A Note on Scheduling Identical Coupled Tasks in Constant Time

Philippe Baptiste

LIX, UMR CNRS 7161, École Polytechnique, F-91128 Palaiseau

Abstract

The coupled tasks problem consists in scheduling n jobs on a single machine. Each job i is made of two operations with processing times a_i and b_i and a fixed required delay L_i between them. Operations cannot overlap in time but operations of different jobs can be interleaved. The objective is to minimize the makespan of the schedule. In this note we show that the problem with identical jobs ($\forall i, a_i = a, b_i = b, L_i = L$) can be solved in constant time when a, b, L are fixed. This problem is motivated by radar scheduling applications where tasks corresponding to transmitting radiowaves and listening to potential echoes are coupled.

Key words: Scheduling, Coupled Tasks, Radar

1 Introduction

A radar is a system using radiowaves to detect the presence of objects in a given domain. It can also compute the range as well as the relative radial velocity of these objects. Most radars consist of a transmitter, a single antenna and a receiver. The *transmitter* generates radiowaves which are sent out in a narrow beam by the antenna in a specific direction. Objects located in the beam intercept this signal and scatter back the energy in all directions. A portion of this energy is scattered back to the *receiver* of the radar listening to all potential echoes. See [7] and [12] for a detailed description of (airborne) radars.

There are many interesting combinatorial optimization problems related to radar management. Barbaresco [3] as well as Winter and Baptiste [13] study real-time scheduling of airborne radars: Such radars have to search, track and identify potential targets. The waveforms of these tasks are most often incompatible and hence, cannot be processed simultaneously. Moreover, these tasks are repeated several times in a cyclic fashion. Altogether, this defines a

complex scheduling problem that impacts a lot on the quality of the radar's output.

In this paper we study an offline problem that consists in interleaving the tasks corresponding to receiving and sending data [11,10,6]. More formally, the coupled tasks problem consists in scheduling n jobs on a single machine. Each job i is made of two operations (O_{i1}, O_{i2}) with integer processing times a_i and b_i and a fixed required integer delay L_i between them. Operations cannot overlap in time and the objective is to minimize the makespan of the schedule. A large amount of research has been carried on this problem, including heuristics for the online version of the problem [2,5] or branch and bound. Orman and Potts [10] have solved almost all complexity issues related to this problem, except one remaining open question where all jobs are identical: $\forall i, a_i = a, b_i = b, L_i = L$. Without loss of generality, we can assume that $a \geq b$. Generalization of the problem with strict precedence constraints and unit processing times is known to be hard [4]. Ahr and others [1] have described an exact algorithm for this problem with time complexity $O(nr^{2L})$ where $r \leq \sqrt[a-1]{a}$. The algorithm is linear in the number of jobs for fixed L. Still this algorithm is not polynomial in the input size and the initial question remains open. Recently Vassilissa Lebacque [8] has introduced some nice conjectures on optimal solutions when $n \to \infty$.

In this note we prove that there is an optimal schedule in which all starting times are integral (Section 2). Although the result is not surprising, this question was not explicitly addressed in the literature while most papers rely on this assumption. We also show (Section 4) that the problem with identical jobs ($\forall i, a_i = a, b_i = b, L_i = L$) can be solved in constant time when a, b, L are fixed. Our proof is based on some basic observations of the $O(nr^{2L})$ algorithm described in [1]. For a seek of completeness, we recall the basic results and notation used in this paper (Section 3).

2 Integrality of solutions

We consider the problem with arbitrary processing times and we prove that there is an optimal schedule in which all starting times are integral. Although the result is not surprising, this question was not explicitly addressed in the literature while most papers rely on this assumption.

Consider an optimal solution and let us assume that it is not integral. Without loss of generality, we can assume that O_{n2} is the last operation. For all operation O_{ij} $(1 \leq i \leq n, 1 \leq j \leq 2)$ except the last one, let $\nu(i, j)$ denote the couple (i', j') that is immediately scheduled after O_{ij} in the schedule.

