



ELSEVIER

Discrete Applied Mathematics 94 (1999) 141–159

---

---

**DISCRETE  
APPLIED  
MATHEMATICS**

---

---

## List schedules for cyclic scheduling

Philippe Chrétienne \*

Laboratoire LIP6, Université Pierre et Marie Curie, 4, place Jussieu, 75252 Paris,  
Cedex 05, France

Received 4 November 1996; revised 2 July 1997; accepted 8 May 1998

---

### Abstract

This paper addresses the definition and properties of list scheduling in the context of scheduling a cyclic set of  $n$  non-preemptive and non-reentrant-dependent tasks on  $m$  identical processors when the reduced precedence graph is assumed to be strongly connected. It is first shown that the average cycle time of an arbitrary list schedule is at most  $(2 - 1/m)$  times the absolute minimum average cycle time.  $K$ -periodic list schedules are then shown to be the list schedules associated with special priority mappings called  $K$ -periodic linear orders. Moreover, given a list that offers the performance ratio  $\rho$  in the non-cyclic case and whose structure is quasi  $K$ -periodic in the cyclic case, it is shown that each of its corresponding  $K$ -periodic linear orders provides a list schedule with the same guarantee. Since the well-known Coffman Graham's list is shown to be quasi  $K$ -periodic, its performance may be transferred to UET cyclic problems. © 1999 Published by Elsevier Science B.V. All rights reserved.

---

### 1. Introduction

It is well known that, due to the computational complexity of most non-cyclic scheduling problems, a lot of research has been devoted to approximation algorithms and their performance ratios. In fact, two problems are particularly challenging in this area: *searching a good lower bound* on the performance ratio yields to study the complexity of decision problems such as “Is there a schedule with makespan less than  $B$ ”? (where  $B$  is a fixed positive integer) whereas *searching for a good upper bound* leads to design heuristics providing good solutions. List algorithms, which select the ready tasks to be assigned processors from a static priority list, are certainly the most studied algorithms in this field [6,9,10].

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 must not overlap. However, approximation algorithms with known performance ratios are relatively rare for cyclic scheduling problems [7,8]. To our knowledge, the

---

\* E-mail address: [philippe.chretienne@lip6.fr](mailto:philippe.chretienne@lip6.fr). (P. Chrétienne)

most notable exception is [5] where it has been shown how non-cyclic list scheduling and the famous Graham's bound could be combined to derive a periodic schedule whose average cycle time is at most  $(2 - 1/m)\lambda_{\text{opt}} + ((m - 1)/m)(p_{\text{max}} - 1)$  ( $\lambda_{\text{opt}}$  being the minimum average cycle time for the corresponding unlimited-processors case).

The purpose of this paper is to extend list scheduling to cyclic scheduling problems and to bring some results about the performance of list schedules in the cyclic case. In Section 2, the cyclic scheduling problem *CSP* is specified. Section 3 gives some basic properties of the schedules of any instance of *CSP* and list scheduling is extended to *CSP* in Section 4. It is proved in Section 5 that the performance ratio  $2 - 1/m$  is satisfied by an arbitrary list schedule. In Section 6, it is shown that  $K$ -periodic list schedules are the list schedules issued from special priority mappings called  $K$ -periodic linear orders. Section 7 concerns the transfer of the performance ratio of a list from the non-cyclic case to the cyclic case. It is first shown that if a list offers the performance ratio  $\rho$  in the non-cyclic case and has a quasi- $K$ -periodic structure in the cyclic case, each of its corresponding  $K$ -periodic linear orders provides a list schedule with the same guarantee. Finally, the Coffman–Graham's list [4] is shown to be quasi- $K$ -periodic and its well-known performance ratio  $2 - 2/m$  may be nicely transferred to the special case of *CSP* with unit execution times.

## 2. The cyclic scheduling problem *CSP*

A scheduling problem is *cyclic* if the set of tasks, the set of precedence constraints and the set of resource constraints have a *periodic* structure. In the case of the basic problem *CSP*, this structure is as follows:

### 2.1. The tasks

The tasks set  $\mathcal{T}$  is partitioned into an infinite number of *iterations*. Each iteration is indexed by a natural number  $n \geq 1$  and the tasks of the iteration  $n$  are denoted by  $T_j^n, T_j \in T$  where  $T = \{T_1, T_2, \dots, T_N\}$  is a finite set of so-called *generic* tasks. The task  $T_j^n$  is called the  $n$ th *execution* of  $T_j$ . Tasks are *not preemptive* and all the executions  $T_j^n, n \geq 1$  of the same *positive integer duration*  $p_j$ . Moreover, any two executions of the same generic task must be scheduled in non-overlapping time intervals (i.e. non-reentrance assumption). The maximum duration is denoted by  $p_{\text{max}}$ .

If  $\tau = \{T_{i_1}^{p_1}, \dots, T_{i_k}^{p_k}\}$ , then it will be convenient to denote by  $\tau^{+h}$  the subset  $\{T_{i_1}^{p_1+h}, \dots, T_{i_k}^{p_k+h}\}$ ; in the same way, if  $L = (T_{i_1}^{p_1}, \dots, T_{i_k}^{p_k})$  is a list of tasks, then  $L^{+h}$  is the list  $(T_{i_1}^{p_1+h}, \dots, T_{i_k}^{p_k+h})$ .

### 2.2. The precedence constraints

The precedence constraints are defined from a finite set  $U = \{u_1, \dots, u_p\}$  of so-called *generic uniform* precedence constraints. Each  $u_p$  is a triple  $(T_i, T_j, h)$  where  $T_i$  and  $T_j$

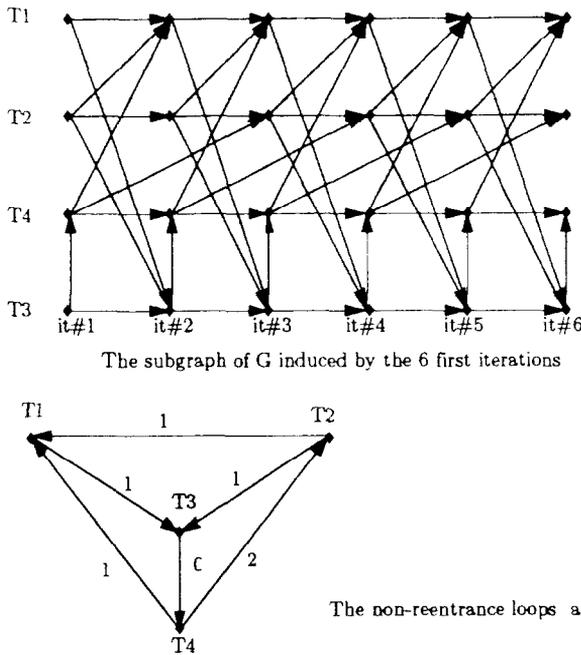


Fig. 1. The graphs  $G$  and  $\widehat{G}$ .

are two generic tasks and where  $h$  is a natural number called the *height* of  $u_p$ . The maximum height of a generic precedence constraint is denoted by  $h_{\max}$ . If  $u_p = (T_i, T_j, h)$  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 being performed.

Note that the non-overlapping constraint for the executions of  $T_j$  is equivalent to the generic precedence constraint  $(T_j, T_j, 1)$ . As shown in Fig. 1, the precedence graph  $G$  is an infinite directed acyclic graph with a periodic structure. 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 Fig. 1). The reduced precedence graph of an instance of CSP is assumed to be *strongly connected*.

### 2.3. The resource constraints

$m$  identical processors  $\{P_1, \dots, P_m\}$  are available to execute the tasks. As usual, each task  $T_j^n, T_j \in T, n \geq 1$  is performed by one processor and, at any instant, one processor may perform at most one task. In order to break ties and simplify some proofs, it will be convenient to assume the priority linear order  $(P_1, \dots, P_m)$  on the set of processors.

