# On Graham's Bound for Cyclic Scheduling

Philippe Chrétienne
Université Pierre et Marie Curie
Laboratoire LIP6

April 10, 1999

**Abstract**

This paper adresses the performance of list scheduling a cyclic set of $N$ non-preemptive dependent generic tasks on $m$ identical processors. The reduced precedence graph is assumed to be strongly connected but the number of simultaneously active instances of a generic task is not restricted to be at most one. Some properties on arbitrary schedules are first given. Then we restrict to regular schedules for which it is shown that the number of ready or active tasks at any instant is at least the minimum height $H^*$ of a directed circuit of the reduced precedence graph. The average cycle time of any regular list schedule is then shown to be at most $(2-\frac{\min\{H^*,m\}}{m})$ times the absolute minimum average cycle time. This result, which is similar well-known $(2-\frac{1}{m})$ Graham's bound applying for non cyclic scheduling, shows to what extent regular list schedules take the parallelism of the cyclic task system into account.

## 1 Introduction

Cyclic scheduling addresses the problem of scheduling a cyclic set of interdependent tasks that may for example model the body of a program loop or the tasks involved in the mass production of an equipment.
Cyclic scheduling is not less difficult than non-cyclic scheduling since any non-cyclic scheduling problem polynomially reduces to a cyclic problem where successive iterations do not overlap. Most of the research effort in cyclic scheduling has concerned the basic cyclic scheduling problem [7], [8], [10], dominant subsets of periodic schedules for the cyclic scheduling problem with $m$ identical processors [3], complexity and efficient algorithms for

special cases [4],[6]. Unlike for non-cyclic scheduling where a lot of research has been devoted to approximation [1], [2], searching for the performance ratio of approximation algorithms has been relatively rare for cyclic scheduling problems. In [5], non-cyclic list scheduling and the famous Graham's bound have been combined to derive a strictly periodic schedule whose average cycle time is at most $2 - (1/m)\lambda_{opt} + (m - 1/m)(p_{max} - 1)$ where $\lambda_{opt}$ is the maximum time-to-height ratio of a circuit in the reduced precedence graph. In [9], list schedules have been defined for cyclic scheduling problems with non-reentrant generic tasks and have been shown to provide the performance ratio $(2 - \frac{1}{m})$. In [13], new priority lists have been defined and tested for more general cyclic-task systems modelled by timed Petri nets.
This paper concerns the performance of list schedules for the general cyclic scheduling problem on $m$ identical machines ($GCSP$ in abbreviated form). In Section 2, the general cyclic scheduling problem on $m$ identical processors is specified. In Section 3, a dominance property in the set of schedules as well as some properties of arbitrary schedules are given; in particular the number of ready or active tasks at any instant is shown to be at least the minimum height $H^*$ of a directed circuit of the reduced precedence graph. In Section 4, the performance ratio of an arbitrary regular list schedule is shown to be $(2 - \frac{\min\{H^*, m\}}{m})$. This result, which may be seen as the analog of the Graham's bound for non-cyclic list scheduling, shows that $\min\{H^*, m\})$ is a good measure of the parallelism of the cyclic task system. The last section is devoted to some conclusions.

## 2 The cyclic scheduling problem $GCSP$

A scheduling problem is said to be *cyclic* if its infinite task graph has a *periodic* structure. In the case of $GCSP$, this structure is as follows:

### The tasks

The task set $\mathcal{T}$ is partitioned into an infinite number of *iterations* where each iteration is an instance of a finite set $T = \{T_1, T_2, \cdots, T_N\}$ of so-called *generic* tasks. Each iteration is indexed by a natural number $n \geq 1$ and the tasks of the iteration $n$ are denoted by $T_j^n, T_j \in T$. The task $T_j^n$ is called the *instance* $n$ of $T_j$. Tasks are *not preemptive* and all the instances $T_j^n, n \geq 1$ of the same generic task $T_j$ have the same *positive integer duration* $p_j$. The maximum duration of a generic task is denoted by $p_{max}$.

