Complexity and approximation for scheduling problem for a torpedo

G. Simonin[1], **R. Giroudeau**[1] **and J.C. König**[1]

[1]LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France, UMR 5506

ABSTRACT

This paper considers a special case of the coupled-tasks scheduling problem on one processor. The general problems were analyzed in depth by Orman and Potts [1]. In this paper, we consider that all processing times are equal to 1, the gap has exact length L, we have precedence constraints, compatibility constraints are introduced and the criterion is to minimize the scheduling length. We use this problem to study the problem of data acquisition and data treatment of a torpedo under the water. We show that this problem is $\mathcal{NP}$-complete and we propose an ρ-approximation algorithm where $\rho \leq \frac{(L+6)}{6}$.
Keywords: scheduling, coupled-tasks, compatibility constraints, complexity, approximation

1. Introduction

1.1. Presentation

In this paper, we present the problem of data acquisition according to compatibility constraints in a submarine torpedo, denoted $TORPEDO$ problem. The torpedo is used in order to make cartography, topology studies, temperature measures and many other tasks in the water. The aim of this torpedo is to collect and process a set of data as soon as possible on a mono processor. In this way, it possess few sensors, a mono processor and two types of tasks which must be schedule: Acquisition tasks and treatment tasks.

First, the acquisition tasks $\mathcal{A} = \{A_1, \ldots, A_n\}$ can be assigned to coupled-tasks introduced by Shapiro [2], indeed the torpedo sensors emit a wave which propagates in the water in order to collect the data. Each acquisition tasks A_i have two sub-tasks, the first a_i sends an echo, the second b_i receives it. For a better reading, we will denote the processing time of each sub-task a_i and b_i. Between the sub-tasks, there is an incompressible idle time L_i which represents the spread of the echo in the water.

Second, treatment tasks $\mathcal{T} = \{T_1, \ldots, T_n\}$ are obtained from acquisition tasks, indeed after the return of the echo, various calculations will be executed from gathered informations. These tasks are preemptive and have precedence constraints with the acquisition tasks. In this paper, we will study the problem where every acquisition task have a precedence relation with only one treatment task.

At last, there exist compatibility constraints between acquisition tasks, due to the fact that some acquisition tasks cannot be processed in same the time that another tasks. In order to represent this constraint, a compatibility graph $G_c = (\mathcal{A}, E_c)$ is introduced, where $\mathcal{A}$ is the set of coupled-tasks and E_c represents the edges which link two coupled-tasks which can be executed simultaneously. In other words, at least one sub-task of a task A_i may be executed during the idle time of another task A_j (see example in Figure 1).

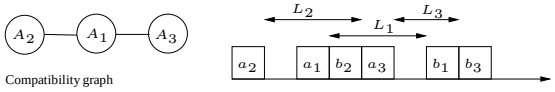

Fig. 1: Example of compatibility constraints with $L_1 = L_2 = 3$ and $L_3 = 2$

The aim of the TORPEDO problem is to produce a shortest schedule (i.e. to minimize the moment after the execution of the last task in the schedule denoted C_{max}) in which compatibility contraints between acquisition tasks and precedence constraints are respected. In scheduling theory, a problem is categorized by its machine environment, job characteristic and objective function. So using the notation scheme $\alpha|\beta|\gamma$ proposed by [3], the TORPEDO problem will be defined by $1|prec, (a_i, L_i, b_i) \cup (T_i, pmtn), G_c|C_{max}$[1].

Our work consists in measuring the impact of the compatibility graph on the complexity and approximation of scheduling problems with coupled-tasks on a mono processor. This paper is focusing on the limit between polynomial problems and $\mathcal{NP}$-complete problems, when the compatibility constraint is introduced.