#### 2.4. Schedule, average cycle time and optimization

An instance  $I = (G, h, p, m)$  of CSP 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  $S$  of  $I$  assigns each task  $T_j^k, T_j \in T, k \geq 1$  a starting time and a processor such that all the resource and precedence constraints are satisfied. The starting time of the task  $T_i^k$  in the schedule  $S$  is denoted by  $S(i, k)$ . The completion time  $C_n(S)$  of iteration  $n$  is  $\max\{S(j, n) + p_j | T_j \in T\}$  and the *average cycle time*  $\omega(S)$  of  $S$  is defined by  $\limsup_{n \rightarrow \infty} (C_n(S)/n)$ . 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 every generic task  $T_i$ , the sequence  $S(i, n)$  satisfies for every  $n \geq N_0$ :  $S(i, n + K) = S(i, n) + r$ . Note that in this case, we have  $\omega(S) - \limsup_{n \rightarrow \infty} C_n(S)/n = r/K$ .  $K$  is called the *periodicity factor* whereas  $N_0$  is the length of the *transient phase*.  $K$ -periodic schedules are particularly interesting in the field of cyclic scheduling since they constitute a dominant subset of schedules [11].

### 3. Schedule properties

Before studying list schedules, which form a specific class of schedules, we introduce in this section some general definitions and properties that refer to an arbitrary schedule  $S$  of an instance  $I$  of CSP. We let  $t \geq 0$  be an arbitrary instant.

A generic task  $T_i$  is said to be *active* at time  $t$  in  $S$  if there is one  $k \geq 1$  such that  $T_i^k$  is in progress at time  $t$  (i.e.,  $S(i, k) < t < S(i, k) + p_i$ ). Note that task  $T_i^k$  is active neither when it starts nor when it completes. The *residual execution time*  $R_i(t)$  of a generic task  $T_i$  at time  $t$  in  $S$  is then defined as follows:

$$\begin{aligned} R_i(t) &= S(i, k) + p_i - t \quad \text{if } T_i \text{ is active at time } t, \\ R_i(t) &= 0 \quad \text{otherwise,} \end{aligned}$$

where  $T_i^k$  is the active execution of  $T_i$  at time  $t$ .

The number of executions of  $T_i$  started in the time interval  $[0, t[$  is denoted by  $D_i(t)$ . Respectively, the number of executions of  $T_i$  completed in the time interval  $[0, t]$  is denoted by  $F_i(t)$ . Note that, if one execution of  $T_i$  is started at time  $t$ , this start is not taken into account in  $D_i(t)$ , whereas if one execution of  $T_i$  is completed at time  $t$ , this completion is counted in  $F_i(t)$ . The following lemma shows that the so-called *balance property* is satisfied since  $\widehat{G}$  is strongly connected.

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

**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  $\mu$  and let  $q = D_i(t)$ . Since  $T_j^{q-1+h(\mu)}$  is a successor of  $T_i^{q+1}$  in  $G$  and  $T_i^{q+1}$  completes its execution strictly after time  $t$ , we have  $S(j, q + 1 + h(\mu)) > t$ , which implies  $D_j(t) \leq q + h(\mu)$ .  $\square$

Let  $u_k = (T_i, T_j, h)$  be an arc of  $\widehat{G}$ . The *instantaneous label* of  $u_k$  is defined as  $M_k(t) = h + F_i(t) - D_j(t)$ .

**Lemma 2.** Let  $H_2$  be the maximum height of a simple circuit of  $\widehat{G}$ . For any arc  $u_k = (T_i, T_j, h)$  of  $\widehat{G}$  and for any time  $t$ , we have  $0 \leq M_k(t) \leq H_2$ .

**Proof.** Let  $q = F_i(t)$ . Since  $T_j^{q+1+h}$  is an immediate successor of  $T_i^{q+1}$  in  $G$  and  $T_i^{q+1}$  completes strictly after time  $t$ , we have  $S(j, h + q + 1) > t$ , which implies  $D_j(t) \leq h + q$ . Now let  $\rho$  be a simple circuit of  $\widehat{G}$ . By summing the instantaneous labels of the arcs of  $\rho$  and rearranging the terms, we get  $\sum_{u_k \in \rho} M_k(t) = \sum_{u_k \in \rho} h(u_k) + \sum_{T_i \in \rho} (F_i(t) - D_i(t))$ . As  $F_i(t) \leq D_i(t)$ , we get:  $\sum_{u_k \in \rho} M_k(t) \leq H_2$ .  $\square$

The components of the *instantaneous state*  $E(t) = [M(t), R(t)]$  of  $S$  at time  $t$  are the labels  $M_k(t), u_k \in U$  and the residual execution times  $R_i(t), T_i \in T$ . The *residual graph*  $G^+(t)$  is the (infinite) subgraph of  $G$  induced by the tasks  $T_i^k$  scheduled at time  $t$  or after.

Let  $u < t$ . The residual graphs  $G^+(u)$  and  $G^+(t)$  correspond in a translation with shift  $h$  if for every  $T_i \in T$ ,  $D_i(t) - D_i(u) = h$ . Fig. 2 illustrates the above definitions.

#### 4. List schedules

Let  $I = (G, h, p, m)$  be an instance of CSP. By analogy with the well-known notion of a list algorithm in the non-cyclic case, we need to define, in order to solve the resource conflicts, the analog of the usual priority list. To that end, we assume that a one-to-one *priority mapping*  $A$ , mapping the positive integers into the task set, is known as a data of the scheduling problem. The task  $T_i^k$  has a higher priority than the task  $T_j^q$  if  $A^{-1}(T_i^k) < A^{-1}(T_j^q)$ . As a quite specific example, it is easy to define a priority mapping from a priority list  $L = (T_{i_1}, \dots, T_{i_N})$  of the generic tasks by taking  $A^{-1}(T_j^q) = p + (q - 1)N$  if  $j = i_p$ . In Section 5, a more general type of priority mappings, called  $K$ -periodic linear orders, will be shown to generate the  $K$ -periodic list schedules.

Given a priority mapping  $A$ , the analog of the usual non-cyclic list algorithm is the infinite loop below where the (initially empty) current partial schedule  $LS$  is updated at each new decision time  $\theta_n$  ( $\theta_0 = -1$  is assumed).

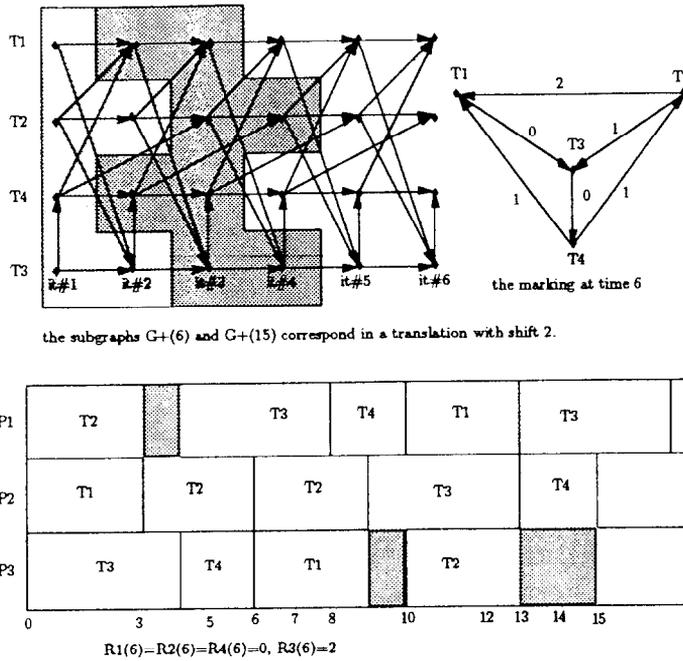


Fig. 2. Definitions associated with a schedule.

for  $n := 1$  to  $+\infty$  do

Let  $\theta_n$  be the smallest time greater than  $\theta_{n-1}$  such that

$READY(LS, \theta_n) > 0$  and  $FREE(LS, \theta_n) > 0$ ;

$\rho := READY(LS, \theta_n)$ ;  $\phi := FREE(LS, \theta_n)$ ;

$a := \min\{Card(\rho), Card(\phi)\}$ ;

Schedule in  $LS$  the  $a$  first (w.r.t.  $A$ ) tasks of  $\rho$

at time  $\theta_n$  on the first  $a$  processors of  $\phi$ ;

endfor.

