Graph decomposition into paths under length constraints

Nicolas Teypaz, Christophe Rapine

Graph decomposition into paths under length constraints

*

Nicolas Teypaz
Christophe Rapine
G-SCOP, Laboratoire des Sciences pour la Conception, l'Optimisation et la Production,
46, avenue Felix Viallet, 38031 Grenoble, France.
{nicolas.teypaz, christophe.rapine}@g-scop.inpg.fr

April 2, 2008

Abstract

Given 2 integers $a \leq b$, we define an (a,b)-decomposition of a graph $G = (V,E)$ as a partition of E into paths where the length of each path lies between a and b. In this definition paths are requested to be simple but not necessarily elementary. In other words an (a,b)-decomposition corresponds to a partition $E_1, \ldots, E_k$ of E with $a \leq |E_i| \leq b$ for all i, and such that each induced subgraph (V_i, E_i) admits an Eulerian path. In this paper we investigate the complexity of the (a,b)-decomposition problem. We give a characterization of $(2,b)$-decomposable graphs, from which it is possible to decide in linear time the $(2,b)$-decomposition problem. When $a \geq 3$, we identify 2 classes of graphs, trees and Eulerian graphs, for which the (a,b)-decomposition problem remains polynomial. However we prove that the $(3,3)$-decomposition problem is $\mathcal{NP}$-complete, even on bipartite graphs. We give some evidence to conjecture that the more general $\mathcal{H}$-decomposition problem into a given family $\mathcal{H} = \{H_1, \ldots, H_n\}$ of connected graphs is $\mathcal{NP}$-complete if and only if all H_i-decomposition problems are $\mathcal{NP}$-complete.

1 Introduction

We define an (a,b)-decomposition of a graph $G = (V,E)$ as a partition of E into simple paths such that the length of each path is at least a and at most b, with $a \leq b$. Given 2 integers a and b, the (a,b)-decomposition problem is to decide whether an input graph G admits an (a,b)-decomposition. We denote by $\mathcal{L}_{a,b}$ the corresponding language on graphs, i.e. the subset of graphs that are (a,b)-decomposable. In this paper we will consider mainly the case of unweigthed graphs. Recall that the length of a path is then defined by its number of edges. Given a and b, we can then define the finite set $\mathcal{F}_{a,b}$ of connected traversable graphs, i.e. admitting an Eulerian walk, with $a \leq m \leq b$ edges. An (a,b)-decomposition problem can be seen equivalently as a partition of G into subgraphs, where each subgraph is isomorphic to a graph of $\mathcal{F}_{a,b}$. As an example we give the family $\mathcal{F}$ of graphs for $a = b = 3$ and $a = b = 4$ in figures 1 and 2 on simple graphs. The family for the $(3,4)$-decomposition problem on simple graphs is then the union of the previous two sets.

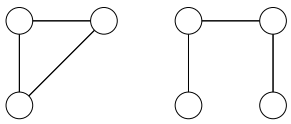


Figure 1: The family $\mathcal{F}_{3,3}$ of graphs.

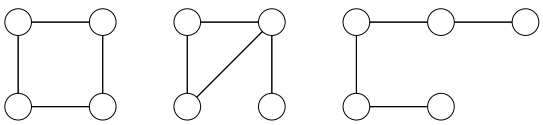


Figure 2: The family $\mathcal{F}_{4,4}$ of graphs.

The study of this graph decomposition problem has been motivated by a transportation application, we encounter during an industrial collaboration. A firm was developping activities

*This research has been supported by the following grant: BQR INPG "Optimisation du transport de fret par l'utilisation de plateformes logistiques"

in freight transportation. Typically it receives orders of transportation from different clients worldwide, each order consisting in carrying some goods from a specified loading location to some destination. A vehicle can stop at intermediate locations on its way to load/unload orders of other clients. In our case the transportation application contains two main steps :

- The first step consists in determining the transportation network in order to carry the different goods. A transportation network is a set of transportation links between locations. The problem is close to multi-commodity network design problem, see [GCF99, BJV00, CGG03]. It is to notice that the network requested in our problem verifies Eulerian cycle properties.
- The second step consists in designing the vehicle routes, drawn from the transportation designed network. We assume that the fleet size is sufficiently large, thus the number of routes is not constrained. A travel time is associated to each links of the network. The route length corresponds to the total transportation time (discounting waiting times and loading operations). On one hand, due to a limitation on vehicle autonomy, the travel time along a route must not exceed a given value (upper bound). On the other hand, in order to make regular maintenance checks, the vehicles must travel a minimum amount of time (lower bound) before returning to their depot. These constraints imply that the length of each vehicle route is included between two bounds. The solution of this problem gives the set of routes to be used, and the size of fleet needed.

This last problem can be seen as a decomposition into paths of a directed graph, where the length of each path must be inside a specified interval. This example incitated us to define the (a, b)-decomposition problem and to study its complexity. Notice that in our industrial application, the limitation on the length of each vehicle route is due to hard constraints. Nevertheless it corresponds also to a general smouth goal of the company to use in an equitable way its fleet of vehicles.

In literature, some problems of graph decomposition have been investigated. In [Hol81], Holyer showed that the decomposition into subgraphs isomorphic to K_n is $\mathcal{NP}$-complete for any integer $n \geq 3$. He conjectured that the problem of decomposition of a graph into subgraphs isomorphic to a fixed graph H is $\mathcal{NP}$-complete for all graphs H with at least 3 edges. In [DT97], Dor and Tarsi presented the proof of this conjecture restricted to the case where H is connected with at least 3 edges. Bialostocki and Roditty [BR83] and Alon [Alo83] give a polynomial time algorithm for the case $H = 3K_2$ (a matching of 3 edges). In [FLT85], Favaron *et al.* have obtained a similar result for the case $H = K_2 \cup P_3$. In [CT91], Cohen and Tarsi have proved two theorems. Firstly, the S_n-decomposition (or $K_{1,n}$-decomposition) is $\mathcal{NP}$-complete for any integer $n \geq 3$. Secondly, given graphs $H_1, \ldots, H_n$, if the H_1-decomposition is $\mathcal{NP}$-complete then the H-decomposition is also $\mathcal{NP}$-complete, where $H = \cup_i H_i$. Recently, in [PT05, PT04], Priesler and Tarsi have investigated the computational complexity of the H-decomposition of a multigraph.

All these results do not allow us to draw some conclusions on the $\mathcal{NP}$-completeness of the (a, b)-decomposition problem. Indeed if both decomposition into K_3 and decomposition into P_4 are $\mathcal{NP}$-complete, the complexity of the decomposition into K_3 and P_4, *i.e.* graphs of family $\mathcal{F}_{3,3}$, is unknown. The problem of decomposing G into isomorphic copies of graphs in a given family $\mathcal{H} = \{H_1, \ldots, H_k\}$ does not seem to reduce in a straightforward way into the H_i-decomposition of G. In section 3 we conjecture that this decompostion problem is hard for a family of connected graphs if and only if all the H_i-decomposition problems are hard.

