# The equivalence of two classical list scheduling algorithms for dependent typed tasks with release dates, due dates and precedence delays

Aurélien Carlier, Claire Hanen, Alix Munier-Kordon

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

# Equivalence of two classical list scheduling algorithms for dependent typed tasks with release dates, due dates and precedence delays

**Aurélien Carlier · Claire Hanen · Alix Munier Kordon**

Received: date / Accepted: date

**Abstract** We consider a finite set of unit time execution tasks with release dates, due dates and precedence delays. The machines are partitionned into $k$ classes. Each task requires one machine from a fixed class to be executed. The problem is the existence of a feasible schedule.

This general problem is known to be $\mathcal{NP}$-complete; many studies were devoted to the determination of polynomial time algorithms for some special subcasses, most of them based on a particular list schedule. The Garey-Johnson and Leung-Palem-Pnueli algorithms (respectively GJ and LPP in short) are both improving the due dates to build a priority list. They are modifiying them using necessary conditions until a fix point is reached. The present paper shows that these two algorithms are different implementations of a same generic one. The main consequence is that all the results valid for GJ algorithm, are also for LPP and vice versa.

**Keywords** list scheduling algorithms · polynomial sub-problems · approximation algorithms

## 1 Introduction

Scheduling problems with release and due dates have been considered for a long time, either in their decision version (is there a schedule meeting all the constraints ?) or in their optimization version, by minimizing the maximum lateness $L_{max}$. Most of the decision problems are $\mathcal{NP}$-complete once precedence and resource constraints are considered [6,12,15]. However, some particular instances have led to polynomial algorithms. Garey and Johnson [5] solved polynomially $P2|prec, p_i = 1, r_i, d_i|L_{max}$ and its preemptive version $P2|prec, pmtn, r_i, d_i|L_{max}$ by using a particular list scheduling algorithm. This algorithm was extended to get approximation algorithms for the $L_{max}$ criteria with various hypothesis: parallel processors [8], preemptive jobs [7], communication delays [7], typed tasks systems, and unitary resource constraints scheduling problems [1].

A.Carlier
Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris E-mail: Aurelien.Carlier@lip6.fr

C.Hanen
Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris and Univeristé Paris-Lumières, Université Paris Ouest Nanterre La Défense, 92000 Nanterre E-mail: Claire.Hanen@lip6.fr

A.Munier Kordon
Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris E-mail: Alix.Munier@lip6.fr

A similar class of scheduling problems was also studied in the context of parallel computing. Processing times are here unitary (*i.e.*, $p_i = 1$ for each task $i$) since each task corresponds to a basic instruction. Precedence delays, corresponding to a minimum duration between the executions of two dependent tasks, are introduced to model data transfers between them. Minimizing the makespan or the maximum lateness is $\mathcal{NP}$-complete even for one machine [2,9,14]. Leung *et al.* [13] developped a new list scheduling algorithm that solves polynomially several sub-problems including most of them for which a polynomial algorithm was already known. For an interval order precedence graph with monotone delays, the decision problem $P|int.\ order, monotone\ \ell_{ij}, p_i = 1, r_i, d_i|\star$ is solved polynomially using the LPP algorithm [13]. This algorithm was extended to handle monotone interval orders with typed tasks systems [4] or communications delays and dedicated procesors [11].

Garey and Johnson (*in short* GJ) and Leung, Palem and Pnueli (*in short* LPP) algorithms rely both on an iterative process that modifies the due dates until either a fixed point is reached (due dates are then said to be consistent), or a contradiction is observed. These modified due dates are then used as priorities to build a feasible schedule using a list scheduling algorithm. The main differences between these two algorithms are the definition of the fix point, or equivalently the definition of a set of consistent modified due dates, and the modification step of due dates. The aim of our study is to prove that GJ and LPP algorithms are two implementations of a generic modified due date algorithm for the general typed task decision problem $P\Sigma^k, |prec, \ell_{ij}, p_i = 1, r_i, d_i|\star$; they thus converge to the same unique fixed point. LPP algorithm has a better worst case complexity and thus must be preferred to compute efficiently modified due dates. From a theoretical point a view, our main conclusion is that all the results proved for GJ algorithm are true for LPP, and vice versa.

The computation of the fix point is polynomial for both GJ and LPP algorithms. However, the list priority algotithm based on the due dates obtained does not necessarily provide an optimal solution. Thus, there is no contradiction with as example the NP-hardness result from Yu et al. [16] for $F2|\ell_{ij}, p_i = 1, |C_{max}$.

The paper is organized as follows: Section 2 introduces the problem and notations. Extended GJ and LPP consistencies are introduced in Section 3. Section 4 is devoted to the proof of the equivalence between LPP and GJ consistency. A generic definition of the due date modification algorithm is then presented in Section 5. It is also proved the convergence to a unique fixed point, if it exists. The consequence is that LPP and GJ algorithms are converging to the same modified due dates. The implementation of LPP and GJ algorithms and their complexity are briefly recalled in Section 6, followed by an experimental comparison of their performances for $m$ identical machines and a usual precedence graph. A list of polynomial sub-problems and approximation results valid for both algorithms is established in Section 7. Section 8 is our conclusion.

## 2 Problem definition and notations

Let $\mathcal{T} = \{1, \cdots, n\}$ a set of tasks with unit processing times $p_i = 1$, release time $r_i$ and due dates $d_i$ for $i \in \mathcal{T}$.

Precedence relations are given by a directed acyclic graph noted $G = (\mathcal{T}, E)$. Each arc $e = (i, j) \in E$ is valued by a non negative integer $\ell_{ij}$; this value models a minimum delay between the end of $i$ and the beginning of $j$. If $t_i$, $i \in \mathcal{T}$ denotes the starting time of task $i$, the relation expressed by any arc $e = (i, j) \in E$ is $t_i + p_i + \ell_{ij} \leq t_j$. We assume without loss of generality that release times are consistent with respect to precedence delays, *i.e.* if $(i, j) \in E$ then $r_i + 1 + \ell_{ij} \leq r_j$. All the values considered in this paper in relation to time are integers.

Machines (or processors) are partitionned into $k$ classes $\{C_1, \cdots, C_k\}$, each of them containing $m_p$ identical processors for $p \in \{1, \cdots, k\}$. Each task $i$ requires one processor from a fixed class $\tau_i \in \{C_1, \cdots, C_k\}$ and is then called typed task. Typed tasks systems

include both identical processors (only one class of processor) and dedicated processors (only one processor per class).

The main problem consists in computing a starting times vector $\mathbf{t} = (t_1, \cdots, t_n)$ of tasks such that release and due dates vectors $\mathbf{r} = (r_1, \cdots, r_n)$ and $\mathbf{d} = (d_1, \cdots, d_n)$, precedence delays and resource constraints are met. It is refered to $P\Sigma^k|prec, \ell_{ij}, p_i = 1, r_i, d_i|\star$ using standard notations. The minimization of the maximum lateness, defined as $L_{max} = \max_{i\in\mathcal{T}}(t_i + 1 - d_i)$ is also discussed; this last optimization problem is noted as $P\Sigma^k|prec, \ell_{ij}, p_i = 1, r_i|\ L_{max}$.

