2-cover definition 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 - CC 477
161 rue Ada, 34392 Montpellier Cedex 5 - France
(e-mail: simonin@lirmm.fr)

Abstract. This paper presents a scheduling problem with coupled-tasks in presence of a compatibility graph on a mono 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 2-cover technique. According on the compatibility graph type, the 2-cover technique provides an $\frac{13}{12}$-approximation or $\frac{10}{9}$-approximation algorithm.

1 Introduction

1.1 Presentation

In this paper, we present the problem of data acquisition according to compatibility constraints in a submarine torpedo, denoted $TORPEDO$ problem. The torpedo is used in order to make 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 mono processor. In this way, it possess few sensors, a mono processor and two types of tasks which must be schedule: Acquisition tasks and treatment tasks. First, the acquisition tasks $\mathcal{A} = \{A_1, \ldots, A_n\}$ can be assigned to coupled-tasks introduced by [8], 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. The processing time of sub-tasks are denoted p_{a_i} and p_{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 study the problem where every acquisition task have a precedence relation with one treatment task of one unit long.

At last, there exist compatibility constraints between acquisition tasks, due to the fact that some acquisition tasks cannot be processed in same the 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).

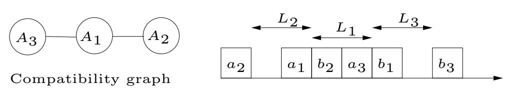


Fig. 1. Example of compatibility constraints with $L_i = 2$

In the 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 [5], the problem, denoted as TORPEDO, will be defined by $1|prec, (p_{a_i} = p_{b_i} = 1, L_i = 2) \cup (p_{T_i} = 1), G_c|C_{max}$[1].

1.2 Related work

The complexity of the scheduling problem, with coupled-tasks and a complete compatibility graph[2], has been investigated by [3] (i.e. $G_c = K_n$, [7], [1]. In existing works about coupled-tasks on a mono processor, authors focus their studies on precedence constraints between the A_i's. We have study the complexity of this type of problem according to the value of the different parameters, and we find the line between the polynomial cases and $\mathcal{NP}$-complete ones. We have shown in [9] that the relaxation of the compatibility constraint imply the $\mathcal{NP}$-completeness of the problem $TORPEDO_0 : 1|prec, coupled-task, (p_{a_i} = p_{b_i} = 1, L_i = \alpha) \cup (p_{T_i} = 1), G_c|C_{max}$, in the case where $\alpha \geq 3$. In this article we present two results, first we will study a special case of $TORPEDO_0$ problem where $L_i = 2$, and so $t_{b_i} = t_{a_i} + p_{a_i} + L_i = t_{a_i} + 3$ where t_{a_i} is the starting time of a task a_i. Second, we 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 definition and notation used in the rest of the article. All the graphs in this paper are non-oriented. We will call *path* a non-empty

[1] prec represents the precedence constraints between $\mathcal{A}$ et $\mathcal{T}$

[2] Notice, the lack of compatibility graph is equivalent to a fully connected graph. In this way, all tasks may be compatible each other.

graph $C = (V, E)$ of the form $V = \{x_0, x_1, \dots, x_k\}$ and $E = \{x_0x_1, x_1x_2, \dots, x_{k-1}x_k\}$, where the x_i are all distinct. The number of edges of a path corresponds to its length. The path of length k is denoted C_k in the rest of the paper. Note that $k = 0$ is allowed, thus C_0 is a simple vertex. The study of the TORPEDO 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 different ways of scheduling the coupled-tasks with this structure. There are four possibilities (see illustration Figure 2).

