7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 - Paris - France
«Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination»

# BRANCH-AND-BOUND ALGORITHM TO MINIMIZE THE MAKESPAN FOR THE DEDICATED TWO-PROCESSORS TASK SCHEDULING PROBLEM WITH RELEASE DATES

**A. MANAA, C. CHU, A. YALAOUI**

Institut Charles Delaunay - OSI CNRS FRE 2848
Université de Technologie de Troyes 10010 Troyes cedex
adel.manaa@utt.fr, chengbin.chu@utt.fr, alice.yalaoui@utt.fr

**ABSTRACT :** *In this paper, we address the two-processors task scheduling problem on dedicated processors with release dates in order to minimize the makespan. In this problem a set of tasks using one or two processors simultaneously is to be scheduled without preemption. This problem is known to be NP-Hard in the strong sense. We propose a lower bound based on processor relaxation and show that it is equal to the optimal solution for the preemptive case. We propose two heuristics and a generalization of worst case analysis for any semi-active schedule. A set of dominance properties are established and a branch-and-bound algorithm is developed and tested on a set of randomly generated instances.*

**KEYWORDS :** *Scheduling, branch-and-bound, dedicated multiprocessor tasks, makespan.*

## 1. INTRODUCTION

Dedicated multiprocessor scheduling takes into account the fact that a task can be processed by more than one processor at a time. For this type of problems, a task may require a certain number of processors from a set of identical processors or a fixed set of different processors. In the latter we talk about dedicated multiprocessor task scheduling. Practical usefulness and some examples in computer systems or in some production systems can be found in ( Drozdowski, 1996).

In this paper, we focus on a two-processors task scheduling problem on dedicated processors in order to minimize the makespan. In this problem, a set of tasks with release dates requiring a dedicated processor or two dedicated processors simultaneously is to be scheduled without preemption. This problem is denoted in the literature as $P2|fix_j, r_j|C_{max}$ ($fix_j$ refers to the dedicated variant of multiprocessor problems). It is proved to be NP-hard in the strong sense in (Hoogeveen *et* al., 1994).

For the dedicated multiprocessor tasks scheduling class of problems, an overview of literature results was proposed in (Drozdowski, 1996). Many papers studied the minimization of the makespan for the multiprocessor problem $Pm|fix_j|C_{max}$, see (Blazewicz *et* al., 1986), ( Krawczyk and Kubale, 1985). branch-and-bound algorithm was also developed in (Bozoki and Richard, 1970) and in (Bianco *et* al., 1994). Its preemptive version was solved in *O(n)* in (Amoura *et* al., 1997). For the problem $Pm|fix_j, r_j|C_{max}$ no exact method was developed in the literature but a PTAS was given in (Bampis and Kononov, 2001). The special case with release dates and unitary processing times $Pm|fix_j, r_j, p_j = 1|C_{max}$ was solved in (Brucker and Krämer, 1996) in $O(R2^R n^{R+1})$ ($R = 2^m - 1$ is the number of task types) as a special case of a general RCPSP problem. For the preemptive cases we can mention polynomial time algorithms using linear programming and binary search for problems $Pm|fix_j, r_j, pmtn|f$ (kramer, 1995), (Bianco *et* al., 1997) and $Pm, win|fix_j, r_j, pmtn|f$ with $f \in \{C_{max}, L_{max}\}$ (Bianco *et* al., 1997).

Another typical set of problems which interested the multiprocessor scheduling community is the case were the number of processors is specified. Polynomial algorithms for the $Pm_{\in\{2,3,4\}}|fix_j, pmtn|Cmax$ are presented in (Bianco *et* al., 1994). An approximation algorithm is given in (Dell'Olmo *et* al., 1993) with worst case bound of 5/4 for $P3|fix_j|C_{max}$.

For The two processors case which is the basic theoretical case where multiprocessor scheduling difficulties arises, (Blazewicz *et* al., 1999) studied the problem with two parallel processors to minimize the makespan with release dates and optimal solution for unit execution time and and preemptable tasks was given. For the dedicated case, the optimal solution for $P2|fix_j|C_{max}$ is simply obtained by scheduling tasks of the same type together (Kämer, 1995). However, dealing with another criteria or considering release dates often yield to a strongly NP-hard problem.