We denote $i \to j$ if there is a path from $i$ to $j$ in $G$ with $i \neq j$. We also set $\mathcal{S}_i = \{j \in \mathcal{T}, i \to j\}$ and $\mathcal{P}_i = \{j \in \mathcal{T}, j \to i\}$ for respectively the successors and the predecessors of $i$. For any task $i \in \mathcal{T}$, $Indep(i)$ is the set of task from $\mathcal{T}$ with no precedence relation with $i$, thus neither $i \to j$ nor $j \to i$. For any task $i \in \mathcal{T}$, let $\mathcal{T}_i = Indep(i) \cup \mathcal{S}_i$. We set that $\forall i \in \mathcal{T}$, $i \notin Indep(i)$ and $i \notin \mathcal{T}_i$.

For convenience, we denote by $\ell^+_{ij}$ the transitive latency between all pairs of tasks $(i, j) \in \mathcal{T}^2$ with $j \in \mathcal{S}_i$. Formally, for any path $\nu = i_1, \cdots, i_p$ of $G$ from $i_1$ to $i_p$ with $p > 1$, we set $\ell(\nu) = \sum_{\alpha=1}^{p-1}(\ell_{i_\alpha i_{\alpha+1}} + 1) - 1$. $\ell^+_{ij}$ is then the maximum value $\ell(\nu)$ of any path of $G$ from $i$ to $j$. These values may be easily computed in time-complexity $\Theta(n^3)$ using a simple modification of the Floyd-Warshall algorithm [3].

Consider the instance of our decision problem pictured by Figure 1 for 3 dedicated processors, *i.e.* $m_1 = m_2 = m_3 = 1$. Figure 2 presents a feasible solution.

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $r_i$ | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 1 | 2 |
| $d_i$ | 3 | 3 | 3 | 3 | 2 | 4 | 3 | 3 | 3 |
| $\tau_i$ | $C_1$ | $C_1$ | $C_1$ | $C_2$ | $C_2$ | $C_2$ | $C_3$ | $C_3$ | $C_3$ |

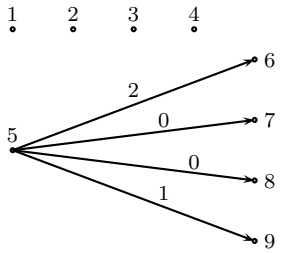



Fig. 1: Release-dates, due dates, type of tasks and precedence graph with delays.

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| $t_i$ | 0 | 1 | 2 | 1 | 0 | 3 | 1 | 2 | 3 |

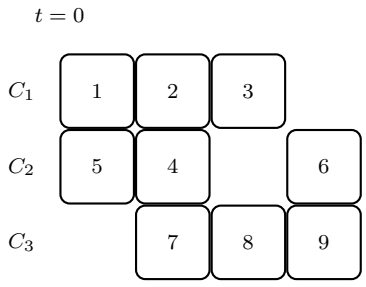



Fig. 2: A feasible schedule for the instance pictured by Figure 1.

## 3 GJ and LPP consistencies

GJ and LPP algorithms modify due dates until a fixed point $\mathbf{D} \leq \mathbf{d}$ is reached, expressing in both cases necessary conditions for the existence of a feasible schedule that can be computed polynomially. This section aims at presenting these two consistency conditions extended to typed-tasks systems and precedence delays.

### 3.1 GJ consistency

GJ consistency was introduced in [5] for solving polynomially the decision problem $2|prec, r_i, d_i, p_i = 1|\star$. It was extended to typed-tasks systems in [1]. A wider extension is presented here to integrate precedence delays.

The main idea is expressed necessary conditions on ressource for testing if a fixed due dates vector $\mathbf{D} = (D_1, \cdots, D_n) \leq \mathbf{d}$ is feasible or not. Let $i$ be a fixed task from $\mathcal{T}$. Assuming that $i$ is completed at time $D_i$, release date for any task $j$ from $\mathcal{T}_i$ is adjusted following :

$$r'_j = \begin{cases} r_j & \text{if } j \in Indep(i) \\ \max(r_j, D_i + \ell^+_{ij}) & \text{if } j \in \mathcal{S}_i. \end{cases}$$

For any class of processors $C_p$, $p \in \{1, \cdots, k\}$ and any tuple $(\alpha, \beta)$ of integers such that $\alpha < \beta$ and $D_i \leq \beta$, let $S_p(i, \alpha, \beta)$ be the set of tasks from $\mathcal{T}_i$ that must be executed during the interval $[\alpha, \beta]$ if $i$ is completed at time $D_i$. More formally,

$$S_p(i, \alpha, \beta) = \{j \in \mathcal{T}_i \mid \tau_j = C_p,\ \alpha \leq r'_j \text{ and } D_j \leq \beta\}.$$

The total number of slots available for scheduling tasks from $S_p(i, \alpha, \beta)$ may depends on the availability of the resource needed for $i$. Two cases must be considered:

- If $\alpha < D_i$ and $\tau_i = C_p$, then $D_i - 1$ belongs to $[\alpha, \beta]$. Task $i$ is thus executed during the interval $[\alpha, \beta]$ on a machine of type $p$ (since it is executed at time $D_i - 1$), and the number of remaining idle slots equals $m_p(\beta - \alpha) - 1$. $S_p(i, \alpha, \beta)$ is said consistent if $|S_p(i, \alpha, \beta)| < m_p(\beta - \alpha)$;
- Otherwise, $\alpha \geq D_i$ or $\tau_i \neq C_p$. Thus $i$ is not executed in the interval $[\alpha, \beta]$ on a machine of type $p$. $S_p(i, \alpha, \beta)$ is then said consistent if $|S_p(i, \alpha, \beta)| \leq m_p(\beta - \alpha)$.

For any task $i \in \mathcal{T}$, the due dates vector $\mathbf{D}$ is *GJ-consistent* for $i$ if $\forall p \in \{1, \cdots, k\}$, $\forall (\alpha, \beta) \in \mathbb{N}^2$ such as $\alpha < \beta$ and $D_i \leq \beta$, the set $S_p(i, \alpha, \beta)$ is consistent. Due dates vector $\mathbf{D}$ is said to be *GJ-consistent* if it is GJ-consistent for any task $i \in \mathcal{T}$. A non-consistent set is said to be *critical*.

Consider for example the instance pictured by Figure 1 and set $\mathbf{D} = \mathbf{d}$, $i = 5$, $\alpha = 2$ and $\beta = 4$. Observe that $\mathcal{T}_5 = \mathcal{T} - \{5\}$ and that $\mathbf{r}' = (0, 0, 0, 0, -, 4, 2, 2, 3)$. We get $S_1(5, 2, 4) = \emptyset$, $S_2(5, 2, 4) = \{6\}$ and $S_3(5, 2, 4) = \{7, 8, 9\}$.

