Polynomial-Time Algorithms for Scheduling Problem for Coupled-Tasks in Presence of Treatment Tasks

Gilles Simonin, Rodolphe Giroudeau, Jean-Claude König

Polynomial-time algorithms for scheduling problem for coupled-tasks in presence of treatment tasks

G. Simonin, R. Giroudeau and J.-C. König [1,2]

LIRMM, 161 rue Ada,34392 Montpellier Cedex 5, France, UMR 5056

Abstract

We consider the problem to schedule n coupled-tasks in presence of treatment tasks. This work is motivated by the problem of data acquisition for a torpedo. In such context, we developp a $O(nlog(n))$ polynomial-time algorithm for a specific coupled-tasks scheduling problem.

Keywords: polynomial-time algorithm, scheduling, coupled-tasks, matching

1 Introduction

In this paper, we present the problem of data acquisition according to incompatibility constraints in a submarine torpedo. 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 possesses few sensors, a mono processor and two types of tasks which must be scheduled: Acquisition and Treatment tasks. First, the acquisition tasks $\mathcal{A} = \{A_1, \ldots, A_n\}$ can be assigned to coupled-tasks introduced by [6], indeed the torpedo sensors emit a wave which propagates in the water in order to collect the data. Each acquisition task A_i has two sub-tasks, the first a_i represents the send of an echo, and b_i its response. 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

[1] Email: gilles.simonin@lirmm.fr
[2] Email: rgirou@lirmm.fr, konig@lirmm.fr

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 information. 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 (a T_i after a A_i).

At last, there exists incompatibility constraints between acquisition tasks, due to the fact that some acquisition tasks cannot be processed at the same time as another task. 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 incompatibility constraints (i.e. the ability to perform two coupled-tasks into each other). At least one sub-task of A_i may be executed during the idle time of another task A_j (see example in Figure 1).

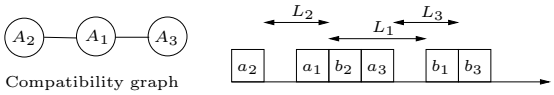


Figure 1. Example of incompatibility constraints with $L_1 = 3$, $L_2 = L_3 = 2$

The aim of this problem is to produce a shortest schedule (i.e. to minimize the completion time of the last processed task) denoted by C_{max}, in presence of precedence constraints between acquisition tasks and treatment tasks. 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 [4], this problem will be defined as $1|prec, (a_i, L_i, b_i) \cup (\tau_i, pmtn), G_c|C_{max}$. [3].

Our work consists in measuring the impact of the treatment tasks on the complexity and approximation of scheduling problems with coupled-tasks on a mono processor. This paper is focusing on the limit between the polynomiality and the $\mathcal{NP}$-completeness of our problem, when the treatment tasks are introduced.

The complexity of the scheduling problem, with coupled-tasks and a complete compatibility graph [4], has been investigated first by [5], [3], [2] and

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

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

[1]. Nevertheless, in this article we study a different problem in which treatment tasks are introduced. By comparing the results of [5] and our results, we can measure the impact of the treatment taks on this kind of problem. In such context, several results [7] have been recently obtained according to coupled-tasks parameters. In the following, we consider the specific problem $\Pi = 1|prec, (a_i = L_i = p, b_i) \cup (\tau_i, pmtn), complete - G_c|C_{max}$, where $p \in \mathbb{N}^*$. We give a polynomial-time algorithm in order to solve the problem Π.

2 Computational complexity

In this section we consider the following assumption: Let $K = \{A_i = (a_i, L_i, b_i) | b_i \leq p\}$ and $S = \mathcal{A} \backslash K$ be two sets of tasks. It is clear that if A_i and $A_j \in S$ then the edge $(A_i, A_j) \notin G_c$. Hereafter, G_c is called a complete graph if and only if the set of tasks K induces a clique, the set of tasks S induces an independent set and $\forall x \in K, \forall y \in S$, we have $\{x, y\} \in G_c$.

