Extended Matching Problem for a Coupled-Tasks Scheduling Problem

Gilles Simonin, Rodolphe Giroudeau, Jean-Claude König

Extended matching problem for a coupled-tasks scheduling problem

G. Simonin, R. Giroudeau and J.-C. König
LIRMM - CNRS, UM2 UMR 5506
161 rue Ada, 34392 Montpellier Cedex 5 - France
simonin@lirmm.fr

Abstract

This paper introduces a scheduling problem with coupled-tasks in presence of a compatibility graph on a single processor. We investigate a specific configuration, in which the coupled-tasks possess an idle time equal to 2. The complexity of these problems will be studied according to the presence or absence of triangles in the compatibility graph. As an extended matching, we propose a polynomial-time algorithm which consists in minimizing the number of non-covered vertices, by covering vertices with edges or paths of length two in the compatibility graph. This type of covering will be denoted by 2-cover technique. According on the compatibility graph type, the 2-cover technique provides a polynomial-time ρ-approximation algorithm with $\rho = \frac{13}{12}$ (resp. $\rho = \frac{10}{9}$) in absence (resp. presence) of triangles.

1. Introduction

1.1. Presentation

In this paper, we present the problem of data acquisition according to compatibility constraints in a submarine torpedo, denoted by $TORPEDO$ problem. The torpedo is used in order to realize cartography, topology studies, temperature measures and many other tasks in the water. The aim of this torpedo is to collect and process a set of data as soon as possible on a single processor. In this way, it possess few sensors, a single processor and two types of tasks which must be scheduled: Acquisition tasks and Treatment tasks. First, the acquisition tasks $\mathcal{A} = \{A_1, \ldots, A_n\}$ can be assigned to coupled-tasks introduced by [1]. Indeed the torpedo sensors emit a wave which propagates in the water in order to collect the data. Each acquisition tasks A_i have two sub-tasks, the first a_i sends an echo, the second b_i receives it. For a better reading, we will denote the processing time of each sub-task a_i and b_i. Between the sub-tasks, there is an incompressible idle time L_i which represents the spread of the echo in the water. Second, treatment tasks $\mathcal{T} = \{T_1, \ldots, T_n\}$ are obtained from acquisition tasks. Indeed after the return of the echo, various calculations will be executed from gathered informations. These tasks are preemptive and have precedence constraints with the acquisition tasks. In this paper, we will consider the problem where every acquisition task A_i have a precedence relation with one treatment task T_i of one unit long (the processing time is denoted by T_i for a better reading).

Lastly, there exist compatibility constraints between acquisition tasks, due to the fact that some acquisition tasks cannot be processed at the same time that another tasks. In order to represent this constraint a compatibility graph $G_c = (\mathcal{A}, E_c)$ is introduced, where $\mathcal{A}$ is the set of coupled-tasks and E_c represents the edges connecting two coupled-tasks which can be executed simultaneously. In other words, at least one sub-task of a task A_i may be executed during the idle time of another task A_j (see example in Figure 1). There is none compatibility constraints between $\mathcal{T}$ and $\mathcal{A}$, and so, treatment tasks can be executed in the idle time of the acquisition tasks.

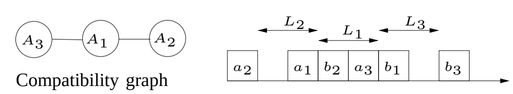


Figure 1. Example of compatibility constraints with L_i=2

The aim of the TORPEDO problem is to produce a shortest schedule (i.e. to minimize the moment after the execution of the last task in the schedule denoted by C_{max}) in which compatibility contraints between acquisition tasks and precedence constraints are respected. In scheduling theory, a problem is categorized by its machine environment, job characteristic and objective function. So using the notation scheme $\alpha|\beta|\gamma$ proposed by [2], the TORPEDO problem will be defined by $1|prec, (a_i,L_i,b_i) \cup (T_i = 1), G_c|C_{max}$[1].

Our work consists in measuring the impact of the compatibility graph on the complexity and approximation of scheduling problems with coupled-tasks on a single processor. This paper is focusing on the limit between polynomial problems and $\mathcal{NP}$-complete problems, when the compatibility constraint is introduced.

1.2. Related work

The complexity of the scheduling problem, with coupled-tasks and a complete compatibility graph[2], has been inves-

1. prec represents the precedence constraints between $\mathcal{A}$ and $\mathcal{T}$
2. Notice, the lack of compatibility graph is equivalent to a fully connected graph. In this way, all tasks may be compatible each other.

tigated by [3], [4], [5]. In existing works about coupled-tasks on a single processor, authors focus their studies on precedence constraints between the A_i's. We have studied the complexity of this type of problem according to the value of the different parameters, and we find the gap between the polynomial cases and $\mathcal{NP}$-complete ones. We have shown in [6] that the relaxation of the compatibility constraint imply the $\mathcal{NP}$-completeness of the problem $TORPEDO_1$: $1|prec,(a_i = b_i = 1, L_i = \alpha) \cup (T_i = 1), G_c|C_{max}$, in the case where $\alpha \geq 3$. In this article we present two results, first we will study a special case of $TORPEDO_1$ problem where $L_i = 2$, and so $t_{b_i} = t_{a_i} + a_i + L_i = t_{a_i} + 3$ where t_{a_i} is the starting time of a task a_i. Second, we will design an interesting polynomial-time approximation algorithm with non-trivial ratio guarantee for this problem, which can be generalized for the $TORPEDO$ problem.

1.3. Presentation of the TORPEDO problem

This section is devoted to definitions and notations used in the rest of the article. All the graphs in this paper are non-oriented. 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. A path of length k is denoted by C_k in the rest of the paper. Note that $k = 0$ is allowed, thus C_0 corresponds to a simple vertex. The study of the $TORPEDO_1$ problem depends on two essential points, the structure of the coupled-tasks and the compatibility graph G_c. This structure gives special constraints for the schedule which provides specific covering problems in G_c. To begin the study, we will investigate the four different possibilities of scheduling the coupled-tasks with this structure (see Figure 2).

Observation 1.1: The inactivity time between the two sub-tasks restraints the possibilities of scheduling. Indeed on the Figure 2 we can see the four types of scheduling, and so four types of covering on G_c. For each case, we have at most two slots and more than two treatment tasks which can be executed after the coupled-tasks. So, if we schedule triangles, chains, and edges the ones after the others, there is no idle slot (except from the first slot if there is no triangle). The only idle slots we can get, come from the simple vertices C_0. Because of their structure, they are the last to be scheduled.