$READY(LS, t)$  is the subset of the tasks whose predecessors are already completed by time  $t$  in the current schedule  $LS$ ,  $FREE(LS, t)$  is the subset of the processors that are free on and after time  $t$  in the current schedule  $LS$ . If  $LS(I, A)$  is the processor and time assignment constructed by the infinite loop from the input data  $(I, A)$ , we have:

**Lemma 3.**  $LS(I, A)$  is a schedule of  $I$ .

**Proof.** Since the precedence graph  $G$  has no circuit and any arc of  $\widehat{G}$  has a non-negative height, the decision-time  $\theta_n$ , as defined in the first line of the loop body, exists for any  $n \geq 0$ . So, the loop computes an infinite sequence of strictly increasing decision times.

Moreover, it is clear that the precedence and resource constraints are met by the tasks which are scheduled in  $LS(I, A)$ . What remains only to show is that for each

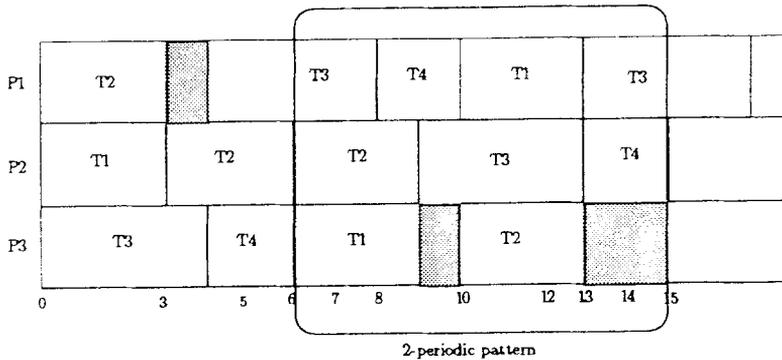


Fig. 3. A list schedule.

generic task, an infinite number of executions are performed. This is true for at least one generic task  $T_j$  since the sequence  $\theta_n$  is infinite. Assume now that only  $k$  executions of  $T_i$  are scheduled in  $LS(I, A)$ . Since  $\widehat{G}$  is strongly-connected, there is a simple path  $\mu$  from  $T_i$  to  $T_j$  in  $\widehat{G}$ . Since  $T_i^{k+1}$  is a predecessor of  $T_j^{k-1+h(\mu)}$ , we get a contradiction.  $\square$

Fig. 3 shows a list schedule for the instance of Fig. 1 and the priority mapping associated with the priority list  $L = (T_2, T_1, T_3, T_4)$  of the generic tasks.

The following lemma will be useful when studying  $K$ -periodic list schedules.

**Lemma 4.** *Let  $S = LS(I, A)$  be a list schedule and let  $\delta$  be a positive natural number. There are in  $S$  two decision times  $u$  and  $v$  ( $u < v$ ) such that  $E(u) = E(v)$ ,  $G^-(u)$  and  $G^+(v)$  correspond in a translation whose shift  $\Delta$  is at least  $\delta$  and  $READY(S, v) = (READY(S, u))^{+\Delta}$ .*

**Proof.** Let us first show that if  $E(u) = E(v)$ , ( $u < v$ ), then  $G^-(u)$  and  $G^+(v)$  correspond in a translation. From the label definition, we get for any arc  $u_k = (T_i, T_j, h)$  of  $\widehat{G}$ :  $F_i(v) - D_j(v) = F_i(u) - D_j(u)$ , which also writes  $F_i(v) - F_i(u) = D_j(v) - D_j(u)$ . Now the equality  $R_i(v) = R_i(u)$  implies  $F_i(v) - F_i(u) = D_i(v) - D_i(u)$ . So we have  $D_j(v) - D_j(u) = D_i(v) - D_i(u)$ . Since  $\widehat{G}$  is connected, we finally get that  $G^-(u)$  and  $G^+(v)$  correspond in a translation.

Now let  $\theta_n, n \geq 0$  be the (strictly increasing) sequence of the decision times of  $S$ . It is easy to see that for any generic task  $T_j$ , the residual execution time  $R_j(\theta_n)$  belongs to  $\{0, 1, \dots, p_j - 1\}$ . Moreover, we know from Lemma 2 that for any arc  $u_k$ , the label  $M_k(\theta_n)$  belongs to  $\{0, 1, \dots, H_2\}$ . The number of states at decision times is thus bounded above by a constant that only depends on  $I$ . So, there is at least one state  $E$  such that  $E(\theta_{n_k}) = E$  for an infinite increasing subsequence  $n_k, k \geq 1$ .

Assume that  $k > h_{\max}$ ,  $l - k \geq \delta$ ,  $u = \theta_{n_k}$ ,  $v = \theta_{n_l}$  and let  $\Delta = D_j(v) - D_j(u)$ . Note that  $\Delta \geq \delta$  and that for any  $T_i \in T$ , we have  $\Delta = D_i(v) - D_i(u) = F_i(v) - F_i(u)$ . For

any generic task  $T_i$ , we have  $D_i(u) \geq k > h_{\max}$ . If task  $T_j^q \in \text{READY}(S, u)$ , then we have  $q > h_{\max}$  and  $IN(T_j^{q+\Delta}) = (IN(T_j^q))^{+\Delta}$ . Moreover  $T_j^q \in \text{READY}(S, u)$  if and only if  $q > D_j(u)$  and for any task  $T_i^p \in IN(T_j^q)$ ,  $F_i(u) \geq p$ , which is equivalent (since  $E(u) = E(v)$ ) to  $q + \Delta > D_j(u) + \Delta = D_j(v)$  and for any task  $T_i^{p+\Delta} \in (IN(T_j^q))^{+\Delta}$ ,  $F_i(v) \geq p + \Delta$  and this in turn is equivalent to  $T_j^{q+\Delta} \in \text{READY}(S, v)$ . We thus have  $\text{READY}(S, v) = (\text{READY}(S, u))^{+\Delta}$ .  $\square$

### 5. The performance of list schedules

In this section, we show that the average cycle time of any list schedule of an instance  $I$  of CSP is at most  $(2 - 1/m)\alpha(I)$ . As a first step, we give a characterization of the absolute minimum average cycle time  $\alpha(I)$  in terms of the limit of a sequence. The second step uses a comparison between two partial schedules and the Graham’s bound.

We need first to introduce some additional definitions. If  $S$  is a schedule of  $I$ , the restriction of  $S$  to its  $k$  first iterations is denoted by  $S_k$ . Clearly,  $S_k$  is a schedule of the subgraph  $G_k$  induced by the first  $k$  executions of each generic task. The makespan of  $S_k$  is denoted by  $M(S_k)$ . From  $S_k$  we define a time assignment  $S_k^\infty$  of all the tasks as follows:

$$S_k^\infty(i, n) = S_k(i, n) \quad \text{if } n \leq k,$$

$$S_k^\infty(i, n) = S_k^\infty(i, n - k) + M(S_k) \quad \text{otherwise,}$$

that simply consists in “repeating” the pattern of  $S_k$  every  $M(S_k)$  time units. The following lemma shows that  $S_k^\infty$  is a  $k$ -periodic schedule of  $I$ .

**Lemma 5.** *If  $S$  is a schedule of  $I$  and if  $k \geq 1$ , then  $S_k^\infty$  is a  $k$ -periodic schedule of  $I$ .*

**Proof.** The resource constraint is obviously satisfied by  $S_k^\infty$  since  $S_k$  is a feasible schedule of  $G_k$ . Moreover, from the definition of  $S_k^\infty$ , we have  $S_k^\infty(i, x) = S_k^\infty(i, z + 1) + yM(S_k)$  where  $x - 1 = ky + z$ , ( $0 \leq z < k$ ). Now let  $(T_i, T_j, h)$  be a generic precedence constraint, let  $n \geq 1$  and assume that  $n - 1 + h = yk + z$ , ( $0 \leq z < k$ ) and  $n - 1 = sk + \rho$ , ( $0 \leq \rho < k$ ). We clearly have  $s \leq y$  and  $|S_k^\infty(j, z + 1) - (S_k^\infty(i, \rho + 1) + p_i)| \leq M(S_k)$ .

