SOME COMPLEXITY AND APPROXIMATION RESULTS FOR COUPLED-TASKS SCHEDULING PROBLEM ACCORDING TO TOPOLOGY

Benoit Darties[1], Rodolphe Giroudeau[2],
Jean-Claude König[2] and Gilles Simonin[3]

Abstract. We consider the makespan minimization coupled-tasks problem in presence of compatibility constraints with a specified topology. 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 several problems in framework of classic complexity and approximation for which the compatibility graph is bipartite (star, chain, ...). In such a context, we design some efficient polynomial-time approximation algorithms for an intractable scheduling problem according to some parameters.

Mathematics Subject Classification. 90B35, 68W25, 68Rxx, 68R10.

1. Introduction and model

The detection of an object by a common radar system is based on the following principle: a transmitter emits a uni-directional pulse that 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 position of the object can be computed by the transmitter. Formally this acquisition process is divided into three parts: (i) pulse transmission; (ii) wave propagation and reflection, (iii) echo reception. Thus the detection system must perform two tasks (parts (i) and (iii)) separated by an idle time (part (ii)). Such systems generally run in non-preemptive mode: once started, a task cannot be interrupted and resumed later. However, the idle time of an acquisition task can be reused to perform another task. On non-preemptive mono-processor systems, scheduling issues appear when in parallel several sensors using different frequencies are working: the idle time of an acquisition task can be reused to perform partially on entirely a second acquisition process using another sensor, but only if both sensors use different frequencies to avoid interferences. Otherwise these two acquisitions processes should be scheduled sequentially.

Coupled-tasks, introduced first by Shapiro [15], seem to be a natural way to model, among others, data acquisition processes: a coupled-task t_i is composed by two sub-tasks with processing time a_i and b_i and whose execution must be separated by an incompressible and not flexible time l_i (called the idle time of the task). For an acquisition process, a sensor emits a radio pulse as a first sub-task, and listens for an echo reply as a second sub-task, while the radio pulse propagation operates during an idle time l_i.

[1] LE2I, UMR CNRS 6306, University of Burgundy, 8 Rue Alain Savary, 21000 Dijon, France. benoit.darties@u-bourgogne.fr
[2] LIRMM, UMR CNRS 5506, Université de Montpellier, 161 rue Ada, 34392 Montpellier cedex 5, France.
[3] Insight Centre for Data Analytics, University College Cork, Cork, Ireland.

Coupled-tasks are also an efficient way to model acquisition systems designed to detect changes in an environment for a given period, by producing two measurements before and after the given period. Here each measurement can be modeled as a sub-task.

We note $\mathcal{T} = \{t_1, \ldots, t_n\}$ the collection of coupled-tasks to be scheduled. In order to minimize the makespan (schedule length) of $\mathcal{T}$, it is necessary to execute one or several different sub-tasks during the idle time of a coupled-task. In the original model, all coupled-tasks may be executed in each other according to processing time of sub-tasks and the duration of the idle time.

Some papers investigated the problem of minimizing the makespan for various configurations depending on the values of a_i, b_i and l_i [1, 2, 14]. In [14], authors present a global visualization of scheduling problems complexity with coupled-tasks, and give main complexity results.

In a multi-sensors acquisition system, incompatibilities may arise between two tasks t_i and t_j if they operate with two different sensors working at the same channel. Thus any valid schedule would require t_i and t_j to be scheduled sequentially. Hereafter, we propose a generalization of an original coupled-tasks model by considering the notion of compatibility constraint among tasks: original coupled-task model, by introducing compatibility constraint among tasks: two tasks t_i and t_j are *compatibles* if any sub-task of t_i may be executed during the idle time of t_j or reciprocally. In [17], we introduced a compatibility graph $G = (V, E)$ to model such this compatibility, where $V = \mathcal{T}$ is the entire collection of coupled-tasks, and each pair of compatible tasks are linked by an edge $e \in E$. We proposed in [17, 18] new results focused on the impact of the addition of G on the complexity of the problem.

Our work is motivated by the acquisition of data for automatic vehicle under water, as a TAIPAN torpedo. With the growth in robotic technologies, several applications and works are emerging and the theoretical needs are a priority. For example, the torpedo is used to execute several submarine topographic surveys, including topological or temperature measurements. These acquisitions tasks can be partitioned into specific sub-problems, where their modelling is very precise.

Since the engineers have a wide degree of freedom to create and transform the different tasks, they required a strong theoretical analysis of coupled tasks with compatibility constraint. Indeed, they needed to have a better knowledge of the difficulty of scheduling coupled-tasks on such systems, and to compare their scheduling heuristics to the optimal one.

1.1. Contribution

In this work, we propose new results of complexity and approximation for particular problem instances composed by *stretched coupled-tasks* only: a stretched coupled-task is a coupled-task $t_i = (a_i, l_i, b_i)$ for which the three parameters a_i, b_i and l_i are equal to the same value $\alpha(t_i)$, called the *stretch factor* of t_i.

We investigate here the problem of scheduling on a mono-processor a set of stretched coupled-tasks, subject to compatibility constraint in order to minimize the completion time of the latest task. For clarity, a_i (resp b_i) refers either to the first (resp. second) sub-task, or to its processing time according to the context.

A major research issue concerns the impact of the class of the compatibility graph G on the complexity of the problem: it is known that the problem is $\mathcal{NP}$-hard even when all the tasks are compatibles between each other, *i.e.* G is a complete graph (see [14]). On the other side, when G is an empty graph a trivial optimal solution would consist in scheduling tasks sequentially. Our aim is to determine the complexity of the problem when G describes some sub-classes of bipartite graphs, and to propose approximation algorithms with performance guarantee for $\mathcal{NP}$-hard instances.

Remark 1.1. If two compatibles stretched coupled-tasks t_i and t_j, with $\alpha(t_i) \leq \alpha(t_j)$, are scheduled in parallel in any solution of the scheduling problem, then one of the following conditions must hold:

(1) either $\alpha(t_i) = \alpha(t_j)$: 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 execution of the two tasks is done without idle time.

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

The others configuration $\alpha(t_i) < \alpha(t_j) < 3\alpha(t_i)$ is unavailable, otherwise some sub-tasks would overlap in the schedule.