Now consider the following LP where S_{ij} denotes the starting time of O_{ij}.

$$\min S_{n2}$$
$$\begin{cases} S_{i2} = S_{i1} + L_i + a_i & 1 \leq i \leq n \\ S_{i1} + a_i \leq S_{\nu(i,1)} & 1 \leq i \leq n \\ S_{i2} + b_i \leq S_{\nu(i,2)} & 1 \leq i \leq n-1 \end{cases}$$

Any solution of the LP corresponds to a feasible schedule with the same sequence of operations as in the initial one. Moreover there is a feasible solution of the LP (starting times of the initial schedule). Finally, note that the LP has exactly two variables per constraint, with respective coefficients +1 and -1. Hence the matrix is totally unimodular and therefore there is an optimal integral solution.

3 Patterns and Graph Model [1]

Patterns consist of 0's and 1's indicating if the machine is idle or busy during some time slot. A $P(a, b, L)$ pattern is a sequence of L 0-1 in which 1 are only in blocks of length b and where each such block is followed by at least $a - b$ 0's. As shown by Ahr et al [1], the total number of possible $P(a, b, L)$ patterns is at most $a^{\frac{L}{a-1}}$.

As stated in [1], a schedule can be seen as a list of consecutive $P(a, b, L)$ patterns:

> Suppose we have started exactly k jobs and the schedule has the property, that no new job can be started before the first task of the last job. This means that job $k + 1$ can only be started after the start of the last job, possibly before but in any case after its second task. The starting time of the new job depends on the idle time periods between the two tasks of the last job.

Consider for instance the (optimal) schedule described in Figure 1 for $n = 15$. To simplify notation the jobs are identified by `A`, `B`, ..., `O`. The first line is the schedule itself: At each time slot the job being processed (first or second operation) is displayed. When the machine is idle, the time slot is identified by a "`-`". The patterns are provided under each column that corresponds to the starting time of a job. For instance, the pattern corresponding to the starting time of job `J` is $(1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0)$.

This leads immediately to a dynamic programming algorithm. Alternatively, this is a special shortest path problem in a valuated graph $G = (V, A)$ where

V is the set of all possible $P(a, b, L)$ patterns. There is an edge $(p, q) \in A$ if the "distance" from p to q is finite (in this case the valuation of the edge is the distance itself).

Given a pattern p, we denote by $p[i]$ the ith value in the sequence p. We can then define the "distance" from two patterns p to q as the smallest value $x \geq a$ such that

$$\begin{cases} \forall 0 \leq i < L - x, q[i] = p[i + x] \\ \forall x - a \leq i < x, p[i] = 0 \end{cases}$$

If no such x exists, the distance is ∞. The distance between two patterns can be computed in $O(L^2)$ as the length of the patterns is L and testing whether the above relation holds for a given x can be done in linear time.

Considering the first schedule of Figure 1, the distance between the patterns associated to the starting time of jobs `E` and `F` is 12; the distance between the patterns associated to the starting time of jobs `N` and `O` is 5.

Any left shifted schedule corresponds exactly to a path with n vertices in G and hence, an optimal schedule is a shortest (in term of distance) path containing n vertices.

As stated in [1], the total number of patterns $|V|$ is less than $O(a^{\frac{L}{a-1}})$

4 Compact Representation of Paths

In this section we prove some dominance properties of optimal paths and we show that the problem can be solved in constant time when a, b, L are fixed.

In an elementary cycle $[x, \sigma, x]$, there are no repeated vertices except x (the initial and terminal one). A *dominant* path is a path of n vertices in which all *identical elementary cycles are consecutive.*

Lemma 1 *There is an optimal dominant path.*

PROOF. Given a path p, and an integer $k \leq |p|$, we denote by $p_{\leq k}$ (respectively $p_{>k}$) the subpaths that respectively consists of the k first (respectively $|p| - k$ last) vertices of p. Now consider the largest value z for which there is an optimal path Π such that all identical elementary cycles of $\Pi_{\leq z}$ are consecutive.