In this paper we study the computational complexity of language $\mathcal{L}_{a,b}$ depending on the values of a and b, and its restriction to special subclasses of graphs. For polynomial cases, we provide constructive algorithms to build an (a, b)-decomposition. In section 2 we give a characterization of $(2, b)$-decomposable graphs. We conjecture in section 3 that the more general decomposition problem into a given family $\{H_1, \ldots, H_k\}$ of graphs is $\mathcal{NP}$-complete if all the H_i-decomposition problems are $\mathcal{NP}$-complete. In section 4 we investigate the complexity of the decomposition of bipartite graphs. We establish the $\mathcal{L}_{a,b}$ is polynomial on trees for any integer $a \leq b$, while the $(3, 3)$-decomposition problem is $\mathcal{NP}$-complete on general bipartite graphs. In section 5 we investigate the case of traversable graphs. Finally section 6 extends some results to the class of weighted graphs.

2 Characterization of $(2, b)$-decomposable graphs

In this section, we investigate the complexity class of the (a, b)-decomposition of a graph when $a < 3$. We give a characterization of $\mathcal{L}_{2,b}$, and derive polynomial-time algorithm to decide if a graph admits a $(2, b)$-decomposition together with a constructive algorithm to provide such a decomposition. We restrict our attention to unweigthed multiple-graphs, *i.e.* we do not require graphs to be simple graphs but allow multiple edges.

First notice that decomposition problems with $a = 1$ admits a trivial solution, simply taking one path per edge. Hence $\mathcal{L}_{1,b}$ contains all graphs. Consider next the $(2, 2)$-decomposition problem : it consists in decomposing a graph into isomorphic copies of $K_{1,2}$. Lonc in [Lon97] proved that this problem is polynomial : its algorithm constructs the line-graph of G, and then looks for a maximum matching in it. As a matching in the line-graph defines edge-disjoint paths of length 2 in the original graph, a graph G is $(2, 2)$-decomposable if and only if its line-graph admits a perfect matching. Our results are inspired from [Lon97] analyzing the structure of the line-graph. We first give the following lemma on the maximum matching in a line-graph. It states that for G a connected graph, its line-graph admits a perfect matching if and only if G has an even number of edges. Although lemma 2.1 is equivalent to a result independently found by Sumner [Sum74] and Las Vergnas [Las75], we give here a simple and direct proof.

Lemma 2.1 *In a line graph, a maximum matching does not cover at most one vertex per connected component.*

Proof. Let M be a matching in a line graph, and assume that one connected component, say H, contains 2 unmatched vertices u and v. We prove that then there exists an augmenting path in H, which implies that M is not maximum.

To build this augmenting path, we start from a shortest path $p = (u = x_0, \ldots, x_k = v)$ between u and v. Without loss of generality we can assume that vertices $x_1, \ldots, x_{k-1}$ are matched, otherwise we choose for v the first unmatched vertex on the path. We now show by induction that for any vertex x_i on p, there exists an alternating path q_i from u to x_i. In the remaining of the proof we implicitely require that an alternating path is a simple path. For $i = 1$, (u, x_1) is certainly an alternating path. For $i \geq 1$, assume that an alternating path $q_i = (u = z_0, z_1, \ldots, z_{l+1} = x_i)$ exists, and let us exhibit a path q_{i+1}. The only non-trivial case to be considered is when neither edge (z_l, x_i) nor edge (x_i, x_{i+1}) belongs to M. We claim that it exists then an alternating path r from z_l to x_{i+1} whose first edge is not in M. To see this consider y the vertex such that $(x_i, y) \in M$, see figure 3

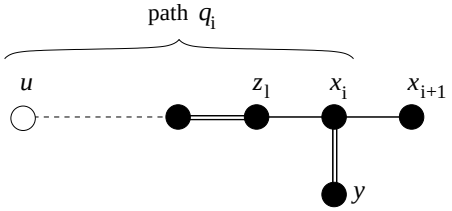


Figure 3: The alternating path q_i. Egdes of M are drawn with double lines.

In [Bei70], Beineke showed that a line-graph does not contain $K_{1,3}$ as an induced subgraph. As a consequence one of the edges (z_l, x_{i+1}), (z_l, y) or (y, x_{i+1}) is present in H. In each case we can easily find a desired alternating path r, see figure 4.

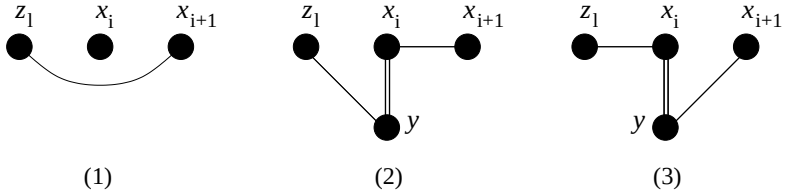


Figure 4: The alternating path r from z_l to x_{i+1}, in case (1) $(z_l, x_{i+1}) \in E$, (2) $(z_l, y) \in E$, (3) $(y, x_{i+1}) \in E$

Of course it is tempting to append path r to the begining $(u, z_1, \ldots, z_l)$ of path q_i to exhibit the alternating path q_{i+1}. If the resulting path is simple, we have effectively found an alternating path. Otherwise it means that either y or x_{i+1} are visited by q_i. For the sake of contradiction assume that node y is visited in q_i. As q_i is alternated, node x_i must be the predecessor or the successor of y in the path (as $y \neq u$ and $y \neq z_l$). This contradicts the fact that q_i is an alternating (simple) path. Hence the only remaining case corresponds to the situation where node x_{i+1} is on path q_i. By construction path q_i truncated at node x_{i+1} is then an alternating path from u to x_{i+1}.

The induction proves the existence of an alternating path from u to v, which is by definition an augmenting path. □

As a direct consequence we can state the following corollary:

Corollary 2.1 (Characterization of $\mathcal{L}_{2,2}$) *A multiple-graph G is $(2,2)$-decomposable iff all its connected components have an even number of edges. As a consequence we can decide in linear time $\mathcal{O}(|E|)$ if a graph $G \in \mathcal{L}_{2,2}$.*

Proof. Clearly having an even number of edges in each connected component is a necessary condition. To prove this is a sufficient condition, we use the remark in Lonc [Lon97] that a matching in the line-graph defines edge-disjoint paths of length 2 in the original graph. Thus a connected graph G is $(2,2)$-decomposable if its line-graph admits a perfect matching. Due to Lemma 2.1 a line-graph admits a perfect matching if and only if the number of edges in the original graph is even. Thus to decide if G is $(2,2)$-decomposable we only need to determine its connected components, which can be done in linear time using Tarjan's algorithm [Tar72]. □

We now turn attention to the $(2,b)$-decomposition problem, with $b \geq 3$. Next lemma permits to concentrate our efforts on the search of a $(2,3)$-decomposition using at most one path of length 3:

Lemma 2.2 *A connected graph G admits a $(2,b)$-decomposition for $b \geq 3$ iff it admits a $(2,3)$-decomposition with at most one path of length 3.*

Proof. For short we denote by $\mathcal{L}^+_{2,3}$ the language of connected graphs that can be decomposed into paths of length 2 and at most one path of length 3. Lemma 2.2 can be rephrased as $\cup_{b\geq 3}\mathcal{L}_{2,b} = \mathcal{L}^+_{2,3}$.

