Parameterized complexity of a coupled-task scheduling problem

Stéphane Bessy, Rodolphe Giroudeau

Noname manuscript No.
(will be inserted by the editor)

Parameterized Complexity of a coupled-task scheduling problem

S. Bessy · R. Giroudeau

Received: date / Accepted: date

Abstract In this article, we investigate the parameterized complexity of coupled-task scheduling in the presence of compatibility constraint given by a compatibility graph. In this model, each task contains two sub-tasks delayed by an idle time, and a sub-task of a task can be performed during the idle time of another one if, and only if, the two tasks are compatible. We consider a parameterized version of the scheduling problem: is there a schedule in which at least k coupled-tasks possess a completion time before a fixed due date? It is known that this problem is $\mathcal{NP}$-complete ([23] and [24]), and we prove that it is *fixed-parameter tractable* ($\mathcal{FPT}$) if the total duration time of each task is bounded by a constant, whereas the problem becomes $\mathcal{W}[1]$-hard if it is not the case. We also show that in the former case the problem does not admit a polynomial kernel under some standard complexity assumptions. Moreover, we obtain also an $\mathcal{FPT}$ algorithm when the problem is parameterized by the size of a vertex cover of the compatibility graph.

Keywords Coupled-task scheduling model · $\mathcal{FPT}$ algorithms · $\mathcal{W}[1]$-hardness · Kernel lower bound

1 Introduction

Scheduling problems are in the mainstream of operations research, industrial engineering, manufacturing systems and computer science. More particularly, scheduling theory deals with executing a set of constrained tasks on a set of machines/processors. Here we focus on a single machine scheduling problem. The problems on a single machine are considered central in scheduling theory due to the historic nature and since the simplest type of scheduling models and a special case of all other more complex environments. Various examples of scheduling questions lead to $\mathcal{NP}$-complete problems on which classical algorithmic tools have been used to find acceptable solutions: approximation algorithms, randomized algorithms, exact algorithms with exponential time,

S. Bessy · R. Giroudeau
LIRMM UMR 5506, rue Ada,
34392 Montpellier Cedex 5 - France
E-mail: {bessy,girou}@lirmm.fr

etc (see the following references for a general presentation of scheduling models: [2], [7] and [13]). In this paper we present parameterized algorithms for a specific scheduling problem, the *coupled-task scheduling problem*. A problem *parameterized* by some integer k is said to be *fixed-parameter tractable* ($\mathcal{FPT}$ for short) whenever there exists a constant c such that it can be solved in time $f(k) \cdot n^c$, where f is a function, on an instance with size n and parameter k. An algorithm to solve the problem with such a running time is called an $\mathcal{FPT}$ algorithm. However, for some problems, like the $\mathcal{W}[1]$-*hard* problems, there is no hope to find $\mathcal{FPT}$ algorithms under standard assumptions in complexity theory. We refer to [9], [11] or [18] for parameterized algorithms theory. The originality of our work is that, up to our knowledge, very few results concerning parameterized complexity on scheduling problems have been shown (most of them are $\mathcal{W}[1]$-hard) and no such results are known for the coupled-task scheduling problem.

The coupled-task scheduling problem. The coupled-task scheduling problem consists of executing n tasks $\mathcal{A} = \{A_1, \ldots, A_n\}$ on a single machine. Each task A_i possess two sub-tasks, the first a_i and the second b_i, separated by a exact idle time l_i. This model of scheduling was introduced first by Shapiro in [21], motivated by an application to production scheduling, transportation, and radar operations (the first task models a send of information, the idle time is the delay to receive an answer and the second task is the treatment of it). Concerning the task-allocation on a single machine it is possible to *interleave* some of them. A task A_i interleaves with a task A_j if one of the sub-tasks a_j or b_j is processed during the idle time l_i of the task A_i. Moreover, there are usually two types of constraints on the tasks. The first one is actual processing times of the subtasks $\{a_1, b_1, \cdots, a_n, b_n\}$ and the idle times $\{l_1, \ldots, l_n\}$. We will abusively denote also by a_i (resp. b_i) the duration of the subtask a_i (resp. b_i), and we assume that we have $a_i > 0$ and $b_i > 0$ for all $i = 1, \ldots, n$. The other type of constraints on the tasks come from their possible heterogeneity. Indeed it is possible that some tasks could not interleave with some other tasks. It occurs when several tasks need to use a resource, different from the processor, available sequentially (this model is proposed in [23] and motivated by data acquisition and processing in a mono-processor torpedo used for underwater exploration). In order to represent these constraints, a *compatibility graph* $G_c = (\mathcal{A}, E_c)$ is introduced, where the set of G_c-vertices is the set $\mathcal{A}$ of coupled-tasks and $[A_i, A_j] \in E_c$ if, and only if, the two coupled-tasks A_i and A_j can be interleaved. An example of a coupled-task schedule is depicted in Figure 1[1].

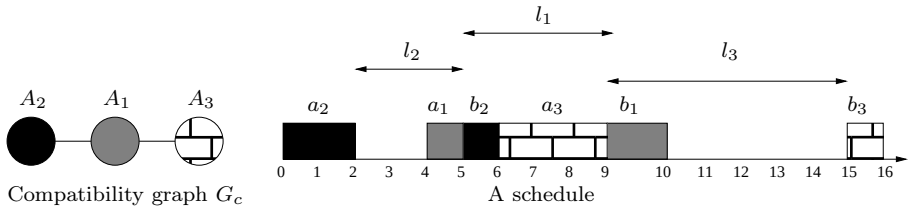

Fig. 1 A coupled-task schedule consistent with a compatibility graph such that $A_1 = (1, 4, 2)$, $A_2 = (2, 3, 1)$ et $A_3 = (3, 5, 1)$.

