International Journal of Operations Research

International Journal of Operations Research Vol. 4, No. 2, 90–97 (2007)

Performance of Critical Path Type Algorithms with Communication Delay

Gaurav Singh*

CSIRO Mathematical and Information sciences, Private Bag 10, South Clayton, VIC 3169, Australia

Received October 2005; *Revised October 2005*; *Accepted November 2006*

Abstract—The Brucker-Garey-Johnson algorithm and Hu's algorithm are based on the idea of the critical path method and were developed for the model with unit execution time tasks, precedence constraints and parallel identical processors. The performance guarantees for these algorithms have been presented in Singh and Zinder (2000a, 2000b). We present upper bounds for the Brucker-Garey-Johnson algorithm with communication delays, which can be seen as a generalization of the performance guarantees in Singh and Zinder (2000a, 2000b). As a particular case this also gives performance guarantees for Hu's algorithm with communication delays and therefore, also generalizes the previously known performance guarantees for this algorithm.

Keywords—Scheduling theory, Unit execution and communication times, Precedence, Maximum lateness, Worst-case analysis

1. INTRODUCTION

In this paper we consider an extension of the classical unit execution time (UET) maximum lateness problem, the problem with a unit communication delay. Unit communication delay (UCT) problem is a straightforward generalization of UET problem and has one of its applications in multiprocessor environment. In multiprocessor environments there are time delays between the execution of dependent tasks on different processors. These time delays are required to send the results of the computation of a task from one processor to another.

Using the popular three field notation UCT problem can be denoted as $P \mid prec,\ p_j = 1,\ c_{ij} = 1 \mid L_{\max}$ and can be described as follows: Suppose that a set $N = \{1, ..., n\}$ of n tasks is to be processed on $m > 1$ identical processors 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 $i \to j$, then processing of task i must be completed before task j can begin its processing. It is convenient to represent a partially ordered set of tasks by an acyclic directed graph where nodes correspond to the tasks and the arcs reflect the precedence constraints. Each task can be processed on any processor, and once a processor begins executing a task then it continues until completion (i.e. no preemptions are allowed). The processing starts at time $t = 0$. Each task $j \in N$ has a due date d_j, the desired point in time by which the processing should be completed. Each processor can process at most one task at a time. All the tasks have the same processing time which, without loss of generality, can be taken as the unit of time.

Since no preemptions are allowed, to specify a schedule s it suffices to designate for each task $j \in N$ a processor and its completion time $C_j(s)$. If task j has a completion time $C_j(s)$, then its processing commences at time $C_j(s) - 1$. Due to unit communication delay, if two tasks i and j such that $i \to j$ are processed on different processor then $C_j(s) \geq C_i(s) + 2$. Since the processing of the tasks commences at $t = 0$ and each task requires one unit of time, without loss of generality, we will consider only schedules in which all completion times are positive integers. The goal is to find a schedule which minimizes the criterion of maximum lateness

$$L_{\max}(s) = \max_{j \in N}(C_j(s) - d_j).$$

If all due dates are equal to zero this problem converts to the makespan problem with the criterion

$$C_{\max}(s) = \max_{a \in N} C_a(s).$$

The makespan problem, and therefore the maximum lateness problem, is *NP*-hard as shown by Rayward-Smith (1987). Moreover, this makespan problem remains *NP*-hard even if the partially ordered set of tasks is represented by an in-tree (Lenstra et al. (1996)). Several polynomial time algorithms were developed for various particular cases by Finta et al. (1996), Lenstra et al. (1996), Verriet (2000) and Singh (2005). It was also shown in Rayward-Smith (1987) that the worst-case performance of an arbitrary list schedule can be characterized as

* Corresponding author's email: Gaurav.Singh@csiro.au

$$C_{\max}(s^{LS}) \le \left(3-\frac{2}{m}\right)C_{\max}(s^*)-\left(2-\frac{1}{m}\right),$$

where s^{LS} is a schedule constructed by an arbitrary list algorithm and s^* is an optimal schedule for the $P\,|\,prec,\ p_j=1,\ c_{ij}=1\,|\,C_{\max}$ problem. An example is presented in Rayward-Smith (1987) to show that this performance guarantee is asymptotically achievable.

As far as the $P\,|\,prec,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem is concerned, only few results are known and have been obtained very recently by Singh (2005, 2001), Verriet (2000). Verriet (2000) shows that a straightforward extension of the Garey-Johnson algorithm presented in Garey and Johnson (1976) for the $P2\,|\,prec,\ p_j=1\,|\,L_{\max}$ problem also solves the $P2\,|\,outtree,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem, and its worst-case performance can be characterized as

$$L_{\max}(s^V) \le \left(2-\frac{2}{m}\right)L_{\max}(s^*)+\left(1-\frac{2}{m}\right)\max_{j\in N}(d_j+1),$$

where s^V is a schedule constructed by the algorithm presented in Verriet (2000) and s^* is an optimal schedule for the $P\,|\,outtree,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem. In Singh (2001) an extension of the Brucker-Garey-Johnson algorithm presented in Brucker et al. (1977) is presented, with the worst-case performance guarantee as

$$L_{\max}(s^D)-L_{\max}(s^*) \le 2\left(\frac{m-1}{m}\right)\ell - w(m),$$

where s^D is a schedule constructed by the algorithm presented in Singh (2001), s^* is an optimal schedule for the $P\,|\,prec,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem, ℓ is the length of the longest path in the corresponding graph and

$$w(m)=\begin{cases}\dfrac{m-1}{m} & \text{for } m \text{ odd}\\[2ex] 1 & \text{otherwise}\end{cases}$$