**The precedence constraints**

The precedence constraints are defined from a finite set $U = \{u_1, \cdots, u_P\}$ of so-called *generic uniform* precedence constraints. Each $u_k$ is a triple $(T_i, T_j, h)$ where $T_i = u_k^-$ and $T_j = u_k^+$ are two generic tasks and where $h$ is a natural number called the *height* of $u_k$. The maximum height of a generic precedence contraint is denoted by $h_{max}$. If $u_k = (T_i, T_j, h)$ is a generic precedence constraint then for each iteration $n \geq 1$, the task $T_i^n$ must be completed before the task $T_j^{n+h}$ starts its execution.
The precedence graph $G$ is an infinite directed acyclic graph with a periodic structure (see Figure 1). The set of the immediate predecessors (resp. successors) in $G$ of the task $T_i^p$ is denoted by $IN(T_i^p)$ (resp. $OUT(T_i^p)$).
The directed graph $\widehat{G} = (T, U)$ whose nodes are the generic tasks and whose arcs correspond to the generic precedence constraints is called the *reduced precedence graph* (see Figure 2).
The reduced precedence graph of an instance of $GCSP$ is assumed to be *consistent* (i.e: every simple circuit has a strictly positive height) and *strongly connected.* Consistency is needed for the set of schedules to be non empty while strong connectivity provides essential stability properties to schedules.
Let $H^*$ be the minimum height of a simple circuit of $\widehat{G}$. Since the height of an arc is a non-negative integer, then any circuit of $\widehat{G}$ has a non-negative height and $H^*$ may be computed in polynomial time using any "all shortest paths" algorithm that defines the cost of an arc to be its height. As a consequence, the consistency property may be decided in polynomial time.

**The resource constraints**

$m$ identical processors $\{P_1, \cdots, P_m\}$ are available to execute the tasks. As usual, the execution of each task $T_j^n, T_j \in T, n \geq 1$ requires one processor and, at any instant, one processor may execute at most one task.

**Schedule, average cycle time and optimization**

An instance $I = (G, h, p, m)$ of $GCSP$ is thus specified by a strongly-connected graph $\widehat{G} = (T, U)$, non-negative integral arc heights $h(u), u \in U$, positive integral processing times $p_i, T_i \in T$ and the number $m$ of processors. A *schedule* $S = (s, \pi)$ of $I$ assigns each task $T_j^k, T_j \in T, k \geq 1$ a starting time $s(i, k)$ and a processor $\pi(i, k)$ such that all the resource and precedence constraints are satisfied. The completion time $C_n(S)$ of iteration $n$ is equal to $\max\{s(j, n) + p_j \mid T_j \in T\}$ and the *average cycle time* $\omega(S)$ of $S$ is defined

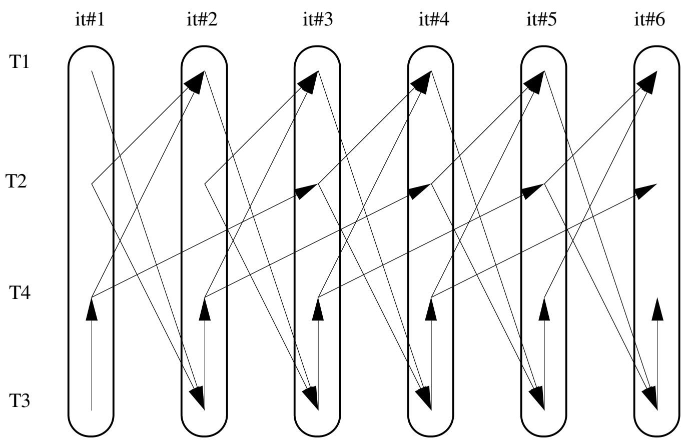



Figure 1: The precedence graph $G$.

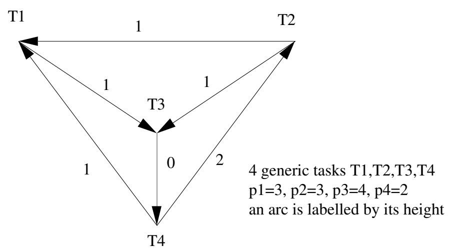



Figure 2: The reduced precedence graph $\widehat{G}$.

by $\limsup_{n\to\infty}(C_n(S)/n)$. The *absolute minimum average cycle time* $\alpha(I)$ is the greatest lower bound of the values $\omega(S)$ over the set of schedules of $I$. The scheduling problem is to determine a schedule whose average cycle time is as small as possible.
Let $K$ be a positive integer and let $r$ be a positive rational number. A schedule $S$ is said to be $K$-periodic with period $r$ if there exists a positive integer $N_0$ such that for any generic task $T_i$, the sequence $\{s(i,n) \mid n \geq 0\}$ satisfies: $\forall n \geq N_0$: $s(i, n+K) = s(i,n) + r$. Note that in this case, we have $\omega(S) = \lim_{n\to\infty} C_n(S)/n = r/K$. $K$ is called the *periodicity factor* of $S$ whereas $N_0$ is the length of the *transient phase* of $S$.

# 3 Schedule properties

## 3.1 Arbitrary schedules

We introduce in this section some general definitions and properties that refer to an arbitrary schedule $S = (s, \pi)$ of an instance $I$ of $GCSP$.
The number of instances of $T_i$ started in the time interval $[0,t]$ (respectively $[0,t[$) is denoted by $D_i^+(t)$ (respectively $D_i^-(t)$). The number of instances of $T_i$ completed in the time interval $[0,t]$ is denoted by $F_i(t)$. The task $T_i^k$ is said to be *active* at time $t$ in $S$ if $s(i,k) < t < s(i,k) + p_i$. The number $A_i(t)$ of instances of $T_i$, which are active at time $t$ in $S$ is thus equal to $D_i^-(t) - F_i(t)$. These definitions are illustrated for the schedule shown in Figure 3. The following lemma shows that every schedule satisfies the so-called *balance property*.

