Scheduling stretched coupled-tasks with compatibilities constraints : model, complexity and approximation results for some class of graphs

Benoit Darties, Rodolphe Giroudeau, Jean-Claude König, Gilles Simonin

Scheduling stretched coupled-tasks with compatibilities constraints : model, complexity and approximation results for some class of graphs

Benoît Darties[1], Rodolphe Giroudeau[2], Jean-Claude Konig[2], Gilles Simonin[3]

1. LE2I, UMR CNRS 6306, Université de Bourgogne, 9 Rue Alain Savary, 21 000 Dijon, France
2. LIRMM-CNRS-UMR 5506-161, rue Ada 34090 Montpellier, France
3. Insight Centre for Data Analytics, University College Cork, Ireland

Internal Research Report - LE2I

February, 12th, 2014

Abstract :
We tackle the makespan minimization coupled-tasks problem in presence of compatibility constraints. In particular, we focus on stretched coupled-tasks, *i.e.*coupled-tasks having the same sub-tasks execution time and idle time duration. We study severals problems in frame works of classic complexity and approximation for which the compatibility graph G_c is bipartite (star, chain, ...) In such context, we design some efficient polynomial-time approximation algorithms according to difference parameters of the scheduling problem. When G_c is a k-stage bipartite graph, we propose, among other, a $\frac{7}{6}$-approximation algorithm when $k = 1$, and a $\frac{13}{9}$-approximation algorithm when $k = 2$.

keyword: coupled-tasks, scheduling, complexity, approximation algorithm, compatibility graph

---

Résumé :
Nous nous intéressons au problème de minimiser le temps d'ordonnancement d'un ensemble de tâches couplées en présence de contraintes de compatibilité. En particulier, nous étudions les tâches couplées *étirées* et montrons que le problème est difficile même lorsque le graphe G_c modélisant les contraintes de compatibilité est une étoile. Nous concentrons notre étude lorsque G_c est un graphe biparti k-étapes, et proposons plusieurs résultats d'approximation, notamment un algorithme $\frac{7}{6}$-approché lorsque $k = 1$, et $\frac{13}{9}$-approché pour $k = 2$.

mot-clés: tâches couplées, ordonnancement, complexité, algorithmes d'approximation, graphe de compatibilité

1 Introduction

The detection of an object by a radar system generally uses the following process: a transmitter emits a pulse in some direction which propagates though the environmental medium. If the pulse encounters an object, it is reflected back to the transmitter. Using the transmit time and the direction of the pulse, the transmitter can compute the position of the object. Formally this process is divided into three parts: (1) a first operation of duration a a sensor emits the pulse; (2) then the system waits for a fixed amount of time L the propagation of the pulse and its potential reflexion; (3) then in a second operation of duration b the sensor listen to any pulse echo to conclude of the presence or not of an object and compute its position. Due to the nature of the application, the system works in a non-preemptive mode. Varying the values of parameters a, b and L allows, among others, to adapt the detection range. On mono-processor systems, the idle processing time L can be reused to perform other operations, *i.e.*to schedule another object detection process using another sensor.

Scheduling issues appear when several sensors using different frequencies can work in parallel, while acquisitions using the same frequency have to be delayed in order to avoid interferences. Two acquisition processes i and j are said *compatible* if they can work in parallel.

We consider in this paper a mono-processor system using several sensors, some of them using the same frequencies. Given a set of data acquisitions with they duration and the list of compatible acquisitions, finding a optimal schedule which minimizes the makespan is a problem hard to solve in general, even under restricted hypothesis on the values of a, b, L and/or on the list of compatible acquisitions. We study the variation of the complexity when for any acquisition i, the durations of each of its operations and the idle time between them are equal. We propose exact and approximation results according to different hypothesis we made on the list of compatible acquisitions.

This article is organized as follows: first we present the general coupled-task model, a natural way to model such a data acquisition process, and the related work. In the next section we introduce the stretched coupled-tasks model and summarize the contribution of this paper. The computational complexity results are detailed in Section 4, while Section 5 focuses on polynomial-time approximation algorithms with performance guarantee for $\mathcal{NP}$-Hard instances

2 Presentation of coupled-tasks and related work

A natural way to model data acquisition process presented in introduction is to use *coupled-tasks*, introduced first by Shapiro [11]: each acquisition task is a coupled-task $A_i = (a_i, L_i, b_i)$ composed by two sub-tasks of processing time a_i and b_i, respectively dedicated for wave transmission and echo reception. Between these two sub-tasks there is a fixed idle time L_i which represents the spread of the echo in the medium. We work in a non-preemptive mode: once started, a sub-task cannot be stopped and then continued later. A valid schedule implies here that for any task started at t, the first sub-task is fully executed between t and $t+a_i$, and the second between $t+a_i+L_i$ and $t + a_i + L_i + b_i$. We note $\mathcal{A} = \{A_1, \dots, A_n\}$ the collection of coupled-tasks to be scheduled.

Two tasks A_i and A_j are said *compatible* if they use different wave frequencies; any sub-task of A_i may be executed during the idle time of A_j or reciprocally. A valid schedule implies here that for any tasks A_i and A_j, if either the first and/or the second sub-task of A_i is scheduled during the idle time of A_j, then A_i and A_j must be two compatible tasks. For clarity we say that A_i and A_j are executed in parallel in such a schedule. The parallel execution of A_i and A_j may exist under to configurations, according to the values of a_i, L_i, b_i, a_j, L_j, b_j. A graph $G = (\mathcal{A}, E)$ is used to model such this compatibility, where edges from E link any pair of compatible coupled-tasks.

Due to the combinatory of the parameters of the problem, we use the Graham's notation scheme $\alpha|\beta|\gamma$ [8] (respectively the machine environment, job characteristic and objective function) to characterize the problems related to coupled-tasks. The job characteristics summarizes the conditions made on the values of a_i, L_i, b_i (independent between tasks, or equal to a constant), and the shape of the compatibility graph G. The coupled-tasks scheduling problems under compatibility constraints has been studied in the framework of classic complexity and approximation (see Table 1 - only main results are retained).

(a_i, L_i, b_i)	Complexity	Approximation	Ref.
(a_i, a_i, a_i)	$\mathcal{NPC}$	$3/2$	[13]
(p, L, p)	$\mathcal{NPC}$	$7/4 + \frac{L}{4p}$	[15]
(a, L, b)	$\mathcal{NPC}$ if $L \geq a + b$ else $\mathcal{P}oly$	$\frac{3a+2b}{2a+2b}$	[12]
$(1, 2, 1)$	$\mathcal{NPC}$	$\frac{10}{9}$ if G_c is triangle free else $\frac{13}{12}$	[14]
(p, p, b_i) or (a_i, p, p)	$\mathcal{P}oly$		[15]

Table 1: Computational complexity and polynomial-time approximation algorithms for $1|(a_i, L_i, b_i), G_c|C_{max}$ according to the triplet (a_i, L_i, b_i).