For $P2|fix_j|f$ with $f \in \{\sum C_j, \sum w_j * C_j\}$ a branch-and-bound algorithm was developed in (Manaa *et* al., 2007) and in this paper we will focus on the strongly NP-Hard problem $P2|fix_j, r_j|C_{max}$. We propose a lower bound based on processor relaxation in $O(nlogn)$. We show that it has a criterion value equal to the one of the optimal solution of the preemptive case (Bianco *et* al., 1997) by linking it to the problem of scheduling with availability constraints . We give two different upper bounds with worst case analysis which we generalize to any semi-active schedule. We give a set of relevant dominance properties and we develop an efficient branch-and-bound algorithm.

In the next section some notations and a schedule representation are detailed. In section 3, the lower bound by relaxation $LB_r$ is given then a special method to build the optimal solution of the preemptive case $LB_p$ is detailed and is used to prove the equality of criterion values of the two bounds. Section 4 deals with two upper bounds $H^r$ and $H^p$ and gives a worst case analysis of them and a generalized performance result to any semi-active schedule. In section 5 the branch-and-bound algorithm is developed. Dominance properties are presented and computational results are given. Conclusions and prospects end this paper.

## 2. NOTATIONS AND SCHEDULE REPRESENTATION

We consider the scheduling of a set of $n$ tasks $N = \{1, .., j, ..n\}$ on two dedicated processors $\mathrm{P}_1$ and $\mathrm{P}_2$. A task $j$ is released at $r_j$ and has to be processed without preemption during its processing time $p_j$. $C_j$ is the completion time of task $j$.

- Task $j$ is called a $P_1 - task$ (respectively $P_2 - task$) if it requires processor $\mathrm{P}_1$ (respectively $\mathrm{P}_2$ ). Both type of tasks are monoprocessor tasks.
- Task $j$ is called $P_{12} - task$ if $S_j = \{P_1, P_2\}$ and is a biprocessor task.

$n_1$, $n_2$ and $n_{12}$ will denote the number of $P_1-$, $P_2-$ and $P_{12} - tasks$. We will also consider the following representation of a schedule $S$ as a sequence of a $P_{12} - task$ denoted by $P_i^{12}$ and a bloc of $P_1 - tasks$ and $P_2 - tasks$ denoted by $B_i$ where $P_0^{12}$ and any $B_i$ may be empty.

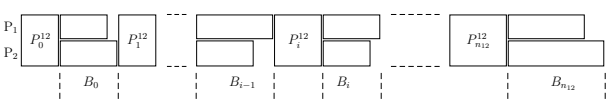


Figure 1: Schedule representation

## 3. LOWER BOUNDS

In this section we present the $LB_r$ lower bound based on biprocessor tasks relaxation. Then we give another way of looking to the preemptive case and the computation of its optimal solution $LB_p$. Finally we prove the equality of values of both bounds.

### 3.1. LOWER BOUND BY RELAXATION $LB_r$

The main idea here is to consider $P_{12} - tasks$ as the set of two separate tasks ($P_{12}^1 - tasks$ and $P_{12}^2 - tasks$) that have to be scheduled on each processor separately. If we consider this relaxation, we will have two different and simple problems. Scheduling $P_{12}^1 - tasks$ and $P_1 - tasks$ on processor $\mathrm{P}_1$ and scheduling $P_{12}^2 - tasks$ and $P_2 - tasks$ on processor $\mathrm{P}_2$. The optimal solutions ($C_{opt}^1$ and $C_{opt}^2$) are simply obtained by scheduling tasks in non-decreasing order of their release dates on each processor.

As the optimal solution for $P2|fix_j, r_j|C_{max}$ is clearly a feasible solution for the problem $1|r_j|C_{max}$ on processor $\mathrm{P}_1$ by considering $P_1 - tasks$ and $P_{12}^1 - tasks$ and a feasible solution for the same problem on processor $\mathrm{P}_2$ by considering $P_2 - tasks$ and $P_{12}^2 - tasks$. Each of the optimal solutions is a lower bound for $P2|fix_j, r_j|C_{max}$ and in the following $LB_r$ will correspond to the largest one.

Since the main step is to sort tasks according to their release dates, the algorithm to compute this bound is polynomial and bounded by $O(nlogn)$.

### 3.2. OPTIMAL SOLUTION FOR THE PREEMPTIVE CASE

An optimal solution for this problem was given in (Bianco*et* al., 1997) in two steps algorithm by suspending processing of monoprocessor tasks if there are any, and schedule biprocessor tasks first. Then start processing of the remaining monoprocessor tasks or their parts immediately after completion of biprocessor tasks. Here we give another way to look at this problem as a variant of scheduling tasks with unavailability periods and arbitrary profiles.

