Algorithmic Operations Research Vol.7 (2012) 1–12

Theoretical aspects of scheduling coupled-tasks in the presence of compatibility graph

G. Simonin [a] B. Darties [c] R. Giroudeau [b] J.-C. König [b]

[a]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France, Univ de Toulouse, LAAS, F-31400 Toulouse, France
[b]LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France
[c]LE2I - UMR CNRS 5158,University of Burgundy, BP 47 870, 21078, Dijon, France

Abstract

This paper presents a generalization of the coupled-task sche-duling problem introduced by Shapiro [12], where considered tasks are subject to incompatibility constraints depicted by an undirected graph. The motivation of this problem comes from data acquisition and processing in a mono-processor torpedo used for underwater exploration. As we add the compatibility graph, we focus on complexity of the problem, and more precisely on the boundary between $\mathcal{P}$ and $\mathcal{NP}$-completeness when some other input parameters are restricted (e.g. the ratio between the durations of the two sub-tasks composing a task): we adapt the global visualization of the complexity of scheduling problems with coupled-task given by Orman and Potts [11] to our model, determine new complexity results, and thus propose a new visualization including incompatibility constraints. In the end, we give a new polynomial-time approximation algorithm result which completes previous works.

Key words: complexity, approximation algorithm, scheduling, coupled-tasks

1. Introduction

1.1. *Motivation*

This paper deals with the problem of data acquisition subject to incompatibility constraints in a submarine torpedo. Many scheduling issues arise in several situations, e.g. in a radar pulsing context [1,13], a radar system [9,10], or a particular application [4]. In our context, the torpedo is used to execute several submarine topographic surveys, including topological or temperature measurements. Its aim is to collect data and to process them on a mono-processor within a minimum timeframe. A collection of sensors acquires data for the torpedo. Each data consists in an acquisition task which is divided into two sub-tasks: a sensor first emits a wave which propagates in the water, then he gets a corresponding echo. Scheduling issues appear when several sensors using different frequencies can work in parallel, while acquisitions using the same frequency have to be delayed in order to avoid interferences. It is necessary for robotic engineers to have a good theoretical knowledge of this kind of problem. Thus, the aim of this work is to study many sub-configurations and determine complexity and approximation results on them.

Email: G. Simonin [simonin@lirmm.fr], B. Darties [Benoit.Darties@u-bougogne.fr], R. Giroudeau [rgirou@lirmm.fr], J.-C. König [konig@lirmm.fr].

1.2. *Modelling and related work*

Coupled-tasks [12] are a natural way to model data acquisition by our torpedo: each acquisition task can be viewed as a coupled-task A_i composed by two sub-tasks, respectively dedicated for wave transmission and echo reception.We note a_i and b_i the processing time of each sub-task. Between these two sub-tasks there is an incompressible and inextensible idle time L_i which represents the spread of the echo in the water. Due to hardware constraints, we do not work in a preemptive mode: once started, a sub-task cannot be stopped and then continued later. A valid schedule implies here that for any task started at t, the first sub-task is fully executed between t and $t + a_i$, and the second between $t + a_i + L_i$ and $t + a_i + L_i + b_i$. We note $\mathcal{A} = \{A_1, \ldots, A_n\}$ the collection of coupled-tasks to be scheduled. Incompatibility constraints also exist between tasks due to wave interferences. We say two tasks A_i and A_j are compatible if and only if they use different wave frequencies; thus any sub-task of A_i may be executed during the idle time of A_j, as in Figure 1. We introduce a graph $G_c = (\mathcal{A}, E_c)$ to model such this compatibility, where edges from E_c link any pair of compatible coupled-tasks.

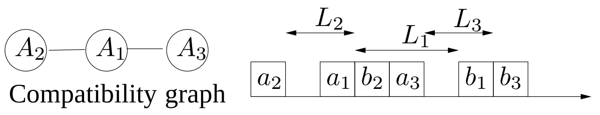


Fig. 1. Example of compatibility constraint with $A_1 = (a_1 = b_1 = 1, L_1 = 3)$, $A_2 = (a_2 = b_2 = 1, L_2 = 2)$, $A_3 = (a_3 = b_3 = 1, L_3 = 2)$

The aim is to produce a shortest schedule, i.e. to minimize the completion time C_{max} of the latest executed task, while respecting the incompatibility constraints between tasks. As this main problem is decomposable, we use the Graham's notation scheme $\alpha|\beta|\gamma$ [8] (respectively the machine environment, job characteristic and objective function) to characterize the sub-problems we study. We define the TORPEDO main problem as $1|a_i, L_i, b_i, G_c|C_{max}$, when there is not condition on the values of a_i,L_i,b_i for any task A_i.

In existing works, complexity of scheduling problems with coupled-tasks and no incompatibility constraint has been investigated [2,3,11]. Authors focus here their studies on precedence constraint between the acquisition tasks, which is different from the work presented in this paper: we study here a generalization which consists in introducing a compatibility graph G_c between tasks, and measuring the impact of the existence of G_c on the actual complexity and approximation results. In particular we focus on the boundary between polynomial and $\mathcal{NP}$-complete problems when we plan hypothesis on the values of a_i, b_i and/or L_i, and on the establishment of approximated solutions for difficult instances. In [11], authors give a global visualization of scheduling problems complexity with coupled-tasks through three trellis presented in Figure 2. Our approach is to achieve the same type of study in presence of a compatibility graph G_c. By comparing results of [11] with those obtained by relaxing the incompatibility constraint, we can measure impact of this constraint on this kind of problem.

Remark 1 *Note that, due to symmetrical features, sub-problems which consider restrictive hypothesis on the first sub-tasks a_i only have the same complexity than sub-problems which consider the same hypothesis on the second sub-tasks b_i only, and reciprocally.*

For example in Figure 2, problems $1|a_i = L_i, b_i|C_{max}$ and $1|a_i, L_i = b_i|C_{max}$ have the same complexity: indeed any algorithm which would compute an optimal solution for one would also produce an optimal solution for the other, simply by "reverting" each task A_i, computing an optimal solution, and then reverting again the obtained schedule. This fact is formally announced and proved in [11]. This reasoning can be extended when configurations suppose several hypothesis conjointly on sub-tasks a_i and b_i. We give a similar proof with incompatibility constraints in Lemma 5.

