Isomorphic coupled-task scheduling problem with compatibility constraints on a single processor

Gilles Simonin*, Benoît Darties**, Rodolphe Giroudeau*, and Jean-Claude König*

*LIRMM UMR 5506, rue Ada, 34392 Montpellier Cedex 5 - France
**LIG, CNRS, Grenoble-INP, 681 rue de la passerelle,
BP72 38402 Saint Martin d'Heres Cedex, France

contact : gilles.simonin@lirmm.fr

Abstract. The problem presented in this paper is a generalization of the usual coupled-tasks scheduling problem in presence of compatibility constraints. The reason behind this study is the data acquisition problem for a submarine torpedo. We investigate a particular configuration for coupled-tasks (any task is divided into two sub-tasks separated by an idle time), in which the idle time of a coupled-task is equal to the sum of its two sub-tasks. We prove $\mathcal{NP}$-completeness of the minimization of the schedule length, and we show that finding a solution to our problem amounts to solving a graph problem, which in itself is close to the minimum-disjoint path cover (min-DCP) problem. We design a $\left(\frac{3a+2b}{2a+2b}\right)$-approximation, where a and b (the processing time of the two sub-tasks) are two input data such as $a > b > 0$, and that leads to a ratio between $\frac{3}{2}$ and $\frac{5}{4}$. Using a polynomial-time algorithm developed for some class of graph of min-DCP, we show that the ratio decreases to $\frac{1+\sqrt{3}}{2} \approx 1.37$.

1 Introduction

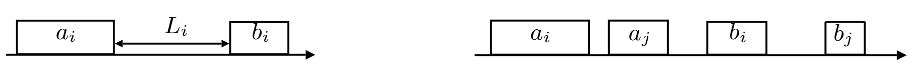
In this paper, we present a scheduling problem of coupled-tasks subject to compatibility constraints, which is a generalization of the scheduling problem of coupled-tasks first introduced by Shapiro [18]. This problem is motivated by the problem of data acquisition in a submarine torpedo. The aim amounts to treating various environmental data coming from sensors located on the torpedo, that collect information which must be processed on a single processor. A single acquisition task can be described as follows: a sensor of the torpedo emits a wave at a certain frequency (according to the data that must be collected) which propagates in the water and reflects back to the sensor. This acquisition task is divided into two sub-tasks: the first sends an echo, the second receives it. Between them, there is an incompressible idle time which represents the spread of the echo under the water. Thus acquisition tasks may be assigned to coupled-tasks.

In order to use idle time, other sensors can send more echoes. However, the proximity of the waves causes disruptions and interferences. In order to handle information error-free, a compatibility graph between acquisition tasks is created. In this graph, which describes the set of tasks, we have an edge between two compatible tasks. A task is compatible with another if at least one of its sub-tasks can be executed during the idle time of another task. Given a set of coupled-tasks and such a compatibility graph, the aim is to schedule the coupled-tasks in order to minimize the time required for the completion of all the tasks.

1.1 Problem formulation

For a graph G, we note $V(G)$ the set of its vertices and $E(G)$ the set of its edges. The cardinality of both sets are noted $n = |V(G)|$ and $m = |E(G)|$. In the following, we will call *path* a non-empty graph $C = (V, E)$ of the form $V = \{x_0, x_1, \ldots, x_k\}$ and $E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$, where the x_i are all distinct. The number of edges of a path corresponds to its length. The input of the general problem is described with a set $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ of coupled-tasks and a compatibility graph $G_c = (\mathcal{A}, E(G_c))$. Using the notation proposed by Shapiro [18], each task $A_i \in \mathcal{A}$ is composed of two sub-tasks a_i and b_i (the same notations are used for the processing time: $a_i \in \mathbb{N}$ and $b_i \in \mathbb{N}$), and separated by a fixed idle time $L_i \in \mathbb{N}$ (see fig. 1(a)). For each i the second sub-task b_i must start its execution exactly L_i time units after the completion time of a_i.

According to the torpedo problem, a task may be started during the idle time of a running task if it uses another frequency, is not dependant from the execution of the running task, or does not require to access to the resources used by the running tasks. Formally, we say that two tasks A_i and A_j are compatible if and only if we can execute at least a sub-task of A_i during the idle time of A_j (see fig. 1(b)). On the other side, some tasks cannot be compatible due to previously cited reasons. The compatibility graph G_c summarizes these compatibilities constraints, whose edges $E(G_c)$ represent all pairs of compatible task.



