Identical Coupled Task Scheduling Problem: The Finite Case

Michaël Gabay, Gerd Finke, Nadia Brauner

Identical Coupled Task Scheduling Problem: The Finite Case

Michaël Gabay, Gerd Finke, and Nadia Brauner

Laboratoire G-SCOP 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 - France

Abstract: Scheduling tasks on a radar system can be modeled as a multi-operation scheduling problem where tasks are composed of two operations separated by a fixed duration. Tasks have to be scheduled on a single machine and the aim is to minimize the makespan. The complexity status of this problem has been settled for various cases but the one where all tasks are identical remains open. This paper aims at showing that this case raises original and difficult problems and is highly unlikely to be polynomial.

Keywords: Scheduling, Coupled tasks, Complexity, High multiplicity.

1 Introduction

A radar system usually has a single antenna which is both used to send and receive radio waves. Moreover, after the emission, a wave has to travel some distance, be reflected, and then travel back to the radar before it can be received and processed. Other operations can be scheduled on the radar system during the wave travel time.

In 1980, Shapiro [1] introduced coupled tasks to model radar operations. A coupled task i is a two-operations task. The operation lengths are a_i (emit wave) and b_i (receive wave) and these operations are separated by a fixed duration L_i (travel time). A coupled task is illustrated Figure 1. The objective is to find a schedule that minimizes the makespan on a single machine (the radar). Shapiro [1] proved the $\mathcal{NP}-$completeness of this problem and proposed 3 heuristics.

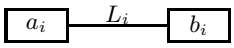

Fig. 1. A single coupled task

Orman and Potts [2] studied this problem and settled its complexity for various cases (Table 1). However, they claim that the identical coupled task scheduling problem where $\forall i$, $a_i = a$, $L_i = L$, $b_i = b$ remains open. This special case differs from the other cases since the input consists of 4 integers: a, b, L, and the number of tasks n (w.l.o.g. assume $a > b$ and $a + b \leq L$). Hence the input length is $O(log(L) + log(n))$ and an algorithm that is polynomial in n would then be exponential. Therefore, the schedule is not a valid certificate.

Strongly $\mathcal{NP}$-Hard	a_j; L_j; b_j
	$a_j = L_j = b_j$
	$a_j = b_j = p$; L_j
	$a_j = a$; $L_j = L$; b_j
Open	$a_j = a$; $L_j = L$; $b_j = b$
Polynomial	$a_j = L_j = p$; b_j
	$a_j = b_j = p$; $L_j = L$

Table 1. Complexity of coupled task scheduling problems

Such a problem is called a *high multiplicity scheduling problem*. Those problems discussed in [3–6] for instance.

Ahr *et al.* [7] proposed an exact algorithm using graphs for the identical coupled task scheduling problem. Its complexity is $O(nr^{2L})$ where $r \leq \sqrt[a-1]{a}$ holds. Baptiste [8] improved this result by proving that, for fixed a, b and L, the problem can be solved in $O(log(n))$ time. However, the constant is exponential in L and therefore this algorithm remains exponential.

Lehoux-Lebacque *et al.* [9] solved the cyclic case. In this case, the aim is to maximize the throughput rate.

We worked on the finite case (n tasks) and showed that it is significantly different from the cyclic case. Especially, none of the properties crucial for the cyclic case hold. More details about the results exposed in this paper and proofs can be found in [10].

2 The Cyclic Case

Lehoux-Lebacque *et al.* [9] proposed a polynomial time algorithm (complexity $O(log(L)^2)$) to solve the cyclic case. They use a pattern to describe cycles and return a concise certificate. In order to prove their results, they point out and use the following properties that hold in the cyclic case:

1. We always place as many operations as possible on the separation time L;
2. The order of operations in a cycle does not matter;
3. The pattern of operations within each coupled task is an invariant;

These properties allow to create a polynomial certificate in the length of the instance. This certificate is composed of 3 integers and its length is $O(log(L))$. An example of cycle is shown Figure 2.

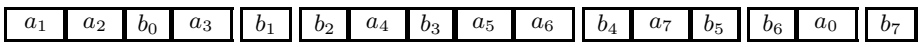

Fig. 2. $a = 5$, $b = 4$, $L = 15$, an optimal cycle

3 The Finite Case

3.1 From the Cyclic to the Finite Case

We started to work on the finite case with the idea that an optimal solution in this case may look like a cyclic solution surrounded by an extension to the left (to start the schedule) and an extension to the right (to finish it). We defined a *pure strategy* using the optimal cyclic pattern as the solution consisting to repeat this pattern (omitting the first b's as their a's were not scheduled) until n operations a have been scheduled and finish the last tasks. We proved the following theorem.