From Remark 1.1 one can propose an orientation to each edge $e = (t_i, t_j) \in E$ from the task with the lowest stretch factor to the task with the highest one, or set e as a bidirectional edge when $\alpha(t_i) = \alpha(t_j)$. In the following, we consider only oriented compatibility graphs. Abusing notation, dealing with undirected topologies for G refers in fact to its underlying undirected graph.

We use various standard notations from graph theory: $N_G(x)$ is the set of neighbors of x in G. Δ_G is the maximum degree of G. We denote respectively by $d_G^-(v)$ and $d_G^+(v)$ the indegree and outdegree of v, and $d_G(v) = d_G^-(v) + d_G^+(v)$. We denote by $G[S]$ the graph induced from G by vertices from S.

Reusing the Graham's notation scheme [10], we define the main problem of this study as $1|\alpha(t_i), G|C_{\max}$. We study the variation of the complexity when the topology of G varies, and we propose approximation results for $\mathcal{NP}$-hard instances.

We study thee subclasses of bipartite graphs in particular: the chain, the star, and the k-stage bipartite graphs. A k-stage bipartite graph is a graph $G = (V_0 \cup V_1 \cup \cdots \cup V_k, E_1 \cup E_2 \cup \ldots \cup E_k)$, where each arc in E_i has its extremities in V_i and in V_{i+1}, for $i \in \{1, \ldots, k\}$. For a given k-stage bipartite graph G, we denote by $G_k = G[V_{k-1} \cup V_k]$ the kth stage of G. In this paper, we focus our study on 1-stage bipartite graphs (**1-SBG**) and 2-stage bipartite graphs (**2-SBG**). We also study the problem when the compatibility graph G is a 1-stage complete bipartite graph (**1-SCBG**), *i.e.* E_1 contains all the edges (x, y), $\forall x \in V_0$, $\forall y \in V_1$.

For 1-SBG (or 2-SBG) with $G = (X \cup Y, E)$, we denoted by X-tasks (resp. Y-tasks) the set of tasks represented by X (resp. Y) in G. For any set of X-tasks, let $seq(X)$ be the time required to schedule sequentially all the tasks from X. Formally, we have:

$$seq(X) = \sum_{t \in X} 3\alpha(t).$$

Remark 1.2. Given an instance of $1|\alpha(t_i), G|C_{\max}$. If X is an independent set for G, then all the tasks from X are pairwise non-compatibles. Thus $seq(X)$ is a lower bound for the cost of any optimal solution.

The results obtained in this article are summarized in Table 1.

TABLE 1. Complexity and approximation results discussed in this paper.

Topology	Complexity	Approximation
$G = $ Chain graph	$O(n^3)$ (Thm. 2.1)	
$G = $ Star graph [1]	$\mathcal{NP} - \mathcal{C}$ (Thm. 2.3)	$\mathcal{FPTAS}$ (Thm. 4.1)
$G = $ Star graph [2]	$O(n)$ (Thm. 2.2)	
$G = $ 1-SBG, $d_G(Y) \leq 2$	$O(n^3)$ (Thm. 3.1)	
$G = $ 1-SCBG	$\mathcal{NP} - \mathcal{C}$ (see [16])	$\mathcal{PTAS}$ (Thm. 4.3) $\frac{13}{12}$-$\mathcal{APX}$ (Thm. 4.3)
$G = $ 2-SBG	$\mathcal{NP} - \mathcal{C}$ (Thm. 3.2)	$\frac{13}{9}$-$\mathcal{APX}$ (Thm. 4.4)

[1] Star graph with only incoming arcs for the central node arc.

[2] Star graph with at least one outcoming arc for the central node.

1.2. Prerequisites

1.2.1. Performance ratio

Recall that the performance ratio ρ for a minimization (resp. maximization) problem is given as the ratio between the value of the approximation solution returned by the algorithm $\mathcal{A}$ on an instance $\mathcal{I}$ and the optimum *i.e.* $\rho \leq \max_{\mathcal{I}} \frac{\mathcal{A}(\mathcal{I})}{\mathcal{OPT}(\mathcal{I})}$ (resp. $\rho \geq \min_{\mathcal{I}} \frac{\mathcal{OPT}(\mathcal{I})}{\mathcal{A}(\mathcal{I})}$). Notice that for a minimization problem the ratio is greater than one (resp. lower than one).

1.2.2. Definition of problems

To prove the different results announced in this paper, we use several well-known approximation results on four packing-related problems:

(1) The SUBSET SUM (SS) problem is a well-known problem in which, given a set $\mathcal{S}$ of n positive values and $v \in \mathbb{N}$, one asks if 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 [8]). The optimization version problem is sometimes viewed as a KNAPSACK problem, where each item profits and weights coincide to a value in $\mathcal{S}$, the knapsack capacity is v, and the aim is 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 *et al.* [4] 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 [5].

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

(4) As a generalization of MSSDC, MULTIPLE KNAPSACK ASSIGNMENT RESTRICTION (MKAR) problem consists to pack weighted items into non-identical capacity-constrained bins, with the additional constraint 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 that the profit of each item equals its weight, [6] proposed a $\frac{1}{2}$-approximation.

We also use a well-known result concerning a variant of the $\mathcal{NP}$-complete problem 3SAT [8], denoted subsequently by ONE-IN-$(2,3)$SAT$(2,\bar{1})$: An instance of ONE-IN-$(2,3)$SAT$(2,\bar{1})$ is described by the following elements: we use $\mathcal{V}$ to denote the set of n variables. Let n be a multiple of 3 and let $\mathcal{C}$ be a set of clauses of cardinality 2 or 3. There are 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 n 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.

The aim of ONE-IN-$(2,3)$SAT$(2,\bar{1})$ is to find if there exists a truth assignment $I : \mathcal{V} \to \{0,1\}$, 0 for false and 1 for true, whereby each clause in $\mathcal{C}$ has exactly one true literal. ONE-IN-$(2,3)$SAT$(2,\bar{1})$ has been proven $\mathcal{NP}$-complete in [9].

As an example, the following logic formula is the smallest 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 $x_0 = 1$, $x_3 = 1$ and $x_i = 0$ 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.

2. Computational complexity for some classes
of compatibility graphs

In this section, we present two preliminary results of complexity for the problem that consists in scheduling a set of stretched-coupled tasks with compatibility constraints. In such a context, we will consider the topologies of chain and star.

