Complexity and approximation results for scheduling multiprocessor tasks on a ring

Giuseppe Confessore\textsuperscript{a}, Paolo Dell’Olmo\textsuperscript{b}, Stefano Giordani\textsuperscript{c,d,∗}

\textsuperscript{a}Istituto di Tecnologie Industriali e Automazione, Sezione di Roma, Consiglio Nazionale delle Ricerche, Area della Ricerca di Roma Tor Vergata, Via del fosso del cavalieri 100, Roma I-00133, Italy
\textsuperscript{b}Dipartimento di Statistica, Probabilità e Statistiche Applicate, Università di Roma “La Sapienza”, Piazzale Aldo Moro 5, Roma I-00185, Italy
\textsuperscript{c}Dipartimento di Informatica, Sistemi e Produzione, Università di Roma “Tor Vergata”, Via del Politecnico 1, Roma I-00133, Italy
\textsuperscript{d}Centro Interdipartimentale “Vito Volterra”, Università di Roma “Tor Vergata”, Via Columbia 2, Roma I-00133, Italy

Received 2 November 2000; received in revised form 6 June 2002; accepted 15 December 2002

Abstract

We study a multiprocessor task scheduling problem, in which each task requires a set of \( m \) processors with consecutiveness constraints to be executed. This occurs, for example, when multiple processors are interconnected by communication means, and the minimization of communication time may require the processors to be physically adjacent and each multiprocessor task to use only one subset of adjacent processors. In particular, we consider the case in which we have \( m \) processors arranged in a ring, and we want to find a schedule with minimum makespan. We investigate problem complexity, showing that the problem is \( \mathcal{NP} \)-hard in almost all the possible cases, and provide an approximation algorithm that finds a feasible schedule whose makespan is not greater than two times the optimal value.

© 2003 Elsevier B.V. All rights reserved.

Keywords: Multiprocessor task scheduling; Graph model; Acyclic orientations; Complexity analysis; Approximation results

∗ Corresponding author. Dipartimento di Informatica, Sistemi e Produzione, Università di Roma “Tor Vergata”, Via del Politecnico 1, Roma I-00133, Italy.
E-mail addresses: g.confessore@itia.cnr.it (G. Confessore), paolo.dellolmo@uniroma1.it (P. Dell’Olmo), giordani@disp.uniroma2.it (S. Giordani).

0166-218X/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0166-218X(03)00432-3
1. Introduction

One of the subjects receiving growing attention by researchers and in industries is parallel processing. The efficient management of limited processing resources requires the use of scheduling models and algorithms.

In this paper, activities have to be scheduled on a parallel architecture and are assumed to be multiprocessor tasks, i.e. tasks requiring the simultaneous allocation of a set of processors for their execution. An extensive description of multiprocessor task models can be found in [3,6,14]. The study of multiprocessor task models is motivated by the fact that, although classical scheduling theory assumes that a task can be processed by only one processor at a time, modern production and computer systems do not fit into such restrictive assumptions.

In order to capture the different aspects of real systems, several abstract scheduling models have been proposed. In particular, we refer to the case in which processor allocation can be prespecified or, equivalently, processors can be dedicated.

A further aspect to be examined is the processor configuration or architecture. Processor architecture design is becoming a difficult task as more requirements are imposed on the processors. This is the case of multiple processors interconnected by communication means where the objective ranges from minimizing field wiring, as for distributed control systems (see [9]), to minimizing communication delays (see [1,12]). In particular, the minimization of communication time may require the processors to be physically adjacent and each multiprocessor task to use only one subset of adjacent processors. In such a scenario, the communication time involved in the execution of a multiprocessor task can be included in the processing time, because it is small with respect to the execution time and no other processor out of those of the prespecified set is overloaded by communication. An example of such systems is wireless infrared processor networks [4]. Further on, we will refer to this latter requirement as the consecutive allocation constraint or simply, consecutiveness constraint.

One of the limits of multiprocessor task scheduling models with the consecutiveness allocation constraint is the limit in the number of possible processor configurations. This is especially evident if processors are arranged in a linear configuration as in an array. Indeed, no task requiring processors at the end and at the beginning of the arrangement can be processed. This problem can be circumvented by arranging the processors in a ring, i.e. as in a circular array. The ring configuration, with respect to the linear arrangement, is more powerful as it contains the entire task processor set of a linear arrangement plus all the configurations which include the extreme processors which are adjacent in the ring and separated in the array.

All the above considerations motivate the study of multiprocessor task scheduling on a ring of dedicated processors with consecutiveness constraints.

We formulate now the scheduling problem. Let \( \mathcal{P} = \{P_0, P_1, \ldots, P_{m-1}\} \) be a set of \( m \) dedicated processors organized in a ring, and \( \mathcal{T} = \{T_1, T_2, \ldots, T_n\} \) a set of \( n \) multiprocessor tasks. In general, in a multiprocessor scheduling problem each task \( T_i \) simultaneously requires a subset of dedicated processors \( \mathcal{P}(T_i) \subseteq \mathcal{P} \) for a processing time equal to \( p_i \). We consider the case in which each task \( T_i \) requires a subset of \( \mu_i \) consecutive dedicated processors \( \mathcal{P}(T_i) = (P_{j_i}; \mu_i) = \{P_{j_i}, P_{j_i+1}, \ldots, P_{j_i+\mu_i-1 (\text{mod} m)}\} \subseteq \mathcal{P} \).
\( \mathcal{P} \) of the ring, with \( 0 \leq j_i \leq (m - 1) \) and \( 1 \leq \mu_i \leq m \). Each processor \( P_j \) can execute at most one task at a time. No precedence constraint exists between tasks, and task preemption is not allowed. The objective is to find a schedule which minimizes the maximum completion time, that is the makespan.

In this paper, we study the case in which \( \mu_i = \mu \), for all \( i = 1, \ldots, n \), that is, each task \( T_i \) requires \( \mu \) (consecutive) processors \( \mathcal{P}(T_i) = (P_{j_i}; \mu) = \{P_{j_i}, P_{j_i+1}, \ldots, P_{j_i+\mu-1 \mod m}\} \) to be processed. Let \( (j_i; \mu) \) be the type of task \( T_i \). In the following we refer to this problem as a \( \mu \)-processor task scheduling problem on a ring of \( m \) processors.

