Coupled-Tasks in Presence of Bipartite Compatibilities Graphs

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

Coupled-tasks in presence of bipartite compatibilities graphs

B. Darties[1], G. Simonin[2], R. Giroudeau[3], and J.-C. König[3]

[1] LE2I-CNRS-UMR 6306-8 Rue Alain Savary 21000 Dijon, France
[2] Insight Centre for Data Analytics, University College Cork, Ireland
[3] LIRMM-CNRS-UMR 5506-161, rue Ada 34090 Montpellier, France

Abstract. We tackle the makespan minimization coupled-tasks problem in presence of incompatibility 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 several problems in the framework of classic complexity and approximation for which the compatibility graph is bipartite (star, chain, ...). In such context, we design efficient polynomial-time approximation algorithms according to different parameters of the scheduling problem.

1 Introduction

We consider a non-preemptive coupled-tasks scheduling problem in presence of incompatibility constraint on a single processor. From the point of view of scheduling theory, the problem is also defined as a scheduling problem with exact delays on single machine. In this article, we will show the close relationship between coupled-task in presence of incompatibility constraint and the classic bin packing problem in the framework of complexity and approximation.

The coupled-tasks model, was first introduced by Shapiro [13] in order to model some data acquisition processes *i.e.* radar sensors: a sensor emits a radio pulse (first sub-task), and finally listen for an echo reply (second sub-task). Between these two instants (emission and reception), clearly there is an idle time due to the propagation, in both sides, of radio pulse. Therefore, a coupled-task is constituted by the triplet: the two sub-tasks and the idle between them. Thus, in order to minimise the makespan (schedule length), it is necessary to execute one or several different sub-tasks during the idle time of a coupled-task. Therefore, the aim is to find a best packing of coupled-tasks in which the sum of idle times is minimised. Notice that in the basic model, all coupled-tasks may be executed in each other according to processing time of sub-tasks and the duration of the idle time. Hereafter, we consider a relaxation of the previous model in which for a fixed coupled-task $\mathcal{A}$ there are only a subset of coupled-tasks compatible to be processed in the idle time of $\mathcal{A}$. This model is motivated by the problem of data acquisition in presence of incompatibility constraint in a submarine torpedo. A collection of sensors acquires data for the torpedo. The incompatibility constraint is expressed to prevent interference issues caused by tasks using sensors working

at the same frequency. So, the constraints are represented by a compatibility graph in which vertices are the coupled-tasks and edges represent compatibility between two tasks. In this article the variation of the complexity according to severals structural parameters are considered and some efficient polynomial-time approximation results on $\mathcal{NP}$-hard instances are presented without omitting the relationship to bin packing problems.

Above all, we will show the close relationship between the studied problem and four packing-related problems, for which known approximation will be used as routine for scheduling coupled-tasks problem:

1. The SUBSET SUM (SS) problem: given a set $\mathcal{S}$ of n positive values and $v \in \mathbb{N}$, the aim is to find a subset $\mathcal{S}^* \subseteq \mathcal{S}$ such that $\sum_{i \in \mathcal{S}^*} i = v$. This problem is known to be $\mathcal{NP}$-complete (see [8]). The optimization version is sometimes viewed as a KNAPSACK problem, where each item profit and weight 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: variant of BIN PACKING in which a number of identical bins are given and one aims to maximize the overall weight of the items packed in the bins without violating the constraint on the capacity of each bin. The problem is a special case of the MULTIPLE KNAPSACK problem in which all knapsacks have the same capacity and the item profits and weights coincide. MSS admits a $\mathcal{PTAS}$[2] and a $\frac{3}{4}-$approximation algorithm [3], but does not admit a $\mathcal{FPTAS}$ even for only two knapsacks.
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 in packing weighted items into non-identical capacity-constrained bins, with the additional constraint that each item can be packed into 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 is equal to its weight, [5] proposed a $\frac{1}{2}$-approximation.

2 Presentation of coupled-tasks and related work

We model a task $\mathcal{A}_i$ with a triplet (a_i, L_i, b_i), where a_i (resp. b_i) is the duration of the first (resp. second) sub-task, and L_i the idle time to respect between the execution of sub-tasks. We note $\mathcal{A}$ the set of tasks, and describe the incompatibility constraint between tasks with a graph $G_c = (\mathcal{A}, E)$. There is an edge $(\mathcal{A}_i, \mathcal{A}_j) \in E$ iff a (or both) sub-task from $\mathcal{A}_i$ may be scheduled during the idle time of $\mathcal{A}_j$ or reciprocally. In a valid schedule, we said that $\mathcal{A}_i$ is *packed* into $\mathcal{A}_j$ if the entire task $\mathcal{A}_i$ is scheduled during the idle time of $\mathcal{A}_j$. This is only possible when $a_i + L_i + b_i \leq L_j$. We call *stretched coupled-task* a task $\mathcal{A}_i$ such that $a_i = L_i = b_i = \alpha(A_i)$, where $\alpha(A_i)$ is the *stretch factor* of task $\mathcal{A}_i$. And for any set W of tasks, we define $\boldsymbol{seq(W) = 3 \sum_{x \in W} \alpha(x)}$.