1.3. *Our contribution and Organization*

1.3.1. *Our contribution*

We will sharpen the demarcation line between the polynomial and $\mathcal{NP}$-hard case of the coupled-tasks scheduling problem in presence of compatibility graph G_c. Moreover, we design an efficient polynomial-time approximation algorithm based on maximum matching algorithm. We will also prove that a polynomial-time algorithm exists for some particular cases.

1.3.2. *Organization of the paper*

This paper is organized as follows:

- In the next section, we present some $\mathcal{NP}$-complete and polynomial results for different sub-problems of TORPEDO. This leads us to present global visualization inspired by the one presented in Figure 2 which takes into account the presence of compatibility graph G_c between tasks, and highlights its importance on problem complexity (see Table 2 and Figure 7);
- In the last section we give a polynomial-time approximation algorithm for the first studied problem, taking into account the values of some instance parameters.

2. Complexity results in presence of a compatibility graph

2.1. *Introduction*

In this section, we present several complexity results on different TORPEDO sub-problems. In order to perform a full study, we reuse problems identified on Figure 2. Taking into account incompatibility constraint makes problems more difficult than they were, thus problems which were $\mathcal{NP}$-complete without incompatibility constraint remain trivially $\mathcal{NP}$-complete when such constraint are introduced. Considering hierarchy of our problems, we will focus our study on problems which are at the limit of polynomiality and $\mathcal{NP}$-completeness or still open, and identified as problems Π_1, Π_2, Π_3 and Π_4 according to the diagram of Figure 2. For a better visibility, we will use the problem notation Π'_i as a reference of problem Π_i on which compatibility

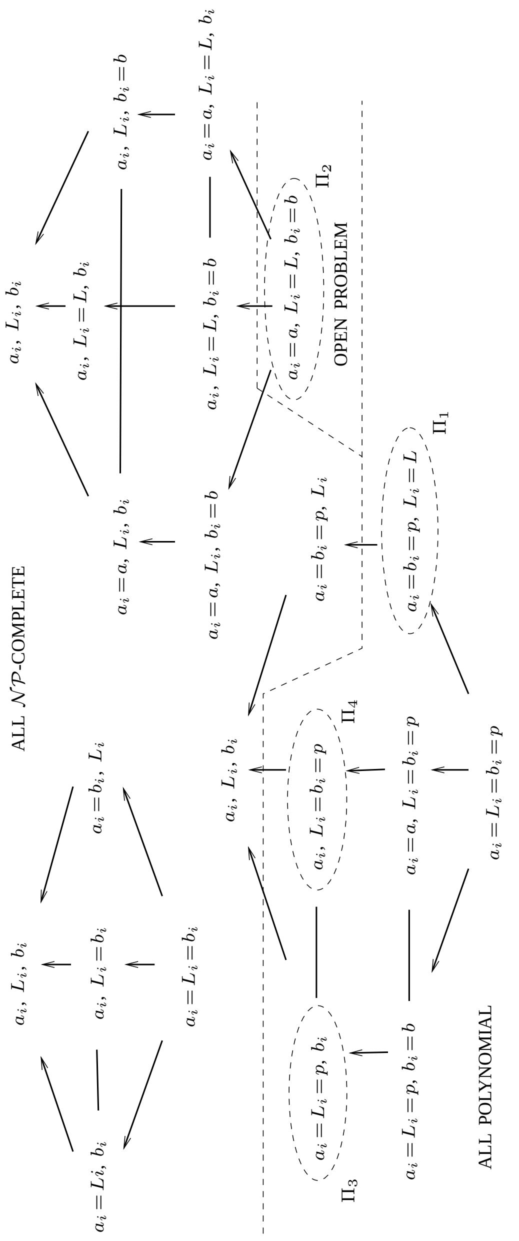


Fig. 2. Global visualization of the complexity of scheduling problems with coupled-tasks described by three distinct trellises in [11]. Triplet (a_i, L_i, b_i) describes the type of problem studied, where each variable a_i, b_i and L_i can take any value or be equal to a given constant. Finally, there is an arc from a specific problem to a more general problem, and an edge between two symmetrical problems.

graph is added. Results of this section are divided into four main parts, each part being devoted to the complexity study of a given sub-problem: first, we will prove the $\mathcal{NP}$-completeness of two scheduling problems:

- $\Pi'_1 : 1|a_i = b_i = p, L_i = L, G_c|C_{max}$
- $\Pi'_2 : 1|a_i = a, L_i = L, b_i = b, G_c|C_{max}$

Then we show the polynomiality of following problems:

- $\Pi'_3 :\ 1|a_i = L_i = p, b_i, G_c|C_{max}$
- $\Pi'_4 :\ 1|a_i, L_i = b_i = p, G_c|C_{max}$

We will prove in particular that the $\mathcal{NP}$-completeness of Π'_1 implies the $\mathcal{NP}$-completeness of Π'_2. For these problems, we will set some parameters in order to measure the influence of G_c on the evolution of the complexity.

In the rest of this paper, given a valid schedule σ and a task A_i, we note $\sigma(A_i)$ the starting time of the task A_i, i.e. sub-tasks a_i (respectively b_i) are fully executed between $\sigma(A_i)$ and $\sigma(A_i) + a_i$ (respectively between $\sigma(A_i) + a_i + L_i$ and $\sigma(A_i) + a_i + L_i + b_i$).

2.2. *Study of Problem* Π'_1

In sub-problem $\Pi'_1 = 1|a_i = b_i = p, L_i = L, G_c|C_{max}$, each sub-task requires the same execution time $p \in \mathbb{N}^*$, while the idle time L_i is identical for each task and fixed to a constant L. According to Orman and Potts, problem $\Pi_1 = 1|a_i = b_i = p, L_i = L|C_{max}$ is polynomial. We are going to study the complexity of Π'_1 by varying the value of parameter L according to the value of p. We study three disjoint cases, respectively $0 < L < p$, $p \leq L < 2p$ and $2p \leq L$. We prove that the first two cases are polynomial (Lemmas 1 and 2), the third being $\mathcal{NP}$-complete (Lemma 3):

Lemma 1 *When $0 < L < p$, Π'_1 is solvable in polynomial-time.*

Proof. When $0 < L < p$, it is easy to see that no task can overlap with the execution of another task. An optimal schedule consists in executing tasks sequentially without delay side by side. This algorithm admits a linear time complexity and produces a schedule of length $C_{max} = |\mathcal{A}| \times (2p + L)$. ■

Lemma 2 *When $p \leq L < 2p$, Π'_1 is solvable in polynomial-time.*

Proof.