(a) (b)

Fig. 1. Understanding a single coupled-task and two compatible coupled-tasks.

A valid schedule $\sigma : \mathcal{A} \to \mathbb{N}$ consists in determining the starting time of each sub-task a_i of each task $A_i \in \mathcal{A}$. The tasks are processed on a single processor while preserving the constraints given by the compatibility graph.

The notation $\sigma(A_i)$ denotes the starting time of the task A_i. We use the following abuse of notation: $\sigma(a_i) = \sigma(A_i)$ (resp. $\sigma(b_i) = \sigma(A_i) + a_i + L_i$) denotes the starting time of the first sub-task a_i (resp. the second sub-task b_i).

Let $C_{max} = max_{A_i \in \mathcal{A}}(\sigma(A_i)+a_i+L_i+b_i)$ be the required time to complete all the tasks. Then the objective is to find a feasible schedule which minimizes C_{max}. At last, using the notation scheme $\alpha|\beta|\gamma$[1] proposed by Graham and al. [10], the main problem denoted as Π will be defined by $1|coupled-task,(a_i, b_i, L_i), G_c|C_{max}$.

1.2 Related work

The problem of coupled-tasks has been studied in regard to different conditions on the values of a_i, b_i, L_i for $1 \leq i \leq n$, and precedence constraints [1, 4, 16]. Note that, in the previous works, all tasks are compatible by considering a complete graph [1, 4, 16]. Moreover, in presence of any compatibility graph, we find several complexity results [19, 20], which are summarized[2] in Table 1:

Problem	Complexity
$1\|coupled-task,(a_i = b_i = L_i), G_c\|C_{max}$	$\mathcal{NP}$-complete
$1\|coupled-task,(a_i = a, b_i = b, L_i = L), G_c\|C_{max}$	$\mathcal{NP}$-complete
$1\|coupled-task,(a_i = b_i = p, L_i = L), G_c\|C_{max}$	$\mathcal{NP}$-complete
$1\|coupled-task,(a_i = L_i = p, b_i), G_c\|C_{max}$	Polynomial
$1\|coupled-task,(a_i, b_i = L_i = p), G_c\|C_{max}$	Polynomial

Table 1. Complexity for scheduling problems with coupled-tasks and compatibility constraints

Our work consists in measuring the impact of the compatibility graph on the complexity and approximation of scheduling problems with coupled-tasks on a mono processor. In this way, we focus our work on establishing the limits between polynomiality and $\mathcal{NP}$-completeness of these problems according to some parameters, when the compatibility constraint is introduced. In [20, 21], we have studied the impact of the parameter L, and have shown that the problem $1|coupled-task,(a_i = b_i = p, L_i = L), G_c|C_{max}$ was $\mathcal{NP}$-complete as soon as $L \geq 2$, and polynomial otherwise. In the following, we complete complexity results with the study of other special cases according to the value of a_i and b_i, and we propose several approximation algorithms.

1.3 Organization of this paper

In the rest of the paper, we restrict our study to a special case, by adding new hypotheses to the processing time and idle time of the tasks. We consider the processing time a_i (resp. b_i) of all sub-tasks a_i (resp. b_i) equal to a constant a

[1] Where α denotes the environment processors, β the characteristics of the jobs and γ the criteria.

[2] The notation $a_i = a$ implies that for all $1 \leq i \leq n$, a_i is equal to a constant $a \in \mathbb{N}$. This notation can be extended to b_i and L_i with the constants b, L and $p \in \mathbb{N}$.

(resp. b), and $\forall i \in \{1,\dots,n\}$ the length of the idle time is L. Considering homogeneous tasks is a realistic hypothesis according to the tasks that the torpedo has to execute. Let $\mathbf{\Pi_1 = 1|coupled-task, (a_i = a, b_i = b, L_i = L), G_c|C_{max}}$ be this new problem.