2.1 $\Pi : 1|prec, (a_i = L_i = p, b_i) \cup (\mathcal{T}_i, pmnt), complete - G_c|C_{max}$

We add the treatment tasks τ_i^K (resp. τ_i^S) for a task $A_i = (a_i = L_i = p, b_i) \in K$ (resp. $A_i = (c_i = L_i = p, d_i) \in S$) in this problem. Let O be a schedule. We deduce from O a matching M of G_c in follows: if A_i and A_j are overlapped then $\{i, j\} \in M$. If the subtask a_i is executed before a_j, A_i (resp. A_j) is called the first-task (resp. second-task) of the pair. Notice that a first-task is an element of K. The edges of M are partitioned into two subsets: $K - K$ or $K - S$. The tasks, which are not matched, are denoted as isolated tasks.

We can suppose without lost of generality that there exists an optimal solution for which the followings proprieties are true:

Lemma 2.1 *We may suppose that if $\tau_i^S > \tau_j^S$ then S_i is processed before S_j.*

Proof A contrario, we have $\tau_i^S < \tau_j^S$. By swapping the two tasks of S, the length may be decreased (see Figure 2).

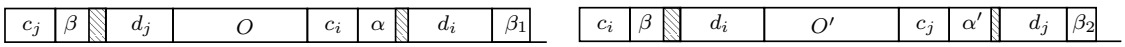


Figure 2. First and second configuration by swapping

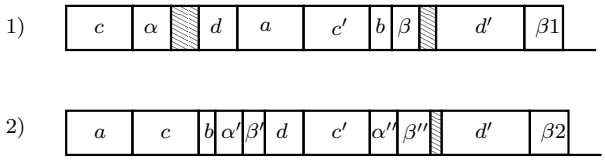
□

In the same way as previously, we suppose that if K_i and S_j are second-tasks, and if $\tau_i^K > \tau_j^S$ (resp. $\tau_j^S > \tau_i^K$) then K_i is processed before S_j (resp.

S_j before K_i) as second-task[5]. Moreover, if K_i and K_j are two first-tasks and if $b_i + \tau_i^K > b_j + \tau_j^K$, thus we may suppose that K_i is processed before K_j as first-task.

Lemma 2.2 *We may suppose that the isolated tasks of S are processed at the end of the schedule.*

Proof A contrario, we can suppose that a task S_1 is executed before $K - S_2$. Let τ_1^S (resp. τ_2^S) be the duration of the treatment of S_1 (resp. S_2)



$\tau_1^S \geq \tau_2^S$ from Remark 2.1

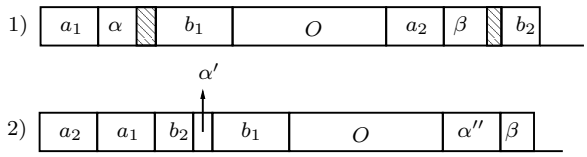
Figure 3. Second configuration where $\alpha' = min(p - b, \alpha)$ (resp. $\beta' = min(p - b, \beta)$) is the sequential time in the first slot, and $\alpha'' = \alpha - \alpha'$ (resp. $\beta'' = \beta - \beta'$) is the sequential time not executed in the first slot

We have $\boldsymbol{C_{max}(\text{case1}) = 2p + d + 3p + d' + \beta_1}$. By swapping the edge $K - S_2$ by $K - S_1$, we obtain $\boldsymbol{C_{max}(\text{case2}) = 3p + d + 2p + d' + \beta_2}$.

