The worst-case analysis of the Garey-Johnson Algorithm

Claire Hanen, Yakov Zinder

The Worst-case Analysis of the Garey-Johnson Algorithm

Claire Hanen, LIP6, Université Pierre et Marie Curie and
Université Paris Ouest-Nanterre-La défense, France
Yakov Zinder, Department of Mathematics,
Faculty of Science, University of Technology, Sydney, Australia

December 17, 2008

Abstract

The Garey-Johnson algorithm is a well known polynomial-time algorithm constructing an optimal schedule for the maximum lateness problem with unit execution time tasks, two parallel identical processors, precedence constraints and release times. The paper is concerned with the worst-case analysis of a generalization of the Garey-Johnson algorithm to the case of arbitrary number of processors. In contrast to other algorithms for the maximum lateness problem, the tight performance guarantee for the even number of processors differs from the tight performance guarantee for the odd number of processors.

keywords: scheduling,parallel machines, UET tasks, precedence constraints, maximum lateness

1 Introduction

In this paper we consider the problem of scheduling a set N={1,2,...,n} of n tasks (jobs, operations) on $m > 1$ parallel identical processors (machines) subject to precedence constraints in the form of an anti-reflexive, anti-symmetric and transitive relation on N. If task i precedes task j, denoted $i \rightarrow j$, then the processing of task i must be completed before the processing of task j begins. Each processor can process only one task at a time, and each task can be processed by any processor. Once a processor begins executing a task, it processes this task until its completion (i.e. no preemptions are allowed). Each task j requires one unit of processor's time and its processing can commence only after the specified non-negative integer release time r_j.

Since no preemptions are allowed and all processors are identical, any schedule σ can be determined by specifying for each task j its completion time $C_j(\sigma)$ in such a way that

- $C_j(\sigma) \geq r_j + 1$, for all $j \in N$;
- not more than m tasks are assigned the same completion time;
- if $i \rightarrow j$, then $C_j(\sigma) \geq C_i(\sigma) + 1$.

The goal is to find a schedule that minimizes the criterion of maximum lateness

$$L_{max}(\sigma) = \max_{j \in N}[C_j(\sigma) - d_j], \tag{1}$$

where d_j is an integer due date associated with task j.

In the three-field notation (see for example [2]), the above problem is denoted by $P|prec, p_j = 1, r_j|L_{max}$, where the terms $prec$ and r_j indicate

the presence of precedence constraints and release times, and $p_j = 1$ reflects the fact that all processing times are equal to one unit of time. If the partially ordered set of tasks is an in-tree, then the term $prec$ is replaced by $in-tree$. Analogously, the term $out-tree$ indicates that the partially ordered set of tasks is an out-tree. If all due dates are equal to zero, the maximum lateness problem becomes the makespan problem $P|prec, p_j = 1, r_j|C_{max}$ with the criterion

$$C_{max}(\sigma) = \max_{j \in N} C_j(\sigma).$$

If the term r_j is omitted, then all tasks have the same release time of zero.

It is well known that even $P|prec, p_j = 1|C_{max}$ is NP-hard in the strong sense [8], [7]. Moreover, as has been shown in [11], the $P|prec, p_j = 1|C_{max}$ problem remains NP-hard in the strong sense even if the partially ordered set of tasks is a bipartite graph. As far as the maximum lateness problem is concerned, $P|out-tree, p_j = 1|L_{max}$ is also NP-hard in the strong sense [3]. This implies the NP-hardness in the strong sense of $P|in-tree, p_j = 1, r_j|C_{max}$. These NP-hardness results boost the interest in the worst-case performance of various approximation algorithms [1], [9], [10], [12].

Notice that approximation analysis for L_{max} criteria can be done by two ways. Let σ be a schedule obtained by an algorithm, and σ^* denote the optimal schedule for a given instance. The first approach defines an upper bound on $L_{max}(\sigma) - L_{max}(\sigma^*)$. Relative approximation seems to be difficult since L_{max} criteria might have negative values. However, it is straightforward that $P|prec, p_j = 1, r_j|L_{max}$ is equivalent to the problem $P|prec, p_j = 1, r_j, q_j|G_{max}$, where each job j is associated with a release time r_j and a positive tail q_j, such that for a given schedule σ, $G_{max}(\sigma) = \max_{j \in N} C_j(\sigma) + q_j$. Indeed, from an instance of the L_{max} problem we define

an instance of the G_{max} problem by setting $\forall j \in N, q_j = \max_{i \in N} d_i - d_j$ (and conversely $d_j = \max_{i \in N} q_i - q_j$), so that for a given schedule σ, $L_{max}(\sigma) = G_{max}(\sigma) - \max_{i \in N} d_i$. Hence G_{max} is always positive and a worst case approximation ratio can be found:

$$G_{max}(\sigma) \leq \alpha G_{max}(\sigma^*) - \beta$$

This will lead to an expression for L_{max} as:

$$L_{max}(\sigma) \leq \alpha L_{max}(\sigma^*) + (\alpha - 1) \max_{i \in N} d_i - \beta$$

All performance guarantee that can be found in the litterature for L_{max}, have this form and the best known ratio is $\alpha = 2 - \frac{2}{m}$ [9][12].

This paper is concerned with the Garey-Johnson algorithm [4]. Although the Garey-Johnson algorithm was originally developed for the $P2|prec, p_j = 1, r_j|L_{max}$ problem, i.e. the problem with only two processors, it can be generalized to the case of arbitrary number of processors. We will refer to this generalization by GJ-algorithm. Although the Garey-Johnson algorithm is among the most popular scheduling algorithms, its worst-case performance in the case of arbitrary number of processors has remained unknown for almost three decades. In what follows we analyze the worst-case performance of the GJ-algorithm.

2 GJ-algorithm

In order to define what GJ-algorithm is, we need to introduce the framework on which it relies. Let us consider an instance of $P|prec, p_j = 1|L_{max}$, defined by an acyclic graph G, and for any node i, an integer d_i the due-date of task i. Notice that in the rest of the paper, g, h, i, j will always denote tasks.

Let us call $Decision(G, (d_i)_{i \in N})$ the following decision problem: does-it exists a schedule which meets all due-dates (called in-time schedule)?

The GJ-algorithm relies on an oracle $\mathcal{O}$ which answers partially this question: from original due-dates d_i, an algorithm $\mathcal{A}$ computes consistent modified due dates which should be met by any in-time schedule. If the algorithm fails, then there is no in-time schedule, so that the oracle answers "No". Otherwise, the true answer is still unknown, but the oracle answers "Yes". Consistent due-dates are defined in the first subsection.

Then, the GJ-algorithm will compute, by binary search, a minimum value Δ^* such that the oracle $\mathcal{O}$ answers yes to the problem $Decision(G, (\Delta^* + d_i)_{i \in N})$. It is proven in subsection 2.2 that Δ^* is a lower bound on the optimal lateness for the initial problem.

Finally, modified due-dates for the instance $(G, (\Delta^* + d_i)_{i \in N})$ are used as priorities to build a schedule of the tasks with a list scheduling algorithm. This part is described in subsection 2.3.

2.1 Consistent due dates

If $i \rightarrow j$ and $r_i \geq r_j$, then the replacement of r_j by $r_i + 1$ does not affect the feasibility of any schedule. Since one can recalculate all release times in $O(n^2)$ operations, without loss of generality we will assume that $i \rightarrow j$ implies the inequality $r_i + 1 \leq r_j$. Without loss of generality we will also assume that $\min_{j \in N} r_j = 0$.

Let D_1, ..., D_n be arbitrary nonnegative integers, then for any task i and any two numbers s and d such that $S(i, s, d)$ will denote the set of all tasks j with due date at most d ($D_j \leq d$) such that i precedes j according to G ($i \rightarrow j$), or $r_j \geq s$.