3 Stretched coupled-task: model and contribution

3.1 Model

This paper focuses on *stretched* coupled-tasks, *i.e.*coupled-tasks for what the durations of the first sub-task, the second sub-task and the idle time are equal to a stretch-factor applied to an original task $(a_i, L_i, b_i) = (1, 1, 1)$. Formally, a stretched coupled-task A_i is a task such that $a_i = L_i = b_i = \alpha(A_i)$, where $\alpha(A_i)$ is the stretch factor of the task. In the rest of the paper, coupled-tasks are always stretched coupled-tasks, and noted A_i when we need to refer to the values a_i, b_i and L_i, or with a single identifier, *i.e.* x, otherwise. In such configuration, for two compatible tasks A_j and A_j to be scheduled in parallel, one of the following conditions must hold:

1. either $\alpha(A_i) = \alpha(A_j)$: then the idle time of one task is fully exploited to schedule a sub-task from the other (i.e. b_i is scheduled during L_j, and a_j is scheduled during L_i), and the completion of the two tasks is done without idle time.

2. or $3\alpha(A_i) \leq \alpha(A_j)$: then task A_i is fully executed during the idle time L_j of A_j. For sake of simplify, we say we *pack* A_i into A_j.

Topology	Complexity	Approximation
$uug(G_c)$=Star graph	$\mathcal{NP}-\mathcal{C}$ (Theorem 2)	$\mathcal{FPTAS}$ (Theorem 6)
$uug(G_c)$=Chain graph	$O(n^3)$ (Theorem 3)	
G_c= 1-stage bipartite, $\Delta(G_c)=2$	$O(n^3)$ (Theorem 4)	
G_c= 1-stage bipartite, $\Delta(G_c)=3$	$\mathcal{NP}-\mathcal{C}$ (Theorem 5)	$\frac{7}{6}$-$\mathcal{APX}$(Theorem 7)
G_c= complete 1-stage bipartite	$\mathcal{NP}-\mathcal{C}$ (see [13])	$\mathcal{PTAS}$ (Theorem 7)
G_c= complete 1-stage bipartite with constraint $\alpha(x)=\alpha(y)$, $\forall x,y\in X_1$	$\mathcal{NP}-\mathcal{C}$ (see [13])	$\mathcal{PTAS}$ (Theorem 7) $\frac{13}{12}$-$\mathcal{APX}$(Theorem 7)
G_c= 2-stage bipartite	$\mathcal{NP}-\mathcal{C}$ (Theorem 5)	$\frac{13}{9}$-$\mathcal{APX}$ (Theorem 8)

Table 2: Complexity and approximation results.

From this observation, one can deriver from the compatibility graph $G=(\mathcal{A},E)$ a directed compatibility graph $G_c=(\mathcal{A},E_c)$ by assigning a direction to each edge E from the task with the lowest stretch factor to the task with the highest one. If two compatible tasks x and y have the same stretch factor, then E_c contains both the arc (x,y) and the arc inverted (y,x). Remark that if for any pair of compatible tasks x and y we have $\alpha(x)\neq\alpha(y)$, then G_c is a directed acyclic graph.

Note that when the job characteristics refer to an undirected topology for the compatibility graph (i.e. star, chain), we consider in fact a graph G_c such that their undirected underlying graph $uuc(G_c)$ correspond to the given class.

Given a valid schedule σ and a task A_i, we note $\sigma(A_i)$ the date when A_i is being executed, *i.e.*the first (resp. second) sub-task is executed between $\sigma(A_i)$ and $\sigma(A_i)+a_i$ (resp. between $\sigma(A_i)+a_i+L_i$ and $\sigma(A_i)+a_i+L_i+b_i$). We also denote by $seq(W)$ the sum of the processing time of the tasks in any set W:

$$seq(W)=3\sum_{x\in W}\alpha(x)$$

Remark that, when W is an independent set for G_c, the cost of any optimal schedule is at least $seq(W)$. We note $N_G(v)$ the neighborhood of v in G. We note $d_G(v)=|N(x)|$ the degree of v in G, and Δ_G the maximum degree of G.

As we focus our work on bipartite graphs, we recall that a *k-stage bipartite graph* is a digraph $G=(V_0\cup\cdots\cup V_k, E_1\cup\cdots\cup E_k)$ where $V_0\ldots V_k$ are disjoint vertex sets, and each arc in E_i is from a vertex in V_i to a vertex in V_{i+1}. The vertex of V_i are said to be at rank i, and the subgraph $G_i=(V_{i-1}\cup V_i, E_i)$ is called the i-th stage of G, and we write $G=G_1+\cdots+G_k$. Note that G is acyclic, and that vertices from V_0 are always source in G (nodes only incident to outgoing arcs) , while vertices from V_k are sink (nodes only incident to ingoing arcs). For clarity, a 1-stage bipartite graphs may be referred as triplet (X,Y,E).

3.2 Contribution

We define the main problem of this study as $1|a_i=L_i=b_i=\alpha(A_i),G_c|C_{max}$, study the variation of the complexity when G_c or $uug(G_c)$ varies, and propose approximation results for instances hard to solve. The results proved in this article are summarized in Table 2.

3.3 Prerequisites

We use in this paper known complexity results on four packing-related problems:

1. The SUBSET SUM (SS) problem is a known problem in which, given a set $\mathcal{S}$ of n positive values and $v \in \mathbb{N}$, one ask is there exists a subset $\mathcal{S}^* \subseteq \mathcal{S}$ such that $\sum_{i \in \mathcal{S}^*} i = v$. This decision problem is well-known to be $\mathcal{NP}$-complete (see [6]). The optimization version problem is sometimes view as a KNAPSACK problem, where each item profits and weights coincide to a value in S, the knapsack capacity is v, and one try to find the set of packable items with maximum profit.

2. The MULTIPLE SUBSET SUM (MSS) problem is a variant of well-known BIN PACKING in which a number of identical bins is given and one would like to maximize the overall weight of the items packed in the bins such that the sum of the item weights in every bin does not exceed the bin capacity. The problem is also a special case of the MULTIPLE KNAPSACK problem in which all knapsacks have the same capacities and the item profits and weights coincide. Caprara, and al. [2] proved that MSS admits a $\mathcal{PTAS}$, but does not admit a $\mathcal{FPTAS}$ even for only two knapsacks. They also proposed a $\frac{3}{4}-$approximation algorithm in [3].

3. MULTIPLE SUBSET SUM WITH DIFFERENT KNAPSACK CAPACITIES (MSSDC) [1] is an extension of MSS considering different bin capacities. MSSDC also admits a $\mathcal{PTAS}$ [1].

4. As a generalization of MSSDC, MULTIPLE KNAPSACK ASSIGNMENT RESTRICTION (MKAR) problem consists to packs weighted items into non-identical capacity-constrained bins, with the additional constraints that each item can be packed in some bins only. Each item as a profit, the objective here is to maximize the sum of profits of packed items. Considering the profit of each item equals its weight, [4] proposed a $\frac{1}{2}$-approximation.