These four types of scheduling immediately imply four types of covering in G_c: non-covered vertices, edges, paths of length greater than one, and triangles denoted by TR. The presence of triangles in G_c will raise problems in our study. For better results, the $TORPEDO_1$ problem is divided into two cases: depending on whether G_c contains triangles or not. We denote these problems $TORPEDO_1 + TR$ and $TORPEDO_1 - TR$.

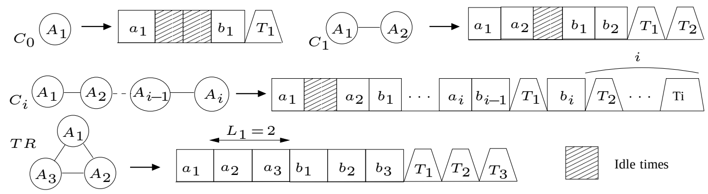


Figure 2. Illustration of the four types of scheduling

Theorem 1.1: $TORPEDO_1 + TR$ and $TORPEDO_1 - TR$ are $\mathcal{NP}$-complete.

It is not difficult to prove that $TORPEDO_1 + TR$ (resp. $TORPEDO_1 - TR$) can be reduced from the well-known triangle packing problem[3] (resp. hamiltonian path problem[4]).The proof of the Theorem 1.1 is given in Appendix 5.1.

With the previously hypotheses, it is clear to have an approach based on the covering of the vertices of G_c, but from [8] or [9] we know that covering a graph by paths of different length greater than two is $\mathcal{NP}$-complete. In order to obtain a good polynomial-time approximation algorithm in the two cases, we will use the same approach. It consists in finding a maximum covering of vertices with only edges and paths of length two. In the next section, we will define this covering and we will prove that it can be found in polynomial-time.

2. 2-cover definition

In the following, we will present several definitions concerning 2-cover.

Definition 2.1 (2-cover): Let $G = (V, E)$ be a graph, a 2-cover M is a set of edges such that the connected components of the partial graph induced by M are either simple vertices, edges, or paths of length two.

Definition 2.2 (M-covered vertex): An M-covered vertex (resp. M-non-covered) is a vertex which belongs (resp. does not belong) to at least one edge of M. The set of M-covered vertices (resp. M-non-covered vertices) will be denoted by $S(M)$ (resp. $NS(M)$).

Definition 2.3 (Maximum 2-cover): In a maximum 2-cover, the number of covered vertices is maximum, therefore the number of non-covered vertices is minimum.

Definition 2.4 (Vertex degree in relation to M): Let M be a 2-cover in a graph $G = (V, E)$. For each $i = 1, \ldots, k$, let $d_M(x_i)$ be the number of edges of M which are incident to x_i.

3. In a graph $G = (V, E)$, a triangle packing is a collection $V_1, \ldots, V_k$ of disjoint subsets of V, each containing exactly three vertices linked by three edges which belong to E (see [7]).

4. In a graph $G = (V, E)$, a hamiltonian path is a path compound by all the vertices of V (see [7]).

We will now give the definition of the alternating path in a 2-cover which is similar to the classical alternating path in a maximum matching by [10].

Definition 2.5 (M-alternating path): Let M be a 2-cover in a graph $G = (V, E)$, an M-alternating path $C = x_0, x_1, \ldots, x_k$ is a path in G such that for $i = 0, \ldots, \lceil \frac{k}{2} \rceil - 1$, $x_0 \in NS(M)$, $\{x_{2i}, x_{2i+1}\} \notin M$, and if $k \neq (2i+1)$ $\{x_{2i+1}, x_{2i+2}\} \in M$

Definition 2.6 (Vertebral column of an M-alternating path): Let M be a 2-cover in a graph $G = (V, E)$, and $C = x_0, x_1, \ldots, x_k$ an M-alternating path in G. The vertebral column, denoted by T, associated with the path C is composed of C, the edges of M which are incident to C, and eventually the extremity of these edges (see Figure 3).

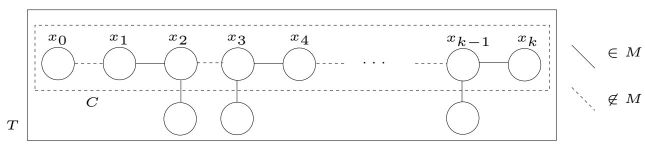
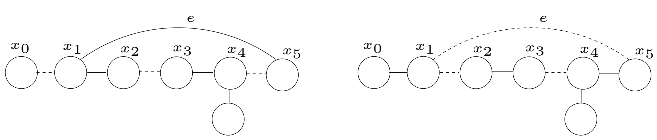


Figure 3. Example of a vertebral column T associated with an M-alternating path C

Remark 2.1: Therefore, if T contains a cycle, there exists $e \in M$ which links two vertices of C. If one of these vertices is not an extremity of C then we will have a path of length three in M (all the vertices are covered by edges of C except eventually the extremities). By definition, $x_0 \in NS(M)$, thus T contains a cycle when the last vertex x_k of C is connected to another vertex of C by an edge $e \in M$ and $e \notin C$ (see illustration in Figure 4(a)). Note that $d_M(x_k) = 1$.



(a) Example of a cycle in the vertebral column T

(b) Example of the augmenting operation with a cycle in T

Figure 4. Illustrations for the Remark 2.1

Definition 2.7 (Augmenting M-alternating path): Let $C = x_0, \ldots, x_k$ be an M-alternating path, and $x_0 \in NS(M)$. C is an augmenting M-alternating path if the cardinality of the 2-cover given by C can be increased (i.e. the number of non-covered vertices is reduced by at least one) by changing the belonging to M of the edges of C.

Remark 2.2: From Remark 2.1, a path of length three or four can be created thanks to the augmenting operation used in Definition 2.7. Let e be the edge of M which creates the cycle in T and thus creates the path of length three or four after the augmenting operation. Then, the edge e can be removed from M in order to increase the number of covered vertices by the 2-cover (see Figure 4(b)).

Definition 2.8 (Even vertex and odd vertex): Let $C = x_0, \ldots, x_k$ be an M-alternating path, and $x_0 \in NS(M)$. A vertex x_i with an index equal to an even number (resp. odd number) in C is called an even vertex (resp. odd vertex).

2.1. Results

This section is devoted to Lemma 2.1 about the augmenting M-alternating paths and the fundamental Theorem 2.1 of the 2-cover with M-alternating path.

Lemma 2.1: Let M be a 2-cover, $C = x_0, x_1, \ldots, x_k$ an M-alternating path with $x_0 \in NS(M)$, and let T be the vertebral column associated with the M-alternating path C. C is an augmenting M-alternating path if and only if there exists a vertex x_{2i-1}, $i \in \mathbb{N}^*$, such that $d_M(x_{2i-1}) \neq 2$.