Unlike other known performance guarantees of Rayward-Smith (1987), Verriet (2000) for problems with communication delay the performance guarantee in Singh (2001) is tight for arbitrary large instances of the $P\,|\,prec,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem. Singh (2005) shows that the worst-case performance of the Brucker-Garey-Johnson algorithm presented in Brucker et al. (1977) for the $P\,|\,outforest,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem is

$$L_{\max}(s^D) \le \left(2-\frac{1}{m}\right)L_{\max}(s^*)+\left(1-\frac{1}{m}\right)\max_{j\in N}(d_j),$$

where s^D is a schedule constructed by the algorithm presented in Singh (2005) and s^* is an optimal schedule for the $P\,|\,outforest,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem. It is also shown that the presented algorithm solves the $P\infty\,|\,outforest,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem. Here the term $P\infty$ indicates that the number of processors are unlimited or sufficiently large. Singh (2005) also presents another polynomial time algorithm which solves the $P2\,|\,outforest,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem.

For the $P\,|\,prec,\ p_j=1\,|\,L_{\max}$ problem the worst-case performance of the Brucker-Garey-Johnson algorithm presented in Brucker et al. (1977) has been analyzed in Singh and Zinder (2000a, 2000b), with the following performance guarantees

$$L_{\max}(s^{BGJ}) \le \left(2-\frac{1}{m}\right)L_{\max}(s^*)+\left(1-\frac{1}{m}\right)\max_{j\in N}(d_j-1) \qquad (1)$$

and

$$L_{\max}(s^{BGJ})-L_{\max}(s^*) \le \begin{cases}\min\left\{\dfrac{n-l}{m}-1,\left(\dfrac{m-1}{m}\ell\right)\right\} & \text{if } n-l\ge m\\ 0 & \text{otherwise}\end{cases} \qquad (2)$$

In this paper we present performance guarantees which as a particular case give the performance guarantees (1) and (2). Thus, our results, in some sense, generalize the performance guarantees in Singh and Zinder (2000a, 2000b).

2. THE WORST-CASE PERFORMANCE OF THE BRUCKER-GAREY-JOHNSON ALGORITHM

Both the Brucker-Garey-Johnson algorithm (Brucker et al. (1977)) and Hu's algorithm (Hu (1961)) can be viewed as a two phase procedures. The first phase of each algorithm assigns to each task a number indicating the urgency of the task. The second phase schedules the tasks to processors in accord with the task's urgency. In this section we consider the $P\,|\,prec,\ p_j=1,\ c_{ij}=1\,|\,L_{\max}$ problem and analyze the worst-case performance of a generalization of the Brucker-Garey-Johnson algorithm (Brucker et al. (1977)) presented in Singh (2001). For completeness, we present the algorithm below.

Let $K(j')$ be the set of all immediate successors of some arbitrary task j'. A task i is an immediate successor of task j' if $j'\to i$ and there is no task i' such that $j'\to i'\to i$. Let j be an arbitrary task such that $K(j)\neq\varnothing$, and let $j_1\in K(j)$ satisfy $d_{j_1}=\min_{i\in K(j)}d_i$. For an arbitrary schedule s, denote

$$L_j(s) = \begin{cases} C_j(s)+1-d_{j_1}, & \text{if } |K(j)|=1 \\ \max\{C_j(s)+1-d_{j_1}, C_j(s)+2-\min\limits_{i\in[K(j)-\{j_1\}]} d_i\}, & \text{if } |K(j)|>1. \end{cases}$$

Since task j_1 can be processed only after the completion of task j, and at most one task from the set $K(j)$ can be scheduled immediately after task j, we have

$$L_{\max}(s) \geq \max\{L_j(s), C_j(s)-d_j\}.$$

Therefore, the replacement of the original due dates by due dates d'_j calculated according to the following algorithm does not change the value $L_{\max}(s)$ for all feasible schedules s.

Algorithm 1. Due date modification algorithm for the model with a unit communication delay

for each task j such that $K(j)=\varnothing$ **do** $d'_j = d_j$
while *there is a task j which has not been assigned its modified due date* d'_j *and all of whose immediate successors have been assigned their modified due dates* **do**
Select $j_1 \in K(j)$ such that $d'_{j_1} = \min_{i\in K(j)} d'_i$ and set

$$d'_j = \begin{cases} \min\{d_j, d'_{j_1}-1\}, & \text{if } |K(j)|=1 \\ \min\{d_j, d'_{j_1}-1, \min\limits_{i\in[K(j)-\{j_1\}]}\{d'_i-2\}\}, & \text{if } |K(j)|>1 \end{cases}$$

To construct the desired schedule we arrange the tasks in a list, L, in nondecreasing order of their modified due dates and apply the following algorithm. We will say that a task is available for processing in some time slot, if this task has no unscheduled predecessors and there is an idle processor which can process this task in that time slot.