In the preemptive case there is an optimal schedule in which biprocessor tasks are scheduled exactly at their release dates. By scheduling $P_{12} - task$ at their release dates, we obtain on each processor an equivalent problem to $1, h_k|r_j, pmtn|C_{max}$ which is scheduling with availability constraints.

Each processor is unavailable during $h_k = [r_k^{12}, r_k^{12} + p_k^{12}]$ $(k = 1..n_{12})$.

According to (Sanlaville, 1992) even for $M$ parallel machines and arbitrary profiles (periods of unavailability) any schedule produced by $L(R)PT$ (Largest (Remaining) Processing Time first) is optimal for the

equivalent problems.

In our case, *L(R)PT* is then also optimal and we can add that any schedule obtained by ordering tasks in their release dates is also optimal and we have the same optimal solution as in the literature. But it is still an interesting interpretation and a good equivalence to make with scheduling with unavailability periods and arbitrary profiles which can be useful in the multiprocessor context in general and which we will use for an upcoming result in this paper.

In the following $LB_p$ will denote the lower bound of $P2|fix_j, r_j|C_{max}$ obtained from the optimal solution of the preemptive case.

### 3.3. LOWER BOUNDS CRITERION VALUES $LB_r = LB_p$

Here we prove that lower bound from relaxation $LB_r$ gives an optimal criterion value of the makespan of the preemptive case, *i.e* $LB_r = LB_p$.

*Proof:* Consider here the preemptive case of the problem: $P2|fix_j, r_j, pmtn|C_{max}$ and an instance $I$. Let denote by $LB_p(I)$ its optimal solution when preemption is allowed, $LB_r(I)$ its relaxed lower bound. Without lost of generality we assume that $LB_r(I)$ corresponds to the maximum completion time on the first processor ($P1$).

Let now consider on $P1$ the equivalent instance $I_1^H$ with $P_1 - tasks$ and unavailability periods $h_k(I_1^H) = [t_k^{12}, t_k^{12} + p_k^{12}]$ $(k = 1..n_{12})$ where $t_k^{12}$ is the start time of the $k^{th}$-$P_{12} - task$. We will thus obtain a set of unavailability periods or more precisely a profile (as used in the context of scheduling with availability constraints).

The problems $1, h_k(I_1^H)|r_j, pmtn|C_{max}$ and $1, h_k|r_j, pmtn|C_{max}$ on processor $P1$ with $h_k = [r_k^{12}, r_k^{12} + p_k^{12}]$ $(k = 1..n_{12})$ only differ by their respective profiles and referring to result in (Sanlaville, 1992) they have the same criterion value and we can conclude that $LB_r$ gives the optimal criterion value for the problem $P2|fix_j, r_j, pmtn|C_{max}$.

## 4. UPPER BOUNDS AND PERFORMANCE RESULTS

In this section we give two upper bounds obtained from the schedule sequences given by $LB_r$ and $LB_p$. We prove that for these upper bounds and for any semi-active schedule the worst case performance ratio is 2.

### 4.1. SEMI-ACTIVE SCHEDULE WORST CASE ANALYSIS

**Proposition:** Any semi-active schedule has a worst case performance ratio of 2.

*Proof:* Let $C_{max}^*$ be the makespan of the optimal solution and $r_{max} = max_j r_j$. Consider now the schedule $S_m$ obtained by scheduling tasks of the same type together beginning with biprocessor tasks at time $t = r_{max}$ and let $C_{max}(S_m)$ its completion time.

We have $C_{max}(S_m) - r_{max} \leq C_{max}^*$ which is obvious since $C_{max}(S_m) - r_{max}$ is the optimal criterion value when no release dates are considered. As $r_{max} < C_{max}^*$ then $C_{max}(S_m) < 2 * C_{max}^*$.

Now, if we consider any semi-active schedule $S$ we have $C_{max}(S) \leq C_{max}(S_m)$ and $C_{max}(S) \leq C_{max}^*$.

In the following we will focus on two heuristics to obtain upper bounds for our problem: the $H^r$ heuristic which gives a semi-active schedule and the $H^p$ heuristic and for both of them we will prove that 2 is a tight performance ratio.

### 4.2. THE $H^r$ UPPER BOUND

Consider the following heuristic $H^r$:

- *Step 1*: Solve the relaxed problem on each processor and choose the largest lower bound between $C_{opt}^1$ and $C_{opt}^2$( suppose it is $C_{opt}^1$ on processor $P_1$).
- *Step 2*: Consider the schedule of tasks thus obtained (here on processor $P_1$) and insert on processor $P_2$ when it is possible $P_2 - tasks$ between $P_{12} - tasks$ otherwise at the end of the sequence.

