The scheduling algorithms PF et PD² are monotonous.

Annie Choquet-Geniet
25 juin 2010

1 PFair Scheduling

For any real $x$, $\lfloor x \rfloor$ denotes the greatest integer less than or equal to $x$ and $\lceil x \rceil$ the smallest integer greater than or equal to $x$. We consider a platform composed of $m$ identical processors and applications composed of $n$ periodic tasks with implicit deadlines $\tau_i(C_i, T_i)$. $C_i$ is the worst case execution time of the task (WCET) and $T_i$ is its period. We assume that the tasks are submitted to hard deadlines, and that they can share critical resources (shared variables, memory segments, ...). A task consists of an infinity of instances (or jobs) activated at times $k.T_i$ ($k \in \mathbb{N}$). At each new activation, the previous instance must have completed execution. We assume that parallelism is forbidden: a task cannot be processed on several processors at the same time. We finally assume that the temporal parameters are known and deterministic. We denote $T$ the hyperperiod of the system defined by $T = LCM\{T_1, T_2, \ldots, T_n\}$.

Formally, a schedule is an application $S : \mathbb{N} \times \{1, \ldots, n\} \rightarrow \{0, 1\}$ such that $\sum_{i=1}^{n} S(t, i) \leq m$. Intuitively, $S(t, i) = 1$ means that the task $\tau_i$ is processed within the time interval $[t, t+1)$ on one processor. In the sequel, we use the notation $S(t, i) = S_i(t)$.

PFair strategies have been developed for multiprocessor systems. They are optimal for independent task sets, in multiprocessor environment. The basic idea is that the tasks are processed at regular rate, equal to their utilization factor ($u_i = \frac{C_i}{T_i}$). This means that at time $t$, the task $\tau_i$ must have been processed for $\frac{C_i}{T_i}.t$ time units. Now, the number of already processed time units must be integer, it is thus approximated either by $\lfloor u_i.t \rfloor$ or by $\lceil u_i.t \rceil$. Formally, we have the following definition:

**Definition 1.** A schedule $S$ is PFair if and only if

$$\forall t \in \mathbb{N}, \forall i \in \{1, \ldots, n\}, -1 < u_i.t - \sum_{j=0}^{t-1} S_i(j) < 1.$$
The task is Urgent
The task is contending

Processed execution time

\[ W(t) = u \times t \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ A \text{ non } P\text{-fair execution} \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = u \times t \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]

\[ W(t) = C_i(t) \pm 1 \]

\[ Ideal(t) = u \times t \]

\[ Ideal(t) = C_i(t) \pm 1 \]

\[ C_i(t) \]

\[ P \]

\[ 2P \]

\[ 2C \]

\[ \text{Processed execution time} \]

\[ \text{Time} \]
where \( t' \) is the first time after \( t \) such that \( \alpha_i(t') = 0 \). The algorithm PF is depicted as follows [2]:

1. Urgent tasks are scheduled (let \( k \) be their number)
2. Contending tasks are sorted, using the lexicographical order on their characteristic substrings (with \(-\prec 0 \prec +\)).
3. The \((m-k)\) first Contending tasks are scheduled. Tie-breaks are resolved by a deterministic rule.

1.2 The algorithm PD\(^2\) [1]

![Fig. 2 – Feasibility windows](image)

Each task \( \tau_i \) is decomposed into an infinity of unitary subtasks \( \tau_{ij} \) (\( j \geq 1 \)). The \( j\)th subtask must then be scheduled within its feasibility window \( I_{ij} \) delimited by \( r_{ij} = \lfloor \frac{j-1}{u_i} \rfloor \) and \( d_{ij} = \lceil \frac{j}{u_i} \rceil \) (see figure 2). We denote \( |I_{ij}| \) the size of the window \( I_{ij} \): \( |I_{ij}| = d_{ij} - r_{ij} \). For each subtask \( \tau_{ij} \), the successor bit is defined by

\[
b_{ij} = d_{ij} - r_{ij} + 1 = \lceil \frac{j}{u_i} \rceil - \lfloor \frac{j}{u_i} \rfloor
\]

It is equal to 1 if the two consecutive windows \( I_{ij} \) and \( I_{ij+1} \) overlap by one time unit and it is equal to 0 if they are disjoint. Finally, each subtask \( \tau_{ij} \) has a group deadline \( D_{ij} \).