Let $b \geq 3$ be an integer. Trivially we have $\mathcal{L}^+_{2,3} \subseteq \mathcal{L}_{2,b}$. For the reverse inclusion, consider a connected graph G belonging to $\mathcal{L}_{2,b}$. If the number of edges in G is even, corollary 2.1 implies that G can be decomposed into paths of length 2, *i.e.* belongs to $\mathcal{L}_{2,2} \subseteq \mathcal{L}^+_{2,3}$. Thus we can assume that the number of edges in G is odd. Now consider one decomposition of G into paths of length in $[2,b]$. Clearly a path of length greater than 3 can be cut into paths of length 2 and 3. Hence without loss of generality we can restrict our attention to a $(2,3)$-decomposition of G. As the number of edges is odd, this decomposition has an odd number q of paths of length 3.

For a path p of the decomposition, we define $G(p)$ as the edge-induced subgraph obtained by removing the edges of p. For each connected component H of $G(p)$, we can count the number of paths of length 3 in the decomposition that belongs to it. Let $q(H)$ be this number. For simplicity we say that a component is *even* if its number q is even, and *odd* otherwise. Due to corollary 2.1, an even component admits a $(2,2)$-decomposition. Hence we only need to prove that there exists a path p of length 3 in the decomposition such that all the connected components of $G(p)$ are even.

For the sake of contradiction, assume that $G(p)$ contains at least one odd component whatever our choice of p. We can then define $\underline{q}(p)$ as the minimum of $q(H)$ on all the odd connected components of $G(p)$. Let p^* be a path of length 3 minimizing the quantity $\underline{q}(p)$, and $H_1, \ldots, H_k$ be the connected components of $G(p^*)$. Without loss of generality we can assume that $\underline{q}(p^*) = q(H_1)$. Then we choose arbitrarily in this component one path p' of the decomposition, and we consider the induced sub-graph $G(p')$ with connected components $H'_1, \ldots, H'_l$. The important fact to notice is that $H_2, \ldots, H_k$ and p^* belongs to the same component, say H'_1, while the other components H'_i are included in H_1. As a consequence, since p' is in H_1 but not in H'_i, we have $q(H'_i) < q(H_1) = \underline{q}(p^*)$ for all $i \geq 2$. It involves that all H'_i are even for $i \neq 1$, since by definition of p^* any odd connected component in any subgraph $G(p)$ contains at least $\underline{q}(p^*)$ paths of length 3. Let $m = q(H_1 \cap H'_1)$ be the number of paths in H_1 that belongs to H'_1. The paths of the decomposition in H_1 are exactly those of H'_2, ..., H'_k, plus p', plus those from $H_1 \cap H'_1$. Thus

we have $q(H_1) = \sum_{i\geq 2} q(H'_i) + 1 + m$. It results that m is even as $q(H_1) = \underline{q}(p^*)$ is odd. In the same way the paths of H'_1 are exactly those of $H_2, \ldots, H_k$, plus p^*, plus those from H_1. Hence $q(H'_1) = \sum_{i\geq 2} q(H_i) + 1 + m = q(G) - q(H_1) + m$. As the decomposition of G contains an odd number of paths of length 3, it implies that H'_1 is even. Hence all connected components $H'_1, H'_2, \ldots, H'_l$ in $G(p')$ are even. This contradicts our assumption that there is no induced subgraph with only even connected components. □

We have as an immediate corollary of lemma 2.2 and corollary 2.1:

Corollary 2.2 *The $(2, b)$-decomposition problem is polynomial for all b.*

Proof. This corollary follows directly from proof of lemma 2.2. Without loss of generality assume that G is connected. We can check in polynomial time if a graph is $(2, 2)$-decomposable. If not we have to decide if graph G belongs to $\mathcal{L}^+_{2,3}$. One naive algorithm can enumerate all possible paths of length 3 in the graph. There are at most $\mathcal{O}(n^4)$ possibilities. From proof of lemma 2.2 the graph is decomposable iff it exists a path p such that the induced subgraph $G(p)$ belongs to $\mathcal{L}_{2,2}$, which can be checked in polynomial time. □

In the following we are interested in a characterization of $(2, 3)$-decomposable graphs. This characterization will lead to a linear time algorithm to decide $\mathcal{L}^+_{2,3}$. Since corollary 2.1 provides yet a characterization of $\mathcal{L}_{2,2}$, we restrict our attention here to connected graphs with an odd number of edges.

First notice that a path a length 3 can be seen as the concatenation of a path of length 2 and one edge. Let (e, e') be 2 incident edges of G. We define graph $G_{<e,e'>}$ as graph G where the path of lenght 2 constituted of egdes e and e' is contracted into one new edge (see figure 5).

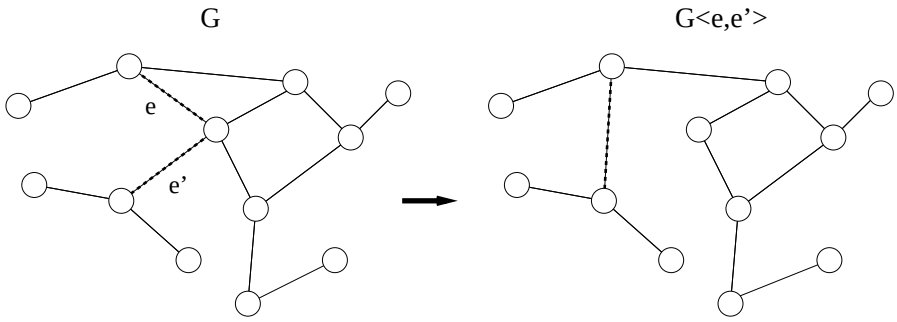


Figure 5: Construction of graph $G_{<e,e'>}$.