Intuitively if each task meets its due-date, then all tasks of $S(i,s,d)$ including i should end before d. Now assume that $r_i \leq s \leq D_i \leq d$. If $D_i = s$ then obviously, if i ends exactly at its due-date, all tasks of $S(i,s,d)$ should be computed in the time interval $[s,d)$. Similarly, if $D_i > s$ and if i ends at its due-date, all tasks of $S(i,s,d)$ and i should be performed in the time interval $[s,d)$. The notion of consistency derives from this intuition:

We will say that integers $D_1,\ldots,D_n$ are consistent if for every task j

$$r_j \leq D_j - 1, \tag{2}$$

And for any task i, and any integers s,d linked to i by the following relation:

$$r_i \leq s \leq D_i \leq d, \tag{3}$$

we have the following property:

$$\begin{cases} either & |S(i,s,d)| < m(d-s) \\ or & |S(i,s,d)| = m(d-s) \quad and\ D_i = s \end{cases} \tag{4}$$

It is easy to see that this definition is equivalent to the definition of consistency given in [4]. Indeed, according to [4], integers $D_1,\ldots,D_n$ are consistent if the inequality (2) holds for every $j \in N$, and for any task i and any two integers s and d satisfying (3), the inequality

$$|S(i,s,d)| \geq m(d-s) \tag{5}$$

implies

$$D_i \leq d - \left\lceil \frac{|S(i,s,d)|}{m} \right\rceil. \tag{6}$$

The equivalence of both definitions follows from the fact that the inequalities $|S(i,s,d)| > m(d-s)$ and (6) contradict (3), and that the equality $|S(i,s,d)| = m(d-s)$ together with (6) and (3) implies $D_i = s$.

Example: Let us consider the problem instance pictured in figure 1. And assume that $m = 7$. Let us consider the upper-left most node and call it a. Notice that $r_a = 0, d_a = 1$. Let us consider $s = 1, d = 2$. The set $S(a, 1, 2)$ comprises all nodes of the second row, since they all have a release date not less than 1 and a due date not more than 2. As there are $9 > 7$ nodes in the second row,the due-dates are not consistent.

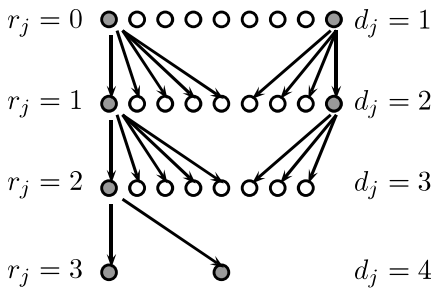


Figure 1: An example with inconsistent due-dates for $m = 7$.

Lemma 1 *If integers D_1, ..., D_n are consistent and $i \to j$, then $D_i \leq D_j - 1$.*

Proof

Suppose that $i \to j$ and $D_i \geq D_j$. Since D_1, ..., D_n are consistent, $r_i < D_i$, and analogously to [4], $s = d = D_i$ satisfy (3). For these s and d, $j \in S(i, s, d)$ and hence $|S(i, s, d)| > m(d - s)$, which contradicts the definition of consistency. □

2.2 Δ-modified due dates

Let Δ be an arbitrary integer. We will say that integers D_1, ..., D_n are Δ-modified due dates if they are consistent and $D_i \leq d_i + \Delta$ for all $i \in N$.

Example: Let us consider again the problem instance pictured in figure 1. Setting $D_j = d_j + 1 = i + 2$ for jobs j of row i (rows are indexed from 0 to 4) defines 1-modified due dates. One can easily check the new due-dates are consistent.

As has been shown in [4], there exists an algorithm which for any given Δ in $O(n^3)$ operations either calculates Δ-modified due dates or establishes that such due dates do not exist at all. The idea of this algorithm is based on the definition of consistency in [4] and can be outlined as follows. In order to compute for a given Δ a set of Δ-modified due dates we first set $D_j = d_j + \Delta$ for all $j \in N$. If (2) holds for all $j \in N$ and the inequality (6) holds for all triples (i, s, d) satisfying (3) and (5), integers D_1, ..., D_n themselves are Δ-modified due dates. If some j does not satisfy (2), the desired set of Δ-modified due dates does not exist at all. Suppose that $D_j \geq r_j + 1$ for all $j \in N$, but for some triple (i, s, d) satisfying (3) and (5)

$$D_i > d - \left\lceil \frac{|S(i,s,d)|}{m} \right\rceil . \tag{7}$$

It is easy to see that if Δ-modified due dates exist, then the due date associated with i is not greater than the right-hand side of the inequality (7). Hence we set

$$D_i = d - \left\lceil \frac{|S(i,s,d)|}{m} \right\rceil$$

and again check (2) and the inequality (6) for all triples (i, s, d) satisfying (3) and (5). At each such iteration we either conclude that the desired set of Δ-modified due dates does not exist, or establish that the current integers D_1, ..., D_n are Δ-modified due dates, or reduce value of some D_i. The result that this procedure terminates in $O(n^3)$ operations is based on three observations. First, that it suffices to consider only values of d coinciding with one of the current integers D_i. Second, that for a given d it suffices

to consider only values of s coinciding with one of release times or d itself. Third, that if the procedure is structured as three nested loops, where the outer loop selects d in decreasing order, the next loop selects i, and for fixed d and i the inner loop selects s in increasing order, each triple cannot appear in more than one iteration.

The following lemma shows that the existence of a schedule that meets due dates $d_1 + \Delta$, ..., $d_n + \Delta$ implies the existence of Δ-modified due dates.

Lemma 2 *Let σ be an arbitrary schedule and $D_j = C_j(\sigma)$, for all $j \in N$, then D_1, ..., D_n are consistent.*

Proof

Since σ is a feasible schedule, $r_j \leq C_j(\sigma) - 1 = D_j - 1$ for all $j \in N$. Suppose that some triple (i, s, d) satisfies (3) and the equality $|S(i, s, d)| = m(d - s)$. Then at least $(d - s) + 1$ time units are required to complete all tasks constituting the set $S(i, s, d) \cup \{i\}$. Since $C_j(\sigma) = D_j \leq d$ for each $j \in S(i, s, d) \cup \{i\}$, there exists a task $j \in S(i, s, d) \cup \{i\}$ such that $C_j(\sigma) \leq s$. From the definition of $S(i, s, d)$, either $i = j$ or $i \rightarrow j$. Therefore $D_i = C_i(\sigma) \leq s$, and by (3), $D_i = s$.

Now suppose that some triple (i, s, d) satisfies (3) and the inequality $|S(i, s, d)| > m(d - s)$. Then

$$d - \left\lceil \frac{|S(i, s, d)|}{m} \right\rceil \leq d - \frac{|S(i, s, d)|}{m} < s,$$

and since s and d are integers,

$$d - \left\lceil \frac{|S(i, s, d)|}{m} \right\rceil + 1 \leq s.$$

Because at least $\left\lceil \frac{|S(i, s, d)|}{m} \right\rceil$ time units are required to complete $|S(i, s, d)|$

tasks, there exists a task $j \in S(i,s,d)$ such that

$$C_j(\sigma) \leq d - \left\lceil \frac{|S(i,s,d)|}{m} \right\rceil + 1.$$

Since for this task $C_j(\sigma) \leq s$, $r_j < s$, and by the definition of $S(i,s,d)$, $i \rightarrow j$. Hence, $D_i = C_i(\sigma) < s$, which contradicts (3). ⌋

We will say that schedule σ is active if there is no schedule σ' such that $C_j(\sigma') \leq C_j(\sigma)$ for all $j \in N$ and at least one of these inequalities is strict. In what follows, for any integer t, we will refer to the time interval $[t-1, t]$ also as a time slot t.

Lemma 3 *If a schedule σ is active, then $C_j(\sigma) - r_j \leq n$ for all $j \in N$.*

Proof

Suppose that there exists a task j such that $C_j(\sigma) - r_j > n$. Then all m processors are idle in at least one time slot t satisfying the inequalities $r_j < t < C_j(\sigma)$. Because σ is active, this implies the existence of a task g such that $g \rightarrow j$ and $t < C_g(\sigma) < C_j(\sigma)$. Among all these tasks g select a task with the smallest completion time. Let it be task i. Since $i \rightarrow j$, $r_i < r_j < t$ and task i can be processed in the time slot t without changing completion times of all other tasks which contradicts the fact that σ is active. ⌋

The following lemma establishes upper and lower bounds on the optimal value of the criterion.