If \( u_i \geq \frac{1}{2} \) (the task is light), this group deadline is equal to 0: \( D_{ij} = 0 \).
If \( u_i < \frac{1}{2} \) (the tasks is heavy), it has been proved that its first feasibility window is of size 2, and the other ones have sizes equal either to 2 or to 3. A group is a sequence of consecutive subtasks \( \tau_{ij}, \tau_{ij+1}, \ldots, \tau_{iv} \) such that:

- \( b_{ij} = 1 \)
- \( |I_{ij}| = 2 \) and \( b_{ij} = 1 \), \( \forall k \) such that \( u < k \leq v \)
- \( |I_{ij+1}| = 3 \) or \( |I_{ij+1}| = 2 \) and \( b_{ij+1} = 0 \)
In this case, the group deadline is equal to: 
\[
\begin{cases} 
  d_i^v + 1 & \text{if } |I_i^{v+1}| = 3 \\
  d_i^v & \text{if } b_i^v = 0
\end{cases}
\]

And finally we have \( D_i^2 = \text{Min}\{ t \text{ s.t. } d_i^u \leq t \text{ and } t \text{ is a group deadline}\} \). The algorithm PD\(^2\) is then defined as a dynamic priority algorithm: a subtask \( \tau_i^u \) has priority over a subtask \( \tau_k^v \) if and only if:

1. \( d_i^u < d_k^v \). Its deadline is the nearest.
2. \( d_i^u = d_k^v \) and \( b_i^u > b_k^v \).
3. \( d_i^u = d_k^v \) and \( b_i^u = b_k^v = 1 \) and \( D_i^u > D_k^v \).

## 2 Monotonous schedules

Monotonous scheduling are such that at any time \( t \), the pending instance of a task cannot have been processed less than the instance pending one hyperperiod later. In the sequel, we use the following notations:

- \( P_i(t) \) is the pending instance of the task \( \tau_i \) at time \( t \) (the last instance released at or before \( t \)).
- \( RCT_i(t) \) is the remaining computation time for the instance \( PI_i(t) \) of the task \( \tau_i \).
- \( overline{RCT}_i(t) \) corresponds to the elapsed computation time of the instance \( PI_i(t) \).
- \( W_i(t,t') \) is the processed execution time for the task \( \tau_i \) between time \( t \) and time \( t' \).

The monotony property is formally expressed by:

**Definition 2.** A schedule is **monotonous** iff \( \forall t, \forall i \in 1..n \),

\[
RCT_i(t) + overline{RCT}_i(t + T) \leq C_i
\]

This property can equivalently be expressed as:

\[
overline{RCT}_i(t) \geq overline{RCT}_i(t + T)
\]

We consider PF and PD\(^2\) schedules. We assume that ties are resolved by a deterministic rule. E.g., the task with the smallest number will have the higher priority.

**Proposition 3.** PF and PD\(^2\) schedules with a deterministic tie-break rule are monotonous.

**Proof**

In both proofs, we reason by contradiction. We thus assume that the result doesn’t hold. We want to prove that \( \forall \tau_i, \forall t \geq r_i, RCT_i(t) \geq RCT_i(t + T) \).

We assume the opposite. There is thus a time \( t_0 \) which is the first time where at least one task \( \tau_{i_0} \) verifies \( RCT_{i_0}(t_0) < RCT_{i_0}(t_0 + T) \). We thus have

\[
\begin{cases} 
  RCT_{i_0}(t_0) < RCT_{i_0}(t_0 + T) & (1) \\
  RCT_{i_0}(t_0 - 1) = RCT_{i_0}(t_0 - 1 + T) & (2)
\end{cases}
\]

4
The task $\tau_{i_0}$ is thus processed at time $t_0 + T - 1$ but not at time $t_0 - 1$.

1 - We first consider the algorithm PF.

**Lemma 4.** The characteristic substrings of $\tau_i$ are periodic with period $T_i$.

This is a straightforward consequence of the definitions.

We assume that $t_0$ et $\tau_{i_0}$ are defined. We discuss on the status of the task $\tau_{i_0}$ at time $t_0 - 1$.