In section 2, we establish the complexity of Π_1 according to the values of a, b and L. On the one hand, we show that the problem is polynomial for any $L < a+b$. On the other hand, the problem becomes $\mathcal{NP}$-complete for $L \geq a+b$. When $L = a+b$ the problem may be considered as a new graph problem very close to the Minimum DISJOINT PATH COVER problem (Min-DPC). In section 3, we show that the problem is immediately 2-approximated by a simple approach, and we design a polynomial-time approximation algorithm with performance guarantee lower than $\frac{3}{2}$. In fact, we show that the approximation ratio obtained by this algorithm is between $\frac{3}{2}$ and $\frac{5}{4}$, according to the values of a and b. The last section is devoted to the study of Π_1 for some particular topology of the graph G_c. Some of presented results can be applied to the case $L > a+b$, which is not considered here and will be treated in future work.

2 Computational complexity

First, we prove that Π_1 is polynomial when $L < a+b$: it is obvious that a maximum matching in the graph G_c gives an optimal solution. Indeed, during the idle time L of a coupled-task A_i, we can process at most one sub-task a_j or b_k. Since the idle time L is identical, so it is obvious that finding an optimal solution consists in computing a maximum matching. Thus, the problem $1|coupled-task, (a_i = a, b_i = b, L_i = L < a+b), G_c|C_{max}$ admits a polynomial-time algorithm with complexity $O(m\sqrt(n))$ (see [17]).

The rest of the paper is devoted to the case $L = a+b$. Without loss of generality, we consider the case[3] of $b < a$. The particular case $b = a$ will be discussed in subsection 2.2.

2.1 From a scheduling problem to a graph problem

Let us consider a valid schedule σ of an instance $(\mathcal{A}, G_c)$ of Π_1 with $b < a$, composed of a set of coupled-tasks $\mathcal{A}$ and a compatibility graph G_c. For a given task A_i, at most two sub-tasks may be scheduled between the completion time of a_i and the starting time of b_i, and in this case the only available schedule consists in executing a sub-task b_j and a sub-task a_k during the idle time L_i with $i \neq j \neq k$ such that $\sigma(b_j) = \sigma(a_i) + a$ and $\sigma(a_k) = \sigma(a_i) + a + b$. Figure 2 shows a such configuration.

We may conclude that any valid schedule σ can be viewed as a partition $\{T_1, T_2, \dots T_k\}$ of $\mathcal{A}$, such that for any T_i the subgraph $P_i = G_c[T_i]$ of G_c induced

[3] The results we present here can be symmetrically extended to the instances with $b > a$.

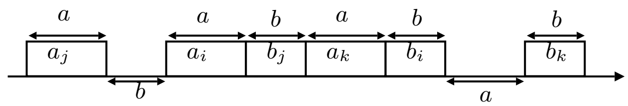
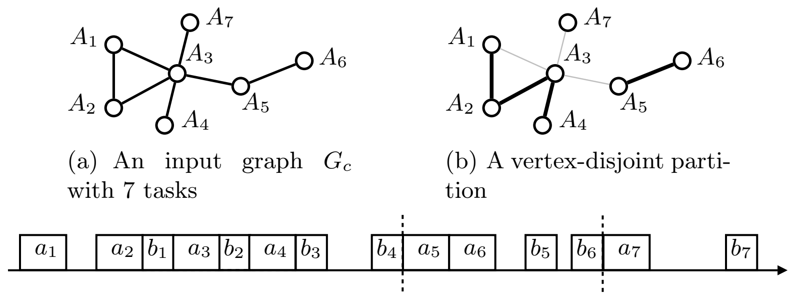


Fig. 2. At most two sub-tasks may be scheduled between a_i and b_i

by vertices T_i is a path (here, isolated vertices are considered as paths of length 0). Clearly, $\{P_1, P_2 \dots P_k\}$ is a partition of G_c into vertex-disjoint paths. Figure 3 shows an instance of Π_1 (fig. 3(a)), a valid schedule (fig. 3(c)) - not necessarily an optimal one -, and the corresponding partition of G_c into vertex-disjoint paths (fig. 3(b)).