When $p \leq L < 2p$, at most one sub-task with processing time p may be scheduled during the idle time L of another task. Thus, any scheduling of Π'_1 can be associated with a matching on G_c: tasks associated with the vertices covered by the matching edges are executed in pairs, creating "blocks" with an inactivity time of $(2L - 2p)$. For two tasks A_i and A_j we have $\sigma(A_j) = \sigma(A_i) + a_i$.

After ordering the tasks corresponding to the matching, we execute the remaining tasks sequentially. The length of the schedule will therefore depend on the size of the matching, and thus finding a matching with maximum cardinality in G_c provides an optimal schedule. Finding a maximum matching in a general graph has complexity $O(n^3)$ using Gabow's algorithm [6], and therefore the case $p \leq L < 2p$ is polynomial. ■

When $L \geq 2p$, we can now overlap the execution of more than two acquisition tasks, which leads us to search for cliques in the compatibility graph in order to reduce the inactivity time on the processor. We show this results in the $\mathcal{NP}$-completeness of Π'_1: we restrict our study of Π'_1 to the sub-case where $L_i = 2p$ for any task. We propose lemma 3 and prove the $\mathcal{NP}$-completeness of Π'_1 when $L_i = 2p$; then the generalization when $L_i \geq 2p$ is immediate.

Lemma 3 *Deciding if an instance of Π'_1 where $L_i = 2p$ for any task has a schedule of length $\beta = \sum_{i=1}^{n}(a_i + b_i) = 2np$ is a $\mathcal{NP}$-complete problem.*

Proof. Obviously, Π'_1 is in $\mathcal{NP}$. We prove the $\mathcal{NP}$-completeness of Π'_1 thanks to a polynomial time reduction from TRIANGLE PARTITION [7] whose purpose is to determinate if the vertices of a graph can be covered by disjoint triangles:

Let I^* be an instance of TRIANGLE PARTITION, i.e. a graph $G = (V, E)$ with $|V| = 3q, q \in \mathbb{N}^*$. From I^* we construct in polynomial-time an instance I of Π'_1 with a compatibility graph $G_c = (\mathcal{A}, E_c)$ as follows:

- $\forall i \in V$, an acquisition task A_i is introduced in $\mathcal{A}$, composed by two sub-tasks a_i and b_i of executed length $a_i = b_i = p$ and by an inactivity time between them of length $L_i = 2p$.
- For each edge $e = \{i, j\} \in E$, an edge $e_c = \{A_i, A_j\}$ is added in E_c. we have a non-exclusive relationship between the two tasks A_i and A_j.

Figure 3 illustrates such a transformation, which is clearly computable in polynomial time.

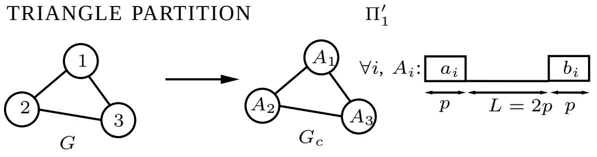


Fig. 3. Example of the polynomial-time transformation

Let us prove that the existence of a perfect TRIANGLE PARTITION on vertices of graph G implies the existence of an optimal schedule without idle time (then $C_{max} = n \times 2p$) , and reciprocally:

$\Rightarrow$ Suppose that there exists a TRIANGLE PARTITION on vertices of G. Then, let us show that there is a schedule without idle time of length $2np$ (being the sum of processing times). To do this, it is sufficient to form blocks of exactly 3 acquisition tasks A_i in G_c according to the TRIANGLE PARTITION produced on G. Figure 4 presents an example of such block formed with three tasks. The execution of these blocks forms a schedule without idle time. If all tasks can be included into such a block, then we obtain a schedule of length $2np$.

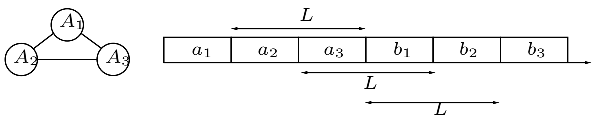


Fig. 4. Illustration of a block of three acquisition tasks, $\sigma(a_3) = C(a_2) = C(a_1) + a_2$ where $C(a_i)$ designates the completion of the sub-task a_i.

$\Leftarrow$ Conversely, if there is a schedule of length $2np$ on instance I, then let us show that vertices of G can be covered by exactly q triangles.

It is obvious that if $C_{max} = 2np$, then there is no idle time on the processor. This means that every idle slot of length $L_i = 2p$ is bound to be filled. However, we need three acquisition tasks carried into each other in order to obtain a block of three tasks without idle time. So, with exactly q blocks, we obtain a schedule without inactivity time. Since three acquisition tasks carried into each other are necessarily compatible in G_c, there exists a TRIANGLE PARTITION on vertices of G_c, and thus on vertices of G by construction.

Thus, we have TRIANGLE PARTITION $\propto$ [1] Π'_1. We know that TRIANGLE PARTITION is $\mathcal{NP}$-complete [7], So we can conclude that the problem Π'_1 is $\mathcal{NP}$-complete. ■

From the proof of $\mathcal{NP}$-completeness of Π'_1, we note that for $L = kp$ with $k \geq 2$ and $k \in \mathbb{N}$, the existence of a schedule without idle slot is equivalent to finding a partition of the vertices of G_c by disjoint cliques of size $(k+1)$ (which is equivalent to the $\mathcal{NP}$-complete problem PARTITION INTO SUBGRAPHS ISOMORPHIC TO H , where H is a clique of size $(k+1)$). The approximation study of problem Π'_1 is presented on the last section of this paper.

[1] We use a Polynomial-time reduction from TRIANGLE PARTITION to Π'_1.

