepsilon. \end{aligned} \tag{14}$$

Let n' be the number of tasks in G'. It is clear that $n' \leqslant km$. By mimicking the proof of Lemma A.1 in Appendix A, we can show (cf. relations (5) and (6)) that

$$L^*_{\mathrm{p}}(G', M') \leqslant L^*_{\mathrm{p}}(G_e, M') + \frac{m^{n'} - 1}{m - 1}(mn'\bar{r} + m)\varepsilon. \tag{15}$$

Finally, by the definition of profile M' (cf. (11) and (12)), for any $0 \leqslant i \leqslant k-1$, the number of available processors in the time interval $[t_i, t_{i+1})$ under M is the same as that in $[t'_i, t'_i + t_{i+1} - t_i)$ under M'. Therefore, for any optimal preemptive schedule S^* of (G_e, M), we can construct a preemptive and idling schedule χ of (G_e, M') in such a way that for all $0 \leqslant i \leqslant k-1$, χ schedules tasks in the time interval $[t'_i, t'_i + t_{i+1} - t_i)$ in the same manner as S^* schedules tasks in the time interval $[t_i, t_{i+1})$. Clearly,

$$L_\chi(G_e, M') \leqslant L^*_{\mathrm{p}}(G_e, M) + \max_{1 \leqslant i \leqslant k} (t'_i - t_i) \leqslant L^*_{\mathrm{p}}(G_e, M) + k\varepsilon.$$

This implies that

$$L_{\mathrm{p}}^*(G_\varepsilon, M') \leqslant L_{\mathrm{p}}^*(G_\varepsilon, M) + k\varepsilon. \tag{16}$$

Putting the inequalities (14)–(16) all together, we finally obtain

$$\begin{aligned} L_{\mathrm{SLF}}(G, M) &< L_{\mathrm{p}}^*(G, M) + \left(3k + 2 + \frac{m^{n'} - 1}{m - 1}(mn'\bar{r} + m)\right)\varepsilon \\ &< L_{\mathrm{p}}^*(G, M) + (3k + 2m^{n'+1}n'\bar{r})\varepsilon. \end{aligned} \tag{17}$$

Using the inequalities $n' \leqslant km$ and $k \leqslant 2n + \bar{r} - 1$ in (17) entails

$$L_{\mathrm{p}}^*(G, M) \leqslant L_{\mathrm{SLF}}(G, M) < L_{\mathrm{p}}^*(G, M) + 3(2n + \bar{r})m^{(2n + \bar{r})m}\varepsilon. \tag{18}$$

This last relation readily implies the assertion of the theorem since ε can be arbitrarily small.

Appendix C. Proof of Lemma 4.2

We first show the following lemma which slightly extends Theorem 1 of [1].

Lemma C.1 (Extension of Theorem 1 of Brucker et al. [1]). *Let $G \in \mathscr{G}_{\mathrm{if}}$ be an in-forest, and $M \in \mathscr{M}_{\mathrm{iz}}$ be an increasing zigzag profile. Assume that the tasks have UET and that the profile changing epochs are integer. Then any EDD schedule $S \in \mathscr{E}(G', M)$ defined on the modified due dates meets all the original due dates if and only if such a feasible nonpreemptive schedule exists.*

Proof. The proof is similar to that of Theorem 1 in [1] except that we have to consider the variable profile. It suffices to show that, if an EDD schedule $S \in \mathscr{E}(G', M)$ defined on the modified due dates does not meet all the original due dates, then there is no feasible nonpreemptive schedule which achieves this.

Owing to Lemma 4.1, a given schedule meets all the original due dates if and only if it meets all the modified due dates. Assume there is an EDD schedule $S \in \mathscr{E}(G', M)$ defined on the modified due dates which does not meet all the original due dates. Let u be the task having the smallest modified due date among the tasks for which S fails to meet their modified due dates, i.e.

$$C_u(S) > d'_u, \quad \text{and} \quad d'_u \leqslant d'_i \quad \text{if } C_i(S) > d'_i.$$

Let $b = C_u(S) - 1$ be the time at which task u is assigned to a processor. We will show by contradiction that there is no idle processor in the time interval $[0, b)$ under S and that all the tasks assigned for execution under S during the time interval $[0, b)$ have smaller modified due dates. These facts trivially imply that there is no feasible schedule that meets all the modified due dates.

Assume that these facts are not true. Then, there is an integer $0 \leqslant t \leqslant b - 1$ such that during the time interval $[t, t + 1)$ of schedule S, there is an available processor which is either idle or executing a task j with strictly larger modified due date than task u ($d'_j > d'_u$). Let t be the largest such integer. Then during the time interval $[t+1, b)$ of schedule S, all the available processors are busy and are executing tasks whose modified due dates are smaller or equal to d'_u.

According to the definition of due date modifications, all the predecessors of a task $i \in G$ have strictly smaller modified due dates than i. Therefore, in the EDD schedule S defined on the modified due dates, if task i is assigned for execution before task j which has strictly smaller modified due date ($d'_j < d'_i$), then there is no precedence constraint between i and j.

Thus, if $t = b - 1$, then there is at least a predecessor task i of u such that i is executed during $[t, t+1)$ (otherwise task u would be assigned by time $t = b - 1$). This implies $C_i(S) = C_u(S) - 1 > d'_u - 1 \geqslant d'_i$, which contradicts the assumption on task u. Hence, $t = b - 1$ is impossible.