Observation 11 *The inactivity time between the two sub-tasks restraints the possibilities of scheduling, indeed on the illustration 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.*

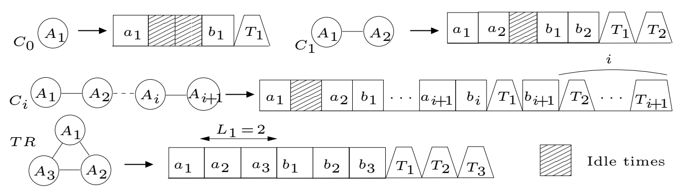


Fig. 2. Illustration of the four types of scheduling

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 TR. The presence of triangles in G_c will raise problems in our study. For better results, the TORPEDO problem is divided into two cases: depending to whether G_c contains triangles or not. We denote these problems TORPEDO+TR and TORPEDO-TR.

Theorem 11 *TORPEDO+TR and TORPEDO-TR are $\mathcal{NP}$ -complete.*

It is not difficult to prove that TORPEDO+TR (resp. TORPEDO-TR) can be reduced from the well-known triangle packing problem[3] (resp. hamiltonian

[3] In a graph $G = (V, E)$, a triangle packing is a collection $V_1, \dots, V_k$ of disjoint subsets of V, each containing exactly three vertices linked by three edges which belong to E (see [4]).

path problem[4]). Due to lack of space, the proof of the Theorem 11 is not described here.

The approximation of these problems requires that vertices in G_c be covered, but from [10] or [6] 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 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 21 (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 22 (M-covered vertex) *A 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 $S(M)$ (resp. $NS(M)$).*

Definition 23 (Maximum 2-cover) *In a maximum 2-cover the number of covered vertices is maximum, therefore the number of non-covered vertices is minimum.*

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

Definition 24 (M-alternated path) *Let M be a 2-cover in a graph $G = (V, E)$, an M-alternated path $C = x_0, x_1, \ldots, x_k$ is a path in G such that for $i = 0, \ldots, \lfloor \frac{k}{2} \rfloor - 1$, $x_0 \in NS(M)$, $\{x_{2i}, x_{2i+1}\} \notin M$, and $\{x_{2i+1}, x_{2i+2}\} \in M$*

Definition 25 (Vertebral column of an M-alternated path) *Let M be a 2-cover in a graph $G = (V, E)$, and $C = x_0, x_1, \ldots, x_k$ an M-alternated path in G. The vertebral column denoted T associated to the path C is composed of C and M edges which are incident to C (and possibly their extremity).*

Remark 21 *The only case when T contains a cycle is when the last vertex 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))*

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

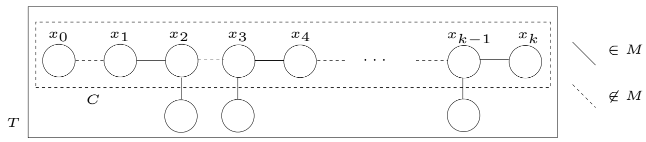
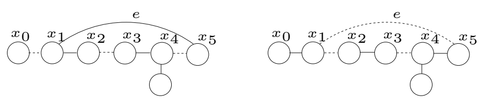


Fig. 3. Example of a vertebral column T associated to an M-alternated path C



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

(b) Example of improvement with a cycle in T

Fig. 4. Illustrations for the proof of the Lemma 51

Definition 26 (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.*

Definition 27 (Improvement of an M-alternated path) *Let $C = x_0, \ldots, x_k$ be an M-alternated path, and $x_0 \in NS(M)$. C becomes improving if we can reduce by at least one the number of non-covered vertices in C by changing the belonging to M of the edges of C.*

Remark 22 *From Remark 21, a path of length three or four can be created thanks to the improvement operation used in Definition 27. Let e be the edge of M which creates the cycle in T and thus creates the path of length three or four. Then, the edge e can be removed from M in order to improve the 2-cover (see Figure 4(b)).*

2.1 Results

This section is devoted to a lemma about the improvement of alternated paths and the fundamental Theorem of the 2-cover with M-alternated path.