As all processing time are integers, we use a discrete time for the starting time of each task. Thus, in Figure 1, the schedule starts at time 0 and ends at time 16. Remark

[1] In all the figures, the two sub-tasks of a single task are colored with the same pattern.

that this schedule is optimal, as it is not possible to interleave the task A_2 and the task A_3 because the corresponding vertices in G_c are not adjacent. Given a schedule of the tasks, the *makespan*, denoted by C_{max} is the completion time of the last task of the schedule. The aim of the problem is to produce a shortest schedule (*i.e.* to minimize the makespan) according to compatibility constraint. In scheduling theory, a problem is categorized by its machine environment, job characteristics and objective function. Using the classical notation scheme $\alpha|\beta|\gamma$[2] proposed in [13], our problem will be defined as $1|(a_i, l_i, b_i), G_c|C_{max}$. The decision version we are interested in is more precisely the following.

Coupled-Tasks Scheduling Problem (CTS Problem):
Instance: A set $\mathcal{A} = \{A_1, \ldots, A_n\}$ of n coupled-tasks with associated processed and idle times $(a_i, l_i, b_i)_{1 \leq i \leq n}$ and with compatibility graph G_c and an integer C_{max}.
Question: Is there a schedule for the tasks that obeys the compatibilities given by G_c with a makespan at most C_{max}?

We also pay attention to some restrictive versions of the CTS Problem. If every involved subtask takes one unit of time and every idle time has same duration (*i.e.* $a_i = b_i = 1$ and $l_i = l_j$ for every $i, j \in \{1, \ldots, n\}$) we will refer to the problem as the unit CTS Problem. Moreover, let $L \geq 2$ be a fix integer. If every task of the instance has total duration bounded by L (*i.e.* we have $a_i + l_i + b_i \leq L$ for every $i = 1, \ldots, n$) we say that the problem is *L-bounded* and will refer it as the L-bounded CTS Problem. Finally combining the two previous notions, if every involved subtask takes one unit of time and every idle time has duration $L-2$ exactly (*i.e.* $a_i = b_i = 1$ and $l_i = L-2$ for every $i = 1, \ldots, n$) we will refer to the problem as the L-unit CTS Problem. In this paper, we study this problem from the parameterized complexity point of view. We will precise later the different parameters retained.

Related works. The coupled-task problem without compatibility constraints was first investigated from an algorithmic point of view in [19]. From this pioneer article on complexity, severals articles keep classifying in both modes (non cyclic and cyclic[3]). For the non cyclic mode, we suggest the two following references [3] or [8] for more recent results. A recent result given by [15], in cycle mode, propose a efficient polynomial-time algorithm for the special case (a, L, b) (notice that in non cycle case the status of this problem is unknown). The absence of compatibility constraints corresponds to the case of a complete compatibility graph G_c. The model in which the complete compatibility graph is relaxed has been studied in the framework of classic complexity and approximation, mainly in [23] and [24]. Remark that for $L = 2$, the L-unit CTS Problem is trivial, as there are no idle times and any permutation of the tasks is an optimal scheduling. For $L = 3$, roughly speaking, any scheduling can be associated with a matching on G_c (at most only one coupled-task A_i may be executed in the idle time of the coupled-task A_j with $[A_i, A_j] \in E_c$), and the length of the schedule will therefore depend on the size of the matching. Thus a maximum matching in the compatibility graph leads to an optimal solution for the 3-unit CTS Problem, see [24] for more details. In the same paper, the authors prove that for $L \geq 5$ the L-unit CTS Problem is $\mathcal{NP}$-complete (the hardness for $L = 4$ is given in [23]), and so is the L-unit CTS Problem for $L \geq 4$ and also the CTS Problem. Thus, we assume

[2] where α designates the environment processors, β the characteristics of the job and γ the criteria.

[3] In this problem multiple copies of a coupled-task are to be processed.

that $L \geq 4$ for the rest of the paper. Concerning approximation algorithms, it is shown in [23] that the 4-UNIT CTS PROBLEM admits a polynomial $\frac{13}{12}$-approximation algorithm. Whereas, in [24] a $\frac{4+L}{4}$-approximation algorithm is given for the general L-UNIT CTS PROBLEM. In the case of the graph G_c is a bipartite graph some new results on complexity/approximation are proposed in [22]. Nevertheless, the status of CTS PROBLEM with G_c being a tree remains unknown.

As mentioned before, there are few results in the parameterized complexity approach for scheduling theory: on negative side, in [4] the authors prove the $\mathcal{W}[2]$-hardness for the precedence constrained k-processors scheduling. A contrario, in [10], the authors give a positive answer to the question "Is there a one-processor schedule for a set of unit-tasks in presence of precedence contraints admitting a individual deadline, and contains no more than k late tasks (resp. on time) ?" Theses problems are proven as $\mathcal{W}[1]$-hard, but become $\mathcal{FPT}$ for bounded partial orders for the two previous cases. In [25], the authors propose some $\mathcal{FPT}$ results for interval scheduling based on an Independent Set formulation. Recently, in [17], the authors propose some fixed-parameter algorithms for some classical scheduling problems as makespan minimization, scheduling with job-dependent cost functions and scheduling with rejection.

As mentioned by Marx in [16] (pp. 86), an interesting parameterization for a scheduling problem, which admits a polynomial-time algorithm to solve it, is to ask if an instance of the problem admits a schedule with at most k-tasks late. The challenging question here is to obtain an $\mathcal{FPT}$ algorithm for this parameterized version, avoiding then all the $O(n^k)$ possible choices for the late tasks. However, this parameterization could not fit to our problem, as for $L \geq 4$, the L-UNIT CTS PROBLEM is $\mathcal{NP}$-complete (*i.e.* there is no polynomial algorithm even for $k = 0$). Another natural way to parameterize a scheduling problem is used by Fellows and McCartin in [10]. They study a scheduling problem dealing with simple tasks having a due date and a set of precedence constraints. The corresponding optimization problem being $\mathcal{NP}$-complete, they were interested in a 'k tasks on time' version of it. More precisely, they look for a set of k tasks that could be scheduled according to precedence constraints and such that each task ends before its due date. They obtain a $\mathcal{W}[1]$-hardness result for this parameterized problem.