If  $s < y$ , then  $S_k^\infty(j, n + h) - S_k^\infty(i, n) - p_i = S_k^\infty(j, z + 1) - (S_k^\infty(i, \rho + 1) + p_i) + (y - s)M(S_k) \geq 0$ .

If  $s = y$ , then  $h = (z - \rho)$  and  $S_k^\infty(j, n + h) - S_k^\infty(i, n) = S_k^\infty(j, z + 1) - (S_k^\infty(i, \rho + 1) + p_i) \geq 0$  since that constraint is satisfied by  $S_k$ .

So  $S_k^\infty$  is a  $k$ -periodic schedule.  $\square$

Let us now denote by  $O_n$  an optimal schedule of the task graph  $G_n$ , i.e., a schedule of  $G_n$  whose makespan is minimum. We can now give a characterization of  $\alpha(I)$ .

**Theorem 1.** *Let  $I$  be an instance of CSP. The absolute minimum average cycle time  $\alpha(I)$  is  $\limsup_{n \rightarrow \infty} M(O_n)/n$ .*

**Proof.** Let  $\lambda = \limsup_{n \rightarrow \infty} M(O_n)/n$  and let  $\varepsilon > 0$ . There is a schedule  $S$  whose average cycle time  $\omega(S)$  satisfies  $\alpha(I) \leq \omega(S) < \alpha(I) + \varepsilon$ . As  $\omega(S) = \limsup_{n \rightarrow \infty} C_n(S)/n$  and  $M(O_n) \leq C_n(S)$ , we have  $\lambda < \alpha(I) + \varepsilon$ . So we get  $\lambda \leq \alpha(I)$ .

Assume that  $\lambda < \alpha(I)$  and let  $\varepsilon > 0$  be such that  $\lambda + \varepsilon < \alpha(I)$ . From the definition of  $\lambda$ , we know there is one  $k \geq 1$  such that  $(1/k)M(O_k) < \lambda + \varepsilon$ . Consider now the schedule  $O_k^\infty$ ; the average cycle time of this schedule is  $(1/k)M(O_k)$ , which is strictly less than  $\alpha(I)$ , a contradiction.  $\square$

Let us now take an arbitrary priority mapping  $A$  and let  $S = LS(I, A)$ . The performance bound essentially relies on a comparison between two schedules of the (finite) task graph  $G_n$ : the first is  $S_n$ , the restriction of  $S$  to  $G_n$ , and the second is  $\Sigma_n$ , the list schedule we get using the restriction of  $A$  to the tasks of  $G_n$ .

**Theorem 2.** *The average cycle time  $\omega(S)$  of an arbitrary list schedule  $S$  of  $I$  satisfies  $\omega(S) \leq (2 - 1/m)\alpha(I)$ .*

**Proof.** Let  $S$  be a list schedule of  $I$ . From the definitions of  $S$  and  $\Sigma_n$ , we know that if for each decision time  $u < t$  of  $S$ , each task in  $READY(S, u)$  belongs to  $G_n$ , then the two schedules are the same within the time window  $[0, t]$ . For convenience, let  $H = H_1 + 1$ .

Assume that  $n \geq H$  and let  $t = C_{n-H}(S) = S(i, n-H) + p_i$ . We thus have  $D_i(t) = n-H$ . If task  $T_j^q$  belongs to  $READY(S, u)$ , task  $T_j^{q-1}$  is completed by time  $u$  and  $D_j(t) \geq q-1$ . Since from Lemma 1, we have  $|D_j(t) - D_i(t)| \leq H_1$ , we have  $q \leq n$  and the task  $T_j^q$  is in  $G_n$ . Now let  $u = C_n(S) = S(j, n) + p_j$  and assume that  $t < S(i, p) \leq u$ . We have  $p > n - H$  since  $t = C_{n-H}(S)$  and  $p \leq n - H_1$  since otherwise  $D_i(u) > n + H_1$  and  $D_j(u) = n$  would contradict the balance property. So, at most  $2H_1 + 1$  executions of each generic task are scheduled in the window  $[t, u]$  and, since a list schedule has no dormant time,  $u - t \leq (2H_1 + 1) \sum_{T_i \in T} p_i$ .

Consider now the schedule  $\Sigma_n$ . Since it coincides with  $S$  in the time window  $[0, t]$ , at most  $H$  executions of each generic task are executed after time  $t$ . Since  $\Sigma_n$  is a list schedule, we have  $t \leq M(\Sigma_n) \leq t + H \sum_{T_i \in T} p_i$ . So we have  $|M(S_n) - M(\Sigma_n)| \leq (2H_1 + 1) \sum_{T_i \in T} p_i$  and the average cycle time of  $S$  satisfies:

$$\omega(S) = \limsup_{n \rightarrow \infty} M(S_n)/n = \limsup_{n \rightarrow \infty} M(\Sigma_n)/n.$$

We finally get from Theorem 1:

$$\omega(S) \leq \left(2 - \frac{1}{m}\right) \limsup_{n \rightarrow \infty} M(O_n)/n = \left(2 - \frac{1}{m}\right) \alpha(I). \quad \square$$

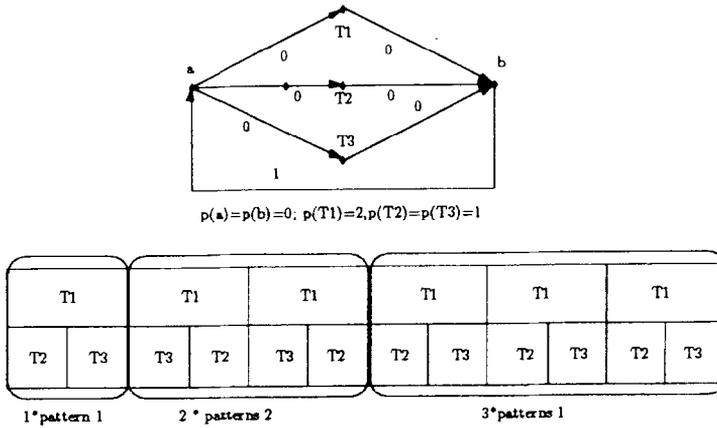


Fig. 4. An optimal list schedule with no periodic pattern.

### 6. K-periodic list schedules

In many applications, such as microcode optimization, it is often desirable to search for a periodic or even a  $K$ -periodic schedule because such schedules are far more easy to implement. However, an arbitrary priority mapping  $A$  does not guarantee the corresponding list schedule to have such a periodic structure. In the example of Fig. 4, where there is no overlapping between successive iterations, the priority mapping may be such that pattern 1 is chosen for the first iteration, pattern 2 for the next two iterations, pattern 1 for the next three iterations,... This leads to a list schedule (in fact an optimal schedule) that is not  $K$ -periodic.

In this section, we show that  $K$ -periodic list schedules are generated by a special class of priority mappings called  $K$ -periodic linear orders. If  $\tau = \{T_i^{p_1}, \dots, T_{i_k}^{p_k}\}$  is a finite subset of the tasks, the ordered list (w.r.t. a priority mapping  $A$ ) of the tasks of  $\tau$  is denoted by  $L_A(\tau)$ . Moreover, the subset of the  $n$  first executions of the generic task  $T_j$  is denoted by  $\mathcal{F}_j^n$  (with by convention  $\mathcal{F}_j^0 = \emptyset$ ).