Lemma 21 *Let M be a 2-cover, $C = x_0, x_1, \ldots, x_k$ an M-alternated path with $x_0 \in NS(M)$, and T its M-alternated column associated. C is improving if and only if there exists a vertex x_{2i-1} such that $d_M(x_{2i-1}) \neq 2$ or if T contains a cycle.*

Proof

$\Rightarrow$ Suppose that C may be improved, we will show that there exists a vertex x_{2i-1} such that $d_M(x_{2i-1}) \neq 2$. Suppose that an odd vertex x_{2i-1} such that

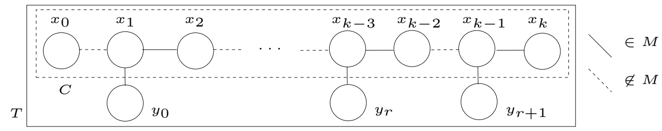


Fig. 5. Skeleton of the column T associated to C

$d_M(x_{2i-1}) \neq 2$ does not exist or that T does not contain any cycle. Thus, C and its column T have the shape of Figure 5.

From Definition 27, if T does not contain any cycle, we can simply improve the cardinality of the path 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. In this way, we change the belonging to M of the edge $\{x_1, x_2\}$, which means that we must change $\{x_2, x_3\}$. Recursively, we will change the belonging to M of all C edges. Thus, the last vertex x_k will not be covered, and our M-alternated path will not be improving. This is inconsistent with the former assumptions. Therefore, either there exists a vertex x_{2i-1} such that $d_M(x_{2_i-1}) \neq 2$, or T contains a cycle .

$\Leftarrow$ On the contrary, we study two possibilities. First, let us assume that T does not contain a cycle. Suppose that there exists a vertex x_{2i-1} such that $d_M(x_{2i-1}) \neq 2$. We will show that C becomes improving. Let $x_j = x_{2i-1}$ be the first vertex on the M-alternated path with a degree inferior to 2. We have three cases:

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

Now, let us assume that T contains a cycle:

Suppose that path C is without a odd vertex x_{2i-1} such as $d_M(x_{2i-1}) \neq 2$ except the last one which is connected to an odd vertex of C by an edge $e \in T$.

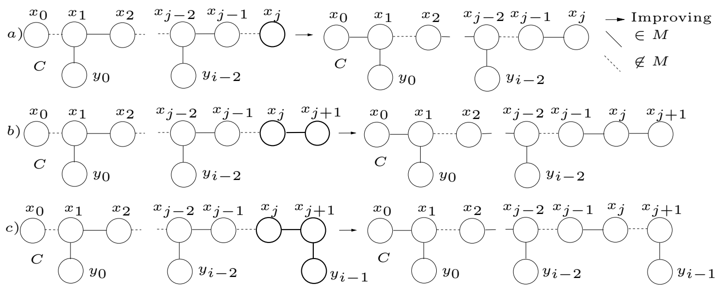


Fig. 6. Improvement of the three cases with $j = 2i - 1$

Thus, T contains a cycle, it is easy to see that path C becomes improving if we change the belonging to M of all C edges and e (see illustration in Figure 7). $\square$

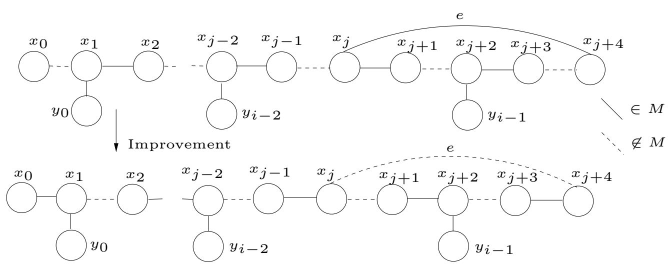


Fig. 7. Case of an M-alternated path with cycle

Theorem 21 *Let M be a 2-cover in a graph G, M admits maximum cardinality if and only if G does not possess an improved M-alternated path.*

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

Definition 28 (Leaf and root) *Let M be a 2-cover in a graph G, and let C be a non-improving M-alternated 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 $j = 2i$ is a leaf, moreover, all the vertices of the vertebral column associated are also leaves. On the contrary, a vertex $x_j \in C$ with $j = 2i + 1$ is a root.*

Proof of the Theorem 21

This proof is drawn from the classical proof given in [2].