Therefore, $t < b - 1$. Assume without loss of generality that the profile is specified every unit of time, i.e. $a_r = r - 1$, $r = 1, 2, \ldots$. Since all the m_{t+1} tasks which start execution at time $t + 1$ have (nonstrictly) smaller modified due dates than u, all these tasks have predecessors executing during the time interval $[t, t+1)$. As the precedence graph is an in-forest, two tasks can not have the same predecessor. Hence at least m_{t+1} tasks which have (nonstrictly) smaller modified due dates than u start execution at t. Thus, $m_t > m_{t+1} + 1$.

Since the profile is zigzag increasing, we have necessarily

$$m_t - 1 = m_{t+1} \stackrel{\text{def}}{=} \hat{m}.$$

Moreover, for all $t' \geqslant t + 1$, $m_{t'} \geqslant \hat{m}$. By the definitions of u and t, all the tasks running during the time interval $[t+1, b)$ under schedule S have at least one (not necessarily immediate) predecessor which is assigned for execution at time t. Therefore, these tasks form $\hat{m}$ chains, one of which contains predecessors of u. Let i be the immediate predecessor of u such that $C_i(S) = b$. It then follows that $C_i(S) = C_u(S) - 1 > d'_u - 1 \geqslant d'_i$, which, again, contradicts the assumption on task u. The proof is thus completed. $\square$

Proof of Lemma 4.2. We use a standard argument to show that Lemma C.1 implies the assertion of Lemma 4.2. Let $L^* = L^*_{\mathrm{np}}(G, M)$. Let $\bar{G}$ be the task graph which differs from G only in the due dates: for all $i \in G$, $\bar{d}_i = d_i + L^*$. It is clear that for any schedule S,

$$L_S(\bar{G}, M) = L_S(G, M) - L^*,$$

so that $L^*_{\mathrm{np}}(\bar{G}, M) = 0$.

Moreover, one can easily see that for the modified due dates, we have the relation: $\bar{d}'_i = d'_i + L^*$, $i \in G$. Thus, an application of Lemma C.1 to $(\bar{G}, M)$ implies that $L_{\mathrm{EDD}'}(\bar{G}, M) \leqslant 0$. Hence,

$$L_{\mathrm{EDD}'}(G, M) = L_{\mathrm{EDD}'}(\bar{G}, M) + L^*_{\mathrm{np}}(G, M) \leqslant L^*_{\mathrm{np}}(G, M),$$

so that $L_{\mathrm{EDD}'}(G, M) = L^*_{\mathrm{np}}(G, M)$. $\square$

Acknowledgement

The authors are very grateful to the referees for their various constructive comments on both the contents and the style of the paper.

References

[1] P. Brucker, M.R. Garey and D.S. Johnson, Scheduling equal-length tasks under treelike precedence constraints to minimize maximum lateness, Math. Oper. Res. 2 (1977) 275–284.

[2] E.G. Coffman Jr, ed., Computer and Job-Shop Scheduling Theory (Wiley, New York, 1976).

[3] E.G. Coffman Jr and R.L. Graham, Optimal scheduling for two-processor systems, Acta Inform. 1 (1972) 200–213.

[4] D. Dolev and M.K. Warmuth, Scheduling precedence graphs of bounded height, J. Algorithms 5 (1984) 48–59.

[5] D. Dolev and M.K. Warmuth, Scheduling flat graphs, SIAM J. Comput. 14 (1985) 638–657.

[6] D. Dolev and M.K. Warmuth, Profile scheduling of opposing forests and level orders, SIAM J. Algebraic Discrete Methods 6 (1985) 665–687.

[7] M.R. Garey, D.S. Johnson, R.E. Tarjan and M. Yanakakis, Scheduling opposite forests, SIAM J. Algebraic Discrete Methods 4 (1983) 72–93.

[8] E.L. Lawler, Preemptive scheduling of precedence constrained scheduling, in: Dempster et al., eds., Deterministic and Stochastic Scheduling (Reidel, Dordrecht, 1982) 101–123.

[9] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, Sequencing and scheduling: Algorithms and complexity, Report BS-R8909, CWI (1989).

[10] Z. Liu and E. Sanlaville, Stochastic scheduling with variable profile and precedence constraints, Rapport de Recherche INRIA, No. 1525, 1991, submitted.

[11] R. McNaughton, Scheduling with deadlines and loss functions, Management Sci. 6 (1959) 1–12.

[12] R.R. Muntz and E.G. Coffman Jr, Optimal preemptive scheduling on two-processor systems, IEEE Trans. Comput. 18 (1969) 1014–1020.

[13] R.R. Muntz and E.G. Coffman Jr, Preemptive scheduling of real-time tasks on multiprocessor systems, J. ACM 17 (1970) 325–338.

[14] E. Sanlaville, Conception et analyse d'algorithmes de liste en ordonnancement préemptif, Thèse de l'Université Paris VI (September 1992).

[15] G. Schmidt, Scheduling independent tasks with deadlines on semi-identical processors, J. Oper. Res. Soc. 39 (1988) 271–277.

[16] J.D. Ullman, NP-complete scheduling problems, J. Comput. System Sci. 10 (1975) 384–393.