A  $K$ -periodic linear order  $\Omega$  of the tasks is a priority mapping such that there exist  $N$  natural numbers  $\omega_j, j \in \{1, \dots, N\}$  and a positive integer  $N_0$  satisfying

$$\Omega(\{1, \dots, N_0 - 1\}) = \bigcup_{j=1}^N \mathcal{F}_j^{\omega_j}, \tag{1}$$

$$\forall n \geq N_0: \Omega(n) = T_i^p \Rightarrow \Omega(n + K) = T_i^{p+K}. \tag{2}$$

Note that  $\sum_{j=1}^N \omega_j = N_0 - 1$  and that condition (1) is irrelevant if  $N_0 = 1$ .

Let  $N_0^i(\Omega)$  be the number of executions of the generic task  $T_i$  appearing in  $(\Omega(1), \Omega(2), \dots, \Omega(N_0 - 1))$ . The first constraint means that for each  $T_i \in T$ , the subset of the executions of  $T_i$  that appear in  $(\Omega(1), \Omega(2), \dots, \Omega(N_0 - 1))$  is  $\{T_i^1, \dots, T_i^{N_0^i(\Omega)}\}$ . The second constraint is the periodicity requirement.

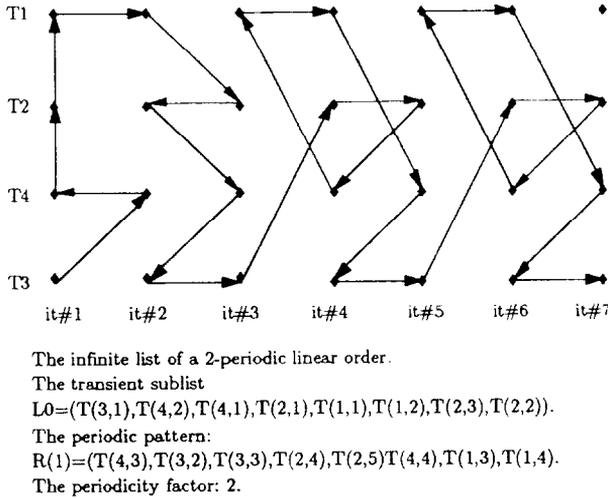


Fig. 5. A 2-periodic linear order.

The infinite list  $(\Omega(1), \Omega(2), \dots, \Omega(n), \dots)$  has thus a *transient* prefix sublist  $L_0 = (\Omega(1), \dots, \Omega(N_0 - 1))$  and a *periodic* infinite tail  $R = R(1) \cdots R(p) \cdots$  where  $R(p) = (\Omega(N_0 + (p - 1)K), \dots, \Omega(N_0 + pK - 1))$ .

Note that since  $\Omega$  is one to one,  $K$  successive executions of each generic task must appear in any  $R(p)$ . Moreover, if  $T_i^p$  is in  $L_0$ , then  $T_i^q$  is in  $L_0$  for any  $1 \leq q \leq p$ . Fig. 5 shows a 2-periodic linear order  $\Omega$ .

The periodic structure of  $\Omega$  implies the following simple lemma:

**Lemma 6.** *If  $\tau \subset R$  and  $h = 0 \pmod K$  then  $L_\Omega(\tau^{+h}) = (L_\Omega(\tau))^{+h}$ .*

We are now ready to prove that  $K$ -periodic linear orders generate all  $K$ -periodic list schedules.

**Theorem 3.** *Let  $I$  be an instance of CSP. If  $\Omega$  is a  $K$ -periodic linear order, then  $LS(I, \Omega)$  is a  $K'$ -periodic schedule where  $K'$  is a multiple of  $K$ . Conversely, if  $S$  is a  $K$ -periodic list schedule, there is a  $K$ -periodic linear order  $\Omega$  such that  $S = LS(I, \Omega)$ .*

**Proof.** Let  $L_0 \cdot R$  be the infinite list of  $\Omega$  and let  $S = LS(I, \Omega)$ . From Lemma 4, there is an infinite sequence  $w_n$  of increasing decision times such that for any  $n \geq 1$ ,  $E(w_n) = E$ . From Lemma 1, there is also a natural number  $N$  such that for any  $n \geq N$ :

1.  $READY(S, w_n) \subset R$ ,
2. for any  $T_i^q \in READY(S, w_n)$ ,  $q \geq h_{\max}$ .

Since  $G^+(w_n)$  and  $G^+(w_{n+1})$  correspond in a translation whose shift is denoted by  $h_n$ , we have

$$\forall n \geq N, \quad READY(S, w_n) = (READY(S, w_N))^{-(h_N + h_{N+1} + \dots + h_n)}$$

Let us now consider the sequence  $r_n = (h_N + h_{N+1} + \dots + h_{n-1}) \bmod K$  for  $n \geq N$ . Since  $r_n \in \{0, \dots, K-1\}$ , there exist  $p \geq N$  and  $q \geq N$  with  $p < q$  such that  $r_p = r_q$ . We thus have

$$READY(S, w_q) = (READY(S, w_p))^{+(h_p + \dots + h_{q-1})},$$

where  $H = (h_p + \dots + h_{q-1}) = 0 \bmod K$ . But now Lemma 6 implies

$$L_\Omega(READY(S, w_q)) = (L_\Omega(READY(S, w_p)))^{+(h_p + \dots + h_{q-1})}.$$

This last relation means that the task  $T_k^l$  is scheduled at time  $w_p$  in  $S$  if and only if the task  $T_k^{l+(h_p+\dots+h_{q-1})}$  is scheduled at time  $w_q$  in  $S$ .

Assume now that  $\theta_{p'}$  (respectively,  $\theta_{q'}$ ) is the decision time in  $S$  that follows immediately  $w_p$  (respectively,  $w_q$ ) in  $S$ . We have  $\theta_{p'} - w_p = \theta_{q'} - w_q$  and the relations:

1.  $E(\theta_{p'}) = E(\theta_{q'})$ ,
2.  $READY(S, \theta_{q'}) = (READY(S, \theta_{p'}))^{+H}$ ,
3.  $L_\Omega(READY(S, \theta_{q'})) = (L_\Omega(READY(S, \theta_{p'})))^{+H}$

are again satisfied. Using the same arguments for the next decision times, we get that the three preceding relations apply for all pairs  $\{\theta_{p'+k}, \theta_{q'+k}\}$  where  $k$  is a natural number. This ensures that the pattern of  $S$  within the time window  $[w_p, w_q]$  is repeated every  $(w_q - w_p)$  time units. So,  $S$  is  $H$ -periodic with period  $(w_q - w_p)$  and periodicity factor  $H = rK$ .

Conversely, let  $S$  be a  $K$ -periodic list schedule. Since  $S$  is  $K$ -periodic, the lexicographic order defined by:  $(\Omega(n) = T_i^p) < (\Omega(m) = T_j^q)$  if and only if

$$[S(i, p) < S(j, q)] \quad \text{or} \quad [(S(i, p) = S(j, q)) \text{ and } (i < j)]$$

is a  $K$ -periodic linear order. From the definition of  $\Omega$ , we have  $S = LS(I, \Omega)$ .  $\square$

The 2-periodic schedule shown in Fig. 3 is the list schedule associated with the 1-periodic linear order  $\Omega$  whose transient list is empty and whose periodic pattern is  $R_1 = (T_2^1, T_1^1, T_3^1, T_4^1)$ .

### 7. Transfer of non-cyclic performance ratios

In this section, we first show that the performance ratio of a list algorithm solving a non-cyclic scheduling problem is still satisfied in the cyclic case if the lists associated with the task graphs  $G_n$  make a *quasi K-periodic* sequence. We then show that the well-known  $2 - 2/m$  performance of the Coffmann–Graham’s list for the UET non-cyclic problem extends to the cyclic case.

#### 7.1. Quasi-K-periodicity

Before defining the quasi- $K$ -periodicity, we extend the notation  $L^{*h}$  to  $L^{*h}$  where  $L^{*h} = LL^{+1} \dots L^{+h}$  if  $h \geq 1$  and  $L^{*0} = L$ .