Algorithm 2. List scheduling with communication delay

Set $t=0$ and $A=\varnothing$
while $L\neq\varnothing$ **do**
repeat
Scan the list from left to right and select the first task available for processing. Let it be task j.
if *task j can be processed in the time slot t on only one specific processor* **then**
assign j to that processor
else $A = A\cup\{j\}$
$L = L\setminus\{j\}$
until $|A|$ *becomes equal to the number of processors which remain idle, or the end of the list is reached*
Allocate the tasks from A to the idle processors
Set $t=t+1$ and $A=\varnothing$

Let s^D be a schedule constructed by the above algorithm. From the set of all tasks v such that

$$C_v(s^D) - d'_v = L_{\max}(s^D),$$

choose a task p with the smallest completion time. Let $M(p)$ be the set of all tasks $j \in N$ such that $d'_j \leq d'_p$. It is easy to see that for any task $i \in M(p)$, the inequality $C_i(s^D) \leq C_p(s^D)$ holds, and that any predecessor of a task from $M(p)$ also belongs to $M(p)$. If $C_p(s^D) = 1$, then s^D is optimal, so let us assume that $C_p(s^D) > 1$. For any positive integer $t < C_p(s^D)$, we say that the time slot t is complete, if exactly m tasks from $M(p)$ are processed on the time interval $[t-1, t]$, and the time slot t is incomplete, if the number of tasks from $M(p)$ which are processed on the time interval $[t-1, t]$ is greater than or equal to one but less than m. Because of communication delay a time slot t may contain no tasks from $M(p)$ at all. In this case we will call this time slot empty. Corresponding to this classification of time slots we introduce the following notations. Let $l_c(t)$, $l_i(t)$, and $l_0(t)$ be the number of complete, incomplete, and empty time slots before time slot t. Because we consider a model with a unit communication delay, in a list schedule for any empty time slot t the time slot $t-1$ is either complete or incomplete. The number of empty time slots t' such that $t'<t$ and the time slot $t'-1$ is complete will be denoted by $l_{0c}(t)$. Correspondingly, the numbers of empty time slots t' such that $t'<t$ and the time slot $t'-1$ is incomplete will be denoted by $l_{0i}(t)$. Hence, for any t, $l_0(t) = l_{0c}(t) + l_{0i}(t)$. It is also convenient to introduce the following notation. Consider an arbitrary task j and all paths, which terminate at the node corresponding to j in the graph representing the partially ordered set of tasks. Let $\ell(j)$ be the length of the longest of these paths. For every pair of tasks i and j such that $j \in K(i)$, let

$$C(i,j) = \begin{cases} 1 & \text{if there is a task } j'\in K(i) \text{ such that } j'\neq j \text{ and } C_{j'}(s^D) = C_i(s^D)+1; \\ & \text{or, if there is a task } i'\to j \text{ such that } i'\neq i \text{ and } C_{i'}(s^D) = C_i(s^D). \\ 0 & \textit{otherwise} \end{cases}$$

Let for any task j, $P(j)$ be the set of all vectors $(i_1, i_2, \ldots, i_k = j)$, where each vector represents a path terminating on task j, i.e. $i_1 \to i_2 \to \ldots i_k = j$. Then we define

$$Com(j) = \max_{(i_1,\ldots,i_k=j)\in P(j)} \sum_{x=1}^{k-1} C(i_x, i_{x+1}) \qquad (3)$$

Note that $Com(j)$ computes the maximum amount of communication delay incurred in the schedule s^D for task j.

Lemma 1. The time slot $C_p(s^D) - 1$ is either complete or the time slot $C_p(s^D) - 2$ contains at least two predecessors

of task *p*.

Proof. Suppose that there exists a task *j* such that $C_j(s^D) = C_p(s^D) - 1$ and $j \to p$. Then $d'_j \le d'_p - 1$ and $C_j(s^D) - d'_j \ge C_j(s^D) + 1 - d'_p = C_p(s^D) - d'_p$, which contradicts the selection of task *p*.

If the time slot $C_p(s^D) - 1$ is empty then the time slot $C_p(s^D) - 2$ must contain at least two predecessors of task *p*, which proves the lemma.

Suppose the time slot $C_p(s^D) - 1$ is incomplete. As has been shown above, time slot $C_p(s^D) - 1$ does not contains a predecessor of task *p*. Therefore, the time slot $C_p(s^D) - 2$ contains a predecessor of task *p*. Suppose the time slot $C_p(s^D) - 2$ contains exactly one predecessor of task *p*, say task *j*. Since the time slot $C_p(s^D) - 1$ is incomplete and task *p* was not scheduled on this time slot, another successor of *j*, say j', was processed during this time slot. As j' was preferred in place of *p*, we have $d'_{j'} \le d'_p$. Then, by the due date modification algorithm, $d'_j \le d'_p - 2$ and $C_j(s^D) - d'_j \ge C_j(s^D) + 2 - d'_p = C_p(s^D) - d'_p$, which again contradicts the selection of task *p*. Hence if the time slot $C_p(s^D) - 1$ is incomplete then the time slot $C_p(s^D) - 2$ contains at least two predecessors of task *p*.