First we show that the problem is solvable within a $O(n^3)$ time complexity algorithm when G is a chain (Thm. 2.1). Then we prove that it is $\mathcal{NP}$-hard even when the compatibility graph is a star (Thm. 2.3),

2.1. Chain graph

Despite of the simplicity of a chain topology, solving the scheduling problem on a chain is not as simple as it appears: a main issue arise when two adjacent vertices x and y have the same stretch factor. In this configuration, we cannot determine locally if x and y can be packed together in an optimal solution or not (this requires to examine the neighbourhood of x and y, and this problematic configuration can be repeated all along the chain). However, we show that the scheduling problem with a chain is polynomial using a similar method as developed in [18].

Theorem 2.1. *The problem* $1|\alpha(t_i), G = chain|C_{\max}$ *admits a polynomial-time algorithm.*

Proof. This problem can be solved in polynomial-time by a reduction to the search for a minimum weighted perfect matching. This problem can be polynomially solved in $O(n^3)$ time complexity [7].

First, note that if for a task x with two neighbors y and z, we have $3(\alpha(y) + \alpha(z)) \leq \alpha(x)$, the idle duration of x is high enough to schedule both y and z. Thus one can schedule y and z into x without decreasing the cost of any optimal solution, and remove tasks x, y and z from the studied graph. Thus, in the rest of the proof, one can restrict our study to chains $G = (V, E)$ such that for any $x \in V$, we have $3 \sum_{y \in N_G(x)} \alpha(y) > \alpha(x)$.

In order to obtain a graph with an even number of vertices and to find a perfect matching, we construct a graph $H = (V_H, E_H, w)$ and we define a weighted function $w : E \to \mathbb{N}$ as follows:

(1) Let $\mathcal{I}_1$ be an instance of our problem with a compatibility graph $G = (V, E)$, 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, consisting of two copies of G denoted by $G' = (V', E')$ and $G'' = (V'', E'')$. The vertex corresponding to $x \in V$ is denoted by x' in G' and x'' in G''. Moreover, $\forall i = 1, \dots, n$, an edge $\{x', x''\}$ in E_H is added and we state $w(\{x', x''\}) = \mathbf{3 \times \alpha(x)'}$. This weight represents the sequential time of the x'-task. We have $H = G' \cup G'' = (V' \cup V'', E' \cup E'')$, with $|V' \cup V''|$ of even size.
(2) For two compatible tasks x' and y' with $3 \times \alpha_{x'} \leq \alpha_{y'}$ or $3 \times \alpha_{y'} \leq \alpha_{x'}$, we add the edges $\{x', y'\}$ and $\{x'', y''\}$ in E_H and we state $w(\{x', y'\}) = w(\{x'', y''\}) = \frac{3 \times \max\{\alpha_{x'}, \alpha_{y'}\}}{2}$.
(3) For two compatible tasks x' and y' with $\alpha_{x'} = \alpha_{y'}$, we add the edges $\{x', y'\}$ and $\{x'', y''\}$ in E_H, and we state $w(\{x', y'\}) = w(\{x'', y''\}) = \frac{4 \times \alpha_{x'}}{2}$.

Figure 1 shows an example of construction of H when G is a chain with 3 vertices.

One can show that there is a (weighted) perfect matching on H, which cover all the vertices of H. In fact the construction implies that for any perfect matching W of cost C on H, one can provide a valid schedule of processing time C for the scheduling problem: an edge $e \in W$ with $e = x', x'', x' \in G' \wedge x'' \in G''$ implies that task x is scheduled alone, while an edge $e \in W$ with $e = x', y', x', y' \in G'$ implies that tasks x and y are packed together in the resulting schedule – and the edge $e = x'', y'', x'', y'' \in G''$ belong also to the matching -.

For a minimum weight perfect matching C, we can associate a schedule of minimum processing time equal to C and vice versa. The detailed proof of the relationship between a solution to our problem with G and a solution of a minimum weight perfect matching in H is presented in [18].

In the review of the literature, the Edmonds algorithm determines a minimum weight perfect matching in $O(n^3)$ [7]. So the optimization problem with G is polynomial, and if one adds the execution of the blocks created by removed vertices, this leads to the polynomiality of the problem $1|\alpha(t_i), G = chain|C_{\max}$.   □
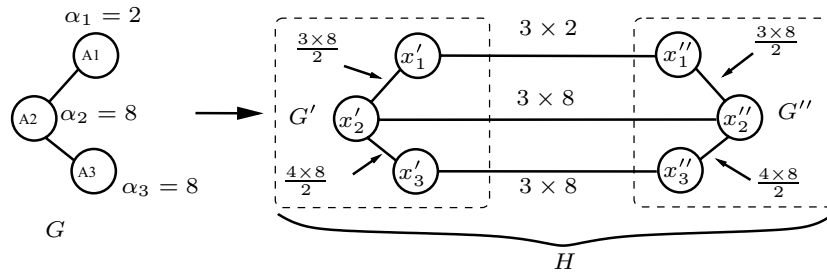
FIGURE 1. Example of the transformation.

2.2. Star graph

We focus on the case with a star graph, *i.e.* a graph with a central node β. In such a context, we show that the complexity depends on the number of outgoing arcs from β. The following results also imply that the studied problem can be $\mathcal{NP}$-hard even on acyclic low-diameter graphs, when the degree of G is unbounded.

Theorem 2.2. *The problem* $1|\alpha(t_i), G = \mathrm{star}|C_{\max}$ *is polynomial if the central node admits at least one out-coming arc.*

Proof. Let S be the set of satellite nodes. According to the Remark 1.2, $seq(S)$ is a lower bound for the cost of an optimal solution. This bound is achieved if we can execute the central node in a satellite node. $\square$

Theorem 2.3. *The problem* $1|\alpha(t_i), G = \mathrm{star}|C_{\max}$ *is* $\mathcal{NP}$-*hard if the central node admits only incoming arcs.*

Proof. We propose a reduction from the SUBSET SUM (SS) problem (see Sect. 1.2). From an instance of SS composed by a set $\mathcal{S}$ of n positive values and $v \in \mathbb{N}$ (with $v \geq x, \forall x \in \mathcal{S}$), we construct an instance of $1|\alpha(t_i), G = \mathrm{star}|C_{\max} = \sum_{t \in V} \alpha(t) + 2\alpha(\beta)$ in the following way:

(1) For each value $i \in \mathcal{S}$ we introduce a coupled-task t with $\alpha(t) = i$. Let V be the set of these tasks.
(2) We add a task β with $\alpha(\beta) = a_\beta = l_\beta = b_\beta = 3 \times v$.
(3) We define a compatibility constraint between each task $t \in V$ and β.