Due to the combinatorial nature of the parameters of the problem, we use the Graham's notation scheme $\alpha|\beta|\gamma$ [9] (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 incompatibility constraints has been studied in the framework of classic complexity and approximation in [7, 12].

3 Stretched coupled-task: model and contribution

3.1 Model

This paper focuses on *stretched* coupled-tasks. In the rest of the paper, all tasks are always stretched coupled-tasks, and, for two compatible tasks A_j and A_j to be scheduled in parallel, one of the following conditions must hold:

1. $\alpha(A_i) = \alpha(A_j)$: 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. $3\alpha(A_i) \leq \alpha(A_j)$: task A_i is fully executed during the idle time L_j of A_j.

From this observation, one can obtain 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.

We note $N_G(v)$ the neighbourhood 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 vertices 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. For clarity, 1-stage bipartite graphs can be referred as triplet (X, Y, E) instead of (V_0, V_1, E).

4 Computational complexity

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

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.*
- $\mathcal{NP}$*-hard if the central node admits only incoming arcs.*

Proof

If there exists at least one outgoing arc $(x, y) \in G_c$ from the central node x, then the optimal solution consists in executing the x-task into the y-task, then in processing sequentially the remaining tasks after the completion of the y-task.

In the case where the central node admits only incoming arcs, first one can easily see that $1|\alpha(A_i) = a_i = L_i = b_i, G = star|C_{max}$ is $\mathcal{NP}$. Second, we propose the following polynomial construction from an instance of SS to an instance of our problem: $\forall i \in \mathcal{S}$ we add a coupled-task x with $\alpha(x) = i$; let $\mathcal{T}$ be the set of these tasks; we add a task y with $\alpha_y = a_y = L_y = b_y = 3 \times v$; we define an incompatibility constraint between each task $x \in \mathcal{T}$ and y modelled by the compatibility graph G. In brief, G is a star with y as the central node. From this transformation, one can easily show the reduction between both problems. □

Theorem 2. *The problem* $1|a_i = L_i = b_i = \alpha(A_i), G = chain|C_{max}$ *admits a polynomial-time algorithm.*

Sketch of proof Due to space limitation, we give only the main idea of the proof. Nevertheless, one can find the entire proof in the technical report [4].

The proof consists first in simplifying the original instance by defining some elements of an optimal solution, in order to obtain a sub-instance where in any solution at most one task can be packed in another. This new instance of the problem can be solved in polynomial time by reducing it to the search of a minimum weighted perfect matching. Basically, this reduction consists in duplicating the compatibility graph G, then in linking each node to its clone with an edge. From this new graph, we add for each edge $\{x, y\}$ a weight $w(\{x, y\})$ corresponding to half the execution time of these two tasks, and to the processing time of x if y is the clone of x. Then we perform a minimum weighted perfect matching in $O(n^2m)$ by [6]. □

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

Theorem 3. *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 $3 \times \alpha(x_1) + 3 \times \alpha(x_2) \leq \alpha(y)$ where x_1 and x_2 are the only two neighbors of Y, we add y to the schedule and execute $x1$ 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 4. *The problem of deciding whether an instance of* $1|a_i = L_i = b_i = \alpha(A_i), G_c = 1-stage\ bipartite, \Delta_{G_c} = 3|C_{max}$ *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 our problem is in $\mathcal{NP}$. Our proof is based on a reduction from ONE-IN-$(2,3)$SAT$(2,\bar{1})$: does there exist an assignment of 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 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.

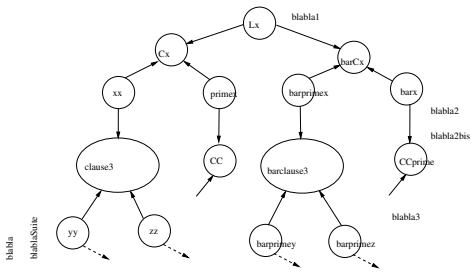

Fig. 1. A partial compatibility graph for the $\mathcal{NP}$-completeness of the scheduling problem $1|bipartite\ of\ depth\ one, d(G_c) \leq 3, \alpha_i = a_i = L_i = b_i|C_{max}$

We construct an instance π of our problem in following way (see Figure 1):

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 incompatibility 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 $(\bar{x}', \bar{C}^x)$.