Lemma 2. For any task *j*,

$$\ell(j) + Com(j) \ge 1 + 2l_0(C_j(s^D)) + l_i(C_j(s^D)) - l_{0i}(C_j(s^D)). \tag{4}$$

Proof. We will prove (4) by induction on $l_0(C_j(s^D)) + l_i(C_j(s^D))$. Since any path which terminates at some node includes this node, $\ell(j) \ge 1$. Therefore, if $l_0(C_j(s^D)) + l_i(C_j(s^D)) = 0$, then (4) holds as $Com(j) \ge 0$. Suppose that $l_0(C_j(s^D)) + l_i(C_j(s^D)) = 1$. The existence of time slot *t* such that $t < C_j(s^D)$ and is either incomplete or empty indicates that task *j* has a predecessor, because otherwise according to the list algorithm task *j* can be processed on the time interval $[t-1, t]$. Therefore, $\ell(j) \ge 2$. Also, if $l_0(C_j(s^D)) = 1$ then $l_i(C_j(s^D)) = 0$ and $Com(j) \ge 1$,or if $l_i(C_j(s^D)) = 1$ then $l_i(C_j(s^D)) = 0$ and $Com(j) \ge 0$. In either case, it is easy to see that (4) holds.

Suppose that for each task j' such that

$$l_0(C_{j'}(s^D)) + l_i(C_{j'}(s^D)) \le k$$

$$\ell(j') + Com(j') \ge 1 + 2l_0(C_{j'}(s^D)) + l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)). \tag{5}$$

Let *j* be a task satisfying $l_0(C_j(s^D)) + l_i(C_j(s^D)) = k + 1$. Among all non-negative integers *t* such that $t < C_j(s^D)$ and the time slot *t* is either incomplete or empty, select the largest one. Let it be t'. If the time slot t' is empty, then the time slot $t'-1$ contains at least two predecessors of task *j*. Denote one of the predecessors by j'. We have

$$l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)) = l_i(C_j(s^D)) - l_{0i}(C_j(s^D)),$$
$$l_0(C_{j'}(s^D)) + 1 = l_0(C_j(s^D))$$

and $Com(j) \ge Com(j') + 1$, which together with $\ell(j) \ge \ell(j') + 1$ and (5) leads to (4).

If the time slot t' is incomplete, then either time slot t' or $t'-1$ contains a predecessor of task *j*. Again denote this predecessor by j'. If $C_{j'}(s^D) = t'$, we have $Com(j) \ge Com(j')$ and $l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)) + 1 \ge l_i(C_j(s^D)) - l_{0i}(C_j(s^D))$

Therefore using (5), we get

$$\ell(j) + Com(j) \ge \ell(j') + Com(j') + 1$$
$$\ge 1 + 2l_0(C_{j'}(s^D)) + l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)) + 1,$$
$$\ge 1 + 2l_0(C_j(s^D)) + l_i(C_j(s^D)) - l_{0i}(C_j(s^D)),$$

which proves the lemma in this case.

If $C_{j'}(s^D) = t' - 1$, then either another successor of task j' is processed on the time slot t' or at least two predecessors of task *j* are processed on the time slot $t'-1$, as otherwise task *j* should have been processed on the time slot t'. In either case, we have $Com(j) \ge Com(j') + 1$ and $l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)) + 2 \ge l_i(C_j(s^D)) - l_{0i}(C_j(s^D))$.

Therefore, using (5), we get

$$\ell(j) + Com(j) \ge \ell(j') + Com(j') + 2$$
$$\ge 1 + 2l_0(C_{j'}(s^D)) + l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)) + 2$$
$$\ge 1 + 2l_0(C_j(s^D)) + l_i(C_j(s^D)) - l_{0i}(C_j(s^D)),$$

which completes the proof.

Lemma 3. For any task $j \in M(p)$ and for any schedule *s*,

$$C_j(s) + Com(j) \ge 1 + 3l_0(C_j(s^D)) + l_i(C_j(s^D)) - l_{0i}(C_j(s^D)). \tag{6}$$

Proof. We will again prove (6) by induction on $l_0(C_j(s^D)) + l_i(C_j(s^D))$. Since for any task *j* in any schedule *s* we have $C_j(s) \ge 1$ and $Com(j) \ge 0$, clearly (6) holds if $l_0(C_j(s^D)) + l_i(C_j(s^D)) = 0$. Suppose that $l_0(C_j(s^D)) + l_i(C_j(s^D)) = 1$. If $l_0(C_j(s^D)) = 1$ and $l_i(C_j(s^D)) = 0$, then the existence of an empty time slot indicates that task *j* has at least two predecessors, and therefore, due to a unit communication delay, in any schedule *s*, $C_j(s) \ge 3$ and $Com(j) \ge 1$. On the other hand, if $l_0(C_j(s^D)) = 0$ and $l_i(C_j(s^D)) = 1$, then the existence of an incomplete time slot indicates that task *j* has a predecessor, and therefore, in any schedule *s*, $C_j(s) \ge 2$ and $Com(j) \ge 0$. In either case, it is easy to see that (6) holds. Suppose that

for each task j' such that

$$\begin{aligned} &l_0(C_{j'}(s^D)) + l_i(C_{j'}(s^D)) \le k \\ &C_{j'}(s) + Com(j') \\ &\ge 1 + 3l_0(C_{j'}(s^D)) + l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)). \end{aligned} \tag{7}$$

Let j be a task satisfying $l_0(C_j(s^D)) + l_i(C_j(s^D)) = k + 1$. Among all positive integers t such that $t < C_j(s^D)$, and the time slot t is either incomplete or empty, select the largest one. Let it be t'. If the time slot t' is empty then the time slot $t'-1$ contains at least two predecessors of j, and therefore for at least one of the predecessors, j', due to unit communication delay, $C_j(s) \ge C_{j'}(s) + 2$ and $Com(j) \ge Com(j') + 1$. Using these inequalities together with

$$\begin{aligned} &l_0(C_{j'}(s^D)) + 1 \;=\; l_0(C_j(s^D)), \\ &l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)) \;=\; l_i(C_j(s^D)) - l_{0i}(C_j(s^D)), \end{aligned}$$

and (7), it is easy to see that in this case (6) holds.