We investigate the problem complexity in relation to its parameters, namely the number \( m \) of processors and the number \( \mu \) of consecutive processors required by the tasks, and, for the \( \mathcal{NP} \)-hard cases, provide an approximation algorithm that finds a feasible schedule whose makespan is not greater than two times the optimal value.

This paper also extends the results in [5,10,11], where the problem of scheduling multiprocessor tasks requiring \( \mu = 2 \) processors is considered.

We adopt a graph model which allows us to formulate the scheduling problem as the problem of finding an acyclic orientation of a (weighted) constraint graph, representing the conflicting relationships between tasks sharing at least one processor and, minimizing the length of the longest (weighted) path. Our approach finds a schedule by considering the orientation of the constraint graph which is obtained independently of the vertex weights (task processing times). This implies that the structure of the solution holds for any vectors of task processing times.

The paper is organized as follows: Section 2 contains the description of the graph model. In Section 3, we analyze the complexity of the studied problem. In Section 4, we provide a 2-approximation algorithm for the scheduling problem. Finally, Section 5 concludes the paper.

2. A graph model

Let \( G = (\mathcal{F}, E) \) be a graph associated to the multiprocessor task scheduling problem, where the vertex set is the set of tasks \( \mathcal{F} \), and an edge \( (T_i, T_j) \in E \) if and only if \( \mathcal{P}(T_i) \cap \mathcal{P}(T_j) \neq \emptyset \), that is tasks \( T_i \) and \( T_j \) are in conflict because they share at least one processor. Moreover, let \( w_{T_j} = p_j \) be the weight associated with vertex \( T_j \). The graph \( G \) is called the constraint graph.

In this graph model, as no pair of tasks (vertices) \( T_i, T_j \) joined by an edge can be simultaneously processed, it is necessary to decide which one has to be processed before the other, that is, it is necessary to find an orientation for the edge \( (T_i, T_j) \).

On that basis, it is possible to associate an acyclic orientation of the constraint graph \( G \) to a feasible schedule. In fact, a feasible schedule implicitly defines a precedence relation for each pair \( (T_i, T_j) \) of conflicting tasks, that is an orientation for the edge \( (T_i, T_j) \in E \). Vice versa, given an acyclic orientation of \( G \), it is possible to obtain a (semiactive [13]) schedule by scheduling each task at the earliest possible time while respecting precedence relations defined by the oriented graph. Moreover, the schedule makespan is equal to the length of the maximum weighted (oriented) path.
According to this graph model, optimal solutions to the scheduling problem correspond to acyclic orientations of the constraint graph having the length of the longest weighted path as short as possible; we call these orientations optimal. Any clique of the graph (where a clique is a set of mutually adjacent vertices), once oriented, produces a path of length equal to the weight of the clique, therefore the weight of the maximum weighted clique is a valid lower bound on the optimal solution value.

According to the well-known definition of a composition graph (e.g., see [8]), the constraint graph $G$ can be represented as $G_0[X^{(0,j)}, X^{(1,j)}, \ldots, X^{(m-1,j)}]$, where $X^{(j,j)}$, with $0 \leq j \leq m-1$, is the complete graph representing (mutually conflicting) tasks of the same type $(j; \mu)$, that is, requiring the same subset of processors $(P_{j}; \mu) = \{P_{j}, P_{j+1}, \ldots, P_{j+\mu-1 \mod m}\}$ and $G_0 = (V_0, E_0)$ is called the quotient graph, which is defined as follows. Each vertex $v_j$ of $G_0$, for $j = 0, \ldots, m-1$, represents, therefore, a task type $(j; \mu)$, and there is an edge $(v_i, v_j)$ if the task types $(i; \mu)$ and $(j; \mu)$ are (mutually) in conflict, that is $(P_{i}; \mu) \cap (P_{j}; \mu) \neq \emptyset$. Let us give the weight $w_j = p^{(j; \mu)}_j = \sum_{T_i \in \mathcal{F}^{(j; \mu)}} p_i$ to vertex $v_j$, where $\mathcal{F}^{(j; \mu)} = \{T_i | P(T_i) = \{P_{j}, P_{j+1}, \ldots, P_{j+\mu-1 \mod m}\}\}$. As in [2,3], we refer to a normal schedule as one in which tasks requiring the same set of processors are scheduled consecutively. The problem of finding the best normal schedule is equivalent to the problem in which there exists a single task $T^{(j; \mu)} \in \mathcal{F}^{(j; \mu)}$, for each subset of tasks $\mathcal{F}^{(j; \mu)} \subseteq \mathcal{F}$, with processing time $p^{(j; \mu)}_i = \sum_{T_i \in \mathcal{F}^{(j; \mu)}} p_i$, and hence $G \equiv G_0$.

Without loss of generality, we may suppose that, for each task type $(j; \mu)$, there exists at least one task; if it is not the case, we consider a dummy task (with processing time equal to zero). Hence, we may suppose that there exists a non-empty set $\mathcal{F}^{(j; \mu)}$ of tasks of type $(j; \mu)$, for $j = 0, \ldots, m-1$, with $p^{(j; \mu)}$ possibly equal to zero. In such way, the graph model and the related analysis is independent from specific instances. Therefore, we hereafter refer to the case where $G_0$ is a special case of unit circular arc graph, namely the graph $C^k_m$, with $k = \mu - 1$, that is the graph with vertices $v_0, v_1, \ldots, v_{m-1}$, and edges $(v_i, v_j)$ for $|i - j| \leq k \mod m$. In fact, by ordering the vertices $v_0, v_1, \ldots, v_{m-1}$ of $G_0$ in a clockwise order, each vertex $v_i$ of $G_0$ has to be exactly adjacent to $k = \mu - 1$ clockwise consecutive vertices preceding it, and $k = \mu - 1$ clockwise consecutive vertices succeeding it, to represent the conflicting relations between task types. In Fig. 1, the cases in which $m = 7$ and $k = 1$, and $m = 7$ and $k = 2$ are shown, respectively.
3. Complexity analysis

Hereafter, we analyze the complexity of the $\text{WSYN}$-processor task scheduling problem on a ring of $m$ processors, by classifying polynomial and $\mathcal{NP}$-hard cases. The analysis is based on the study of the structure of the quotient graph $G_0$ of the constraint graph $G$ related to the scheduling problem.