We also use a known result concerning a variant of the $\mathcal{NP}$-complete problem 3SAT [6], denoted subsequently by ONE-IN-(2,3)SAT(2,$\bar{1}$): this problem aim to ask is there exists an assignment of n boolean variables, with $n \mod 3 \equiv 0$, which satisfies a set of n clauses of cardinality 2 and $n/3$ clauses of cardinality 3 such that:

- Each clause of cardinality 2 is equal to $(x \vee \bar{y})$ for some $x,\ y \in \mathcal{V}$ with $x \neq y$.
- Each of the n literals x (resp. of the literals $\bar{x}$) for $x \in \mathcal{V}$ belongs to one of the n clauses of cardinality 2, thus to only one of them.
- Each of the n (positive) literals x belongs to one of the $n/3$ clauses of cardinality 3, thus to only one of them.
- Whenever $(x \vee \bar{y})$ is a clause of cardinality 2 for some $x,\ y \in \mathcal{V}$, then x and y belong to different clauses of cardinality 3.

Question: Is there a truth assignment $I : \mathcal{V} \rightarrow \{0, 1\}$ whereby each clause in $\mathcal{C}$ has exactly one true literal?

Example: The following logic formula is a valid instance of ONE-IN-(2,3)SAT(2,$\bar{1}$): $(x_0 \vee x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\bar{x}_0 \vee x_3) \wedge (\bar{x}_3 \vee x_0) \wedge (\bar{x}_4 \vee x_2) \wedge (\bar{x}_1 \vee x_4) \wedge (\bar{x}_5 \vee x_1) \wedge (\bar{x}_2 \vee x_5)$.

The answer to ONE-IN-(2,3)SAT(2,$\bar{1}$)is *yes*. It is sufficient to choose $x_0 = 1$ (1 for true), $x_3 = 1$ and $x_i = 0$ (0 for false) for $i = \{1, 2, 4, 5\}$. This yields a truth assignment that satisfies the formula, and there is exactly one true literal for each clause. The proof of the $\mathcal{NP}$-completeness is given in [7].

4 Computational complexity

We present severals $\mathcal{NP}-$complete and polynomial results. We first show the problem is $\mathcal{NP}$-hard even when the compatibility graph is a star (Theorem 2), but then show it is solvable with a $O(n^3)$ time complexity algorithm when G is a chain (Theorem 3). Then we focus our analysis when G_c is a 1-stage bipartite graph. We prove the problem is solvable with a $O(n^3)$ polynomial algorithm if $\Delta_G = 2$ (Theorem 4), but becomes $\mathcal{NP}$-hard when $\Delta_G = 3$ (Theorem 5).

Theorem 1 *The problem $1|a_i = L_i = b_i = \alpha(A_i), G = star|C_{max}$ is polynomial if the central node admits at least one outcoming arc.*

Proof If it exists a least one outgoing arc from the central node x, then the optimal solution consists in executing the central node in one coupled-task y such that $(x, y) \in G_c$. The remaining tasks are processed sequentially after the completion of the y-task. □

Theorem 2 *The problem $1|a_i = L_i = b_i = \alpha(A_i), G = star|C_{max}$ is $\mathcal{NP}$-hard if the central node admits only incoming arc.*

Proof It is easy to see that $1|\alpha(A_i) = a_i = L_i = b_i, G = star|C_{max}$ is in $\mathcal{NP}$. We propose a reduction to SS problem. From an instance of SS composed by a set $\mathcal{S}$ of n positive values and $v \in \mathbb{N}$ (with $v \geq x, \forall x \in S$), we construct an instance of $1|star, \alpha_i = a_i = L_i = b_i|C_{max}$ in the following way:

1. For each value $i \in \mathcal{S}$ we introduce a coupled-task x with $\alpha(x) = i$. Let $\mathcal{T}$ be the set of these tasks.
2. We add a task y with $\alpha_y = a_y = L_y = b_y = 3v$.
3. We define a compatibility constraint between each task $x \in \mathcal{T}$ and y.

Clearly the compatibility graph G is a star with y as the central node. The transformation is clearly polynomial. It easy to see that $1|\alpha(A_i) = a_i = L_i = b_i, G = star|C_{max}$ is $\mathcal{NP} - hard$ as following:

- Considering the characteristics of the instance when G_c is a star, any (optimal) valid scheduling consists in scheduling sequentially a subset $\mathcal{T}' \subseteq \mathcal{T}$ of task during the idle time of y, and in scheduling after this the other tasks from T sequentially. Then the optimal schedule would consist in maximizing $w = 3\sum_{t\in\mathcal{T}'} \alpha_t$ under the constraint $w \leq L_y$, and in producing a schedule with a total length equal to the time to schedule y (i.e. $3\alpha_y$) plus the time to schedule tasks not executed during L_y the idle time of y (ie. $3\sum_{t\notin\mathcal{T}'} \alpha_t$); in other words a schedule of time $3\alpha_y + 3\sum_{t\in\mathcal{T}} \alpha_t - 3\sum_{t\in\mathcal{T}'} \alpha_t$.

  If one can find an optimal schedule of length $3\sum_{t\in\mathcal{T}} \alpha_t + 2\alpha_y$, then one can exhibit a subset $\mathcal{T}^* \subseteq \mathcal{T}$ with $3\sum_{t\in\mathcal{T}^*} \alpha_t = L_y = 3v$ and by construction one can deduce a solution to SSby taking $\mathcal{S}^* = \left\{ \cup\, \alpha_x | x \in \mathcal{T}^* \right\}$.

- Conversely, if one can exhibit a subset $\mathcal{S}^* \subseteq \mathcal{S}$ thus that $\sum_{i \in \mathcal{S}^*} i = v$, then one can produce an optimal schedule by executing sequentially tasks x with $\alpha_x = i$, where $i \in \mathcal{S}^*$, during the idle time $L_y = 3v$ of task y, and by executing sequentially tasks $z \in \mathcal{S}/\mathcal{S}^*$ immediately after the execution of y.

$\square$

Theorem 3 *The problem* $1|a_i = L_i = b_i = \alpha(A_i), G = chain|C_{max}$ *admits a polynomial-time algorithm.*

Proof When the compatibility graph is a chain, compatibility constraints require tasks to be executed either alone, by pair only, or two consecutively tasks in another big one. The last case occurs only when a vertex x of degree two, called peak, has its two neighbor y and z which can be entirely executed in the inactivity time of x. Thus if $3\alpha_y + 3\alpha_z \leq \alpha_x$, then one can execute y and z during the idle time of x; The schedule length for this block is exactly $3\alpha_x$. We can observe that the peaks can not be executed in another tasks. Therefore, w.l.o.g. we can assume that there exists an optimal solution where the peak tasks and their neighbors are executed together.

From this result, we can in polynomial time modify G in G_m where all peak vertices, their neighbors and the associated edges are removed. Thus G_m is a collection of chains and the best scheduling associated to this graph requires tasks to be executed either alone or by pair.

Given x and y two compatibles tasks, only the two following configurations allow them to be scheduled pairwise (by blocks):

1. if $\alpha_x = \alpha_y$, then one can execute a_y during the idle time of x and b_x during the idle time of y. The makespan for this block is exactly $4\alpha_x$.
2. if $\alpha_x \leq \frac{\alpha_y}{3}$, then one can execute entirely x during the idle time of y, the makespan for this block is exactly $3\alpha_y$, including an inactivity period of $\alpha_y - 2\alpha_x$.