1.2. Related work

The complexity of the scheduling problem, with coupled-tasks and a complete compatibility graph[2], has been investigated by Blazewicz and al. [4], Orman and Potts [1], Ahr and al. [5]. Nevertheless, in the article we study a different problem in which coupled-tasks (or acquisition tasks) must respect a compatibility graph. Morever, in our model, we consider a set of treatment tasks whose have a precedence constraint with the set of acquisition tasks, whereas in existing works the authors ([4],[1],[5]) focus their studies on precedence constraints between the acquisition tasks. By comparing the results of Orman and Potts [1] and those obtained by relaxing the constraint of compatibility, we can measure the impact of compatibility constraint on this kind of problem.

[1]prec (resp. pmtn) represents the precedence constraints between $\mathcal{A}$ et $\mathcal{T}$ (resp. the preemtivity of the treatment tasks)

[2]Notice, the lack of compatibility graph is equivalent to a fully connected graph. In this way, all tasks may be compatible each other.

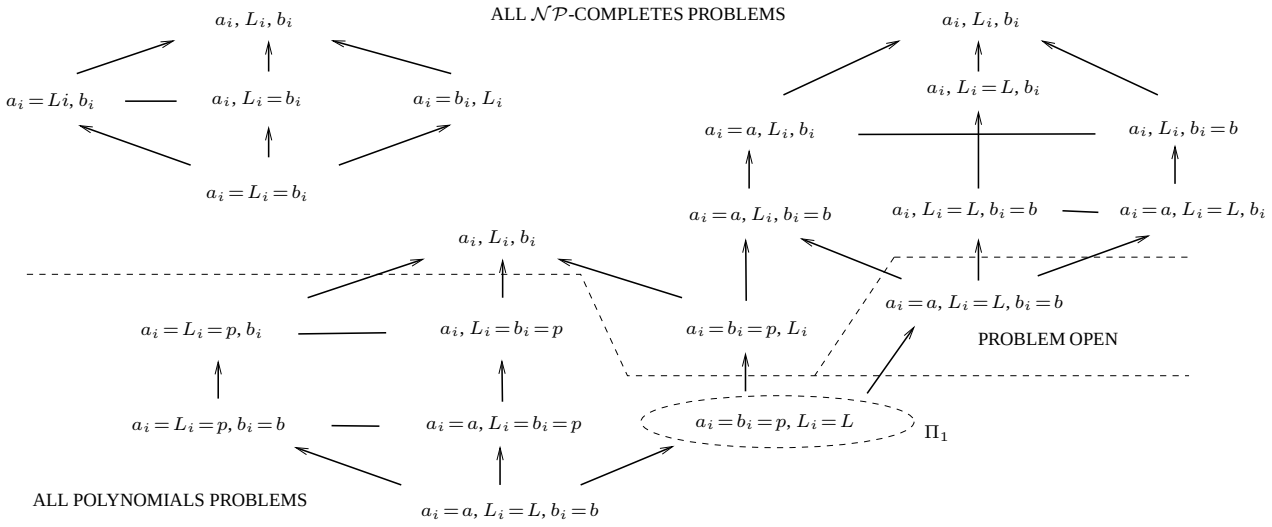

Fig. 2: Global visualisation of the complexity of scheduling problems with coupled-tasks given by Orman and Potts [1]. The graphs are described like this: a_i, L_i, b_i represents the type of problem studied, where a_i, b_i and L_i can take any value or be all equal to a constant. Finally, there is an arc from a specific problem to a more general problem, and an edge between two symmetrical problems.

We derive two main results: First, starting from the complexity results of Orman and Potts resumed in Figure 2, we show the complexity of a special problem, denoted Π_1 which becomes $\mathcal{NP}$-complete when the compatibility constraint is relaxed. The $\mathcal{NP}$-completeness of Π_1 imply the $\mathcal{NP}$-completeness of all the problems which are more general (see Figure 2). Second, we develop a polynomial-time approximation algorithm based on a maximum matching on the compatibility graph for Π_1.