**Lemma 1** *Let $H_1$ be the maximum height of any simple path of $\widehat{G}$. For any two generic tasks $T_i$ and $T_j$ and for any time $t \geq 0$, $|D_j^-(t) - D_i^-(t)| \leq H_1$.*

*Proof.* — Let $T_i$ and $T_j$ be two generic tasks. Since $\widehat{G}$ is strongly connected, there is a simple path $\mu$ in $\widehat{G}$ from $T_i$ to $T_j$. Let $h(\mu)$ be the height of $\mu$. For every $k > h(\mu)$, any start in $[0,t[$ of a task $T_j^k$ is preceded by the start in $[0,t[$ of the task $T_i^{k-h(\mu)}$, so we have $D_j^-(t) \leq h(\mu) + D_i^-(t)$. We thus conclude that $|D_j^-(t) - D_i^-(t)| \leq H_1$ ∎

Let $u_k = (T_i, T_j, h)$ be an arc of $\widehat{G}$. The *pre-marking* $M_k^-(t)$ (respectively *post-marking* $M_k^+(t)$) of $u_k$ at time $t$ is defined as $h + F_i(t) - D_j^-(t)$ (respectively $h + F_i(t) - D_j^+(t)$).

**Lemma 2** *For any time $t \geq 0$ and any arc $u_k$, we have $M_k^+(t) \geq 0$.*

*Proof.* — Let $u_k = (T_i, T_j, h)$. Let $N_1(j)$ (respectively $N_2(j)$) be the number of tasks $T_j^k$ with $k > h$ (respectively $k \leq h$) started in $[0, t]$. Since for any $k > h$, a start in $[0, t]$ of a task $T_j^k$ with $k > h$ is preceded by the completion in $[0, t]$ of the task $T_i^{k-h}$, we have $F_i(t) \geq N_1(j)$. It is straightforward from the definitions that: $N_1(j) + N_2(j) = D_j^+(t)$ and $N_2(j) \leq h$. We thus conclude that $D_j^+(t) \leq h + F_i(t)$ or equivalently that $M_k^+(t) \geq 0$. ∎

Even if $S = (s, \pi)$ is such that the resource constraint is satisfied and the post-marking of every arc remains positive, then $S$ may not be a schedule. Consider for example a single generic task $T_1$ with $p_1 = 1$, a single generic precedence constraint $(T_1, T_1, 1)$ and only one machine. The assignment $s(1,2) = 0$,$s(1,1) = 1$,and $s(1,k) = k - 1$ for $k \geq 3$ is such that for any $t \geq 0$, $M_1^+(t) \geq 0$ but is not a schedule.

**Lemma 3** *Let $\rho$ be an arbitrary simple circuit of $\widehat{G}$. At any time $t \geq 0$, we have $\sum_{u_k \in \rho} M_k^-(t) + \sum_{T_i \in \rho} A_i(t) = h(\rho)$.*

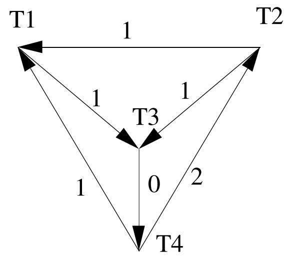
*Proof.* — Let $u_k = (T_i, T_j, h)$ be an arc of $\rho$. Since $A_i(t) = D_i^-(t) - F_i(t)$, we have $M_k^-(t) = h + F_i(t) - D_j^-(t)$=$h + (D_i^-(t) - D_j^-(t)) - A_i(t)$. Summing over the arcs of $\rho$, we get: $\sum_{u_k \in \rho} M_k^-(t) + \sum_{T_i \in \rho} A_i(t) = h(\rho)$. ∎

## 3.2 Regular schedules

$S = (s, \pi)$ is said to be *regular* if for every generic task $T_i$, the time sequence $s(i, k)$ satisfies: $\forall k \geq 1, s(i, k+1) \geq s(i, k)$. The next lemma shows that if $S = (s, \pi)$ is regular and meets the resource constraint, then the non-negativity of the post-marking ensures that $S$ is a schedule.