2.3. *Study of problem* Π'_2

From the results obtained by Orman and Potts [11] (see Figure 2), we know that finding the complexity of Π_2 is still an open problem. We focus here on problem $\Pi'_2 : 1|a_i = a, L_i = L, b_i = b, G_c|C_{max}$, with $a, b, L \in \mathbb{N}^*$. By observing the values of parameters a_i and b_i, we state the following observation: Π'_2 is a generalization of Π'_1. Indeed, instances of Π_1 are particular cases of Π'_2 when $a = b = p$. This lead us to propose Theorem 1:

Theorem 1 *Decision problem Π'_2 is $\mathcal{NP}$-complete by generalization.*

We have shown that the problem $\Pi'_2 : 1|a_i = a, b_i = b, L_i = L, G_c|C_{max}$ was $\mathcal{NP}$-complete in the general case, a deeper complexity study has been performed in [14] when values of a and b are linked to each others.

2.4. *Study of problem* Π'_3

This problem consists of scheduling n acquisition tasks having the same model $a_i = L_i = p, b_i$. The first sub-task and idle time are set at the same constant p, $p \in \mathbb{N}^*$, while the second sub-task can take any value.

The set of these acquisition tasks contains two subsets: the first subset denoted K is composed by all the acquisition tasks A_i such that $b_i \leq p$, while the second subset denoted S is composed by all other tasks. Note that two tasks $A_i = (a_i, L_i, b_i)$ and $A_j = (a_j, L_j, b_j)$ in S cannot be executed one inside the other, so the edge (A_i, A_j) would be automatically removed if it appeared in G_c.

Theorem 2 *The scheduling problem $\Pi'_3 : 1|\ a_i = L_i = p, b_i, G_c\ |C_{max}$ is polynomial.*

Proof. First, note that not only there is at most on task in a idle time of another task, but a task in te idle time of another task can not admit a third task in its own idle time. The configuration proposed by problem Π'_3 allows only at most one sub-task to be scheduled during the idle time of a given task. As a consequence, tasks have to be scheduled either alone - then their idle time cannot be reused for scheduling other sub-tasks and is simply wasted on the processor -, or by blocks of exactly two tasks.

By weighting each edge of the graph with the sequential time of the overlap of the two tasks linked by the edge, our problem has an optimal solution if we can find a matching that minimizes not only the number of isolated vertices but also the sum of the weights of the matching edges.

For these purposes, we will use a known complexity result on THE MINIMUM WEIGHT PERFECT MATCHING PROBLEM, which consists in finding a perfect matching in a weighted graph where the sum of perfect matching edges is minimized. This matching can be computed within polynomial time [5].

From any instance of Π'_3, we propose the following polynomial-time construction:

Let $\mathcal{I}_1$ be an instance of our problem with a compatibility graph $G_c = (V_c, E_c)$, and $\mathcal{I}_2$ an instance of the minimum weight perfect matching problem in graph constructed from $\mathcal{I}_1$. Let $G'_c = (V'_c, E'_c)$ and $G''_c = (V''_c, E''_c)$ be two copies of compatibility graph G_c. The vertex corresponding to A_i is denoted A'_i in G'_c and A''_i in G''_c. From G'_c and G''_c we construct a graph $H_c = (V'_c \cup V''_c, E'_c \cup E''_c \cup E'''_c)$ with $E'''_c = \big\{\{A'_i, A''_i\} | A_i \in V_c\big\}$. We define the following weights on the edges of H_c:

- Each edge $\{A'_i, A'_j\}$ (resp. $\{A''_i, A''_j\}$), where $b_i > p$ or $b_j > p$, is weighted by $\frac{3p+max\{b_i,b_j\}}{2}$. This value represents half of the execution time used in the scheduling by the two coupled-tasks, where the second task belongs to S.
- Each edge $\{A'_i, A'_j\}$ (resp. $\{A''_i, A''_j\}$), where $b_i \leq p$ and $b_j \leq p$, is weighted by $\frac{3p+min\{b_i,b_j\}}{2}$. This value represents half of the execution time used in the scheduling by the two coupled-tasks that belong to K. The second executed task will be the one with the smallest b_i.
- Each edge $\{A'_i, A''_i\}$ is weighted by $2p+b_i$. This value represents the execution time used in the schedule by an isolated task.

Figure 5 illustrates such a construction where $w(A'_i, A'_j)$ denotes the weight of the edge (A'_i, A'_j). The equivalence between Π'_3 and MINIMUM WEIGHT PERFECT MATCHING is given by Lemma 4: and reciprocally.

Lemma 4 *For a minimum weight perfect matching of C, a schedule of minimum processing times C exists and reciprocally (see Figure 6).*

Proof. Indeed, the weight of each edge $e = \{A'_i, A'_j\} \in \{V'_c, V'_c\}$ (resp. $e = \{A''_i, A''_j\} \in \{V''_c, V''_c\}$), with $i \neq j$, corresponds to half the length of the scheduling on the processor for the acquisition tasks A'_i and A'_j (A''_i and A''_j) if they overlap. This overlap can be represented by a block. The weight of each edge $e = \{A'_i, A''_i\} \in \{V'_c, V''_c\}$ is the length of the scheduling on the processor for a simple acquisition task.

Note that H_c contains by construction an even number of vertices. Moreover, while each vertex of G'_c is connected to an equivalent vertex in G''_c, then a perfect matching on H_c is always available. This means that there exists a schedule such that each task is executed exactly once. Note that the matching in G'_c is not necessarily identical to the one in G''_c, but they still have the same weight. So, we can take the same matching in G'_c and G''_c without loss of generality. The makespan obtained is equal to sum of the processing times of the obtained blocks and those of isolated tasks. And since each isolated task (respectively block) has an execution time equal to the weight of the equivalent edge (respectively the two equivalent edges on G'_c and G''_c) in the perfect matching, we have the sum of edges weights of the matching which is equal to the blocks sum of the scheduling obtained. Thus, for a minimum weight perfect matching C, there exists a schedule of minimum length C and reciprocally.