The obtained result is an upper bound for $P2|fix_j, r_j|C_{max}$.

#### Worst case analysis

**Result:** The heuristic $H^r$ provides a semi-active schedule and 2 is its tighter performance ratio.

Let consider the following instance with three tasks: a $P_1$-task $A$ (with $p_A = 2, r_A = k$) where $k$ is a non negative integer, a $P_2$-task $B$ (with $p_B = k, r_B = 0$) and a $P_{12}$-task $C$ (with $p_C = 1, r_C = k - 1$).

We get the schedules of *figures 2, 3, 4* and *5*.

For this instance we have $\frac{C_{max}^{H^r}}{C_{max}^{opt}} = \frac{2*k}{k+3}$.

This ratio tends to 2 when $k$ tends towards $+\infty$. As a consequence 2 is a tight worst-case performance bound.

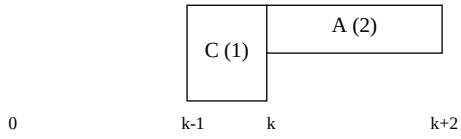


Figure 2: Lower bound on $P1$:$C_{max}^1 = k + 2$.

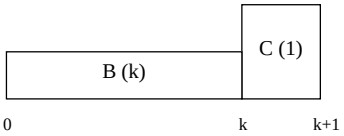


Figure 3: Lower bound on $P2$:$C^2_{max} = k + 1$.

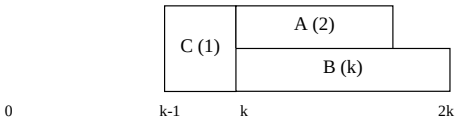


Figure 4: Heuristic $H^r$ schedule: $C^{H^r}_{max} = 2k$.

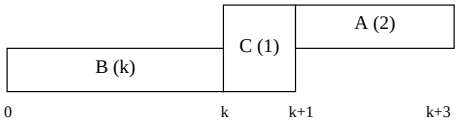


Figure 5: Optimal schedule: $C^{opt}_{max} = k + 3$.

### 4.3. THE $H^p$ UPPER BOUND

This upper bound is obtained from the optimal solution of the preemptive case which we solve optimally by scheduling $P_{12}-tasks$ exactly at their release dates and then inserting $P_1-tasks$ and $P_2-tasks$ preemptively in non decreasing order of their release dates.

Once the preemptive case is solved, for each biprocessor $P^{12}_i$ that preempts monoprocessor tasks we shift to the right all tasks scheduled after $P^{12}_i$ by inserting an idle time equal to the preempted part preceding $P^{12}_i$ ( *figures 6* and *7*). When a $P^{12}_i$ preempts a $P_1-$ and a $P_2-task$, the inserted idle time is equal to the maximum value of the preempted parts of monoprocessor tasks preceding $P^{12}_i$.

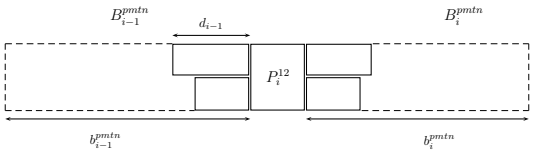


Figure 6: Preempted sequence

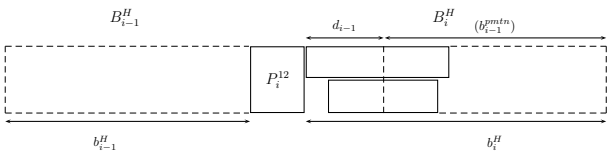


Figure 7: Heuristic sequence

Note that as it is described, the schedule obtained by this heuristic can be non semi-active. Even in this case we prove the following result:

**Worst case analysis**

**Result:** 2 is the tight performance upper bound of the $H^p$ heuristic.

*Proof:*For a schedule $S$, by considering the schedule representation of *figure 1*, we have $C_{max}(S) = \sum_{i=0}^{n_{12}}(p^{12}_i + b_i)$ with $p^{12}_0 = 0$ and $b_i$ is the length of bloc $B_i$ .

Let $S_{pmtn}$ be the optimal schedule of the preemptive case and $S_H$ the schedule obtained by $H^p$.

Let us focus on a sequence $B_{i-1}, P^{12}_i, B_i$ in the $S_{pmtn}$ schedule such that $P^{12}_i$ preempts a $P_1$ and a $P_2-task$ (*figure 6*) and also consider its corresponding sequence obtained by heuristic $H^p$: $B^H_i, P^{12}_i, B^H_{i+1}$ in $S_H$ (*figure 7*).