Observe that $|S_1(5, 2, 4)| = 0 \leq 2$ and $|S_2(5, 2, 4)| = 1 \leq 2$. These two sets are thus consistent. Now, we get that $|S_3(5, 2, 4)| = 3 > 2$. If task 5 is performed at time $D_5 - 1 = 1$, all tasks from $S_3(5, 2, 4)$ have to be performed in the time slot $[2, 4]$ by processor $C_3$ to satisfy precedence constraints and deadlines, which is impossible. $S_3(5, 2, 4)$ is thus a critical set, and the due dates vector $\mathbf{D}$ is not GJ-consistent.

### 3.2 LPP consistency

LPP consistency [13] was initially given for solving $P|prec, \ell_{ij}, r_i, d_i, p_i = 1|\star$. It is based on a polynomial algorithm checking necessary conditions for the feasibility of a due dates vector $\mathbf{D}$. This algorithm for computing feasible due dates vectors has already been extended by [4] to include typed tasks.

Let $i \in \mathcal{T}$. Consider release and due dates vectors $\mathbf{r}$ and $\mathbf{D}$ and a value $t \in \{r_i + 1, \cdots, D_i\}$. Then, $Existence_i(t, \mathbf{r}, \mathbf{D})$ defined as follows is a necessary condition for the existence of a feasible schedule such that $i$ ends at time $t$:

$Existence_i(t, \mathbf{r}, \mathbf{D})$: is there a schedule of tasks from $\mathcal{T}_i \cup \{i\}$ considered as independent meeting resource constraints, release dates $\mathbf{r}'$ and due dates $\mathbf{D}'$ defined as follows:

1. $r'_i = t - 1$ and $D'_i = t$;
2. $\forall j \in Indep(i)$, $r'_j = r_j$ and $D'_j = D_j$;
3. $\forall j \in \mathcal{S}_i$, $r'_j = \max(r_j, t + \ell^+_{ij})$ and $D'_j = D_j$.

The tasks considered here are unitary (*i.e.* $p_j = 1$, $\forall j \in \mathcal{T}_i$). So, $Existence_i(t, \mathbf{r}, \mathbf{D})$ can be solved using the Jackson priority list algorithm [10] (*i.e.* an earliest deadline first priority list scheduling) until either a task miss its deadline or all tasks are scheduled, in a time-complexity $\Theta(n \log n)$.

Let the due dates vector $\mathbf{D} = (D_1, \cdots, D_n)$, a fixed task $i \in \mathcal{T}$ and $t \leq D_i$ the maximum integer value such that $Existence_i(t, \mathbf{r}, \mathbf{D})$ is true. The due dates vector $\mathbf{D}$ is *LPP-consistent* for task $i$ if $t = D_i$. It is said to be *LPP-consistent* if it is LPP-consistent for all tasks $i \in \mathcal{T}$.

Consider the decision problem $Existence_5(D_5, \mathbf{r}, \mathbf{D})$ with $\mathbf{D} = \mathbf{d}$ for the example pictured by Figure 1. Release and due dates vectors are respectively $\mathbf{r}' = (0, 0, 0, 0, 1, 4, 2, 2, 3)$ and $\mathbf{D}' = \mathbf{D}$. The schedule obtained using Jackson's algorithm is pictured by Figure 3. Tasks 6, 8 and 9 are not meeting their due date, thus the answer to $Existence_5(D_5, \mathbf{r}, \mathbf{D})$ is negative and $\mathbf{D}$ is not LPP-consistent for task 5.

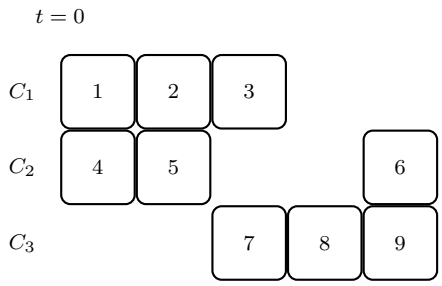



Fig. 3: The Jackson's schedule for the instance $Existence_5(D_5, \mathbf{r}, \mathbf{D})$.

**Lemma 1** *Assume that* $\mathbf{D}$ *is LPP-consistent. For any couple* $(i, j) \in \mathcal{T}^2$ *with* $j \in \mathcal{S}_i$, *the inequality* $D_i < D_j$ *holds.*

*Proof* Suppose by contradiction that $j \in \mathcal{S}_i$ and $D_i \geq D_j$. Since $\mathbf{D}$ is LPP-consistent, the answer to $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ is positive. Thus, $r'_j = \max(r_j, D_i + \ell^+_{ij}) \geq D_i \geq D_j = D'_j$ *i.e.* $r'_j \geq D'_j$, a contradiction.

## 4 Equivalence between GJ and LPP consistencies

As pointed before, GJ and LPP consistencies were defined separately using different approaches. This section aims to prove that they are in fact equivalent for our general problem. Lemmas 2 and 3 are considered to show by contraposition in Lemma 4 that any LPP-consistent due dates vector $\mathbf{D}$ is GJ-consistent. The inverse is deduced in Lemma 7 from Lemmas 5 and 6. Theorem 1 is then a simple outcome.

**Lemma 2** *Let* $\mathbf{D} = (D_1, \cdots, D_n)$ *be a due dates vector. For any* $p \in \{1, \cdots, k\}$, *for any task* $i \in \mathcal{T}$ *and any tuple* $(\alpha, \beta) \in \mathbb{N}^2$ *such as* $\alpha < \beta$ *and* $D_i \leq \beta$, *if* $j \in S_p(i, \alpha, \beta)$, *then* $j$ *is scheduled in* $[\alpha, \beta]$ *for any solution of the decision problem* $Existence_i(D_i, \mathbf{r}, \mathbf{D})$.

*Proof* Since $j \in S_p(i, \alpha, \beta)$, we have $\alpha \leq r'_j$ and $D_j \leq \beta$. Thus, $j$ must be scheduled during $[\alpha, \beta]$ for any solution of the decision problem $Existence_i(D_i, \mathbf{r}, \mathbf{D})$.

Lemma 3 simply derives from Lemma 2.

**Lemma 3** *Let* $i \in \mathcal{T}$ *and a due date vector* $\mathbf{D}$. *If* $\mathbf{D}$ *is not GJ-consistent for* $i$, *then* $\mathbf{D}$ *is not LPP-consistent for* $i$.