- The task $\tau_{i_0}$ cannot be Urgent at time $t_0 - 1$, else, it would have been processed.

- If it is Tnegru, this means that $W_{i_0}(0, t_0 - 1) + 1 \geq u_{i_0} \times t_0 + 1$ (should the task be processed, then its execution curve would be above the upper limit line). But in this case, according to the equality (2), we have:

  $W_{i_0}(0, t_0 + T - 1) + 1 = W_{i_0}(0, t_0 - 1) + 1 + \frac{T}{\tau_{i_0}} \times C_{i_0}$ and thus we have

  $W_{i_0}(0, t_0 + T - 1) + 1 \geq u_{i_0} \times t_0 + 1 + \frac{T}{\tau_{i_0}} \times C_{i_0} = u_{i_0} \times (t_0 + T) + 1$. Thus $\tau_{i_0}$ is also Tnegru at time $t_0 + T - 1$ thus it is processed. This contradicts our assumption.

- The task $\tau_{i_0}$ is thus Contending at time $t_0 - 1$. It is not processed, thus there exist $m$ tasks $\tau_{i_1}, \tau_{i_2}, \ldots, \tau_{i_m}$ which are processed. They are thus either Urgent or Contending.

  - If a task $\tau_{i_j}$ is Urgent at time $t_0 - 1$, we have $W_{i_j}(0, t_0 - 1) \leq u_{i_j} \times t_0 - 1$ (shouldn’t the task be processed, then its execution curve would be located below the lower limit line). Besides, from the definition of $t_0$, we have $\overline{RCT}_{i_j}(t_0 - 1) \geq RCT_{i_j}(t_0 + T - 1)$ thus we have $W_{i_j}(0, t_0 - 1) + \frac{\tau_{i_j}}{T} \times C_{i_j} \geq W_{i_j}(0, t_0 + T - 1)$. It follows that $W_{i_j}(0, t_0 + T - 1) \leq u_{i_j} \times t_0 - 1 + \frac{\tau_{i_j}}{T} \times C_{i_j} = u_{i_j} \times (t_0 + T) - 1$. The task $\tau_{i_j}$ is thus also Urgent at time $t_0 + T - 1$ and it is thus processed.

  - If a task $\tau_{i_j}$ is Contending, it has priority over $\tau_{i_0}$ because:

    - either its characteristic substring is smaller (for the lexicographical order) than the substring of $\tau_{i_0}$: $\alpha(i_j, t_0 - 1) \prec \alpha(i_0, t_0 - 1)$;

    - either both substrings are equal, and the tie-break rule gives $\tau_{i_j}$ the higher priority.

    At time $t_0 + T - 1$, we have $\overline{RCT}_{i_j}(t_0 - 1) \geq \overline{RCT}_{i_j}(t_0 + T - 1)$, thus either $\tau_{i_j}$ is Urgent, and it is thus processed, either it is Contending. Since characteristic substrings are periodic (lemma 4), we have $\alpha(i_j, t_0 + T - 1) \prec \alpha(i_0, t_0 + T - 1)$ or $\alpha(i_j, t_0 + T - 1) = \alpha(i_0, t_0 + T - 1)$ and $\tau_{i_j}$ has priority over $\tau_{i_0}$ (because of the determinism of the tie-break rule). Thus $\tau_{i_j}$ has again priority over $\tau_{i_0}$ at time $t_0 + T - 1$. 

5
It follows that \( \tau_{t_0} \) cannot be processed at time \( t_0 + T - 1 \). This contradicts our assumption. Thus we have \( \forall t \geq r_i, RCT_i(t) \geq RCT_i(t + T) \). The algorithm PF is thus monotonous.

2 - We consider now the algorithm PD\(^2\). We recall that \( \tau_i^j \) denotes the \( i \)th unitary subtask of \( \tau_i \), \( r_i^j \) denotes its pseudo release and \( d_i^j \) its pseudo deadline. We consider a time \( t \) and a task \( \tau_i \). We denote \( j_i(t) \) the number of the next subtask of \( \tau_i \) to be processed. The subtask number \( j_i(t) - 1 \) has already been processed, but not the subtask number \( j_i(t) \). We thus have \( j_i(t) = W_i(0, t) + 1 \). We assume that \( t_0 \) and \( \tau_{t_0} \) are defined as previously described.