**Lemma 4** *If $S = (s, \pi)$ is a time and processor assignment such that a) $S$ is regular, b) for any time $t \geq 0$ and any arc $u_k \in P$: $M_k^+(t) \geq 0$ and c) the resource constraint is satified, then $S$ is a schedule.*

*Proof.* — Consider an arc $u_k = (T_i, T_j, h)$. From b), we know that for any $t \geq 0$, $M_k^+(t) = h + F_i(t) - D_j^+(t) \geq 0$. Let us assume that for $k > h$, $s(j, k) = t$. Since $S$ is regular, we have $D_j^+(t) \geq k$ and we get that




| i | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $D_i^-$ | 1 | 2 | 2 | 1 |
| $D_i^+$ | 2 | 3 | 2 | 1 |
| $F_i$ | 1 | 2 | 1 | 1 |

Instantaneous values at time 6

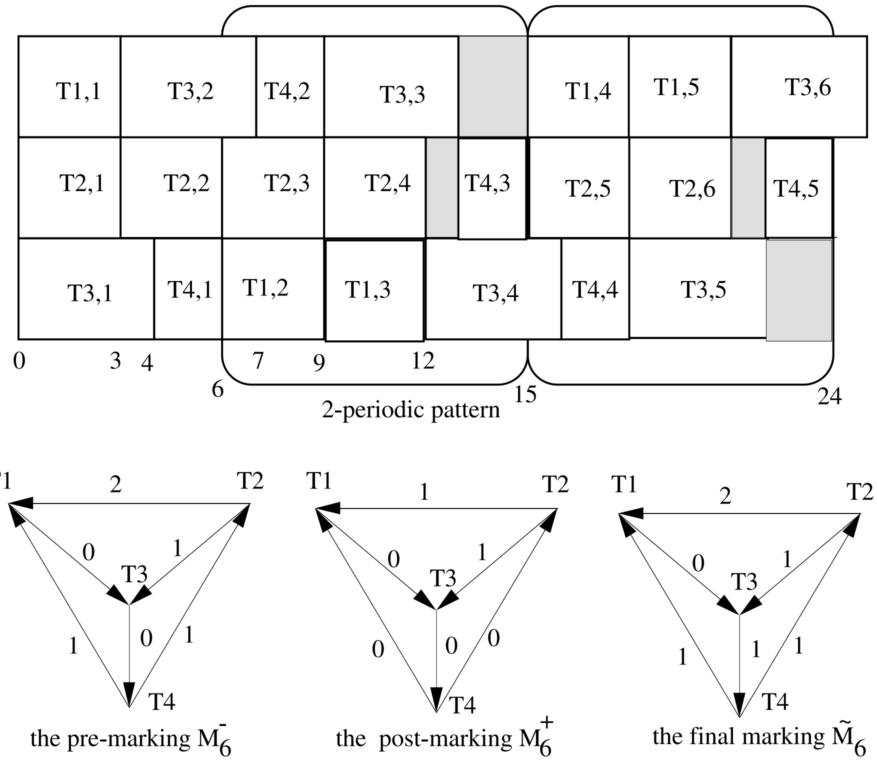



Figure 3: Definitions associated with a schedule

$F_i(t) \geq k-h$. Again from the regularity of $S$ we get that $s(i, k+h)+p_i \leq t$. We thus conclude that the generic precedence $u_k$ is satisfied by $S$. So $S$ is a schedule since it also meets the resource constraint (assumption c) of the lemma). ∎

It is now easy to derive from Lemmas 2 and 4 that the regular schedules make a dominant subset.

**Theorem 1** *For any instance of GCSP, there is an optimal schedule which is regular.*

*Proof.* — Let $S=(s,\pi)$ be a schedule of an instance $I$. The task subset $\{T_i^n \mid n \geq 0\}$ may be totally ordered by $\prec_S$ with respect to $S=(s,\pi)$ where $T_i^p \prec_S T_i^q$ if $s(i,p) < s(i,q)$ or $((s(i,p) = s(i,q))$ and $(\pi(i,p) < \pi(i,q)))$. For each generic task $T_i$, let us denote by $T_i^{i_k}$ the instance of $T_i$ whose rank is $k$ with respect to $\prec_S$. Consider now the time and processor assignment $S'=(s',\pi')$ we get by replacing in the processor-time diagram of $S$ each task $T_j^{j_k}$ by the task $T_j^k$, i.e: for each task $T_i^k$: $s'(i,k) = s(i,i_k), \pi'(i,k) = \pi(i,i_k)$. Clearly $S'$ is regular and satisfies the processor constraint. Moreover from the definition of $S'$, the marking functions $M^+$ and $M'^+$ associated respectively with $S$ and $S'$ are identical. Since $S$ is a schedule, we know from Lemma 2 that for any $u_k \in P$ and for any $t \geq 0$, $M_k^+(t) \geq 0$. We thus have that for any $u_k \in P$ and for any $t \geq 0$, ${M'}_k^+(t) \geq 0$. Finally we conclude from Lemma 4 that $S'$ is a regular schedule of $I$ such that $\omega(S) = \omega(S')$. ∎