Due to lemma 2.2, graph G is $(2, 3)$-decomposable iff it exists two incident edges e and e' such that all the connected components of graph $G_{<e,e'>}$ are even. We can then notice the following fact:

Remark 2.1 *If a connected graph G contains a cycle, then G is $(2, 3)$-decomposable*

Proof. Consider a cycle C in the graph. If C contains all the edges of E, clearly we can decompose C into paths of length 2 and 3. Hence we can assume that there exists an edge (y, z) in the co-cycle of C, with y in the cycle. Let x be one of the neighbours of y in C. Then consider graph G' obtained by contracting edges (x, y) and (y, z) into one new edge (x, z). We claim that G' remains connected. Indeed by definition of a cycle, there still exists in G' a path from y to x, and thus to z. As the number of edges in G' has decrease by one compare to G, we get an even component. Thus G' is $(2, 2)$-decomposable. □

As a consequence, trees are the only class of graphs that may not be in $\mathcal{L}_{2,3}$. Clearly some trees are not decomposable, by instance $K_{1,3}$. We define an *odd-degree* tree to be a tree with all nodes of odd degree. We have the following result:

Theorem 2.1 (Characterization of $\mathcal{L}_{2,b}$) *A connected graph G is $(2, b)$-decomposable for any integer $b \geq 3$ iff G is not an odd-degree tree. As a consequence we can decide in linear time $\mathcal{O}(|E|)$ if a graph $G \in \mathcal{L}_{2,b}$.*

Proof. Necessary condition. Consider an odd-degree tree T. By immediate induction T has an odd number of edges. Thus we have to prove that the contraction of any 2 incident edges leads to a non $(2,2)$-decomposable graph. Let e, e' be two incident edges, defining a path (x,r,y). We then root tree T on r and denote by $T(u)$ the subtree rooted on a node u, *i.e.* the induced subgraph of nodes such that u is on their (unique) path to r. The contraction of (x,r,y) creates a new connected component T_1, constituted of $T(x)$ and $T(y)$ linked by the new edge (x,y) (see Figure 6). Clearly T_1 is odd-degree tree, since the degree of x and y has not changed in the contraction. It results that T_1 has an odd number of edges, and thus is not $(2,2)$-decomposable due to corrollary 2.1

Sufficient condition. Assume that G is not an odd-degree tree. We have to prove that G is $(2,3)$-decomposable. Due to remark 2.1, we only need to consider the case where G is a tree T with an odd number of edges. We arbitrarily root the tree on a node r of even degree. Since T has an odd number of edges, it necessarily exists an odd number of subtrees with an odd number of edges. And consequently, as the degree of r is even, an odd number of even subtrees. Let us pick one even subtree $T(x)$ and one odd subtree $T(y)$. We then contract the path (x,r,y) into the new edge (x,y). We obtained 2 connected components : T_1 containing $T(x)$ and $T(y)$, and T_2 containing the other subtrees rooted on r (see Figure 6). By construction both T_1 and T_2 have an even number of edges, which proves that T is in $(2,3)$-decomposable.

For a graph G we can then answer to the decision problem related to the $(2,b)$-decomposition in linear time, using Tarjan's algorithm to determine each connected component, and then counting the number of edges and eventually checking the degree of the nodes in each component. □

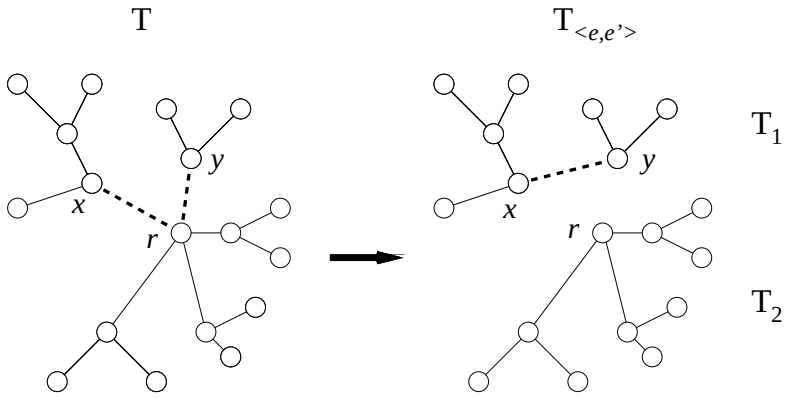


Figure 6: The tree T and the resulting graph $T' = T_1 \cup T_2$

Notice that if one wants to exhibit a decomposition on a positive instance, she needs to find the couple of edges to contract. If the graph is a tree, it is sufficient to chose two subtrees of different parity rooted on a node of even degree. Otherwise it is enough to find a cycle. In both case the computation can be done in linear time. The construction of the line graph of $G_{<e,e'>}$ can be done in time $\mathcal{O}\left(|E|^2\right)$ in the size of G. The overall time complexity of the algorithm is thus dominated by the determination of a maximal matching in the line graph. Using Edmonds' algorithm [Edm65] on the line-graph $L(G_{<e,e'>})$, we get a complexity in $\mathcal{O}\left(|E|^4\right)$.

3 The (a,b)-decomposition problem on general graphs

The $(3,3)$-decomposition problem on simple graphs consists in partitionning E into subgraphs isomorphic to K_3 and P_4 (see figure 1). The K_3-decomposition problem and the P_4-decomposition problem are known to be $\mathcal{NP}$-complete, but theses results do not enable us to conclude on our problem. However we conjecture that the (a,b)-decomposition problem on an unspecified graph is $\mathcal{NP}$-complete if a is greater than 3. We give here a more general conjecture for the decomposition problem into a family $\mathcal{H} = \{H_1, \ldots, H_k\}$ of graphs. Recall that Cohen and Tarsi [CT91] prove that if the H_i-decomposition is $\mathcal{NP}$-complete for some graph H_i, then the $\cup_i H_i$-decomposition is $\mathcal{NP}$-complete. Our conjecture is similar to this result, except that the existential quantifier is

replaced by an universal quantifier:

Conjecture 3.1 *Let $\mathcal{H} = \{H_1, \ldots, H_k\}$ be a given family of graphs. If the H_i-decomposition problem is $\mathcal{NP}$-complete for all $i = 1, \ldots, k$, then the $\mathcal{H}$-decomposition problem is $\mathcal{NP}$-complete.*

For instance, one can ask if decomposing a simple graph into isomorphic copies of connected graphs with 3 edges is a polynomial problem. This corresponds to the family $\mathcal{F}_3 = \{K_3, P_4, K_{1,3}\}$ represented figure 7. Notice that the decomposition problem into any of these graphs is known to be $\mathcal{NP}$-complete due to the result of Dor and Tarsi [DT97], but it gives no evidence to conclude on the complexity status of the $\mathcal{F}_3$-decomposition problem. At first sight almost any connected graph with a number of edges multiple of 3 can be decomposed into a combinaison of connected graphs of 3 edges. In fact we establish in next section that the $\mathcal{F}_3$-decomposition problem is $\mathcal{NP}$-complete.

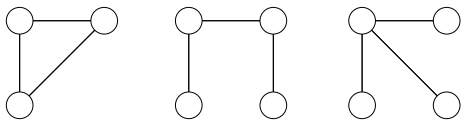


Figure 7: The family $\mathcal{F}_3$ of graphs.