Our contribution. To turn the CTS PROBLEM into a parameterized problem, we were inspired by the 'k tasks on time' version used by Fellows and McCartin in [10]. Indeed, we fix a due date $d \in \mathbb{N}$ and try to find a schedule with a maximum number of coupled-tasks executed before d. The parameter is then the number of tasks admitting a completion time before d. Namely, we consider the following problem.

> COUPLED-TASKS SCHEDULING TARDY PROBLEM (CTST PROBLEM):
> **Instance**: A set $\mathcal{A} = \{A_1, \ldots, A_n\}$ of n coupled-tasks with associated processed and idle times $(a_i, l_i, b_i)_{1 \leq i \leq n}$, a compatibility graph G_c and a deadline $d \in \mathbb{N}$.
> **Parameter**: An integer k.
> **Question**: Is there a schedule for the tasks that obeys the compatibilities given by G_c and contains at least k tasks that are completed before the deadline d?

As for the CTS PROBLEM, we also pay attention to the unit, L-bounded and L-unit versions of the CTST PROBLEM.
Section 2 is devoted to parameterized complexity results for the CTST PROBLEM. Using a parameterized reduction from the k-CLIQUE PROBLEM, we show that the

CTST Problem is $\mathcal{W}[1]$-hard. Whereas if L is fixed, the L-bounded CTST Problem is fixed-parameter tractable. We obtain this result using the color coding technique (see [1]) and dynamic programming.
One of the standard method to design fixed-parameter algorithms is to obtain a *kernelization algorithm* (or *kernel*) for the studied problem. A kernelization algorithm is a polynomial-time algorithm that given an instance I with parameter k of a parameterized problem computes an instance I' with parameter k' such that (I, k) is a positive instance if and only if (I', k') is a positive instance of the problem, $|I'| \leqslant h(k)$ for some computable function $h(k)$ depending only on k and $k' \leqslant k$. We say that the kernel is a *polynomial kernel* if the function $h(k)$ is a polynomial in k. It is well-known that a decidable parameterized problem is $\mathcal{FPT}$ if and only if it has a kernelization algorithm [18]. But this equivalence only yields kernels of super-polynomial size. To design efficient fixed-parameter algorithms, a kernel of small size - polynomial (or even linear) in k - is highly desirable. Using the cross composition technique developed by Bodlaender et al. in [5], we prove that there is no hope to find a polynomial kernel for the L-unit CTST Problem unless $co - \mathcal{NP} \subseteq \mathcal{NP}/Poly$.
In Section 3, we obtain an $\mathcal{FPT}$ result for the L-bounded CTST Problem parameterized by a structural invariant of the compatibility graph G_c. A *vertex cover* of a graph G is a subset X of the vertices of G such that for all edge $[u, v]$ of G we have $u \in X$ or $v \in X$. We prove that there exists an $\mathcal{FPT}$ algorithm for the L-unit CTST Problem when parameterized by a size of the vertex cover of G_c. More precisely, we obtain a (non polynomial) kernel for this problem. We also argue that such an $\mathcal{FPT}$ algorithm does not exist for the unit CTST Problem. Finally we conclude the paper by some remarks and open questions.

2 Parameterized complexity results for the CTST Problem

In this section, we show that the status in term of parameterized complexity of the CTST Problem changes if the problem is L-bounded or not. First, when it is not the case we obtain the following negative result for the CTST Problem.

Theorem 1 *The* unit CTST Problem *(and so the* CTST Problem*) is $\mathcal{W}[1]$-hard with respect to the parameter k, ie. it has no $f(k)n^{O(k)}$ time algorithm for any computable function f (unless the W-hierarchy collapses at first level).*

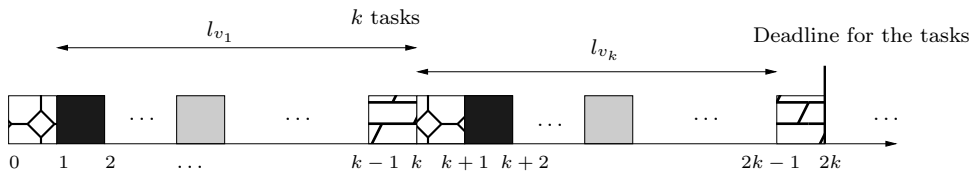

Fig. 2 Description of the beginning of the schedule.

Proof We propose a parameterized reduction from the k-clique problem which is known to be $\mathcal{W}[1]$-hard (see [6]). Given an instance $(G = (V, E), k)$ of this problem,

we provide an instance $(\mathcal{A} = \{A_1, \ldots, A_n\}, G_c, d)$ with parameter k' of the UNIT CTST PROBLEM with $k' = k$. We consider one coupled-task A_v for each vertex v of G and choose $G_c = G$. Then, we set $a_v = b_v = 1$ and $l_v = k - 1$ for every vertex v of G and $d = 2k$. This transformation can be clearly computed in polynomial time on n. We claim that G has a clique of size k if, and only if, $(\{A_1, \ldots, A_n\}, G_c, d)$ admits a schedule in which at least k coupled-tasks end before d. Indeed, assume that G contains a clique $C = \{v_1, \ldots, v_k\}$ of size k. The coupled-tasks $\{A_{v_1}, \ldots, A_{v_k}\}$ are processed during $[0, 2k]$ without idle time: for $1 \leq i \leq k$, the starting time of the sub-task a_{v_i} is $(i-1)$ and for the sub-task b_{v_i} is $(k+i-1)$ (see Figure 2). All A_{v_i}-tasks interleave into each other. These allocation respect the constraints since C is a clique, and all coupled-tasks are completed at time $2k$. Conversely, assume that $(\{A_1, \ldots, A_n\}, G_c, d)$ admits a schedule such that the completion time of a k coupled-tasks is at most $2k$. Since each coupled-task contains two sub-tasks with processing time equal to one, it is clear that there is no idle time before the deadline. Hence, the only possibility to schedule the k first coupled-tasks in this way is as described before. As all these tasks must interleave, G has a clique of size k. □

Now, we fix an integer $L \geq 4$ and we prove that the L-BOUNDED CTST PROBLEM admits an $\mathcal{FPT}$ algorithm. For this, we use the color-coding technique developed by Alon, Yuster and Zwick in [1].