Lemma 4 *Let σ^* be an optimal schedule and Δ^* be the minimal Δ allowing Δ-modified due dates. Then*

$$\max_{i \in N}(r_i - d_i) < \Delta^* \leq L_{max}(\sigma^*) \leq n + \max_{i \in N}(r_i - d_i).$$

Proof

If $\Delta = r_j - d_j$ for some $j \in N$, then Δ-modified due dates $D_1, \ldots, D_n$ do not exist, because in this case

$$D_j \leq d_j + \Delta = r_j,$$

which contradicts the inequalities $D_j \geq r_j + 1$. Hence,

$$\max_{i \in N}(r_i - d_i) < \Delta^*.$$

Without loss of generality we assume that σ^* is active, because otherwise σ^* can be replace by an active schedule σ' such that $C_j(\sigma') \leq C_j(\sigma^*)$, for all $j \in N$, and hence also optimal. Consider any task j satisfying the equality $C_j(\sigma^*) - d_j = L_{max}(\sigma^*)$. Then using lemma 3 we have

$$L_{max}(\sigma^*) = C_j(\sigma^*) - r_j + r_j - d_j \leq n + r_j - d_j \leq n + \max_{i \in N}(r_i - d_i).$$

To complete the proof, we observe that $C_j(\sigma^*) \leq d_j + L_{max}(\sigma^*)$, for all $j \in N$, and therefore by lemma 2 the integers $C_1(\sigma^*)$, $C_2(\sigma^*)$, ..., $C_n(\sigma^*)$ are $L_{max}(\sigma^*)$-modified due dates. Hence, $\Delta^* \leq L_{max}(\sigma^*)$. □

2.3 GJ-algorithm for an arbitrary number of processors

The algorithm considered in this paper is a straightforward generalization of that presented in [4] for the two-processor case. Both algorithms use as a subroutine the so called list algorithm (see for example [2]). The list algorithm assumes that the tasks are arranged in a list $\mathcal{L}$ and constructs an active schedule $\sigma_{\mathcal{L}}$ iteratively from time $t = 0$, computing at each time t the set F_t of feasible jobs at time t according to the partial schedule of the interval $[0, t-1]$, schedule all of them at time t if $|F_t| \leq m$, otherwise choose

among F_t the m jobs with lowest index in $\mathcal{L}$ and schedule them at t. t is then updated to the next time slot in which a job is feasible.

As in lemma 4, let Δ^* be the minimal Δ allowing Δ-modified due dates.

GJ-algorithm

1. Set $\Delta_L = \max_{j \in N}(r_j - d_j)$ and $\Delta_U = n + \max_{j \in N}(r_j - d_j)$. Using the binary search, compute Δ^* and Δ^*-modified due dates.

2. Construct a list schedule for a list where tasks are arranged in the non-decreasing order of there Δ^*-modified due dates.

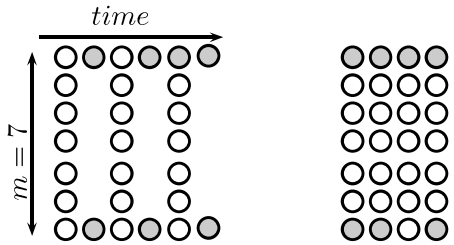
Figure 2 illustrates the schedule produced by GJ-algorithm applied to the graph of figure 1 on 7 processors, with $\Delta^* = 1$ in comparison to the optimal schedule.

Recall that, for a given Δ, the number of operations needed to compute Δ-modified due dates or to determine that these due dates do not exist is $O(n^3)$. The selection of Δ_U and Δ_L is justified by lemma 4. So, the calculation of Δ^* and the corresponding Δ^*-modified due dates can be accomplished in $O(n^3 \log_2 n)$ operations.

In what follows, σ^* will denote a schedule minimizing the maximum lateness, D_1, ..., D_n will denote Δ^*-modified due dates calculated in accord with the first step of the GJ-algorithm, and $\bar{\sigma}$ will denote a schedule constructed by this algorithm.

3 Decomposition procedure

In order to analyze the performance of the GJ-schedule, we use a decomposition procedure which will build a sequence $j_0, \ldots j_q$ of jobs and bounds on D_{j_k} that will be useful in the next sections.



GJ Schedule: $L_{max}(\bar{\sigma}) = 2$ Optimal schedule: $L_{max}(\sigma^*) = 1$

Figure 2: GJ and optimal schedule for the example of figure 1.

Decomposition of schedules is a classical tool to analyze their worst-case performance, see for example [2],[1] for makespan minimization analysis. However, with each algorithm a specific decomposition is needed to derive the performance bound.

For any integers t and D, denote by $\delta(t, D)$ the set of all tasks j such that $C_j(\bar{\sigma}) = t$ and $D_j \leq D$.

Lemma 5 *For any task j and any integers D and t, satisfying*

$$D_j \leq D, \quad r_j < t < C_q(\bar{\sigma}) \quad \textit{and} \quad |\delta(t, D)| < m,$$

there exists a task $h \in \delta(t, D-1)$ such that $h \rightarrow j$.

Proof

Since the schedule $\bar{\sigma}$ is constructed in accord with the list algorithm and since $|\delta(t, D)| < m$ and $r_j < t$, there exists a task i such that $C_i(\bar{\sigma}) \geq t$ and $i \rightarrow j$. Among all such tasks i select a task with the smallest completion time. Let it be task h. Lemma 1 implies that $D_h \leq D_j - 1 \leq D - 1$. Hence the lemma holds if $C_h(\bar{\sigma}) = t$. In order to prove this equality, assume that $C_h(\bar{\sigma}) > t$. The relation $h \rightarrow j$ implies $r_h < r_j$, and since the schedule $\bar{\sigma}$ was constructed in accord with the list algorithm, there exists a task f such

that $C_f(\bar{\sigma}) \geq t$ and $f \rightarrow h$. Then by transitivity $f \rightarrow j$, which contradicts the selection of h. $\square$

Further exploring the structure of $\bar{\sigma}$, we observe that

$$\max_{j \in N}[C_j(\bar{\sigma}) - D_j] \geq \max_{j \in N}[C_j(\bar{\sigma}) - (d_j + \Delta^*)] = L_{max}(\bar{\sigma}) - \Delta^*$$

and by lemma 4,

$$\geq L_{max}(\bar{\sigma}) - L_{max}(\sigma^*) \geq 0.$$

Hence, if

$$\max_{j \in N}[C_j(\bar{\sigma}) - D_j] = 0,$$

then $\bar{\sigma}$ is an optimal schedule for the original problem. Since our goal is the worst-case analysis of the GJ-algorithm, we will assume that there exists a task g such that

$$C_g(\bar{\sigma}) - D_g > 0. \tag{8}$$

The following procedure, which will be referred to as a decomposition procedure, constructs for any task g, satisfying (8), a sequence of tasks $j_0 = g$, ..., $j_{l(g)}$ and the sequence of corresponding sets of tasks M^0, ..., $M^{l(g)}$. Suppose that the sequence j_0, ..., j_i and the corresponding sequence of sets M^0, ..., M^{k-1} have been already constructed. Obviously, if $k = 0$, no sets have been constructed yet. Let t be an integer such that $t < C_{j_k}(\bar{\sigma})$ and $|\delta(t, D_{j_k})| < m$. Observe that both inequalities hold for example for $t = 0$. Among all such t select the largest one and denote it by τ. Then

$$M^k = \cup_{\tau < l \leq C_{j_k}(\bar{\sigma})} \delta(t, D_{j_k}).$$

If $r_h \geq \tau$ for all $h \in M^k$, then the procedure terminates with $l(g) = k$. If $r_h < \tau$ for at least one $h \in M^k$, then according to lemma 5 $\delta(\tau, D_{j_k} - 1) \neq \emptyset$.

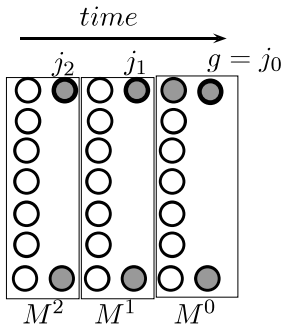


Figure 3: Decomposition of the GJ-schedule of figure 2.