At any time $t \geq 0$, we denote by $E(t)$ the subset of the tasks whose execution has started in $[0,t[$ and by $G(t)$ the subgraph of $G$ induced by the tasks of $E(t)$. The restriction of $S$ to the tasks of $E(t)$ is denoted by $S(t)$. $S(t)$ is clearly a schedule of $G(t)$. The *final marking* $\tilde{M}(t)$ of $S(t)$ is such that for any arc $u_k = (T_i, T_j, h)$ of $\hat{G}$: $\tilde{M}_k(t) = M_k^-(t) + A_i(t)$. Figure 3 shows the final marking at time 6 for the corresponding schedule. The task $T_i^k$ *is said to be ready at time $t$ in $S$* if $T_i^k \notin E(t)$ and if each task in $IN(T_i^k)$ is completed by time $t$ in $S$. The number of instances of $T_j$ that are *ready* at time $t$ in $S$ is denoted by $R_j(t)$. The following lemma characterizes the ready tasks.

**Lemma 5** *If* $\min\{M_k^-(t) \mid u_k^+ = T_j\} = r$ *and* $D_j^-(t) = q$, *then the instances of $T_j$ ready at time $t$ in $S$ are the tasks* ${T_j}^{q+1}, \cdots, {T_j}^{q+r}$.

*Proof.* — Let $u_k = (T_i, T_j, h)$. If $r + q < h$, then there is no instance of $T_i$ in $IN(T_j^{r+q})$. Otherwise, from the definitions of $r$ and $q$, we have $M_k^-(t) = h + F_i(t) - q \geq r$, from which we get that $F_i(t) \geq r + q - h$. In either case the task $T_i^{r+q-h}$ is completed by time $t$ in $S$. So $T_j^{r+q}$ is ready at time $t$ in $S$ and the same is true for the tasks $T_j^{1+q}, \cdots, T_j^{r-1+q}$. We thus have $R_j(t) \geq r$. Moreover from the definition of $r$, we know there is $u_{k_0} = (T_{i_0}, T_j, h_0)$ such that $M_{k_0}^- = r$, from which we get that $F_{i_0}(t) = r + q - h_0$. Thus $T_{i_0}^{r+q-h_0+1}$ is not completed by time $t$ in $S$. So the task $T_j^{r+q+1}$ is not ready at time $t$ in $S$ and the same is true for the tasks $T_j^{r+q+k}$ with $k > 1$. We thus have $R_j(t) = r$. ■

The above characterization of the ready tasks leads us to derive a lower bound on the number $RA(t)$ of the ready or active tasks at time $t$ in $S$. This will be the key property for the performance analysis of the regular list schedules.

**Theorem 2** *Let $H^*$ be the minimal height of a simple circuit of $\hat{G}$. For any schedule $S$ and any time $t \geq 0$, we have $RA(t) \geq H^*$.*

*Proof.* — Let $t \geq 0$. With each generic task $T_j$, we associate an arc $u_{k(j)}$ such that $u_{k(j)}^+ = T_j$ and $\tilde{M}_{k(j)}(t) = \min\{\tilde{M}_k(t) \mid u_k^+ = T_j\}$. The subgraph of $\hat{G}$ induced by the arcs $u_{k(j)}, T_j \in T$ has at least one simple circuit $\rho$. From Lemma 5 and since $A_j(t) = D_j^-(t) - F_j(t)$, we have

$$R_j(t) + A_j(t) = M_{k(j)}^-(t) + A_j(t) = h_{k(j)} + F_{u_{k(j)}^-}(t) - F_j(t)$$

Let us now consider the number of ready or active instances of the generic tasks of $\rho$. From the definition of the arcs $u_{k(i)}$, we have:

$$\sum_{T_j \in \rho} (R_j(t) + A_j(t)) = \sum_{(T_i, T_j, h) \in \rho} (h + F_i(t) - F_j(t)) = h(\rho)$$

We thus conclude that

$$RA(t) = \sum_{T_j \in T} (R_j(t) + A_j(t)) \geq \sum_{T_j \in \rho} (R_j(t) + A_j(t)) \geq H^*$$

■

We now show that $H^*$ is a best lower bound for $RA(t)$ since there is a schedule $S$ and a time $t$ such that $RA(t) = H^*$.
Without loss of generality, let us assume that $C = \{T_{N-c+1}, \cdots, T_N\}$ are the generic tasks of a simple circuit of $\hat{G}$ whose height is $H^*$. A task subset $\tau \subset \mathcal{T}$ is said to be $\bar{C}$*-initial* (where $\bar{C} = \{T_1, \cdots, T_{N-c}\}$) if:

1. $T_j^q \in \tau \Rightarrow T_j \in \bar{C}$;
2. $(T_j^q \in \tau$ and $T_i^p \in IN(T_j^q)) \Rightarrow T_i^p \in \tau$.