If $z = n$ then our claim obviously holds. Now assume that $z < n$. The definition of z ensures that $\Pi_{\leq z+1}$ ends with an elementary cycle, say $[x, \sigma, x]$ where σ is

```
n = 15, a = 5, b = 2, L = 19, OPT = 128
AAAAA--BBBBBCCCCC--DDDDDAAEEEEEBB---CCFFFFFDDGGGGGEEHHHHHIIIIIFFJJJJJGGKKKKKHH---IILLLLLJJMMMMMKKNNNNNOOOOOLL-----MM-----NN---OO
0     0    0     1     1          1     1     0    1     1     1          1     1     0    1
0     0    0     1     1          1     1     0    1     1     1          1     1     0    1
0     0    0     0     0          0     0     0    0     0     0          0     0     0    0
0     0    0     0     0          0     0     0    0     0     0          0     0     0    0
0     0    0     0     0          0     0     0    0     0     0          0     0     0    0
0     0    0     0     1          0     0     1    0     0     1          0     0     1    0
0     0    0     0     1          0     0     1    0     0     1          0     0     1    0
0     0    1     1     0          1     0     0    1     1     0          1     0     0    1
0     0    1     1     0          1     0     0    1     1     0          1     0     0    1
0     0    0     0     0          0     0     0    0     0     0          0     0     0    0
0     0    0     0     0          0     0     0    0     0     0          0     0     0    0
0     0    0     0     0          0     0     0    0     0     0          0     0     0    0
0     1    0     1     1          0     1     1    0     1     1          0     1     1    0
0     1    0     1     1          0     1     1    0     1     1          0     1     1    0
0     0    1     0     0          0     0     0    1     0     0          0     0     0    1
0     0    1     0     0          0     0     0    1     0     0          0     0     0    1
0     0    0     0     0          0     0     0    0     0     0          0     0     0    0
0     0    0     0     0          0     0     0    0     0     0          0     0     0    0
0     0    0     0     0          0     0     0    0     0     0          0     0     0    0
```

```
n = 14 a = 5 b = 2 L = 19 OPT = 121
AAAAABBBBB--CCCCC--DDDDDAA---BBEEEEECCFFFFFDDGGGGGHHHHHEEIIIIIFFJJJJJGG---HHKKKKKIILLLLLJJMMMMMNNNNNKK-----LL-----MM---NN
0    0     0     1          1     1     0    1     1     1          1     1     0    1
0    0     0     1          1     1     0    1     1     1          1     1     0    1
0    0     0     0          0     0     0    0     0     0          0     0     0    0
0    0     0     0          0     0     0    0     0     0          0     0     0    0
0    0     0     0          0     0     0    0     0     0          0     0     0    0
0    0     0     1          0     0     1    0     0     1          0     0     1    0
0    0     0     1          0     0     1    0     0     1          0     0     1    0
0    0     1     0          1     0     0    1     1     0          1     0     0    1
0    0     1     0          1     0     0    1     1     0          1     0     0    1
0    0     0     0          0     0     0    0     0     0          0     0     0    0
0    0     0     0          0     0     0    0     0     0          0     0     0    0
0    0     0     0          0     0     0    0     0     0          0     0     0    0
0    0     1     1          0     1     1    0     1     1          0     1     1    0
0    0     1     1          0     1     1    0     1     1          0     1     1    0
0    1     0     0          0     0     0    1     0     0          0     0     0    1
0    1     0     0          0     0     0    1     0     0          0     0     0    1
0    0     0     0          0     0     0    0     0     0          0     0     0    0
0    0     0     0          0     0     0    0     0     0          0     0     0    0
0    0     0     0          0     0     0    0     0     0          0     0     0    0
```

Fig. 1. Two optimal schedules and their patterns

a path and x a vertex, that has occurred before in $\Pi_{\leq z+1}$, i.e.,

$$\Pi_{\leq z+1} = [\pi, x, \sigma, x, \pi', x, \sigma, x]$$

where π, π' are some paths with $|\pi'| > 0$. The path $[\pi, x, [\sigma, x]^2, \pi', x]$ has the same length as $\Pi_{\leq z+1}$. Moreover, its identical elementary cycles are consecutive. Finally, note that $[\pi, x, [\sigma, x]^2, \pi', x, \Pi_{>k+1}]$ has the same cost as Π. This contradicts our initial hypothesis. $\square$

We are now ready to describe the structure of optimal dominant paths. In the following, $v = |V|^{|V|}$.