As we said in the previous section, optimal solutions to the scheduling problem correspond to optimal orientations of $G$. If $G$ is a comparability graph, then it permits a transitive orientation that can be found in polynomial time [8], in which, by transitivity, the vertices of each path are contained in a maximal clique of $G$. Hence, any such orientation is optimal; this implies that the (semiactive) schedule induced by such an orientation is optimal, and can be found in polynomial time. In particular, since $G$ can be represented as $G_0[X(0;\mu), X(1;\mu), \ldots, X(m-1;\mu)]$, where $X(j;\mu)$, with $0 \leq j \leq m-1$, is a complete graph representing (mutually conflicting) tasks of the same type $(j;\mu)$, $G$ is a comparability graph if and only if the quotient graph $G_0$ is a comparability graph. Therefore, any transitive orientation of $G_0$ induces an optimal normal schedule.

Without loss of generality, in the following we suppose that there is no task requiring all the $m$ processors. Moreover, without loss of generality, we suppose that $G_0$ is the graph $C_k^m$, where $k = \mu - 1$. We will characterize the cases when $C_k^m$ is a comparability graph. For any other possible cases, we will show that $C_m$ contains, as a subgraph, a hole (i.e., a chordless cycle) $C_v$ or an anti-hole (i.e., the complement of a hole) $\bar{C}_v$, with an odd number $v \geq 5$ of vertices, which are comparability graph forbidden subgraphs. A hole and an anti-hole of seven vertices are shown in Fig. 1, respectively.

While if $G_0$ is a comparability graph, the scheduling problem is polynomially solvable; in all other cases the presence of $C_v$, or $\bar{C}_v$, with $v \geq 5$ and odd, as a subgraph of $G_0$, implies that the scheduling problem is $\mathcal{NP}$-hard, as shown next.

It is known that

**Theorem 3.1** (Dell’olmo et al. [5]). The scheduling problem, with the quotient graph $G_0$ of the constraint graph $G$ being a hole $C_v$ with an odd number $v \geq 5$ of vertices, is $\mathcal{NP}$-hard.

From this theorem it directly follows that:

**Corollary 3.1.** The scheduling problem is $\mathcal{NP}$-hard if $G_0$ contains as a subgraph the hole $C_v$, with $v \geq 5$ and odd.

Now, we show that also when the quotient graph $G_0$ is an anti-hole $\bar{C}_v$, with an odd number $v \geq 5$ of vertices, the problem is $\mathcal{NP}$-hard, by exhibiting a polynomial transformation from the PARTITION problem, which is known to be $\mathcal{NP}$-hard [7]. The PARTITION problem is defined as follows:

**PARTITION problem.** Given a set $A=\{a_1, a_2, \ldots, a_z\}$ of $z$ integers such that $\sum_{i=1}^{z} a_i = 2B$, is there a partition of $A$ into 2 subsets $A_1$ and $A_2 = A \setminus A_1$ such that $\sum_{a_j \in A_1} a_j = \sum_{a_j \in A_2} a_j = B$?
Without loss of generality, we can restrict our analysis to the case in which the quotient graph $G_0$ is $C_{2k+3}^k$, with $k \geq 1$, that is the anti-hole $\tilde{C}_{2k+3}$. This is the case when we have to schedule a set of $(k+1)$-processor tasks on a ring of $m = 2k + 3$ processors.

**Theorem 3.2.** The $(k+1)$-processor task scheduling problem on a ring of $m = 2k + 3$ processors, where $G_0$ is the anti-hole $\tilde{C}_{2k+3}$ with $2k+3$ vertices, is $\mathcal{NP}$-hard.

**Proof.** For a given instance of the PARTITION problem, let us define a corresponding instance of the scheduling problem with $G_0 \equiv \tilde{C}_{2k+3}$.

Let $T^{(j;k+1)}$ be a task of type $(j;k + 1)$, with processing time $p^{(j;k+1)} = 2B$, for $j = 0,\ldots,2k+1 = m-2$, and let $T^{(2k+2;k+1)}_i$ be a task of type $(2k+2;k + 1)$, with processing time $p^{(2k+2;k+1)}_i = a_i$, for $i = 1,\ldots,z$, where the integers $a_i$, for $i = 1,\ldots,z$, represent the given instance of the PARTITION problem.

It is clear that the constraint graph is $G = G_0[X^{(0;k+1)},X^{(1;k+1)},\ldots,X^{(2k+2;k+1)}]$, with $G_0 \equiv \tilde{C}_{2k+3}$, $X^{(2k+2;k+1)}$ being a complete graph of $z$ vertices, and $X^{(j;k+1)}$ being a graph of a single vertex, for $j = 0,\ldots,2k+1$. See for example the case with $k = 3$ and $z = 3$ shown in Fig. 2.

If the PARTITION problem has a yes answer it is possible to construct a schedule $S^*$ of length $(2k + 3)B$ (see Fig. 3). Note that $S^*$ is the shortest possible schedule because there is at most one idle processor for each unit time period, and this is the best that one can have, since we have $m = 2k + 3$ processors and each task requires $k + 1$ dedicated processors. The implication is that in each unit time period at most two tasks can be simultaneously executed leaving exactly one processor in idle time.