If the time slot t' is incomplete, then either the time slot t' or $t'-1$ contains a predecessor of task j. Again, denote this predecessor as j'. If $C_{j'}(s^D) = t'$, then

$$\begin{aligned} &l_0(C_{j'}(s^D)) = l_0(C_j(s^D)), \\ &Com(j') \le Com(j),\; C_{j'}(s) + 1 \le C_j(s), \\ &l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)) + 1 \\ &= l_i(C_j(s^D)) - l_{0i}(C_j(s^D)), \end{aligned}$$

which together with (7) gives (6). If $C_{j'}(s^D) = t' - 1$, then either another successor of j' is processed during the incomplete time slot t' or j has at least two predecessors processed during the time slot $t'-1$, as otherwise due to list scheduling task j should have been processed during this incomplete time slot t'. In either case,

$$\begin{aligned} &l_0(C_{j'}(s^D)) = l_0(C_j(s^D)), \\ &Com(j') + 1 \le Com(j),\; C_{j'}(s) + 1 \le C_j(s), \\ &l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)) + 2 \\ &\ge l_i(C_j(s^D)) - l_{0i}(C_j(s^D)), \end{aligned}$$

which together with (7) gives

$$\begin{aligned} &C_j(s) + Com(j) \ge C_{j'}(s) + Com(j') + 2 \\ &\ge 1 + 3l_0(C_{j'}(s^D)) + l_i(C_{j'}(s^D)) - l_{0i}(C_{j'}(s^D)) + 2 \\ &\ge \quad 1 + 3l_0(C_j(s^D)) + l_i(C_j(s^D)) - l_{0i}(C_j(s^D)), \end{aligned}$$

and therefore completes the proof.

Lemma 4. If the time slot $C_p(s^D) - 1$ is not empty, then for any schedule s

$$\begin{aligned} &L_{\max}(s) + \max_{k \in N} Com(k) \ge 2 + 3l_0(C_p(s^D)) \\ &+ l_i(C_p(s^D)) - l_{0i}(C_p(s^D)) - d'_p. \end{aligned} \tag{8}$$

Proof. Let U be the set comprising all tasks from $M(p)$ in the time slot $C_p(s^D) - 1$ and task p. By Lemma 1 the time slot $C_p(s^D) - 1$ is either complete or the time slot $C_p(s^D) - 2$ contains at least two predecessors of task p.

If the time slot $C_p(s^D) - 1$ is complete, then $|U| = m + 1$ and therefore for any schedule s there is a task $j \in U$ such that $\max_{i \in U} C_i(s) \ge C_j(s) + 1$. Using this inequality together with Lemma 3 and the obvious equalities $l_0(C_j(s^D)) = l_0(C_p(s^D))$ and $l_i(C_j(s^D)) = l_i(C_p(s^D))$, we get

$$\begin{aligned} &L_{\max}(s) + \max_{k \in N} Com(k) \ge \max_{i \in U}(C_i(s) - d'_i) + Com(j) \\ &\ge \max_{i \in U} C_i(s) - d'_p + Com(j) \ge C_j(s) + 1 + Com(j) - d'_p \\ &\ge 1 + 3l_0(C_j(s^D)) + l_i(C_j(s^D)) - l_{0i}(C_j(s^D)) + 1 - d'_p \\ &= 2 + 3l_0(C_p(s^D)) + l_i(C_p(s^D)) - l_{0i}(C_p(s^D)) - d'_p. \end{aligned}$$

If the time slot $C_p(s^D) - 1$ is incomplete then by Lemma 1 the time slot $C_p(s^D) - 2$ contains at least two predecessors of task p. Therefore, in any schedule s for at least one of such predecessors j, from Lemma 3 together with the obvious inequalities $l_0(C_j(s^D)) = l_0(C_p(s^D))$, $l_i(C_j(s^D)) + 2 \ge l_i(C_p(s^D))$, and $Com(p) \ge Com(j) + 1$, we have

$$\begin{aligned} &L_{\max}(s) + \max_{k \in N} Com(k) \ge C_p(s) + Com(p) - d'_p \\ &\ge C_j(s) + 2 + Com(j) + 1 - d'_p \\ &\ge 1 + 3l_0(C_j(s^D)) + l_i(C_j(s^D)) - l_{0i}(C_j(s^D)) + 3 - d'_p \\ &\ge 2 + 3l_0(C_p(s^D)) + l_i(C_p(s^D)) - l_{0i}(C_p(s^D)) - d'_p. \end{aligned}$$

This completes the proof.