If $|\delta(\tau, D_{j_k} - 1)| = 1$, then the procedure terminates with $l(g) = k$. In this case, the task constituting the set $\delta(\tau, D_{j_k} - 1)$ will be denoted by $a(g)$. If $|\delta(\tau, D_{j_k} - 1)| \geq 2$, then choose as j_{k+1} any task h satisfying

$$D_h = \max_{i \in \delta(\tau, D_{j_k} - 1)} D_i$$

and start a new iteration by constructing the set M^{k+1}.

Figure 3 illustrates the decomposition, in a case where $a(g)$ does not exists. The following lemmas express the properties of the decomposition, firstly by defining lower bounds on release times of tasks of subsets M^k(lemmas 8 and 7) and then a lower bound on the due-date of g (lemma 8).

Lemma 6 *If task g satisfies (8) and the decomposition procedure cannot determine $a(g)$, then*

$$\min_{i \in \cup_{k=0}^{l(g)} M^k} r_i = \min_{i \in M^{l(g)}} C_i(\bar{\sigma}) - 1.$$

Proof

Since $a(g)$ does not exist, according to the decomposition procedure

$$\min_{i \in M^{l(g)}} r_i = \min_{i \in M^{l(g)}} C_i(\bar{\sigma}) - 1. \tag{9}$$

Suppose that for some $k < l(g)$ there exists $j \in M^k$ such that

$$r_j < \min_{i \in M^{l(g)}} C_i(\bar{\sigma}) - 1.$$

Then by lemma 5 there exists task $h \in M^{l(g)}$ such that $h \to j$. Since $h \to j$ implies $r_h < r_j$,

$$r_h < \min_{i \in M^{l(g)}} C_i(\bar{\sigma}) - 1,$$

which contradicts (9). □

Lemma 7 *Let g satisfy (8) and let, in the corresponding decomposition, $j \in M^k$, then*

$$r_j \geq \begin{cases} r_{a(g)} + l(g) - k + 1 & \text{if } a(g) \text{ exists} \\ l(g) - k & \text{if } a(g) \text{ does not exist} \end{cases} \tag{10}$$

Proof

Suppose that $a(g)$ exists and $k = l(g)$. If

$$r_j < \min_{i \in M^{l(g)}} C_i(\bar{\sigma}) - 1,$$

then according to the decomposition procedure and lemma 5, $a(g) \to j$. Hence, $r_j \geq r_{a(g)} + 1$ and the lemma holds. If

$$r_j \geq \min_{i \in M^{l(g)}} C_i(\bar{\sigma}) - 1,$$

then the same result follows from the observation that

$$\min_{i \in M^{l(g)}} C_i(\bar{\sigma}) - 1 = C_{a(g)}(\bar{\sigma}) \geq r_{a(g)} + 1.$$

Since $\min_{u \in N} r_u = 0$, the lemma also holds if $a(g)$ does not exist.

Suppose that the lemma holds for any $k > w$ for some nonnegative integer $w < l(g)$. Let $k = w$, then by the assumption, for any $u \in$

$\delta(C_{j_{k+1}}(\bar{\sigma}), D_{j_k} - 1)$

$$r_u \geq \begin{cases} r_{a(g)} + l(g) - (k+1) + 1 & \text{if } a(g) \text{ exists} \\ l(g) - (k+1) & \text{if } a(g) \text{ does not exist} \end{cases}$$

If $r_j < C_{j_{k+1}}(\bar{\sigma})$, then by lemma 5 there exists $h \in \delta(C_{j_{k+1}}(\bar{\sigma}), D_{j_k} - 1)$ such that $h \rightarrow j$ and therefore

$$r_j \geq C_h(\bar{\sigma}) \geq r_h + 1 \geq \begin{cases} r_{a(g)} + l(g) - k + 1 & \text{if } a(g) \text{ exists} \\ l(g) - k & \text{if } a(g) \text{ does not exist} \end{cases}$$

If $r_j \geq C_{j_{k+1}}(\bar{\sigma})$, then (10) follows from $r_j \geq C_{j_{k+1}}(\bar{\sigma}) \geq r_{j_{k+1}} + 1$. Hence, the lemma holds for any k. □

Lemma 8 *If task g satisfies (8), then*

$$D_g \geq \begin{cases} D_{a(g)} + l(g) + 1 & \textit{if } a(g) \textit{ exists} \\ l(g) + 2 & \textit{if } a(g) \textit{ does not exist} \end{cases}$$

Proof

If $a(g)$ exists, then the inequality

$$D_g \geq D_{a(g)} + l(g) + 1$$

follows from the fact that according to the decomposition procedure $D_{j_k} \geq D_{j_{k+1}} + 1$ for all $0 \leq k \leq l(g)$.

Suppose that $a(g)$ does not exist, then by lemma 7 and the condition (2) of consistency, $D_g \geq l(g) + 1$. Suppose that the lemma does not hold, i.e. suppose that $D_g = l(g) + 1$. Then, taking into account that according to the decomposition procedure $D_{j_k} \geq D_{j_{k+1}} + 1$ for all $0 \leq k \leq l(g)$, $D_{j_{l(g)}} = 1$ and $r_{j_{l(g)}} = 0$. If $|M^k| \leq m$ for all $0 \leq k \leq l(g)$, then all tasks constituting each M^k are processed in the same time slot, and therefore $C_g(\bar{\sigma}) = l(g) + 1$ which contradicts (8).

Let k be the largest l among all l satisfying $|M^l| > m$. In order to show that $k = l(g)$, assume that $k < l(g)$. Then, for each $k < l \leq l(g)$, all tasks constituting M^l are processed in the same time slot $C_{j_l}(\bar{\sigma}) = l(g) - l + 1$. Hence,

$$\min_{h \in M^k} C_h(\bar{\sigma}) - 1 = C_{j_{k+1}}(\bar{\sigma}) = l(g) - k.$$

This by virtue of lemma 7 implies that for any $j \in M^k$

$$r_j \geq \min_{h \in M^k} C_h(\bar{\sigma}) - 1,$$

which contradicts the decomposition procedure which should terminate after the construction of set M^k. Hence, $k = l(g)$. Let $s = 0 = r_{j_{l(g)}}$ and $d = 1 = D_{j_{l(g)}}$. Then, since $|\delta(C_{j_{l(g)}}(\bar{\sigma}), D_{j_{l(g)}})| \geq 2$,

$$|S(j_{l(g)}, s, d)| \geq |M^{l(g)} - \{j_{l(g)}\}| > m(d - s)$$

which contradicts the fact that the Δ^*-modified due dates are consistent. ⌋

4 Completion times in $\bar{\sigma}$ and Δ^*-modified due dates

In this section we derive bounds on C_g from the decomposition procedure defined in the previous section (lemmas 10 and 11.)

It is convenient to introduce the following notation:

$$\alpha(m) = \begin{cases} \dfrac{2}{m+1} & \text{if} \quad m \quad \text{is odd} \\ \dfrac{2}{m} & \text{if} \quad m \quad \text{is even} \end{cases}.$$

Lemma 9 *For any positive integer p*

$$\left\lceil \frac{2p-1}{m} \right\rceil \geq \alpha(m)p.$$

Proof

Let m be even. Since $2p-1$ is odd, $\frac{2p-1}{m}$ cannot be integer, and therefore

$$\left\lceil \frac{2p-1}{m} \right\rceil = \left\lceil \frac{2p-1}{m} + \frac{1}{m} \right\rceil \geq \frac{2p}{m} = \alpha(m)p.$$

If m is odd and $\frac{2p-1}{m} < 1$, then $\frac{2p-1}{m} + \frac{1}{m} < 1$, and therefore

$$\left\lceil \frac{2p-1}{m} \right\rceil = \left\lceil \frac{2p-1}{m} + \frac{1}{m} \right\rceil > \frac{2p}{m} > \alpha(m)p.$$