Theorem 2 *For every fixed $L \geq 4$, there exists an $\mathcal{FPT}$ algorithm for the L-BOUNDED CTST PROBLEM with running time $O(2^{O(k)} n^{2L} \log^2 n)$.*

Proof Consider an instance $(\mathcal{A} = \{A_1, \ldots, A_n\}, (a_i, l_i, b_i)_{1 \leq i \leq n}, G_c, d)$ with parameter k of the L-BOUNDED CTST PROBLEM. We can presume that $d < Lk$, otherwise we schedule the tasks in turn with no interleaving and answer yes to the problem. Following the standard use of the color coding technique, we define a colored version of the problem. Assume that the coupled-tasks are colored with colors $\{1, \ldots, k\}$, we find a feasible schedule of k coupled-tasks that are on time and that involves coupled-tasks of all different colors. We will argue later that if a schedule forms a positive answer for our initial problem, then it is possible to find 'quickly' a coloring for which this schedule will be a positive answer for the associated colored version.
To solve the colored version, we use dynamic programming in order to extend the partial schedules of color tasks. Indeed, for this we need only informations on the last L slots of time of the partial schedule under construction. Then, we can decide how to add a new task in this schedule. More precisely, let $\mathcal{S}$ be a partial schedule of $\{A_1, \ldots, A_n\}$ respecting the compatibility constraints given by G_c. We focus on the set $\mathcal{T}$ of tasks of $\mathcal{S}$ occupying one of the $L-1$ last unit time slots of $\mathcal{S}$. For these tasks there are at most $(L-1)^{|\mathcal{T}|} \leq L^L$ possibilities: each of the last slot of time is used by one task of $\mathcal{T}$ at most (see Figure 3 for an illustrative case with $L = 4$). Note that all the configurations possibly do not occur, because of the shape of the tasks and of the constraints on the tasks given by G_c.

We encode the behaviors of the tasks of $\mathcal{T}$ by an integer B of $\{1, \ldots, (L-1)^{|\mathcal{T}|}\}$. We say that a schedule $\mathcal{S}$ *has end* $(\mathcal{T}, B, s)$ if the makespan of $\mathcal{S}$ is s and if the set of tasks of $\mathcal{S}$ using one of the L last time slot of $\mathcal{S}$ is exactly $\mathcal{T}$ and have behavior B. Finally, we say that a triple $(\mathcal{T}', B', s')$ *fits with* $(\mathcal{T}, B, s)$ if there exists a schedule $\mathcal{S}$ which has end $(\mathcal{T}, B, s)$ and such that $\mathcal{S}' = \mathcal{S} \setminus A_i$ has end $(\mathcal{T}', B', s')$, where A_i is the last task of $\mathcal{S}$. Remark that in the case $\mathcal{T} \neq \{A_i\}$ the penultimate task of $\mathcal{S}$ is the last task of $\mathcal{S}'$ and that $\mathcal{S}$ and B constrain $\mathcal{S}'$ and B'. Remark also that s' takes value in

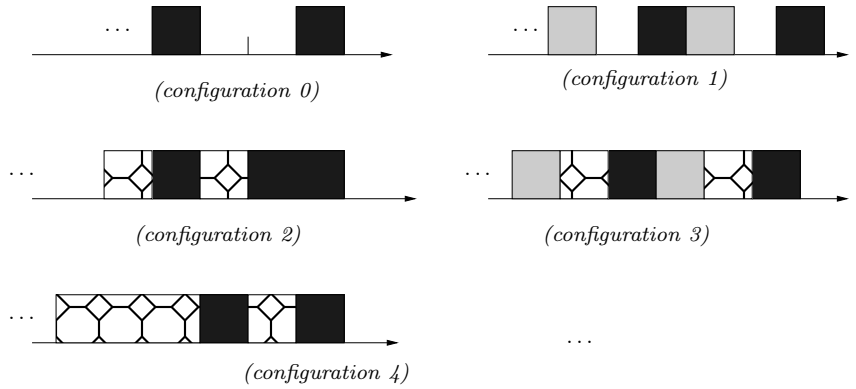

Fig. 3 Examples of possible configurations with $L = 4$. The last task is represented by a black pattern

$\{s-L, \ldots, s-1\}$. For instance, we have $s' = s-1$ if the sub-task b_i of A_i has duration one and just follows the last sub-task of the last task of $\mathcal{S}'$ as in configurations 2 and 3 in Figure 3. And we have $s' = s - L$ if A_i has total duration L and $\mathcal{T} = \{A_i\}$ as in configuration 0 of Figure 3. Now, given X a set of colors (*i.e.* a subset of $\{1, \ldots, k\}$), a set $\mathcal{T}$ of at most L tasks, a behavior B for these tasks and a time $s \leq d$, we set $Q[X, \mathcal{T}, B, s] = 1$ if there exists a partial schedule $\mathcal{S}$ of $\mathcal{A}$ that uses exactly one task of each color of X and which has end $(\mathcal{T}, B, s)$. Otherwise, we set $Q[X, \mathcal{T}, B, s] = 0$. Then we dynamically compute all the values $Q[X, \mathcal{T}, B, s]$ by increasing the size of X. If X has cardinality one, say $X = \{l\}$, we set $Q[X, \mathcal{T}, B, s] = 1$ if, and only if, $\mathcal{T}$ contains only one task which has color l, behavior B and total duration s. Now, assume that we have computed all the values $Q[X, \mathcal{T}, B, s]$ with $|X| \leq p$ and consider an input $Q[X, \mathcal{T}, B, s]$ with $|X| = p + 1$. Then we set the following.