Theorem 5. If s^* is an optimal schedule for the maximum lateness problem then

$$\begin{aligned} &L_{\max}(s^D) \le \left(2 - \frac{1}{m}\right) L_{\max}(s^*) \\ &+ \left(1 - \frac{1}{m}\right)\left(\max_{j \in N} d_j + \max_{k \in N} Com(k) - 1\right). \end{aligned} \tag{9}$$

Proof. If the time slot $C_p(s^D) - 1$ is empty then $l_0(C_p(s^D)) \ge 1$, and therefore by Lemma 3

$$\begin{aligned} &L_{\max}(s^D) \\ &= C_p(s^D) - d'_p \\ &= 1 + l_c(C_p(s^D)) + l_0(C_p(s^D)) + l_i(C_p(s^D)) - d'_p \end{aligned}$$

$$
\begin{aligned}
&= 1 + l_c(C_p(s^D)) + l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) - d'_p \\
&= l_c(C_p(s^D)) + \frac{l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) + 1}{m} + l_{0c}(C_p(s^D)) \\
&+1 + \left(\frac{m-1}{m}\right)(l_{0i}(C_p(s^D)) + l_i(C_p(s^D))) - \frac{1}{m} - d'_p \\
&\le \frac{|M(p)|}{m} - d'_p \\
&+ \left(\frac{m-1}{m}\right)(1 + 2l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D)) + l_i(C_p(s^D))) \\
&\le L_{\max}(s^*) \\
&+ \left(\frac{m-1}{m}\right)(1 + 2l_0(C_p(s^D)) + l_i(C_p(s^D)) - l_{0i}(C_p(s^D))) \\
&= L_{\max}(s^*) \\
&+ \left(\frac{m-1}{m}\right)(1 + 3l_0(C_p(s^D)) + l_i(C_p(s^D)) - l_{0i}(C_p(s^D))) \\
&- \left(\frac{m-1}{m}\right) l_0(C_p(s^D)) \\
&\le L_{\max}(s^*) + \left(\frac{m-1}{m}\right)(C_p(s^*) + Com(p) - d'_p + d'_p - 1) \\
&\le \left(2 - \frac{1}{m}\right) L_{\max}(s^*) + \left(1 - \frac{1}{m}\right)\left(\max_{j\in N} d_j + \max_{k\in N} Com(k) - 1\right),
\end{aligned}
$$

which proves the theorem in this case.

If the time slot $C_p(s^D) - 1$ is not empty, then from Lemma 4, and as above we have

$$
\begin{aligned}
&L_{\max}(s^D) \\
&\le L_{\max}(s^*) \\
&+ \left(\frac{m-1}{m}\right)(1 + 2l_0(C_p(s^D)) + l_i(C_p(s^D)) - l_{0i}(C_p(s^D))) \\
&\le L_{\max}(s^*) \\
&+ \left(\frac{m-1}{m}\right)(2 + 3l_0(C_p(s^D)) + l_i(C_p(s^D)) - l_{0i}(C_p(s^D))) \\
&- \left(\frac{m-1}{m}\right)\left(1 + l_0(C_p(s^D))\right) \\
&\le L_{\max}(s^*) + \left(\frac{m-1}{m}\right)(L_{\max}(s^*) + d'_p + \max_{k\in N} Com(k) - 1) \\
&\le \left(2 - \frac{1}{m}\right) L_{\max}(s^*) + \left(1 - \frac{1}{m}\right)\left(\max_{j\in N} d_j + \max_{k\in N} Com(k) - 1\right).
\end{aligned}
$$

Therefore proves the theorem in this case as well and also completes the proof.

Note that in the case when communication delays are zero the $P \mid prec,\ p_j = 1,\ c_{ij} = 1 \mid L_{\max}$ problem converts to the $P \mid prec,\ p_j = 1 \mid L_{\max}$ problem, and in this case (9) gives (1).

Theorem 6. If s^* is an optimal schedule for the maximum lateness problem, then

$$
\begin{aligned}
&L_{\max}(s^D) - L_{\max}(s^*) \le \\
&\begin{cases}
\min\left\{\dfrac{n-\ell+1}{m} - 1, \dfrac{m-1}{m}\ell\right\} + \dfrac{m-1}{m}\max\limits_{k\in N} Com(k), & \text{if } n - \ell \ge m \\
\dfrac{m-1}{m}\max\limits_{k\in N} Com(k), & \text{otherwise}
\end{cases}
\end{aligned}
\tag{10}
$$

where n is the number of tasks and ℓ is the length of the longest path in the corresponding graph.

Proof. It is easy to see that schedule s^D is optimal, and therefore (10) holds, if $C_p(s^D) - 1$. As shown before, s^D is also optimal when $C_p(s^D) > 1$ and $l_i(C_p(s^D)) + l_0(C_p(s^D)) = 0$. Hence, we only need to prove that (10) holds when $C_p(s^D) > 1$, and $l_i(C_p(s^D)) + l_0(C_p(s^D)) > 0$. Since $\ell(p)$ is the length of a longest path in the subgraph corresponding to the set $M(p)$, we have $L_{\max}(s^*) \ge \ell(p) - d'_p$. Suppose that $L_{\max}(s^*) \ge \ell(p) + 1 - d'_p$. By Lemma 2 we get