Finally, let m be odd and $\frac{2p-1}{m} \geq 1$, then $\frac{2p-1}{m} \geq \frac{2p}{m+1}$, and therefore

$$\left\lceil \frac{2p-1}{m} \right\rceil \geq \left\lceil \frac{2p}{m+1} \right\rceil \geq \frac{2p}{m+1} = \alpha(m)p$$

which completes the proof. $\square$

Lemma 10 *If task g satisfies (8) and $a(g)$ does not exist, then*

$$C_g(\bar{\sigma}) \leq [1-\alpha(m)][l(g)+2] + D_g - [1-\alpha(m)].$$

Proof

From the construction of $M^{l(g)}$, there exists a task $h \in M^{l(g)}$ such that

$$r_h = \min_{j \in M^{l(g)}} C_j(\bar{\sigma}) - 1.$$

Let $s = r_h$ and $d = D_g$, then by lemma 6 and the fact that $D_j \leq D_g$, for all $j \in \cup_{i=0}^{l(g)} M^i$,

$$r_h = s < D_h \leq d \quad \text{and} \quad (\cup_{k=0}^{l(g)} M^k - \{h\}) \subseteq S(h,s,d).$$

If $|S(h,s,d)| = m(d-s)$, then by the consistency of the Δ^*-modified due dates $D_h = s$, which contradicts the inequality $s < D_h$. Therefore $|S(h,s,d)| < m(d-s)$. On the other hand, each time slot $C_{j_i}(\bar{\sigma})$, where $1 \leq i \leq l(g)$, contains at least two tasks from M^i, and any other time slot

t, satisfying the inequalities $r_h < t < C_g(\bar{\sigma})$, contains exactly m tasks from $\cup_{k=0}^{l(g)} M^k$. Consequently,

$$m[C_g(\bar{\sigma}) - r_h - l(g) - 1] + 2l(g) \leq |S(h, s, d)| < m(d - s) = m(D_g - r_h).$$

Hence

$$C_g(\bar{\sigma}) < l(g) + 1 - \frac{2l(g)}{m} + D_g. \tag{11}$$

If m is even, then since $2l(g)$ is also even and (11) is a strict inequality,

$$C_g(\bar{\sigma}) \leq l(g) + 1 - \frac{2l(g)}{m} + D_g - \frac{2}{m}.$$

Consequently,

$$C_g(\bar{\sigma}) \leq l(g)+2-\frac{2[l(g)+2]}{m}+D_g-1+\frac{2}{m} = [1-\alpha(m)][l(g)+2]+D_g-[1-\alpha(m)].$$

If m is odd, then from (11)

$$C_g(\bar{\sigma}) < l(g) + 1 - \frac{2l(g)}{m+1} + D_g,$$

and because both $2l(g)$ and $m+1$ are even

$$C_g(\bar{\sigma}) \leq l(g) + 1 - \frac{2l(g)}{m+1} + D_g - \frac{2}{m+1}.$$

Hence

$$C_g(\bar{\sigma}) \leq l(g) + 2 - \frac{2[l(g)+2]}{m+1} + D_g - 1 + \frac{2}{m+1} = [1 - \alpha(m)][l(g) + 2]$$

$$+D_g - [1 - \alpha(m)]$$

which completes the proof. □

Lemma 11 *If task g satisfies (8) and $a(g)$ exists, then*

$$C_g(\bar{\sigma}) - C_{a(g)}(\bar{\sigma}) \leq [1 - \alpha(m)][l(g) + 1] + D_g - \min[C_{a(g)}(\bar{\sigma}), D_{a(g)}].$$

Proof

According to the decomposition procedure $D_{j_i} \leq D_{j_{i-1}} - 1$ for all $1 \leq i \leq l(g) + 1$. Adding all these inequalities, we have

$$D_{a(g)} \leq D_{j_0} - l(g) - 1,$$

which gives

$$D_g - \min[C_{a(g)}(\bar{\sigma}), D_{a(g)}] \geq D_g - D_{a(g)} \geq l(g) + 1. \tag{12}$$

Because the Δ^*-modified due dates are consistent, $r_{a(g)} < D_{a(g)}$. Let $d = D_g$ and $s = \min[C_{a(g)}(\bar{\sigma}), D_{a(g)}]$, then

$$r_{a(g)} < s \leq D_{a(g)} < d.$$

By lemma 5 and the decomposition procedure, for any $j \in \cup_{k=0}^{l(g)} M^k$ either $r_j \geq C_{a(g)} \geq s$ or $a(g) \to j$. Moreover, by the decomposition procedure, $D_j \leq D_g = d$ for all $j \in \cup_{k=0}^{l(g)} M^k$. Hence

$$\cup_{k=0}^{l(g)} M^k \subseteq S(a(g), s, d).$$

On the other hand, each time slot $C_{j_k}(\bar{\sigma})$, where $1 \leq k \leq l(g)$, contains at least two tasks from M^k, and any other time slot t, satisfying the inequalities $C_{a(g)}(\bar{\sigma}) < t < C_g(\bar{\sigma})$, contains exactly m tasks from $\cup_{k=0}^{l(g)} M^k$. Consequently,

$$|S(a(g), s, d)| \geq m[C_g(\bar{\sigma}) - C_{a(g)}(\bar{\sigma}) - l(g) - 1] + 2l(g) + 1. \tag{13}$$

If $|S(a(g), s, d)| < m(d - s)$, then taking into account that d and s are integers, we have

$$\left\lceil \frac{|S(a(g), s, d)|}{m} \right\rceil \leq d - s = D_g - \min[C_{a(g)}(\bar{\sigma}), D_{a(g)}].$$

If $|S(a(g), s, d)| = m(d - s)$, then

$$\min[C_{a(g)}(\bar{\sigma}), D_{a(g)}] = s = d - \left\lceil \frac{|S(a(g), s, d)|}{m} \right\rceil = D_g - \left\lceil \frac{|S(a(g), s, d)|}{m} \right\rceil.$$

Therefore in both cases

$$D_g - \min[C_{a(g)}(\bar{\sigma}), D_{a(g)}] \geq \left\lceil \frac{|S(a(g), s, d)|}{m} \right\rceil$$

and using (13) and lemma 9

$$\geq C_g(\bar{\sigma}) - C_{a(g)}(\bar{\sigma}) - l(g) - 1 + \left\lceil \frac{2l(g) + 1}{m} \right\rceil = C_g(\bar{\sigma}) - C_{a(g)}(\bar{\sigma})$$

$$-l(g) - 1 + \left\lceil \frac{2[l(g) + 1] - 1}{m} \right\rceil \geq C_g(\bar{\sigma}) - C_{a(g)}(\bar{\sigma}) - [1 - \alpha(m)][l(g) + 1].$$

Hence

$$C_g(\bar{\sigma}) - C_{a(g)}(\bar{\sigma}) \leq [1 - \alpha(m)][l(g) + 1] + D_g - \min[C_{a(g)}(\bar{\sigma}), D_{a(g)}].$$

which completes the proof. □

5 Performance guarantees

We now have enough material to prove the performance guarantee of GJ algorithm. In the last subsction, we show that the bounds are tight.

5.1 Worst case performance analysis

In order to use the lemma of the previous sections, we need a second decomposition procedure, which will produce a sequence of jobs $g_0, \ldots, g_k$ with interesting properties.

Suppose that $L_{max}(\bar{\sigma}) > L_{max}(\sigma^*)$. Let g_0 be any task satisfying the equality

$$C_{g_0}(\bar{\sigma}) - d_{g_0} = L_{max}(\bar{\sigma}). \tag{14}$$

By the definition of Δ^*-modified due dates and lemma 4

$$L_{max}(\bar{\sigma}) = C_{g_0}(\bar{\sigma}) - d_{g_0} \leq C_{g_0}(\bar{\sigma}) - (D_{g_0} - \Delta^*) \leq C_{g_0}(\bar{\sigma}) - D_{g_0} + L_{max}(\sigma^*),$$

which implies $C_{g_0}(\bar{\sigma}) - D_{g_0} > 0$. Using the decomposition procedure and starting with g_0 we can construct a sequence of tasks as follows:

- if $a(g_0)$ - given by the first decomposition of the previous section - does not exist, then this sequence contains only one task g_0;
- if $a(g_0)$ exists, then repeatedly applying the decomposition procedure, we get a sequence of tasks $g_0, \ldots, g_k$ such that $C_{g_i}(\bar{\sigma}) - D_{g_i} > 0$ and $g_{i+1} = a(g_i)$ for all $0 \leq i < k$, and either $C_{g_k}(\bar{\sigma}) - D_{g_k} \leq 0$ or $C_{g_k}(\bar{\sigma}) - D_{g_k} > 0$ but $a(g_k)$ does not exist.