Conversely, we show that if there exists a schedule of length $(2k + 3)B$ then there is a yes answer for the PARTITION problem. First, let us note that there is no normal schedule with such a length because any normal schedule has at least one unit time
period where more than \( k + 1 \) consecutive processors are idle. Therefore, the task set
\[ \mathcal{F}(2k+2;k+1) = \{ T_j^{(2k+2;k+1)} | i = 1, \ldots, z \} \]
has to be partitioned into two or more subsets. Note that only the possibility of Fig. 3 (or its mirror image) exists to schedule the tasks
\[ T^{(j/k+1)}, \text{ for } j=0, \ldots, 2k+1, \]
in a time interval equal to \((2k+3)B\), leaving two separated periods of length \( B \) where processors \( P_{2k+2}, P_0, \ldots, P_{k-1} \) are idle, in which the tasks in \( \mathcal{F}(2k+2;k+1) \) have to be processed. These tasks correspond to the \( z \) elements of the PARTITION instance. Then, we can conclude that if a schedule of length \((2k+3)B\) exists, the PARTITION instance has a yes answer. □

From this theorem it directly follows that:

**Corollary 3.2.** The scheduling problem is \( \mathcal{NP} \)-hard if \( G_0 \) contains as a subgraph the anti-hole \( \tilde{C}_v \), with \( v \geq 5 \) and odd.

Let us now analyze the structure of \( G_0 = C^k_m \), and show that it is a comparability graph, or contains a hole \( C_v \), or an anti-hole \( \tilde{C}_v \), with an odd number \( v \geq 5 \) of vertices, as a subgraph.

For \( k = 1 \), we have the holes \( C_m \) with \( m \geq 3 \) vertices. The complexity analysis for these cases was done in [5]. Summarizing, if \( m \geq 5 \) and odd, the problem is \( \mathcal{NP} \)-hard (see Theorem 3.1), while in all other cases \( C_m \) is a comparability graph, and hence the problem is polynomially solvable.

Now, we consider \( k \geq 2 \). Since the degree of each vertex of \( C^k_m \) is \( 2k \), without loss of generality, we can restrict our analysis to the case in which \( m \geq 2k + 1 \geq 5 \).

In particular, for \( m = 2k + 1 \), \( C^k_m \) is the complete graph \( K_m \) of \( m \) vertices; hence, it is transitively orientable, and so the scheduling problem is polynomially solvable.

The problem remains polynomial when \( m = 2k + 2 \), since it is possible to show that \( C^k_{2k+2} \) is a comparability graph. In fact, the graph \( C^k_{2k+2} \), which is a complete graph without \( m/2 = k + 1 \) edges \((v_i, v_{i+(m/2)})\), for \( i = 0, \ldots, (m/2) - 1 \), can be expressed as the
composition $C^k_{2k+2} = H_0[H_1, \ldots, H_{m/2}]$, where $H_0$ is a clique of $m/2$ vertices and each $H_i$, for $i = 1, \ldots, m/2$, is an induced subgraph of $C^k_{2k+2}$ formed by two non-adjacent vertices $v_{i-1}, v_{i-1+(m/2)}$ of $C^k_{2k+2}$. Since each $H_i$, for $i = 0, \ldots, m/2$, is a (trivial) comparability graph, the graph $C^k_{2k+2}$ is also a comparability graph [8].

Summarizing we have that

**Theorem 3.3.** The $(k+1)$-processor task scheduling problem on a ring of $m \leq 2k + 2$ processors, with $k \geq 2$, is polynomially solvable.

To complete our analysis we consider the cases with $k \geq 2$ and $m \geq 2k + 3$. We will show that in these cases $G_0 = C^k_m$ contains, as a subgraph, a hole $C_v$, or an anti-hole $\tilde{C}_v$, with an odd number $v \geq 5$ of vertices and conclude that in these cases the scheduling problem is $\mathcal{NP}$-hard.

Recalling the definition of $C^k_m$, as being the graph with vertices $v_0, v_1, \ldots, v_m$ and edges $(v_i, v_j)$ for $|i-j| \leq k \pmod{m}$, let us define with $l_{ij} = |i-j| \pmod{m}$ the length of the edge $(v_i, v_j)$.

Let us start by showing that

**Proposition 3.1.** For $k \geq 2$, and $m \geq \frac{5}{2}k + 5$, the graph $C^k_m$ contains a hole $C_v$, with $v \geq 5$ and odd.

**Proof.** Note that by hypothesis we have $m \geq 10$. Let $h = \min[k, \lceil(m+5)/5\rceil] \geq 2$, and let us consider a path $\pi$ in $C^k_m$ formed by $\pi = m - (h - 1)[m/h]_o \geq 0$ edges of length $h$, and $\beta = h[m/h]_o - m \geq 0$ edges of length $h-1$, where $\lceil x \rceil_o$ is the lowest odd integer not less than $x$.

First, let us consider the case in which $k=2$, hence $h=k=2$. In this case it is possible to show that $x > \beta$; in fact, for $m > 9$, which is our case, it is true that $2/3m > m/2 + \frac{3}{2}$, but $m/2 + \frac{3}{2} \geq [m/2]_o$, hence $\frac{2}{3}m > [m/2]_o$, that is $x > \beta$.

In particular, let $\pi$ be a path obtained by first alternating, for $\beta$ times, edges of length $h$ and edges of length $h-1$, and then adding the remaining $x - \beta$ edges of length $h$, e.g. $\pi = (v_0, v_h, v_{h+(h-1)}, \ldots, v_{\beta h + (\beta - 1)(h-1)}, v_{\beta h + \beta (h-1)}, v_{(\beta+1)h + \beta (h-1)}, \ldots, v_{((\beta + x - \beta)h + \beta (h-1))(mod m)})$, with $h=2$. The path $\pi$ is chordless since it results $2h - 1 \geq k + 1$, which assures that for every pair of non-consecutive vertices of the path there is no edge of $C^k_m$. Note that the path is a cycle in $C^k_m$, since $xh + \beta(h-1) = m$, which implies that $v_0 \equiv v_{(xh+\beta(h-1))(mod m)}$. Moreover, since $x + \beta = [m/h]_o$, the cycle is composed of an odd number of vertices, and $[m/h]_o \geq 5$, since $m \geq 10$ and $h=2$.