⇒ Let M be a maximum 2-cover in G and suppose that G contains an improved M-alternated path. It leads to a contradiction by Lemma 21 because M would not be maximum.

⇐ Let M_1 be a 2-cover in G. Suppose that G does not contain an improved M_1-alternated path. We will show that M_1 is maximum.

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 8):

- 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-improving M_1-alternated path. Let S_2 be the set of vertices covered by M_1, which belongs to these non-improving 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 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.
- At last, there exist vertices not covered by M_1 nor by M_2, this set is denoted by $S_4 = \{x_i | x_i \in NS(M_1) \cap NS(M_2)\}$.

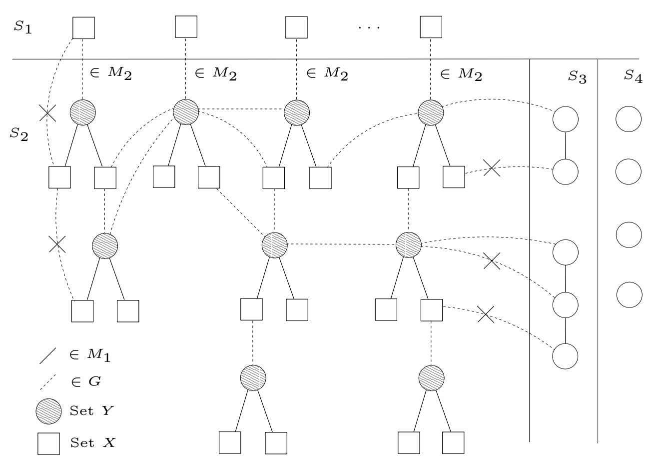


Fig. 8. Diagram of the proof of Theorem 21

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

- S_1 is necessarily a stable, otherwise there would be two vertices non-covered by M_1 connected by an edge. Then, we would have an improving M_1-alternated path.

- In set S_2, a root may be connected by an edge of G to any vertices of S_1 and S_2. Two leaves in S_2 cannot be connected by an edge. Moreover, a leaf in S_2 does not possess a neighbor in S_1. As a matter of fact, in both cases we would have either a cycle or an improving path (see illustration in figure 8).
- Every root of S_2 may be connected by an edge of G to any vertices of S_3. But, a leaf of S_2 cannot be connected by an edge of G to a vertex of S_3. As a matter of fact, if a leaf of S_2 were connected to an extremity of an edge or path of length two, then there would be an improving M_1-alternated path. Moreover, if a leaf in S_2 were connected to the center of a path of length two, this path would belong to S_2. Finally, leaves of S_2 are only connected to roots of S_2.
- 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 leaves of S_2 and of all vertices of S_1, and its cardinality is $|X| = (2N + K)$. Furthermore, the set X is a stable in regard to previous properties.

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

- Assume that M_2 covers all vertices of S_2, then M_2 covers all vertices of X and Y. In the case when a maximum number of vertices of X are covered, an edge of M_2 covers one vertex of X and one of Y, and a path of length two of M_2 covers at most two vertices of X and at least one of Y. So, M_2 cannot cover all vertices of S_2.
- Due to the fact that M_2 covers K vertices non-covered by M_1 in S_1, M_2 does not cover at least $(K–1)$ leaves in S_2. So, M_2 will cover $|X'| = |X| - (K–1) = 2N+1$ vertices. But in the case when a maximum number the vertices of X are covered, a path of length two of M_2 covers at most two vertices of X and at least one of Y. Thus, M_2 cannot cover more vertices than M_1. As a result, there does not exist a 2-cover M_2 such that $|M_2| \geq |M_1|$. So, $|M_2| = |M_1|$ and M_1 is a maximum 2-cover.