This article is organized as follows: In the next section, we will prove the $\mathcal{NP}$-completeness of $\Pi_1 = 1|prec, (a_i = b_i = p, L_i = L, G_c) \cup (T_i, pmtn)|C_{max}$ where the two sub-tasks a_i and b_i are equal to a constant p and the inactivity time L_i is equal to a constant L (the reduction is based from the $\mathcal{NP}$-complete Clique, Garey and Johnson [6] GT19), and so $t_{b_i} = t_{a_i} + a_i + L_i = t_{a_i} + p + L$ where t_{a_i} (resp. t_{b_i}) is the starting time of the sub-task a_i (resp. t_{b_i}). In the last section, we develop a polynomial-time approximation algorithm for Π_1 with performance guarantee less than $\frac{L+6}{6}$.

2. Complexity result

In this section, notice that in the special case where $L = 1$, the problem is polynomial. It is sufficient to find a maximum matching in the compatibility graph. We focus the case $L \neq 1$ and L is a data of the problem. The case where $L = 2$, is studied in another paper [7]. In order to prove the $\mathcal{NP}$-completeness of Π_1, we will prove the $\mathcal{NP}$-completeness of the specific case $\Pi_2 = 1|prec, (a_i = b_i = 1, L_i = L) \cup (T_i, pmtn), G_c|C_{max}$.

Theorem 2.1 *Let n be the acquisition tasks number, the problem, to decide if an instance of the problem Π_2 has a scheduling length $C_{max} = 2n + \sum_{T_i \in \mathcal{T}} T_i$, is $\mathcal{NP}$-complete.*

Proof

Our approach is similar to the proof of Lenstra and Rinnoy Kan [3] for the problem $P|prec; p_j = 1|C_{max}$. This demonstration is based on the *Clique* decision problem (see Garey and Johnson GT19 [6]):
INSTANCE: A graph $G = (V, E)$ where $|V| = n$, and an integer K.
QUESTION: Can we find a clique of size K in G ?

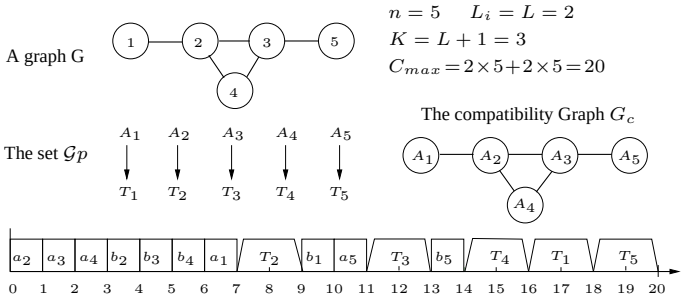

Fig. 3: Illustration of polynomial-time transformation Clique $\propto \Pi_2$

Our proof is based on the polynomial-time transformation Clique $\propto \Pi_2$. It is easy to see that the problem Π_2 is in $\mathcal{NP}$.

Let I^* an instance of Clique, we will construct an instance I of Π_2 with $C_{max} = 2n + \sum_{T_i \in \mathcal{T}} T_i$ in the following way:

Let $G = (V, E)$ a graph in the instance I, with $|V| = n$:

- $\forall v \in V$, an acquisition task A_v is introduced, composed of two sub-tasks a_v and b_v with processing time $a_v = b_v = 1$ and with a latency time L_v, between these two sub-tasks, of length $L = (K - 1)$, called $slot$.
- For each edge $e = (v, w) \in E$, there is a compatibility relation between the two acquisition tasks A_v and A_w.

- For each task A_v, we introduce a treatment task T_v which is its successor.
- Each T_v has a processing time noted $T_v = L$. Thus, the treatment tasks will replace all the inactivity slot of all the A_v after the clique.

- We suppose that there is a clique of length $K=(L+1)$ in the graph G. Let us show that there is a scheduling in $C_{max} = (2n + \sum_{T_i \in \mathcal{T}} T_i)$ units of time. For that, consider the following scheduling:
  - From time $t=0$ to $t=L$, we schedule the $K=(L+1)$ tasks which represent the vertices of the clique of size K.
  - From time $t=(2L+2)$, we schedule the $(n-K)$ remaining tasks A_v.
  - In each slot from these $(n-K)$ tasks A_v, we schedule the tasks T_v. Since each T_v has as a value $T_v = L$, by scheduling $(n-K)$ tasks T_v, we will fill each slot of length L of the $(n-K)$ tasks A_v.
  - Remeaning treatment tasks are scheduled at the end of the schedule.