Now, let us consider the case $k \geq 3$, and let $\pi$ be a path obtained by considering $x$ edges of length $h$, followed by $\beta$ edges of length $h-1$, e.g. $\pi = (v_0, v_h, v_{2h}, \ldots, v_{zh}, v_{zh+(h-1)}, v_{zh+2(h-1)}, \ldots, v_{zh+(\beta(h-1))(mod m)})$. The path $\pi$ is chordless since it is possible to show that $2(h-1) > k$, which assures that for every pair of non-consecutive vertices of the path there is no edge of $C^k_m$. In fact, if $h=k$ it is obviously true that $2(h-1) > k$. If $h = (m+5)/5$, we have that $h = (m+5)/5 \geq [(m+5)/5] - \frac{4}{5} > m/5$, that is $2(h-1) > \frac{2}{5}m - 2$; but, since by hypothesis $m \geq \frac{5}{2}k + 5$, it results $2(h-1) > k$. Also in this case $\pi$ is a cycle in $C^k_m$, since $xh + \beta(h-1) = m$, which implies that $v_0 \equiv v_{(xh+\beta(h-1))(mod m)}$. Since $x + \beta = [m/h]_o$, the cycle is composed of an odd number
\(v = \lceil m/h \rceil_o\) of vertices. Moreover, \(v = \lceil m/h \rceil_o \geq 5\); in fact, if \(h = \lceil (m + 5)/5 \rceil\) we have \(v = \lceil m/[(m + 5)/5] \rceil_o \geq \lceil 5m/(m + 5) \rceil_o \geq 5m/(m + 5) > 3 + \frac{1}{5}\), since \(m \geq 10\); obviously this also implies that \(v = \lceil m/h \rceil_o \geq 5\) if \(h = \min[k, \lceil (m + 5)/5 \rceil] = k\) (since \(h = k \leq \lceil (m + 5)/5 \rceil\)). \(\Box\)

Now, let us show that

**Proposition 3.2.** For \(k \geq 2\), and \(2k + 3 \leq m \leq 3k + 2\), the graph \(C^k_m\) contains an anti-hole \(C_v\), with \(v \geq 5\) and odd.

**Proof.** In order to prove this, let us consider the complement \(C^k_m\) of \(C^k_m\), and show that it contains a hole \(C_v\), with \(v \geq 5\) and odd. Note that an anti-hole with 5 vertices is a hole. It is simple to see that each vertex \(v_i\) in \(C^k_m\) has degree \(d = m - 2k - 1\), where \(N(v_i) = \{v_{i+1}, v_{i+2}, \ldots, v_{i+d} \mod m\}\) is its neighborhood. Let us consider the path \(\pi = v_0, v_1, v_2, \ldots, v_{2h-1}(k+1)\mod m, v_{2h}(k+1)\mod m, \ldots, v_{(v-2)(k+1)}\mod m, v_{(v-1)(k+1)}\mod m, v_{(v-k+1+q)\mod m}, v_{(2h+1)\mod m}, \ldots, v_{(v-2)(k+1)}\mod m\), where \(v = \lceil m/(m - 2(k + 1)) \rceil_o\), and \(q = [(v - 1)/2]m - v(k + 1) = (v/2)(m - 2(k + 1)) - m/2\). Note that \(\pi\) is a valid path since it is possible to show that \(0 \leq q < d - 1\); in fact, being \(v = \lceil m/(m - 2(k + 1)) \rceil_o \geq m/(m - 2(k + 1))\), that is \(v(m - 2(k + 1)) \geq m\), we have that \(q \geq 0\); moreover, knowing that \(v = \lceil m/(m - 2(k + 1)) \rceil_o < m/(m - 2(k + 1)) + 2\), we have that \((v/2)(m - 2(k + 1)) - (m/2) < m - 2(k + 1)\), that is \(q < d - 1\). Note that the path \(\pi\) is a cycle in \(C^k_m\), since \(v(k + 1) + q = [(v - 1)/2]m\), which implies that \(v_0 = v_{(v-k+1+q)\mod m}\).

In particular, the cycle is composed of \(v \geq 5\) vertices; in fact, if this is not the case, it would be \(m/(m - 2(k + 1)) \leq 3\), that is \(m \geq 3(k + 1)\) which contradicts the hypotheses. Moreover, since by definition \(v\) is odd, the cycle is composed of an odd number of vertices, with \(v \geq 5\).

Now let us show that the cycle \(\pi\) is chordless. The cycle \(\pi\) can be re-written as \(\pi = v_0, v_{(v-k+1)}, v_{(v-k+1)}(k+1)-[(2h-1-1)/2]m, v_2(k+1)-[(2h-2)/2]m, \ldots, v_{(v-k+1)}(k+1)-[(v-2)/2]m, v_{(v-1)(k+1)}(k+1)-[(v-2)/2]m, v_{(v-k+1)+q}-(v-2)/2]m)\); in fact, knowing that \(v = \lceil m/(m - 2(k + 1)) \rceil_o < m/(m - 2(k + 1)) + 2\), it is simple to see that \(0 < (2h-1)(k+1)-[(2h-1)/2]m \leq k + 1\), and \(k + 1 < (2h)(k+1)-[(2h-2)/2]m \leq 2(k + 1) < m\), for \(h = 1, \ldots, (v - 1)/2\), and \(v(k + 1) + q = [(v - 1)/2]m\).

Moreover, since \((2h-1)(k+1)-[(2h-1-1)/2]m=(k+1)-[(2h-1-1)/2]d-1)\), being \(d = m - 2k - 1\), the cycle \(\pi\) can also be written as \(\pi = (v_0, v_{(v-k+1)}, v_{(v-k+1)}(k+1)-[(2h-1-1)/2]m, v_2(k+1)-[(2h-1-1)/2]d-1)\), \(v_{(v-k+1)}(k+1)-[(2h-1-1)/2]d-1)\).