(a) An input graph G_c with 7 tasks

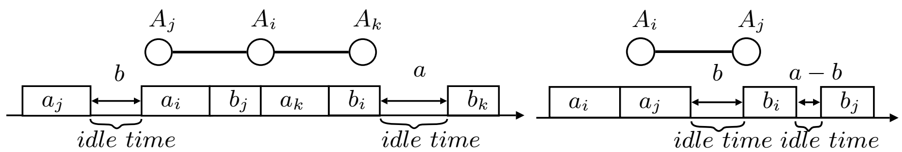
(b) A vertex-disjoint partition

(c) A schedule on a monoprocessor

Fig. 3. Relation between a schedule and a partition into vertex-disjoint paths

For a given feasible schedule σ, let us analyse the relation between the length of the schedule C_{max} and the corresponding partition $\{P_1, P_2, \dots P_k\}$ into disjoint-vertex paths. Clearly, we have $C_{max} = t_{seq} + t_{idle}$ where $t_{seq} = n(a + b)$ and t_{idle} is the inactivity time of the processor. Since t_{seq} is fixed for a given instance, t_{idle} obviously depends on the partition: for any path of length greater than 1, t_{idle} is incremented by $(a + b)$ (fig. 4(a)). A path of length 0 corresponds to a single task which increments t_{idle} by $L = a+b$. For any path of length 1, t_{idle} is only increased by a, because the two corresponding tasks may be imbricated as on figure 4(b).

Thus, finding an optimal schedule may be considered as a graph problem that we call **Minimum SCHEDULE-LINKED DISJOINT-PATH COVER (Min-SLDPC)** defined as follows:
Instance : a graph $G = (V, E)$ of order n, two natural integers a and b with $b < a$.
Result : a partition $\mathcal{P}$ of G into vertex-disjoint paths (which can be of size 0)



(a) Chains of length > 2 increase t_{idle} by $a+b$ (b) Chains of length 1 increase t_{idle} by a

Fig. 4. Impact of the length of the paths on the idle time

Objective : Minimize $n(a+b)+\sum_{p\in\mathcal{P}} w(p)$ where $w:\mathcal{P}\rightarrow\mathbb{N}$ is a cost function such that $w(p)=a$ if and only if $|E(p)|=1$, and $w(p)=a+b$ otherwise.

In any solution, each path increments the cost of the idle time by at least a (when the path has a length 1), and at most $a+b<2a$. So, we can deduce that an optimal solution to Min-SLDPC consists in finding a partition $\mathcal{P}$ with a particular cardinality k^*, and a maximal number of paths of length 1 among all possible k^*-partitions.

Clearly, Min-SLDPC is equivalent to Π_1 with $b<a$ and $L=a+b$, and can be viewed as the graph problem formulation of a scheduling problem. This problem is very close to the well-known **Minimum DISJOINT PATH COVER problem (Min-DPC)** which consists in covering the vertices of a graph with a minimum number of vertex-disjoint paths[4]. This problem has been studied in depth in several graph classes: it is known that this problem is polynomial on cographs [15], blocks graphs and bipartite permutation graphs [22], distance-hereditary graph [12], and on interval graphs [2]. In [5] and [9], the authors have proposed (independently) a polynomial-time algorithm in the case where the graph is a tree. Few years later, in [13] the authors showed that this algorithm can be implemented in linear time. Among the other results, there is a polynomial-time algorithm for the cacti [14], and another for the line graphs of a cactus [7]. In circular-arc graphs the authors [11] have proposed an approximation algorithm of complexity $O(n)$, which returns an optimal number of paths to a nearly constant additive equal to 1.

The problem Min-DPC is directly linked to the **HAMILTONIAN COMPLETION problem** denoted by **HC** [8], which consists in finding the minimum number of edges, noted $HC(G)$, that must be added to a given graph G, in order to make it hamiltonian (to guarantee the existence of a Hamiltonian cycle). It is known that if G is not hamiltonian, then the cardinality of a minimum disjoint path cover is clearly equal to $HC(G)$.

The dual of Min-DPC is found in the literature under the name Maximum DISJOINT-PATH COVER [8]. It consists in finding in G a collection of vertex-disjoint paths of length at least 1, which maximises the edges covered in G. This problem is known to be $\frac{7}{6}$-approximable [3].

[4] Sometimes referenced as the PATH-PARTITION problem (PP).

The following immediate theorem establishes the complexity of Min-SLDPC:

Theorem 1. *Min-SLDPC is an $\mathcal{NP}$-hard problem.*

Proof. Our proof is based on the polynomial-time transformation HAMILTONIAN PATH $\propto$ Min-SLDPC. Let us consider a graph G. The optimal solution of Min-SLDPC consists in finding a particular k-partition of G with k as small as possible. If k can be equal to 1, then the graph G contains an hamiltonian path. Reciprocally, if G contains an hamiltonian path, we can deduce a schedule with $C_{max} = (n+1)(a+b)$: as $t_{seq} = n(a+b)$, t_{idle} must be equal to $(a+b)$, which is possible if and only if the schedule is represented with only one chain.