To simplify we will denote $b^{pmnt}_i$ by $b_i$.

We have: $b^H_0 = b_0$ and for $i \geq 1$, $b^H_i = b_i + d_{i-1}$. $d_{i-1}$ represents here the length of the inserted part.

As $d_{i-1} \leq b_{i-1}$ then $b^H_i \leq b_i + b_{i-1}$.

So: $C^H_{max} = \sum_{i=0}^{n_{12}}(p^{12}_i + b^H_i) \leq p^{12}_0 + b_0 + \sum_{i=1}^{n_{12}}(p^{12}_i + b_i + b_{i-1}) = \sum_{i=0}^{n_{12}}(p^{12}_i + b_i) + \sum_{i=1}^{n_{12}} b_{i-1} = C^{pmtn}_{max} + \sum_{i=1}^{n_{12}} b_{i-1}$.

And $\sum_{i=1}^{n_{12}} b_{i-1} \leq C^{pmtn}_{max}$. Then we have $C^H_{max} \leq 2 * C^{pmtn}_{max}$ and the following result: $C^H_{max} \leq 2 * C^{opt}_{max}$.

Now we will prove that this bound is tight by considering the following instance with three tasks:

A $P_1$-task $A$ (with $p_A = k, r_A = 0$) where $k \geq 0$, a $P_2$-task $B$ (with $p_B = k + 1, r_B = 0$) and a $P_{12}$-task $C$ (with $p_C = 1, r_C = k$).

We get the following schedules :

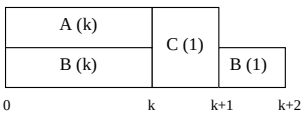


Figure 8: Preemptive schedule:$C^{pmtn}_{max} = k + 2$.

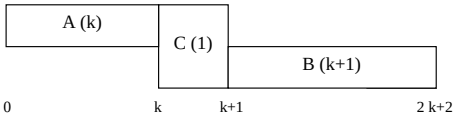


Figure 9: $H^p$ schedule: $C^{H^p}_{max} = 2k + 2$.

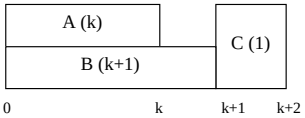


Figure 10: Optimal schedule: $C^{opt}_{max} = k + 2$.

For this special schedule we have $\frac{C^{H^p}_{max}}{C^{opt}_{max}} = \frac{2*k+2}{k+2}$.

This ratio tends to 2 when $k$ tends towards $+\infty$. As a consequence 2 is a tight worst-case performance bound.

## 5. THE BRANCH-AND-BOUND ALGORITHM

Our branch-and-bound algorithm will consist on choosing among a set of possible child nodes, the ones that can lead to an optimal schedule. A child node chooses a task which must fulfill a set of conditions to be added to its ascendants to obtain a partial schedule. Comparing the lower bound of this partial schedule to the upper bound decide wether to prune the node or not.

We use a breadth first exploration technique: At each level of the search tree and for each node belonging to this level we generate its child nodes. So that we obtain all the nodes which form the next level to be explored.

### 5.1. DOMINANCE PROPERTIES

In the following we present three dominance properties.

*Property 1*: In an optimal schedule, the $P_{12} - tasks$ are sequenced in non decreasing order of their release dates.

*Property 2*: In a bloc $P_1 - tasks$ (respectively $P_2 - tasks$) are sequenced in non decreasing order of their release dates.

The precedent properties are obvious and can be easily proved by an interchange argument.

*Property 3*: A schedule $S$ is dominated if a $P_{12} - task$ $C$ is scheduled after a $P_1 - task$ $A$ and a $P_2 - task$ $B$ (*figures 11, 12* ) and:

1- $r_C \leq min(r_A, r_B)$ or

2- $r_A \leq r_C \leq r_B$ and $p_B \geq p_A$ or

3- $r_A \leq r_C \leq r_B$, $r_B - r_C \geq p_A - p_B \geq 0$ and $p_C \geq p_A - p_B \geq 0$.

*Proof*:

Let consider the case where $r_A \leq r_C \leq r_B$ and let denote by $S_1$ the schedule where tasks $A$ and $B$ precede task $C$ and $C(S_1)$ its completion time and $S_2$ (with completion time $C(S_2)$) the other case where task $C$ precedes tasks $A$ and $B$.