In the conjecture we require all the H_i-decomposition problems to be $\mathcal{NP}$-complete. A good reason to impose this condition arises giving a look at family $\mathcal{F}_{2,3}$. It contains graphs P_4 and K_3, whose decomposition problem is known to be $\mathcal{NP}$-complete, but also graph P_3 whose decomposition problem is polynomial: corollary 2.2 shows that the $\{P_3, P_4, K_3\}$-decomposition problem is polynomial. In fact we can generalize this result to any family of graphs containing graph P_3:

Theorem 3.1 *Let $\mathcal{H} = \{H_1, \ldots, H_k\}$ be a given family of connected graphs. If P_3 belongs to $\mathcal{H}$, then the $\mathcal{H}$-decomposition problem is polynomial.*

Proof. Proof of lemma 2.2 can be easily adaptated to family $\mathcal{H}$. Notice that graphs H_i with an even number of edges can be dropped, since they are P_3-decomposable. Given a decomposition, we then count for the number of copies of the H_i's used in each connected component in the partial subgrap $G(h)$, where h now is a copy of one of the H_i. The connexity of the H_i's is here necessary to determine in which connected component a copie lies. In the same way we can prove that it exists a copy h such that the partial subgraph $G(h)$ is 2-decomposable. It results that if G is $\mathcal{H}$-decomposable, then it exists a decomposition using at most one copy of the H_i different from P_3. Since family $\mathcal{H}$ is given, i.e. of constant size, we can use the naive approach to check all possible partial graph $G(h)$. This algorithm is polynomial. □

Notice that if conjecture 3.1 is true, then the decomposition problem into a family $\mathcal{H}$ of connected graphs is polynomial if and only if $\mathcal{H}$ contains either P_2 or P_3. Indeed for any connected graph H different from P_2 and P_3 the H-decomposition problem is known to be $\mathcal{NP}$-complete. It results that conjecture 3.1 can be rephrased as an equivalence for a family of connected graphs: the $\mathcal{H}$-decomposition problem is $\mathcal{NP}$-complete if and only if the H_i-decomposition problems are $\mathcal{NP}$-complete for all i.

Looking back to the $(3,3)$-decomposition problem, one way to prove its $\mathcal{NP}$-completeness would be to establish that the P_4-decomposition problem remains $\mathcal{NP}$-complete on a class of graphs that do not admit K_3 as an induced subgraph. Indeed for these graphs a $(3,3)$-decomposition is in fact a P_4-decomposition. For these reason we study the status of the P_4-decomposition problem on bipartite graphs and trees in section 4.

4 Decomposition of Bipartite Graphs

In [HLY99, HJ04], some sufficient conditions for a graph to be decomposable into P_4 are given. In particular, a graph G being 3-regular and having a perfect matching can be decomposed into P_4. The matching is used for the middle edges of the paths in the decomposition. This argument allows us to deduce the following result:

Theorem 4.1 *The P_4-decomposition ($(3,3)$-decomposition) problem on complete bipartite graphs is polynomial.*

Proof. A necessary condition for a graph to be $(3,3)$-decomposable is to have a number of edges multiple of 3. This implies for bipartite graph $K_{m,n}$ that either 3 divides n or 3 divides m. We assume that $m = 3q$. As $K_{3q,1}$ is clearly not $(3,3)$-decomposable, we can restrict our attention to the case where $n > 1$. Consider now the parity of n:

- If $n = 2q'$ then we can identify easily qq' edge-distinct complete bipartite graphs $K_{3,2}$ in the graph $K_{3q,2q'}$. It is quite easy to see that $K_{3,2}$ can be decomposed into 2 paths of length 3. In fact $K_{3,2}$ is a transversable graph of 6 edges.
- If $n = 2q' + 1$ then we can identify easily $q(q' - 1)$ edge-distinct complete bipartite graphs $K_{3,2}$ and q graphs $K_{3,3}$ in the graph $K_{3q,2q'+1}$. Clearly $K_{3,3}$ is a 3-regular graph and admits a perfect matching. Thus it is P_4-decomposable [HLY99, HJ04]. Hence we can find 2 disjoint paths of length 3 in each $K_{3,2}$ and 3 disjoint paths of length 3 in each $K_{3,3}$.

In conclusion graph $K_{m,n}$ is $(3,3)$-decomposable iff 3 divides mn and $\min\{m,n\} > 1$. This test can clearly be done in time $\mathcal{O}(\log mn)$ if the partition of G into two independent sets is given. □

Trees are particular bipartite graphs. Notice that for this class of graphs the (n,n)-decomposition problem coincides with the P_{n+1}-decomposition problem for all integer n. We establish here that the P_{n+1}-decomposition problem on trees is polynomial. We need the following definitions : for an arbitrary root node r, we denote by $T_r(x)$ the subtree rooted on x, i.e. the tree induced by the set $V_r(x)$ of nodes such that x is on their (unique) path to r. If node x is a child of r, we introduce $T_r^+(x)$ as the tree $T[V_r(x) \cup \{r\}]$ induced by $V_r(x)$ and r. The trees $T_r^+(x)$, for x the children of r, define clearly a partition of E. Let $\mathcal{S}_r(i)$ be the subset of children x such that the number of edges in $T_r^+(x)$ is equal to i modulo n. We have the following characterization:

Theorem 4.2 *A tree admits a P_{n+1}-decomposition (a (n,n)-decomposition) if and only if*

$$\forall r \text{ a node}, \forall i = 1, \ldots, n \quad |\mathcal{S}_r(i)| = |\mathcal{S}_r(n-i)|$$

In addition if n is even, $|\mathcal{S}_r(n/2)|$ has to be even.

Proof. First we establish that this is a necessary condition. Consider a tree T that admits a P_{n+1}-decomposition, and let r be one of its nodes. Assume that there exists a child x of r whose subtree $T_r^+(x)$ has i edges modulo n, with $i > 0$. Let p be the path containing edge (x,r) in the decomposition. Since the number of edges in $T_r^+(x)$ is not a multiple of n, necessary p contains an edge (r,y), with y another child of r. Notice that p disconnects subtrees $T_r(x)$ and $T_r(y)$ from the rest of the tree. Thus the subtree forms by $T_r^+(x) \cup T_r^+(y)$ must be P_{n+1}-decomposable. It implies that its number of edges is a multiple of n, and hence that $T_r^+(y)$ has $n-i$ edges modulo n. This proves that a decomposition induces a perfect matching between $\mathcal{S}_r(i)$ and $\mathcal{S}_r(n-i)$. The condition follows.

We establish now that the condition is sufficient, by induction on the number of edges of the tree. Notice that in particular the condition implies that $|E|$ is a multiple of n. If $|E| = n$, then any node has a degree at most 2 to respect the condition. Thus T is a simple path of length n, obviously decomposable. If $|E| > n$, we choose arbitrarily one node r of degree at least 3. Notice that if all nodes have degree at most 2, as previously T is a simple path, and hence is decomposable. Let i be the largest integer smaller than n such that $\mathcal{S}_r(i)$ is not empty. The condition implies that we can pick one child x from $\mathcal{S}_r(i)$ and one child y from $\mathcal{S}_r(n-i)$ (this is also true if $i = 0$ setting by convention $\mathcal{S}_r(n) = \mathcal{S}_r(0)$). Consider then the partial subgraphs T' induced by the edges of $T_r^+(x) \cup T_r^+(y)$, and $T'' = T \backslash T'$ containing the other edges. Both T and T' are trees fulfilling the condition. As they have strictly less edges than T, they are P_{n+1}-decomposable by induction. □