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 incompatibility constraint from x to y; *i.e.*the coupled-task x may be executed during the idle of coupled-task y.

$\Rightarrow$ 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. 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 $54n = 3\alpha_{\mathcal{CT}}|\mathcal{CT}| + \alpha_{\mathcal{LT}}(|\mathcal{LT}| - |\{\mathcal{L}_x, x \in \mathcal{V}\}|) = 9|\{C^i \in \mathcal{CT} \text{ of length 2 and 3}\}| + 18|\{\bar{C}^i \in \mathcal{CT}\}| + 18|\{C^x \text{ and } \bar{C}^x \in \mathcal{LT}\}| = 9 \times \frac{4n}{3} + 18 \times \frac{n}{3} + 18 \times 2n$.
   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".

$\Leftarrow$ 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.

1. 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 $\bar{C}^x$.
2. 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 5. *$1|a_i = L_i = b_i = \alpha(A_i), G = star|C_{max}$ admits a $\mathcal{FPTAS}$.*

Proof We may use the solution given by the SUBSET SUM (SS) (see [10] and [11])). Indeed, the schedule is follows: first the central node is executed, second during its idle time we process the coupled-tasks chosen by an $\mathcal{FPTAS}$ algorithm from SS, and finally 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 tasks can be related to packing problems, especially when the compatibility graph G_c is a bipartite graph. In the following, we propose several approximation results 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-approximation, then the following problems*

1. $1|a_i = L_i = b_i = \alpha(A_i), G_c = 1\text{-}stage\ bipartite|C_{max}$,
2. $1|\alpha_i = a_i = L_i = b_i, complete\ 1\text{-}stage\ bipartite|C_{max}$
3. $1|\alpha_i = a_i = L_i = b_i, complete\ 1\text{-}stage\ bipartite|C_{max}$ *where the compatibility graph is a complete bipartite G=(X,Y), and all the tasks from Y have the same $\alpha(y)$.*

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

Proof

1. Let consider an instance of $1|\alpha_i = a_i = L_i = b_i, G_c = 1-stage\ bipartite|C_{max}$ with $G_c = (X, Y, E)$and a stretch factor function $\alpha : X \cup Y \to \mathbb{N}$. In such 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 tasks of X packed into a task from Y, and let $X_{\bar{p}}$ be 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_c, tasks from Y have to be scheduled sequentially in any (optimal) solution. The length of any schedule S is then the processing time of Y-tasks plus the execution time of the $X_{\bar{p}}$-tasks. Formally:

$$\begin{aligned} C_{max}(S) &= seq(Y) + seq(X_{\bar{p}}) \\ &= seq(Y) + seq(X) - seq(X_p). \end{aligned} \tag{2}$$

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 $\{x, y\}$ belongs to G_c.
Using algorithms and results from the literature, one can compute the set X_p of 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:

$$seq(X_p) \geq \rho \times seq(X_p^*), \tag{3}$$

where X_p^* is the set of packable items with the maximum profit. Combining Eq. (2) and (3), we obtain a schedule S with a length equal to:

$$C_{max}(S) \leq seq(Y) + seq(X) - \rho \times seq(X_p^*) \tag{4}$$

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 $X_p = X_p^*$. Therefore, the ratio obtained between our solution S and the optimal one S^* is:

$$\frac{C_{max}(S)}{C_{max}(S^*)} \leq \frac{seq(Y) + seq(X) - \rho \times seq(X_p^*)}{seq(Y) + seq(X) - seq(X_p^*)} \leq 1 + \frac{(1-\rho) \times seq(X_p^*)}{seq(Y) + seq(X) - seq(X_p^*)} \tag{5}$$

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: $seq(X_p^*) \leq \frac{1}{3} seq(Y)$. And thus combined to Eq. (5), we obtain the desired upper bound:

$$\frac{C_{max}(S)}{C_{max}(S^*)} \leq 1 + \frac{(1-\rho)}{3} \tag{6}$$

2. For the problem $1|\alpha_i = a_i = L_i = b_i, complete\ 1{-}stage\ bipartite|C_{max}$, the proof is identical using MSSDC as a special case of MKAR where each item can be packed in any bin.
3. For the problem $1|\alpha_i = a_i = L_i = b_i, complete\ 1{-}stage\ bipartite|C_{max}$ where all the tasks from Y have the same stretch factor $\alpha(y)$, the proof is identical as previously since MSSDC is a generalisation of MSS.