$\square$

2.2 Polynomial-time algorithm for maximum 2-cover

From Theorem 21, we can now introduce the algorithm which gives a maximum 2-cover. Let M be a 2-cover, and let C be an improved M-alternated path. The algorithm substitutes covered edges for non-covered edges in path C, except one of the edges at the end according to different cases. We denote this operation $Improving(M, C)$, 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 presented in appendix 5.

The algorithm which searches an improving 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 described in appendix 5.

The breadth first search has a complexity $O(n+m)$ with n (resp. m) the number of vertices (resp. edges), in the worst case we search n times an improving path. The Algorithm is performed in $O(n^2)$.

3 Study of approximation for TORPEDO+TR and TORPEDO-TR

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

3.1 First case: TORPEDO-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 11, we will compute the number of idle slots after executing all the n coupled-tasks and treatment tasks. In this way, let $Nb(Ci)$ be the number of a path C_i ($Nb(C_0)$ for non-covered vertices) in an optimal covering. In the following, $n = Nb(C_0) + n_2$ where n_2 is the number of covered vertices in an optimal solution (see Figure 9). 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) 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 $= Nb(C_0)+Nb(C_1)+(2\Sigma_{i=2}^{n_2-1}Nb(C_i)-1)-(Nb(C_0)-1)-2Nb(C_1)-\Sigma_{i=2}^{n_2-1}(i+1)Nb(C_i) = Nb(C_0)-Nb(C_1)-\Sigma_{i=2}^{n_2-1}(i-1)Nb(C_i)$

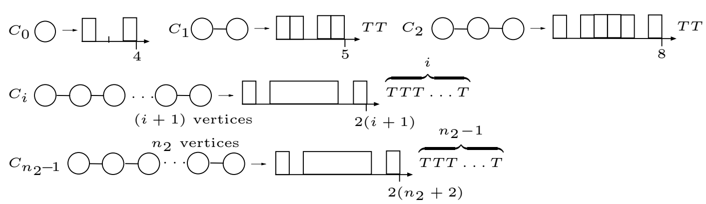


Fig. 9. Illustration of different paths of the covering of G_c

According to f, the lower bound[5] will be equal to:

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

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

Lemma 31 *It exist an optimal solution to TORPEDO-TR that minimizes* $Nb(C_0)$.

Proof

It is obvious that minimizing f provides an optimal solution to TORPEDO-TR. Let us show that if $Nb(C_0)$ is minimized in f, the value $Nb(C_1)+\sum_{i=2}^{n}(i-1)Nb(C_i)$ will be increased. Let's suppose that we have $Nb(C_0)$ non-covered vertices in a non-maximum covering M[6]. 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-TR. □

Lemma 32 *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 32 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 (Due to lack of place, the proof is not described here). In the optimal solution, 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 $Nb^h(C_1)=Nb(C_1)+\sum_{i(odd)=3}^{n_2-1}[(\frac{i+1}{2})Nb(C_i)]+\sum_{i(even)=2}^{n_2-1}[(\frac{i-2}{2})Nb(C_i)]=Nb(C_1)+\sum_{i=2}^{n_2-1}[(\frac{i+1}{2})Nb(C_i)]-\sum_{i(even)=2}^{n_2-1}\frac{3}{2}Nb(C_i)$, 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 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 the worst case is

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

obtained when $Nb(C_0) = Nb(C_1) + \sum_{i=2}^{n_2-1}[(i-1)Nb(C_i)]$ (see Figure 10). With the substitutions of $Nb(C_0)$ and n in C_{max}^{opt} and C_{max}^{h}, we obtain:

$$\rho \leq \frac{C_{max}^{h}}{C_{max}^{opt}} =$$

$$\frac{1+3Nb(C_0)+6Nb(C_1)+3\sum_{i=2}^{n_2-1}(i+1)Nb(C_i)+n_1-Nb^h(C_1)-Nb^h(C_2)}{1+3Nb(C_0)+6Nb(C_1)+3\sum_{i=2}^{n_2-1}(i+1)Nb(C_i)} =$$