■

This shows the relationship between a solution to the problem Π'_3 and a minimum weight perfect matching in H_c. This relationship is illustrated on Figure 6. While Edmonds algorithm gives a minimum weight perfect matching in $O(n^2m)$ [5], then problem Π'_3 can be solved in polynomial time. ■

Thus, the optimal polynomial-time algorithm to solve $\Pi'_3 : 1|\ a_i = L_i = p, b_i, G_c\ |C_{max}$, is decomposed into three steps:

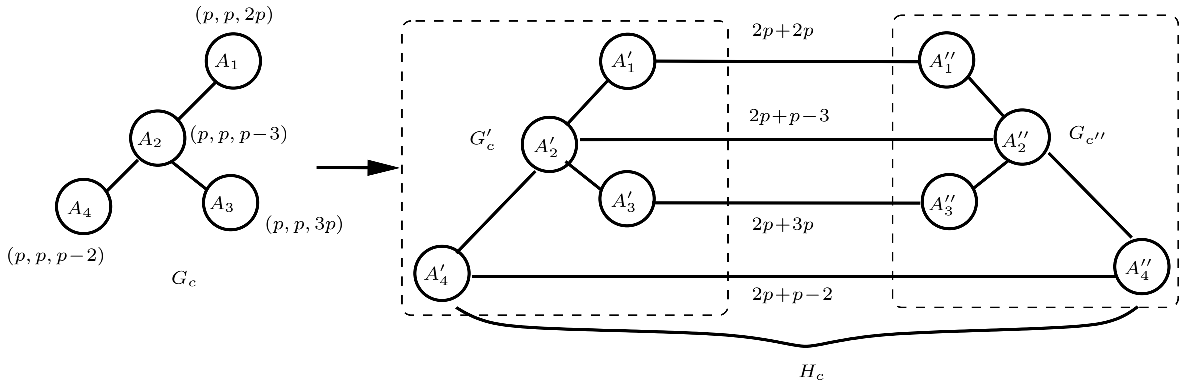
(1) First we create the graph H_c from G_c,
(2) then we compute a perfect matching on H_c with Edmond's algorithm;
(3) and to finish we produce the optimal schedule on the processor from the resulting matching.

We detail steps 2 and 3 through Algorithm 1, which returns a solution within complexity time of $O(n^2m)$.

2.5. *Study of problem* Π'_4

Problem $\Pi'_4 : 1|\ a_i, L_i = b_i = p, G_c|C_{max}$ is composed by n acquisition tasks with the following hypothesis: each sub-task a_i has a random duration, while all sub-tasks and idle times L_i have the same execution time p. :

Orman and Potts [11] gave us a theorem saying that a scheduling problem with acquisition tasks, where the objective is makespan, has the same complexity than

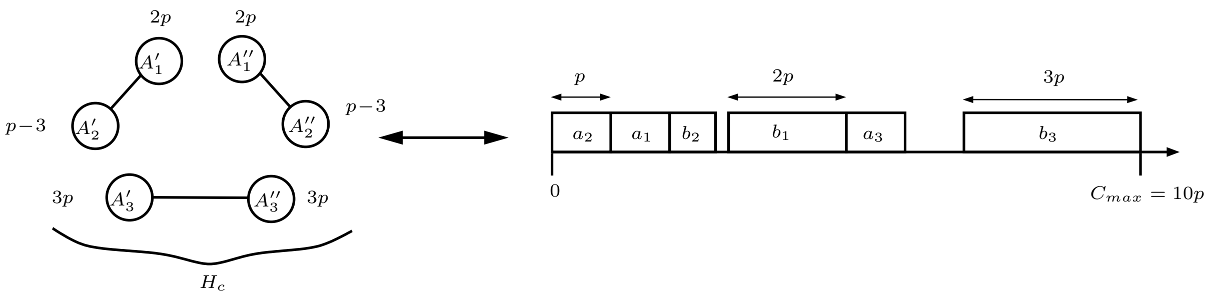


$$w(A'_1,A'_2)=w(A''_1,A''_2)=\frac{3p+2p}{2}$$

$$w(A'_2,A'_3)=w(A''_2,A''_3)=\frac{3p+3p}{2}$$

$$w(A'_2,A'_4)=w(A''_2,A''_4)=\frac{3p+p-2}{2}$$

Fig. 5. Example of the polynomial-time transformation.



$$w(A'_3,A''_3)=2p+3p \qquad w(A'_1,A'_2)=w(A''_1,A''_2)=\frac{3p+2p}{2}$$

Fig. 6. Correspondence between a perfect matching and an optimal schedule.

its symmetrical problem (this is not true for approximation). In Figure 2, there is a symmetry between $1|a_i=L_i=p,b_i|C_{max}$ and $1|a_i,L_i=b_i=p|C_{max}$ (Remind also Remark 1). We give the following Lemma for symmetric complexity equivalence by relaxing the incompatibility constraint:

Lemma 5 *Two problems which are symmetrical for the complexity, stay symmetric by relaxing the incompatibility constraint.*

Proof. The proof is the same as Orman et Potts; we consider any feasible schedule S with makespan $C_{max}(S)$ in which task i completes at time C_i for $i=1,...,n$. For the reverse problem, the schedule in which task i starts at time $C_{max}(S)-C_i$ for $i=1,,..,n$ is also feasible and has makespan $C_{max}(S)$. Similarly, any schedule for the reverse problem converts into a schedule for the original problem with the same makespan. By relaxing the incompatibility constraint, the proof stay the same. If the length change between the two solutions, that means our solution is not optimal. Thus, the two problems are equivalent. ■

Using Lemma 5, we propose the following corollary of Theorem 2:

Corollary 1 *Problem Π'_4 admits a polynomial-time algorithm.*

Proof. Problem Π'_4 is symmetrical to Π'_3 thanks to Lemma 5, and we know from Theorem 2 that problem Π'_3 is *polynomial*, then also Π'_4. The scheduling is optimal with Algorithm 1 by exchanging b_i and a_i values. ■