$$(\star) \quad Q[X, \mathcal{T}, B, s] = \max\{Q[X \setminus \{c\}, \mathcal{T}', B', s'] \ : \ \text{the last task of } \mathcal{T} \text{ has color } c \text{ and } (\mathcal{T}', B', s') \text{ fits with } (\mathcal{T}, B, s)\}$$

Where the maximum runs over all the possible triples $(\mathcal{T}', B', s')$. It is clear that if there exists a partial schedule $\mathcal{S}$ which uses exactly one task of each color of X, which has end $(\mathcal{T}, B, s)$ and where its last task, A_i, has color c, then the end of $\mathcal{S} \setminus A_i$ will fit with $(\mathcal{T}, B, s)$ and $\mathcal{S} \setminus A_i$ will use one task of each color of $X \setminus \{c\}$. So in this case, the right part of the equality $(\star)$ is 1. Conversely if one of the value $Q[X \setminus \{c\}, \mathcal{T}', B', s']$ used in $(\star)$ is 1, then we can extend the corresponding schedule with A_i.
Now, we answer yes to the colored version of the problem if one of the value $Q[X, \mathcal{T}, B, s]$ with $X = \{1, \ldots, k\}$ is 1. Let us see how long could be the overall computation. The number of values $Q[X, \mathcal{T}, B, s]$ to compute is bounded by $2^k \times n^L \times L^L \times d$, where we bound the number of subsets of at most L tasks by n^L. Thus, as we have $d \leq Lk$, the number of values $Q[X, \mathcal{T}, B, s]$ is finally at most $k2^k L^{L+1} n^L$. Moreover, to compute a value $Q[X, \mathcal{T}, B, s]$ with $(\star)$ we have to examine all the triples $(\mathcal{T}', B', s')$ which fit with $(\mathcal{T}, B, s)$, agree with constraints given by G_c and with the colors constraints. We roughly list all these triple but in $\mathcal{FPT}$ time. First, knowing the last task of $(\mathcal{T}, B)$, it is possible to compute exactly s' from s. We already argue that there is at most $L^L n^L$ possible choices for the couples $(\mathcal{T}', B')$. Once such a couple is fix, we check that the

interleaves given by $(\mathcal{T}', B')$ are compatible with the graph G_c, what can be done in time $O(L^2)$ with the adjacency matrix of G_c. Then we check unit slot of time by unit slot of time that $(\mathcal{T}', B', s')$ fits with $(\mathcal{T}, B, s)$, requiring $O(L)$ time. Finally we have to verify that the colors of the tasks of $\mathcal{T}' \setminus \mathcal{T}$ are pairwise distinct, distinct from the color of the tasks of $\mathcal{T}' \cap \mathcal{T}$ and belong to X. This checking has to be done for at most L tasks and so ask for a $O(L)$ time.
For all, we can compute the right part of $(\star)$ in time $O(L^2).L^L n^L = O(n^L)$ and answering to the colored version of the problem could be done in time $O(k2^k n^{2L})$.
As usual when using color coding, to conclude we need a *k-perfect family of hash functions* from $\{1, \ldots, n\}$ to $\{1, \ldots, k\}$, that is a set of colorations such that for every subset U of $\{1, \ldots, n\}$ of size k there exists one of these colorations which colors the elements of U with k different colors. Schmidt and Siegal [20] explicitly provide such a family of colorations of size $2^{O(k)} \log^2 n$ which can be computed in time $O(2^{O(k)} \log^2 n)$.
Finally we generate all these colorations of our initial set of tasks $\mathcal{A}$ and for every coloration from this family, we run the previous dynamic programming algorithm. If one of these computations give a positive answer then we answer positively to the L-BOUNDED CTST PROBLEM on the initial instance. And if none of these algorithms return a positive answer, then we answer negatively to the L-BOUNDED CTST PROBLEM on our starting instance. Indeed if a suitable schedule of k tasks would exist, then one of the chosen coloration would have color these tasks with different colors and the corresponding dynamic program would have answer positively on it.
Finally, we obtain a total time for the $\mathcal{FPT}$ algorithm of $O(2^{O(k)} n^{2L} \log^2 n)$. □

Hereafter, we propose a new negative result in term of parameterized complexity. A *cross-composition*[4] is a polynomial-time algorithm that, given a sequence $(x_1, \ldots, x_t)$ of t instances with the same size of an $\mathcal{NP}$-complete problem A, computes an instance (y, k) of a parameterized problem B such that k is a polynomial in $|x_1| + \log t$ and (y, k) is a positive instance of B if and only if there is some $1 \leq i \leq t$ such that x_i is a positive instance of A. Using the result proposed by Bodlaender et al. in [5], we know that if a parameterized problem B admits such a cross-composition, then there is no polynomial-size kernel for B unless $co - \mathcal{NP} \subseteq \mathcal{NP}/poly$.
In order to show the kernel lower bound, we propose a cross-composition from the $\mathcal{NP}$ complete HAMILTONIAN ODD BIPARTITE GRAPH PROBLEM where we ask if a given bipartite graph with an odd number of vertices has a Hamiltonian path (it exists a reduction from the $\mathcal{NP}$-complete HAMILTONIAN BIPARTITE GRAPH PROBLEM to this problem).