Let \(L_k = (v_0, v_{(v-k+1)}-[(v-2)/2]m, v_{(v-k+1)}-[(v-2)/2]m, v_{(v-k+1)}-[(2h-1-1)/2]d-1), \ldots, v_{(v-k+1)}(k+1)-[(v-2)/2]m, v_{(v-k+1)}(k+1)\mod m, v_{(2k+1)}\mod m, \ldots, v_{(v-k+1)}\mod m\) be the (circular) list of the vertices of \(\pi\) ordered by non-decreasing vertex indices. By definition of \(q\), it follows that \((k+1)=[(v-1)(d-1)]-q\); then, we have \(L_k=(v_0, v_{(v-k+1)\mod m}(v-1)/2]m, v_{(v-k+1)\mod m}(v-2)/2m, \ldots, v_{(v-k+1)\mod m}(v-2)/2m\mod m, v_{(v-k+1)}\mod m, \ldots, v_{(v-k+1)}\mod m\) preceding \(v_0\) in the circular list \(L_k\), the distance between the first and the third vertex is not less than \(d\). Since in \(C^k_m\) the
vertices of $N(v_i)$ are (circular) consecutive indexed and the distance (in the ordered list) between the first vertex and the last vertex of $N(v_i)$ is less than $d$, the previous property implies that there is no vertex $v_i$ of $\pi$ that has more than two adjacent vertices in $\pi$, which implies that the cycle $\pi$ is chordless.  

For $k \geq 2$, Propositions 3.1 and 3.2 do not cover the analysis of the graphs $C_5^2$ and $C_{12}^3$, but it is simple to verify that both graphs contain $C_5$ as a subgraph, for example the chordless cycles $(v_0, v_2, v_4, v_5, v_7, v_0)$ and $(v_0, v_2, v_5, v_7, v_{10}, v_0)$, respectively.

Summarizing we have that, for $k \geq 2$ and $m \geq 2k + 3$, the graph $C_m^k$ contains a hole $C_\gamma$ or an anti-hole $\tilde{C}_\gamma$, with an odd number $\gamma \geq 5$ of vertices, as a subgraph.

Therefore, by Corollaries 3.1 and 3.2 it follows that:

**Theorem 3.4.** The $(k+1)$-processor task scheduling problem on a ring of $m \geq 2k + 3$ processors, with $k \geq 2$, is $NP$-hard.

### 4. An approximation result

In this section we give an approximation algorithm for the $NP$-hard cases of the $(k+1)$-processor task scheduling problem on a ring of $m$ processors. As we have seen in the previous section, these occur when the number of processors is $m \geq 2k + 3$, with $m$ odd if $k = 1$.

As we said in Section 2, the best normal schedules correspond to optimal acyclic orientations of $G_0$. If $G_0$ is a comparability graph, each transitive orientation of $G_0$ is optimal, since the vertices of each path are contained in (covered by) one maximal clique of $G_0$, and, hence, such an orientation implicitly defines a normal schedule which is also optimal. In all other cases, we proved the scheduling problem to be $NP$-hard, any acyclic orientation of $G_0$ has at least one path whose vertices cannot be covered by one maximal clique. One can attempt to provide in polynomial time an acyclic orientation of $G_0$ which guarantees that the vertices of any (oriented) path can be covered by a limited number $\alpha$ of cliques. Such an orientation induces a normal schedule whose makespan, being equal to the length of the longest weighted path in the oriented graph, is at most $\alpha$ times the optimum, for any set of task processing time values.

Next, we provide an approximation algorithm that finds a normal schedule by determining an acyclic orientation of the quotient graph $G_0 = (V_0, E_0)$ of the constraint graph $G$. The orientation of $G_0$ is obtained by partitioning the vertex set $V_0$ into subsets, each one inducing a transitively orientable subgraph of $G_0$, and, on the basis of this partition, by acyclically orienting $G_0$ such that the vertices of each path in the oriented graph are covered by at most $\alpha = 2$ cliques of $G_0$. Based on this orientation, we are in a position to state that the induced normal schedule has a makespan not greater than $2$ times the optimal solution value.

Without loss of generality, we consider the case in which $G_0 \equiv C_m^k$, with $m \geq 2k + 3$. By definition of $C_m^k$, this graph contains $m$ maximal cliques of $k + 1$ vertices.
Next, we provide a partition of the vertex set of $G_0$ into the subsets $R_1, \ldots, R_{2t}$, with $t \geq 1$, for which each $R_i$ induces a transitively orientable subgraph $G_0(R_i)$ of $G_0$. First, we partition the vertex set $V_0$ into $h = \lceil m(k+1) \rceil \geq 2$ cliques $Q_i$ of $k+1$ vertices, for $i = 1, \ldots, h$, plus a (possibly empty) clique $Q_0$ of $x = m - h(k+1)$ vertices, where it is clear that $0 \leq x \leq k$. In fact, let $Q_i = \{v_{(i-1)(k+1)}, v_{(i-1)(k+1)+1}, \ldots, v_{(i-1)(k+1)+k}\}$, for $i = 1, \ldots, h$, and $Q_0 = \{v_{(h-1)(k+1)+k+1}, \ldots, v_{(h-1)(k+1)+k+1+(m-h(k+1))}\}$ if non-empty. Second, from the partition into cliques $Q_0, Q_1, \ldots, Q_h$ of $V_0$, let $R_1 = Q_0 \cup Q_1$, $R_i = Q_i$ for $i = 2, \ldots, h$, and $R_{2t} = Q_h$ if $h$ is even otherwise $R_{2t} = Q_{h-1} \cup Q_h$.

Let us show that each subgraph $G_0(R_i)$ of $G_0$ is transitively orientable. Clearly, this is true if $R_i$ is a clique (e.g., for $i = 2, \ldots, 2t - 1$). But also for $R_1$ and $R_2$, even if they are not cliques, $G_0(R_1)$ and $G_0(R_2)$ are transitively orientable. In fact, we have that:

**Proposition 4.1.** Given the subgraph $G_0(Q_i \cup Q_{i+1})$ of $G_0 = (V_0, E_0) = C_m^k$, there exists a transitive orientation of $G_0(Q_i \cup Q_{i+1})$ in which each edge $(v_{(i-1)(k+1)+r}, v_{(i-1)(k+1)+s})$ is oriented from $v_{(i-1)(k+1)+s}$ to $v_{(i-1)(k+1)+r}$.