Lemma 2 *An optimal dominant path has the following structure*

$$[\pi_1, (\sigma_1)^{q_1}, \pi_2, (\sigma_2)^{q_2}, ..., \pi_h, (\sigma_h)^{q_h}, \pi_{h+1}]$$

where (1) $h \leq v$, (2) $\pi_1, ..., \pi_{h+1}$ and $\sigma_1, ..., \sigma_h$ are elementary paths in which the last vertex of π_i is also the last vertex of σ_i (3) $q_1, ..., q_h$ are non-negative integer values and (4) $\sum_{i=1}^{h+1} |\pi_i| + \sum_{i=1}^{h} q_i = n$

PROOF. Consider an optimal dominant path. In this path, we have consecutive elementary cycles and between these cycles we have elementary paths (Lemma 1). Hence without loss of generality the path is

$$[\pi_1, (\sigma_1)^{q_1}, ..., \pi_h, (\sigma_h)^{q_h}, \pi_{h+1}]$$

where paths π_i and σ_i are elementary paths and where the last vertex of π_i is also the last vertex of σ_i. As π_i and σ_i are elementary paths, their lengths are upper bounded by $|V|$. Thanks to Lemma 1 the σ_i are pairwise distinct. As the total number of distinct elementary cycles in the directed graph $G = (V, A)$ is at most v, we have $h \leq v$. The number of vertices in the path is $\sum_{i=1}^{h+1} |\pi_i| + \sum_{i=1}^{h} q_i \times |\sigma_i|$ and it must be equal to n.

In the following, (h, π, σ) is said to be the *backbone* of the path. Note that in the backbone definition there is no reference to the number of times each cycle σ_i is repeated in a solution.

Lemma 3 *There are at most v^{2v+3} distinct backbones.*

PROOF. The total number of distinct elementary cycles in the directed graph $G = (V, A)$ is at most $v = |V|^{|V|}$. Hence the number of cycles h in a backbone varies from 0 to v. For a fixed h value we have to pick (i) $h + 1$

elementary paths π among v and (ii) h elementary paths σ among v. So we have at most $\sum_{h=0}^{v} v^{2h+1} \leq v^{2v+3}$ distinct backbones.

Lemma 4 *Given a backbone (h, π, σ), the best positive values $q_1, ..., q_h$ can be found in $O(h(\max_i |\sigma_i|)^2)$.*

PROOF. Given a path π, let $c(\pi)$ denote the cost associated to the path (sum of the distances between the consecutive vertices). Let $\bar{c}(\pi)$ denote the the cost associated to π plus the distance from the last vertex to the first one. We then have to solve the following problem:

$$\min c([\pi_1, ..., \pi_{h+1}]) + \sum_{i=1}^{h} q_i \times \bar{c}(\sigma_i)$$
$$u.c. \begin{cases} |[\pi_1, ..., \pi_{h+1}]| + \sum_{i=1}^{h} q_i \times |\sigma_i| = n \\ \forall i, q_i \in \{0, ..., n\} \end{cases}$$

This problem can be solved by dynamic programming in $O(nh)$ (knapsack like problem). Also note that for fixed a, b, L, we have an integer program with a fixed number of variables and hence it can be solved in polynomial time [9]. This would be enough to conclude the proof but as it is a special case of integer programming, we can setup a specific algorithm.

Without loss of generality, assume that $\forall i, \frac{\bar{c}(\sigma_i)}{|\sigma_i|} \geq \frac{\bar{c}(\sigma_1)}{|\sigma_1|}$. We claim that there is an optimal solution in which $\forall i \geq 2, q_i \leq |\sigma_1|$. Indeed, consider an optimal solution in which q_1 is maximal and assume that there is some i such that $q_i > |\sigma_1|$. Then we change the solution as follows: Increase q_1 of $|\sigma_i|$ and decrease q_i of $|\sigma_1|$. All variables remain non negative and the value of $\sum_{i=1}^{h} |\sigma_i| q_i$ does not change. Finally, note that the exchange does not increase the objective as $\bar{c}(\sigma_1)|\sigma_i| - \bar{c}(\sigma_i)|\sigma_1| \geq 0$. Hence we have again an optimal solution and the value of the first variable q_1 is larger than in the initial one. Contradiction.