$$\frac{1+9Nb(C_1)+3\sum_{i=2}^{n_2-1}\left[[Nb(C_i)(2i)]+[Nb(C_i)(\frac{i-3}{2})]\right]+\frac{1}{2}\sum_{i(even)=2}^{n_2-1}Nb(C_i)}{1+9Nb(C_1)+3\sum_{i=2}^{n_2-1}[(2i)Nb(C_i)]}$$

$$= 1 + \frac{\frac{1}{2}\sum_{i=2}^{n_2-1}[Nb(C_i)(i-3)]+\frac{1}{2}\sum_{i(even)=2}^{n_2-1}Nb(C_i)}{1+9Nb(C_1)+6\sum_{i=2}^{n_2-1}[iNb(C_i)]} \leq \frac{13}{12}$$

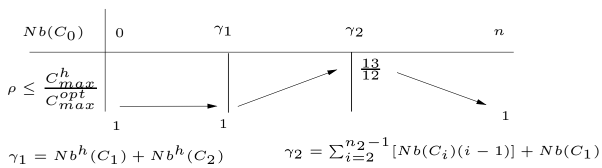


Fig. 10. Variation of ρ depending on $Nb(C_0)$

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-TR.

Algorithm 1 Use of 2-cover for TORPEDO-TR

1: **Data**: $G_c = (V, E)$, with $|V| = n$
2: **Result**: C_{max}^{h}, schedule length with this heuristic
3: **Begin**
4: $M_1 :=$ maximum matching in G_c
5: $M_2 :=$ maximum 2-cover from M_1
6: $M_3 := Transformation(M_2)$
7: Schedule vertices covered by edges in M_3, then by paths in M_3
8: Schedule isolated vertices, then schedule treatment tasks at first idle time

Remark 31 *$Transformation(M_2)$ 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 in 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 11).*

3.2 Second case: TORPEDO+TR problem

This study is almost the same as the previous one. In this case we must simply add triangles to the set of all the paths in the optimal solution, in order to

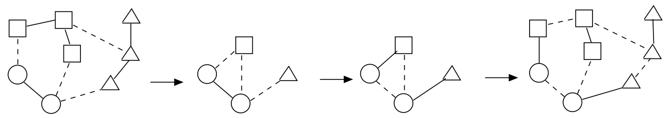


Fig. 11. Illustration of the transformation from paths of length two into edges

obtain an optimal covering of G_c. We denote $Nb(TR)$ the number of triangles in the optimal covering and 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 $C_{max}^{opt} \geq 3n+\mathbb{1}_{\{Nb(TR)>0\}}+max\{0, Nb(C_0)-Nb(C_1)-\sum_{i=2}^{n_2-1}[Nb(C_i)(i-1)] - 3Nb(TR)\}$.

In this case, minimizing $Nb(C_0)$ does not imply the minimizing 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 we need the same $Nb(C_0)$ for the calculus, in this way we can say without loss of generality that the worst case is when the two $Nb(C_0)$ are the same.

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 is 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 $C_{max}^h \leq 3n + 1 + max\{0, Nb(C_0) - Nb^h(C_1) - Nb^h(C_2)\}$.

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

$$\rho \leq \frac{1+3Nb(C_0)+6Nb(C_1)+3\sum_{i=2}^{n_2-1}(i+1)Nb(C_i)+n_1-Nb^h(C_1)-Nb^h(C_2)}{\mathbb{1}_{\{Nb(TR)>0\}}+3Nb(C_0)+6Nb(C_1)+3\sum_{i=2}^{n_2-1}(i+1)Nb(C_i)+9Nb(TR)}$$

$$\leq 1+\frac{2Nb(TR)+\frac{1}{2}\sum_{i=2}^{n_2-1} i[Nb(C_i)]-\frac{1}{2}\sum_{i(even)=2}^{n_2-1} Nb(C_i)}{1+9Nb(C_1)+18Nb(TR)+6\sum_{i=2}^{n_2-1}[iNb(C_i)]} \leq \frac{10}{9}$$