Let us turn our attention to the complexity of the P_4-decomposition problem on general bipartite graphs. A first natural idea is to tranpose the problem into the line graph, as it was the key of the analysis of the $(2,3)$-decomposition problem in section 2. Here we need the following definition: for G and H two graphs, a H-factor of G is a set of vertex-disjoint subgraphs of G, each subgraph being isomorphic to H. Clearly the P_4-decomposition problem

on a bipartite graph G corresponds in a one-to-one relation to the P_3-factor problem on the line-graph of G. The complexity of the graph factor problems have been studied by Kirkpatrick and Hell [KH78, KH83]. They proved that this problem is $\mathcal{NP}$-complete as soon as H is a graph with at least three vertices. Hence the P_3-factor problem is $\mathcal{NP}$-complete on general graph. However the status of the P_3-factor problem on the line-graph of a bipartite graph was unknown. Next theorem proves it is $\mathcal{NP}$-complete:

Theorem 4.3 *The P_4-decomposition problem on bipartite graphs is $\mathcal{NP}$-complete.*

Proof. We reduce the P_4-decomposition problem from Exact Cover by 3 sets (X3C):

Problem X3C
Instance: A set X of $3m$ elements together with a collection $\mathcal{S}$ of subsets of 3 elements of X.
Question: Does there exists $\mathcal{C} \subseteq \mathcal{S}$ such that each element of X occurs in exactly one element of $\mathcal{C}$?

This problem is $\mathcal{NP}$-complete even if no element occurs in more than 3 subsets, see Garey and Johnson [GJ79] page 221, SP2. Given an instance $(X, \mathcal{S})$ of X3C, we construct in polynomial time the following graph $G(X, \mathcal{S})$.

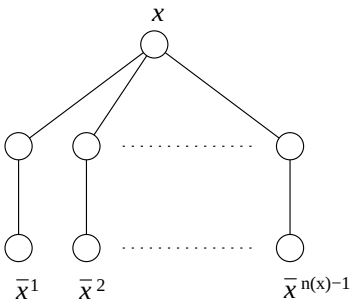


Figure 8: Subgraph $T(x)$.

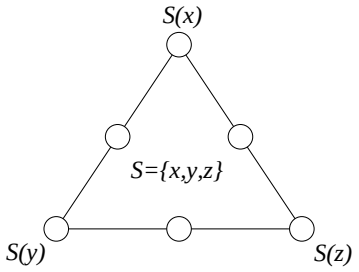


Figure 9: Cycle graph C_6 for set S.

We define one vertex x per element $x \in X$. If x occurs in $n(x)$ subsets of $\mathcal{S}$, we then add $n(x) - 1$ vertices labelled $\bar{x}^1, \ldots, \bar{x}^{n(x)-1}$. Each of these vertices is linked with a path of length 2 to vertex x (see figure 8). We denote by $T(x)$ this subgraph. Each subset $S \in \mathcal{S}$ is associated to one cycle graph C_6, we call identically (see figure 9). If S contains the elements $\{x, y, z\}$, we arbitrarily choose in its C_6 3 non-adjacent vertices to label by $S(x)$, $S(y)$ and $S(z)$.

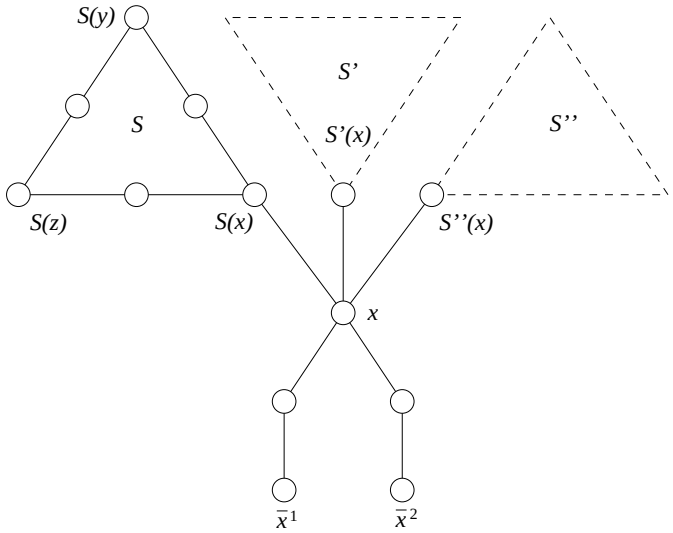
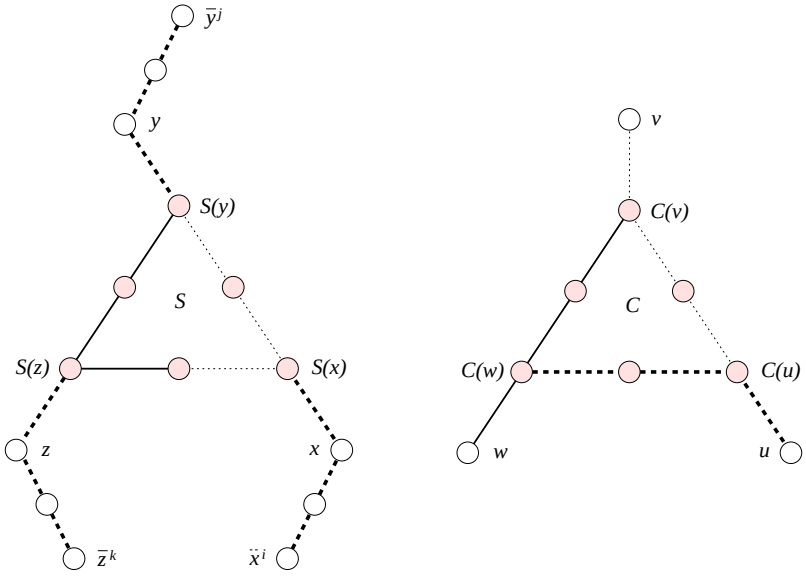


Figure 10: Graph $G(X, \mathcal{S})$.

Finally we add an edge for each element $x \in X$ and each subset $S \in \mathcal{S}$ containing x between vertex x from $T(x)$ and vertex $S(x)$ from the C_6 of set S. Figure 10 presents the graph $G(X, \mathcal{S})$. Notice that this graph is a bipartite graph.

We now establish that an instance of the X3C problem is positive if and only if graph $G(X, \mathcal{S})$ admits a P_4-decomposition. First assume that an exact cover $\mathcal{C}$ exists. For each subset $S \in \mathcal{S} \backslash \mathcal{C}$, we decompose its associated cycle graph C_6 into two paths of length 3. As each element x occurs in exactly one set of $\mathcal{C}$, by definition $n(x) - 1$ sets S in $\mathcal{S} \backslash \mathcal{C}$ contain x. Thus we can match each edge $(S(x), x)$ with one of the paths of length 2 between x and a leaf of $T(x)$. Figure 11(a) shows this P_4-decomposition. Now it only remains to decompose subgraphs formed by a cycle C_6 of set $C \in \mathcal{C}$, together with edges $(C(x), x)$. Clearly this subgraph is decomposable into three paths of length 3, see figure 11(b)).