$C(S_1) = max(r_A + p_A, r_B + p_B) + p_C$.

$C(S_2) = max(r_C + p_C + p_A, max(r_C + p_C, r_B) + p_B)$

if $p_B \geq p_A$ we have two possible cases:

case1: $r_C + p_C \geq r_B$

$C(S_1) = r_B + p_B + p_C$ and $C(S_2) = max(r_C + p_C + p_A, r_C + p_C + p_B) = r_C + p_C + p_B$ and as $r_B \geq r_C$ then $C(S_1) \geq C(S_2)$

case2: $r_C + p_C \leq r_B$

$C(S_1) = r_B + p_B + p_C$ and $C(S_2) = max(r_C + p_C + p_A, r_B + p_B) = r_B + p_B$ and then $C(S_1) \geq C(S_2)$

else if $p_B \leq p_A$, for the each possible case we have:

case1: $r_C + p_C \geq r_B$

$C(S_2) = max(r_C + p_C + p_A, r_C + p_C + p_B) = r_C + p_C + p_A$ and $C(S_1) - C(S_2) = max(r_A + p_A, r_B + p_B) - p_A - r_C \geq r_B + p_B - p_A - r_C$ so if $r_B - r_C \geq p_A - p_B$ then $C(S_1) \geq C(S_2)$

case2: $r_C + p_C \leq r_B$

$C(S_1) - C(S_2) = max(r_A + p_A, r_B + p_B) - max(r_C + p_C + p_A, r_B + p_B) \geq r_B + p_B + p_C - max(r_C + p_C, r_B) - p_A \geq p_B + p_C - p_A$ and if $p_C \geq p_A - p_B$ then $C(S_1) \geq C(S_2)$.

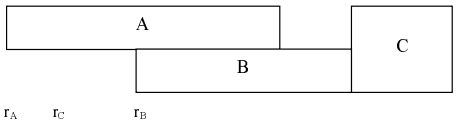


Figure 11: Dominated case.

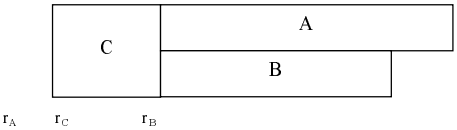


Figure 12: Non dominated case.

*Property 4 - symmetrical nodes*:

An important technique to reduce the number of nodes in a level of the tree especially when breadth-first exploration is chosen consists on eliminating symmetrical nodes. Two nodes are said symmetrical when they have the same set of already scheduled tasks.

Consider two nodes $N(i)$ and $N(j)$ in the same level $l$ and $S_c^1(i)$, $S_c^2(i)$ and $S_c^{12}(i)$ (respectively $S_c^1(j)$, $S_c^2(j)$ and $S_c^{12}(j)$) be the set of child nodes of the remaining $P_1 - tasks$, $P_2 - tasks$ and $P_{12} - tasks$ of $N(i)$ (respectively of $N(j)$).

Let $C_m^k(i)$ (respectively $C_m^k(j)$) be the completion time of the last task scheduled on processor $k$ in node $N(i)$ (respectively of $N(j)$), $k \in \{1, 2\}$.

- if $C_m^1(i) \leq C_m^1(j)$ no child node from $S_c^1(j)$ is generated.

As $S_c^1(j) = S_c^1(i)$ and $C_m^1(i) \leq C_m^1(j)$ then adding the same $P_1 - task$ to the partial schedule of $N(i)$ at level $l$ will always lead to a better partial schedule at level $l+1$ then adding it to $N(j)$.

- if $C_m^2(i) \leq C_m^2(j)$ no child node from $S_c^2(j)$ is generated.
- if $max(C_m^1(i), C_m^2(i)) \leq max(C_m^1(j), C_m^2(j))$ no child node from $S_c^{12}(j)$ is generated.
- if $(C_m^1(i) \leq C_m^1(j), C_m^2(i) \leq C_m^2(j)$ the node $N(j)$ is pruned.

### 5.2. COMPUTATIONAL RESULTS

In this section we give some computational results of randomly generated instances. The Branch-and-Bound algorithm was programmed using a C/C++ compiler on a $P4$ $3.00GHz$ and $1GB$ $RAM$ workstation.