*Proof* Assume that $\mathbf{D}$ is not GJ-consistent for a fixed task $i \in \mathcal{T}$. Then, there exists $p \in \{1, \cdots, k\}$, and two integers $\alpha, \beta$ such as $\alpha < \beta$, $D_i \leq \beta$ and the set $S_p(i, \alpha, \beta)$ is critical. Two cases must be considered following the definition of GJ consistency:

1. Assume first that $\alpha < D_i$ and $\tau_i = C_p$. Since $S_p(i, \alpha, \beta)$ is critical, $|S_p(i, \alpha, \beta)| \geq m_p(\beta - \alpha)$. Now, $i$ must be scheduled at $D_i - 1$ by $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ and $i$ does not belong to $S_p(i, \alpha, \beta)$. Moreover, by Lemma 2, every task $j \in S_p(i, \alpha, \beta)$ must be scheduled by $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ in the interval $[\alpha, \beta]$. Thus, there are at least $m_p(\beta - \alpha) + 1$ tasks to be scheduled by $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ in the time interval $[\alpha, \beta]$ by the $m_p$ processors of type $C_p$, which is impossible.
2. Now, if $\alpha \geq D_i$ or $\tau_i \neq C_p$, we get that $|S_p(i, \alpha, \beta)| > m_p(\beta - \alpha)$. By Lemma 2, tasks from $S_p(i, \alpha, \beta)$ have to be scheduled by $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ in the interval $[\alpha, \beta]$. As those tasks have to be schedule by a processor of type $C_p$, $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ is also false.

In both cases, $\mathbf{D}$ is not LPP-consistent for $i$, which proves the lemma.

For the example given by Figure 1, the set $S_3(5, 2, 4) = \{7, 8, 9\}$ is critical and thus $\mathbf{D}$ is not GJ-consistent. The corresponding scheduling problem $Existence_5(D_5, \mathbf{r}, \mathbf{D})$ pictured by Figure 3 does not meet due dates and $\mathbf{D}$ is not LPP-consistent.

**Lemma 4** *If* $\mathbf{D}$ *is a LPP-consistent due dates vector, then* $\mathbf{D}$ *is GJ-consistent.*

*Proof* If $\mathbf{D}$ is not GJ-consistent, then there exists $i \in \mathcal{T}$ such that $\mathbf{D}$ is not GJ-consistent for $i$. By Lemma 3, $\mathbf{D}$ is not LPP-consistent for $i$, which proves the lemma.

In the following, we show that the reverse of Lemma 4 is true. Lemma 5 is a simple technical property of Jackson algorithm widely used to get next lemmas.

**Lemma 5** *Let* $I$ *be an unfeasible instance of* $P\Sigma^k|r_i, p_i = 1, d_i|\star$ *with* $\forall i \in \mathcal{T}$, $r_i < d_i$. *Let also* $\sigma = \{t_1, \cdots, t_n\}$ *be a schedule obtained using Jackson algorithm (*i.e. *a priority list algorithm using earliest deadline first). If* $j$ *is a task of minimum starting time* $t_j$ *such that* $d_j \leq t_j$, *then* $d_j = t_j$.

*Proof* Any task $u$ performed at $t_j - 1$ meets its deadline, thus $t_j = t_u + 1 \leq d_u$. Now, $r_j < d_j$, thus $j$ was ready at time $t_j - 1$, so by the priority list algorithm, $d_u \leq d_j$. We get thus $t_j = t_u + 1 \leq d_u \leq d_j \leq t_j$, and then $t_j = d_j$.

Lemma 6 shows that a critical set may be associated to any execution of *Existence* which answer is negative.

**Lemma 6** *Let* $i$ *be a task from* $\mathcal{T}$ *such that for any task* $\ell \in \mathcal{T}_i$, $r_\ell < D_\ell$ *and* $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ *is false. Then, there exist* $p \in \{1, \cdots, k\}$, *a task* $u \in \mathcal{T}_i \cup \{i\}$ *and a tuple of integers* $(\alpha, \beta)$ *with* $\alpha < \beta$ *and* $D_u \leq \beta$ *such that* $S_p(u, \alpha, \beta)$ *is a critical set.*

*Proof* Consider a task $i \in \mathcal{T}$ such that for any task $\ell \in \mathcal{T}_i$, $r_\ell < D_\ell$ and $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ is false. Let us denote by $r'_j$ and $D'_j$, for $j \in \mathcal{T}_i \cup \{i\}$ respectively release and due dates computed locally by this function. Note that $\forall j \in \mathcal{T}_i \cup \{i\}$, $D'_j = D_j$. Let $\sigma = \{t_j, j \in \mathcal{T}_i \cup \{i\}\}$ be the schedule obtained by using Jackson algorithm.

Let us consider the first task $j \in \mathcal{T}_i \cup \{i\}$ missing its due date ($D_j \leq t_j$). Let $p \in \{1, \cdots, k\}$ be such that $\tau_j = C_p$. By Lemma 5, $t_j = D_j$.

Let $\alpha \leq t_j$ be the rightmost date for which either there exists an idle processor of type $p$ at time $\alpha - 1$ (situation 1) or there exists a task $v$ with $\tau_v = C_p$ and $D_v > D_j$ performed in this interval (situation 2). If no such event occurs, then set $\alpha = 0$.

Let us set now $\beta = t_j = D_j$ and denote by $\mathcal{X}$ the set of tasks $k$ with $\tau_k = C_p$ performed in the interval $[\alpha, \beta]$, to which we add $j$. Formally, this set can be define as

$$\mathcal{X} = \{k \in \mathcal{T}_i \cup \{i\} \mid \tau_k = C_p \text{ and } \alpha \leq t_k \leq \beta - 1\} \cup \{j\}.$$

Notice that any task $\ell$ of $\mathcal{X}$ fulfills $\alpha \leq r'_\ell$, where $\mathbf{r}'$ is the release dates vector computed by $Existence_i(D_i, \mathbf{r}, \mathbf{D})$, and $D_\ell \leq D_j = \beta$. If $\alpha = 0$, the inequality $r'_\ell \geq 0$ is trivial. Else, in both situations (1 and 2), $\alpha \leq r'_\ell$ otherwise $\ell$ would have been scheduled at time $\alpha - 1$ (either on the idle processor in situation 1 or instead of task $v$ in situation 2). Furthermore, every tasks of $\mathcal{X} - \{j\}$ are considered before $j$ in the Jackson's schedule, meaning that $D_\ell \leq D_j = \beta$. And finally, we observe that $\mathcal{X}$ contains exactly $m_p(\beta - \alpha) + 1$ tasks.

Now, two main cases are considered:

Case 1: Assume that, $\forall \ell \in \mathcal{X}$, $\alpha \leq r_\ell$ and let $u \in \mathcal{X}$ be a task with $t_u = \alpha$. Since $u \in \mathcal{X}$, we have $D_u \leq D_j$. Note that any task $\ell \in \mathcal{X} - \{u\}$ satisfies $\alpha \leq r'_\ell$ and $D_\ell \leq \beta$. Thus, $\mathcal{X} - \{u\} \subseteq S_p(u, \alpha, \beta)$ and therefore that $|\mathcal{X} - \{u\}| = m_p(\beta - \alpha)$ with $\alpha \leq r_u < D_u$. We deduce that $S_p(u, \alpha, \beta)$ is critical, so that $\mathbf{D}$ is not GJ-consistent for $u$.