(a) Decompossition for $S \notin \mathcal{C}$ (b) Decomposition for $C \in \mathcal{C}$

Figure 11: The P_4-decomposition from an exact cover $\mathcal{C}$

Conversely assume that graph $G(X, \mathcal{S})$ admits a P_4-decomposition. Each vertex $\bar{x}^i$ being of degree 1, this node is an extremity of a path in the decomposition. Such a path necessarily passes through vertex x and then ends on a vertex $S(x)$ of a C_6. Indeed ending on a node of $T(x)$ would leave uncovered an edge from another vertex $\bar{x}^j$. As a consequence exactly one edge $(S(x), x)$ for each vertex x does not belong to the paths described previously.

Consider the partial graph G' obtained when removing these paths, i.e. paths ending on an $\bar{x}^i$ node. Let $\mathcal{C}$ be the set of cycles C_6 with a non empty co-cycle in G'. We claim that $\mathcal{C}$ defines an exact cover: from what precedes, each vertex x is of degree 1 in G', its unique incident edge being of the form $(x, S(x))$. Thus $\mathcal{C}$ is a cover. Notice that it also implies that each C_6 is in a separate connected component. Recall that a cycle S has 3 edges in its co-cycle in G, one per element u in S. Since a C_6 contains 6 edges, either all the 3 edges of its co-cycle are present in G', or none of them, otherwise graph G' would clearly not be P_4-decomposable. It implies that $|\mathcal{C}| = |X|/3$, and hence $\mathcal{C}$ is an exact cover. □

As a corollary, the $(3, 3)$-decomposition problem is $\mathcal{NP}$-complete. In fact we can use the previous construction to answer the question on the decomposition into the family $\mathcal{F}_3$ of connected graphs of 3 edges, see figure 7:

Corollary 4.1 *The $\mathcal{F}_3$-decomposition problem is $\mathcal{NP}$-complete*

Proof. We use the same reduction as in theorem 4.3. We only need to check that a $\mathcal{F}_3$-decomposition implies the existence of an exact cover. As previously leaves of $G(X, \mathcal{S})$ are necessarily covered by a P_4, ending on a C_6. We define again $\mathcal{C}$ as the sets of $\mathcal{S}$ whose C_6 in G'

has a non empty co-cycle, where G' is the partial graph obtained by deleting paths containing a node of degree 1. It is then quite immediate to see that $\mathcal{C}$ still defines an exact cover, since whatever the way the connected component of C_6 is decomposed, the co-cycle either contains 0 or 3 edges in G'. □

In the proof of corollary 4.1 and in the Cohen and Tarsi's $\mathcal{NP}$-completeness proof of the $K_{1,n}$-decomposition [CT91], the graph used in the polynomial reduction is a bipartite graph, thus it does not admit K_3 as an induced subgraph. It demonstrates that both the $\{P_4, K_{1,3}\}$-decomposition problem and the $\{K_3, K_{1,3}\}$-decomposition problem are $\mathcal{NP}$-complete. From these results, we conclude that for all subfamily $\mathcal{F}'$ of $\mathcal{F}_3$, the $\mathcal{F}'$-decomposition problem is $\mathcal{NP}$-complete. This reinforces our believe in conjecture 3.1.

Finally one can consider the following *path-decomposition* problem: the inputs are a graph G and two integers a and b, and we are asked if an (a, b)-decomposition exists for G. This corresponds to an (a, b)-decomposition problem where a and b are now part of the instance. From the theorem 4.3, we deduce immediately that the path-decomposition problem is $\mathcal{NP}$-complete.

5 Decomposition of Eulerian Graphs

An Eulerian graph contains a cycle which visits each edge exactly once. A traversable graph contains an eulerian walk. Euler algorithm build this cycle/walk in linear time $\mathcal{O}(|E|)$. We can view this path as a word of length $|E|$ on the alphabet of the vertices. Now, the problem consists in cutting this word into sub-words of length between a and b. By extension we say that an integer m is (a, b)-decomposable if it can be written as a sum of integers all laying in interval $[a, b]$. We prove the following simple fact :

Fact 5.1 *An integer m admits an (a, b)-decomposition if and only if it exists an integer K such that $Ka \leq m \leq Kb$*

Proof. If m can be written as $\sum_{i=1}^{K} x_i$, with $a \leq x_i \leq b$, clearly $Ka \leq m \leq Kb$. Conversely assume that inegality $Ka \leq m \leq Kb$ holds for some K. We denote by α and β the quotient of the euclidian division of m by a and b, respectively. To avoid trivial cases we assume that neither a nor b divide m, which implies that $\beta b < m < (\alpha + 1)a$. The assumption on the existence of K imposes thus that $\alpha < \beta$. Now for each integer i, let (q_i, r_i) be the quotient and the rest in the euclidian division of $m - ia$ by b. We have then $q_0 = \beta$ and $q_\alpha = 0$. It implies that $q : \{0, \alpha\} \mapsto \{0, \beta\}$ is not injective. As in addition q is clearly non-increasing, it necessarily exists an index j such that $q_j = q_{j+1}$. We immediately get $r_{j+1} = r_j + a$, and thus $m = (j + 1)a + r_{j+1} + q_{j+1}b$, with $a \leq r_{j+1} < b$, is a valid decomposition. □

Notice that the proof establishes that if m admits an (a, b)-decomposition, then it exists a decomposition where at most one term is not equal neither to a nor b. This results can also be derived from elementary LP theory. Indeed if the linear program $\max \sum_{i=1}^{K} x_i$ subject to $\sum x_i \leq m$ and $a \leq x_i \leq b$ for $i = 1 \dots K$ has an optimum with objective value m, then it admits an optimum basis solution where K variables are null. As only slack variables can be zero, at most one variables x_j does not belong to $\{a, b\}$. Since m is integral, and all other x_i are in $\{a, b\}$, variable x_j is thus integral, which provides a valid decomposition of m.

We thus have the immediate following characterization:

Corollary 5.1 *A traversable graph is (a, b)-decomposable if and only if $\lceil |E|/b \rceil \leq \lfloor |E|/a \rfloor$*

Proof. Due to fact 5.1, graph G is (a, b)-decomposable if and only if it exists an integer K such that $Ka \leq |E| \leq Kb$. This is equivalent to write that it eists an integer K such that $K \leq |E|/a$ and $|E|/b \leq K$. The result follows. □

We are interested now in finding a decomposition for traversable graph that fulfill condition of corollary 5.1. This is clearly not a difficult problem:

Lemma 5.1 *Finding an (a, b)-decomposition on traversable graphs is polynomial for all integers a and b.*

Proof. The algorithm we propose to find the decomposition of a word of length m is directly derived from the proof of fact 5.1: we build the series of the ia and $m - q_i b$, represented respectively by variables A and B. We initially set A to αa and B to m. At a step i we check if $A - a$ is greater than $B - b$. If it is the case then we have found an index i such that $q_i = q_{i+1}$. Otherwise we have $(i-1)a < m - q_i b - b$ and trivially $q_{i+1} = q_i + 1$. Algorithm 1 runs in time $\mathcal{O}(m)$, which is polynomial for the graph decomposition problem. □