2.2 A particular case $\Pi_2 = 1|coupled-task, (a_i = b_i = p, L_i = L = 2p), G_c|C_{max}$

Let us suppose, only in this subsection, that both sub-tasks are equal to a constant p and that the inactivity time is equal to a constant $L = 2p$.

The previous proof cannot be used for this case. Indeed the structure of these tasks allows to schedule three compatible tasks together without idle time (see figure 5). Another solution consists in covering vertices of G_c by triangles and paths (length 0 allowed), where we minimize the number of paths and then maximize the number of path of length 1.

This problem is a generalization of TRIANGLE PACKING [8] since an optimal solution without idle time consists in partitioning into triangles the vertices of G_c. This problem is well known to be $\mathcal{NP}$-complete, and leads to the $\mathcal{NP}$-completeness of problem Π_2.

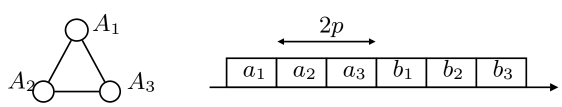


Fig. 5. Illustration of a schedule without idle time

A correct approximation algorithm for this problem is an algorithm close to the general case. Indeed, finding an optimal solution to this problem amounts to finding a covering of the graph G_c by triangles and paths, which minimize the idle time. In the following section, we will develop an efficient polynomial-time approximation algorithm for the general problem min-SLDPC.

3 Approximation algorithm for Min-SLDPC

First we show that any polynomial-time approximation algorithm[5] admits a performance guarantee of 2. We also develop a polynomial-time $\frac{3}{2}$-approximation

[5] Between two independent tasks, idle time is not allowed when there are available tasks.

algorithm based on a maximum matching in the graph G_c. In fact, we show that this algorithm has an approximation ratio of at most $\frac{3a+2b}{2a+2b}$, which leads to a ratio between $\frac{3}{2}$ and $\frac{5}{4}$ according to the values of a and b (with $b < a$). This result, which depends on the values a and b, will be discussed in section 4, in order to propose a better ratio on some class of graphs.

For any instance of Min-SLDPC, an optimal schedule has a length $C_{max}^{opt} = t_{seq} + t_{idle}^{opt}$ where $t_{seq} = n(a+b)$.

Remark 1. For any solution given by a heuristic h of cost C_{max}^h, we necessarily have[6] $t_{idle}^h \geq (a+b)$ and also[7] $t_{idle}^h \leq n(a+b)$. Then, for any solution h of Min-SLDPC we have a performance ratio $\rho(h)$ such that:

$$\rho(h) \leq \frac{C_{max}^h}{C_{max}^{opt}} \leq \frac{2n(a+b)}{(n+1)(a+b)} < 2. \tag{1}$$

In the following, we develop a polynomial-time approximation algorithm based on a maximum matching in the graph G_c, with performance guaranty in $[\frac{5}{4}, \frac{3}{2}]$ according to the values of a and b.

Let I be an instance of our problem. An optimal solution is a disjoint-paths cover. The n vertices are partitioned in three disjoint sets: n_1 uncovered vertices, n_2 vertices covered by $\alpha_2 = \frac{n_2}{2}$ paths of length 1, and n_3 vertices covered by exactly α_3 paths of length strictly greater than 1 (see illustration figure 6). An optimal solution is equal to the sum of sequential time and idle time: $C_{max}^{opt} = n(a+b)+n_1(a+b)+\frac{n_2}{2}a+\alpha_3(a+b)$.

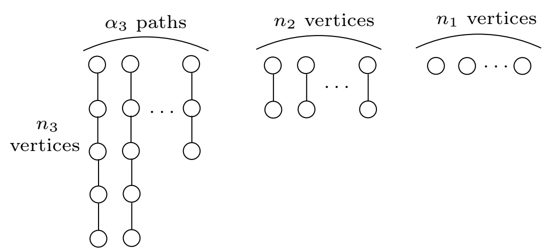


Fig. 6. Illustration of the optimal solution on an instance I

Now, we propose a polynomial-time approximation algorithm with non trivial ratio on an instance I. This algorithm is based on a maximum matching in

[6] The equality is obtained when the graph G_c possesses an hamiltonian path, otherwise we need at least two paths to cover G_c (where G_c is not only an edge), which leads to increase t_{idle}^h by at least $2a \geq a+b$ units of time.