A subset $\tau \subset \mathcal{T}$ is *regular* if $\tau = \cup_{i=1}^{N} \{T_i^1, \cdots, T_i^{k_i}\}$ where by convention $\{T_i^1, \cdots, T_i^{k_i}\} = \emptyset$ if $k_i = 0$. A regular $\bar{C}$-initial subset may thus be denoted by $\tau(k_1, \cdots, k_i, \cdots, k_{N-c})$. A regular $\bar{C}$-initial subset $\tau$ is *locally maximal* if for any $i \in \{1, \cdots, N-c\}$, $\tau(k_1, \cdots, k_i + 1, \cdots, k_{N-c})$ is not a $\bar{C}$-initial subset. The next lemma shows that there is a locally maximal $\bar{C}$-initial subset.

**Lemma 6** *If $C = (T_{N-c}, T_{N-c+1}, \cdots, T_N, T_{N-c})$ is a simple circuit of the reduced precedence graph, there is a locally maximal $\bar{C}$-initial subset.*

*Proof.* — Since $\hat{G}$ is strongly connected, for every $T_j, j \in \{1, \cdots, N-c\}$, there is a path from $T_N$ to $T_j$ in $\hat{G}$. So for every $T_j, j \in \{1, \cdots, N-c\}$, there is a positive integer $h_j$ such that $T_j^{h_j}$ does not belong to any $\bar{C}$-initial subset. Since the empty set is a regular $\bar{C}$-initial subset, there is a locally maximum $\bar{C}$-initial subset. ∎

**Theorem 3** *There is a schedule and a time $t$ such that $RA(t) = H^*$.*

*Proof.* — Let $\hat{K}$ be a regular $\bar{C}$-initial subset and let $t = \sum_{T_j^q \in \hat{K}} p_j$. Consider a regular schedule $S$, which first executes on the same processor the tasks of $\hat{K}$. Since $\hat{K}$ is locally maximal, any task $T_j^q$ with $T_j \in \bar{C}$ is neither ready for $S$ at time $t$ nor active at time $t$ in $S$. So the only tasks that might be ready or active at time $t$ in $S$ are the tasks $T_j^q$ with $T_j \in C$. Let $u_{k(j)}$ be the arc in $C$ whose output node is $T_j$. From Lemma 3, we know that $\sum_{T_j \in C} (M_{k(j)}^-(t) + A_j(t)) = H^*$. Since from Lemma 5, we have $R_j(t) \leq M_{k(j)}^-(t)$ for any $T_j \in C$, we get that $RA(t) \leq H^*$ and from Theorem 2 that $RA(t) = H^*$. ∎

Let us illustrate this result on the instance of Figure 2. Here we have $H^* = 2$ and $C = (T_1, T_3, T_4, T_1)$ is a circuit with height 2. $\{T_2^1, T_2^2\}$ is a locally maximal $\bar{C}$-initial subset . Once these two tasks are executed, the only ready tasks are $T_1^1$ and $T_3^1$.

# 4 The performance of regular list schedules

## 4.1 List schedules and regular list schedules

A schedule $S$ is a *list schedule* if for any idling interval $[t, t+\epsilon[$ $(\epsilon > 0)$ of a processor , any task scheduled at time $u > t$ has at least one of its predecessors being processed at time $t$ in $S$.
A *list algorithm* is such that at any time when at least a new task may be performed (decision time), as many ready tasks as possible are assigned to the free processors.
A list algorithm is said to be regular if , at each decision time $\theta$, the ready instances of a generic task are assigned to free processors according to increasing iteration numbers. More precisely, if $T_i^p$ is assigned a free processor at decision time $\theta$, then so is every task $T_i^q$ with $q < p$ that is ready at time $\theta$.
The following property shows that regular list algorithms generate regular list schedules.

**Property 1** *The schedule provided by a regular list algorithm is a regular list schedule. Conversely, a regular list schedule is the schedule provided by a regular list algorithm.*

*Proof.* — Let $S = \mathcal{A}(I)$ be the schedule provided by the regular list algorithm $\mathcal{A}$ on the instance $I$ of $GCSP$ and let $\theta_n, n \geq 0$ be the sequence of the decision times of $S$. We show by induction on $n$ that the partial schedule $S^+(\theta_n)$ associated with the tasks started in $[0, \theta_n]$ is regular. This holds for $n = 0$ since a) $\mathcal{A}$ is regular and b) if $T_j^q$ is ready at time $\theta_0 = 0$, then so is every task $T_j^p$ with $p \leq q$. Assume now that $S^+(\theta_{n-1})$ is regular and let $last(i)$ be the instance of $T_i$ in $S^+(\theta_{n-1})$ with the highest iteration index. Again since $S^+(\theta_{n-1})$ is regular, then if $T_i^p$ is ready at time $\theta_n$ in $S$, then so is every instance $T_i^r$ with $last(i) < r \leq p$. Now since $\mathcal{A}$ is regular, we get that $S^+(\theta_n)$ is a regular list schedule.
Conversely, if $S$ is a regular list schedule of the instance $I$ of $GCSP$, let us denote by $L(S)$ the (infinite) list of the tasks ordered by increasing starting

times using increasing processor numbers for tie-breaking. Since $S$ is regular, it is clear that the list algorithm algorithm associated with $L(S)$ is a regular list algorithm that provides the the schedule $S$. ■

The periodic schedule shown on Figure 3 is a list schedule of the instance of the GCSP instance shown in Figure 2.