Case 2: Assume now that there exists at least one task $\ell \in \mathcal{X}$ such that $r_\ell < \alpha$. As $\ell \in \mathcal{X}$, we deduce that $r_\ell < \alpha \leq r'_\ell$ and thus $\ell \in \mathcal{S}_i$ or $\ell = i$.

We also observe that $r_i < \alpha$ and $D_i \leq \beta$. Indeed:

- if $\ell \in \mathcal{S}_i$, then $r_i < r_\ell < \alpha$ since release dates are consistent with precedence constraints. Furthermore, since $\ell \in \mathcal{X}$ must be completed at the latest time $\beta$ (*i.e.* $D_\ell \leq \beta$) thus by Lemma 1 task $i$ must complete at the latest time $D_\ell - 1 < \beta$ (*i.e.* $D_i < \beta$);
- if $\ell = i$, then $r_i = r_\ell < \alpha$ and since $\ell \in \mathcal{X}$, then $D_\ell = D_i \leq \beta$.

We prove in the following that $S_p(i, \alpha, \beta)$ is critical by considering two cases depending on task $i$:

Case 2.1: Assume that $\tau_i = C_p$ and $\alpha < D_i$. Since $\alpha \leq D_i - 1 = r'_i$ and $D_i \leq \beta$, then $i$ is scheduled between times $\alpha$ and $\beta$. Furthermore, since $\tau_i = C_p$, then $i \in \mathcal{X}$.

Now we show that $\mathcal{X} - \{i\} \subseteq S_p(i, \alpha, \beta)$. Let $\ell$ be a task picked in $\mathcal{X} - \{i\}$. By definition of $\mathcal{X}$, we have $\tau_\ell = C_p$, $\alpha \leq r'_\ell$ and $D_\ell \leq \beta$. Thus, $\ell \in S_p(i, \alpha, \beta)$.

Now, $|\mathcal{X} - \{i\}| = m_p(\beta - \alpha)$. Since $\mathcal{X} - \{i\} \subseteq S_p(i, \alpha, \beta)$, then $m_p(\beta - \alpha) \leq |S_p(i, \alpha, \beta)|$ with $\alpha < D_i$ and thus $S_p(i, \alpha, \beta)$ is critical.

Case 2.2: We suppose now that $\tau_i \neq C_p$ or $D_i \leq \alpha$. If $\tau_i \neq C_p$, then $i \notin \mathcal{X}$. Otherwise $D_i \leq \alpha$; since $r'_i = D_i - 1$, then $i$ is not scheduled between times $\alpha$ and $\beta$ and thus $i \notin \mathcal{X}$.

Now we show that $\mathcal{X} \subseteq S_p(i, \alpha, \beta)$. Let $\ell$ be a task picked in $\mathcal{X}$. By definition of $\mathcal{X}$, we have $\tau_\ell = C_p$, $\alpha \leq r'_\ell$ and $D_\ell \leq \beta$. Thus, $\ell \in S_p(i, \alpha, \beta)$.

Now, $|\mathcal{X}| = m_p(\beta - \alpha) + 1 > m_p(\beta - \alpha)$. Since $\mathcal{X} \subseteq S_p(i, \alpha, \beta)$, then $m_p(\beta - \alpha) < |S_p(i, \alpha, \beta)|$ and thus $S_p(i, \alpha, \beta)$ is critical.

Consiter for example the Jackson's schedule pictured by Figure 3 which violates some due dates. We get $i = 5$, and the first task missing its due date is $j = 8$. Thus $p = 3$, $\alpha = 2$ and $\beta = 3$. As $r_7 = 1 < \alpha$, it illustrates Case 2. Now, $\tau_i = \tau_5 = C_2$, thus we get Case 2.2 and $|S_3(5, 2, 3)| = |\{7, 8, 9\}| = 3 > 1$ and thus is critical.

**Lemma 7** *If* $\mathbf{D}$ *is a GJ-consistent due dates vector, then* $\mathbf{D}$ *is LPP-consistent.*

*Proof* If $\mathbf{D}$ is not LPP-consistent, then there exists $i \in \mathcal{T}$ such that $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ is false. By Lemma 6, we deduce that there exists a critical set associated with $\mathbf{D}$ and thus $\mathbf{D}$ is not GJ consistent.

Theorem 1 is thus a simple outcome of lemmas 4 and 7.

**Theorem 1** *Consider an instance of the problem* $P\Sigma^k|prec, \ell_{ij}, r_i, d_i, p_i = 1|\star$ *and let* $\mathbf{D} = (D_1, \cdots, D_n)$ *be a vector of due dates such that* $D_i \leq d_i, \forall i \in \mathcal{T}$. $\mathbf{D}$ *is GJ-consistent if and only if* $\mathbf{D}$ *is LPP-consistent.*

## 5 A generic fixed point algorithm for the due dates modifications

GJ and LPP consistencies are naturally leading to a simple algorithm, starting from initial due dates, and iteratively decreases them until either a contradiction is found, or the modified due dates are consistent. In this latter case, any feasible schedule meets these due dates.

This simple generic algorithm is formulated by Algorithm 1. For any task $i \in \mathcal{T}$ which is not consistent for $\mathbf{D}$ (in the sense of LPP or GJ consistency), the function Adjust computes an integer value $\Delta < D_i$ such that, for any value $\delta \in \{\Delta+1, \cdots, D_i\}$, the vector $\overline{\mathbf{D}}$ defined as

$$\forall j \in \mathcal{T}, \overline{D}_j = \begin{cases} \delta & \text{if } i = j \\ D_j & \text{otherwise} \end{cases}$$

is not consistent for $i$. Note that the vector $\overline{\mathbf{D}} = \mathbf{D}$ is not consistent for $i$, thus the adjustement always returns a value strictly lower than $D_i$ insuring the convergence of the algorithm.

**Algorithm 1** A generic modified due dates algorithm

**Require:** An instance of $P\Sigma^k|prec, \ell_{ij}, p_i = 1, r_i, d_i|\star$
**Ensure:** Modified feasible due dates $\mathbf{D}^\star$ or a contradiction.
  $\forall i \in \mathcal{T}$, set $D_i = d_i$
  **while** $\mathbf{D}$ is not consistent **do**
    choose a task $i \in \mathcal{T}$ not consistent for $\mathbf{D}$
    $D_i = \text{Adjust}(i, D)$
    **if** $D_i \leq r_i$ **then**
      exit with a contradition
    **end if**
  **end while**

Notice that GJ (resp. LPP) algorithm is a particular implementation of algorithm 1, (see the next section for details), using GJ (resp. LPP) consistency in the loop test, and defining a particular version of the adjustement function. By Theorem 1, LPP and GJ consistencies are strictly equivalent. Next Lemma is a simple fundamental property for insuring the unicity of the convergence for both algorithms.