Proof by contradiction

$\Rightarrow$ Suppose that C may be an augmenting M-alternating path, and suppose that an odd vertex x_{2i-1} such that $d_M(x_{2i-1}) \neq 2$ does not exist (so T does not contain any cycle). Thus, C and its vertebral column T have the shape of Figure 5.

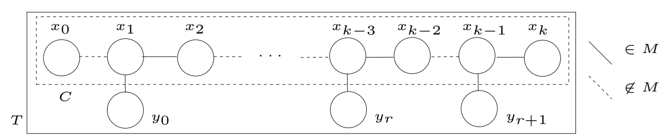


Figure 5. Skeleton of the vertebral column T associated to C

From Definition 2.7 and Remark 2.2, if T does not contain any cycle, we can simply increase the cardinality of the covered vertices in the path C by changing the belonging to M of the edges of C. If we change the belonging to M of the edge $\{x_0, x_1\}$ in order to cover x_0, the edge $\{x_1, x_2\}$ must change, else x_1 will be a star center[5]. In this way, we change the belonging to M of the edge $\{x_1, x_2\}$, which means that we must be changed $\{x_2, x_3\}$. Recursively, we will change the belonging to M of all C edges. Thus, the last vertex x_k will not be covered, and C will not be an augmenting M-alternating path. This is inconsistent with the former assumptions. Therefore, there exists a vertex x_{2i-1} such that $d_M(x_{2_i-1}) \neq 2$.

$\Leftarrow$ If T contains a cycle, then C contains an augmenting M-alternating path (see Figure 2.1). Else, suppose that T does not contain a cycle, but there exists a vertex x_{2i-1}, $i \in \mathbb{N}^*$, such that $d_M(x_{2i-1}) \neq 2$. We will show that C is an augmenting M-alternating path. Let $x_j = x_{2i-1}$, $i \in \mathbb{N}^*$, be the first odd vertex on the M-alternating path with $d_M(x_j) < 2$. We have three cases:

5. A star center is the vertex with degree greater than 1 in a star.

1) $d_M(x_j) = 0$, the M-alternating path C ends with an non-covered vertex. So C is an augmenting M-alternating path (see illustration in Figure 6.a).
2) $d_M(x_j) = 1$ and $d_M(x_{j+1}) = 1$, the M-alternating path C contains an edge $(x_j, x_{j+1}) \in M$ whose extremities have a degree equal to 1. We remove the part of the path which is after this edge, this part is already covered. Thus, we have an M-alternating sub-path, in which all the vertices of odd index have a degree equal to 2 and the sub-path end is an edge (x_j, x_{j+1}). It is easy to see that this sub-path is an augmenting M-alternating path by changing the belonging to M of the edges of C, except the last one (x_j, x_{j+1}). So C is an augmenting M-alternating path (see illustration in figure 6.b).
3) $d_M(x_j) = 1$ and $d_M(x_{j+1}) = 2$, the M-alternating path C owns an odd vertex with degree equal to 1 adjacent to an even vertex with degree equal to 2. We remove the path part which is after the even vertex with degree equal to 2, this part is already covered. Thus, we have an M-alternating sub-path, in which all the vertices of odd index have a degree equal to 2, and the sub-path end is a path of length two. It is easy to see that this sub-path is an augmenting M-alternating path by changing the belonging to M of all C edges. So C is an augmenting M-alternating path (see Figure 6.c).

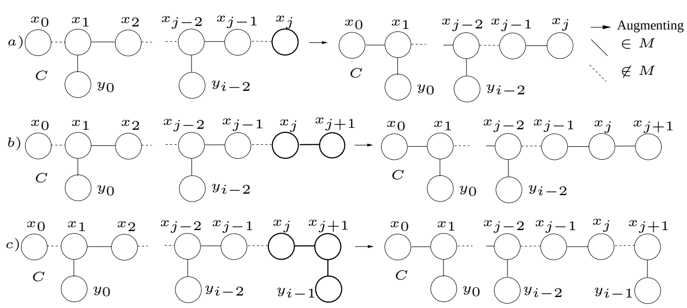


Figure 6. Augmenting operation of the three cases with $j = 2i - 1$

□

Theorem 2.1: Let M be a 2-cover in a graph G, M admits maximum cardinality if and only if G does not possess an augmenting M-alternating path.

Before giving the proof, we define two types of vertices in non-augmenting M-alternating paths:

Definition 2.9 (Leaf and root): Let M be a 2-cover in a graph G, and let C be a non-augmenting M-alternating path. A leaf (resp. a root) is defined as a vertex which admits only one neighbor (resp. two neighbors) in M. A vertex $x_j \in C$ with $j = 2i$, $i \in \mathbb{N}$, is a leaf. Moreover, all the vertices $y_i \in T \backslash C$ are also leaves. On the contrary, a vertex $x_j \in C$ with $j = 2i + 1$ is a root, $i \in \mathbb{N}$.

Proof of the Theorem 2.1

This proof is drawn from the classical maximum matching proof given in [10].

$\Rightarrow$ Let M be a maximum 2-cover in G. If G contains an augmenting M-alternating path, the cardinality of M will be able to increase by Lemma 2.1, and then M will be not maximum.

$\Leftarrow$ Let M_1 be a 2-cover in G. Suppose that G does not contain an augmenting M_1-alternating path.

Suppose that M_1 is not maximum, and let M_2 be another 2-cover in G which is maximum. Clearly, M_2 covers more vertices than M_1. From these hypotheses, the following structure is defined (see illustration in Figure 7):

- Suppose that M_2 covers K vertices non-covered by M_1, this set is denoted by $S_1 = \{x_i | x_i \in S(M_2) \cap NS(M_1)\}$.
- From any vertex x_i of S_1, there is necessarily an edge in G between x_i and a non-augmenting M_1-alternating path. Let S_2 be the set of vertices covered by M_1, which belongs to these non-augmenting M_1-alternating paths. $|S_2| = 3N$ is the number of covered vertices in these paths with N roots and $2N$ leafs.
- By hypothesis, we know that there may exist vertices covered by M_1, which do not belong to S_2. These vertices are covered either by edges or by paths of length two. Let $S_3 = \{x_i | x_i \in S(M_1) \wedge x_i \notin S_2\}$ be the set of these vertices.
- Lastly, there exists vertices non-covered by M_1 nor by M_2. This set is denoted by $S_4 = \{x_i | x_i \in NS(M_1) \cap NS(M_2)\}$.