By weighting each edge of the graph G_m with the sequential time of the overlap of the two tasks which form the edge, our problem has a solution if we find a matching that minimizes the weight of the matching edges and the isolated vertices.

This problem can be solved in a polynomial-time by reducing the problem to the search for a minimum weighted perfect matching. This problem can be polynomially solved in $O(n^2m)$ time complexity [5]. In order to obtain a graph with even number of vertices and such that finding a perfect matching is possible, we construct a graph $H_c = (V_H, E_H, w)$ and define a weighted function $w : E \rightarrow \mathbb{N}$ as follows:

1. Let $\mathcal{I}_1$ be an instance of our problem with a compatibility graph $G_m = (V_m, E_m)$, and $\mathcal{I}_2$ an instance of the minimum weight perfect matching problem in graph constructed from $\mathcal{I}_1$. We consider a graph H_c, consisting of two copies of G_m denoted $G'_m = (V'_m, E'_m)$ and $G''_m = (V''_m, E''_m)$. The vertex corresponding to $x \in V_m$ is denoted x' in G'_m and x'' in G''_m. Moreover, $\forall i = 1, \ldots, n$, an edge $\{x', x''\}$ in E_H is added and we state $w(\{x', x''\}) = 3\alpha'_x$. This weight represents the sequential time of the task alone x'. We have $H_c = G'_m \cup G''_m = (V'_m \cup V''_m, E'_m \cup E''_m)$, with $|V'_m \cup V''_m|$ of even size.
2. For two compatibles tasks x' and y' with $3\alpha_{x'} \leq \alpha_{y'}$ or $3\alpha_{y'} \leq \alpha_{x'}$, we add the edges $\{x', y'\}$ and $\{x'', y''\}$ in E and we state $w(\{x', y'\}) = w(\{x'', y''\}) = \frac{\mathbf{3 \times max\{\alpha_{x'}, \alpha_{y'}\}}}{\mathbf{2}}$.

3. For two compatibles tasks x' and y' with $\alpha_{x'} = \alpha_{y'}$, we add the edges $\{x', y'\}$ and $\{x'', y''\}$ in E, and we state $w(\{x', y'\}) = w(\{x'', y''\}) = \frac{4\times\alpha_{x'}}{2}$.

In order to provide a polynomial-time algorithm solving our problem, we will prove firstly the following proposition.

Proposition 1 *For a minimum weight perfect matching C, we can associate a schedule of minimum processing times C and vice versa.*

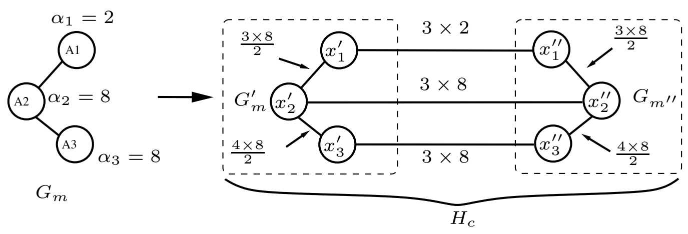

Figure 1: Example of the transformation

Proof

By construction H_c contains an even number of vertices, and the fact that each vertex of G'_c is connected to an equivalent vertex in G''_c, finding a perfect matching on the graph H_c is possible. This means that there exists a schedule such that each task is executed only once time. Note that the matching in G'_c is not necessarily identical to the one in G''_c, but they still have the same weight. The makespan obtained is equal to the sum of the processing times of the obtained blocks and those of isolated tasks. And since each block has an execution time equal to the weight of the equivalent edge in the perfect matching, we have the sum of edges weights of the matching which is equal to the blocks sum of the scheduling obtained.

Thus, for a minimum weight perfect matching C, we can associate a schedule of minimum length C and vice versa. This ends the proof of the Proposition 1.

□

Proof continuation of Theorem 3

The proposition 1 shows the relationship between a solution to our problem with G_m and a solution of a minimum weight perfect matching in H_c. However, the Edmonds algorithm can find a minimum weight perfect matching in $O(n^2m)$ [5]. So the optimization problem with G_m is polynomial, and if one adds the execution of the blocks created by removed vertices, this leads to the problem $1|a_i = L_i = b_i = \alpha(A_i), G = chain|C_{max}$ is polynomial.

□

In following, we study the variation of the complexity in the case of the compatibility graph is oriented in presence of a 1-stage bipartite graph according to the different values.

Theorem 4 *The problem of deciding whether an instance of* $1|a_i = L_i = b_i = \alpha(A_i), G_c =$*1−stage bipartite*$, \Delta_{G_c} = 2|C_{max}$ *is polynomial.*

Proof Let $G_c = (X, Y, E)$ be a 1-stage bipartite compatibility graph. Y-tasks will always be scheduled sequentially. The aim is to fill their idle time with a maximum of tasks of X, while the remained tasks will be executed after the Y-tasks. We just have to minimize the length of the remained tasks. Note that $d_{G_c}(y) \leq 2$. The algorithm use three steps :

1. for each task $y \in Y$ such that $3\alpha(x_1) + 3\alpha(x_2) \leq \alpha(y)$ where x_1 and x_2 are the only two neighbors of Y, we add y to the schedule and execute $x1$ and x_2 sequentially during the idle time of y. Then we remove y, x_1 and x_2 from the instance.
2. Each remaining task $y \in Y$ admits at most two incoming arcs (x_1, y) and / or (x_2, y). We add a weight $\alpha(x)$ to the arc (x, y) for each $x \in N(y)$, then perform a maximum weight matching on G_c in order to minimize the length of the remained tasks of X. Thus, the matched coupled-tasks are executed, and these tasks are removed from G_c.
3. Then, remaining tasks from X are allotted sequentially after the other tasks.

The complexity of an algorithm is $O(n^3)$. □

Theorem 5 *The problem of deciding whether an instance of* $1|a_i = L_i = b_i = \alpha(A_i), G_c = 1{-}stage\ bipartite, \Delta_{G_c} = 3|C_{max}$ *has a schedule of length at most* $54n$ *is* $\mathcal{NP}$*-complete with* n *the number of tasks.*

Proof It is easy to see that $1|a_i = L_i = b_i = \alpha(A_i), G_c = 1{-}stage\ bipartite, \Delta_{G_c} = 3|C_{max} = 54n \in \mathcal{NP}$. Our proof is based on a reduction from ONE-IN-(2,3)SAT(2,$\bar{1}$): given a set $\mathcal{V}$ of n boolean variables with $n \mod 3 \equiv 0$, a set of n clauses of cardinality two and $n/3$ clauses of cardinality three, we construct an instance π of the problem $1|a_i = L_i = b_i = \alpha(A_i), G_c = 1{-}stage\ bipartite, \Delta_{G_c} = 3|C_{max} = 54n$ in following way (Figure 2 illustrates the construction):