Theorem 1 *For any* $m \geq 2$

$$L_{max}(\bar{\sigma}) - L_{max}(\sigma^*) \leq [1 - \alpha(m)][1 + \max_{j \in N} r_j]. \tag{15}$$

Proof

Suppose that $a(g_0)$ does not exist. Then taking into account lemma 10, lemma 7 and lemma 4,

$$C_{g_0}(\bar{\sigma}) \leq [1 - \alpha(m)][l(g_0) + 1] + D_{g_0} \leq [1 - \alpha(m)][r_{g_0} + 1] + d_{g_0} + \Delta^*$$

$$\leq [1 - \alpha(m)][1 + \max_{j \in N} r_j] + d_{g_0} + L_{max}(\sigma^*)$$

which by virtue of (14) implies (15).

Suppose that $k \geq 1$, $C_{g_k}(\bar{\sigma}) - D_{g_k} > 0$ and $a(g_k)$ does not exist. Then by lemma 11 and lemma 7, for all $0 \leq i \leq k-1$,

$$C_{g_i}(\bar{\sigma}) - C_{g_{i+1}}(\bar{\sigma}) \leq [1 - \alpha(m)][l(g_i) + 1] + D_{g_i} - \min[C_{g_{i+1}}(\bar{\sigma}), D_{g_{i+1}}]$$

$$= [1-\alpha(m)][l(g_i)+1] + D_{g_i} - D_{g_{i+1}} \leq [1-\alpha(m)][r_{g_i} - r_{g_{i+1}}] + D_{g_i} - D_{g_{i+1}}.$$

Adding inequalities

$$C_{g_i}(\bar{\sigma}) - C_{g_{i+1}}(\bar{\sigma}) \leq [1-\alpha(m)][r_{g_i} - r_{g_{i+1}}] + D_{g_i} - D_{g_{i+1}} \tag{16}$$

for all $0 \leq i \leq k-1$, we have

$$C_{g_0}(\bar{\sigma}) - C_{g_k}(\bar{\sigma}) \leq [1-\alpha(m)][r_{g_0} - r_{g_k}] + D_{g_0} - D_{g_k}. \tag{17}$$

On the other hand, by lemma 10 and lemma 7

$$C_{g_k}(\bar{\sigma}) \leq [1-\alpha(m)][l(g_k)+1] + D_{g_k} \leq [1-\alpha(m)][r_{g_k}+1] + D_{g_k}.$$

This together with (17) and lemma 4 gives

$$C_{g_0}(\bar{\sigma}) \leq [1-\alpha(m)][r_{g_0}+1] + D_{g_0} \leq [1-\alpha(m)][1+\max_{j\in N} r_j] + d_{g_0} + \Delta^*$$

$$\leq [1-\alpha(m)][1+\max_{j\in N} r_j] + d_{g_0} + L_{max}(\sigma^*)$$

which implies (15).

Suppose that $k \geq 1$, $C_{g_k}(\bar{\sigma}) - D_{g_k} \leq 0$. If $k \geq 2$, then by adding inequalities (16) for all $0 \leq i \leq k-2$, we obtain

$$C_g(\bar{\sigma}) - C_{g_{k-1}}(\bar{\sigma}) \leq [1-\alpha(m)][r_{g_0} - r_{g_{k-1}}] + D_{g_0} - D_{g_{k-1}}. \tag{18}$$

It is easy to see that (18) also holds when $k = 1$. On the other hand, by lemma 11, the fact that $C_{g_k}(\bar{\sigma}) - D_{g_k} \leq 0$ and lemma 7,

$$C_{g_{k-1}}(\bar{\sigma}) - C_{g_k}(\bar{\sigma}) \leq [1-\alpha(m)][l(g_{k-1})+1] + D_{g_{k-1}} - \min[C_{g_k}(\bar{\sigma}), D_{g_k}]$$

$$\leq [1-\alpha(m)][r_{g_{k-1}} - r_{g_k}] + D_{g_{k-1}} - C_{g_k}(\bar{\sigma})$$

which together with (18) and lemma 4 gives

$$C_{g_0}(\bar{\sigma}) \leq [1-\alpha(m)][r_{g_0} - r_{g_k}] + D_{g_0} \leq [1-\alpha(m)][1+\max_{j\in N} r_j] + d_{g_0} + \Delta^*$$

$$\leq [1-\alpha(m)][1+\max_{j\in N} r_j] + d_{g_0} + L_{max}(\sigma^*)$$

which implies (15). □

Theorem 2 *For any* $m \geq 2$

$$L_{max}(\bar{\sigma}) \leq [2-\alpha(m)]L_{max}(\sigma^*) + [1-\alpha(m)]\max_{v\in N} d_v - [1-\alpha(m)]. \quad (19)$$

Proof

Let j be an arbitrary task. Since $r_j \geq 0$ and therefore $C_j(\sigma^*) \geq 1$,

$$L_{max}(\sigma^*) \geq C_j(\sigma^*) - d_j \geq 1 - \max_{v\in N} d_v.$$

Hence, $L_{max}(\sigma^*) + \max_{v\in N} d_v \geq 1$, and by virtue of $1-\alpha(m) \geq 0$

$$[1-\alpha(m)][L_{max}(\sigma^*) + \max_{v\in N} d_v] - [1-\alpha(m)] \geq 0.$$

Hence, if $L_{max}(\bar{\sigma}) = L_{max}(\sigma^*)$, then (19) holds.

Suppose that $L_{max}(\bar{\sigma}) > L_{max}(\sigma^*)$. Let g_0 be a task, satisfying the equality

$$C_{g_0}(\bar{\sigma}) - d_{g_0} = L_{max}(\bar{\sigma}),$$

and let $g_0, \ldots, g_k$ be the sequence of tasks build according to the second decomposition procedure, such that $C_{g_i}(\bar{\sigma}) - D_{g_i} > 0$ and $g_{i+1} = a(g_i)$ for all $0 \leq i < k$, and either $C_{g_k}(\bar{\sigma}) - D_{g_k} \leq 0$ or $C_{g_k}(\bar{\sigma}) - D_{g_k} > 0$ but $a(g_k)$ does not exist.

If $C_{g_k}(\bar{\sigma}) - D_{g_k} \leq 0$, then $k \geq 1$. For all $0 \leq i \leq k-1$, by lemma 11 and lemma 8

$$C_{g_i}(\bar{\sigma}) - C_{g_{i+1}}(\bar{\sigma}) \leq [1-\alpha(m)][l(g_i)+1] + D_{g_i} - \min[C_{g_{i+1}}(\bar{\sigma}), D_{g_{i+1}}]$$

$$\leq [1-\alpha(m)][D_{g_i} - D_{g_{i+1}}] + D_{g_i} - \min[C_{g_{i+1}}(\bar{\sigma}), D_{g_{i+1}}]$$

$$= [2 - \alpha(m)]D_{g_i} - [1 - \alpha(m)]D_{g_{i+1}} - \min[C_{g_{i+1}}(\bar{\sigma}), D_{g_{i+1}}]$$

$$\leq [2 - \alpha(m)]\{D_{g_i} - \min[C_{g_{i+1}}(\bar{\sigma}), D_{g_{i+1}}]\}.$$

Adding all these inequalities and taking into account that $C_{g_k}(\bar{\sigma}) \leq D_{g_k}$ and $C_{g_{i+1}}(\bar{\sigma}) > D_{g_{i+1}}$ if $i + 1 < k$, we have

$$C_{g_0}(\bar{\sigma}) - C_{g_k}(\bar{\sigma}) \leq [2 - \alpha(m)]\{D_{g_0} - C_{g_k}(\bar{\sigma})\},$$

and since $C_{g_k}(\bar{\sigma}) \geq 1$,

$$C_{g_0}(\bar{\sigma}) \leq [2 - \alpha(m)]D_{g_0} - [1 - \alpha(m)]. \tag{20}$$