Clearly the compatibility graph G is a star with β as the central node, and the transformation is computed in polynomial time.

We will prove that there exists a positive solution for the SUBSET SUM (SS) problem iff there exists a feasible solution for the scheduling problem with a length $\sum_{t \in V} \alpha(t) + 2\alpha(\beta)$.

It is easy to see that $1|\alpha(t_i), G = \mathrm{star}|C_{\max} = \sum_{t \in V} \alpha(t) + 2\alpha(\beta) \in \mathcal{NP}$.

Let W be the set of the nodes executed in the central node for a scheduling. The cost of this scheduling in clearly $seq(\mathcal{T}) - seq(W)$. Therefore, the problem of finding a scheduling of cost $seq(\mathcal{T}) - \alpha(\beta)$ is clearly equivalent to an instance of the subset sum with $v = \alpha(\beta)$ and S the set of the processing time of the satellite tasks.

This concludes the proof of Theorem 2.3. $\square$

3. ON THE BOUNDARY BETWEEN POLYNOMIAL-TIME ALGORITHM AND $\mathcal{NP}$-COMPLETENESS ON 1-STAGE BIPARTITE GRAPHS

Preliminary results of Section 2 show that the problem is $\mathcal{NP}$-hard on acyclic low-diameter instances when the degree is unbounded. They suggest that the complexity of the problem may be linked to the maximum degree of the graph.

This section is devoted to the $\mathcal{NP}$-completeness of several scheduling problems in presence of a 1-stage bipartite compatibility graph, according to the maximum degree of vertices and some structural parameters like the number of different values of coupled-tasks.

We will sharp the line of demarcation between the polynomially solvable cases and the $\mathcal{NP}$-hardness ones according to several topologies. We focus our analysis when G is a 1-stage bipartite graph. We prove that the problem is solvable within a $O(n^3)$ polynomial algorithm if $\Delta_G = 2$ (Thm. 3.1), but becomes $\mathcal{NP}$-hard when $\Delta_G = 3$ (Thm. 3.2).

We start by designing a polynomial-time algorithm for the scheduling problem in which the maximum degree of incoming arcs on Y-tasks is at most two.

Theorem 3.1. *The problem of deciding whether an instance of $1|\alpha(t_i)$, $G = 1 - stage bipartite, d_G(Y) \leq 2|C_{\max}$ is polynomial. In fact, the previous result may be extended to a graph G (not necessarily bipartite) such that $\forall x, d^-(X) \leq 2$ with $3(\alpha(x_1) + \alpha(x_2)) > \alpha(x)$, where x_1 and x_2 are the 2 neighbors of x.*

Proof. Let $G = (X \cup Y, E)$ be a 1-stage bipartite compatibility graph (arcs oriented from X to Y only, implying that only X-tasks can be executed in the idle time of and Y-task). Y-tasks will always be scheduled sequentially as Y is an independent set of G (*cf.* Rem. 1.2). The aim is to fill their idle time with a maximum of X-tasks, while the remained tasks will be executed after the Y-tasks. We just have to minimize the length of the remained tasks. It is easy to see that all Y-tasks with incoming degree equal to one are executed sequentially with their only X-task in their idle time. The following algorithm is focused on the case $\Delta_G = 2$. It is defined in two steps:

(1) For each task $y \in Y$ such that $3 \times \alpha(x_1) + 3 \times \alpha(x_2) \leq \alpha(y)$ where x_1 and x_2 are the only two neighbors of Y, we add y to the schedule and execute x_1 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 $x \in N_G(y)$, then we perform a maximum weight matching on G 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.
(3) Then, remaining tasks are processed sequentially after the other tasks.

The complexity of this algorithm is $O(n^3)$ using the Hungarian method [13]. For the extension, it is sufficient to use a maximum weight perfect matching [7]. $\qquad\square$

Theorem 3.2. *The problem of deciding whether an instance of $1|\alpha(t_i), G = 1 - stage bipartite, d_G(X) = 2, d_G(Y) \in \{2, 3\}|C_{\max}$ has a schedule of length at most $54n$ is $\mathcal{NP}$-complete where n is the number of tasks.*

Proof. It is easy to see that our problem is 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 $1|\alpha(t_i), G = 1-stage bipartite, d_G(X) = 2, d_G(Y) \in \{2, 3\}|C_{\max} = 54n$ in following way:

(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 constraint:
    (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) Fgr 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 $(\bar{x}, \bar{C}^x)$, $(\bar{x}', \bar{C}^x)$.

This transformation can be clearly computed in polynomial time and an illustration is depicted in Figure 2. The proposed compatibility graph is a 1-stage bipartite and $d_G(x) \leq 3, \forall x \in \mathcal{VT} \cup \mathcal{LT} \cup \mathcal{CT}$.

In follows, we say that a task x is merged to a task y, if there exists a compatibility constraint from x to y; *i.e.* the coupled-task x may be executed during the idle time 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 $I : \mathcal{V} \to \{0, 1\}$ such that each clause in $\mathcal{C}$ has exactly one true literal (*i.e.* one literal equal to 1). We make several essential 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, then 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 way, the variable-tasks $\bar{x}$ et $\bar{x}'$ are merged simultaneously with $\bar{C}^x$ if they have to be into it.

  (6) Hence, first x and x' are merged with C^x or with a 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 $I : \mathcal{V} \to \{0, 1\}$, 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.