[7] The worst case consists in executing tasks sequentially without scheduling any subtask a_j or b_j of task A_j during the idle time of a task A_i.

G_c in order to process two coupled-tasks at a time. For two coupled-tasks A_i and A_j connected by an edge of the matching, we obtain an idle time of length a (see figure 4(b)).

With this algorithm, the n_1 uncovered vertices (resp. the n_2 vertices covered by edges) in an optimal solution are still uncovered (resp. covered by edges) with a maximum matching. At last, we consider the worst case in which the α_3 paths have odd lengths. So, from the n_3 vertices, there are α_3 vertices uncovered and $(n_3 - \alpha_3)$ vertices covered with a maximum matching. The upper bound is:

$$\begin{aligned} C^h_{max} &= n(a+b) + n_1(a+b) + \frac{n_2}{2}a + \alpha_3(a+b) + (n_3 - \alpha_3)\frac{a}{2} \\ &\leq \frac{3na}{2} + nb + \frac{\alpha_3 a}{2} + \alpha_3 b + \frac{n_1 a}{2} + n_1 b \end{aligned}$$

Obviously, the worst case is when $n_1 = n_2 = 0$, $\alpha_3 = 1$ and $n_3 = n$. Thus, we obtain for the relative performance:

$$\rho(h) \leq \frac{C^h_{max}}{C^{opt}_{max}} \leq \frac{n(a+b) + (a+b) + (n-1)\frac{a}{2}}{n(a+b) + (a+b)} \leq \frac{3a+2b}{2a+2b} \tag{2}$$

4 Instances with particular topologies

We conclude this work by a study of min-SLDPC when G_c admits a particular topology. We show that finding a ρ_{dpc}-approximation for Min-DPC on G_c allows to find a strategy with performance ratio $\rho_{sldpc} \leq \min\{\rho_{dpc} \times (\frac{a+b}{a}), \frac{3a+2b}{2a+2b}\}$. This leads to propose, independently from the values a and b, a $\frac{1+\sqrt{3}}{2}$-approximation for min-SLDPC when Min-DPC can be polynomially solved on G_c.

From the literature, we know that Min-DPC is polynomial on trees [9, 13, 5], distance-hereditary graphs [12], bipartite permutation graphs [22], cactis [14] and many others classes. There are currently no result about the complexity of min-SLDPC on such graphs: since the values of of a and b have a high impact, techniques used to proove the polynomiality of Min-DPC cannot be adapted to proove the polynomiality of Min-SLDPC. Despite all our effort, the complexity of Min-SLDPC remains an open problem. However using known results on Min-DPC, we show how approximation ratio can be decreased for these class of graphs. We propose the following lemma:

Lemma 1. *If min-DPC can be polynomially solved, then there exists a polynomial-time* $\frac{(a+b)}{a}$*-approximation for min-SLDPC.*

Proof. Let $I_1 = (G)$ be an instance of min-DPC, and $I_2 = (G, a, b)$ an instance of min-SLDPC. Let $\mathcal{P}^*_1$ be an optimal solution of min-DPC of cost $|\mathcal{P}^*_1|$, and $\mathcal{P}^*_2$ an optimal solution of min-SLDPC of cost OPT_{sldcp}. According to the definition of min-SLDPC, we have $|\mathcal{P}_2^*| \geq |\mathcal{P}_1^*|$[8]. Since each path of a min-SLDPC solution increments the cost of the solution by at least a, then we have:

$$OPT_{sldcp} = \sum_{p \in \mathcal{P}_2^*} w(p) + n(a+b) \geq a|\mathcal{P}_2^*| + n(a+b) \geq a|\mathcal{P}_2^*| \quad (3)$$

$$\Rightarrow \frac{OPT_{sldcp}}{a} \geq |\mathcal{P}_2^*| \quad (4)$$

Let us analyse the cost given by the partition $\mathcal{P}_1^*$ if we consider it as a solution (not necessarily an optimal) to the instance I_2 of min-SLDPC. Since each path of a min-SLDPC solution increments the cost of the solution by at most $a+b$, then we have:

$$\begin{aligned}\sum_{p \in \mathcal{P}_1^*} w(p) + n(a+b) &\leq (a+b)|\mathcal{P}_1^*| + n(a+b) \leq (a+b)|\mathcal{P}_2^*| + n(a+b)\\ &\leq b|\mathcal{P}_2^*| + OPT_{sldcp} \qquad \text{according to (3)}\\ &\leq \frac{b}{a} OPT_{sldcp} + OPT_{sldcp} \qquad \text{according to (4)}\\ &\leq \frac{a+b}{a} OPT_{sldcp} \qquad (5)\end{aligned}$$