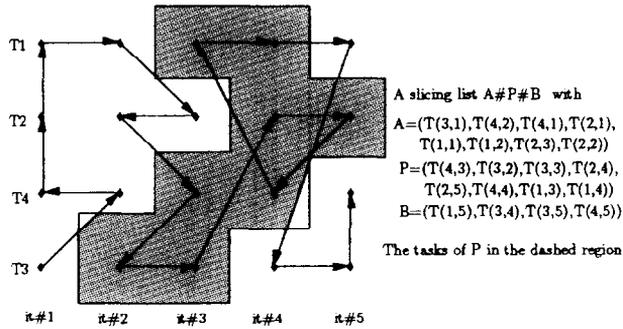


Fig. 6. A slicing list  $A\#P\#B$ .

Let  $L_n$  be the list associated with the finite task graph  $G_n$ .  $L_n$  is a *slicing list* if  $L_n = APB$  where

- $T_i^k \in A \Rightarrow \forall q \in \{1, \dots, k\}, T_i^q \in A,$
- $T_i^k \in B \Rightarrow \forall q \in \{k, \dots, n\}, T_i^q \in B,$
- Each generic task has the same number of instances in  $P$ .

Fig. 6 shows a slicing list for which  $P$  contains two executions of each generic task.

A convenient way to indicate that  $L_n$  is a slicing list is to write  $L = A\#P\#B$ .

The sequence  $L_n, n \geq 1$  is *quasi  $K$ -periodic* if there are a positive natural number  $K$  and  $K$  ordered pairs  $(x_r, Q_r), r \in \{0, \dots, K - 1\}$  of natural numbers such that for each  $r \in \{0, \dots, K - 1\}$ :

- $L_{KQ_r+r} = A_r\#P_r^{*x_r}\#B_r,$
- $\forall q \geq 0, L_{K(Q_r+q)+r} = A_r\#P_r^{*(x_r+q)}\#B_r^{+qK},$
- $P_r$  contains  $K$  instances of each generic task.

This rather tedious definition simply says that we get the list  $L_{K(Q_r+q)+r}$  from the list  $L_{KQ_r+r}$  by taking the same prefix sublist, descending  $q$  steps further than in  $L_{KQ_r+r}$  through the periodic pattern, and finally taking the correctly translated suffix sublist. Fig. 7 illustrates the structure of a quasi 2-periodic sequence of lists.

Let  $\Pi$  be a special case of  $P/prec/C_{\max}$ . We define the corresponding cyclic scheduling problem  $\Pi^c$  as the set of instances  $(\hat{G}, p, m)$  of CSP such that for any  $n \geq 1$ , the (non-cyclic) scheduling problem  $(G_n, p, m)$  is an instance of  $\Pi$ . We further assume that there is a list algorithm for  $\Pi$  whose performance ratio is  $\rho$ . So, for each instance of  $\Pi$ , there is a list  $L$  yielding a list schedule  $S$  such that  $C_{\max}(S) \leq \rho C_{\max}^*$  where  $C_{\max}^*$  is the minimum makespan of a schedule.

**Theorem 4.** Let  $I$  be an instance of  $\Pi^c$ , let  $L_n$  be the list associated with the instance  $(G_n, p, m)$  of  $\Pi$  and assume that  $L_n$  guarantees the performance ratio  $\rho$ . If the sequence  $L_n$  is quasi- $K$ -periodic, for any  $r \in \{0, \dots, K - 1\}$ , the list schedule  $S^r = LS(I, \Omega^r)$  where  $\Omega^r$  is the  $K$ -periodic linear order  $A_r P_r^{*\infty}$ , satisfies  $\omega(S^r) \leq \rho \omega(I)$ .

**Proof.** Let  $r \in \{0, \dots, K - 1\}$ ,  $b_r$  be the length of  $B_r$  and  $\beta_r = b_r + H_1 + 2$  (recall that  $H_1$  is the maximum height of any simple path in  $\hat{G}$ ). For any natural number  $q$ , we

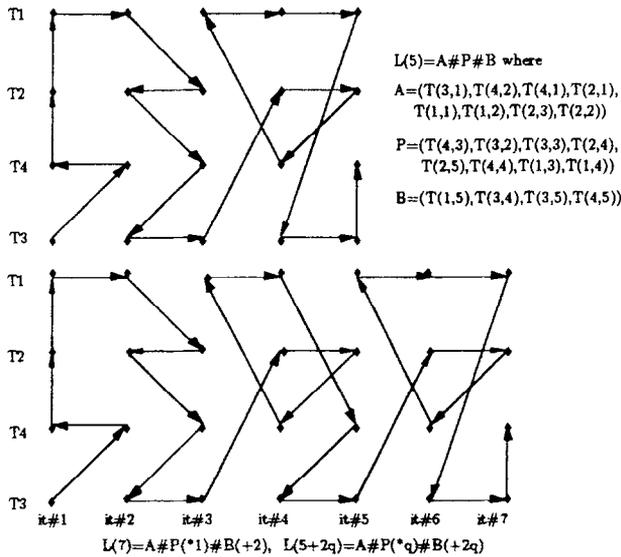


Fig. 7. A quasi 2-periodic sequence of lists.

define  $n_q = K(Q_r + q) + r$  and we denote by  $\Sigma_q$  the list schedule of  $G_{n_q}$  we get using the priority list  $L_{n_q}$  to schedule the task graph  $G_{n_q}$ .

By definition, the two schedules  $S^r$  and  $\Sigma_q$  are identical in the time window  $[0, t]$  if at any decision time  $u < t$  of  $S^r$ ,  $READY(S^r, u)$  is a subset of the tasks in  $A_r P_r^{*(x_r+q)}$ . Let us show that for any  $q$  such that  $n_q > \beta_r$ ,  $S^r$  and  $\Sigma_q$  are identical within the time window  $[0, t]$  where  $t = C_{n_q - \beta_r}(S^r)$ . Assume that  $t = S^r(j, n_q - \beta_r) + p_j$  and let  $T_i^k \in READY(S^r, u)$  where  $u < t$  is a decision time of  $S^r$ . The task  $T_i^{k-1}$  is thus completed by time  $u$ , so we have  $D_i(t) \geq k - 1$ . Since  $D_j(t) = n_q - \beta_r$ , we get from Lemma 1 that  $|D_j(t) - D_i(t)| \leq H_1$ , what implies  $k \leq n_q - \beta_r - 1$ . Since  $L_{n_q}$  is a slicing list,  $T_i^k$  is a task in  $A_r P_r^{*(x_r+q)}$  and the two schedules  $S^r$  and  $\Sigma_q$  are identical within the time window  $[0, t]$ .

Now let  $v = C_{n_q}(S^r)$  and let  $T_i^p$  be a task such that  $t \leq S^r(i, p) < v$ . We have  $p > n_q - \beta_r$  from the definition of  $t$  and  $p \leq n_q + H_1$  from Lemma 1. So at most  $H_1 + \beta_r$  executions of each generic task are started in the time interval  $[t, v]$  and since  $S^r$  is a list schedule we have  $v - t \leq (H_1 + \beta_r) \sum_{T_i \in T} p_i$ .

Consider now  $\Sigma_q$ . Since  $S^r$  and  $\Sigma_q$  coincide in  $[0, t]$  and  $\Sigma_q$  is a list schedule, at most  $\beta_r$  executions of each generic task are scheduled after time  $t$ . So we have  $M(\Sigma_q) - t \leq \beta_r \sum_{T_i \in T} p_i$ . From the two inequalities we get

$$|M(\Sigma_q) - C_{n_q}(S^r)| \leq (H_1 + \beta_r) \sum_{T_i \in T} p_i$$

from which we derive that  $\limsup_{q \rightarrow \infty} M(\Sigma_q)/n_q = \limsup_{q \rightarrow \infty} C_{n_q}(S^r)/n_q$  since  $\beta_r$  does not depend on  $q$ .

If  $O_{n_q}$  is an optimal schedule of  $G_{n_q}$ , we know from the performance ratio of  $L_{n_q}$  that  $M(\Sigma_q) \leq \rho M(O_{n_q})$ .