**Proof.** After having oriented each edge $(v_{(i-1)(k+1)+r}, v_{(i+1)(k+1)+s}) \in E_0$, with $v_{(i-1)(k+1)+r} \in Q_i$ and $v_{(i+1)(k+1)+s} \in Q_{i+1}$, with $0 \leq r, s \leq k$, from $v_{(i+1)(k+1)+s}$ to $v_{(i-1)(k+1)+r}$ according to the statement of the proposition, let us orient the remaining edges of $G_0(Q_i \cup Q_{i+1})$ between each pair of vertices $v_{(i-1)(k+1)+r_1}$, $v_{(i-1)(k+1)+r_2}$ of $Q_i$, $v_{(i+1)(k+1)+s_1}$, $v_{(i+1)(k+1)+s_2}$ of $Q_{i+1}$ from $v_{(i-1)(k+1)+r_1}$ to $v_{(i-1)(k+1)+r_2}$, with $0 \leq r_1 < r_2 \leq k$ (from $v_{(i+1)(k+1)+s_1}$ to $v_{(i+1)(k+1)+s_2}$, with $0 \leq s_1 < s_2 \leq k$).

Let us show that this orientation is transitive; this can be proved by showing that for each oriented path of three vertices $(x, y, z)$, it results in $(x, z) \in E_0$, with $(x, z)$ (i.e., the edge $(x, z)$ is oriented from $x$ to $z$). This is obviously true if all $x, y, z$ belong to the clique $Q_i$ ($Q_{i+1}$). If the vertices $x, y, z$ do not belong to the same clique $Q_i$ ($Q_{i+1}$), we have to consider only two subcases: the first one, where $x \in Q_{i+1}$ and $y, z \in Q_i$; the second one, where $x, y \in Q_{i+1}$ and $z \in Q_i$. In the former, without loss of generality, we may suppose that $x \equiv v_{(i-1)(k+1)+s_j}$, with $0 \leq s_j \leq k$, $y \equiv v_{(i-1)(k+1)+r_j}$, and $z \equiv v_{(i-1)(k+1)+r_2}$, with $0 \leq r_j < r_2 \leq k$. On the basis of the choices for edge orientation, since $r_j < r_2 \leq k$, the oriented path $(x, y, z)$ follows. Since $(x, y) \in E_0$, by definition of $G_0 = C_m^k$, it also results that $(x, z) \in E_0$, with $(x, z)$. In the latter, without loss of generality, we may suppose that $x \equiv v_{(i-1)(k+1)+s_j}$, $y \equiv v_{(i+1)(k+1)+s_j}$, with $0 \leq s_j < s_j \leq k$, and $z \equiv v_{(i-1)(k+1)+r_j}$, with $0 \leq r_j \leq k$. On the basis of the choices for edge orientation, since $r_j \leq k$ and $s_j < s_j$, the oriented path $(x, y, z)$ follows. Since $(y, z) \in E_0$, by definition of $G_0 = C_m^k$, it also results that $(x, z) \in E_0$, with $(x, z)$. \hfill \Box

Now, let analyze adjacency relations between the partition elements $R_1, \ldots, R_{2t}$ of $V_0$, with $t \geq 1$. We begin the analysis by considering adjacency relations between the partition elements $Q_0, Q_1, \ldots, Q_h$ of $V_0$. Given two disjoint subsets $V'$, $V''$ of $V_0$, we say that $V'$ and $V''$ are adjacent in $G_0 = (V_0, E_0)$ if and only if there exists an edge in $E_0$ between a vertex of $V'$ and a vertex of $V''$. Since $|Q_i| = k+1$ for $i = 1, \ldots, h$, it is simple to see that $Q_i$ is only adjacent to $Q_{i-1}$ and $Q_{i+1}$, for $i = 2, \ldots, h-1,
and if \( Q_0 \) is not empty, \( Q_0 \) is only adjacent to \( Q_1 \) and \( Q_h \); as for \( Q_1 \), in addition to being adjacent to \( Q_0 \) and \( Q_2 \), it is also adjacent to \( Q_h \) if \( |Q_0| < k \). From the adjacency relations between the cliques \( Q_0, Q_1, \ldots, Q_h \), it follows that \( R_1 \) is adjacent to \( R_{2t} \) since, if \( Q_0 \) is not empty \( Q_0 \) is adjacent to \( Q_h \), otherwise \( Q_1 \) is adjacent to \( Q_h \); moreover, \( R_i \), for \( i = 2, \ldots, 2t - 1 \), is only adjacent to \( R_{i-1} \) and \( R_{i+1} \).

On the basis of these adjacency relations, let \( \tilde{G}_0 \) be the acyclic orientation of \( G_0 \) obtained by applying the following algorithm:

**Orienting Algorithm**

**Step 0:** For \( i = 1, \ldots, 2t - 1 \), orient each edge \((x, y)\) of \( G_0 \), with \( x \in R_i \) and \( y \in R_{i+1} \), from \( x \) to \( y \) if \( i \) is odd, otherwise from \( y \) to \( x \).

**Step 1:** Orient the edges of the subgraph \( G_0(R_1) \) according to Proposition 4.1.

**Step 2:** For \( i = 2, \ldots, 2t - 1 \), orient the edges of the complete subgraph \( G_0(R_i) \) in a transitive way, e.g., from \( v_j \) to \( v_l \), with \( j < l \).

**Step 3:** Orient the edges of the subgraph \( G_0(R_{2t}) \) according to Proposition 4.1.

From the adjacency relations between \( R_1, \ldots, R_{2t} \), and since the vertices of each path in a transitively oriented graph are covered by exactly one clique of the graph, it follows that:

**Theorem 4.1.** There exists an acyclic orientation \( \tilde{G}_0 \) of the graph \( G_0 = C_m^k \), in which the vertices of each path of the oriented graph are covered by at most two maximal cliques of \( G_0 \).