1. For all $x \in \mathcal{V}$, we introduce four variable-tasks: x, x', $\bar{x}$ and $\bar{x}'$ with $(a_i, L_i, b_i) = (1, 1, 1), \forall i \in \{x, x', \bar{x}, \bar{x}'\}$. This variable-tasks set is noted $\mathcal{VT}$.
2. For all $x \in \mathcal{V}$, we introduce three literal-tasks $\mathcal{L}_x, C^x$ and $\bar{C}^x$ with $\mathcal{L}_x = (2, 2, 2); C^x = \bar{C}^x = (6, 6, 6)$. The set of literal-tasks is denoted $\mathcal{LT}$.
3. For all clauses with a length of three, we introduce two clause-tasks C^i and $\bar{C}^i$ with $C^i = (3, 3, 3)$ and $\bar{C}^i = (6, 6, 6)$.
4. For all clauses with a length of two, we introduce one clause-task C^i with $C^i = (3, 3, 3)$. The set of clause-tasks is denoted $\mathcal{CT}$.
5. The following arcs model the compatibility constraints:
   (a) For all boolean variables $x \in \mathcal{V}$, we add the arcs $(\mathcal{L}_x, C^x)$ and $(\mathcal{L}_x, \bar{C}^x)$
   (b) For all clauses with a length of three denoted $C_i = (y \vee z \vee t)$, we add the arcs (y, C^i), (z, C^i), (t, C^i) and $(\bar{y}', \bar{C}^i)$, $(\bar{z}', \bar{C}^i)$, $(\bar{t}', \bar{C}^i)$.
   (c) For all clauses with a length of two denoted $C_i = (x \vee \bar{y})$, we add the arcs (x', C^i) and $(\bar{y}, C^i)$.
   (d) Finally, we add the arcs (x, C^x), (x', C^x) and $(\bar{x}, \bar{C}^x)$ and $(\bar{x}', \bar{C}^x)$.

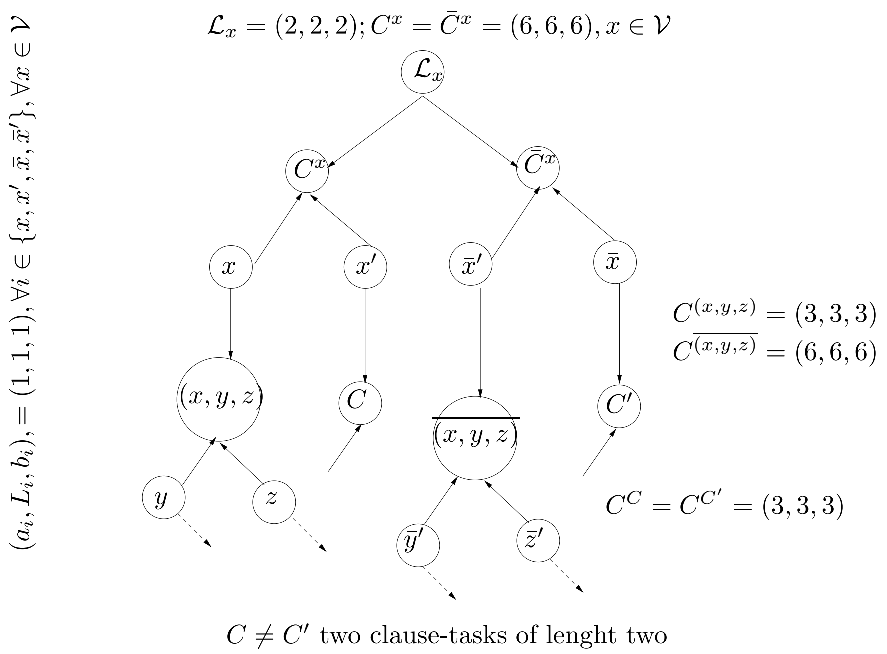

Figure 2: A partial compatibility graph for the $\mathcal{NP}$-completeness of the scheduling problem $1|bipartite\ of\ depth\ one, d(G_c) \leq 3, \alpha_i = a_i = L_i = b_i|C_{max}$

This transformation can be computed clearly in polynomial time. The proposed compatibility graph is 1-stage bipartite and $d_{G_c}(x) \leq 3, \forall x \in \mathcal{VT} \cup \mathcal{LT} \cup \mathcal{CT}$.

In follows, we say that a task x is merged to a task y, if it exists a compatibility constraint from x to y; $i.e.$the coupled-task x may be executed during the idle of coupled-task y.

- Let us first assume that there is a schedule with length of $54n$ at most. We prove that there is a truth assignment $I : \mathcal{V} \rightarrow \{0,1\}$ such that each clause in $\mathcal{C}$ has exactly one true literal (i.e. one literal equal to 1). We make some essentials remarks:

  1. The length of the schedule is given by an execution time of the coupled-tasks admitting only incoming arcs, and the value is $54n = 3\alpha_{\mathcal{CT}}|\mathcal{CT}| + \alpha_{\mathcal{LT}}(|\mathcal{LT}| - |\{\mathcal{L}_x, x \in \mathcal{V}\}|) = 9|\{C^i \in \mathcal{CT}$ of length 2 and 3$\}| + 18|\{\bar{C}^i \in \mathcal{CT}\}| + 18|\{C^x$ and $\bar{C}^x \in \mathcal{LT}\}| = 9 \times \frac{4n}{3} + 18 \times \frac{n}{3} + 18 \times 2n$.
     Thus, all tasks from $\mathcal{VT} \cup \{\mathcal{L}_x, x \in \mathcal{V}\}$ must be merged with tasks from $\mathcal{CT} \cup (\mathcal{LT} - \{\mathcal{L}_x, x \in \mathcal{V}\})$.
  2. By the construction, at most three tasks can be merged together.
  3. $\mathcal{L}_x$ is merged with C^x or $\bar{C}^x$.
  4. The allocation of coupled-tasks from $\mathcal{CT} \cup (\mathcal{LT} - \{\mathcal{L}_x, x \in \mathcal{V}\})$ leads to $18n$ idle time. The length of the variable-tasks $\mathcal{VT}$ and $\mathcal{L}_x$ equals $18n$ (in these coupled-tasks there are $6n$ idle times).
  5. If the variable-tasks x and x' are not merged simultaneously with C^x, $i.e.$only one of these tasks is merged with C^x, so, by with the previous discussion, it is necessary to merge a literal-task $\mathcal{L}_y$, with $x \neq y$ one variable-task ($\bar{y}$ or $\bar{y}'$) with C^y or $\bar{C}^y$. It is impossible by size of coupled-tasks. In

the same ways, the variable-tasks $\bar{x}$ et $\bar{x}'$ are merged simultaneously with $\bar{C}^x$.

6. Hence, first x and x' are merged with C^x or with clause-task where the variable x occurs. Second, $\bar{x}$ and $\bar{x}'$ are merged with $\bar{C}^x$ or a clause-task.

So, we affect the value "true" to the variable l iff the variable-task l is merged with clause-task(s) corresponding to the clause where the variable l occurs. It is obvious to see that in the clause of length three and two we have one and only one literal equal to "true".

- Conversely, we suppose that there is a truth assignment $I : \mathcal{V} \rightarrow \{0, 1\}$, such that each clause in $\mathcal{C}$ has exactly one true literal.