We have \( RCT_i(t_0) = RCT_i(t_0 + T) \).

It follows that \( W_i(0, t_0 - 1) + \frac{T}{T_{t_0}} \times C_{t_0} = W_i(0, t_0 + T - 1) \) and thus:

(i) \( j_i(t_0 + T - 1) = j_i(t_0 + T) \)

(ii) \( d_i(t_0 + T - 1) = d_i(t_0) + T \)

(iii) \( r_i(t_0 + T - 1) = r_i(t_0) + T \)

(iv) \( b_i(t_0 + T - 1) = b_i(t_0) \)

(v) The group deadline of \( \tau_i(t_0 + T - 1) \) is equal to the deadline of \( \tau_i(t_0 + T - 1) + T \):

\[
D_i = D_i(t_0 + T - 1) = D_i(t_0) + T
\]

Moreover, we have \( RCT_i(t_0) < RCT_i(t_0 + T) \) thus:

\[
\begin{cases}
\tau_{t_0}^{j_0(t_0 - 1)} & \text{is not processed at time } t_0 - 1. \\
\tau_{t_0}^{j_0(t_0 + T - 1)} & \text{is processed at time } t_0 + T - 1.
\end{cases}
\]

Now \( \tau_{t_0}^{j_0(t_0 - 1)} \) is not processed at time \( t_0 - 1 \) if:

\( t_0 - 1 < j_0^{j_0(t_0 - 1)} \).

But in this case, we would have \( t_0 + T - 1 < j_0^{j_0(t_0 - 1)} + T = j_0^{j_0(t_0 + T - 1)} \) and thus \( \tau_{t_0}^{j_0(t_0 + T - 1)} \) couldn’t have been processed at time \( t_0 + T - 1 \).

\( \text{o} \) there are \( m \) subtasks with higher priorities. Let \( \tau_{t_1}, \tau_{t_2}, \ldots, \tau_{t_m} \) be their parent tasks (\( \tau_i \) is the parent task of the subtasks \( \tau_i^j \)).

Following the definition of \( t_0 \), we have \( RCT_i(t_0 - 1) \geq RCT_i(t_0 + T - 1) \).

Therefore, \( j_i(t_0 + T - 1) \leq j_i(t_0 - 1) + \frac{T}{T_{t_0}} \times C_{t_0} \), and thus

(A) \( d_{t_0}^{j_i(t_0 + T - 1)} \leq d_{t_0}^{j_i(t_0 - 1)} + T \).

A subtask \( \tau_{t_0}^{j_i(t_0 - 1)} \) has priority over \( \tau_{t_0}^{j_0(t_0 - 1)} \) if:

- \( d_{t_0}^{j_i(t_0 - 1)} < d_{t_0}^{j_0(t_0 - 1)} \).

In this case, following (A) and (ii), \( d_{t_0}^{j_i(t_0 + T - 1)} \leq d_{t_0}^{j_i(t_0 - 1)} + T < d_{t_0}^{j_0(t_0 + T - 1)} + T = d_{t_0}^{j_0(t_0 + T - 1)} \). Thus \( \tau_{t_0}^{j_i(t_0 + T - 1)} \) has also priority over \( \tau_{t_0}^{j_0(t_0 + T - 1)} \).

- \( d_{t_0}^{j_i(t_0 - 1)} = d_{t_0}^{j_0(t_0 - 1)} \) and \( b_{t_0}^{j_i(t_0 - 1)} > b_{t_0}^{j_0(t_0 - 1)} \).
In this case,
• either \( j_{ik}(t_0 + T - 1) = j_{ik}(t_0 - 1) + \frac{T}{T_{ik}} \times C_{ik} \).

Then we have \( d_{ij}^{j_{ik}(t_0+T-1)} = d_{ij}^{j_{ik}(t_0)-1} \). We also have
\[ b_{ij}^{j_{ik}(t_0+T-1)} = b_{ij}^{j_{ik}(t_0)-1} > b_{ij}^{j_{ik}(t_0+T-1)} \].

Thus \( \tau_{ij}^{j_{ik}(t_0+T-1)} \) has priority over \( \tau_{ij}^{j_{ik}(t_0+T-1)} \).
• either \( j_{ik}(t_0 + T - 1) < j_{ik}(t_0 - 1) + \frac{T}{T_{ik}} \times C_{ik} \).