The same proof may be applied if there exists a ρ_{dpc}-approximation for min-DPC, and then there exists a $\rho_{dpc}\times(\frac{a+b}{a})$-approximation for min-SLDPC. Let us suppose that we know a constant ρ_{dpc} such that there exists a ρ_{dpc}-approximation for min-DPC on G_c. Let S_1 be the strategy, which consists in determining a $\rho_{dpc}\times(\frac{a+b}{a})$-approximation for min-SLDPC from ρ_{dpc}, and S_2 the strategy, which consists in using the algorithm introduced in section 3. Clearly, S_1 is particulary revelant when b is very small in comparison with a. Whereas S_2 gives better ratio when b is close to a. Both strategies are complementary along the value of b, which varies from 0 to a. Choosing the best result between the execution of S_1 and S_2, gives a performance ratio ρ_{sldpc} such that:

$$\rho_{sldpc} \leq \min\left\{\rho_{dpc} \times \left(\frac{a+b}{a}\right), \frac{3a+2b}{2a+2b}\right\}. \quad (6)$$

Compared to executing S_1 only, this new strategy increases the obtained results if and only if ρ_{dpc} is lower than $\frac{3}{2}$ (see fig. 5).

We propose the following remark, which is not good news:

Remark 2. There is no ρ_{dpc}-approximation for min-DPC in general graphs for some $\rho_{dpc} < 2$

[8] The best solution for min-SLDPC is not necessarily a solution with a minimum cardinality of k.

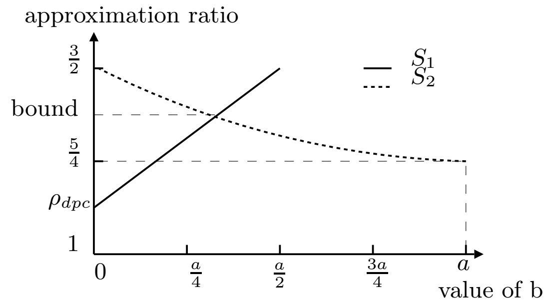


Fig. 7. Finding a good ρ_{dcp}-approximation helps to increase the results of section 3

This result is a consequence of the impossibility theorem [6]. It can be checked by considering an instance of min-DPC which has an hamiltonian path: the optimal solution of Min-DPC has cost 1, thus any polynomial-time ρ_{dpc}-approximation algorithm with $\rho_{dpc} < 2$ will return a solution of cost 1, which is not allowed under the assumption that $\mathcal{P} \neq \mathcal{NP}$. This result also implies that constant-factor approximation algorithms for max-DPC do not necessarily give the same performance guarantees on min-DCP, since the best approximation ratio for max-DCP is $\frac{7}{6}$, which is lower than the inapproximability bound for min-DCP.

It is good news that Min-DPC is polynomial for some compatibility graphs such as trees [9, 13, 5], distance-hereditary graphs [12], bipartite permutation graphs [22], cactis [14] and many others classes; thus $\rho_{dpc} = 1$. For all these graphs we obtain an approximation ratio of $\min\{\frac{(a+b)}{a}, \frac{3a+2b}{2a+2b}\}$ which is maximal when $\frac{(a+b)}{a} = \frac{3a+2b}{2a+2b}$, i.e.:

$$\frac{(a+b)}{a} = \frac{3a+2b}{2a+2b} \Leftrightarrow -a^2 + 2ab + 2b^2 = 0. \tag{7}$$

The only solution of this equation with a and $b \geq 0$ is $a = b(1+\sqrt{3})$. By replacing a by this new value on $\frac{(a+b)}{a}$ or on $\frac{3a+2b}{2a+2b}$, we show that in the worst case the approximation ratio is reduced from $\frac{3}{2}$ down to $\frac{1+\sqrt{3}}{2} \approx 1.37$.