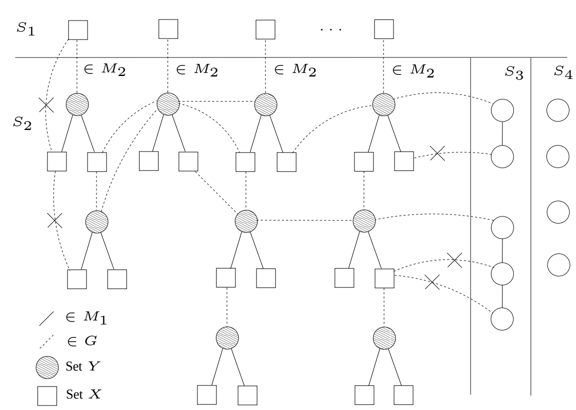


Figure 7. Diagram of the proof of Theorem 2.1

According to previous definitions, we can derive the following properties:

- S_1 is necessarily an independent set, otherwise there would be two vertices non-covered by M_1 connected by an edge. Then, we would have an augmenting M_1-alternating path.

- In set S_2, a root may be connected by an edge of G to any vertices of S_1 and S_2. A leaf of S_2 cannot be connected neither to another leaf of S_2 nor a vertex of S_1, by an edge of G. As a matter of fact, in both cases we would have either a cycle or an augmenting M_1-alternating path (see illustration in figure 7).
- Every root of S_2 may be connected by an edge of G to any vertices of S_3. If a leaf of S_2 is connected by an edge of G to an extremity of an edge or path of length two of S_3, then there will be an augmenting M_1-alternating path. And if a leaf of S_2 is connected to the center of a path of length two of S_3, this path will belong to S_2. Finally, leaves of S_2 may only be connected to roots of S_2 by an edge of G.
- We define two sets X and Y composed of all vertices which belong to $S_1 \cup S_2$. The first set X is composed of all the leaves of S_2 and of all the vertices of S_1, and its cardinality is $|X| = (2N + K)$. Furthermore, the set X is an independent set in regard to previous properties. The second set Y is composed of all the roots of S_2, and its cardinality is $|Y| = N$. Y represents the neighborhood of X, we denote $Y = \Gamma(X)$.
- For all 2-cover M, we have $|S(M) \cap X| \leq 2|Y|$, because $x \in S(M) \cap X \Rightarrow \exists e = \{x, y\} \in M$ and $y \in Y = \Gamma(X)$.

Now, we show that M_2 cannot cover more vertices than M_1, and thus $|M_2| = |M_1|$:

- Due to the fact that $|S(M_2)| > |S(M_1)|$ and that M_2 covers K vertices non-covered by M_1 in S_1, M_2 does not cover at most $(K-1)$ vertices in S_2. So, M_2 must cover at least $|X|-(K-1) = 2N+1$ vertices of X. From previously remarks, the number of covered vertices of X by any covering is inferior to $2|Y|$. There is a contradiction because $|2N+1| > 2|Y|$. As a result, there cannot exist a 2-cover M_2 such that $|M_2| > |M_1|$. So, $|M_2| = |M_1|$ and M_1 is a maximum 2-cover.

□

2.2. Polynomial-time algorithm for maximum 2-cover

From Theorem 2.1, we can now introduce the algorithm which gives a maximum 2-cover. Let M be a 2-cover, and let C be an augmenting M-alternating path. The algorithm substitutes covered edges for non-covered edges in C, except one of the edges at the end according to different cases. We denote this operation $Augmenting(M, C)$, which results in a new 2-cover which covers one or two vertices more than M. The algorithm which creates a maximum 2-cover is:

Algorithm 1: Research of a maximum 2-cover

Data: $G = (V, E)$
Result: A 2-cover M
begin
  $M :=$Ø
  while *there exists an augmenting M-alternating path C* **do**
    $M := Augmenting(M, C)$
  Return M
end

The algorithm which searches an augmenting M-alternating path from a non-covered vertex x_0, is based on ”breadth first search tree” where the root is x_0. For each vertex, we check if the distance to x_0 is odd, and then we select the first vertex whose degree is less than two according to M. This algorithm is:

Algorithm 2: Search of an augmenting M-alternating path

Data: $G = (V, E)$, with $|V| = n$, a vertex x_0, and M a 2-cover
Result: An augmenting M-alternating path C from the vertex x_0
begin
  Let Q (resp. Z) be a queue whose unique element is the vertex x_0 (resp. an empty queue)
  Let F be a function which gives the precedent vertex of an other given vertex
  while $Q \neq \emptyset$ **do**
    Let u be the first element of Q
    if $u \in Z$ **then**
      Push in Q the two neighbors of u according to M
    else
      for *every vertex v which is neighbor of u and $v \notin Z$* **do**
        $F[v] = u$
        if *v is a vertex of odd distance from x_0 and with degree $d_M(x_0) < 2$ according to M* **then**
          Return the augmenting path $C = \{x_0, \ldots, F(F(v)), F(v), v\}$
        else
          if *v is a vertex of odd distance from x_0* **then**
            Push v in Z
          Push v in Q
    Pull u of Q
end

The breadth first search has a complexity $O(n+m)$ where n (resp. m) is the number of vertices (resp. edges). In the worst case we search n times an augmenting M-alternating path. The Algorithm 1 is performed in $O(n^2)$.

3. Study of approximation

In this part, we present a polynomial-time approximation algorithm with a performance ratio bounded by $\frac{13}{12}$ for $TORPEDO_1 - TR$ and $\frac{10}{9}$ for $TORPEDO_1 + TR$.

3.1. First case: $\mathbf{TORPEDO_1 - TR}$

In this case, there will always be an idle time when we will schedule the first acquisition task covered in G_c by an edge or a path. From Observation 1.1, we will compute the number of idle slots after executing all the n coupled-tasks and treatment tasks. In this way, for one optimal scheduling, we extract an covering denoted by Sol_1. In this covering, let $Nb(C_i)$ be the number of paths C_i. In the following, we will have $n = Nb(C_0) + n_2$, where n_2 is the number of covered vertices in an optimal solution (see Figure 8). Now, we can define a function f which depends on all the $Nb(C_i)$ and counts the number of idle slots (except for the first represented by a **-1** in the equation) in the schedule, after the processing of treatment tasks within the slots created by coupled-tasks:

f = Idle slots from non-covered vertices – Treatment tasks remaining after the execution of paths $= 2Nb(C_0) + Nb(C_1) + (2\sum_{i=2}^{n_2-1} Nb(C_i) - \mathbf{1}) - (Nb(C_0) - 1) - 2Nb(C_1) - \sum_{i=2}^{n_2-1}[(i+1)Nb(C_i)] = Nb(C_0) - Nb(C_1) - \sum_{i=2}^{n_2-1}[(i-1)Nb(C_i)]$

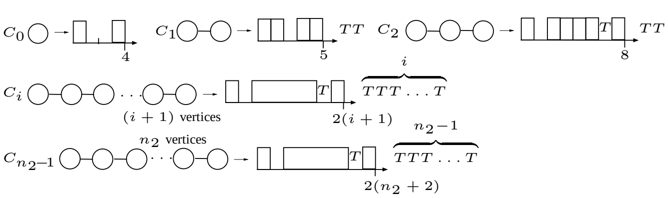


Figure 8. Different paths of the covering of G_c

According to f, the lower bound[6] will be equal to:

$$C_{max}^{opt} \geq T_{sequential} + T_{idle} = 3n + 1 + max\{0, f\} \quad (1)$$

Lemma 3.1: There exists an optimal solution to $TORPEDO_1 - TR$ that minimizes $Nb(C_0)$.