  With this allocation, we fill all the slots and we give a valid scheduling in $(2n + \sum_{T_i \in \mathcal{T}} T_i)$ units of time.

- Reciprocally, let us suppose that there is a scheduling in $(2n+\sum_{T_i \in \mathcal{T}} T_i)$ units of time without inactivity time for Π_2, then let us show that the graph G contains a clique of size $K=(L+1)$.
  From these suppositions, we make essential comments:
  - With the precedence constraints between the tasks A_v and T_v, it is easy to see that we can schedule only tasks A_v at $t=0, \forall v \in V$. Thus, the first treatment task could be scheduled only starting from $t=(L+2)$.
  - Let a_{p_1} be the first sub-task of acquisition scheduled at $t=0$, with a slot of length L. We need a clique of size $(L+1)$ to obtain a scheduling without inactivity slot.

Thus we have $(L+1)$ acquisition tasks which are compatibles. And in the compatibility graph G_c, we will have an edge between each couple of these tasks A_v. Consequently, the tasks $A_{p_1}, A_{p_2}, \ldots, A_{p_L}$, associated to the vertices of the graph G, form a clique of size $K=(L+1)$.

This concludes our proof of Theorem 2.1. □

From this result we can conclude of the NP-completeness of Π_1. With the global visualization of Figure 2, we see that the $\mathcal{NP}$-completeness of Π_1 imply the $\mathcal{NP}$-completeness of all the more general problems. And thus, the open problem in the study of Orman and Potts [1] become $\mathcal{NP}$-complete with the relaxation of the compatibility constraint.

3. Approximation algorithm

In this section, we will present and study a polynomial-time approximation algorithm for Π_2 based on maximum matching.

Remark 3.1 *Notice that, in the case where processing time of treatment tasks is greater than one ($T_i > 1, \forall i$), then the sum of idle time in a schedule cannot be higher as if the processing time of treatment tasks is one ($T_i = 1$).*

In the following, treatment tasks will have processing time equal to $T_i = 1$. We will present an approximation algorithm of the problem Π_2.

3.1. Lower bounds

We will give two lower bounds. For the first, optimal scheduling is taken where we do not have any time of inactivity. Moreover we know that the number of treatment tasks is equal to the number of acquisition tasks and that in worst case all the treatment tasks have an processing time $T_i = 1, \forall i$. Thus, we have:

$$C^{opt}_{max} \geq T_{seq} = 2n + \sum_{T_i \in \mathcal{T}} T_i \geq 2n + n = 3n \qquad (1)$$

For the second bound, the maximum matching is taken of the compatibility graph G_c, its cardinality is m, and thus we have $(n-2m)$ independent vertices. In worst case, optimal scheduling is greater than independent vertices scheduling with the last treatment task. And so we have:

$$C^{opt}_{max} \geq (n-2m)(L+2)+1 \qquad (2)$$

For our study, our lower bound will be

$$C^{opt}_{max} \geq max\{3n, (n-2m)(L+2)+1\} \qquad (3)$$

3.2. Upper bound

Algorithm 1: A polynomial-time approximation algorithm

Instance: $\mathcal{A}$, $\mathcal{T}$, G_c, $L \geq 1$
Result: C^h_{max}
begin
- Compute a maximum matching of G_c
- For each edge (i,j) of the maximum matching, the acquisition tasks A_i and A_j are scheduled such that $t_{a_j} = t_{a_i} + 1$
- For each vertex i remaining, we schedule the acquisition task A_i
- Allocate treatment tasks to the first free slot by respecting the precedence constraints

end

We will give some essential remarks on the structure of the scheduling given by our approximation algorithm. Let us suppose that we have a scheduling given by the approximation algorithm with a maximum matching of size m.