  - If the variable $x = true$ then we merged the vertices $\mathcal{L}_x$ with C^x; x with the clause-task C^i corresponding to the clause of length three which x occurs; x' with the clause-task C^i corresponding to the clause of length two which x occurs; and $\bar{x}, \bar{x}'$ with $\bar{C}^x$.
  - If the variable $x = false$ then we merged the vertices $\mathcal{L}_x$ with $\bar{C}^x$; $\bar{x}$ with the clause-task corresponding to the clause of length two which $\bar{x}$ occurs; $\bar{x}'$ with the clause-task $\bar{C}^i$ corresponding to the clause (C) of length three which x occurs; and x, x' with C^x.

  For a feasible schedule, it is sufficient to merge vertices which are in the same partition. Thus, the length of the schedule is at most $54n$.

□

5 Polynomial-time approximation algorithms

5.1 Star graph

Theorem 6 *The problem* $1|a_i = L_i = b_i = \alpha(A_i), G = star|C_{max}$ *admits a* $\mathcal{FPTAS}$.

Proof The central node admits only incoming arcs (the case of the central node admits at least one outcoming arc is given by Corollary 1). Therefore, we may use the solution given by the SUBSET SUM (SS) (see [9] and [10])). Indeed, the schedule is follows: the central node is executed first with the coupled-tasks chosen by an $\mathcal{FPTAS}$ algorithm, the remaining tasks are processed after the completion of the central node.

□

5.2 1−stage bipartite graph

Scheduling coupled-tasks during the idle time of others can be related to packing problems, especially when the constraint graph G_c is a bipartite graph. In the following, we propose several approximation when G_c is a 1−stage bipartite graph.

Lemma 1 *Let* $\mathcal{P}$ *be a problem with* $\mathcal{P} \in \{\text{MKAR MSSDC}, \text{MSS}\}$ *such that* $\mathcal{P}$ *admits a* ρ*-approximable then the following problems*

1. $1|a_i = L_i = b_i = \alpha(A_i), G_c = 1$*-stage bipartite*$|C_{max}$*,*

2. $1|\alpha_i = a_i = L_i = b_i,$ *complete bipartite*$|C_{max}$

3. $1|\alpha_i = a_i = L_i = b_i,$ *complete bipartite*$|C_{max}$ *where the constraint graph is a complete bipartite G=(X, Y), and all the tasks from* Y *have the same* $\alpha(y)$*.*

is approximable to a factor $1 + \frac{(1-\rho)}{3}$.

Proof

1. Let consider an instance of $1|\alpha_i = a_i = L_i = b_i, G_c = 1-$*stage bipartite*$|C_{max}$ such that $G_c = (X, Y, E)$, where $X \cup Y$ are coupled-tasks, and by a stretch factor function $\alpha : X \cup Y \rightarrow \mathbb{N}$, and arcs from E model the constraints between tasks.

   In such instance, any valid schedule consists to find for each task $y \in Y$ a subset of compatible tasks $X_y \subseteq X$ to pack into $y \in Y$, each task of x being packed at most once. Let $X_p = \cup_{y \in Y} X_y$ be the union of tasks of X packed into a task from Y. Let $X_{\bar{p}}$ the set of remaining tasks, with $X_{\bar{p}} = X/X_p$. Obviously, we have:
$$seq(X_p) + seq(X_{\bar{p}}) = seq(X) \tag{1}$$

   As Y is an independent set in G, tasks from Y have to be scheduled sequentially in any (optimal) solution. The length of any schedule S is then the time to execute entirely tasks from Y plus the length to schedule sequentially the tasks from $X_{\bar{p}}$. Formally:
$$C_{max}(S) = seq(Y) + seq(X_{\bar{p}}) \tag{2}$$

   From Eq. (1) and (2) we have:
$$C_{max}(S) = seq(Y) + seq(X) - seq(X_p). \tag{3}$$

   We use here a reduction to MKAR: each task x from X is an item having a weight $3.\alpha(x)$, each task from Y is a bin with capacity $\alpha(y)$, and each item x can be packed on y if and only if the edge $\{x, y\}$ belong to the bipartite graph.

   Using algorithms and results from the literature, one can obtain an assignment of some items into bins, and note X_p the set of packed items. The cost of the solution for the MKARproblem is $seq(X_p)$. If MKAR is approximable to a factor ρ, then we have :
$$seq(X_p) \geq \rho \times seq(X_p^*), \tag{4}$$
   where X_p^* is the set of packable items with the maximum profit. Combining Eq. (3) and (4) , we obtain a solution for $1|\alpha_i = a_i = L_i = b_i,$ *bipartite*$|C_{max}$ with a length:
$$C_{max}(S) \leq seq(Y) + seq(X) - \rho \times seq(X_p^*) \tag{5}$$

   As X and Y are two fixed sets, a optimal solution S^* with minimal length $C_{max}(S^*)$ is obtained when $seq(X_p)$ is maximum, *i.e.*when $X_p = X_p^*$. The length of any optimal solution is here:
$$C_{max}(S^*) = seq(Y) + seq(X) - seq(X_p^*) \tag{6}$$

Using Eq. (5) and (6), the ratio obtained between our solution S and the optimal one S^* is :

$$\frac{C_{max}(S)}{C_{max}(S^*)} \leq \frac{seq(Y) + seq(X) - \rho \times seq(X_p^*)}{seq(Y) + seq(X) - seq(X_p^*)} \leq 1 + \frac{(1-\rho) \times seq(X_p^*)}{seq(Y) + seq(X) - seq(X_p^*)} \quad (7)$$

By definition, $X_p^* \subseteq X$. Moreover, as the processing time of X_p^* cannot excess the idle time of tasks from Y, we obtain:

$$seq(X_p^*) \leq \frac{1}{3} seq(Y) \quad (8)$$

Combined to Eq. (7) , we obtain the following upper bound:

$$\frac{C_{max}(S)}{C_{max}(S^*)} \leq 1 + \frac{(1-\rho)}{3}. \quad (9)$$

We obtain the desired result.

2. For the problem $1|\alpha_i = a_i = L_i = b_i, complete\ bipartite|C_{max}$, the proof is identical keeping in mind that MSSDC is a special case of MKAR where each item can be packed in any bin.

3. For the problem $1|\alpha_i = a_i = L_i = b_i, complete\ bipartite|C_{max}$ where the constraint graph is a complete bipartite $G = (X, Y)$, and all the tasks from Y have the same $\alpha(y)$, the proof is identical as previously since MSSDC is a generalization of MSS.

□

Theorem 7 *The following problems admits a polynomial-time approximation algorithms:*

1. *The problem $1|a_i = L_i = b_i = \alpha(A_i), G_c = 1\text{-}stage\ bipartite|C_{max}$ is approximable to a factor $\frac{7}{6}$.*
2. *The problem $1|a_i = L_i = b_i = \alpha(A_i), G_c = complete\ 1\text{-}stage\ bipartite|C_{max}$ admits a $\mathcal{PTAS}$.*
3. *The problem $1|a_i = L_i = b_i = \alpha(A_i), G_c = complete\ 1\text{-}stage\ bipartite|C_{max}$, where all the tasks from Y have the same stretch factor $\alpha(y)$:*
   (a) is approximable to a factor $\frac{13}{12}$.
   (b) admits a $\mathcal{PTAS}$.