Suppose that $C_{g_k}(\bar{\sigma}) - D_{g_k} > 0$, then $a(g_k)$ does not exist. By lemma 10 and lemma 8

$$C_{g_k}(\bar{\sigma}) \leq [2 - \alpha(m)]D_{g_k} - [1 - \alpha(m)]. \tag{21}$$

If $k = 0$, then (21) coincides with (20). Let $k \geq 1$. For all $0 \leq i \leq k - 1$, by lemma 11 and the fact that $C_{g_{i+1}}(\bar{\sigma}) > D_{g_{i+1}}$

$$C_{g_i}(\bar{\sigma}) - C_{g_{i+1}}(\bar{\sigma}) \leq [2 - \alpha(m)]\{D_{g_i} - D_{g_{i+1}}\},$$

Adding all these inequalities, we have

$$C_q g_0(\bar{\sigma}) - C_{g_k}(\bar{\sigma}) \leq [2 - \alpha(m)]\{D_{g_0} - D_{g_k}\},$$

which together with (21) gives (20).

Using (20),

$$L_{max}(\bar{\sigma}) = C_q g_0(\bar{\sigma}) - d_{g_0} \leq [2 - \alpha(m)]D_{g_0} - [1 - \alpha(m)] - d_{g_0}$$

and by the definition of Δ^*-modified due dates and lemma 4

$$\leq [2 - \alpha(m)](d_{g_0} + \Delta^*) - [1 - \alpha(m)] - d_{g_0}$$

$$\leq [2 - \alpha(m)]L_{max}(\sigma^*) + [1 - \alpha(m)] \max_{j \in N} d_j - [1 - \alpha(m)]$$

which completes the proof. $\square$

5.2 Tightness

In order to show that, for any $m \geq 5$, (15) and (19) are asymptotically tight, we will consider graphs $G_{x,m}$, each comprising $m \cdot x \cdot u_m$ nodes which form $x \cdot u_m$ rows, where x is a positive integer and

$$u_m = \begin{cases} \dfrac{m+1}{2} & \text{if } m \text{ is odd} \\ \dfrac{m}{2} & \text{if } m \text{ is even} \end{cases} \tag{22}$$

The rows of nodes are numbered from 0 to $x \cdot u_m - 1$ (see figure 4), and

$$[\text{the number of nodes in row } i] = \begin{cases} m+2 & \text{if} \quad i \bmod u_m \leq u_m - 3 \\ m+1 & \text{if} \quad i \bmod u_m = u_m - 2 \\ 2 & \text{if} \quad i \bmod u_m = u_m - 1 \end{cases}$$

Each graph $G_{x,m}$ represents a partially ordered set of tasks, where each node represents a task and the arcs represent precedence constraints. All tasks corresponding to row i have the same release time equals i and the same due-date equals $d_i = i + 2$. We will use the following notation:

- If $i \bmod u_m \leq u_m - 3$ then the $m+2$ nodes, constituting row i, will be denoted by $a_i^1, a_i^2, b_i^1, \ldots, b_i^m$.
- If $i \bmod u_m = u_m - 2$ then the $m+1$ nodes of row i will be denoted by $a_i^1, b_i^1, \ldots, b_i^m$.
- If $i \bmod u_m = u_m - 1$ then the two nodes, constituting row i, will be denoted by a_i^1 and a_i^2.

In figure 4, nodes a_i^1 and a_i^2 are shaded. Only nodes a_i^1 and a_i^2 have successors (see figure 4):

- If $m \geq 7$ and $i \bmod u_m < u_m - 3$, then a_i^1 precedes $a_{i+1}^1, b_{i+1}^1, \ldots, b_{i+1}^{u_m}$ and a_i^2 precedes $a_{i+1}^2, b_{i+1}^{u_m+1}, \ldots, b_{i+1}^m$.

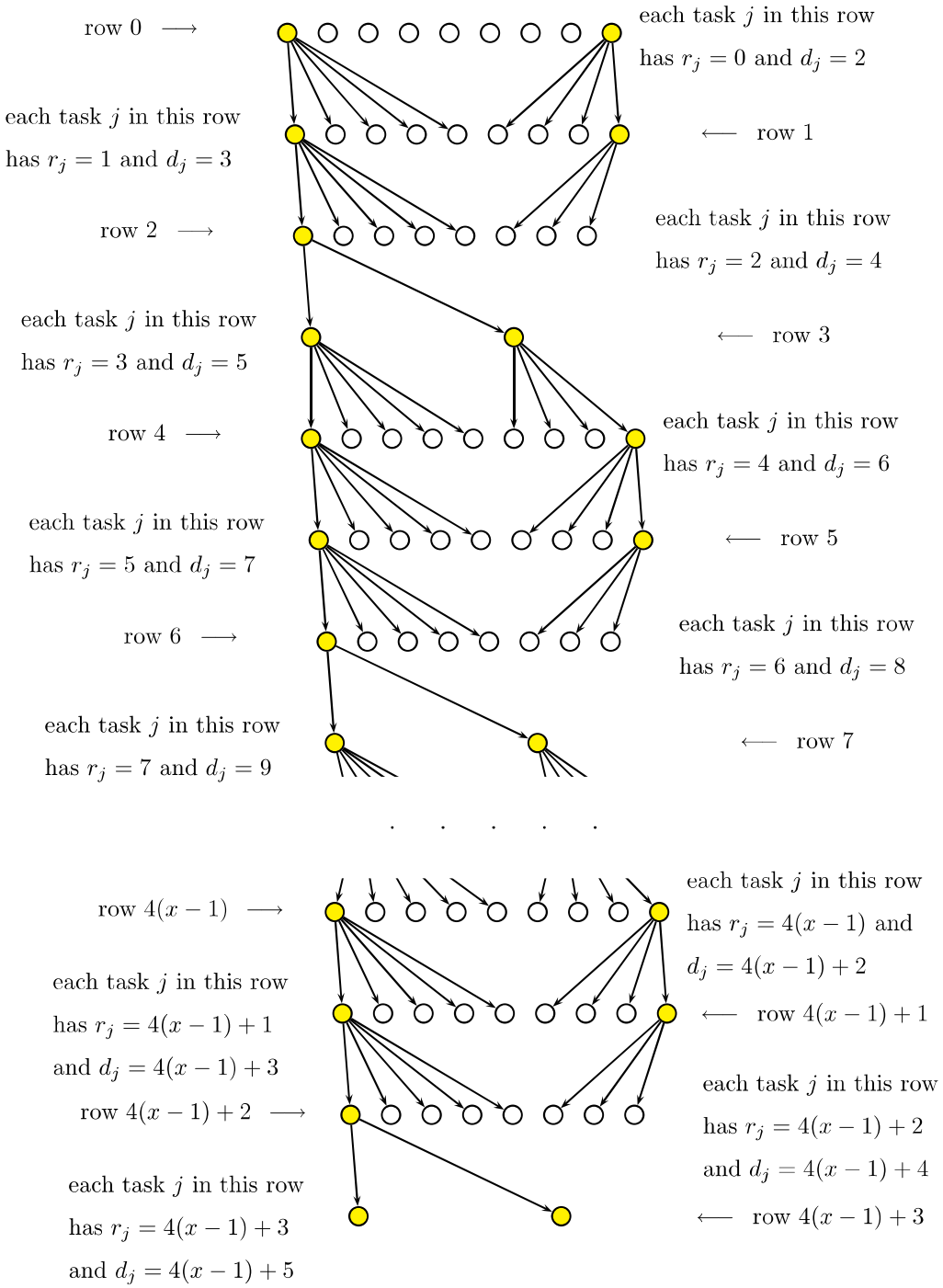


Figure 4: Graph $G_{x,m}$ for $m = 7$.