6. C_{max}^{opt} denotes the length of an optimal schedule, $T_{sequential}$ (resp. T_{idle}) denotes the processing time of all tasks (resp. idle times in the schedule).

Proof

It is obvious that minimizing f provides an optimal solution to $TORPEDO_1 - TR$. Let us show that if $Nb(C_0)$ is minimized in f, the value $Nb(C_1) + \sum_{i=2}^{n_2-1}(i-1)Nb(C_i)$ will be increased. Let's suppose that we have $Nb(C_0)$ non-covered vertices in a non-maximum covering M[7]. Now, we consider that M covers one more vertex x_0, there are two possibilities. If x_0 is connected in G_c to an edge of M, a path is created in M and f increases (indeed, we have two slots for the non-covered vertex plus one slot for the edge, whereas a path only has two slots). If x_0 is connected in G_c to a vertex of a path of M, two paths are created in M and f increases (indeed, cutting the chain in order to create two paths, edges included, gives four slots in the worst case). By minimizing the number of non-covered vertices, f either increases or remains the same, thus we obtain an optimal solution for $TORPEDO_1 - TR$. □

Remark 3.1: From the Lemma 3.1, the covering Sol_1, associated with the optimal scheduling, covers a maximum of vertices. In the following, we will call this covering a maximum covering.

Lemma 3.2: A maximum 2-cover minimizes the same number of non-covered vertices in G_c as any maximum covering with paths of different lengths (edges accepted).

Proof

The proof is trivial, indeed any path can be cut into edges and paths of length two. □

Let's search the upper bound, our heuristic is based on the 2-cover algorithm. From the Lemma 3.2 we know that the number of remaining non-covered vertices is also $Nb(C_0)$. Thus, the aim is to cut paths of different lengths into edges and paths of length two. Let $Nb^h(C_1)$ (resp. $Nb^h(C_2)$) be the number of edges (resp. paths of length two) in our heuristic. An edge creates one slot and leaves two treatment tasks, whereas a path of length two creates two slots and leaves two treatments tasks (because the third is used to fill one slot). For a better upper bound, we maximize the edges in the 2-cover after having minimized the non-covered tasks (see Lemma 5.1 in Appendix 5.2 for the proof). In Sol_1, for each path C_i of odd (resp. even) length, we have $(\frac{i+1}{2})$ edges (resp. $(\frac{i-2}{2})$ edges and one path of length two). Thus we obtain $Nb^h(C_1) = Nb(C_1) + \sum_{i(odd)=3}^{n_2-1}[(\frac{i+1}{2})Nb(C_i)] + \sum_{i(even)=2}^{n_2-1}[(\frac{i-2}{2})Nb(C_i)] = Nb(C_1) + \sum_{i=2}^{n_2-1}[(\frac{i+1}{2})Nb(C_i)] -$

7. In this paper, we call maximum covering a covering which covers a maximum of vertices with a minimum of paths.

$\sum_{i(even)=2}^{n_2-1} \frac{3}{2} Nb(C_i)$, and $Nb^h(C_2) = \sum_{i(even)=2}^{n_2-1} Nb(C_i)$.

Therefore, the length of the makespan is the sequential time plus the idle slots from the non-covered vertices which are not filled by treatment tasks. The upper bound[8] is $C^h_{max} \leq 3n+1+max\{0, Nb(C_0) - Nb^h(C_1) - Nb^h(C_2)\}$.

Now, we will study the relative performance ρ according to $Nb(C_0)$. First, we have $n = Nb(C_0) + 2Nb(C_1) + \sum_{i=2}^{n_2-1} (i+1)Nb(C_i)$, secondly from the equation 1, the worst case occurs when $Nb(C_0) = Nb(C_1) + \sum_{i=2}^{n_2-1} [(i-1)Nb(C_i)]$ (see Figure 9).

With the substitutions of $Nb(C_0)$ and n in C^{opt}_{max} and C^h_{max}, we obtain:

$$\bullet C^{opt}_{max} \geq 3n+1 = 1+3Nb(C_0)+6Nb(C_1)+3\sum_{i=2}^{n_2-1}[(i+1)Nb(C_i)]$$

$$= 1 + 9Nb(C_1) + 6\sum_{i=2}^{n_2-1}[iNb(C_i)]$$

$$\bullet C^h_{max} \leq 3n+1+Nb(C_0) - Nb^h(C_1) - Nb^h(C_2)$$

$$= 3n+1+\sum_{i=2}^{n_2-1}\left[\frac{(i-3)}{2}Nb(C_i)\right] + \frac{1}{2}\sum_{i(even)=2}^{n_2-1} Nb(C_i)$$

$$\leq 3n+1+\frac{1}{2}\sum_{i=2}^{n_2-1}[iNb(C_i)]$$

Thus, we have:

$$\rho \leq \frac{C^h_{max}}{C^{opt}_{max}} \leq \frac{3n+1+\frac{1}{2}\sum_{i=2}^{n_2-1}[iNb(C_i)]}{3n+1}$$

$$= 1 + \frac{\frac{1}{2}\sum_{i=2}^{n_2-1}[iNb(C_i)]}{1 + 9Nb(C_1) + 6\sum_{i=2}^{n_2-1}[iNb(C_i)]} \leq \frac{13}{12}$$

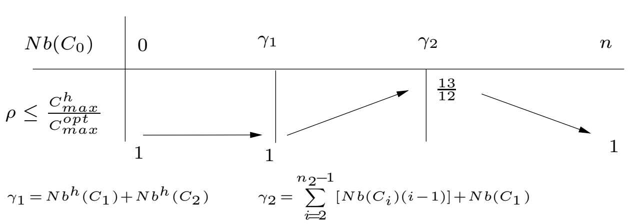


Figure 9. Variation of ρ depending on $Nb(C_0)$ value

The following algorithm first consists in minimizing $Nb(C_0)$, and secondly in maximizing $Nb(C_2)$ from a 2-cover. It gives a $\frac{13}{12}$-approximation for $TORPEDO_1 - TR$.