**Lemma 8** *Let* $i \in \mathcal{T}$. *Let also* $\mathbf{D}$ *and* $\mathbf{D}'$ *be two due dates vectors with* $D_i = D'_i$ *and* $\forall j \in \mathcal{T} - \{i\}$, $D_j \leq D'_j$. *If* $\mathbf{D}'$ *is not GJ-consistent (*resp. *LPP-consistent) for* $i$, *neither is* $\mathbf{D}$.

*Proof* We show the lemma separately for both consistencies:

- Suppose that $\mathbf{D}'$ is not GJ-consistent for $i$, then there exists $p \in \{1, \cdots, k\}$ and a couple $(\alpha, \beta) \in \mathbb{N}^2$ with $\alpha < \beta$ and such that the set $S'_p(i, \alpha, \beta)$ associated with

the due dates vector $\mathbf{D'}$ is critical. We denote by $S_p(i,\alpha,\beta)$ the corresponding set associated to $\mathbf{D}$.
Now, as $D'_i = D_i$, modified release dates $r'_j$ are identical for both sets $S_p(i,\alpha,\beta)$ and $S'_p(i,\alpha,\beta)$. Moreover, as $D_j \leq D'_j$ for any task $j \in \mathcal{T} - \{i\}$, $S'_p(i,\alpha,\beta) \subseteq S_p(i,\alpha,\beta)$. $S'_p(i,\alpha,\beta)$ is thus critical, the lemma.

– Suppose that $\mathbf{D'}$ is not LPP-consistent for $i$, then $Existence_i(D'_i, \mathbf{r}, \mathbf{D'})$ is false. As $D'_i = D_i$, modified release dates $r'_j$ are identical for $Existence_i(D'_i, \mathbf{r}, \mathbf{D'})$ and $Existence_i(D_i, \mathbf{r}, \mathbf{D})$. As $D_j \leq D'_j$ for any task $j \in \mathcal{T} - \{i\}$, $Existence_i(D_i, \mathbf{r}, \mathbf{D})$ is also false, the result.

Theorem 2 insures that the algorithm converges to a unique fixed point if it exists.

**Theorem 2** *For any instance of the decision problem $P\Sigma^k|prec, \ell_{ij}, p_i = 1, r_i, d_i|\star$, all possible executions of the generic algorithm for a fixed consistency (GJ or LPP) end with the same result: either a contradiction is found or the algorithm computes a unique consistent due dates vector $\mathbf{D}^\star$ with $\forall i \in \mathcal{T}, D_i^\star \leq d_i$.*

*Proof* Suppose by contradiction that two distinct executions of the algorithm for a fixed consistency (GJ or LPP) do not end with the same result. Two cases may occur:

Case a: There are two distinct consistent due dates vectors denoted by $\mathbf{D}^{\star 1}$ and $\mathbf{D}^{\star 2}$. Assume without loss of generality that there exists a task $i \in \mathcal{T}$ for which $D_i^{\star 1} < D_i^{\star 2}$.

Case b: An execution ends with a contradiction whereas the second one ends with a consistent due dates vector. In this case, we assume without loss of generality that the first execution of the generic algorithm induces a contradiction. Similarly to Case a, we denote by $\mathbf{D}^{\star 2}$ the set of consistent due dates produced by the second execution of the generic algorithm.

Along the iterative process for both executions, as they start with the same due dates $d_i \geq D_i^{\star 2}$, there was an iteration of the first execution before which the variable $D_i$ became less than $D_i^{\star 2}$. We consider the first such iteration in the first execution of the generic algorithm. Let $\widehat{\mathbf{D}}$ be the due date vector just before this iteration, and $\widetilde{\mathbf{D}}$ the due date vector computed at this iteration.

Clearly, $\widetilde{D}_i < D_i^{\star 2} \leq \widehat{D}_i$, and $\forall j \in \mathcal{T} - \{i\}$, $\widehat{D}_j = \widetilde{D}_j \geq D_j^{\star 2}$. Now, let us define the due date vector $\mathbf{D'}$ as

$$\forall j \in \mathcal{T}, D'_j = \begin{cases} D_i^{\star 2} & \text{if } i = j \\ \widehat{D}_j & \text{otherwise.} \end{cases}$$

$\widehat{\mathbf{D}}$ is not consistent for $i$ by hypothesis, and $\widetilde{D}_i < D_i^{\star 2}$. Thus, by definition of the adjustement function, $\mathbf{D'}$ is not consistent for $i$. As $D'_i = D_i^{\star 2}$ and $\forall j \in \mathcal{T} - \{i\}$, $D'_j \geq D_j^{\star 2}$, $\mathbf{D}^{\star 2}$ is not consistent for $i$ by Lemma 8, a contradiction.

**Corollary 1** *LPP and GJ algorithms do end with the same result.*

*Proof* By Theorem 1, a due date vector is LPP-consistent if and only if it is GJ-consistent. Moreover, by Lemma 3, at each step of the GJ algorithm, the task $i$ chosen is also not LPP-consistent for $\mathbf{D}$. The consequence is that an execution of the GJ algorithm can be seen as a special case of LPP algorithm. By Theorem 2, they are all ending in the same way, the corollary.

## 6 Implementation of LPP and GJ algorithms

This section aims at recalling the implementation details of the GJ and LPP algorithm and their worst-case complexity. An experimental comparison of their performance is also provided for identical processors.

### 6.1 Implementation of GJ algorithm

The implementation developed in [5] concerns a usual precedence graph (without delays) and 2 identical processors. Algorithm 2 is an extended version of Garey-Johnson algorithm to a typed tasks system with precedence delays. The global loop for testing the consistency of a due date vector of the Algorithm 1 is replaced by 4 nested loops to enumerate all possible critical sets. Its time complexity clearly belongs to $\mathcal{O}(n^3 \times k)$.

**Algorithm 2** Garey-Johnson extended algorithm

**Require:** An instance of $P\Sigma^k|prec, \ell_{ij}, p_i = 1, r_i, d_i|\star$
**Ensure:** Modified due dates $\mathbf{D}^\star$
  $\forall i \in \mathcal{T}$, set $D_i = d_i$;
  **for** $\beta = n$ to 1 **do**
    **for** any task $i \in \mathcal{T}$ with $D_i \leq \beta$ **do**
      **for** $\alpha = 0$ to $D_i - 1$ **do**
        **for** any $p \in \{1, \cdots, k\}$ **do**
          set $Y = S_p(i, \alpha, \beta)$;
          **if** $Y$ is critical **then**
            set $D_i = \max_{j \in Y} D_j - \lceil \frac{|Y|}{m_p} \rceil$;
          **end if**
        **end for**
      **end for**
    **end for**
  **end for**

### 6.2 Implementation of LPP algorithm

The implementation initially developed in [13] concerns precedence graphs with delays and identical machines. An extension to typed tasks systems is developed in [4].