□

Theorem 6. *These problems admit a polynomial-time approximation algorithm:*

1. *The problem* $1|a_i = L_i = b_i = \alpha(A_i), G_c = 1$*-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*-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*-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 [5] proposed a $\rho = \frac{1}{2}-$approximation algorithm for MKAR. Reusing this result with Lemma 1, we obtain a $\frac{7}{6}-$approximation.
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 the Lemma 1, we obtain an approximation ratio of $1 + \frac{\epsilon}{3}$ for this problem.
3. (a) Authors from [3] proposed a $\rho = \frac{3}{4}-$approximation algorithm for MSS. Reusing this result and the Lemma 1, we obtain a $\frac{13}{12}-$approximation.
   (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 the Lemma 1, we obtain an approximation ratio of $1 + \frac{\epsilon}{3}$.

□

5.3 2−stage bipartite graph

Theorem 7. *The problem* $1|a_i = L_i = b_i = \alpha(A_i), G_c = 2$*-stage bipartite*$|C_{max}$ *is approximable to a factor* $\frac{13}{9}$.

Proof We consider an instance of the problem whith $G_c = (V_0 \cup V_1 \cup V_2, E_1 \cup E_2)$, where each arc in E_i is oriented from a vertex in V_i to another one in V_{i+1}, for $i \in 1, 2$.

Before presenting our heuristic and the analyse of its approximation factor, we will give several notations, properties and equations in relation with the specificities of this instance, in any (optimal) solution:

- $\forall i = 0, 1$, let V_{ip} (p=packed), (resp. V_{ia} (a=alone)) be the set of tasks merged (resp. remaining) into any task from V_{i+1} in a solution S, and V_{ib} (b=box) 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.
- Given any solution S to the problem and considering the specificities of the instance, note that $\{V_{0p}, V_{0a}\}$ is a partition of V_0, G_c, $\{V_{1p}, V_{1a}, V_{1b}\}$ is a partition of V_1, and G_c, $\{V_{2a}, V_{2b}\}$ is a partition of V_2.

– Any solution would consists in scheduling 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:

$$S^* = seq(V_{1_b}^*) + seq(V_2 b) + seq(V_{0_a}^*) + seq(V_{1_a}^*) + seq(V_{2_a}^*)$$
$$S^* = seq(V_2) + seq(V_{1_b}^*) + seq(V_{0_a}^*) + seq(V_{1_a}^*) \quad (7)$$

One can remark 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 $V_{2_b}^*$.

– Let consider an restricted instance of G_c to a sub-graph $G_0 = G_c[V_0 \cup V_1]$ (resp. $G_1 = G_c[V_1 \cup V_2]$) which is the 1-th (resp. 2-th) stage of G_c. Let $S[G_0]$ (resp. $S^*[G_0]$) be any (an optimal) solution on G_0, $V_{0_p}[G_0]$ (resp. $V_{0_p}^*[G_0]$) is the set of tasks from V_0 packed into tasks from V_1 in $S[G_0]$ (resp. $S^*[G_0]$), and $V_{0_a}[G_0]$ (resp. $V_{0_a}^*[G_0]$) the set of remaining tasks. In addition tho these notation, let $V_{1_b}[G_0]$ be the set of tasks from V_1 with at least one task from V_0 merged into them, and $V_{1_a}[G_0]$ the remaining tasks. A first observation gives for G_0:

$$S^*[G_0] = seq(V_1) + V_{0_a}^*[G_0] \quad (8)$$

– From Theorem 6, Lemma 1, and the demonstration presented in their proof from [5], several equations can be computed for a solution $S[G_0]$:

$$seq(V_{0_p}[G_0]) \geq \frac{1}{2} seq(V_{0_p}^*[G_0]) \quad (9)$$

$$seq(V_{0_a}^*[G_1]) \leq seq(V_{0_a}^*) \quad (10)$$

$$seq(V_{0_p}[G_0]) + seq(V_{0_a}[G_0]) = seq(V_{0_p}^*[G_0]) + seq(V_{0_a}^*[G_0]) = seq(V_0) \quad (11)$$

$$seq(V_{0_a}[G_0]) \leq seq(V_{0_a}^*[G_0]) + \frac{1}{2} seq(V_{0_p}^*[G_0]) \leq seq(V_{0_a}^*) + \frac{1}{2} seq(V_{0_p}^*[G_0]) \quad (12)$$