8. C^h_{max} denotes the length of the schedule given by our heuristic.

Algorithm 3: Use of 2-cover for $TORPEDO_1 - TR$

Data: $G_c = (V, E)$, with $|V| = n$
Result: C^h_{max}: length of the schedule with the heuristic
begin
- $M_1 :=$ Maximum matching in G_c
- $M_2 :=$ Maximum 2-cover from M_1
- $M_3 := Transformation(M_2)$
- Schedule vertices covered by edges in M_3, then by paths in M_3
- Schedule non-covered vertices, then schedule treatment tasks at the first idle time

end

Remark 3.2: $Transformation(M_2)$ is the operation in M_2 which turns the paths of length two into edges in polynomial time. Indeed, in a first time each path of length two in M is contracted into one vertex, which keeps the edges in G_c connected to the extremities of the path. Then in a second time, we search a maximum matching in this new graph with the contracted vertices. Finally, the contracted vertices are transformed into edges (illustration Figure 10).

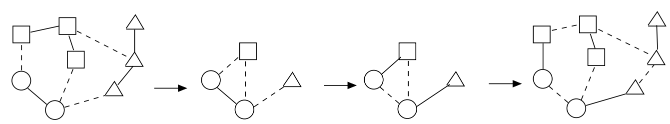


Figure 10. Illustration of the transformation from paths of length two into edges

3.2. Second case: $\mathbf{TORPEDO_1 + TR}$

This study is similar as previously. In this case we must simply add triangles to the set of all the paths in the covering of G_c, in order to obtain an optimal schedule. Let $Nb(TR)$ be the number of triangles in the covering, denoted by Sol_2, associated with the optimal schedule. With the triangles added to the solution, we have $n = Nb(C_0) + 2Nb(C_1) + \sum_{i=2}^{n_2-1}(i+1)Nb(C_i) + 3Nb(TR)$. For the lower bound, we obtain $C^{opt}_{max} \geq 3n + \mathbb{1}_{\{Nb(TR)=0\}} + max\{0, Nb(C_0) - Nb(C_1) - \sum_{i=2}^{n_2-1}[Nb(C_i)(i-1)] - 3Nb(TR)\}$[9].

In this case, minimizing $Nb(C_0)$ does not imply the minimization of $f = Nb(C_0) - Nb(C_1) - \sum_{i=2}^{n_2-1}[Nb(C_i)(i-1)] - 3Nb(TR)$. Indeed, in specific cases, it is wiser to leave a non-covered vertex in order to get a triangle and no idle slot in the scheduling. But for the calculus, we need that the

9. $\mathbb{1}_{\{Nb(TR)=0\}}$ is the indicator function which represents the first slot when there is no triangle in the covering.

number of C_0 in our heuristic, denoted by $Nb^h(C_0)$, equals to $Nb(C_0)$. In this way we can say without loss of generality that the worst case occurs when $Nb^h(C_0) = Nb(C_0)$.

Now, for the upper bound, the heuristic used is still a 2-cover, but in the case with triangles in G_c we cannot predict which vertices of the triangles will be covered by edges or paths of length two in the optimal solution. The worst case occurs when the triangles are covered by paths of length two, and thus $Nb^h(C_2) = \sum_{i(even)=2}^{n_2-1} Nb(C_i) + Nb(TR)$. For the upper bound we still have $C^h_{max} \leq 3n + 1 + max\{0, Nb(C_0) - Nb^h(C_1) - Nb^h(C_2)\}$.

As with the previous case, we will study the relative performance according to $Nb(C_0)$. The worst case occurs when $Nb(C_0) = Nb(C_1) + \sum_{i=2}^{n_2-1}[(i-1)Nb(C_i)] + 3Nb(TR)$.

We obtain:

- $C^{opt}_{max} \geq 3n + \mathbb{1}_{\{Nb(TR)=0\}} \geq 3n$
  $$\geq 9Nb(C_1) + 18Nb(TR) + 6\sum_{i=2}^{n_2-1}[iNb(C_i)]$$
- $C^h_{max} \leq 3n + 1 + Nb(C_0) - Nb^h(C_1) - Nb^h(C_2)$
  $$\leq 3n + 1 + 2Nb(TR) + \frac{1}{2}\sum_{i=2}^{n_2-1}[iNb(C_i)]$$

Thus, we have :

$$\rho \leq \frac{C^h_{max}}{C^{opt}_{max}} \leq \frac{3n+1+2Nb(TR)+\frac{1}{2}\sum_{i=2}^{n_2-1}[iNb(C_i)]}{3n + \mathbb{1}_{\{Nb(TR)=0\}}}$$

$$\leq 1 + \frac{1+2Nb(TR)+\frac{1}{2}\sum_{i=2}^{n_2-1}[iNb(C_i)]}{9Nb(C_1)+18Nb(TR)+6\sum_{i=2}^{n_2-1}[iNb(C_i)]}$$

$$\leq 1 + \frac{2Nb(TR)+\frac{6}{9}\sum_{i=2}^{n_2-1}[iNb(C_i)]}{18Nb(TR)+6\sum_{i=2}^{n_2-1}[iNb(C_i)]} \leq \frac{10}{9}$$

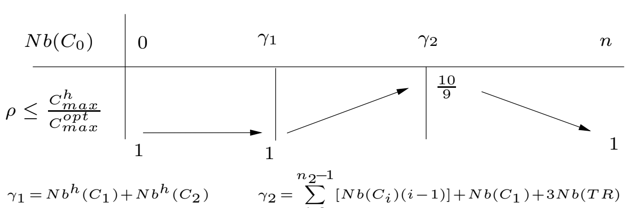


Figure 11. Variation of relative performance ρ dependent of $Nb(C_0)$ value

4. Conclusion

In this paper, we studied two $\mathcal{NP}$-complete scheduling problems with coupled-tasks where the idle time is equal to two. In order to approximate these problems, we introduced the notion of 2-cover which is an extension of the classical matching definition, and we developed the principle of alternating path according to this 2-cover. Then, we have shown two results for the 2-cover. Firstly, the cardinality of a 2-cover M is maximum when there are no augmenting M-alternating paths. Secondly, we defined a polynomial-time algorithm that yields a maximum 2-cover of a graph. From these results, we have shown that our heuristic, based on a 2-cover, provides an $\frac{13}{12}$-approximation for this problem if the compatibility graph has no triangle, and in the case of triangles, our heuristic gives an $\frac{10}{9}$-approximation. This heuristic based on a 2-cover let us suppose that it can be generalized for more general problems.

References

[1] R. Shapiro, “Scheduling coupled-tasks,” *Naval Research Logistics Quarterly*, vol. 27, pp. 477–481, 1980.