  The merged-tasks are given in Figure 2. 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 $54n$. □

Theorem 3.3. *The problem of deciding whether an instance of* $1|\alpha(t_i), G = 1 - stage\ bipartite, d_G(X) \in \{1, 2\}, d_G(Y) \in \{3, 4\}|C_{\max}$ *has a schedule of length at most* $54n$ *is* $\mathcal{NP}$-*complete, where* n *is the number of tasks.*

Proof. We use a similar proof as given for the Theorem 3.2. It is sufficient to add for each clause C with a length of two (resp. C' of length three) a dummy coupled-task $\mathcal{D}_C$ (resp. $\mathcal{D}'_C$) with $\mathcal{D}_C = (1, 1, 1) = \mathcal{D}'_C$, and the value of the clause-task C (resp. C') is now $C = C' = (6, 6, 6)$. In other words, we add these two compatibility
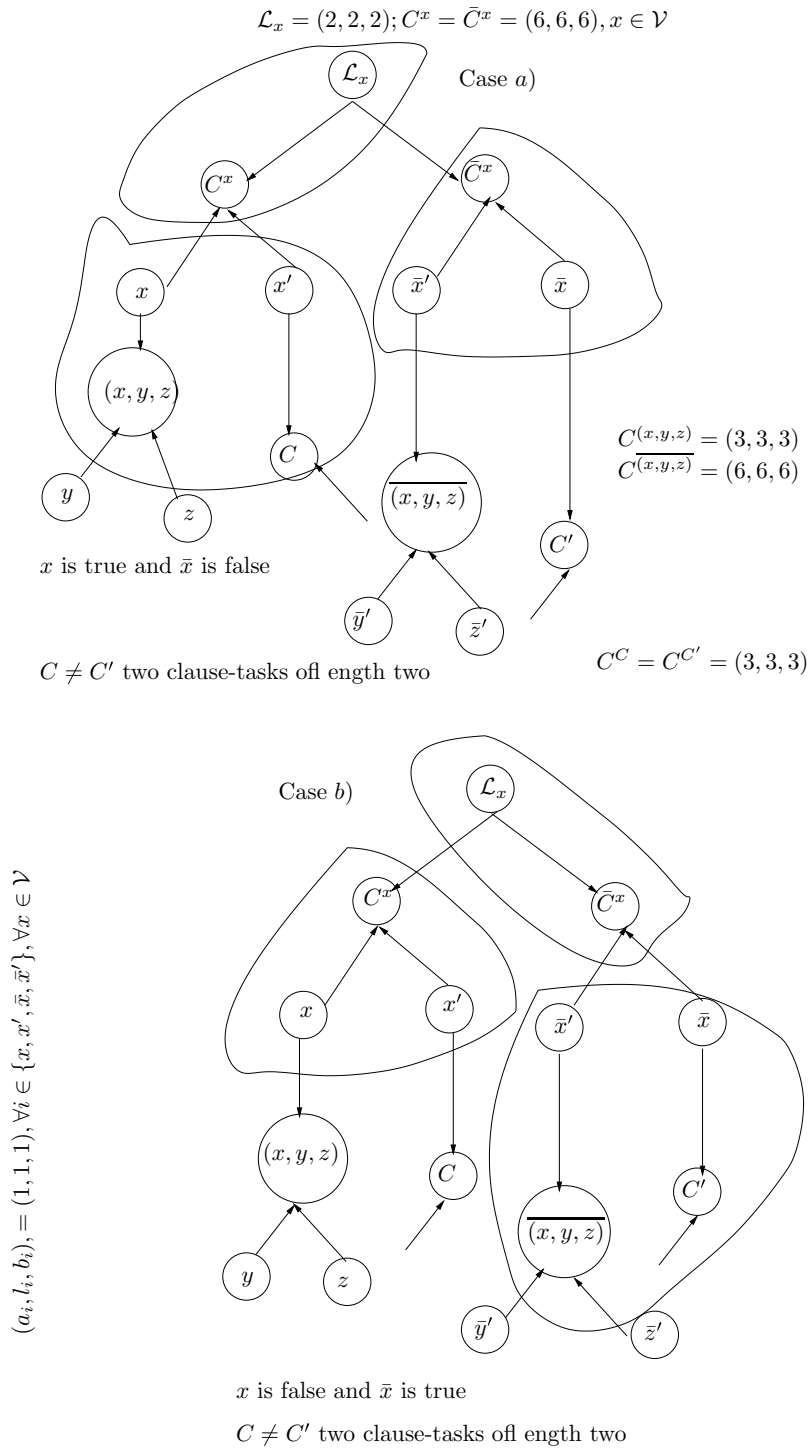
$\mathcal{L}_x = (2,2,2); C^x = \bar{C}^x = (6,6,6), x \in \mathcal{V}$

Case $a)$

$\mathcal{L}_x$

C^x $\widehat{\bar{C}^x}$

x x' $\bar{x}'$ $\bar{x}$

(x,y,z)

C

$\overline{(x,y,z)}$

$C^{(x,y,z)} = (3,3,3)$
$\overline{C^{(x,y,z)}} = (6,6,6)$

y

z

C'

x is true and $\bar{x}$ is false

$\bar{y}'$ $\bar{z}'$

$C \neq C'$ two clause-tasks ofl ength two

$C^C = C^{C'} = (3,3,3)$

$(a_i, l_i, b_i) = (1,1,1), \forall i \in \{x, x', \bar{x}, \bar{x}'\}, \forall x \in \mathcal{V}$

Case $b)$

$\mathcal{L}_x$

C^x $\widehat{\bar{C}^x}$

x x' $\bar{x}'$ $\bar{x}$

(x,y,z) C C'

$\overline{(x,y,z)}$

y z

$\bar{y}'$ $\bar{z}'$

x is false and $\bar{x}$ is true

$C \neq C'$ two clause-tasks ofl ength two

FIGURE 2. A partial compatibility graph for the $\mathcal{NP}$-completeness of the scheduling problem $1|\alpha(t_i), G = 1-stage\ bipartite, d_G(X) = 2, d_G(Y) \in \{2,3\}|C_{\max} = 54n$. A truth assignment and partial schedule.

constraints:

- $\mathcal{D}_C \to C$, for each clause C of length two,
- $\mathcal{D}'_C \to C'$, for each clause C' of length three.

There is a schedule with length of $54n$ at most iff there exists a truth assignment $I : \mathcal{V} \to \{0, 1\}$ such that each clause in $\mathcal{C}$ has exactly one true literal (*i.e.* one literal equal to 1). $\qquad \square$

Corollary 3.4. *The problem of deciding whether an instance of* $1|\alpha(t_i) \neq \alpha(t_j), \forall i \neq j, \Delta_G = 3, G = 1 -$ *stage bipartite*$|C_{\max}$ *has a schedule of length at most $54n$ is $\mathcal{NP}$-complete, where n is the number of tasks.*

Proof. The proof of Theorem 3.2 can be adapted by using the classical scaling arguments assigning $\alpha(x) + \epsilon$ to each task. $\qquad \square$

4. Polynomial-time approximation algorithms

   This section is devoted to design some efficient polynomial-time approximation algorithms for several topologies and mainly for bipartite graphs. In [16], authors proposed a simple algorithm, which consists in scheduling all the tasks consecutively, with a performance ratio bounded by $3/2$ for a general compatibility graph. The challenge for the remaining section, is to propose some efficient algorithms with a ratio strictly lower than $3/2$. We propose a $\mathcal{FPTAS}$ for the star graph whereas some $\mathcal{APX}$-algorithms are developed in the remaining section according to the characteristics of the 1-stage bipartite graph. At last, we extend the result is extended to the 2-stage bipartite graph.