– We use an analog reasoning on the sub-graph G_1 with equivalent notations for V_1 and V_2, and we obtain:

$$seq(V_{1_p}[G_1]) \geq \frac{1}{2} seq(V_{1_p}^*[G_1]) \quad (13)$$

$$seq(V_{1_a}[G_1]) \leq seq(V_{1_a}^*[G_1]) + 1/2 seq(V_{1_p}^*[G_1]) \leq seq(V_{1_a}^*) + 1/2 seq(V_{1_p}^*[G_1]) \quad (14)$$

From this notations and observation, we can propose a good heuristic. 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. $V_{1_p}[G_1]$) in $S[G_1]$.
– Then we compute a solution $S[G_0]$ on G_0, then we add to S each task v from $V_{1_b}[G_0]/V_{1_p}[G_1]$ and the tasks from V_0 merged into them.
– Tasks $V_{1_a}[G_1]/V_{1_b}[G_0]$ and $V_{0_a}[G_0]$ are added to S 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:

$$seq(V_{1b}[G_0]/V_{1p}[G_1]) + seq(V_{1a}[G_1]/V_{1b}[G_0]) = V_{1a}[G_1] \tag{15}$$

Thus the cost of our solution S is

$$S = seq(V_2) + seq(V_{1a}[G_1]) + seq(V_{0a}[G_0]) + seq(V_{conflict}) \tag{16}$$

It is also clear that:

$$seq(V_{conflict}) \leq \frac{1}{3} seq(V_{1p}[G_1]) \leq \frac{1}{3} seq(V_{1p}^*[G_1]) \tag{17}$$

Using Equations (12), (14) and (17) in Equation (16), we obtain

$$S \leq seq(V_2) + seq(V_{1a}^*) + \frac{5}{6} seq(V_{1p}^*[G_1]) + seq(V_{0a}^*) + \frac{1}{2} seq(V_{0p}^*[G_0]) \tag{18}$$

$$\leq S^* + \frac{5}{6} seq(V_{1p}^*[G_1]) + \frac{1}{2} seq(V_{0p}^*[G_0]) \text{ , using Equation (7)} \tag{19}$$

We know that $S^* \geq seq(V_2)$ and $S^* \geq seq(V_1)$, as V_1 is an independent set of G_c. We also know that tasks from $(V_{1p}^*[G_1])$ (resp. $(V_{0p}^*[G_0])$) must be merged into tasks from V_2 (resp. V_1) and cannot exceed the idle time of V_2 (resp. V_1), implying that $seq(V_{1p}^*[G_1])) \leq \frac{1}{3} seq(V_2)$ (resp. $seq(V_{0p}^*[G_0])) \leq \frac{1}{3} seq(V_1)$). One can write the following :

$$\frac{\frac{5}{6} seq(V_{1p}^*[G_1])}{S^*} \leq \frac{\frac{5}{6} \times \frac{1}{3} seq(V_2)}{seq(V_2)} \leq \frac{5}{18} \tag{20}$$

$$\frac{\frac{1}{2} seq(V_{0p}^*[G_0])}{S^*} \leq \frac{\frac{1}{2} \cdot \frac{1}{3} seq(V_1)}{seq(V_1)} \leq \frac{1}{6} \tag{21}$$

Finally, from Equations (19), (20) and (21) the proof is concluded:

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

□

6 Conclusion

The results proposed in this paper are summarised in Table 1. 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 generalisation 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.

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

Table 1. Complexity and approximation results.

Acknowledgment

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

References

1. Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. A PTAS for the Multiple Subset Sum Problem with different knapsack capacities. *Inf. Process. Lett*, 2000.
2. Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. The Multiple Subset Sum Problem. *Siam Journal on Optimization*, 11(2):308–319, 2000.
3. Alberto Caprara, Hans Kellerer, and Ulrich Pferschy. A 3/4-Approximation Algorithm for Multiple Subset Sum. *J. Heuristics*, 9(2):99–111, 2003.
4. Benoit Darties, Gilles Simonin, Rodolphe Giroudeau, and Jean-Claude König. Scheduling stretched coupled-tasks with compatibilities constraints : model, complexity and approximation results for some class of graphs. Report, February 2014.
5. 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.
6. J. Edmonds. Maximum matching and a polyhedron with $0,1$ vertices. *Journal of Research the National Bureau of Standards*, 69 B:125–130, 1965.
7. 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.
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 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.
10. 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.
11. 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.
12. J.-C. König, G. Simonin, and R. Giroudeau. Complexity and approximation for scheduling problem for coupled-tasks in presence of compatibility tasks. In *Project Management and Scheduling*. 2010.
13. R D Shapiro. Scheduling coupled tasks. *Naval Research Logistics Quarterly*, 27:477–481, 1980.