Theorem 3 *For every fixed $L \geq 4$, the L-UNIT CTST PROBLEM and the L-BOUNDED CTST PROBLEM do not admit a polynomial kernel unless $co - \mathcal{NP} \subseteq \mathcal{NP}/Poly$ with respect to the parameter k.*

Proof First let us focus on the L-UNIT CTST PROBLEM.
We consider some instances $G_1, \ldots, G_t$ of the HAMILTONIAN ODD BIPARTITE GRAPH PROBLEM with the same odd number of vertices denoted by n. Let $L \geq 4$ be fixed, we provide an instance $I = (\mathcal{A} = \{A_1, \ldots, A_N\}, G_c, d)$ of the L-UNIT CTST PROBLEM with parameter $k = n$. We choose G_c to be the disjoint union of the graphs $G_1, \ldots, G_t$. The integer N is then the number of vertices of G_c (*i.e.* $N = tn$) and we associate one task A_i to each vertex of G_c. Finally, we set $d = \frac{(n+1)(L+2)}{2}$, which is an integer since

[4] Notice that the given definition, while restricted, is sufficient for your purpose.

n is odd. We claim that I is a positive instance of the L-UNIT CTST PROBLEM if, and only if, G_c contains a path of length n, which will be the case if, and only if, one of G_i contains an Hamiltonian path.
First remark the following. Consider three consecutive tasks $A_{i_1}, A_{i_2}, A_{i_3}$ of any schedule $\mathcal{S}$ of p tasks of I. The tasks A_{i_1} and A_{i_3} cannot interleave, otherwise as a_{i_2} stands between a_{i_1} and a_{i_3}, A_{i_1}, A_{i_2} and A_{i_3} must pairwise interleave, which is not possible as G_c does not contain any clique of size three. Thus, the first task of $\mathcal{S}$ does not interleave with the third task, which itself does not interleave with the fifth one, and so on. So the makespan of the schedule $\mathcal{S}$ is at least $\lceil \frac{p}{2} \rceil (L+2)$.
Now, assume that I is a positive instance and denote by $\mathcal{S} = (A_{i_1}, \ldots, A_{i_k})$ the corresponding partial schedule of k $(= n)$ tasks with makespan at most d. By the previous remark, the makespan of $\mathcal{S}$ is at least $\lceil \frac{k}{2} \rceil (L+2) = \frac{(n+1)(L+2)}{2} = d$. So, the makespan of $\mathcal{S}$ is exactly d and two tasks with consecutive odd index in $\mathcal{S}$ are consecutive along the time. More precisely, for every odd $j \leq n-2$, the sub-task $a_{i_{j+2}}$ immediately follows the sub-task b_{i_j}, and the task $A_{i_{j+1}}$ interleaves with A_{i_j} and $A_{i_{j+2}}$. If we denote by $v_1, \ldots, v_n$ the vertices of G_c respectively associated with the tasks $A_{i_1}, \ldots, A_{i_n}$, it means that $v_1 \ldots v_n$ is a path of G_c, and thus it exists a Hamiltonian path in one of the G_is.
Conversely, if one of graph G_i has an Hamiltonian path, we schedule the corresponding tasks of I as depicted previously.
Finally notice that the provided instance $I = (\mathcal{A} = \{A_1, \ldots, A_N\}, G_c, d)$ is also an instance of the L-BOUNDED CTST PROBLEM. Thus the result holds also for this problem.

□

3 A structural parameter for the CTS PROBLEM

In this section, we focus on the L-bounded version of the original problem parameterized by some structural parameter of the compatibility graph.

So, let $(L, \mathcal{A} = \{A_1, \ldots, A_n\}, G_c, C_{max})$ be an instance of the L-BOUNDED CTS PROBLEM. If the compatibility graph G_c has no edges, then any permutation of the tasks is an optimal scheduling. We extend this observation to obtain an $\mathcal{FPT}$ algorithm for the CTS PROBLEM with the size of a vertex cover of the graph G_c as parameter. The vertex cover of a graph is usually considered as a 'large' parameter in parameterized complexity. However we will argue in the next section that it seems hard to obtain the same kind of $\mathcal{FPT}$ algorithm according to some lower structural parameter for G_c (as for instance, feedback vertex set or treewidth...).
The next lemma will be useful to design the $\mathcal{FPT}$ algorithm. A subset X of vertices of a graph G are *false twins* if it forms an independent set of G and if all the vertices of X have the same neighborhood in G.

Lemma 1 *Let $\mathcal{S}$ be any schedule of the coupled-tasks $\mathcal{A}$ according to compatibility graph G_c and p be a non negative integer. If there exists $(2p+1)$ tasks $A_{i_1}, \ldots, A_{i_{2p+1}}$ of $\mathcal{A}$ which corresponding vertices are false twins in G_c and have a common neighborhood of size at most p in G_c, then one of the task A_{i_j} does not interleave with any task of $\mathcal{A}$ in $\mathcal{S}$.*

Proof Denote by $x_1, \ldots, x_{2p+1}$ the vertices of G_c respectively associated with the tasks $A_{i_1}, \ldots, A_{i_{2p+1}}$. Denote also by Y the common neighborhood of $x_1, \ldots, x_{2p+1}$ in G_c, and by $A_{i'_1}, \ldots, A_{i'_l}$, with $l \leq p$, the coupled-task corresponding to the vertices of Y. As $\{x_1, \ldots, x_{2p+1}\}$ forms an independent set of G_c, the tasks $A_{i_1}, \ldots, A_{i_{2p+1}}$ do not interleave. So, a task $A_{i'_j}$ can interleave with at most two tasks of $\{A_{i_1}, \ldots, A_{i_{2p+1}}\}$. Thus, as $l \leq p$, there exists at least one task A_{i_j} which does not interleave with any task of $A_{i'_1}, \ldots, A_{i'_l}$. Moreover, as x_j has no neighbor in $G_c \setminus Y$, A_j does not interleave with any task of $\mathcal{A}$. □

Theorem 4 *For every fixed $L \geq 4$, there exists an $\mathcal{FPT}$ algorithm for the* L-BOUNDED CTS PROBLEM *parameterized by the size k of a vertex cover of G_c, with running time $O((k2^kL^3)! \cdot L^{k2^kL^3} + n^2)$.*