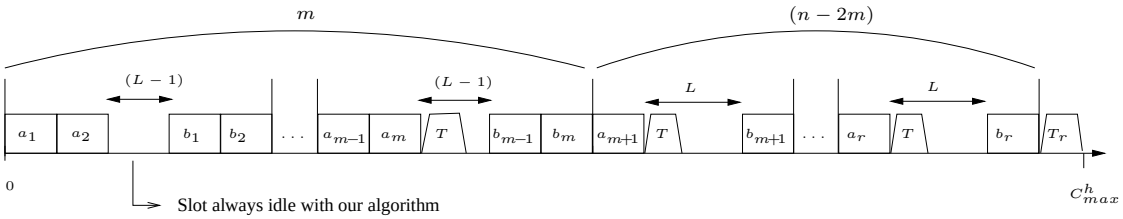

Fig. 4: Illustration of the approximation algorithm

- In the first coupled-task matched, there is an incompressible latency length of size $(L-1)$.
- We have n acquisition tasks, the scheduling length of these tasks is $2n$.
- For two tasks matched, the incompressible latency length is $(L-1)$.
- For each remaining vertex, incompressible latency length is L.
- Considering the last acquisition task A_r. After its execution, we may process one treatment task denoted by T_r (this case occurs when all the treatment tasks, except T_r, are scheduled before the completion time of A_r), or some treatment tasks (this case occurs when there is no idle time before the completion time of A_r). See figure (4), for an illustration of the case where the task T_r is the only treatment task executed after the completion time of A_r.
  So, the number of treatment task executed after A_r is:

$$\begin{aligned} &max\{n-(m-1)(L-1)-(n-2m)L, 1\} \\ &\quad = max\{n-nL+m(L+1)+L-1, 1\} \end{aligned} \quad (4)$$

Finally, our upper bound will be:

$$\begin{aligned} &C^h_{max} \leq T_{\text{execution}} + T_{\text{incompressible latency length}} \\ &\leq [2n + max\{n-nL+m(L+1)+L-1, 1\}] \\ &+ [(L-1)m + L(n-2m)] \\ &\leq [2n + max\{n-nL+m(L+1)+L-1, 1\}] \\ &+ [L(n-m)-m] \end{aligned}$$

3.3. Relative performance

In the first step, the Tables 1 and 2 give a summarize of the ratio of relative performance $\rho \leq \frac{C^h_{max}}{C^{opt}_{max}}$ for $L \in \{1,2,3\}$.

Now we focuses on study for $L \geq 4$. Since that $m \leq \frac{n}{2}$, it easy to see that $max\{n-nL+m(L+1)+L-1, 1\} = 1$. And so $C^h_{max} \leq 2n + L(n-m) - m + 1$.

Moreover, since that $C^{opt}_{max} \geq max\{3n, nL + 2n - 2mL - 4m + 1\}$, the following cases must be considered:

- For $m \in [0, \frac{n(L-1)+1}{2(L+2)}[$, it is easy to see that $C^{opt}_{max} \geq 3n$.
- For $m \in [\frac{n(L-1)+1}{2(L+2)}, \frac{n}{2}]$, we will have $C^{opt}_{max} \geq nL + 2n - 2mL - 4m + 1$.

According to the values of m, we give the upper bound for the length of the scheduling proposed by the heuristic h,

	$\alpha = 1$	$L = 2$
C^h_{max}	$3n$	if $m \geq \frac{n}{3}$, then $C^h_{max} \leq 3n+1$ if $\frac{n+1}{8} \leq m < \frac{n}{3}$, then $C^h_{max} \leq 4n-3m+1$ if $m < \frac{n+1}{8}$, then $C^h_{max} \leq 4n-3m+1$
C^{opt}_{max}	$3n$	if $m \geq \frac{n}{3}$, then $C^{opt}_{max} \geq 3n$ if $\frac{n+1}{8} \leq m < \frac{n}{3}$, then $C^{opt}_{max} \geq 3n$ if $m < \frac{n+1}{8}$, then $C^{opt}_{max} \geq 4n-8m+1$
$\boldsymbol{\rho}$	1	if $m \geq \frac{n}{3}$, then $\boldsymbol{\rho \leq 1 + \frac{1}{3n}}$ if $\frac{n+1}{8} \leq m < \frac{n}{3}$, then $\boldsymbol{\rho \leq \frac{4}{3} + \frac{1}{3n}}$ if $m < \frac{n+1}{8}$, then $\boldsymbol{\rho \leq \frac{11}{6}}$