Theorem 1 (Asymptotic guarantee). *The pure strategy using the optimal cyclic pattern is asymptotically optimal.*

In fact, we proved that this strategy is a $1 + \frac{2}{k}$ approximation where k is the number of times the cycle is repeated. Moreover, computing the makespan of this schedule can be done in polynomial time.

This raised the following question: *is this pure-strategy-solution optimal for an n large enough ?* We have shown that this is not the case: for some instances, for each value of n, there exists a larger n' such that the optimal makespan is strictly smaller than the pure-strategy solution.

3.2 Issues Raised

We have implemented the problem as an integer program and solved instances using the commercial solver `CPLEX`. Running extensive tests allowed to find counterexamples for all properties of the cyclic case. Moreover, some optimal solutions are really combinatorial and the optimal solutions of different instances are usually quite different. Because of these differences, we have not been able to identify patterns (which would have been a way to create a certificate) for solutions in the finite case. Moreover, even with fixed characteristics (same a, b and L), the optimal solutions may change a lot for different values of n. In concrete terms, none of the properties 1, 2, 3 in the cyclic case, section 2, remains valid. We constructed counterexamples for those three properties in the finite case.

Eventually, in some special cases we have been able to prove that weaker versions of some properties remain true (for instance: for some classes of instances, we always place as many operations as possible during the first idle time).

3.3 A Polynomial Case

We have worked on some special cases of this problem. When $\frac{L}{a} + 1 \geq n$, the greedy schedule is obviously optimal. We also found the optimal solutions for very specific values of n when $a = b + 1$. The result can be obtained and checked in polynomial time. In those cases, the optimal solutions are non trivial and require to use different patterns, connected by some irregular transitions as represented Figure 3.

Yet, all the other cases remain unsolved. There is in a way a lack of recursive properties for the optimal solution patterns (given a, b, L and increasing n).

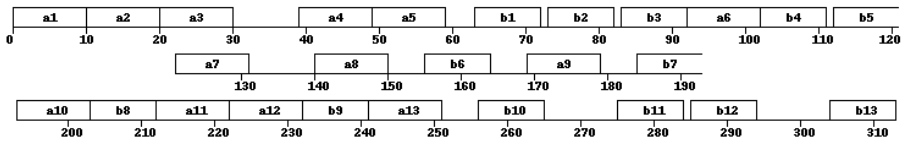

Fig. 3. $a = 10$, $b = 9$, $L = 53$, $n = 13$, an optimal solution.

4 Conclusion

We have shown that the finite case is significantly different from the cyclic case and that cyclic results and properties cannot be extended to the finite case. We demonstrated that a small subcase of the problem is polynomial but its solutions are very specific and elaborate. This strengthened the idea that the identical coupled task scheduling problem may not be polynomial and that we may not be able to find a valid certificate in the general case. The problem remains open but our results encourage us to think that it may not be in $\mathcal{NP}$. It remains to prove this conjecture, for instance by reducing an EXPSPACE-complete problem to this one.

References

1. R.D. Shapiro. Scheduling coupled tasks. *Naval Research Logistics Quarterly*, 27(3):489–498, 1980.
2. A.J. Orman and C.N. Potts. On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72(1-2):141–154, 1997.
3. D.S. Hochbaum and R. Shamir. Minimizing the number of tardy job units under release time constraints. *Discrete Applied Mathematics*, 28(1):45–57, 1990.
4. D.S. Hochbaum and R. Shamir. Strongly polynomial algorithms for the high multiplicity scheduling problem. *Operations Research*, 39(4):648–653, 1991.
5. J.J. Clifford and M.E. Posner. Parallel machine scheduling with high multiplicity. *Mathematical programming*, 89(3):359–383, 2001.
6. N. Brauner, Y. Crama, A. Grigoriev, and J. van de Klundert. A framework for the complexity of high-multiplicity scheduling problems. *Journal of combinatorial optimization*, 9(3):313–323, 2005.
7. 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(2):193–203, 2004.
8. P. Baptiste. A note on scheduling identical coupled tasks in logarithmic time. *Discrete Applied Mathematics*, 158(5):583–587, 2010.
9. V. Lehoux-Lebacque, N. Brauner, and G. Finke. Identical coupled task scheduling: polynomial complexity of the cyclic case. *Cahiers Leibniz 179*, 2009.
10. Michaël Gabay, Gerd Finke, and Nadia Brauner. Identical coupled task scheduling problem: the finite case. *HAL, http://hal.archives-ouvertes.fr/hal-00627902*. Technical report, 2011.