In this case, \( d_{ij}^{j_{ik}(t_0+T-1)} < d_{ij}^{j_{ik}(t_0+T-1)} \). Thus \( \tau_{ij}^{j_{ik}(t_0+T-1)} \) has priority over \( \tau_{ij}^{j_{ik}(t_0+T-1)} \).

\[ d_{ij}^{j_{ik}(t_0+T-1)} = d_{ij}^{j_{ik}(t_0)-1} \), \( b_{ij}^{j_{ik}(t_0+T-1)} = b_{ij}^{j_{ik}(t_0)-1} = 1 \) et \( D_{ij}^{j_{ik}(t_0+T-1)} > D_{ij}^{j_{ik}(t_0)-1} \).

Then,
• either \( j_{ik}(t_0 + T - 1) = j_{ik}(t_0 - 1) + \frac{T}{T_{ik}} \times C_{ik} \).

In this case, \( d_{ij}^{j_{ik}(t_0+T-1)} = d_{ij}^{j_{ik}(t_0+T-1)} \) and \( b_{ij}^{j_{ik}(t_0+T-1)} = b_{ij}^{j_{ik}(t_0+T-1)} = 1 \).

Besides we have \( D_{ij}^{j_{ik}(t_0+T-1)} = D_{ij}^{j_{ik}(t_0)-1} + T > D_{ij}^{j_{ik}(t_0)-1} + T = D_{ij}^{j_{ik}(t_0+T-1)} \).

It follows that \( \tau_{ij}^{j_{ik}(t_0+T-1)} \) has also priority over \( \tau_{ij}^{j_{ik}(t_0+T-1)} \).
• either \( j_{ik}(t_0 + T - 1) < j_{ik}(t_0 - 1) + \frac{T}{T_{ik}} \times C_{ik} \). In this case \( d_{ij}^{j_{ik}(t_0+T-1)} < d_{ij}^{j_{ik}(t_0+T-1)} \). Thus \( \tau_{ij}^{j_{ik}(t_0+T-1)} \) has priority over \( \tau_{ij}^{j_{ik}(t_0+T-1)} \).

\[ d_{ij}^{j_{ik}(t_0+T-1)} = d_{ij}^{j_{ik}(t_0)-1} \), \( b_{ij}^{j_{ik}(t_0+T-1)} = b_{ij}^{j_{ik}(t_0)-1} = 1 \) and \( D_{ij}^{j_{ik}(t_0+T-1)} = D_{ij}^{j_{ik}(t_0)-1} \).

The tie-break rule is then used. Then
• either \( j_{ik}(t_0 + T - 1) = j_{ik}(t_0 - 1) + \frac{T}{T_{ik}} \times C_{ik} \). In this case, \( d_{ij}^{j_{ik}(t_0+T-1)} = d_{ij}^{j_{ik}(t_0+T-1)} \) and \( b_{ij}^{j_{ik}(t_0+T-1)} = b_{ij}^{j_{ik}(t_0+T-1)} = 1 \) and \( D_{ij}^{j_{ik}(t_0+T-1)} = D_{ij}^{j_{ik}(t_0+T-1)} \). There is again a tie between the sub-tasks, and the deterministic rule gives again the higher priority to \( \tau_{ik} \).
• either \( j_{ik}(t_0 + T - 1) < j_{ik}(t_0 - 1) + \frac{T}{T_{ik}} \times C_{ik} \).

In this case, \( d_{ij}^{j_{ik}(t_0+T-1)} < d_{ij}^{j_{ik}(t_0+T-1)} \). Thus \( \tau_{ij}^{j_{ik}(t_0+T-1)} \) has priority over \( \tau_{ij}^{j_{ik}(t_0+T-1)} \).

Consequently, the \( m \) tasks \( \tau_{i1}, \tau_{i2}, \ldots, \tau_{im} \) have priority over \( \tau_{ik} \) at time \( t_0 + T - 1 \).

Thus, in any case, \( \tau_{ik} \) cannot be processed at time \( t_0+T-1 \), what contradicts our assumption. Thus, there exists no couple \( (t_0, \tau_{ik}) \) which verifies the properties (1) and (2). The monotony property is thus verified. □
Références