Tab. 1: Relative performance for $L \in \{1,2,3\}$

	$L = 3$
C^h_{max}	if $m = \frac{n}{2}$, then $C^h_{max} \leq 3n+2$ if $\frac{2n+1}{10} \leq m < \frac{n}{2}$, then $C^h_{max} \leq 5n-4m+1$ if $m < \frac{2n+1}{10}$, then $C^h_{max} \leq 5n-4m+1$
C^{opt}_{max}	if $m = \frac{n}{2}$, then $C^{opt}_{max} \geq 3n$ if $\frac{2n+1}{10} \leq m < \frac{n}{2}$, then $C^{opt}_{max} \geq 3n$ if $m < \frac{2n+1}{10}$, then $C^{opt}_{max} \geq 4n-8m+1$
$\boldsymbol{\rho}$	if $m = \frac{n}{2}$, then $\boldsymbol{\rho \leq 1 + \frac{2}{3n}}$ if $\frac{2n+1}{10} \leq m < \frac{n}{2}$, then $\boldsymbol{\rho \leq \frac{5}{3} + \frac{1}{3n}}$ if $m < \frac{2n+1}{10}$, then $\boldsymbol{\rho \leq \frac{21}{17}}$

Tab. 2: Relative performance for $L \in \{1,2,3\}$

and the lower bound for an optimal scheduling (see illustration figure 5).

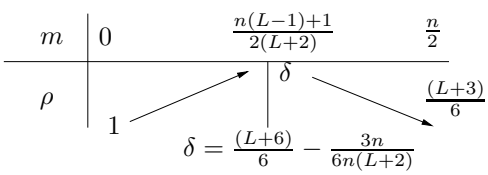

Fig. 5: Behavior of the relative performance ρ as a fonction of m

Notice that for $m = 0$, $\rho = 1$ (it is clear, because the compatibility graph is an independent set), moreover for $m = \frac{n}{2}, \rho = \frac{(L+3)}{6}$.

4. Conclusion

In this paper, we presented a scheduling problem on mono processor with graph constraints and coupled-tasks. On the negative side, we showed that the problem Π_2 is $\mathcal{NP}$-complete, our proof is based on the polynomial-time transformation Clique to Π_2, and imply the $\mathcal{NP}$-completeness of all the more general problems (specially for the open problem in Figure 2 which become $\mathcal{NP}$-complete with the relaxation of the compatibility constraint).

On the positive side, we gave an approximation algorithm for Π_2 with relative performance bounded by $\rho \leq \frac{L+6}{6}$ in the worst case, where L is the inactivity time of acquisition tasks. The relative performance value ρ associated to the algorithm depends on the parameter L, which is one of the problem data. This remark brings a fundamental question: "Is that our problem admits an approximation algorithm with a performance guarantee equal to a constant value?".

REFERENCES

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

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

[3] 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*, 5:287–326, 1979.

[4] J. Blazewicz, K.H. Ecker, T. Kis, and M. Tanas. A note on the complexity of scheduling coupled-tasks on a single processor. *Journal of the Brazilian Computer Society*, 7(3):23–26, 2001.

[5] D. Ahr, J. Békési, G. Galambos, M. Oswald, and G. Reinelt. An exact algorithm for scheduling identical coupled-tasks. *Mathematical Methods of Operations Research*, 59:193–203(11), June 2004.

[6] 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, 1990.

[7] G. Simonin, R.Giroudeau, and J.-C. König. Extended matching problem for a coupled-tasks scheduling problem. 2009. to appear in TMFCS, Orlando, Florida.