$$
\begin{aligned}
&L_{\max}(s^D) \\
&= C_p(s^D) - d'_p \\
&= 1 + l_c(C_p(s^D)) + l_0(C_p(s^D)) + l_i(C_p(s^D)) - d'_p \\
&= 1 + l_c(C_p(s^D)) + l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) - d'_p \\
&= l_c(C_p(s^D)) + \frac{l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) + 1}{m} + l_{0c}(C_p(s^D)) \\
&+ \left(\frac{m-1}{m}\right)(1 + l_{0i}(C_p(s^D)) + l_i(C_p(s^D))) - d'_p \\
&\le \frac{|M(p)|}{m} \\
&+ \left(\frac{m-1}{m}\right)\left(1 + 2l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D)) + l_i(C_p(s^D))\right) - d'_p \\
&\le \frac{|M(p)|}{m} + \left(\frac{m-1}{m}\right)\left(\ell(p) + Com(p)\right) - d'_p \\
&= \frac{|M(p)| - \ell(p)}{m} + \ell(p) - d'_p + \left(\frac{m-1}{m}\right)\max_{k\in N} Com(k) \\
&\le L_{\max}(s^*) + \frac{|M(p)| - \ell(p) + 1}{m} - 1 + \left(\frac{m-1}{m}\right)\max_{k\in N} Com(k).
\end{aligned}
$$

Hence,

$$
\begin{aligned}
&L_{\max}(s^D) - L_{\max}(s^*) \\
&\le \frac{|M| - \ell(p) + 1}{m} - 1 + \left(\frac{m-1}{m}\right)\max_{k\in N} Com(k).
\end{aligned}
\tag{11}
$$

Now suppose that $L_{\max}(s^*) = \ell(p) - d'_p$ and the time slot $C_p(s^D) - 1$ is empty. Then, we have $\max_{j\in M(p)} C_j(s^*) = \ell(p)$. Hence, by Lemma 3

$$
\ell(p) \ge 1 + 3l_0(C_p(s^D)) + l_i(C_p(s^D)) - l_{oi}(C_p(s^D)) - Com(p),
$$

We have

$$
\begin{aligned}
&L_{\max}(s^D)\\
&= C_p(s^D) - d'_p = 1 + l_c(C_p(s^D)) + l_0(C_p(s^D)) + l_i(C_p(s^D)) - d'_p\\
&= 1 + l_c(C_p(s^D)) + l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) - d'_p\\
&= l_c(C_p(s^D)) + \frac{l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) + 1}{m} + l_{0c}(C_p(s^D))\\
&+ \left(\frac{m-1}{m}\right)(1 + l_{0i}(C_p(s^D)) + l_i(C_p(s^D))) - d'_p\\
&\le \frac{|M(p)|}{m} + \left(\frac{m-1}{m}\right)(1 + 2l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D))\\
&+ l_i(C_p(s^D))) - d'_p\\
&\le \frac{|M(p)|}{m} + \left(\frac{m-1}{m}\right)(\ell(p) + Com(p) - l_0(C_p(s^D))) - d'_p\\
&\le \frac{|M(p)| - \ell(p) + 1}{m} - 1 + \ell(p) - d'_p + \left(\frac{m-1}{m}\right)\max_{k\in N} Com(k).
\end{aligned}
$$

and, subtracting $L_{\max}(s^*) = \ell(p) - d'_p$, we again obtain (11).

Now suppose that $L_{\max}(s^*) = \ell(p) - d'_p$ and the time slot $C_p(s^D) - 1$ is not empty. Then by Lemma 4 we have

$$
\begin{aligned}
&\ell(p) + \max_{k\in N} Com(k) = L_{\max}(s^*) + \max_{k\in N} Com(k) + d'_p\\
&\ge 2 + 3l_0(C_p(s^D)) + l_i(C_p(s^D)) - l_{0i}(C_p(s^D)).
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&L_{\max}(s^D)\\
&= C_p(s^D) - d'_p\\
&= 1 + l_c(C_p(s^D)) + l_0(C_p(s^D)) + l_i(C_p(s^D)) - d'_p\\
&= 1 + l_c(C_p(s^D)) + l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) - d'_p\\
&= l_c(C_p(s^D)) + \frac{l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) + 1}{m} + l_{0c}(C_p(s^D))\\
&+ \left(\frac{m-1}{m}\right)(1 + l_{0i}(C_p(s^D)) + l_i(C_p(s^D))) - d'_p \le \frac{|M(p)|}{m} - d'_p\\
&+ \left(\frac{m-1}{m}\right)(1 + 2l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D)) + l_i(C_p(s^D)))\\
&\le \frac{|M(p)|}{m}\\
&+ \left(\frac{m-1}{m}\right)(\ell(p) + Com(p) - l_0(C_p(s^D)) - 1) - d'_p - \frac{1}{m}\\
&\le \frac{|M(p)| - \ell(p) + 1}{m} - 1 + \ell(p) - d'_p + \left(\frac{m-1}{m}\right)\max_{k\in N} Com(k).
\end{aligned}
$$

and, subtracting $L_{\max}(s^*) = \ell(p) - d'_p$, we again obtain(11). Since $n - |M(p)| \ge \ell - \ell(p)$, (11) gives

$$
\begin{aligned}
&L_{\max}(s^D) - L_{\max}(s^*)\\
&\le \frac{n - \ell + 1}{m} - 1 + \left(\frac{m-1}{m}\right)\max_{k\in N} Com(k).
\end{aligned}
\tag{12}
$$