4.1. Star graph

Theorem 4.1. *The problem $1|\alpha(t_i), G = \text{star}|C_{\max}$ 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 Cor. 2.2). Therefore, we may use the solution given by the SUBSET SUM (SS) (see [11] and [12]). Indeed, based on the reduction used in the proof of Theorem 2.3 and the optimization version of SS: the aim is to find W^* (an optimal set of tasks executed during the idle time of the central node) which maximizes $seq(W^*)$ such that $seq(W^*) \leq \alpha(\beta)$.

   Let us suppose that $\frac{seq(W)}{seq(W^*)} \geq 1 - \epsilon$, where W designates the value of the approximation solution for SUBSET SUM.

   Note that $\alpha(\beta) \geq seq(W^*)$ and $seq(\mathcal{T}) \geq 3\alpha(\beta)$ lead to $seq(\mathcal{T}) \geq 3seq(W^*)$.

$$\begin{aligned} \frac{seq(\mathcal{T}) - seq(W)}{seq(\mathcal{T}) - seq(W^*)} &= 1 + \frac{seq(W^*) - seq(W)}{seq(\mathcal{T}) - seq(W^*)} \\ &\leq 1 + \frac{seq(W^*) - seq(W)}{2seq(W^*)} \\ &\leq 1 + \frac{1 - \frac{seq(W)}{seq(W^*)}}{2} \leq 1 + \frac{1 - (1 - \epsilon)}{2} = 1 + \epsilon/2. \end{aligned}$$

Therefore the existence of a $\mathcal{FPTAS}$ for the SUBSET SUM involves a $\mathcal{FPTAS}$ for our scheduling problem. $\quad \square$

4.2. 1-stage bipartite graph

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

Lemma 4.2. *Let Π be a problem with $\Pi \in \{\text{MKAR}, \text{MSSDC}, \text{MSS}\}$ such that Π admits a ρ-approximation, then the following problems*

(1) $1|\alpha(t_i), G = 1 - stage\ bipartite|C_{\max}$,
(2) $1|\alpha(t_i), G = complete\ 1 - stage\ bipartite|C_{\max}$,
(3) $1|\alpha(t_i), G = complete\ 1 - stage\ bipartite|C_{\max}$, *where $G = (X \cup Y, E)$ and all the tasks from Y have the same stretch factor $\alpha(Y)$,*

posses a ρ'-approximables within a factor $\rho' = 1 + \frac{(1-\rho)}{3}$ using an approximability reduction from MKAR, MSSDC *and* MSS *respectively.*

Proof.

(1) Consider now an instance of $1|\alpha(t_i), G = 1 - stage\ bipartite|C_{\max}$ with a graph $G = (X \cup Y, E)$ (for any arc $e = (x, y) \in E$, we have $x \in X$ and $y \in Y$) and a stretch factor function $\alpha : X \cup Y \to \mathbb{N}$.

In such an instance, any valid schedule consists in finding 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 X-tasks packed into a task from Y. Let $X_{\bar{p}}$ the set of remaining tasks, with $X_{\bar{p}} = X \setminus X_p$. Obviously, we have:

$$seq(X_p) + seq(X_{\bar{p}}) = seq(X) \tag{4.1}$$

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

$$C_{\max}(S) = seq(Y) + seq(X_{\bar{p}}) \tag{4.2}$$

From equations (4.1) and (4.2) we have:

$$C_{\max}(S) = seq(Y) + seq(X) - seq(X_p). \tag{4.3}$$

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

Using algorithms and results from the literature, one can obtain an assignment of some items into bins. We denote by X_p the set of these packed items. The cost of the solution for the MKAR problem is $seq(X_p)$. If MKAR is approximable to a factor ρ, then we have:

$$seq(X_p) \geq \rho \times seq(X_p^*), \tag{4.4}$$

where X_p^* is the set of packable items with the maximum profit. Combining equations (4.3) and (4.4), we obtain a solution for $1|\alpha(t_i), G = 1 - stage\ bipartite|C_{\max}$ with a length:

$$C_{\max}(S) \leq seq(Y) + seq(X) - \rho \times seq(X_p^*). \tag{4.5}$$

As X and Y are two fixed sets, an 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 equal to:

$$C_{\max}(S^*) = seq(Y) + seq(X) - seq(X_p^*). \tag{4.6}$$

Using equations (4.5) and (4.6), the ratio obtained between our solution S and the optimal one S^* is:

$$\frac{C_{\max}(S)}{C_{\max}(S^*)} \leq \frac{seq(Y) + seq(X) - \rho \times seq(X_p^*)}{seq(Y) + seq(X) - seq(X_p^*)} \leq 1 + \frac{(1-\rho) \times seq(X_p^*)}{seq(Y) + seq(X) - seq(X_p^*)}. \tag{4.7}$$

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:

$$seq(X_p^*) \leq \frac{1}{3} seq(Y). \tag{4.8}$$

Combined to equation (4.7), we obtain the desired upper bound:

$$\rho' = \frac{C_{\max}(S)}{C_{\max}(S^*)} \leq 1 + \frac{(1-\rho)}{3}. \tag{4.9}$$

(2) For the problem $1|\alpha(t_i), G = complete\ 1\text{-}stage\ bipartite|C_{\max}$, the proof is similar to the previous one. We remind that MSSDC is a special case of MKAR in which each item can be packed in any bin.