5 Conclusion

We investigate a particular coupled-tasks scheduling problem Π_1 in presence of a compatibility graph. We have shown how this scheduling problem can be reduced to a graph problem. We have proved that adding the compatibility graph leads to the $\mathcal{NP}$-completeness of Π_1, whereas the problem is obviously polynomial when there is a complete compatibility graph (each task is compatible with each other). We have proposed a ρ-approximation of Π_1 where ρ is between $\frac{3}{2}$ and

$\frac{5}{4}$ according to value of a and b. We have also decreased the upper bound of $\frac{3}{2}$ down to ≈ 1.37 on instances where the Minimum DISJOINT PATH COVER problem can be polynomially solved on the compatibility graph.

The perspectives this work opens are measuring the pertinence of our polynomial-time approximation algorithm and classify $\Pi_1 = 1|coupled-task, (a_i = a, b_i = b, L_i = L)|C_{max}$ with complete compatibility graph.

References

1. D. Ahr, J. Békési, G. Galambos, M. Oswald, and G. Reinelt. An exact algorithm for scheduling identical coupled-tasks. *Mathematical Methods of Operations Research*, 59:193–203(11), June 2004.
2. S. Rao Arikati and C. Pandu Rangan. Linear algorithm for optimal path cover problem on interval graphs. *Information Processing Letters*, 35(3):149–153, 1990.
3. P. Berman and M. Karpinski. 8/7-approximation algorithm for (1,2)-tsp. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 641–648, New York, NY, USA, 2006. ACM.
4. J. Blazewicz, K.H. Ecker, T. Kis, and M. Tanas. A note on the complexity of scheduling coupled-tasks on a single processor. *Journal of the Brazilian Computer Society*, 7(3):23–26, 2001.
5. F. T. Boesch S. Chen and B. McHugh. On covering the points of a graph with point-disjoint paths. *Graphs and Combinatorics*, 406:201–212, 1974.
6. Ph. Chrétienne and C. Picouleau. Scheduling with communication delays: a survey. In *Scheduling theory and its applications*, pages 641–648. John Wiley & sons, 1995.
7. P. Detti and C. Meloni. A linear algorithm for the hamiltonian completion number of the line graph of a cactus. *Discrete Applied Mathematics*, 136(2-3):197–215, 2004.
8. M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. Freeman, 1979.
9. S. E. Goodman, S. T. Hedetniemi, and P. J. Slater. Advances on the hamiltonian completion problem. *Journal of the ACM*, 22(3):352–360, 1975.
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. *Annals of Discrete Mathematics*, 5:287–326, 1979.
11. R.-W. Hung and M.-S. Chang. Solving the path cover problem on circular-arc graphs by using an approximation algorithm. *Discrete Applied Mathematics*, 154(1):76–105, 2006.
12. R.-W. Hung and M.-S. Chang. Finding a minimum path cover of a distance-hereditary graph in polynomial time. *Discrete Applied Mathematics*, 155(17):2242–2256, 2007.
13. S. Kundu. A linear algorithm for the hamiltonian completion number of a tree. *Information Processing Letters*, 5:55–57, 1976.
14. S. Moran and Y. Wolfstahl. Optimal covering of cacti by vertex-disjoint paths. *Theoretical Computer Science*, 84:179–197, 1988.
15. K. Nakano, S. Olariu, and A. Y. Zomaya. A time-optimal solution for the path cover problem on cographs. *Theoretical Computer Science*, 290(3):1541–1556, 2003.
16. A.J. Orman and C.N. Potts. On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72:141–154, 1997.
17. A. Schrijver. *Combinatorial Optimization : Polyhedra and Efficiency (Algorithms and Combinatorics)*. Springer, July 2004.

18. R.D. Shapiro. Scheduling coupled tasks. *Naval Research Logistics Quarterly*, 27:477–481, 1980.
19. G. Simonin. Étude de la complexité de problèmes d'ordonnancement avec tâches-couplées sur monoprocesseur. *Majecstic '08*, 2008.
20. G. Simonin, R.Giroudeau, and J.-C. König. Complexity and approximation for scheduling problem for a torpedo. *Technical rapport*, January 2009.
21. G. Simonin, R.Giroudeau, and J.-C. König. Extended matching problem for a coupled-tasks scheduling problem. *Technical rapport*, January 2009.
22. R. Srikant, R. Sundaram, K. Sher Singh, and C. Pandu Rangan. Optimal path cover problem on block graphs and bipartite permutation graphs. *Theoretical Computer Science*, 115(2):351–357, 1993.