Proof

1. Authors from [4] proposed a $\rho = \frac{1}{2}-$approximation algorithm for MKAR. Reusing this result with Lemma 1, we obtain a $\frac{7}{6}-$approximation for $1|a_i = L_i = b_i = \alpha(A_i), G_c = 1\text{-}stage\ bipartite|C_{max}$.

2. We know that MSSDC admits a $\mathcal{PTAS}$ [1], *i.e.*$\rho = 1 - \epsilon$. Using this algorithm to compute such a $\mathcal{PTAS}$ and Lemma 1, we obtain an approximation ratio of $1 + \frac{\epsilon}{3}$ for this problem.

3. The problem $1|a_i = L_i = b_i = \alpha(A_i), G_c = complete\ 1\text{-}stage\ bipartite|C_{max}$, where all the tasks from Y have the same stretch factor $\alpha(y)$:
   (a) Authors from [3] proposed a $\rho = \frac{3}{4}-$approximation algorithm for MSS. Reusing this result with Lemma 1, we obtain a $\frac{13}{12}-$approximation for $1|\alpha_i = a_i = L_i = b_i, complete\ bipartite|C_{max}$.
   (b) They also proved that MSS admits a $\mathcal{PTAS}$ [2] , *i.e.*$\rho = 1 - \epsilon$. Using the algorithm to compute such a $\mathcal{PTAS}$ and Lemma 1, we obtain an approximation ratio of $1 + \frac{\epsilon}{3}$ for $1|a_i = L_i = b_i = \alpha(A_i), G_c = complete\ 1\text{-}stage\ bipartite|C_{max}$ when nodes from Y have the same stretch factor.

$\square$

5.3 2−stage bipartite graph

Theorem 8 *The problem $1|a_i = L_i = b_i = \alpha(A_i), G_c = 2\text{-}stage\ bipartite|C_{max}$ is approximable to a factor $\frac{13}{9}$.*

Proof Reusing the notation introduced for k-stage bipartite graph (see Section 5.3), we consider an instance of $1|a_i = L_i = b_i = \alpha(A_i), G_c = 2\text{-}stage\ bipartite|C_{max}$ where $G_c = (V_0 \cup V_1 \cup V_2, E_1 \cup E_2)$, where each arc in E_i is from a vertex in V_i to a vertex in V_{i+1}, for $i \in 1, 2$.

Definition 1 *We note V_{ip} (p=packed), (resp. V_{ia} (a=alone)) $i = 0, 1$ the set of tasks merged (resp. remaining) in any task from $y \in V_{i+1}$ in a solution S. Moreover, V_{ib} (b=box), $i = 1, 2$ the set of tasks scheduled with some tasks from V_{i-1} merged into it.This notation is extended to a optimal solution S^* by adding a star in the involved variables.*

Considering the specificities of the instance, in any (optimal) solution we make some essential remarks:

1. Tasks from V_0 are source nodes for G_c, and can be either scheduled alone, or merged into some tasks from V_1 only. Given any solution S to the problem on G_c, $\{V_{0p}, V_{0a}\}$ is a partition of V_0.
2. Tasks from V_1 can be either scheduled alone, merged into some tasks from V_2, or scheduled with some tasks from V_0 merged into it. Given any solution S to the problem on G_c, $\{V_{1p}, V_{1a}, V_{1b}\}$ is a partition of V_1.
3. Tasks from V_2 form an independent set for G_c, and have to be scheduled sequentially in any schedule, either alone, or with some tasks from V_1 merged into it. Given any solution S to the problem on G_c, $\{V_{2a}, V_{2b}\}$ is a partition of V_2.

Any solution would consist to schedule first each task with at least one task merged into it, then to schedule the remaining tasks (alone). Given an optimal solution S^*, the length of S^* is given by the following equation:

$$S^* = seq(V_{1b}^*) + seq(V_2 b) + seq(V_{0a}^*) + seq(V_{1a}^*) + seq(V_{2a}^*) \tag{10}$$

or, more simply

$$S^* = seq(V_2) + seq(V_{1_b}^*) + seq(V_{0_a}^*) + seq(V_{1_a}^*) \tag{11}$$

Note that $V_{0_p}^*$ and $V_{1_p}^*$ are not part of the equation, as they are scheduled during the idle time of $V_{1_b}^*$ and $V_{2_b}^*$.

The main idea of the algorithm is divided into three parts:

1. First we compute a part of the solution with a $\frac{7}{6}$-approximation on G_0 thanks to Theorem 7, where $G_0 = G_c[V_0 \cup V_1]$ is the 1-th stage of G_c.
2. Then we compute a second part of the solution with a $\frac{7}{6}$-approximation on G_1 thanks to Theorem 7, where $G_1 = G_c[V_1 \cup V_2]$ is the 2-th stage of G_c.
3. To finish we merge these two parts and resolve potential conflicts between them.

Let consider an instance restricted to the graph G_0. Note that G_0 is a 1-stage bipartite graph. Let $S^*[G_0]$ be an optimal solution on G_0. Let us note $V_{0_p}^*[G_0]$ the set of tasks from V_0 packed into tasks from V_1 in $S^*[G_0]$, and $V_{0_a}^*[G_0]$ the set of remaining tasks.

Obviously, we have:

$$S^*[G_0] = seq(V_1) + seq(V_{0_a}^*[G_0]) \tag{12}$$

Given any solution $S[G_0]$, let us note $V_{1b}[G_0]$ the set of tasks from V_1 with at least one task from V_0 merged into them, and $V_{1a}[G_0]$ the remaining tasks. Let us note $V_{0p}[G_0]$ the set of tasks from V_0 merged into V_1, and $V_{0a}[G_0]$ the set of remaining tasks. We use Theorem 7, Lemma 1, and the demonstration presented in their proof from [4], to compute a solution $S[G_0]$ such that:

$$seq(V_{0p}[G_0]) \geq \frac{1}{2} seq(V_{0_p}^*[G_0]) \tag{13}$$

Note that we have

$$seq(V_{0p}[G_0]) + seq(V_{0a}[G_0]) = seq(V_{0_p}^*[G_0]) + seq(V_{0_a}^*[G_0]) = seq(V_0) \tag{14}$$

As $V_{0_a}^*[G_0]$ represents the set of tasks not packed into V_1 in an optimal $S^*[G_0]$ such that $seq(V_{0_a}^*[G_0])$ is minimal, we know that $seq(V_{0_a}^*[G_0]) \leq seq(V_{0_a}^*)$ .Combining Equation (13) and Equation (14), one obtain:

$$seq(V_{0a}[G_0]) \leq seq(V_{0_a}^*[G_0]) + \frac{1}{2} seq(V_{0_p}^*[G_0]) \leq seq(V_{0_a}^*) + \frac{1}{2} seq(V_{0_p}^*[G_0]) \tag{15}$$

We use an analog reasoning on an instance restricted to the graph G_1. Let $S^*[G_1]$ be an optimal solution on G_1. Let us note $V_{1_p}^*[G_1]$ the set of tasks from V_1 packed into tasks from V_2 in $S^*[G_1]$, and $V_{1_a}^*[G_1]$ the set of remaining tasks. Given any solution $S[G_1]$, let us note $V_{2b}[G_1]$ the set of tasks from V_2 with at least one task from V_1 merged into them, and $V_{1a}[G_1]$ the remaining tasks. One can compute a solution $S[G_1]$ based on a set of tasks $V_{1p}[G_1]$ packed in V_2 such that:

$$seq(V_{1p}[G_1]) \geq \frac{1}{2} seq(V_{1_p}^*[G_1]) \tag{16}$$

and

$$seq(V_{1a}[G_1]) \leq seq(V_{1a}^*[G_1]) + 1/2seq(V_{1p}^*[G_1]) \leq seq(V_{1a}^*) + 1/2seq(V_{1p}^*[G_1]) \quad (17)$$

as we know that $seq(V_{1a}^*[G_1]) \leq seq(V_{1a}^*)$.