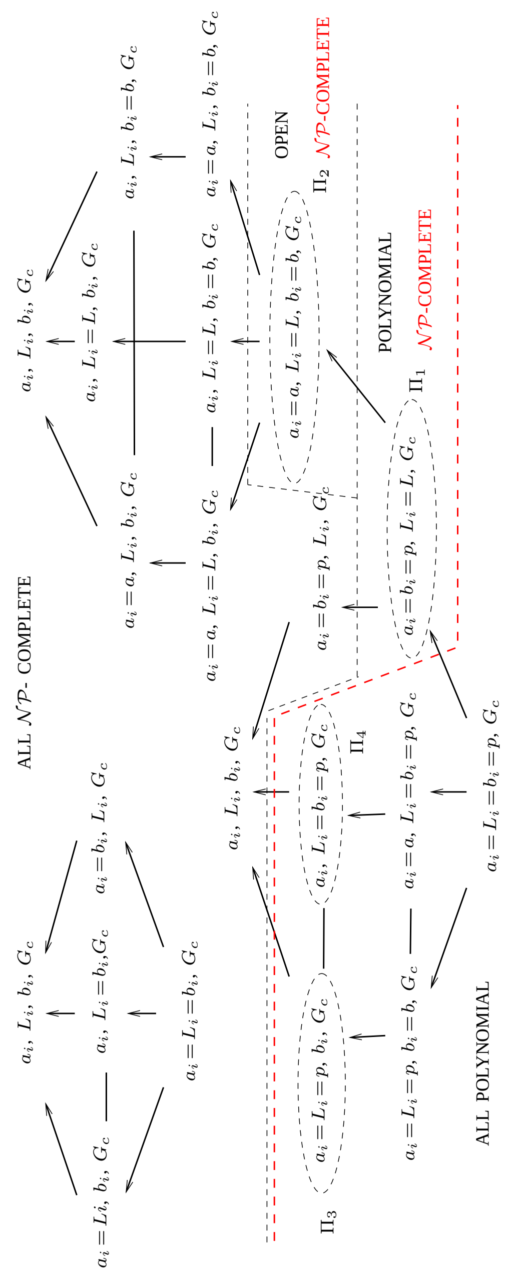

ALL $\mathcal{NP}$- COMPLETE
OPEN
$\mathcal{NP}$-COMPLETE
POLYNOMIAL
$\mathcal{NP}$-COMPLETE
ALL POLYNOMIAL
Π_1
Π_2
Π_3
Π_4
a_i, L_i, b_i, G_c
$a_i, L_i, b_i = b, G_c$
$a_i, L_i = L, b_i, G_c$
$a_i = a, L_i, b_i, G_c$
$a_i = a, L_i, b_i = b, G_c$
$a_i, L_i = L, b_i = b, G_c$
$a_i = a, L_i = L, b_i, G_c$
$a_i = a, L_i = L, b_i = b, G_c$
$a_i = b_i = p, L_i, G_c$
$a_i = b_i = p, L_i = L, G_c$
$a_i = Li, b_i, G_c$
$a_i, L_i = b_i, G_c$
$a_i = b_i, L_i, G_c$
$a_i = L_i = b_i, G_c$
$a_i = L_i = p, b_i, G_c$
$a_i, L_i = b_i = p, G_c$
$a_i = L_i = p, b_i = b, G_c$
$a_i = a, L_i = b_i = p, G_c$
$a_i = L_i = b_i = p, G_c$

Fig. 7. Global visualization of the impact of the incompatibility constraints on scheduling problems complexity with acquisition tasks on a single processor. The black dotted and red dotted lines represent the boundary between polynomiality and $\mathcal{NP}$-completeness respectively without and with the incompatibility constraint.

```
Algorithm 1: An optimal scheduling in polynomial time
input : $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$, $H_c$, $G_c$
output: $C_{max}^{opt}$
begin
    • Search in $H_c$ a perfect matching $M$ minimizing the weight of the matching edges
    • For each edge $e = (A'_i, A'_j) \in H_c$ (resp. $e = (A''_i, A''_j) \in H_c$) of the matching $M$, such that $A'_i$ and $A'_j$ (resp. $A''_i$ and $A''_j$) belong to the same graph $G'_c$ (resp. $G''_c$), the acquisition tasks $A_i$ and $A_j$ associated to the graph $G_c$ are scheduled into each other according to the edge weight.;
    Two cases are possible, if $p \geq b_i \geq b_j$ then $\sigma(A_j) = \sigma(A_i) + a_i$, and if $b_i \geq p$ then $\sigma(A_i) = \sigma(A_j) + a_i$.
    • For each edge $e = (A'_i, A''_i) \in H_c$ of the matching $M$, such that $A'_i \in G'_c$ and $A''_i \in G''_c$, the acquisition task $A_i$ associated to the graph $G_c$ is executed after the scheduling.
```

2.6. *Summary of complexity results*

We have proven the $\mathcal{NP}$-completeness of Π'_1 and Π'_2, and the polynomiality of Π'_3 and Π'_4. As we indicated in the introduction of this paper, all problems which were already $\mathcal{NP}$-complete without incompatibility constraints (see Figure 2) remain $\mathcal{NP}$-complete when G_c is introduced. For problems which were polynomial without a compatibility graph, the introduction of G_c varies the complexity for Π'_1 and Π'_2 while Π'_3 and Π'_4 stay polynomial. This leads us to conclude that the introduction of compatibility graph is an important but not deterministic factor in the complexity of coupled-task scheduling problems. Figure 7 summarizes the complexity results presented in this paper by reusing the global visualization introduced by Orman and Potts.

In the following section, we continue our analysis by proposing a polynomial-time approximation algorithm for $\mathcal{NP}$-complete problem Π'_1.

3. Approximation algorithm for problem Π'_1

This section is devoted to the study of a polynomial approximation algorithm for $\mathcal{NP}$-complete problem $\Pi'_1 : 1|a_i = b_i = p, L_i = L, G_c|C_{max}$. Note that problem Π'_2 has been studied in two respective papers [15,14].

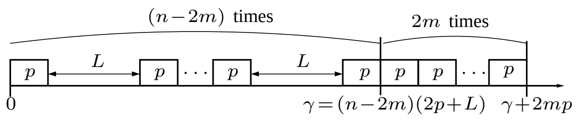


Fig. 8. Illustration of the second lower bound

We are interested in the approximation of $\mathcal{NP}$-complete problem Π'_1. Recall that we work with n acquisition tasks, and when $L \geq 2p$ the adding of the incompatibility constraint leads to the $\mathcal{NP}$-completeness of the problem. In order to achieve a schedule closest to the optimal, our research of an heuristic with non-trivial performance guarantee will focus on a study on the compatibility graph G_c. We will give two lower bounds, and an upper bound obtained by a maximal cliques cover of G_c vertices. In the following, let us call C_{max}^{opt} (resp. C_{max}^{h}) the length of an optimal schedule (resp. a schedule from our heuristic) for Π'_1.