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 is maximum when there are no improving paths according to definition of 2-cover. 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. 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. C. Berge. Two Theorems in Graph Theory. *Proceedings of the National Academy of Science*, 43:842–844, September 1957.
3. 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.
4. M.R. Garey and D.S. Johnson. *Computers and Intractability; A Guide to the Theory of $\mathcal{NP}$-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
5. 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.
6. S. Masuyama and T. Ibaraki. Chain packing in graphs. *Algorithmica*, 6:826–839, juin 1991.
7. A.J. Orman and C.N. Potts. On the complexity of coupled-task scheduling. *Discrete Applied Mathematics*, 72:141–154, 1997.
8. R.D. Shapiro. Scheduling coupled-tasks. *Naval Research Logistics Quarterly*, 27:477–481, 1980.
9. G. Simonin. Proof of $\mathcal{NP}$-completeness for a scheduling problem with coupled-tasks and compatibility graph. *Technical rapport - Lirmm (Laboratoire Informatique, Robotique et Micro-électronique de Montpellier)*, 2008.
10. G. Steiner. On the k-path partition of graphs. *Theoretical Computer Science*, 290:2147–2155, 2003.

5 Appendixes

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

We will show that the TORPEDO+TR problem is $\mathcal{NP}$-complete when G_c contains triangles. In this way, we will use the triangle packing (TP) problem[7].

Proof

The construction of the polynomial transformation is given for the reduction $TP \propto$ TORPEDO+TR. From an instance Π of TP, we built an instance Π' of TORPEDO+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 **??**).

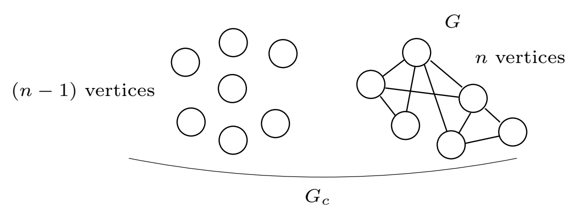


Fig. 12. Illustration of the polynomial time transformation

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 $3(2n-1)$.

$\Leftarrow$ 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.

[7] 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.

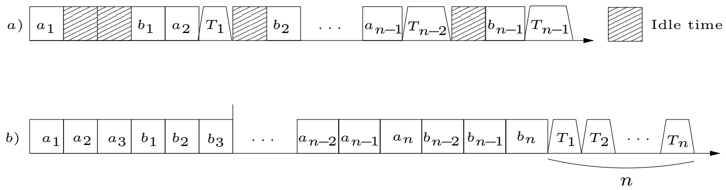


Fig. 13. Illustration of the covering of G

□

We will show that the TORPEDO-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 transformation is given for the reduction $HC \propto$ TORPEDO-TR. From an instance Π of HC, we built an instance Π' of TORPEDO-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).

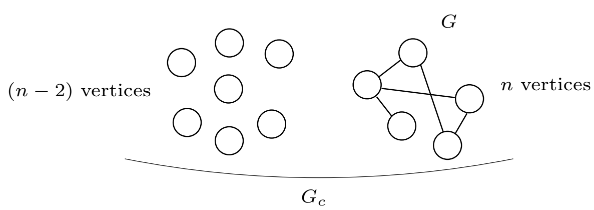


Fig. 14. Illustration of the polynomial transformation

$\Rightarrow$ 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 remain $(n-2)$ 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 $3(2n-2)+1$.