Proof Let $I = (L, \mathcal{A} = \{A_1, \ldots, A_n\}, G_c, C_{max})$ be an instance of the L-BOUNDED CTS PROBLEM. Notice that all tasks must be executed. Let also X be a vertex cover of G_c with size k. The vertices of $G_c \setminus X$ form an independent set of G_c, and in particular their neighborhood is included in X. For each subset X' of X, we compute the set $N_{X'}$ of vertices of $G \setminus X$ with neighborhood exactly X'. This could be done in linear time in the size of G_c, *i.e.* in time[5] $O(n(G_c) + m(G_c))$, using for instance partition refinement, see [14]. By definition, every set $N_{X'}$ is composed by false twins of G_c. Let us refine again this partition. Each task $A_i = (a_i, l_i, b_i)$ has total duration at most L, and the values a_i, b_i and l_i totally determines the time behavior of A_i. So there are at most L^3 possibilities for these values. Thus for each values (a_i, l_i, b_i), the set $N_{X',(a_i,l_i,b_i)}$ contains the tasks of $N_{X'}$ with time behavior (a_i, l_i, b_i). As the possible values for (a_i, l_i, b_i) are known, we can perform this partition of $N_{X'}$ in linear time. Now if, for a subset X' of X and fixed values (a_i, l_i, b_i), we have $|N_{X',(a_i,l_i,b_i)}| > 2k$, then by Lemma 1 $(|N_{X',(a_i,l_i,b_i)}| - 2k)$ coupled-tasks corresponding to some vertices of $N_{X',(a_i,l_i,b_i)}$ do not interleave with any other coupled-tasks of $\mathcal{A}$ in any schedule. Therefore, to design an optimal schedule for the problem, since all the coupled-tasks corresponding to the vertices of $N_{X',(a_i,l_i,b_i)}$ are identical, $|N_{X',(a_i,l_i,b_i)}| - 2k$ arbitrarily coupled-tasks can be selected and allotted at the beginning of the schedule in the following way: not interleaving, consecutive and in any order. We also reduce the value C_{max} by $(|N_{X',(a_i,l_i,b_i)}| - 2k)(a_i + l_i + b_i)$. We repeat this routine for every subset X' of X and every values (a_i, l_i, b_i) with $|N_{X',(a_i,l_i,b_i)}| > 2k$. Once a coupled-task is executed at the beginning of the schedule, we can remove it from the instance I. So, we obtain an equivalent instance I' containing at most $2k \cdot 2^k \cdot L^3$ tasks with an objective makespan C'_{max}: I' forms a kernel for the problem, with exponential size and computed in time $O(n^2)$.
Now, we solve exhaustively the problem on I'. For instance, we can fix an order for the coupled-tasks and execute them in this order. Once a partial schedule is build, there is L possibilities to allot the sub-task a_i of the next task ($L-1$ for a possible interleaving with the previous tasks plus 1 if the task A_i does not interleave with its previous task). So, for each order on the tasks, we compute in time $O(L^{2k2^kL^3})$ the optimal schedule for I' with tasks in the fixed order. Finally, the optimal schedule for I' is found in time $O((k2^kL^3)! \cdot L^{k2^kL^3})$. □

[5] $n(G_c)$ (resp. $m(G_c)$) represents the number of vertices (resp. edges).

If the total duration of the tasks is not bounded, it is not possible to have such an $\mathcal{FPT}$ algorithm. Indeed in this case the CTS Problem is $\mathcal{NP}$-complete even if G_c is a star (that is G_c has a vertex cover of size one).

Lemma 2 *The* CTS Problem *restricted to the instances with compatibility graph being a star is an $\mathcal{NP}$-complete problem.*

Proof We use a reduction from the subset sum problem. An instance of subset sum is a set S of integers and a target integer value t. A positive answer to the problem is a subset S' of S with $\sum_{s\in S'} s = t$. It is known that the subset sum problem is an $\mathcal{NP}$-complete problem (see [12] for instance). Let (S,t) be an instance of the subset sum problem. We denote by $\{s_1,\ldots,s_n\}$ the elements of S. We consider the following instance of the CTS Problem. The task A has two sub-tasks with duration 1 and an idle time of length $2t$. To every element s_i of S we associate a task A_i with two sub-tasks of duration s_i and no idle time. The graph G_c on the vertex set $\{A, A_1,\ldots,A_n\}$ is a star with center A and leaves $\{A_1,\ldots,A_n\}$. Finally we set $C_{max} = 2(\sum_{s_i\in S} s_i) + 2$ and we claim that $(\{A, A_1,\ldots,A_n\}, G_c, C_{max})$ is a positive instance of the CTS Problem if and only if (S,t) is a positive instance of the subset sum problem. Indeed, assume that $(\{A, A_1,\ldots,A_n\}, G_c, C_{max})$ is a positive instance of the CTS Problem. As the sum of the duration of all the sub-tasks of the tasks $\{A, A_1,\ldots,A_n\}$ is exactly C_{max}, it means that these tasks can be schedule without idle time. In particular the set $\{A_i \ : \ i \in I'\}$ of tasks filling the idle time of A has total duration exactly $2t$. So the corresponding set $S' = \{s_i \ : \ i \in I'\}$ has total value t and (S,t) is a positive instance of the subset sum problem. Conversely we produce similarly an optimal schedule of $\{A, A_1,\ldots,A_n\}$ from a solution to the subset sum problem. □

4 Concluding remarks and open problems

In this article, we propose positive results for a coupled-task scheduling problem in presence of compatibility constraint. Note that in scheduling theory few positive results exist for parameterized complexity, mainly because most of known problems are $\mathcal{W}[1]$-hard when parameterized by standard parameters. One of the motivation in designing $\mathcal{FPT}$ algorithms is to obtain efficient algorithms for small value of the parameter. Unfortunately the obtained algorithms, specially Theorem 2, seem not be practically useful. It is interesting to know the existence of such $\mathcal{FPT}$ algorithms, but for practical purpose it could be challenging to have smaller functions of the parameter in the running time of these algorithms.