Lemma 6 *By considering a maximum matching M of size m in G_c, our lower bound will be $C_{max}^{opt} \geq max\{2np, (n-2m)(L+2p) + 2m\}$.*

Proof.

The optimal scheduling is obtained when there is no inactivity time:

$$C_{max}^{opt} \geq \text{Sequential Time } = \sum_{i=1}^{n}(a_i + b_i) = 2np \qquad (1)$$

For the second lower bound, by considering a maximum matching M of size m in the compatibility graph, the number of isolated vertices equals $(n-2m)$. In the worst case, the optimal scheduling length needs to be superior to the scheduling length obtained by isolated vertices, which form an independent set. Furthermore, we know that a task cannot be executed entirely into another, thus we can add at least $2m$ times the execution time p of a sub-task to the scheduling length (see Figure 8). Thus, we obtain a second lower bound according to a maximum matching M of size m:

$$C_{max}^{opt} \geq (n-2m)(L+2p) + 2mp \qquad (2)$$

Therefore, according to the parameters values in our study, our lower bound will be the maximum between the two lower bounds (1) and (2):

$$C_{max}^{opt} \geq max\{2np, (n-2m)(L+2p) + 2mp\} \qquad (3)$$

■

Lemma 7 *The heuristic, based on the research of a partition of G_c by $\mathcal{K}$ maximal cliques of size less than L/p, gives an upper bound equal to $\mathcal{K}(L+p) + np$.*

Proof. The general idea consists in researching maximal cliques of size less than L/p in G_c in order to fill a maximum of slots created by the acquisition tasks. Each maximal clique is associated to the execution of a block of acquisition tasks as previously, but this time the block will not be without inactivity time. In order to compute the achieved scheduling length C^h_{max}, we sum the number of obtained blocks, which create each of them a slot of length L. We add the number of tasks to execute which represents the sequential time of all sub-tasks b_i to execute (See Figure 9).

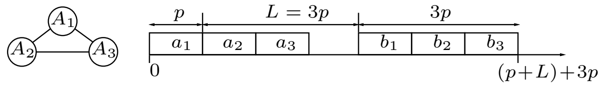


Fig. 9. Possible scheduling for a block

The obtained makespan with a vertex cover in G_c by $\mathcal{K}$ maximal cliques gives the following upper bound: supâĹŽÂl'rieure suivante :

$$C^h_{max} \leq \mathcal{K}(L+p) + \sum_{i=1}^{n} b_i = \mathcal{K}(L+p) + np \tag{4}$$

■

The relative performance ρ using this heuristic is given by Theorem 3: heuristique.

Theorem 3 *This heuristic, based on the maximal cliques covering, gives a relative performance equal to* $\rho \leq \frac{4p+L}{4p}$.

Proof. By using the obtained bounds (equations (3) and (4)), we obtain the following relative performance: performance relative suivante:

$$\rho \leq \frac{C^h_{max}}{C^{opt}_{max}} \leq \frac{\mathcal{K}(L+p)+np}{2np} \tag{5}$$

This ratio is a general result for our problem, but we can search another approach using the second lower bound with the matching. We can analyze the value of the relative performance ratio when the heuristic, used to approximate the problem, consists in finding a maximum matching M of size m. In this case, $\mathcal{K} = \boldsymbol{(n-m)}$ because the matching creates $\boldsymbol{m}$ blocks of size $(L+3p)$ and the isolated tasks form $\boldsymbol{(n-2m)}$ blocks of size $(L+2p)$. By substituting $\mathcal{K}$ in the obtained bound in equation (4), we find a new upper bound:

$$C^h_{max} \leq (n-m)(L+p) + np \tag{6}$$

From the study of the max function in the lower bound (given by equation (2)), we can analyze the behavior of the relative performance. Since $C^{opt}_{max} \geq max\{2np, (n-2m)(L+2p)+2mp\}$, following cases should be considered[2] :

- For $m \in [0, \frac{Ln}{2(p+L)}[$, $C^{opt}_{max} \geq (n-2m)(L+2p) + 2mp$
- For $m \in [\frac{Ln}{2(p+L)}, \frac{n}{2}]$, $C^{opt}_{max} \geq 2np$

According to m values, we obtain a new upper bound for our heuristic and a new lower bound for an optimal schedule (see Figure 10). When $m = \frac{Ln}{2(p+L)}$, we see in Figure 10 that the optimal ratio is obtained. The following equations give us the researched value:

$$\rho \leq \frac{C^h_{max}}{C^{opt}_{max}} \leq \frac{(n - \frac{Ln}{2(p+L)})(p+L) + np}{2np}$$
$$\rho = \frac{\frac{(2(p+L)-L)}{2(p+L)}(p+L)+p}{2p}$$
$$\rho = \frac{2p+\frac{L}{2}}{2p} = \frac{4p+L}{4p} \tag{7}$$

Note that for $m = 0$, $\rho = 1$ (obviously, since the compatibility graph is a set of independent tasks). Moreover, for $m = \dfrac{n}{2}, \rho = \frac{3p+L}{4p}$. ■

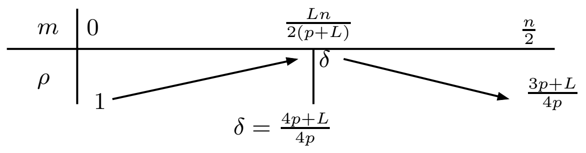


Fig. 10. Behavior of the relative performance ρ according to the value of m.

This ends the problem Π'_1 analysis. On negative side, we have shown that the problem is $\mathcal{NP}$-complete. On positive side, we gave an approximation algorithm with relative performance bounded by $\rho \leq \frac{4p+L}{4p}$, where L and p are problem parameters. The fact that the value of relative performance ρ, associated to the algorithm, depends on parameters L and p, leads to continue our work in research of approximation algorithms with a constant performance guarantee.