Algorithm 3 is an extended version of the Leung-Palem-Pnueli algorithm. Consistent release dates $\mathbf{r}$ are first evaluated and tasks are numbered such that $r_1 \geq r_2 \geq \cdots \geq r_n$. This numbering insures the convergence of the algorithm. Each adjustement is evaluated using a double dichotomic search, leading to an algorithm of global complexity belonging to $\mathcal{O}(n^2 log(n)\alpha(n) + ne)$; $\alpha(n)$ is here the functional inverse of the Ackermann function. It is proved in [13] that, at the end of the algorithm, the computed due date vector is LPP-consistent.

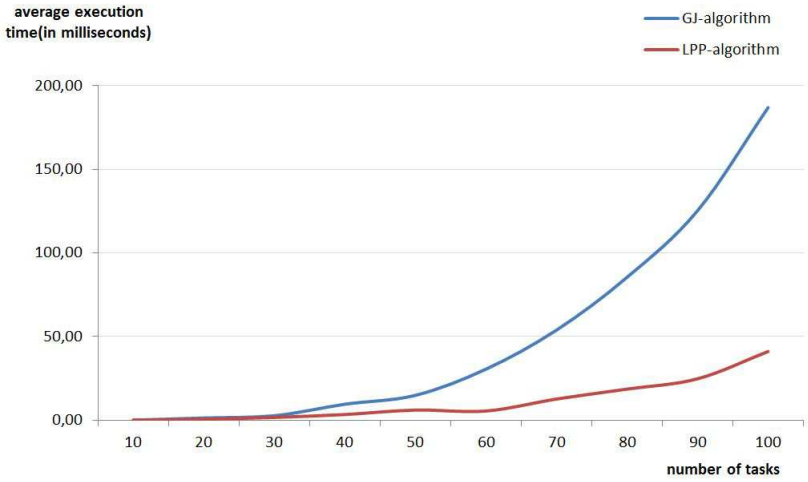
**Algorithm 3** Leung-Palem-Pnueli extended algorithm

**Require:** An instance of $P\Sigma^k|prec, \ell_{ij}, p_i = 1, r_i, d_i|\star$
**Ensure:** Modified due dates $\mathbf{D}^\star$
  Compute $\forall i \in \{1, \cdots, n\}$, $r_i = \max_{j \in \mathcal{P}_i}(r_i, r_j + 1 + \ell^+_{ji})$ following a topological order of $G$;
  Renumber tasks such as $r_1 \geq r_2 \geq \cdots \geq r_n$;
  Compute $\forall i \in \{1, \cdots, n\}$, $D_i = \min_{j \in \mathcal{S}_i}(D_i, D_j - 1 - \ell^+_{ij})$ following an inverse topological order;
  **for** $i = 1$ to $n$ **do**
    $D_i = \text{Adjust}(i, \mathbf{D})$;
    Compute $\forall i \in \{1, \cdots, n\}$, $D_i = \min_{j \in \mathcal{S}_i}(D_i, D_j - 1 - \ell^+_{ij})$ following an inverse topological order;
  **end for**

### 6.3 An experimental comparison of the performance for $m$ identical processors

The time complexity of the two previous implementation of GJ and LPP algorithms were compared for $m$ indentical processors and no delays (*i.e.* $\ell_{ij} = 0$ for any precedence arc $(i, j)$).

Fig. 4: Comparison of the execution time of LPP and GJ algorithms




Random instances were generated as follows. A fixed number of layer $k < n$, each of them with at least one task, allows to generate randomly acyclic precedence graphs. Release dates are them randomly fixed, and then updated by setting $r_i = \max(r_i, \max_{j \in \mathcal{P}_i} r_j + 1)$. Deadlines are lastly determined by $d_j = r_j + \alpha$, where $\alpha$ is a small random value.

Our main surprise was to notice that the deadlines fixed for most of the instances (about 90 percent) are consistent. We consider for our experiments only the remaining ones for which the deadlines are modified. Figure 4 reports the comparison between the execution times of the implementations of the LPP and the GJ algorithms. The performance of the LPP algorithm is clearly better and corresponds to the difference between the two theoretical worst-case complexities.

## 7 Theoretical outcomes of Theorem 1

The aim of this part is to exploit the equivalence between GJ and LPP algorithms. We start by breafly recalling that a polynomial time algorithm for a decision problem with due dates can be considered to minimize the makespan or the maximum lateness. Polynomial problem and approximation algorithms based on GJ or LPP algorithms are then listed.

### 7.1 Computation of a feasible schedule

As seen before, GJ and LPP algorithms are decreasing due dates following necessary conditions until a fixed point is reached. The second phase of the algorithm consists in building a schedule using a classical earliest due date first priority list algorithm. Whereas several tasks can be executed by a same resource, the selected task is the one with a minimum due date.

Schedules build are solving a decision problem. It can be extended easily to minimize the makespan or the maximum lateness. Indeed, assume that the decision problem considered can be solved polynomially for any values of the initial due dates.

1. By setting $d_i = C$ for any task $i \in \mathcal{T}$, LPP or GJ fixed point algorithms can either compute consistent due dates, or a contradiction is found. This process can be used with a dichotomic binary search to minimize $C$ leading to consistent due

dates. Whereas a minimum value $C$ is fixed, a list schedule using the corresponding consistent due dates builds a feasible schedule minimizing the makespan.

2. A similar scheme can be used to minimize the lateness. The idea is to minimize $\Delta$ by a dichotomic search such that a set of consistent due dates can be computed from due dates $d_i + \Delta$, $\forall i \in \mathcal{T}$. As before, a second step consists in building a feasible schedule using a list schedule algorithm for which priority of tasks decreases proportionaly to their (consistent) due date.

### 7.2 Polynomial time algorithms based on LPP consistency

LPP algorithm was considered to get polynomial time algorithms for several classes of instances, solving most of the polynomially solvable $m$ machines problems with precedence constraints. Table 1 summarizes the maximal polynomial problems solved by the LPP algorithm. By Theorem 1, these results also holds for GJ algorithm.

An outtree (*resp.* intree) is an acyclic precedence graph where each task has no more than one immediate predecessor (*resp.* successor). An interval order graph is an acyclic precedence graph $G = (\mathcal{T}, E)$ such that there exists a mapping $f$ from $\mathcal{T}$ to intervals on the real line with the following property: $(i, j) \in E$ if and only if $\forall x \in f(i)$ and $\forall y \in f(j)$, $x < y$. An interval order graph is monotone if for every pair of arcs $(i, j)$ and $(i, j')$ from $E$, $\ell_{ij} \leq \ell_{ij'}$ if $\mathcal{P}_j \subseteq \mathcal{P}_{j'}$.

| Problem | Ref. |
|---|---|
| $1\|prec, \ell_{ij} \in \{0,1\}, r_i, p_i = 1\|L_{max}$ | [13] |
| $1\|prec, \ell_{ij} \in \{0,1\}, p_i \geq 1\|C_{max}$ | [13] |
| $2\|prec, \ell_{ij} \in \{-1,0\}, r_i, p_i = 1\|L_{max}$ | [13] |
| $P\|intree, \ell_{ij} = \ell, p_i = 1\|L_{max}$ | [13] |
| $P\|outtree, \ell_{ij} = \ell, r_i, p_i = 1\|C_{max}$ | [13] |
| $P\Sigma^k\|int.order, monotone\ \ell_{ij}, r_i, p_i = 1\|L_{max}$ | [4] |

Table 1: Maximal polynomial problems solved by LPP and GJ algorithms.