We consider here 5 types of problems ($P_{typ}$) as in table 1: $n$, $n_1$, $n_2$ and $n_{12}$ respectively denote an integer parameter($n \in \{5, 10\}$), the number of $P_1 - tasks$, $P_2 - tasks$ and $P_{12} - tasks$ and let [.] be the floor value.

| . | Problem type $P_{typ}$ 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $n_1 =$ | $n$ | $n$ | $n$ | $n$ | $[n/2]$ |
| $n_2 =$ | $[n/2]$ | $n$ | $[n/2]$ | $n$ | $[n/2]$ |
| $n_{12} =$ | $[n/2]$ | $[n/2]$ | $n$ | $n$ | $n$ |

Table 1: Problem types.

Integer processing times are randomly generated from uniform distribution $[1, 50]$. Release date are randomly generated from a uniform distribution $[0, \alpha * (s_{12} + \frac{s_1+s_2}{2})]$ where $\alpha \in \{0.5, 1, 1.5\}$and $s_1$, $s_2$ and $s_{12}$ are respectively the sum of processing times of $P_1 - tasks$, $P_2 - tasks$ and $P_{12} - tasks$ (see $Table1$). We consider group of instances having the same parameters $n$, $P_{typ}$ and $\alpha$ and for the computational results 10 instances of each group are considered and average values are given.

Tables 2 and 3 summarize the numerical results in term of average computational time (in seconds) and average number of generated nodes. These results show that instances corresponding to the problem type $P_{typ} = 4$ with tightest distribution of release dates ($\alpha = 0.5$) are the hardest ones. Moreover comparing instances of approximatively same total number of tasks ($P_{typ} = 2$ and $P_{typ} = 3$) shows that the ones with more monoprocessor tasks ($P_{typ} = 2$) are all almost harder.

Table 4 gives the relative error of the upper bound with the optimal value. First value is equal to the

| $n = 5$ | $P_{typ}$ | | | | |
|---|---|---|---|---|---|
| $\alpha$ | 1 | 2 | 3 | 4 | 5 |
| 0.5 | 0 | 0.01 | 0.08 | 0.04 | 0 |
| 1 | 0 | 0.11 | 0 | 0.03 | 0 |
| 1.5 | 0 | 0 | 0.01 | 0.01 | 0 |

Instances with $n = 5$

| $n = 10$ | $P_{typ}$ | | | | |
|---|---|---|---|---|---|
| $\alpha$ | 1 | 2 | 3 | 4 | 5 |
| 0.5 | 11.84 | 238.5 | 26.49 | 562.9 | 0.05 |
| 1 | 0.38 | 1.18 | 0.34 | 4.43 | 0.05 |
| 1.5 | 0.08 | 0.66 | 0.15 | 23.71 | 0.01 |

Instances with $n = 10$

Table 2: Computational time.

| $n = 5$ | $P_{typ}$ | | | | |
|---|---|---|---|---|---|
| $\alpha$ | 1 | 2 | 3 | 4 | 5 |
| 0.5 | 183 | 791 | 462 | 3913 | 51 |
| 1 | 80 | 1279 | 28 | 844 | 36 |
| 1.5 | 207 | 136 | 85 | 53 | 0 |

Instances with $n = 5$

| $n = 10$ | $P_{typ}$ | | | | |
|---|---|---|---|---|---|
| $\alpha$ | 1 | 2 | 3 | 4 | 5 |
| 0.5 | 35003 | 129422 | 80857 | 204338 | 152 |
| 1 | 998 | 1692 | 505 | 5544 | 138 |
| 1.5 | 165 | 964 | 206 | 28954 | 48 |

Instances with $n = 10$

Table 3: Number of nodes generated

average value among each group of instances of the deviation of $H^p$ ( $devH^p = 100 * (H^p - Opt)/Opt$) and the second value (between brackets)is the one for $H^r$ ( $devH^r = 100 * (H^r - Opt)/Opt$). Results show good quality of $H^p$ especially for scattered distribution of release dates and detailed results point out that average values are relevant. It also dominates $H^r$ in most cases. Detailed results for $H^r$ show that it reaches high relative error in some instances and low relative error in other instances belonging to the same type of instances (among a set of ten instances) which point out its fluctuant behavior. However this upper bound remains interesting in the case of scattered distributions with small amount of monoprocessor tasks.

| $n = 5$ | $P_{typ}$ | | | | |
|---|---|---|---|---|---|
| $\alpha$ | 1 | 2 | 3 | 4 | 5 |
| 0.5 | 9.6(12.0) | 5.2(15.6) | 0.6(7.2) | 3.4(19.7) | 2.0(21.6) |
| 1 | 9.0(4.1) | 3.3(17.5) | 2.6(4.9) | 3.5(7.5) | 0.8(6.1) |
| 1.5 | 0.7(3.1) | 1.7(9.2) | 2.1(4.9) | 1.8(9.29) | 0.7(6.9) |