### 4.2 The performance ratio of regular list schedules

Let $I = (G, h, p, m)$ be an arbitrary instance of $GCSP$. We show in this section that the average cycle time of every regular list schedule $S$ of $I$ is at most $(2 - \frac{K^*}{m})\alpha(I)$ where $\alpha(I)$ is the absolute minimum average cycle time of $I$ and $K^* = \min\{H^*, m\}$. We first recall the following characterization of $\alpha(I)$ that has has been proved in [9] where

- $G_n$ is the subgraph of $G$ induced by the $n$ first instances of each generic task, i.e: by $\{T_i^p \mid i \in \{1, \cdots, N\}, p \in \{1, \cdots, n\}\}$
- $O_n$ is an optimal schedule of $G_n$.

**Lemma 7** *Let $M(O_n)$ be the makespan of an optimal schedule of $G_n$. The absolute minimum average cycle time of $I$ is* $\limsup_{n\to\infty} \frac{M(O_n)}{n}$

**Theorem 4** *Let $I = (G, h, p, m)$ be an arbitrary instance of $GCSP$. Every regular list schedule $S$ of $I$ satisfies $\omega(S) \leq (2 - \frac{K^*}{m})\alpha(I)$*

*Proof.* — Let $\mathcal{A}$ be a regular list algorithm and let $S = \mathcal{A}(I)$ be the associated regular list schedule. We consider the time window $W = [0, C_n(S)]$ of the Gantt time diagram of $S$. $W$ may be partitionned into *total activity* periods where all the processors are busy and *partial activity* periods where at least one processor is idle (see Figure 4).

Let $[a, b]$ be one partial activity period. We denote by $d_0, \cdots, d_r$ the decision times of $S$ in $[a, b]$. Note that we have $d_0 = a$, $d_0 < \cdots < d_r$ and $d_r = b$. Since $\mathcal{A}$ is a list algorithm, for every decision time $d_i$, the number $m_i$ of busy processors in $[d_i, d_{i+1}]$ is $\min\{m, a_i + r_i\}$ where $a_i$ (respectively $r_i$) is the number of ready (respectively active) tasks at time $d_i$ in $S$. From Theorem 2, we have $a_i + r_i \geq H^*$. Since $[d_i, d_{i+1}]$ belongs to a partial activity period and $\mathcal{A}$ is a list algorithm, we get $a_i + r_i < m$. We thus have $m_i = a_i + r_i$, from which we conclude that $m_i \geq \min\{m, H^*\} = K^*$.
Starting with a task $T_i^n$ of $G_n$ that completes in $S$ at time $C_n(S)$ and

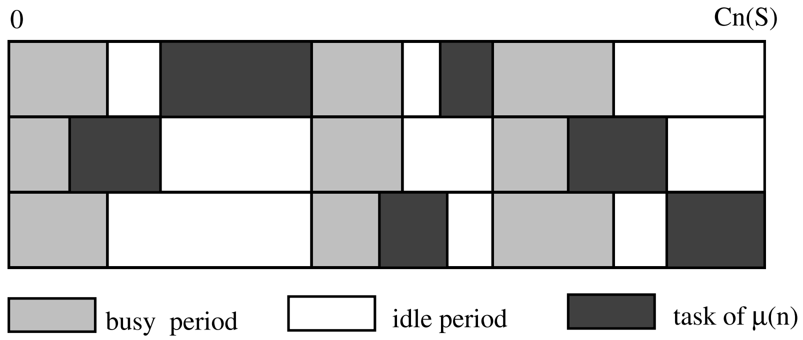



Figure 4: Structure of the time window $W$

following the lines of the well-known Graham's proof in [11], we know that there is a path $\mu_n$ of tasks in $G_n$ whose last task is $T_i^n$ and whose successive execution time intervals in $S$ totally overlaps the partial activity periods of $W$. So, if $L_n$ is the sum of the durations of the tasks on $\mu_n$, we have $C_n(S) \geq L_n$ from which we get:

$$\omega(S) \geq \limsup_{n\to\infty} \frac{L_n}{n}$$

Let us now denote by $R_n(S)$ the sum of the durations of the tasks $T_j^k$ started in the time interval $[0, C_n(S)[$ and such that $k > n$. Let $Q^*$ be the maximum height of a simple circuit of $\hat{G}$. We know from Lemma 3 that at most $Q^*$ instances of the generic task $T_i$ complete at time $C_n(S)$ and from the regularity of $S$, we derive that $D_i^-(C_n(S)) \leq n + Q^*$. From Lemma 1, we know that for every generic task $T_j$, $(j \neq i)$, we have $k \leq n + H_1 + Q^*$. We thus conclude that:

$$R_n(S) \leq (H_1 + Q^*) \sum_{T_i \in T} p_i$$

As usual, we finally consider the time window $W$ of the Gantt-diagram of $S$. Since the cumulative sum of the idle processor periods is at most $(m - K^*)L_n$ and the cumulative sum of the busy processor periods is at most $n \sum_{T_i \in T} p_i + R_n(S)$, we get that:

$$mC_n(S) \leq n \sum_{T_i \in T} p_i + R_n(S) + (m - K^*)L_n$$

from which we get:

$$\frac{C_n(S)}{n} \leq \frac{\sum_{T_i \in T} p_i}{m} + \frac{1}{m}\frac{R_n(S)}{n} + \frac{(m-K^*)}{m}\frac{L_n}{n}$$

Now since $L_n \leq M(O_n)$ and $\frac{\sum_{T_i \in T} p_i}{m} \leq \alpha(I)$, we derive from Lemma 7 that:

$$\omega(S) = \limsup_{n \to \infty} \frac{C_n(S)}{n} \leq (2 - \frac{K^*}{m})\alpha(I)$$

∎

# 5 Conclusion

In this paper, we have studied the performance of list-scheduling a cyclic set of non-preemptive, interdependent and reentrant generic tasks. Regular schedules, where the instances of each generic task must be scheduled in the order of increasing iteration numbers, have been shown to make a dominating set of schedules. Then, from the property that in any regular schedule, the number of active or ready tasks is at least $\min\{m, H^*\}$ (where $H^*$ is the the minimum height of a simple circuit in the reduced precedence graph) the average cycle time of any regular list schedule has been shown to be at most $(2 - \frac{\min\{m,H^*\}}{m})$ times the minimum absolute average cycle time. This latter result, which extends the ratio $2 - \frac{1}{m}$ previously shown to apply to the special case of non-reentrant tasks may be considered as the analog for GCSP of the well-known Graham's bound. It also brings a quantitative insight to the rather intuitive idea that a list scheduling algorithm should behave better on a cyclic set of tasks than on a finite number of its iterations. Another interesting aspect is that if $H^* \geq m$, any regular list schedule is optimal. So $2 - \frac{\min\{m,H^*\}}{m}$ is a best bound and $H^*$ is a good measure of the parallellism of the infinite task graph.

## Acknowledgements

I thank the referees for their helpful remarks and comments.

## References

[1] J.K. Lenstra and D.B. Shmoys (1995). Computing near optimal schedules, Scheduling theory and its applications, chap. 1, 193-224, Eds P. Chrétienne, E.G. Coffman, Jr, J.K. Lenstra and Z. Liu, Wiley.

[2] E.G. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan and D.B. Shmoys (1994). Sequencing and Scheduling: algorithms and complexity, Report BS-R8909, Center for Mathematics and Computer Science, Amsterdam.

[3] C. Hanen and A.Munier (1995). A study of the cyclic scheduling problem on parallel processors, *Disc. Appl. Math.*, 57, 167-192.

[4] C. Hanen and A. Munier (1995). Cyclic scheduling on parallel processors, Scheduling theory and its applications, chap. 9, 193-224, Eds: P. Chrétienne, E.G. Coffman, Jr, J.K. Lenstra and Z. Liu, Wiley.

[5] F. Gasperoni and U. Schwiegelshohn (1994). Generating close to optimal loop schedules on parallel processors, *Par. Proc. Let.*, 4(4), 391-403.

[6] A. Munier (1991). Contribution à l'étude des ordonnancements cycliques, PhD thesis, P. and M. Curie University.

[7] P. Chrétienne (1985). Transient and limiting behavior of timed event graphs, *RAIRO-TSI*, 4, 127-142.

[8] P. Chrétienne (1991). The basic cyclic scheduling problem with deadlines, *Disc. Appl. Math.*, 30, 109-123.

[9] P. Chrétienne (1997). List schedules for cyclic scheduling, to appear in *Disc. Appl. Math.*.

[10] G. Cohen, D. Dubois, J.P. Quadrat and M. Viot (1985). A linear system theoretic view of discrete event process and its use for performance evaluation in manufacturing, *IEEE Trans. Aut. Cont.*, 30, 3.

[11] R.L. Graham (1969). Bounds on multiprocessing timing anomalies, *SIAM J. Appl. Math.*, 17, 416-429.

[12] E.G. Coffman,Jr and R.L. Graham (1972). Optimal scheduling for two processors systems, *Act. Inf.*, 13, 200-213.

[13] T. Watanabe and M.Yamauchi (1993). New priority lists for scheduling in timed Petri nets. ICATPN93, LNCS 691, Chicago.