We design the feasible solution S for G_c as follows:

- We compute a solution $S[G_1]$ on G_1, then we add to S each task from V_2 and the tasks from V_1 merged into them (i.e. $V_{1p}[G_1]$) in $S[G_1]$.
- Then we compute a solution $S[G_0]$ on G_0, then we add to S each task v from $V_{1b}[G_0]/V_{1p}[G_1]$ and the tasks from V_0 merged into them.
- The tasks $V_{1a}[G_1]/V_{1b}[G_0]$ and $V_{0a}[G_0]$ are added to S and scheduled sequentially.
- We note $V_{conflict}$ the set of remaining tasks, *i.e.* the set of tasks from V_0 which are merged into a task $v \in V_1$ in $S[G_0]$, thus that v is merged into a task from V_2 in $S[G_1]$.

Remark that:

$$seq(V_{1b}[G_0]/V_{1p}[G_1]) + seq(V_{1a}[G_1]/V_{1b}[G_0]) = V_{1a}[G_1]) \quad (18)$$

Thus the cost of our solution S is

$$S = seq(V_2) + seq(V_{1a}[G_1]) + seq(V_{0a}[G_0]) + seq(V_{conflict}) \quad (19)$$

It is also clear that:

$$seq(V_{conflict}) \leq \frac{1}{3}seq(V_{1p}[G_1]) \leq \frac{1}{3}seq(V_{1p}^*[G_1]) \quad (20)$$

Using Equations (15), (17) and (20) in Equation (19), we obtain

$$S \leq seq(V_2) + seq(V_{1a}^*) + \frac{5}{6}seq(V_{1p}^*[G_1]) + seq(V_{0a}^*) + \frac{1}{2}seq(V_{0p}^*[G_0]) \quad (21)$$

Using Equations (11) and (21), we obtain

$$S \leq S^* + \frac{5}{6}seq(V_{1p}^*[G_1]) + \frac{1}{2}seq(V_{0p}^*[G_0]) \quad (22)$$

We know that $S^* \geq seq(V_2)$. We also know that tasks from $(V_{1p}^*[G_1])$ must be merged into tasks from V_2 and cannot exceed the idle time of V_2, implying that $seq(V_{1p}^*[G_1])) \leq \frac{1}{3}seq(V_2)$. One can write the following :

$$\frac{\frac{5}{6}seq(V_{1p}^*[G_1])}{S^*} \leq \frac{\frac{5}{6} \times \frac{1}{3}seq(V_2)}{seq(V_2)} \leq \frac{5}{18} \quad (23)$$

We know that tasks from $(V_{0p}^*[G_0])$ must be merged into tasks from V_1 and cannot exceed the idle time of V_1, implying that $seq(V_{0p}^*[G_0])) \leq \frac{1}{3}seq(V_1)$. We also know that $S^* \geq seq(V_1)$, as V_1 is an independent set of G_c. One can write the following :

$$\frac{\frac{1}{2}seq(V_{0p}^*[G_0])}{S^*} \leq \frac{\frac{1}{2} \cdot \frac{1}{3}seq(V_1)}{seq(V_1)} \leq \frac{1}{6} \quad (24)$$

Finally, with Equations (22), (23) and (24) the proof is finished:

$$\frac{S}{S^*} \leq \frac{13}{9} \tag{25}$$

$\square$

6 Conclusion

The results proposed in this paper are summarized in Table 2. New presented results suggest the main problem of coupled tasks scheduling remains difficult even for restrictive instances, here stretched coupled-tasks when the constraint graph is a bipartite graph. When we consider stretched coupled-tasks, the maximum degree Δ_G seems to play an important role on the problem complexity, as the problem is already $\mathcal{NP}$-Hard to solve when the constraint graph is a star. Approximation results presented in this paper show the problem can be approximated with interesting constant ratio on $k-$stage bipartite graphs for $k = 1$ or 2. The presented approach suggests a generalization is possible for $k \geq 3$. This part constitutes one perspective of this work. Other perspective would consists to study coupled-tasks on other significant topologies, including degree-bounded trees, or regular topologies like the grid.

Acknowledgment

This work has been funded by the regional council of Burgundy.

References

[1] A. Caprara, H. Kellerer, and U. Pferschy. A PTAS for the Multiple Subset Sum Problem with different knapsack capacities. *Inf. Process. Lett*, 2000.

[2] A. Caprara, H. Kellerer, and U. Pferschy. The Multiple Subset Sum Problem. *Siam Journal on Optimization*, 11(2):308–319, 2000.

[3] A. Caprara, H. Kellerer, and U. Pferschy. A 3/4-Approximation Algorithm for Multiple Subset Sum. *J. Heuristics*, 9(2):99–111, 2003.

[4] M. Dawande, J. Kalagnanam, P. Keskinocak, F.S. Salman, and R. Ravi. Approximation Algorithms for the Multiple Knapsack Problem with Assignment Restrictions. *Journal of Combinatorial Optimization*, 4(2):171–186, 2000.

[5] J. Edmonds. Maximum matching and a polyhedron with 0, 1 vertices. *Journal of Research the National Bureau of Standards*, 69 B:125–130, 1965.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[7] R. Giroudeau, J.-C. König, F.K. Moulaï, and J. Palaysi. Complexity and approximation for the precedence constrained scheduling problem with large communication delays. *Theoretical Computer Science*, 401(1–3):107—-119, 2008.

[8] 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.

[9] O.H. Ibarra and C.E. Kim. Fast approximation algorithms for the Knapsack and Sum of Subset problems. *Journal of ACM*, 22(4):463–468, 1975.

[10] H. Kellerer, R. Mansini, U. Pferschy, and M.G. Speranza. An efficient fully polynomial approximation scheme for the subset-sum problem. *Journal of Computer and System Sciences*, 66(2):349–370, 2003.

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

[12] G. Simonin, B. Darties, R. Giroudeau, and J.-C. König. Isomorphic coupled-task scheduling problem with compatibility constraints on a single processor. *Journal of Scheduling*, 14(5):501–509, 2011.

[13] G. Simonin, R. Giroudeau, and J.-C. König. Complexity and approximation for scheduling problem for coupled-tasks in presence of compatibility tasks. In *Project Management and Scheduling*. 2010.

[14] 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*, in press, 2013.

[15] G. Simonin, R. Giroudeau, J.-C. König, and B. Darties. Theoretical Aspects of Scheduling Coupled-Tasks in the Presence of Compatibility Graph. *Algorithmic in Operations Research*, 7(1):1—-12, 2012.