Fig. 4 shows the orientation of \( G_0 = C_{16}^2 \) provided by the Orienting Algorithm; two different types of arrows are used to distinguish the orientation of the subgraphs \( G_0(R_i) \), for \( i = 1, \ldots, 2t \), from the orientation of the remaining edges of \( G_0 \). Clearly, the vertices of each oriented path are covered by at most two cliques of \( G_0 \).
Given the orientation $\tilde{G}_0$ of $G_0 = C_m^k$, and considering each vertex $v_j$ with the weight $w_j = \sum_{i \in T \in \mathcal{T}^{(j;j)}} p_i$, we get a normal schedule as the earliest start schedule $ES_{\tilde{G}_0}[w]$ induced by $\tilde{G}_0$ and the weights $w$, where tasks in $\mathcal{T}^{(j;j)}$ (i.e., of the same type $(j;\mu)$) are scheduled consecutively, starting from $t_{(j;j)} = 0$, if $v_j$ has no incoming oriented edge in $\tilde{G}_0$, or $t_{(j;j)} = \max\{t_{(v_i,v_j)}, t_{(v_j,v_i)}\}[w]$, otherwise.

Clearly, the makespan of this schedule is equal to the length of the longest weighted path of $\tilde{G}_0$, and hence

**Theorem 4.2.** The normal schedule for the $(k+1)$-processor task scheduling problem on a ring of $m$ processors induced by $G_0$ has a makespan not greater than two times the optimal one. Moreover, this approximation value is tight.

**Proof.** The proof for the first part of the theorem statement follows from above. Let us now show that the approximation value is tight.

Let us consider the following instance of the scheduling problem. We are given a set of $n = 16$ (3)-processor tasks (hence, $k = 2$), requiring a set of 3 consecutive processors of a ring of $m = 16$ processors, where there is a single task $T_j$ for each task type $(j;3)$, for $j = 0, \ldots, 15$. The vector of task processing times, ordered according to non-decreasing task indices, is $[a, W - 3a, a, W - 2a, a, W - 2a, a, a, W - 2a, a, a, W - 2a, a, a]$ with $W > 3a$. In this case, the constraint graph $G$ is equivalent to the quotient graph $G_0 = C_{16}^2$ (the same graph we used as an example for describing the Orienting Algorithm, see Fig. 4).

Since we have a single task for each task type, feasible solutions are normal schedules. The makespan of each feasible solution cannot be less than the weight of the maximum weighted clique of $G$, which is $W$. Therefore, the schedule of Fig. 5 is optimal.

Let us show that the orientation $\tilde{G}_0$ induces a normal schedule whose makespan is $2W$, proving the theorem. This orientation is obtained by first partitioning the vertex set of $G_0 = C_{16}^2$ into the cliques $\mathcal{Q} = \{Q_0, Q_1, Q_2, Q_3, Q_4, Q_5\}$, then by considering the partition $\mathcal{R} = \{R_1, R_2, R_3, R_4\}$, with $R_1 = Q_0 \cup Q_1$, $R_2 = Q_2$, $R_3 = Q_3$, and $R_4 = Q_4 \cup Q_5$, and, finally, by acyclically orienting $G_0$, according to the Orienting Algorithm (see $\tilde{G}_0$ of Fig. 6, where the values in brackets are the vertex weights $w$). The orientation $\tilde{G}_0$, with the vertex weights $w$, induces the earliest start schedule $ES_{\tilde{G}_0}[w]$ shown in Fig. 7, whose makespan is equal to $2W$. □

Note that our approach finds a normal schedule by considering the orientation of $G_0$ which is obtained independently of the vertex weights, that is the task processing times. This implies that the structure of the solution (i.e., task precedence relations) holds for any vectors of task processing times.

As for time complexity, the vertex partition of the vertices of $G_0 = (V_0, E_0) = C_m^k$ into subsets inducing transitively orientable subgraphs of $G_0$, and the orientation of $G_0$ can be obtained in $O(|V_0| + |E_0|)$ time. Since $|V_0| = m$, and $|E_0| = mk$, we have that the time complexity to find the structure of the normal schedule (i.e., the orientation $\tilde{G}_0$ of $G_0$) is $O(mk)$. The time required to find the schedule from the schedule structure is $O(n + mk)$, i.e. $O(n + mk)$. 

Fig. 5. The optimal solution for the tightness proof.

Fig. 6. The orientation $\tilde{G}_0$ of the weighted graph $G_0$ for the tightness proof.
5. Conclusions

We study a multiprocessor task scheduling problem, in which each task $T_i$ requires a set of $\mu_i=\mu$ processors with consecutiveness constraints to be executed. This occurs, for example, when multiple processors are interconnected by communications means, and the minimization of communication time may oblige the processors to be physically adjacent and each multiprocessor task to use only one subset of adjacent processors. In particular, we consider the case in which we have $m$ processors arranged in a ring, and we want to find a schedule with minimum makespan.

We investigate the problem complexity in relation to its parameters, namely the number of processors and the number of consecutive processors required by the tasks, showing the $\mathcal{NP}$-hard cases, and provide a 2-approximation algorithm.

We adopt a graph model which allows us to formulate the scheduling problem as the problem of finding an acyclic orientation of a weighted constraint graph, representing conflicting relationships between tasks sharing at least one processor, minimizing the length of the longest weighted path. We provide an orienting algorithm that finds an acyclic orientation of the constraint graph in which the vertices of each path are guaranteed to be covered by at most $x$ cliques, with $x = 2$. This orientation, obtained without considering vertex weights (processing times), induces a schedule of length at most two times the optimal one.

Note that this is the best result we can obtain by establishing approximation in terms of clique containments of path, since $x$ is in general an integer value. Possibly, by using the information on task processing times, one may expect to get a better approximation result, although the property of being independent of the task processing times is lost.
Some interesting open problems are proving the $\mathcal{NP}$-hardness in the strong sense or exhibiting the existence of an exact pseudo-polynomial algorithm, and analyzing whether there exists a non-approximability result.

Acknowledgements

The authors owe thanks to the anonymous referees for their helpful comments.

References