The approximation study of Π'_2 had been done in [14]. For this problem, we study the limit between polynomiality and $\mathcal{NP}$-completeness according to the values of parameter L when it depends on a and b.

[2] We search the value of m in order to obtain $C^{opt}_{max} \geq (n-2m)(L+2p)+2mp$ or $C^{opt}_{max} \geq 2np$.

Table 1

Problem	Complexity	Ratio.	Ref.
$\Pi'_1 : (a_i = b_i = p, L_i = L), G_c$	$\mathcal{NP}$-complete	$\frac{4p+L}{4p}$	This paper
$(a_i = L_i = b_i), G_c$	$\mathcal{NP}$-complete	$\frac{3}{2}$	[15]
$\Pi'_2 :$ $(a_i = a, L_i = L, b_i = b), G_c$	$\mathcal{NP}$-complete	No bound	This paper
$(a_i = a, L_i = L = a+b, b_i = b), G_c$	$\mathcal{NP}$-complete	$\left[\frac{3}{2}, \frac{5}{4}\right]$	[14]
$\Pi'_3 : (a_i = L_i = p, b_i), G_c$	Polynomial	1	This paper
$\Pi'_4 : (a_i, L_i = b_i = p), G_c$	Polynomial	1	This paper

Summary of results

4. Conclusion

We have studied throughout this paper the scheduling problem on single processor with coupled-tasks in presence of arbitrary compatibility graph G_c. The different sub-problems encountered arise because we vary basic parameters (a_i, L_i, b_i) of coupled-tasks in the same manner as do Orman and Potts in their paper on the study of coupled-tasks without incompatibility constraint. The goal sought throughout our paper was to determine the impact of incompatibility constraint on these problems, and to analyze critical cases located at the limit between polynomiality and $\mathcal{NP}$-completeness according to parameters value.

We have presented two $\mathcal{NP}$-completeness proofs for problems Π'_1 and Π'_2, and two polynomial proofs for problems Π'_3 and Π'_4. Figure 7 summarizes the complexity results presented in this paper. The first observation is that the introduction of incompatibility constraints has a significant impact on the complexity of some problems: e.g. problem Π_1 which was solvable in polynomial time becomes $\mathcal{NP}$-complete in the presence of compatibility graph (problem Π'_1), leading to the $\mathcal{NP}$-completeness of Π'_2 while Π_2 was still open. From these results, we deduce the $\mathcal{NP}$-completeness of all more general problems. In a second point, we have proposed a polynomial-time algorithm for problems Π'_3 and Π'_4, and show the polynomiality of more specific problems.

In a second part we have presented a polynomial approximation algorithm for Π'_1 with a performance ratio $\frac{4p+L}{4p}$, where p and L are fundamental parameters of the problem. This heuristic completes previous approximation results investigated in previous works, summarized in Table 2.

It is interesting to observe that problems complexity depends largely on the link between parameter L_i and one of the other two: a_i or b_i. If L_i is equal to a_i or b_i, the only way to schedule tasks is either to overlap them two by two, or to execute them consecutively. This configuration leads us to search a maximum matching or a perfect in compatibility graph. When L_i is independent of the other two parameters, the possible schedules of tasks lead to seek chains, or cliques in G_c, and most of these problems are known to be $\mathcal{NP}$-complete.

A general observation that we can do on the approximation of studied problems is the following: introduction of incompatibility constraint is fundamentally changing traditional approach to this kind of problem, and lead to study graph problems known to be hard to approximate. As obtained approximation bounds depend on L_i most of the time, perspectives of this work consist in determining existence or not of constant factor approximation algorithms for $\mathcal{NP}$-complete problems.

References

[1] A. A. Ageev and A. E. Baburin. Approximation algorithms for uet scheduling problems with exact delays. *Operations Research Letters*, 35(2007) 533 – 540.

[2] D. Ahr, J. BÃl'kÃl'si, G. Galambos, M. Oswald, and G. Reinelt. An exact algorithm for scheduling identical coupled-tasks. *Mathematical Methods of Operations Research*, 59(2004) 193–203.

[3] J. Blażewicz, K. Ecker, T. Kis, C.N. Potts, M. Tanas, and J. Whitehead. Scheduling of coupled tasks with unit processing times. *Technical report, Poznan University of Technology*, 2009.

[4] N. Brauner, G. Finke, V. Lehoux-Lebacque, C. Potts, and J. Whitehead. Scheduling of coupled tasks and one-machine no-wait robotic cells. *Computers & OR*, 36(2009) 301–307.

[5] J. Edmonds. Maximum matching and a polyhedron with $0, 1$ vertices. *Journal of Research the National Bureau of Standards*, 69 (1965) 125–130.

[6] H. N. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *Journal of the ACM*, 23(1976) 221 – 234.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[8] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5(1979) 287–326.

[9] A. J. Orman, C.N. Potts, A. K. Shahani, and A. R. Moore. Scheduling for a multifunction phased array radar system. *European Journal of Operations Research*, 90(1996) 13–25.

[10] A. J. Orman, A. K. Shahani, and A. R. Moore. Modelling for the control of a complex radar system. *Computers & OR*, 25(1998) 239–249.

[11] A.J. Orman and C.N. Potts. On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72(1997) 141–154.

[12] R.D. Shapiro. Scheduling coupled tasks. *Naval Research Logistics Quarterly*, 27(1980) 477–481.

[13] H. D. Sherali and J. C. Smith. Interleaving two-phased jobs on a single machine. *Discrete Optimization*, 2(2005) 348–361.

[14] G. Simonin, B. Darties, R. Giroudeau, and J.-C. KÃűnig. Isomorphic coupled-task scheduling problem with compatibility constraints on a single processor. *Journal of Scheduling*, 14(2011) 501–509.

[15] G. Simonin, R. Giroudeau, and J.-C. KÃűnig. Polynomial-time algorithms for scheduling problem for coupled-tasks in presence of treatment tasks. *Electronic Notes in Discrete Mathematics*, 32(2010) 647–654.

Received 6-9-2011; revised 20-3-2012; accepted 29-3-2012