Note that if $n - |[\pi_1, ..., \pi_{h+1}]| \leq |\sigma_1| \times (|\sigma_2| + ... + |\sigma_h|)$ then the overall complexity reduces to $O(h(\max_i |\sigma_i|)^2)$. Now assume that $n - |[\pi_1, ..., \pi_{h+1}]| > |\sigma_1| \times (|\sigma_2| + ... + |\sigma_h|)$ and let us compute a lower bound on q_1.

$$q_1 = \frac{n - |[\pi_1, ..., \pi_{h+1}]|}{|\sigma_1|} - \sum_{i=2}^{h} \frac{|\sigma_i| q_i}{|\sigma_1|} \geq \frac{n - |[\pi_1, ..., \pi_{h+1}]|}{|\sigma_1|} - \sum_{i=2}^{h} |\sigma_i| > 0$$

Hence we can rewrite the variables as follows: $q_1 = q'_1 + \left\lfloor \frac{n - |[\pi_1, ..., \pi_{h+1}]|}{|\sigma_1|} \right\rfloor -$

$\sum_{i=2}^{h} |\sigma_i|$ and $\forall i \geq 2, q_i = q'_i$ where $\forall i, q'_i \geq 0$. The problem reduces to

$$\min \sum_{i=1}^{h} \bar{c}(\sigma_i) q'_i$$
$$u.c. \begin{cases} \sum_{i=1}^{h} |\sigma_i| q'_i = n' \\ \forall i, q'_i \in \mathbb{N} \end{cases}$$

where

$$\begin{aligned} n' &= n - |[\pi_1, ..., \pi_{h+1}]| - |\sigma_1| \times \left\lfloor \frac{n - |[\pi_1, ..., \pi_{h+1}]|}{|\sigma_1|} \right\rfloor + |\sigma_1| \sum_{i=2}^{h} |\sigma_i| \\ &\leq |\sigma_1| \sum_{i=1}^{h} |\sigma_i| \\ &\leq (\max_i |\sigma_i|)^2 \end{aligned}$$

This new problem is again a knapsack problem and it can be solved in $O(h(\max_i |\sigma_i|)^2)$.

There are at most v^{2v+3} backbones. Given a backbone (h, π, σ), the best positive values $q_1, ..., q_h$ can be found in $O(h(\max_i |\sigma_i|)^2)$. As $h \leq v$ and $|\sigma_i| \leq |V|$, the overall complexity is upper bounded by $O(v^{2v+5})$ where $v \leq (a^{\frac{L}{a-1}})^{a^{\frac{L}{a-1}}}$.

5 Conclusion

We have shown that for fixed parameters a, b, L the coupled tasks problem can be solved in constant time. Still, the constant is very large and the existence of a more practical algorithm is still an open question. More generally, the complexity status of the problem for arbitrary a, b, L remains open.

6 Acknowledgment

The author is grateful to Gerd Finke and Nadia Brauner for several enlightening discussions on coupled tasks.

References

[1] Dino Ahr, József Békési, Gábor Galambos, and Gerhard Reinelt Marcus Oswald. An exact algorithm for scheduling identical coupled tasks. *Mathematical Methods of Operations Research*, 59:193–203, 2004.

[2] A. K. Shahani Alex J. Orman, Chris N. Potts and A. R. Moore. Scheduling for a multifunction phased array radar system. *European Journal of Operational Research*, 90:13–25, 1996.

[3] Frederic Barbaresco. Approche cognitive de la gestion radar. In *COGIS, Commande, Optimisation, Gestion Intelligente et architecture des Senseurs pour les systèmes*, 2003.

[4] Jacek Blazewicz, Klaus Ecker, Tamás Kis, and Michal Tanaś. A note on complexity of scheduling coupled tasks on a single processor. *Journal of the Brazilian Computer Society*, 7(3):23 – 26, 2001.

[5] Cyril Duron. *Ordonnancement en temps-réel des activités des radars.* PhD thesis, Universite de Metz, 2002.

[6] Moustafa Elshafei, Hanif D. Sherali, and J. Cole Smith. Radar pulse interleaving for multi-target tracking. *Naval Research Logistics*, 51:72–94, 2003.

[7] Jean-Philippe Hardange, Philippe Lacomme, and Jean-Claude Marchais. *Airborne and Spaceborne Radars.* Masson, 1995.

[8] Vassilissa Lebacque. *Théories et applications en ordonnancement : contraintes de ressources et tâches agrégées en catégories.* PhD thesis, INP Grenoble, 2007.

[9] Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4), 1983.

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

[11] Roy D. Shapiro. Scheduling coupled tasks. *Naval Research Logistics Quarterly*, 27(3):489–498, 1980.

[12] George W. Stimson. *Introduction to Airborne Radar.* SciTech Publishing, 1998.

[13] Emilie Winter and Philippe Baptiste. On scheduling a multifunction radar. *Aerospace Science and Technology*, 11:289–294, 2007.