Instances with $n = 5$

| $n = 10$ | $P_{typ}$ | | | | |
|---|---|---|---|---|---|
| $\alpha$ | 1 | 2 | 3 | 4 | 5 |
| 0.5 | 5.3(9.2) | 1.3(24.8) | 2.0(16.4) | 2.6(27.1) | 2.8(21.9) |
| 1 | 8.5(4.6) | 2.9(11.1) | 1.5(5.2) | 2.2(13.0) | 1.5(5.9) |
| 1.5 | 6.7(4.1) | 1.9(8.3) | 1.0(5.7) | 6.4(17.9) | 1.3(7.3) |

Instances with $n = 10$

Table 4: Relative errors: $devH^P$ $(devH^r)$

## 6. CONCLUSION

In this paper we presented a set of dominance properties, a lower bound which was proved to give optimal criterion value for the preemptive case and two heuristics with tight performance of 2 for the $P2|fix_j, r_j|C_{max}$ problem. A branch-and-bound algorithm was constructed and computational results are given. The results suggest that at this level the algorithm can handle up to than 30 tasks for the hardest instances in less than 10 minutes. Techniques used for this two-processors problem are useful and have to be developed to propose an exact algorithm for the general case with $m$ processors.

## REFERENCES


Amoura. A.K., E. Bampis, C. Kenyon and Y. Manoussakis, 1997. Scheduling Independent Multiprocessor Tasks *European Symposium on Algorithms proceedings*, 1–12.

Bampis. E., A. Kononov, 2001. On the approximability of scheduling multiprocessor tasks with time dependent processing and processor requirements *Proceedings of the 15th International Parallel and Distributed Processing Symposium 2001*, San Francisco,United States of America.

Bianco. L., J. Blazewicz, P. Dell'Olmo and M. Drozdowski, 1994. Scheduling preemptive multiprocessor tasks on dedicated processors *Performance Evaluation 20*, 361–371.

Bianco. L., J. Blazewicz, P. Dell'Olmo and M. Drozdowski, 1997. Preemptive multiprocessor task scheduling with release tims and time windows *Annals of Operations Research 70*, 43–55.

Bozoki. G and J. P. Richard, 1970. A Branch-and-Bound Algorithm for the Continuous-Process Job-Shop Scheduling Problem *Twenty-first Annual Conference of AIIE, Clevland, May 1970, volume II, N 3*, 246–252.

Blazewicz. J, P. Dell'Olmo, and M. Drozdowski, 2002. Scheduling multiprocessor tasks on two parallel processors *RAIRO, Oper. Res. 36 1*, 37-51.

Blazewicz. J, M. Drobowski and J. Werglaz, 1986. Scheduling Multiprocessor Tasks to Minimize Schedule Length *IEEE Transactions on Computers, C-35*, 383–393.

Brücker. P, A. Krämer, 1996. Polynomial algorithms for resource-constrained and Multiprocessor task scheduling problems *European Journal of Operational research 90*, 214–226.

Dell'Olmo. P., M. G. Speranza and Z. Tuza, 1993. Hard cases and approximation results on three dedicated processors scheduling problem *Technical report No. 359, Instituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche, Rome, Italy.*

Drozdowski. M., 1996. Scheduling multiprocessor tasks - An overview *European Journal of Operational Research 94*, 215–230.

Hoogeveen. J.A., S. L. van de Velde and B. Veltman, 1994. Complexity of Scheduling Multiprocessor Tasks with prespecified Processor Allocations *Dscrete Applied Mathematics 55*, 259–272.

Krawczyk. H. and M. Kubale, 1985. An approximation algorithm for diagnostic test scheduling in multicomputer systems, *IEEE Transactions on Computers C-34*, 869–872.

Krämer. A., 1995. *Scheduling Multiprocessor Tasks on dedicated processors*, Thesis, Universität Osnabrück.

Manaa. A., C. Chu and A. Yalaoui, 2007. Branch-and-Bound algorithms to minimize total weighted and total completion time for the dedicated two-processor task scheduling problems $P2|fix_j|\sum w_jC_j$ and $P2|fix_j|\sum C_j$ *Proceedings of the Conference*

*I4E2 on International Conference on Industrial Engineering and Systems Management (IESM'07)*, Beijing, China, 8 pages.

Sanlaville. E, 1992. Scheduling preemptive independent tasks on a variable profile, *technical report, M.A.S.I. 92.4, janvier 1992.*