If $\alpha'' = 0 : \beta' = 0$, $\beta'' = \beta$ and so $\beta_2 = \tau_2^S$. Since $\beta_1 \geq \tau_2^S$, we have $\beta_2 \leq \beta_1$. Else $\alpha' = p - b \Rightarrow \beta' = 0, \alpha'' = \alpha - \alpha' = (\alpha - p) + b \Rightarrow \alpha'' \leq b \Rightarrow \beta'' \geq \beta \Rightarrow \beta_2 \leq \beta_1$. □

Lemma 2.3 *There exists a maximal (resp. maximum) matching M associated to an optimal solution for any graph G_c (complete graph G_c).*

Proof First, we give the maximal matching result. A contrario, we have two isolated tasks A_1 and A_2 with $(A_1, A_2) \in E$ (remark: $min(b_1, b_2) \leq p$).



(Remark $min(b_1, b_2) \leq p$)

with A_1-A_2

Figure 4. First and second configuration

From the first (resp. second) configuration (see Figure 4), we have $C_{max}(case1) = 4p + |O| + b_1 + b_2$ (resp. $C_{max}(case2) = 3p + |O| + b_1 + |\alpha''| + |\beta|$). Let $D = C_{max}(\text{case1}) - C_{max}(\text{case2}) = p + b_2 - \beta - \alpha''$. We have two cases, first $\alpha'' = 0 \Rightarrow \alpha = \alpha'$: since $\beta \leq p$, then $D \geq 0$. Second case, $\alpha' = p - b_2 \Rightarrow$

[5] We have the same result with two second tasks of K : K_i and K_j.

$\alpha'' = \alpha - \alpha' = \alpha - p + b_2$. We obtain $D = p + b_2 - \beta - \alpha + p - b_2 = 2p - \beta - \alpha$, or $\alpha \leq p$ and $\beta \leq p$, so $D \geq 0$. This finishes the first proof.

Now, we give the maximum matching result. We consider a maximal matching (not a maximum matching) that leads an optimal solution. Thus, there exists two isolated tasks processed. These tasks are in S (maximal matching), and at least two K-tasks are matched each other (maximum matching). From the previous results, we suppose that between the two isolated S-tasks and the nearest pair $K-K$, there exists a block, denoted by O, composed by k edges of type $K-S$, $k \geq 0$ (see Figure 5). Let α_i be the slots created by the $k+1$ pairs and the two isolated tasks numbered by 1 to $k+3$.

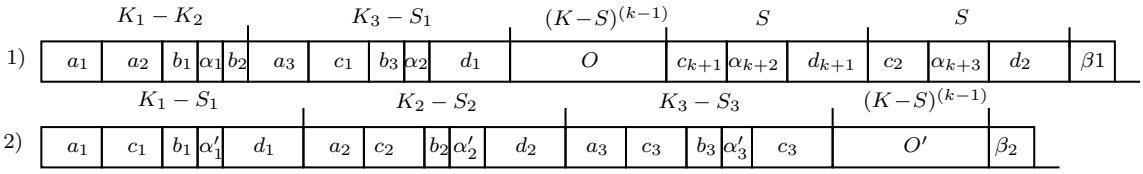


Figure 5. First and second configuration

We have $C_{max}(case1) = T + p + b_2 + \beta_1$, where $T = 3(k+2)p + \sum_{i=1}^{k+2} d_i$. We transform the first configuration $((K-S)^{k+1}\ S\ S\ \beta_2)$ to the second configuration $((K-S)^{k+2}\ \beta_2)$. By hypothesis, the length of the schedule cannot be better.

Let α'_i be the slots created by the k+2 pairs numbered by 1 to k+2. The sequential time is $C_{max}(case2) = T + \beta_2$. Therefore, $C_{max}(case2) - C_{max}(case1) = \beta_2 - p - b_2 - \beta_1$. Moreover, we have $\alpha_1 = \alpha'_1$, $\beta_1 \geq \tau^S_{k+2}$.

We suppose that the tasks are sorted in following way: $b_1 + \tau^K_1 \geq b_2 + \tau^K_2 \geq b_3 + \tau^K_3 \geq \ldots \geq b_{k+2} + \tau^K_{k+2}$ and $\tau^S_1 \geq \tau^S_2 \geq \tau^S_3 \geq \ldots \geq \tau^S_{k+2}$.