It is possible to extend our main result, Theorem 2, to a more general version of the CTS Problem. In the Generalized CTS Problem, every tasks A_i contains severals sub-tasks $a^i_1,\ldots,a^i_{n_i}$ and for every $j = 1,\ldots,n_i-1$, the sub-task a^i_{j+1} must be executed exactly after an idle time l^i_j succeeding the end of the sub-task a^i_j. To interleave now at a certain position, two tasks must have linked corresponding vertices in the compatibility graph and no sub-task have to overlap in this position. The Generalized CTST Problem asks if there exists a schedule of the tasks A_i which contains at least k tasks executed before a given due time. For a constant L, we say that the problem is L-bounded if the total duration of every tasks is bounded by L (*i.e.* for every i we have $a^i_1 + l^i_1 + a^i_2 + \cdots + l^i_{n_i-1} + a^i_{n_i} \leq L$). In the proof of Theorem 2

to run the dynamic programming we use the fact that when L is fixed there is a finite number of possible kind of tasks and a finite number of possible kind of interaction between tasks. As this remark still holds in the case of tasks with several sub-tasks we obtain the following.

Theorem 5 *For every fixed $L \geq 4$, there exists an $\mathcal{FPT}$ algorithm for the L-*bounded Generalized CTST Problem.

However the Generalized CTS Problem does not appear in the literature and is probably less worthy of interest than the (standard) CTS Problem.

Finally, for the CTST Problem, it could also be interesting to choose other (and lower) structural parameter of the graph G_c in order to design a better $\mathcal{FPT}$ algorithm than the one given by Theorem 4. However, it seems hard to choose a parameter like the tree-width or the size of a feedback vertex set of G_c because, the status of CTS Problem is unknown to be polynomialy solvable or $\mathcal{NP}$-complete when G_c is a tree. A possible candidate could be the distance to a forest of path. Let define $\nu(G)$ as the minimal cardinality of a set X of vertices of G such that every connected component of $G \setminus X$ is a path. So, an interesting question could be to know if the CTS Problem admits an $\mathcal{FPT}$ algorithm when it is parameterized by ν?

References

1. N. Alon, R. Yuster and U. Zwick. Color-coding. *Journal ACM*, 42(4):844–856, 1995.
2. J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Handbook on Scheduling.* Springer, 2007.
3. J. Blazewicz, G. Pawlak, M. Tanas, and W. Wojciechowicz. New algorithm for coupled-tasks scheduling, a survey. *RAIRO-Operation Research*, 46:335–353, 2012.
4. H.L. Bodlaender and M.R. Fellows. $W[2]$-hardness of precedence constrained k-processor scheduling. *Operations Research Letters*, 18(2):93–97, 1995.
5. H.L. Bodlaender, B.M.P. Jansen, and S. Kratsch. Cross-composition: A new technique for kernelization lower bounds. *STACS*, pages 165-176, 2011.
6. L. Cai, J. Chen, R.G. Downey, and M.R. Fellows. On the parameterized complexity of short computation and factorisation. *Archive for Mathematical Logic*, 36:321–337, 1997.
7. B. Chen, C.N. Potts, and G.J. Woeginger. *Handbook of Combinatorial Optimization (volume 3)*, Chapter A review of machine scheduling: Complexity, Algorithms and Approximability. Kluwer Academic Publishers, 1998.
8. A. Condotta and N. V. Shakhlevich. Scheduling coupled-operations jobs with exact time-lags. *Discrete Applied Mathematics*, 160:2370–2388, 2012.
9. R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science, Springer 2013.
10. M. R. Fellows and C. McCartin. On the parametric complexity of schedules to minimize tardy tasks. *Theoretical Computer Sciences*, 2(298):317–324, 2003.
11. J. Flum and M. Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science; An EATCS Series. 2006.
12. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.
13. R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling theory: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.
14. M. Habib, C. Paul, L. Viennot Partition refinement techniques: an interesting algorithmic tool kit. *International Journal of Foundations of Computer Science*, 10(2):147–170, 1999.
15. V. Lehoux-Lebacque, N. Brauner and G. Finke. Identical coupled task scheduling: polynomial complexity of the cycle cas. *Journal of Scheduling*, 18(6): 631-644, 2015.
16. D. Marx. Fixed-parameter tractable scheduling problems. *Report from Dagsthul Seminar*, 11091,1(2): pp.86, 2011.

17. M. Mnich, and A. Wiese. Scheduling and Fixed-Parameter Tractability. *Mathematical Programming*, 154(1-2): 533-562, 2015.
18. R. Niedermeier. *Invitation to Fixed Parameter Algorithms*. Volume 31 of Oxford Lectures Series in Math. and its Applications. Oxford University Press, 2006.
19. A.J. Orman and C.N. Potts. On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72:141–154, 1997.
20. J.P. Schmidt and A. Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM Journal of Computing*, 19(5):775–786, 1990.
21. R.D. Shapiro. Scheduling coupled-tasks. *Naval Research Logistics Quarterly*, 20:489–498, 1980.
22. G. Simonin, B. Darties, R. Giroudeau, and J.-C. König. Coupled-tasks in presence of bipartite compatibilities graphs In *Third International Symposium on Combinatorial Optimization*, LNCS, No. 8596 pages 161–172, 2014.
23. G. Simonin, R. Giroudeau and J.-C. König. Approximating a coupled-task scheduling problem in the presence of compatibility graph and additional tasks. *International Journal of Planning and Scheduling*, 1(4):285–300, 2013.
24. G. Simonin, B. Darties, R. Giroudeau, and J.-C. König. Theoretical aspects of scheduling coupled-tasks in presence of compatibility graph. *Algorithmic Operations Research*, 7(1):1–12, 2012.
25. R. Van Bevern, M. Mnich, R. Niedermeier and M. Weller. Interval Scheduling and Colorful Independent Sets. *Journal of Scheduling*, 18(5): 449-469, 2015.