(3) For the problem $1|\alpha(t_i), G = complete\ 1 - stage\ bipartite|C_{\max}$ where $G = (X \cup Y, E)$ and all the Y-tasks have the same stretch factor $\alpha(Y)$, the proof is similar to the previous one since MSSDC is a generalization of MSS. $\qquad\square$

Theorem 4.3. *The following problems admit a polynomial-time approximation algorithm:*

(1) *The problem $1|\alpha(t_i), G = 1 - stage\ bipartite|C_{\max}$ is approximable to a factor $\frac{7}{6}$.*
(2) *The problem $1|\alpha(t_i), G = complete\ 1 - stage\ bipartite|C_{\max}$ admits a $\mathcal{PTAS}$.*
(3) *The problem $1|\alpha(t_i), G = complete\ 1 - stage\ bipartite|C_{\max}$ where $G = (X \cup Y, E)$ and all the Y-tasks 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 [6] proposed a $\rho = \frac{1}{2}$−approximation algorithm for MKAR. Reusing this result with Lemma 4.2, we obtain a $\frac{7}{6}$−approximation for $1|\alpha(t_i), G = 1 - stage\ bipartite|C_{\max}$.
(2) We know that MSSDC admits a $\mathcal{PTAS}$ [3], *i.e.* $\rho = 1 - \epsilon$. Using this algorithm to compute such a $\mathcal{PTAS}$, with Lemma 4.2 we obtain an approximation ratio of $1 + \frac{\epsilon}{3}$ for this problem.
(3) In this case we have two different results:
    (a) Authors from [5] proposed a $\rho = \frac{3}{4}$−approximation algorithm for MSS. Reusing this result with Lemma 4.2, we obtain a $\frac{13}{12}$−approximation for $1|\alpha(t_i), G = complete\ bipartite|C_{\max}$.
    (b) They also proved that MSS admits a $\mathcal{PTAS}$ [4], *i.e.* $\rho = 1 - \epsilon$. Using the algorithm to compute such a $\mathcal{PTAS}$, with Lemma 4.2 we obtain an approximation ratio of $1 + \frac{\epsilon}{3}$ for $1|\alpha(t_i), G = complete\ 1 - stage\ bipartite|C_{\max}$ when Y-tasks have the same stretch factor. $\qquad\square$

4.3. 2−stage bipartite graph

In the following, we extend the previous result for 2-stage bipartite graphs.

Theorem 4.4. *The problem $1|\alpha(t_i), G = 2 - stage\ bipartite|C_{\max}$ is approximable to a factor $\frac{13}{9}$.*

Proof. The main idea of the algorithm is divided into three steps:

(1) First we compute a part of the solution with a $\frac{7}{6}$-approximation on G_0 thanks to Theorem 4.3, where $G_0 = G[V_0 \cup V_1]$ is the 1st stage of G.

(2) Then we compute a second part of the solution with a $\frac{7}{6}$-approximation on G_1 thanks to Theorem 4.3, where $G_1 = G[V_1 \cup V_2]$ is the 2nd stage of G.

(3) To finish we merge these two parts and we resolve potential conflicts between them, *i.e.* by giving a preference to tasks packed in G_1. Computing the cost of this solution gives us an approximation ratio of $\frac{13}{9}$.

Reusing the notation introduced for k-stage bipartite graphs (see Sect. 1.1), we consider an instance of $1|\alpha(t_i), G = 2 - stage\ bipartite|C_{\max}$ with $G = (V_0 \cup V_1 \cup V_2, E_1 \cup E_2)$, where each arc in E_i has its extremities in V_{i-1} and V_i, for $i \in \{1, 2\}$.

$\forall i = \{0, 1\}$ we denote[1] by V_{ip} (resp. V_{ia}) the set of tasks merged (resp. remaining) in any task from $y \in V_{i+1}$ in a solution S. Moreover, $\forall i = \{1, 2\}$ let V_{ib} be the set of tasks scheduled with some tasks from V_{i-1} merged into it. This notation is extended to an optimal solution S^* by adding a star in the involved variables.

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

(1) Tasks from V_0 are source nodes in G, and they can be either scheduled alone, or merged only into some tasks from V_1 only. Given any solution S to the problem on G, $\{V_{0p}, V_{0a}\}$ 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, $\{V_{1p}, V_{1a}, V_{1b}\}$ is a partition of V_1.

(3) Tasks from V_2 form an independent set in G, and have to be scheduled sequentially in any schedule (*cf.* Rem. 1.2), either alone or with some tasks from V_1 merged into it. Given any solution S to the problem on G, $\{V_{2a}, V_{2b}\}$ is a partition of V_2.

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

$$S^* = seq(V_{1b}^*) + seq(V_{2b}) + seq(V_{0a}^*) + seq(V_{1a}^*) + seq(V_{2a}^*) \tag{4.10}$$

or, more simply

$$S^* = seq(V_2) + seq(V_{1b}^*) + seq(V_{0a}^*) + seq(V_{1a}^*). \tag{4.11}$$

Note that V_{0p}^* and V_{1p}^* are not part of the equation, as they are scheduled during the idle time of V_{1b}^* and V_{2b}^*. The main idea of the algorithm is divided into three steps:

(1) First we compute a part of the solution with a $\frac{7}{6}$-approximation on G_0 thanks to Theorem 4.3, where $G_0 = G[V_0 \cup V_1]$ is the 1st stage of G.

(2) Then we compute a second part of the solution with a $\frac{7}{6}$-approximation on G_1 thanks to Theorem 4.3, where $G_1 = G[V_1 \cup V_2]$ is the 2nd stage of G.

(3) To finish we merge these two parts and we solve 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 denote by $V_{0p}^*[G_0]$ the set of tasks from V_0 packed into tasks from V_1 in $S^*[G_0]$, and by $V_{0a}^*[G_0]$ the set of remaining tasks.

Obviously, we have:

$$S^*[G_0] = seq(V_1) + V_{0a}^*[G_0]. \tag{4.12}$$

Given any solution $S[G_0]$, let $V_{1b}[G_0]$ be the set of tasks from V_1 with at least one task from V_0 merged into them, and $V_{1a}[G_0]$ be the remaining tasks. Let us denote by $V_{0p}[G_0]$ the set of tasks from V_0 merged into V_1, and by $V_{0a}[G_0]$ the set of remaining tasks. Using Theorem 4.3, Lemma 4.2, and the demonstration presented in the proof from [6], we compute a solution $S[G_0]$ such that:

$$seq(V_{0p}[G_0]) \geq \frac{1}{2} seq(V_{0p}^*[G_0]). \tag{4.13}$$

---

[1] Notations: p for packed, a for alone, and b for box

Note that we have:

$$seq(V_{0p}[G_0]) + seq(V_{0a}[G_0]) = seq(V_{0p}^*[G_0]) + seq(V_{0a}^*[G_0]) = seq(V_0) \tag{4.14}$$

Combining equations (4.13) and (4.14), we obtain:

$$seq(V_{0a}[G_0]) \le seq(V_{0a}^*[G_0]) + \frac{1}{2}seq(V_{0p}^*[G_0]) \le seq(V_{0a}^*) + \frac{1}{2}seq(V_{0p}^*[G_0]) \tag{4.15}$$

as we know by definition that $seq(V_{0a}^*[G_1]) \le seq(V_{0a}^*)$.