Several cases exist:

(i) there is no idle time in α'_i $(i > 2)$, so $\exists i > 2$ such that $\alpha'_i < p - b_i$ $\Rightarrow b_i + \tau^K_i + \tau^S_{i-1} < p \Rightarrow \beta_2 = \tau^S_{k+2}$ since $\forall j > i$, $b_j + \tau^K_j + \tau^S_{j-1} < p$; therefore there exists an idle slot $\Rightarrow \beta_2 \leq \beta_1 \Rightarrow C_{max}(\text{case2}) \leq C_{max}(\text{case1})$

(ii) $\forall i > 2$ $\alpha'_i = p - b_i$ and $\alpha'_2 = p - b_2$ then the number of idle time in the second configuration is at most the number of idle time in the first configuration $C_{max}(\text{case2}) \leq C_{max}(\text{case1})$. Indeed, in the second configuration the processor rate is 100% from the time $3p$ (the unique idle time in the second configuration occurs when $b_1 + \alpha_1 < p$, but this idle time occurs too in the first configuration).

(iii) there is no idle time excepted in α'_2, $\forall i > 2$ $\alpha'_i = p - b_i$ et $\alpha'_2 < p - b_2$,

then $\beta_2 \leq \beta_1 + p - (b_2 + \alpha'_2) \leq \beta_1 + p \Rightarrow C_{max}(\text{case2}) \leq C_{max}(\text{case1})$

In the three cases, the second configuration is equivalent or better than the first configuration. □

Theorem 2.4 *The problem admits a polynomial-time algorithm with complexity* $O(nlog(n))$.

Proof

Using the previous discussion, we design a polynomial-time algorithm. From the Lemma 2.3, we know that it is sufficient to find a maximum matching. We may suppose that a maximum matching leads to an optimal solution. We have the following relations:

- $b_1 + \tau_1^K \geq b_2 + \tau_2^K \geq b_3 + \tau_3^K \geq \ldots \geq b_{k+2} + \tau_{k+2}^K$
- $\tau_1^S \geq \tau_2^S \geq \tau_3^S \geq \ldots \geq \tau_{k+2}^S$

- If $|K| \leq |S|$, the tasks A_i of K (resp. of S) are sorted in non-increasing order according to $b_i + \tau_i$ (resp. τ_i). It is sufficient to matched the first element of K with the first element of S. When $K = \varnothing$, it is sufficient to schedule sequentially the S-tasks remaining.
- If $|K| > |S|$, without lost of generality, we suppose in follows that $|V| = K + S = 2n$. By Lemma 2.3, there exists a perfect matching giving an optimal solution. Let P the set of n first-tasks, it is easy to create a better schedule:

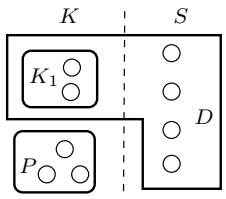


Figure 6. Illustration of the partition of G_c

The algorithm consists first in sorting in non-increasing order the n first-tasks according to $b_i + \tau_i$, and next in sorting the n second-tasks D according to τ_i. We obtain $P_1, P_2, \ldots, P_n$ (resp. $D_1, D_2, \ldots, D_n$)). We execute $\{P_1, D_1\}$, after $\{P_2, D_2\}$ etc ...

We can deduce the length of the schedule according to the type of schedule:

(i) if there exists an idle time after the first slot, the length is $3pn + \sum_{A_i \in D} b_i + \tau'_n$, where τ'_n is the execution time of the treatment task D_n.

(ii) if there exists an idle time in the first slot, the length is $\sum_{A_i \in D \cup P}(b_i + \tau_i) + (2n+1)p - (b_1 + \tau_1)$, where $(b_1 + \tau_1)$ concerns P_1,

(iii) if there is no idle time: the algorithm is optimal and the length is $\sum_{A_i \in D \cup P}(b_i + \tau_i) + 2pn$.