Using Lemma 2, we have,

$$
\begin{aligned}
&L_{\max}(s^D)\\
&= C_p(s^D) - d'_p\\
&= 1 + l_c(C_p(s^D)) + l_0(C_p(s^D)) + l_i(C_p(s^D)) - d'_p\\
&= 1 + l_c(C_p(s^D)) + l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) - d'_p\\
&= l_c(C_p(s^D)) + \frac{l_{0i}(C_p(s^D)) + l_i(C_p(s^D)) + 1}{m} + l_{0c}(C_p(s^D))\\
&+ 1 + \left(\frac{m-1}{m}\right)(l_{0i}(C_p(s^D)) + l_i(C_p(s^D))) - \frac{1}{m} - d'_p\\
&\le \frac{|M(p)|}{m} - d'_p\\
&+ \left(\frac{m-1}{m}\right)(1 + 2l_{0c}(C_p(s^D)) + l_{0i}(C_p(s^D)) + l_i(C_p(s^D)))\\
&\le \frac{|M(p)|}{m} + \left(\frac{m-1}{m}\right)(\ell(p) + Com(p)) - d'_p
\end{aligned}
$$

and subtracting the obvious inequality $L_{\max}(s^*) \ge \frac{|M(p)|}{m} - d'_p$, we obtain

$$
L_{\max}(s^D) - L_{\max}(s^*) \le \frac{m-1}{m}(\ell + \max_{k\in N} Com(k)).
$$

If $n - \ell \ge m$ then the above inequality together with (12) gives (10). On the other hand, if $n - \ell \le m - 1$ then from (12) we have

$$
L_{\max}(s^D) - L_{\max}(s^*) \le \left(\frac{m-1}{m}\right)\max_{k\in N} Com(k)
$$

and therefore proves (10) in this case as well.

It is easy to see that if there is no communication delay then the $P \mid prec,\ p_j = 1,\ c_{ij} = 1 \mid L_{\max}$ problem converts into the $P \mid prec,\ p_j = 1 \mid L_{\max}$ problem, and in this case the performance guarantees presented in Theorems 5 and 6, give the performance guarantees (1) and (2).

When all due dates are equal to zero the maximum lateness problem converts to the makespan problem. In this case, the algorithm of the due date modification with communication delay, presented above, can be viewed as a generalization of the well known Hu's algorithm (critical path) (Hu (1961)) for the model with a unit communication delay and with $-d'_j$ as the priority associated with each task *j*. Hence, (10) gives the following performance guarantees for the makespan problem,

$$C_{\max}(s^{CP}) - C_{\max}(s^*) \leq \begin{cases} \min\left\{\dfrac{n-\ell+1}{m} - 1, \dfrac{m-1}{m}\ell\right\} \\ \quad + \dfrac{m-1}{m}\max\limits_{k\in N} Com(k), & \text{if } n-\ell \geq m \\ \dfrac{m-1}{m}\max\limits_{k\in N} Com(k), & \text{otherwise} \end{cases} \quad (13)$$

where s^{CP} is a schedule constructed by the critical path method and s^* is an optimal schedule for a unit communication delay makespan problem.

3. CONCLUSIONS

We have consider the maximum lateness and makespan problem with a unit execution time task system, unit communication delay, precedence constraints, parallel identical processors and presented performance guarantees for the Brucker-Garey-Johnson and Hu's algorithm. We have presented a performance guarantee which establishes the relationship between the deviation of the criterion value from its optimum and the parameters characterizing the problem. The presented performance guarantees in some sense generalize the previously known performance guarantees for these algorithms. Future research will be directed towards arbitrary communication delays and arbitrary processing times.

REFERENCES

1. Brucker, P., Garey, M.R., and Johnson, D.S. (1977). Scheduling equal-length tasks under tree-like precedence constraints to minimise maximum lateness. *Mathematics of Operations Research*, 2: 275-284.
2. Finta, L., Liu, Z., Milis, I., and Bampis, E. (1996). Scheduling UET-UCT series-parallel graphs on two processors. *Theoretical Computer Science*, 162(2): 323-340.
3. Garey, M.R. and Johnson, D.S. (1976). Scheduling tasks with nonuniform deadlines on two processors. *Journal of the Association for Computing Machinery*, 23(3): 461-467.
4. Hu, T.C. (1961). Parallel sequencing and assembly line problems. *Operations Research*, 9: 841-848.
5. Lenstra, J.K., Veldhorst, M., and Veltman, B. (1996). The complexity of scheduling trees with communication delays. *Journal of Algorithms*, 20(1): 157-173.
6. Rayward-Smith, V.J. (1987). UET scheduling with unit interprocessor communication delays. *Discrete Applied Mathematics*, 18: 55-71.
7. Singh, G. and Zinder, Y. (2000a). Worst-case performance of two critical path type algorithms. *Asia-Pacific Journal of Operational Research*, 17: 101-122.
8. Singh, G. and Zinder, Y. (2000b). Worst-case performance of critical path type algorithms. *International Transactions in Operational Research*, 7: 383-399.
9. Singh, G. (2001). Performance of critical path type algorithms for scheduling on parallel processors, *Operations Research Letters*, 29(1): 17-30.
10. Singh, G. (2005). Scheduling UET-UCT outforests to minimize maximum lateness. *European Journal of Operational Research*, 165(2): 468-478.
11. Verriet, J. (2000). Scheduling tree-like task systems with non-uniform deadlines subject to unit-length communication delays. *Discrete Applied Mathematics*, 101: 269-289.