- If $i \bmod u_m = u_m - 3$, then a_i^1 precedes $a_{i+1}^1, b_{i+1}^1, \ldots, b_{i+1}^{u_m}$ and a_i^2 precedes $b_{i+1}^{u_m+1}, \ldots, b_{i+1}^m$.
- If $i \bmod u_m = u_m - 2$, then a_i^1 precedes a_{i+1}^1 and a_{i+1}^2.
- If $i \bmod u_m = u_m - 1$ and $i < k \cdot u_m - 1$, then $a_i 1$ precedes $a_{i+1}^1, b_{i+1}^1, \ldots, b_{i+1}^{u_m}$, and a_i^2 precedes $a_{i+1}^2, b_{i+1}^{u_m+1}, \ldots, b_{i+1}^m$.

Although the graph presented in [13] and $G_{x,5}$ have the same structure, the example in [13] was developed for a different algorithm and the GJ-algorithm constructs for this example an optimal schedule. The distinct feature of the example presented in this paper is the assignment of release times and due dates in such a way that ensures the consistency of these due dates.

Lemma 12 *For any $m \geq 5$ and any x, the due dates corresponding to $G_{x,m}$ are consistent.*

Proof

In order to prove this lemma, we must study the sets $S(j, s, d)$. As it is long and tedious, please refer to the technical report [5] for a detailed proof. □

For any $G_{x,m}$, consider a list where tasks are arranged in a nondecreasing order of due dates and for each i, where $i \bmod u_m \neq u_m - 2$, all b tasks of this row are listed before the a tasks of the same row. Let $\sigma_{x,m}$ be the corresponding list schedule and let $\sigma_{x,m}^*$ be an optimal schedule for the maximum lateness problem specified by the graph $G_{x,m}$. It is easy to see that

$$\max_{j \in N} d_j = u_m x + 1, \qquad L_{max}(\sigma_{x,m}^*) = 0, \qquad \max_{j \in N} r_j = u_m x - 1$$

and

$$L_{max}(\sigma_{x,m}) = (2u_m - 1)x - u_m x - 1 = u_m x - x - 1.$$

Indeed, GJ-Schedule as well as optimal schedule can be deduced from the example of figure 2 by concatenating schedules of u_m consecutive rows.

Taking into account that $\alpha(m) = \dfrac{1}{u_m}$,

$$\lim_{x\to+\infty} \frac{L_{max}(\sigma_{x,m}) - L_{max}(\sigma^*_{x,m})}{\max_{j\in N} r_j + 1} = \lim_{x\to+\infty} \frac{u_m x - x - 1}{u_m x - 1 + 1}$$

$$= \lim_{x\to+\infty} \frac{u_m x - u_m \alpha(m) x}{u_m x} = 1 - \alpha(m),$$

and (15) is asymptotically tight. Analogously,

$$\lim_{x\to+\infty} \frac{[2-\alpha(m)]L_{max}(\sigma^*_{x,m}) + [1-\alpha(m)]\max_{j\in N} d_j - [1-\alpha(m)]}{L_{max}(\sigma_{x,m})}$$

$$= \lim_{x\to+\infty} \frac{[1-\alpha(m)](u_m x + 1) - [1-\alpha(m)]}{u_m x - x - 2}$$

$$= \lim_{x\to+\infty} \frac{[1-\alpha(m)](u_m x + 1)}{u_m x - x} = \lim_{x\to+\infty} \frac{u_m x - x + [1-\alpha(m)]}{u_m x - x} = 1,$$

and (19) is asymptotically tight.

For $m = 3$ and $m = 4$ the proof of asymptotical tightness is similar to that for $m \geq 5$ and is based on the following graphs $H_{x,m}$. Each graph $H_{x,m}$ has $x \cdot u_m$ rows of nodes, numbered from 0 to $x \cdot u_m - 1$, where u_m is specified by (22) and

$$[\text{number of nodes in row } i] = \begin{cases} m+1 & \text{if} \quad i \bmod u_m = 0 \\ 2 & \text{if} \quad i \bmod u_m = 1 \end{cases}$$

For arbitrary $H_{x,m}$, let $a^1_i, b^1_i, \ldots, b^m_i$ be nodes constituting row i such that $i \, mod \, u_m = 0$, and let a^1_{i+1} and a^2_{i+1} be the only nodes of row i such that $i \, mod \, u_m = 1$. For any row i,

- if $i \bmod u_m = 0$, then a_i^1 precedes both a_{i+1}^1 and a_{i+1}^2;
- if $i \, mod \, m = 1$ and $i \leq u_m x - 3$, then a_i^1 precedes $a_{i+1}^1, b_{i+1}^1, \ldots, b_{i+1}^{u_m}$ and a_i^2 precedes $b_{i+1}^{u_m+1}, \ldots, b_{i+1}^m$.

All tasks corresponding to row i have the same release time equals i and the same due-date equals $d_i = i + 2$.

6 Conclusion

In this paper we generalized the Garey-Johnson algorithm, orginally designed to solve optimally $P2|prec, p_j = 1, r_j|L_{max}$ to find solutions of the problem $P|prec, p_j = 1, r_j|L_{max}$. We analysed its worst case behavior and provided a tight bound on the performance ratio which equals the best known ratio for this problem for even number of processors, and is slighly worse for odd number of processors. The ideas behind Garey-Johnson algorithm are commonly used to solve or find bounds for very different scheduling problems like RCPSP, with so called energy bounds, preemptive scheduling, scheduling with communication delays, scheduling on uniform processors. The structure of the analysis presented in this paper could probably be used in these cases for algorithms based on the same ideas and to derive tight bounds on their worst case performance ratio.

Preemption deserves a particular attention since not much tight bounds have been proven for the worst case performance of scheduling algorithms when preemption is allowed, and since Garey and Johnson derived an algorithm that solves optimally $P2|prec, pmtn|L_{max}$. A first insight of such a study is given in [6].

Existing approximation algorithm for $P|prec, p_j = 1, r_i|L_{max}$ all have a

worst case ratio that tends to 2 when the number of processors grows to infinity. Hence it is a real challenge to design a polynomial algorithm that passes under this threshold. We proved that Garey-Johnson algorithm is not the one, however, its analysis could help the design of new algorithms.

References

[1] B. Braschi and D. Trystram, A new insight into the Coffman-Graham algorithm, SIAM Journal on Computing 23 (1994) 662–669.

[2] P. Brucker, Scheduling algorithms, third edition, Springer, 2001.

[3] P. Brucker, M.R. Garey and D.S. Johnson, Scheduling equal-length tasks under tree-like precedence constraints to minimise maximum lateness, Mathathematics of Opererations Research, 2 (1977) 275–284.

[4] M. R. Garey and D. S. Johnson, Two-processor scheduling with start-time and deadlines, SIAM Journal on Computing 6 (1977) 416-426.

[5] C. Hanen and Y. Zinder, The Worst-case Analysis of the Garey-Johnson Algorithm, technical report LIP6 2008/002, (2008).

[6] C. Hanen and Y. Zinder, The Worst-case Analysis of the Garey-Johnson Algorithm for Preemptive Tasks on m Processors, MISTA, New-York, (2005).

[7] J.K. Lenstra and A.H.G. Rinnooy Kan (1978), Complexity of scheduling under precedence constraints, Operations Research, 26 (1978) 22–35.

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

[9] G. Singh and Y. Zinder, Worst-case performance of two critical path type algorithms, Asia Pacific Journal of Operational Research 17 (2000) 101–122.

[10] Y. Zinder, An Iterative Algorithm for Scheduling UET Tasks with Due Dates and Release Times, European Journal of Operational Research 149 (2003) 404–416

[11] Y.Zinder and D. Roper, A minimax combinatorial optimisation problem on an acyclic directed graph: polynomial-time algorithms and complexity, in: Proceedings of the A.C. Aitken Centenary Conference, Dunedin, (1995) 391–400.

[12] Y. Zinder and D. Roper, An iterative algorithm for scheduling unit-time operations with precedence constraints to minimise the maximum lateness, Annals of Operations Research 81 (1998) 321–340.

[13] Y. Zinder and G. Singh, Preemptive scheduling on parallel processors with due dates, Asia Pacific Journal of Operational Research 22 (2005) 445–462.