[2] R. Graham, E. Lawler, J. Lenstra, and A. R. Kan, “Optimization and approximation in deterministic sequencing and scheduling: a survey,” *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[3] J. Blazewicz, K. 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*, vol. 7, no. 3, pp. 23–26, 2001.

[4] A. Orman and C. Potts, “On the complexity of coupled-task scheduling,” *Discrete Applied Mathematics*, vol. 72, pp. 141–154, 1997.

[5] D. Ahr, J. Bksi, G. Galambos, M. Oswald, and G. Reinelt, “An exact algorithm for scheduling identical coupled-tasks,” *Mathematical Methods of Operations Research*, vol. 59, pp. 193–203(11), June 2004.

[6] G. Simonin, R. Giroudeau, and J.-C. König, “Complexity and approximation for scheduling problem for a torpedo,” *CIE’39, International Conference on Computers and Industrial Engineering, Troyes*, july 2009, to appears.

[7] M. Garey and D. Johnson, *Computers and Intractability; A Guide to the Theory of $\mathcal{NP}$-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[8] G. Steiner, “On the k-path partition of graphs,” *Theoretical Computer Science*, vol. 290, pp. 2147–2155, 2003.

[9] S. Masuyama and T. Ibaraki, “Chain packing in graphs,” *Algorithmica*, vol. 6, pp. 826–839, june 1991.

[10] C. Berge, “Two Theorems in Graph Theory,” *Proceedings of the National Academy of Science*, vol. 43, pp. 842–844, Sep. 1957.

5. Appendixes

5.1. Proof of $\mathcal{NP}$-completeness

5.1.1. [$\mathbf{TORPEDO_1 + TR}$ problem]. We will show that the $TORPEDO_1 + TR$ problem is $\mathcal{NP}$-complete when G_c contains triangles. In this way, we will use the triangle packing (TP) problem.

Proof

The construction of the polynomial time transformation is given for the reduction $TP \propto TORPEDO_1 + TR$. From an instance Π of TP, we built an instance Π' of $TORPEDO_1 + TR$. Let $G = (V, E)$ in Π with $|V| = n$, the construction of G_c in Π' consists in making the union of G and $(n-1)$ isolated vertices (see illustration Figure 12).

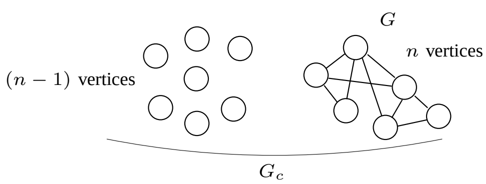


Figure 12. Illustration of the polynomial time transformation

$\Rightarrow$ Let's suppose that there exists a triangle packing in G, we will show that the scheduling of all the tasks of G_c admits a makespan of length $3(2n-1)$, the sequential time without idle slot. If there exists a triangle packing, so all vertices of G are covered, and there remain $(n-1)$ isolated vertices which cannot be covered. The scheduling of this covering is the following, first we process the coupled-tasks covered by the triangle packing, then the non-covered (isolated) coupled-tasks. Thanks to the processing of the treatment tasks, all the idle slots are filled. Thus, the scheduling length is equal to $3(2n-1)$.

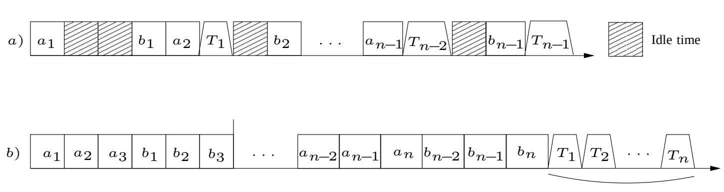


Figure 13. Illustration of the covering of G

$\Leftarrow$ Let's suppose that the scheduling of all the tasks of G_c in Π' admits a makespan of length $3(2n-1)$, we will show that the covering in G is a triangle packing. Notice that the length of the makespan is without idle slot in the scheduling. The scheduling of the isolated coupled-tasks is simple and gives n idle slots (see illustration Figure 13a). Because of the compatibility constraint between the isolated coupled-tasks and the other tasks in G, we can only fill the idle time of these isolated tasks with n treatment tasks. The scheduling of the tasks of G must give n treatment tasks, but it is possible only if all the coupled-tasks are processed without idle slot (see Figure 13b). Thus, the covering of the vertices of G is necessarily a triangle packing. $\square$

5.1.2. [$\mathbf{TORPEDO_1 - TR}$ problem]. We will show that the $TORPEDO_1 - TR$ problem is $\mathcal{NP}$-complete when G_c has no triangle. In this way, we will use the Hamiltonian path (HC) problem.

Proof

The construction of the polynomial time transformation is given for the reduction $HC \propto TORPEDO_1 - TR$. From an instance Π of HC, we built an instance Π' of $TORPEDO_1 - TR$. Let $G = (V, E)$ in Π with $|V| = n$, the construction of G_c in Π' consists in making the union of G and $(n-2)$ isolated vertices (see illustration Figure 14).

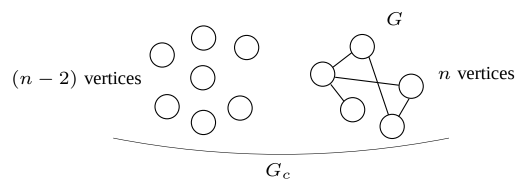


Figure 14. Illustration of the polynomial time transformation

$\Rightarrow$ Let's suppose that there exists a Hamiltonian path in G, we will show that the scheduling of all the tasks of G_c admits a makespan of length $3(2n-2)+1$, the sequential time plus one idle slot. If there exists a Hamiltonian path, so all vertices of G are covered, and there remains $(n-2)$ isolated vertices. The scheduling of this covering is the following, first we process the coupled-tasks covered by the Hamiltonian path, then the non-covered (isolated) coupled-tasks. Thanks to the processing of the treatment tasks, all the idle slots are filled except for the first idle slot created by the Hamiltonian path. Thus, the scheduling length is equal to $3(2n-2)+1$.