So we have  $\limsup_{q \rightarrow \infty} M(\Sigma_q)/n_q \leq \rho \limsup_{q \rightarrow \infty} M(O_{n_q})/n_q$ . Now from Theorem 1 and the definition of the absolute minimum average cycle time  $\alpha(I)$ , we know that  $\limsup_{q \rightarrow \infty} M(O_{n_q})/n_q \leq \alpha(I)$ . Moreover, since from Theorem 3,  $S'$  is a  $K'$ -periodic schedule (where  $K'$  is a multiple of  $K$ ) we have  $\limsup_{q \rightarrow \infty} C_{n_q}(S')/n_q = \lim_{n \rightarrow \infty} C_n(S')/n = \omega(S')$ . We may thus conclude that  $\omega(S') \leq \rho \alpha(I)$ .  $\square$

### 7.2. The UET special case of CSP

A nice consequence of the preceding result concerns the UET special case of CSP (*UET-CSP* in abbreviated form). It is well known that in the non-cyclic case, the Coffman–Graham’s list (C–G list in abbreviated form) provides a list schedule with performance ratio  $2 - 2/m$  and thus an optimal schedule when  $m=2$ . By showing in this section that the sequence of the C–G lists of an instance of *UET-CSP* is quasi- $K$ -periodic, we will conclude that each of the  $K$   $K$ -periodic linear orders associated with the sequence  $L_n$  generates a  $K$ -periodic list schedule with the same performance ratio.

Let us consider an instance  $I = (\widehat{G}, m)$  of *UET-CSP* and let  $L_n$  be the C–G list of the task graph  $G_n$ . The restriction of the C–G list  $L_n$  to a subset  $X$  of the tasks in  $G_n$  is denoted by  $L_n(X)$ . As usual, the level  $l_n(i, k)$  of the task  $T_i^k$  in  $G_n$  is the length (w.r.t. the number of arcs) of the longest path in  $G_n$  starting at node  $T_i^k$ . Note that  $l_n(i, k)$  may also be defined as the maximum length (w.r.t. the number of arcs) of a path in  $\widehat{G}$  with height  $n - k$  starting at node  $T_i$ . From this last definition, we clearly have:

**Lemma 7.** *For any two natural numbers  $k$  and  $n$  such that  $k \leq n$ , we have  $l_n(i, k) = l_{n-k+1}(i, 1)$ .*

As the C–G list of a task graph may not be unique (two tasks at the same level may have the same set of immediate successors), we will assume that the linear order  $(T_1, \dots, T_N)$  on the generic tasks is used to break ties.

We now study the structure of the (finite) sequence  $l_n(i, k)$  for a given generic task  $T_i$ , a given sufficiently large  $n$  and for  $k$  varying in  $\{1, \dots, n\}$ . Since  $\widehat{G}$  is strongly connected, the fundamental results on the basic cyclic scheduling problem [1–3] yield the following lemma where  $K$  is the least common multiple of the heights of the critical circuits of  $\widehat{G}$  and where  $L/K$  is the critical period of  $\widehat{G}$ :

**Lemma 8.** *There exist three natural numbers  $K, L$  and  $n_0$ , which do not depend on the generic task  $T_i$ , such that for any  $n > n_0$ ,  $l_{n+k}(i, 1) = l_n(i, 1) + L$ .*

Lemmas 7 and 8 yield the two following properties:

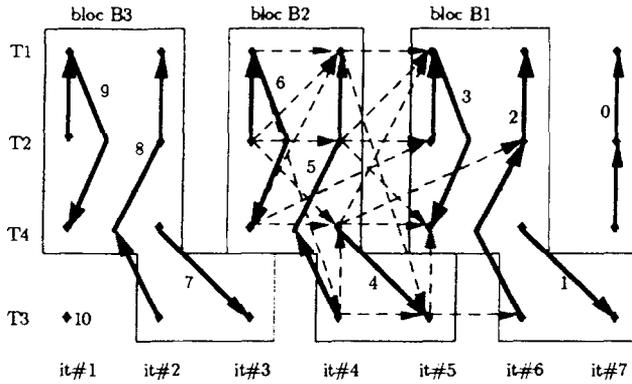


Fig. 8. Levels and blocks.

**Property 1.** *There is a natural number  $C$ , which only depends on  $L$ , such that for any  $n \geq 1$ , any two generic tasks  $T_i$  and  $T_j$  and any two natural numbers  $p, q \in \{1, \dots, n\}$ ,  $|p - q| \leq h_{\max}$  implies  $|l_n(i, p) - l_n(j, q)| < C$ .*

**Proof.** Let  $n - p + 1 = aK + b$  and  $n - q + 1 = cK + d$  where  $b, d \in \{0, \dots, K - 1\}$ . From Lemmas 7 and 8, we have  $l_n(i, p) = l_{n-p+1}(i, 1) = l_b(i, 1) + aL$  and  $l_n(j, q) = l_{n-q+1}(j, 1) = l_d(j, 1) + cL$ . Now the assumption  $|p - q| < h_{\max}$  implies  $|a - c| < h_{\max} + 1$ . Moreover, since the sequence  $l_n(i, 1)$  is positive and strictly increasing, we have  $|l_b(i, 1) - l_d(j, 1)| \leq \max_{T_i \in T} l_K(i, 1)$ . We finally get from the above inequalities:

$$|l_n(i, p) - l_n(j, q)| \leq \max_{T_i \in T} l_K(i, 1) + (1 + h_{\max})L. \quad \square$$

The second property is a straightforward consequence of Lemma 8.

**Property 2.** *For any natural number  $b$  and for any  $k$  such that  $n - k \geq n_0$ , we have  $l_n(i, k - bK) = l_n(i, k) + bL$ .*

In what follows, we define  $b$  as the least natural number such that  $bK \geq C$  and we let  $K' = bK$  and  $L' = bL$ .

Fig. 8 shows the structure of the sequence  $l_7(i, k)$  for the instance of Fig. 1 (where unit processing times are assumed). The thick arrows show the restriction of the C-G list to the tasks at the same level (this level being written close to each sublist). In this case, we have  $h_{\max} = 2$ ,  $n_0 = 0$ ,  $K = 2$ ,  $L = 3$ ,  $l_n(1, n - 2p) = l_n(2, n - 2p) = 3p$ ,  $l_n(1, n - 2p - 1) = l_n(2, n - 2p - 1) = 2 + 3p$ ,  $l_n(3, n - 2p) = l_n(4, n - 2p - 1) = 1 + 3p$ ,  $l_n(3, n - 2p - 1) = l_n(4, n - 2p) = 3p$ . Note that for this example, it is enough to take  $C = 3$ .

**Theorem 5.** *The sequence  $L_n$ ,  $n \geq 1$  is quasi- $K$ -periodic.*

**Proof.** Let us first introduce some notations. The subset of the tasks at level  $l$  in  $G_n$  is denoted by  $Z(l)$  and  $l_0$  is the lowest level such that  $Z(l_0) \cup Z(l_0 + 1) \cup \dots \cup Z(l_0 + L' - 1)$  contains  $K'$  successive executions of each generic task. The tasks in  $Z(l_0) \cup Z(l_0 + 1) \cup \dots \cup Z(l_0 + L' - 1)$  make the block  $B_1$ , and the blocks  $B(k) = Z(l_k) \cup \dots \cup Z(l_k + L' - 1)$ , where  $l_k = l_0 + L'(k - 1)$ , are defined in the same way as long as such blocks exist. Since  $n$  is sufficiently large, there are clearly two blocks  $B_p$  and  $B_q$  such that:  $\forall k \in \{0, \dots, L' - 1\}$ ,  $(L_n(Z(l_q + k)))^{+K'(q-p)} = L_n(Z(l_p + k))$ . Now, from the definition of the C-G list and the tie-breaking rule, the preceding relation implies that

$$(L_n(B_q))^{+K'(q-p)} = L_n(B_p). \tag{3}$$

We assume  $B_p$  is the first block with this property and we let  $d = q - p$ .