### 7.3 Approximation algorithms based on GJ consistency

GJ consistency can be exploited to obtain approximation algorithms for minimizing the maximum lateness, using the scheme described previously. Theorem 3 was proved by Hanen and Zinder [8] using GJ algorithm and is true for the two optimization algorithms by Theorem 1.

**Theorem 3** *An upper-bound of the maximum lateness $L_{max}$ of the solution obtained using a LPP or GJ consistency-based algorithm for the problem $P|prec, p_i = 1, r_i, d_i|\ L_{max}$ is*

$$L_{max} \leq (2 - \frac{2}{u(m)})L_{max}(\sigma^\star) + (1 - \frac{2}{u(m)})d_{max} - (1 - \frac{2}{u(m)})$$

*where $\sigma^\star$ is an optimum schedule, $d_{max} = \max_{i \in \mathcal{T}} d_i$ and $u(m) = m$ if $m$ is even, $m+1$ if $m$ is odd.*

In the same way, Theorem 4 was proved by Benabid and Hanen [1] using initially GJ algorithm and is thus also valid for LPP algorithm:

**Theorem 4** *An upper-bound of the maximum lateness $L_{max}$ of the solution obtained using a LPP or GJ consistency-based algorithm for the problem $P\Sigma^k|prec, \ell_{ij} = \ell, p_i = 1, r_i, d_i|\ L_{max}$ is*

$$L_{max} \leq (k + 1 - \frac{\alpha(m_x)}{\ell + 1})L_{max}(\sigma^\star) + (k - \frac{\alpha(m_x)}{\ell + 1})d_{max} - \frac{\alpha(m_x)}{\ell + 1}$$

*where* $\sigma^\star$ *is an optimum schedule,* $d_{max} = \max_{i \in \mathcal{T}} d_i$, $m_x = \max_{r \in \{1,\cdots,k\}} m_r$ *and*

$$\alpha(m_x) = \begin{cases} \frac{2}{m_x} & \text{if } m_x \text{ is even} \\ \frac{2}{m_x+1} & \text{otherwise.} \end{cases}$$

## 8 Conclusion and further approach

We proved in this paper that GJ and LPP algorithms are equivalent for the general decision problem $P\Sigma^k|prec, \ell_{ij}, p_i = 1, r_i, d_i|\star$. This result allows us to unify prior results of the literature. Several polynomial subcases proved using LPP algorithm are thus also polynomially solved using GJ algorithm. Approximation results proved using GJ algorithms are valid for LPPs one. As the complexity of LPP is slightly better, it should be prefered if an implementation is needed.

This equivalence should be further investigated for variants of the initial problem as communication delays or preemptive tasks. We also hope that a careful investigation of the two algorithms may lead to a more efficient algorithm taking the best of both. One can also think that the due date modification algorithm could be used reversely to modify release times on the reverse graph, and iterate until a fixed point is reached. The question is then to evaluate if a better worst case performance ratio may be achieved for approximation algorithms.

## References


1. Benabid, A., Hanen, C.: Performance of garey johnson algorithm for pipelined type tasks systems. International Transactions on Operational Research **17**(6), 797–808 (2010)
2. Brucker, P., Knust, S.: Complexity results for single-machine problems with positive finish-start time-lags. Computing **63**(4), 299–316 (1999). DOI 10.1007/s006070050036. URL http://dx.doi.org/10.1007/s006070050036
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2 edn. The MIT Press (2001)
4. Dupont de Dinechin, B.: Scheduling monotone interval orders on typed task systems. In: PLANSIG 2007, 26th Worshop of the UK Planning and Scheduling Special Interest Group, pp. 25–31 (2007)
5. Garey, M.R., Johnson, D.S.: Two-processor scheduling with start-time and deadlines. SIAM Journal on Computing **6**, 416–426 (1977)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Studienreihe Informatik. W.H. Freeman and Company, San Francisco (1979)
7. Hanen, C., Zinder, Y.: The worst case analysis of garey-johnson algorithm for preemptive m processors. In: Multidisciplinary International Conference on Scheduling: Theory and Applications, vol. 2, pp. 453–470 (2005)
8. Hanen, C., Zinder, Y.: The worst-case analysis of the garey-johnson algorithm. Journal of Scheduling **12**(4), 389–400 (2009)
9. Hennessy, J.L., Gross, T.: Postpass code optimization of pipeline constraints. ACM Trans. Program. Lang. Syst. **5**(3), 422–448 (1983). DOI 10.1145/2166.357217. URL http://doi.acm.org/10.1145/2166.357217
10. Horn, W.: Some simple scheduling algorithms. Naval Research Logistics Quarterly **21**, 177–185 (1974)
11. Kordon, A.M., Kacem, F., Dupont de Dinechin, B., Finta, L.: Scheduling an interval ordered precedence graph with communication delays and a limited number of processors. RAIRO - Operations Research **47**, 73–87 (2013). DOI 10.1051/ro/2013028. URL http://www.rairo-ro.org/action/article_S0399055913000280
12. Lenstra, J., Kan, A.R., Brucker, P.: Complexity of machine scheduling problems. In: B.K. P.L. Hammer E.L. Johnson, G. Nemhauser (eds.) Studies in Integer Programming, *Annals of Discrete Mathematics*, vol. 1, pp. 343 – 362. Elsevier (1977). DOI http://dx.doi.org/10.1016/S0167-5060(08)70743-X. URL http://www.sciencedirect.com/science/article/pii/S016750600870743X
13. Leung, A., Palem, K.V., Pnueli, A.: Scheduling time-constrained instructions on pipelined processors. ACM Trans. Program. Lang. Syst. **23**, 73–103 (2001). DOI http://doi.acm.org/10.1145/383721.383733. URL http://doi.acm.org/10.1145/383721.383733
14. Palem, K.V., Simons, B.B.: Scheduling time-critical instructions on risc machines. ACM Trans. Program. Lang. Syst. **15**(4), 632–658 (1993). DOI 10.1145/155183.155190. URL http://doi.acm.org/10.1145/155183.155190
15. Ullman, J.D.: Np-complete scheduling problems. J. Comput. Syst. Sci. **10**(3), 384–393 (1975). DOI 10.1016/S0022-0000(75)80008-0. URL http://dx.doi.org/10.1016/S0022-0000(75)80008-0
16. Yu, W., Hoogeveen, H., Lenstra, J.K.: Minimizing makespan in a two-machine flow shop with delays and unit-time operations is np-hard. Journal of Scheduling **7**(5), 333–348 (2004)