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

$$seq(V_{1p}[G_1]) \ge \frac{1}{2}seq(V_{1p}^*[G_1]) \tag{4.16}$$

and

$$seq(V_{1a}[G_1]) \le seq(V_{1a}^*[G_1]) + 1/2seq(V_{1p}^*[G_1]) \le seq(V_{1a}^*) + 1/2seq(V_{1p}^*[G_1]) \tag{4.17}$$

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

We design the feasible solution S for G as follows:

(1) First 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 $S[G_1]$.
(2) Second we compute a solution $S[G_0]$ on G_0, then we add to S each task v from $V_{1b}[G_0] \setminus V_{1p}[G_1]$ and the tasks from V_0 merged into them.
(3) Third the tasks $V_{1a}[G_1] \setminus V_{1b}[G_0]$ and $V_{0a}[G_0]$ are added to S and scheduled sequentially.
(4) At last we denote by $V_{conflict}$ 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 $S[G_1]$.

Observe that:

$$seq(V_{1b}[G_0] \setminus V_{1p}[G_1]) + seq(V_{1a}[G_1] \setminus V_{1b}[G_0]) = V_{1a}[G_1]). \tag{4.18}$$

Thus the cost of our solution S is:

$$S = seq(V_2) + seq(V_{1a}[G_1]) + seq(V_{0a}[G_0]) + seq(V_{conflict}). \tag{4.19}$$

It is also clear that:

$$seq(V_{conflict}) \le \frac{1}{3}seq(V_{1p}[G_1]) \le \frac{1}{3}seq(V_{1p}^*[G_1]). \tag{4.20}$$

Using equations (4.15), (4.17) and (4.20) in equation (4.19), we obtain:

$$S \le seq(V_2) + seq(V_{1a}^*) + \frac{5}{6}seq(V_{1p}^*[G_1]) + seq(V_{0a}^*) + \frac{1}{2}seq(V_{0p}^*[G_0]). \tag{4.21}$$

Using equations (4.11) and (4.21), we obtain:

$$S \le S^* + \frac{5}{6}seq(V_{1p}^*[G_1]) + \frac{1}{2}seq(V_{0p}^*[G_0]) \tag{4.22}$$

We know that $S^* \ge seq(V_2)$, and 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 $seq(V_{1p}^*[G_1])) \le \frac{1}{3}seq(V_2)$. We can write the following:

$$\frac{\frac{5}{6}seq(V_{1p}^*[G_1])}{S^*} \le \frac{\frac{5}{6} \times \frac{1}{3}seq(V_2)}{seq(V_2)} \le \frac{5}{18} \tag{4.23}$$

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

$$\frac{\frac{1}{2} seq(V_{0_p}^*[G_0])}{S^*} \leq \frac{\frac{1}{2} \times \frac{1}{3} seq(V_1)}{seq(V_1)} \leq \frac{1}{6}. \tag{4.24}$$

Finally, with equations (4.22), (4.23) and (4.24) we conclude the proof:

$$\frac{S}{S^*} \leq 1 + \frac{5}{18} + \frac{1}{6} = \frac{13}{9} \tag{4.25}$$

$\square$

5. Conclusion

In this paper, we investigate a particular coupled-tasks scheduling problem $1|a_i = l_i = b_i, G|C_{\max}$ in presence of a compatibility graph with regard to the complexity and approximation. We also establish the $\mathcal{NP}$-completeness for the specific case where there is a bipartite compatibility graph. In such context, we propose a $\frac{7}{6}$-approximation algorithm and the bound is tight. We extend the result to the 2-stage bipartite by designing a 13/9-approximation.

A further interesting question concerns the study of the complexity on tree graphs with bounded degree. As we known, no complexity result exists. Another perspective consists in extending the presented results to k-stage bipartite graphs.

References

[1] D. Ahr, J. Békési, G. Galambos, M. Oswald and G. Reinelt, An exact algorithm for scheduling identical coupled-tasks. *Math. Methods Oper. Res.* **59** (2004) 193–203.

[2] J. Błażewicz, K. Ecker, T. Kis, C.N. Potts, M. Tanas and J. Whitehead, *Scheduling of coupled tasks with unit processing times.* Technical report, Poznan University of Technology (2009).

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

[4] A. Caprara, H. Kellerer and U. Pferschy, The Multiple Subset Sum Problem. *SIAM J. Optimiz.* **11** (2000) 308–319.

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

[6] M. Dawande, J. Kalagnanam, P. Keskinocak, F.S. Salman and R. Ravi, Approximation Algorithms for the Multiple Knapsack Problem with Assignment Restrictions. *J. Combin. Optim.* **4** (2000) 171–186.

[7] J. Edmonds, Maximum matching and a polyhedron with 0, 1 vertices. *J. Res. Nat. Bur. Stand.* **69** (1965) 125–130.

[8] 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).

[9] R. Giroudeau, J.-C. König, F.K. Moulaï and J. Palaysi, Complexity and approximation for the precedence constrained scheduling problem with large communication delays. *Theoret. Comput. Sci.* **401** (2008) 107–119.

[10] 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. *Ann. Discrete Math.* **5** (1979) 287–326.

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

[12] H. Kellerer, R. Mansini, U. Pferschy and M.G. Speranza, An efficient fully polynomial approximation scheme for the subset-sum problem. *J. Comput. Syst. Sci.* **66** (2003) 349–370.

[13] H.W. Kuhn, The hungarian method for the assignment problem. *Naval Res. Logist. Quart.* **2** (1955) 83–97.

[14] A.J. Orman and C.N. Potts, On the complexity of coupled-task scheduling. *Discrete Appl. Math.* **72** (1997) 141–154.

[15] R.D. Shapiro, Scheduling coupled tasks. *Naval Res. Logist. Quarterly* **27** (1980) 477–481.

[16] 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).

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

[18] G. Simonin, R. Giroudeau, J.-C. König and B. Darties, Theoretical Aspects of Scheduling Coupled-Tasks in the Presence of Compatibility Graph. *Algorithm. Oper. Res.* **7** (2012) 1–12.