We have $S \subseteq D$. We denote by K_2 the set of $n - |S|$ tasks of K which minimizes b_i. Consider $D = S \cup K_2$ and $P = K - K_2$. First, we have three types of schedule: $min\{\sum_{A_i \in D} b_i\}$, $(b_1 + \tau_1)$ et τ'_n.

This algorithm allows to obtain a scheduling with $\min\{\sum_{A_i \in D} b_i\}$. According to the type of scheduling (i), if the task, with the minimum τ_i, is in D and if there exists an idle slot after slot 1, the algorithm is necessarily optimal. Indeed $\sum_{A_i \ inD} b_i$ is minimal over all possible D and τ'_n also.

Otherwise, we suppose that the task, with the minimum τ_i, is in P and that there exists on the slot n an idle time $\gamma > 0$. The only possibility to improve the schedule is to reduce τ'_n. Let X be the set of tasks P such that $\tau_i \leq \tau'_n$. Consider the task A_i which maximizes b_i in K_2, and $\forall A_j \in X$ we exchange A_i with A_j. We kept the best scheduling (it may be that one we had already). Note that this scheduling allows to obtain a scheduling with $\min\{\sum_{A_i \ inD} b_i + \tau_i\}$.

If the best scheduling possess on the slot n an idle time $\gamma > 0$, then the scheduling is optimal. Otherwise, we may be in the scheduling type (ii), and the only way to improve the scheduling would be to reduce the inactivity of the first slot. Let Y the set of tasks K_2 such that $\tau'_i + b_i > \tau_1 + b_1$ and A_k the task of P that minimizes b_i. $\forall A_j \in Y$ we exchange A_k with A_j. We kept the best scheduling (it may be that one we had already) is necessarily optimal.

For the complexity of this algorithm, it depends essentially of the first matching. Thus the algorithm gives an optimal solution in time $O(n \log(n))$.

□

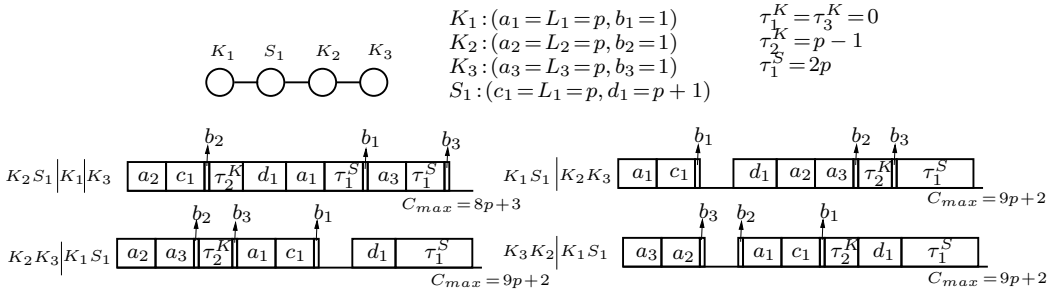


Figure 7. Counter-example

If G_c is relaxed, there exists an instance for which a maximum matching does not lead to an optimal solution (see Figure 7).

3 Conclusion

In this article, we design a $O(nlog(n))$ polynomial-time algorithm for coupled-tasks scheduling problem in presence of treatment tasks. This work is motivated by the problem of data acquisition for a torpedo. Moreover, we exhibit an example, for a non-complete compatibility graph, for which a maximum matching does not lead to an optimal solution. Thus, we show by exhibit an example the limit of using maximum matching for solving coupled-tasks scheduling problem.

References

[1] 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.

[2] P. Baptiste. A note on scheduling identical coupled tasks in constant time. *Discrete Applied Mathematics*, Accepted Pending Minor Revision, 2009.

[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] 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.

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

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

[7] G. Simonin, R.Giroudeau, and J.-C. König. Complexity and approximation for scheduling problem for a torpedo. *CIE'39, Troye, France*, July 2009.