We now show that for any  $k$  such that the block  $B_{q+k}$  exists, the relation  $(L_n(B_{q+k}))^{+dK'} = L_n(B_{p+k})$  is also true.

Assume that  $B_q = Z(l) \cup Z(l - 1) \cup \dots \cup Z(l - L' + 1)$  and that bloc  $B_{q+1}$  exists. Let  $T_i^r$  be a task in  $Z(l + 1)$ . The task  $T_i^{r+dK'}$  is then in  $Z(l - dL' + 1)$ . If task  $T_j^s$  is an immediate successor of task  $T_i^r$ , then we have  $l_n(j, s) \leq l$  and since  $s - r \leq h_{\max}$ , we get from Property 1 that  $l_n(j, s) > l + 1 - C$ . So the task  $T_j^s$  is in  $B_q$  and the task  $T_j^{s+dK'}$  is in  $B_p$ . But now the equalities  $(OUT(T_i^r))^{+dK'} = OUT(T_i^{r+dK'})$  and  $(L_n(B_q))^{+dK'} = L_n(B_p)$  imply that for any task  $T_i^r$  in  $Z(l + 1)$  we have  $(L_n(OUT(T_i^r)))^{+dK'} = L_n(OUT(T_i^{r+dK'}))$ . So, from the definition of the C-G list and the tie-breaking rule, we have  $(L_n(Z(l + 1)))^{+dK'} = L_n(Z(l - dL' + 1))$ . Using the same arguments for each of the  $L'$  successive levels of the block  $B_{q+1}$ , we get that  $(L_n(B_{q+1}))^{+dK'} = L_n(B_{p+1})$  and more generally that

$$(L_n(B_{q+k}))^{+dK'} = L_n(B_{p+k}) \tag{4}$$

as far as the block  $B_{q+k}$  exists. In the example of Fig. 8, although  $C = 8$  should have made us choose  $b = 4$  following Property 2, it is easily seen that taking  $b = 1$  is enough since, as shown for the two first blocks, any task in a block has all its immediate successors in the same block or in the next block.

Let us now consider the  $q - p$  consecutive blocks  $B_p, \dots, B_{q-1}$  and let  $C_1 = B_p \cup B_{p+1} \cup \dots \cup B_{q-1}$  be the first *superblock*. We define in the same way the superblocks  $C_2, \dots, C_s$  as long as such superblocks exist. From the equalities (3) and (4), we get  $(L_n(C_i))^{+dK'} = L_n(C_{i+1})$  for  $i \in \{1, \dots, s - 1\}$ . But this last relation shows that  $L_n$  is a slicing list which writes  $L_n = A\#P^{*s}\#B$  where  $P = L_n(C_s)$ . In the example of Fig. 8, superblocks correspond to blocks since  $q - p = 1$  and we have  $C_1 = B_1$ ,  $C_2 = B_2$  and  $C_3 = B_3$ .

Let us finally consider the C-G list  $L_{n+adK'}$  where  $a$  is a natural number and let  $T_i^s$  be an arbitrary task of  $G_{n+adK'}$ . If  $s \geq adK'$ , then we have from Lemma 7 that  $l_{n+adK'}(i, s) = l_n(i, s - adK')$ . Assume now that  $1 \leq s < adK'$ . We get from Lemma 8 that  $l_{n+adK'}(i, s) = l_{n+adK'}(i, s + K') + L'$  for any  $s \in \{1, \dots, adK'\}$ . So the three following properties

1.  $G_{n+adK'}$  has exactly  $ad$  more blocks than  $G_n$ ,

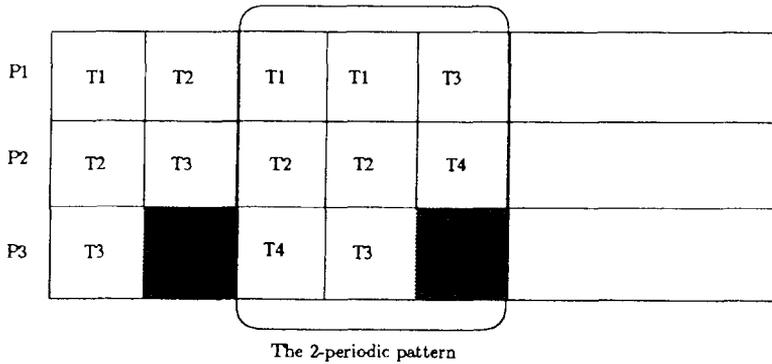


Fig. 9. An optimal list schedule.

- 2.  $G_{n+adK'}$  has exactly  $a$  more superblocks than  $G_n$ ,
  - 3. if  $L_n = A\#P^{*s}\#B$ , then  $L_{n+adK'} = A\#P^{*(s+a)}\#B^{+adK'}$
- are satisfied and the sequence  $L_n, n \geq 1$  is quasi  $K'$ -periodic.  $\square$

Fig. 9 shows the optimal schedule associated with the 2-periodic linear order  $AP^{*\infty}$  where  $A = (T_3^1)$  and  $P = (T_2^1, T_1^1, T_4^1, T_3^2, T_2^2, T_1^2, T_4^2, T_3^3)$ , corresponding to the sublists  $L_{2p+3} = A\#P^{*(p-1)}\#B^{+2p}$  ( $B = (T_4^3, T_2^3, T_1^3)$ ) of the  $UET-CSP$  instance of Fig. 1.

### 8. Concluding remarks

This paper has investigated some first step results about the definition and the performance of list schedules in cycling scheduling problems. Indeed there are many directions for future works on that subject. Among them, the three following questions are directly raised by the paper: Are there special cases of  $CSP$  for which better performance ratios than in the non-cyclic case may be obtained? What are good  $K$ -periodic linear orders for  $CSP$ , or for special cases of  $CSP$ ? May an optimal  $K$ -periodic linear order be computed in polynomial time for the  $UET-CSP$  problem?

### Acknowledgements

I greatly thank the referees for their helpful remarks and comments.

### References

- [1] P. Chrétienne, Transient and limiting behavior of timed event graphs, *RAIRO-TSI* 4 (1985) 127–142.
- [2] P. Chrétienne, The basic cyclic scheduling problem with deadlines, *Discrete Appl. Math.* 30 (1991) 109–123.

- [3] G. Cohen, D. Dubois, J.P. Quadrat, M. Viot. A linear system theoretic view of discrete event process and its use for performance evaluation in manufacturing. *IEEE Trans. Automat Control* 30 (1985) 3.
- [4] E.G. Coffman, Jr., R.L. Graham, Optimal scheduling for two processors systems. *Act. Inference* 13 (1972) 200–213.
- [5] F. Gasperoni, U. Schwiegelshohn, Generating close to optimal loop schedules on parallel processors. *Par. Proc. Lett.* 4 (4) (1994) 391–403.
- [6] R.L. Graham, Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* 17 (1969) 416–429.
- [7] C. Hanen, A. Munier, A study of the cyclic scheduling problem on parallel processors. *Discrete Appl. Math.* 57 (1995) 167–192.
- [8] C. Hanen, A. Munier, (1995). Cyclic scheduling on parallel processors. in: P. Chrétienne, E.G. Coffman Jr., J.K. Lenstra, Z. Liu (Eds.), *Scheduling Theory and its Applications*, Wiley, 1995, pp. 193–224 (Chapter 9).
- [9] E.G. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, D.B. Shmoys, Sequencing and Scheduling: algorithms and complexity. Report BS-R8909, Center for Mathematics and Computer Science, Amsterdam, 1994.
- [10] J.K. Lenstra, D.B. Shmoys, Computing near optimal schedules, in: P. Chrétienne, E.G. Coffman Jr., J.K. Lenstra, Z. Liu (Eds.), *Scheduling Theory and its Applications*. Wiley, New York, 1995, pp. 193–224 (Chapter 1).
- [11] A. Munier, Contribution à l'étude des ordonnancements cycliques. Ph.D. Thesis, P. and M. Curie University, 1991.