$\Leftarrow$ 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

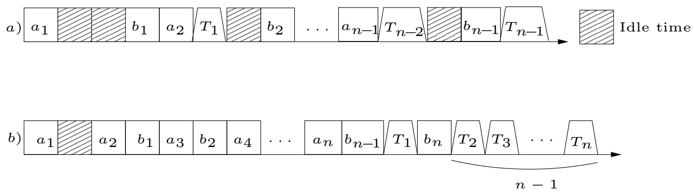


Fig. 15. Illustration of the covering of G

gives $(n-1)$ idle slots (see illustration Figure 15a). 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 15b). Thus, the covering of the vertices of G is necessarily a Hamiltonian path. $\square$

5.2 Fundamental Lemma in order to find a maximum 2-cover

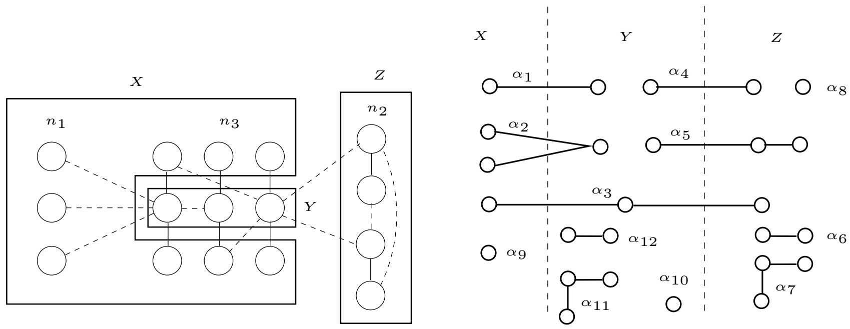
Lemma 51 *A maximum 2-cover consists in firstly minimizing $Nb(C_0)$, then secondly maximizing $Nb(C_1)$.*

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 is not improving implies that there cannot exist edges between X and Z.

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 covers possible 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 51 *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 $\beta_{3,T,U,V} = \beta_{3,V,U,T}$.*



(a) Illustration of the covering of τ_1

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

Fig. 16. Illustrations for the proof of the Lemma 51

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:

$$\alpha_9 = n_1 + \frac{2n_3}{3} - \alpha_1 - 2\alpha_2 - \alpha_3 \tag{1}$$

$$\alpha_{10} = \frac{n_3}{3} - \alpha_1 - \alpha_2 - \alpha_3 - \alpha_4 - \alpha_5 - 3\alpha_{11} - 2\alpha_{12} \tag{2}$$

$$\alpha_8 = n_2 - \alpha_3 - \alpha_4 - 2\alpha_5 - 2\alpha_6 - 3\alpha_7 \tag{3}$$

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:

$f_{\tau_2} =$ 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:

$$n_1 + \alpha_4 + \alpha_5 + \alpha_8 - \alpha_2 - \alpha_6 - \alpha_7 < n_1 - \frac{n_2}{2} - \frac{n_3}{3}$$

$$\alpha_1 + \frac{3\alpha_3}{2} + \frac{5\alpha_4}{2} + 3\alpha_5 + \frac{\alpha_7}{2} + \frac{3\alpha_8}{2} + 4\alpha_{10} + 8\alpha_{11} + 5\alpha_{12} < 0$$

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.

$\square$

5.3 Maximum 2-cover algorithms

The algorithm which creates a maximum 2-cover is as follows:

Algorithm 2 Research of a maximum 2-cover

DATA: $G = (V, E)$
RESULT: A 2-cover M
Begin:
$M := \emptyset$
while there exists an improved M-alternated path **do**
  $M := Improving(M, C)$
end while
Return M

And the following algorithm describes the researching of an improved M-alternated path from a non-covered vertex x_0:

Algorithm 3 Research of an improved M-alternated path

DATA: $G = (V, E)$, with $|V| = n$, a non-covered vertex x_0, and M a 2-cover
RESULT: An improved M-alternated path C from the vertex x_0
Begin:
Let Q (resp. Z) be a queue whose unique element is the vertex x_0
Let F be a function which gives the precedent vertex of another 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 $v \in Z$ **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 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
        end if
        Push v in Q
      end if
    end for
    Pull u of Q
  end if
end while