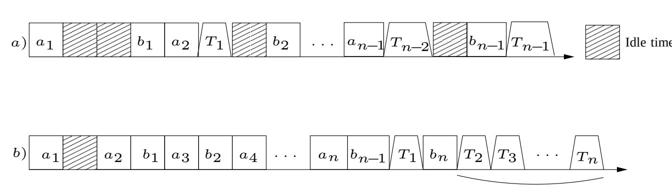


Figure 15. Illustration of the covering of G

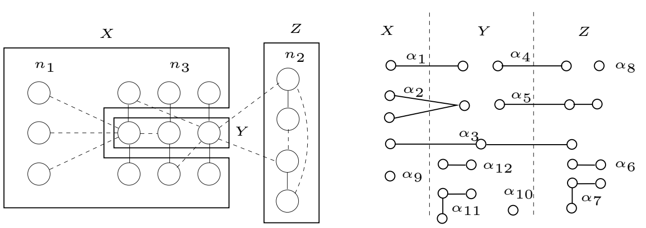
$\Leftarrow$ Let's suppose that the scheduling of all the tasks of G_c in Π' admits a makespan of length $3(2n-2)+1$, we will show that the covering in G is a hamiltonian path. Notice that the length of the makespan leaves only one idle slot in the scheduling. The scheduling of the isolated coupled-tasks is simple and gives $(n-1)$ idle slots (see illustration Figure 5.1.2a). Because of the compatibility constraint between the isolated coupled-tasks and the other tasks in G, we can only fill the idle time of these isolated tasks with $(n-1)$ treatment tasks. The scheduling of the tasks of G must give $(n-1)$ treatment tasks, but it is possible if all the coupled-tasks are processed with only two idle slots (see Figure 5.1.2b). Thus, the covering of the vertices of G is necessarily a hamiltonian path. $\square$

5.2. Proof of the lemma in order to find a maximum 2-cover

Lemma 5.1: A maximum 2-cover consists in firstly minimizing $Nb(C_0)$, then secondly maximizing $Nb(C_1)$.

Proof

Let's τ_1 be a maximum 2-cover of the graph G_c which first minimizes the non-covered vertices, then maximizes the edges in the cover. τ_1 is composed of n_3 (resp. n_2) vertices covered by paths of length two (resp. by edges), and n_1 non-covered vertices. Three sets are defined from τ_1 (see figure 16(a)): X which contains the $\frac{2n_3}{3}$ extremities of paths of length two and the n_1 non-covered vertices, Y contains the $\frac{n_3}{3}$ vertices of the middle of paths of length two, and Z contains the n_2 vertices covered by edges. X is an independent set and the fact that τ_1 does not contain an augmenting τ_1-alternating path implies that there cannot exist edges between X and Z.



(a) Illustration of the covering of τ_1

(b) Illustration of the three sets and the different types of covering in τ_2

Figure 16. Illustrations for the proof of the Lemma 5.1

The proof is by contradiction, let's suppose that there is another maximum 2-cover τ_2 in graph G_c which has its function f and its number of non-covered vertices lower than those of τ_1. From τ_1 and the three sets defined previously, we will give all the possible covers for τ_2 (see figure 16(b)). Let $\beta_{1,T}$ be the set of non-covered vertices in $T \in E$ where $E = \{X, Y, Z\}$. Let $\beta_{2,T,U}$ be the set of edges which has an extremity in $T \in E$ and the other in $U \in E$. And finally, let $\beta_{3,T,U,V}$ be the set of paths of length two which has an extremity in $T \in E$, another in $V \in E$ and the third vertex in $U \in E$.

Remark 5.1: With the definition of the three sets X, Y, Z, we have $\beta_{2,X,X} = \beta_{2,X,Z} = \emptyset$ and all the $\beta_{3,T,U,V}$ are empty except for $\beta_{3,X,Y,X}$, $\beta_{3,X,Y,Z}$, $\beta_{3,Y,Z,Z}$, $\beta_{3,Y,Y,Y}$ and $\beta_{3,Z,Z,Z}$. At least, $\forall U, V, T$ $\beta_{2,T,U} = \beta_{2,U,T}$ and $\beta_{3,T,U,V} = \beta_{3,V,U,T}$.

In order to make the proof more easy visibility, we have the following notations: $\alpha_1 = |\beta_{2,X,Y}|, \alpha_2 = |\beta_{3,X,Y,X}|, \alpha_3 = |\beta_{3,X,Y,Z}|, \alpha_4 = |\beta_{2,Y,Z}|, \alpha_5 = |\beta_{3,Y,Z,Z}|, \alpha_6 = |\beta_{2,Z,Z}|, \alpha_7 = |\beta_{3,Z,Z,Z}|, \alpha_8 = |\beta_{1,Z}|, \alpha_9 = |\beta_{1,X}|, \alpha_{10} = |\beta_{1,Y}|, \alpha_{11} = |\beta_{3,Y,Y,Y}|, \alpha_{12} = |\beta_{2,Y,Y}|$. And thus, we have the following equations:

$$\begin{aligned}
\alpha_9 &= n_1 + \frac{2n_3}{3} - \alpha_1 - 2\alpha_2 - \alpha_3 \\
\alpha_{10} &= \frac{n_3}{3} - \alpha_1 - \alpha_2 - \alpha_3 - \alpha_4 - \alpha_5 - 3\alpha_{11} - 2\alpha_{12} \\
\alpha_8 &= n_2 - \alpha_3 - \alpha_4 - 2\alpha_5 - 2\alpha_6 - 3\alpha_7
\end{aligned}$$

Now we can compute f_{τ_2} for τ_2, f_{τ_2} is the number of slots which stay after the processing of treatment tasks in the inactivity time of the non-covered acquisition tasks. f_{τ_2} is depending of the α_i and the n_i:

f_{τ_2} = Number of slots given by non-covered vertices $-$ Number of treatment tasks available $= (\alpha_{10} + \alpha_9 + \alpha_8) - (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_6 + \alpha_7 + \alpha_{11} + \alpha_{12}) = (n_1 + \frac{2n_3}{3} + \alpha_{10} - \alpha_1 - 2\alpha_2 - \alpha_3) - \alpha_1 - \alpha_2 - \alpha_3 - \alpha_4 - \alpha_5 - \alpha_6 - \alpha_7 + \alpha_8 - \alpha_{11} - \alpha_{12} = n_1 + \alpha_4 + \alpha_5 + \alpha_8 + 2\alpha_{10} + 5\alpha_{11} + 3\alpha_{12} - \alpha_2 - \alpha_6 - \alpha_7$

From hypothesis on τ_1 and τ_2, $f_{\tau_2} < f_{\tau_1}$, and so:

$$\begin{aligned}
n_1 &+ \alpha_4 + \alpha_5 + \alpha_8 - \alpha_2 - \alpha_6 - \alpha_7 < n_1 - \frac{n_2}{2} - \frac{n_3}{3} \\
\alpha_1 &+ \frac{3\alpha_3}{2} + \frac{5\alpha_4}{2} + 3\alpha_5 + \frac{\alpha_7}{2} + \frac{3\alpha_8}{2} + 4\alpha_{10} + 8\alpha_{11} + 5\alpha_{12} < 0
\end{aligned}$$

This equation is impossible because $\forall i$ $\alpha_i \geq 0$. So τ_2 does not exist, and τ_1 is an optimal 2-cover.

$\square$