Algorithm 1 Word Decomposition Algorithm

Require: $m, a, b \in \mathbb{N}$, $a \leq b \leq m$
Ensure: (i, j) such that $m - ia - jb \in [a, b]$
1: $i := 0$; $A := 0$; $q := 0$; $B := m$ *// initialization*
2: **while** $A + a < m$ **do**
3: $\quad i := i + 1$; $A := A + a$ *// computation of* α
4: **end while**
5: **while** $A \geq a$ **and** $A - a < B - b$ **do**
6: $\quad i := i - 1$; $A := A - a$
7: $\quad q := q + 1$; $B := B - b$ *// update of* q_{i-1}
8: **end while**
9: **if** $A \geq a$ **then**
10: $\quad$ **output** $(i - 1, q)$
11: **end if**

Notice that we use the property that at most one term is not equal neither to a not b to describ the output simply by giving the number of a and b used in the decomposition. This allows to design a polynomial time algorithm also for the integer decomposition problem, where the instance size is in $\mathcal{O}(\log m)$. Indeed we are basically looking for a point j where function q is not injective ($q_j = q_{j+1}$). We can use a classic dichotomic search, dividing at each step the interval (initially $[0, \alpha]$) into 2 parts, and selecting the one which is strictly larger than its image under q. Each step only requires the computation of an euclidian division to get a value q_i, which can be implemented in logarithmic time in m.

If a decomposition exists, one may want to obtain the decompostion using the minimum number of paths. We call this optimization problem *minimum (a, b)-decomposition*. In our our industrial application, it corresponds to minimize the size of the fleet of vehicles to use. We prove in next theorem that the minimum (a, b)-decomposition problem is polynomial on traversable graphs.

Theorem 5.1 *If a traversable graph G is (a, b)-decomposable, finding a decomposition using the minimum number of paths can be done in polynomial time.*

Proof. Let G be a traversable (a, b)-decomposable graph of m edges. Clearly any (a, b)-decompositon will use at least $K^* = \lceil \frac{m}{b} \rceil$ paths. We now exhibit a decompositon using exactly K^* paths, which hence solves optimally the minimum (a, b)-decomposition problem. The basic idea is to look for an (α, β)-decomposition, with $a \leq \alpha \leq \beta \leq b$, such that m and K^* can be written as $m = \alpha u + \beta v$ and $K^* = u + v$.

Let $\alpha = \lfloor m/K^* \rfloor$ and $\beta = \lceil m/K^* \rceil$. By definition we have $K\alpha \leq m \leq K\beta$, and thus G is (α, β)-decomposable due to fact 5.1. Notice that corollary 5.1 implies that $K^* a \leq m \leq K^* b$. Hence we have $a \leq \alpha$ and $\beta \leq b$. As a consequence an (α, β)-decomposition is a valid (a, b)-decomposition.

Let q and r be the quotient and the rest of the euclidian division of m by β. If $r = 0$, then $q = K^*$ and we simply use K^* paths of length β. Otherwise $q = K^* - 1$ and $\beta = \alpha + 1$. We then decompose G into $u = \beta - r$ paths of length α and $v = K^* - (\beta - r)$ paths of length β. Since $\beta = \alpha + 1$, the sum of the length of these paths is exactly m. To conclude that this is a valid (α, β)-decomposition, we need to check that both u and v are positive integers. For u it is asserted by the definition of euclidian division. For v, we have $m = (K^* - 1)\beta + r \geq K^* \alpha$. Using the fact that $\beta = \alpha + 1$, we directly get $v \geq 0$. □

6 Decomposition results for weighted graphs

In this section, we considered the (a, b)-decomposition problem on weighted graphs. A valuation w_e is associated to each edge e. We assume here that w is a strictly positive integer valued function. The weight of a path is then defined as the sum of the valuation of its edges. Clearly the (a, b)-decomposition problem on weighted graph is a generalization of the unweighted version, and thus the $(3, 3)$-decomposition problem is $\mathcal{NP}$-complete due to theorem 4.3. We prove in this section that the $(2, b)$-decomposition remains polynomial on weighted graphs, while the $(3, 3)$-decompositon of weighted Eulerian graphs is $\mathcal{NP}$-complete. This result contrasts with lemma 5.1 of section 5.

To avoid trivial cases, we assume that the weigth of any edge is lower or equals to b, otherwise graph G is non (a, b)-decomposable. The $(1, b)$-decomposition problem can be solved using one path per edge. Next theorem states that the $(2, b)$-decomposition remains poynomial on weighted graphs with strictly positive valuation.

Theorem 6.1 *The $(2, b)$-decomposition problem on weighted graph is polynomial for all b.*

Proof. Any edge of weight greater than 2 can be used as a single path. Consider the partial graph G_1 where only edges of weight 1 are present. Notice that G_1 is by construction an unweighted graph.

If $b = 2$, clearly graph G is $(2, 2)$-decomposable if and only if G_1 is $(2, 2)$-decomposable, which can be decided in linear time, see corollary 2.1.

If $b \geq 3$, we know that a connected component of G_1 is $(2, b)$-decomposable if and only if it is not an odd-degree tree, see theorem 2.1. Thus assume that one component T is an odd-degree tree. If in G no edge of weight in $[2, b-1]$ is incident to a node of T, then G is not $(2, b)$-decomposable. Conversely such an edge (y, z) exists, with for instance $y \in T$, we form a path with edge (y, z) plus one edge (x, y) of weight 1 incident to y. Removing this edge (x, y) disconnects T into 2 trees $T(x)$ and $T(y)$. The degree of x and y has decreased by one, and thus is even. It results that $T(x)$ and $T(y)$ are $(2, 3)$-decomposable.

In conclusion, a graph G is $(2, b)$-decomposable if and only if we can find for each odd-degree tree of G_1 one incident edge of weight in $[2, b-1]$. Notice that if one edge is incident to 2 distinct odd-degree trees, we can use it in a path of 3 edges to decompose both trees, as long as its weigth is lower or equal to $b-2$. If one edge is incident to only one odd-degree tree, we can always pick it up to decompose these tree. In fact the only difficult case is when odd-degree trees have incident edges only of degree $b-1$, and those edges are incident to 2 odd-degree trees. To decide this problem, we can use the bipartite graph $(\mathcal{T} \cup I, F)$ when $\mathcal{T}$ is the set of odd-degree trees in G_1, I the edges of weight in $[2, b-1]$, and F the incidence relation between $\mathcal{T}$ and I. Each odd-degree tree is a source of 1, and each edge is a sink of either 1 if its weight is $b-1$, or 2 otherwise. All arcs of F are of capacity 1. We then ask if a flow of $|\mathcal{T}|$ exists. □

Previous theorem may incitate to think that polynomial results can be extended to the weighted version of the decomposition problem. For traversable graphs, however, this is not the case. Consider what happens if we apply the Word Decomposition algorithm 1 on the graph of figure 12, looking for a $(3, 3)$-decomposition.

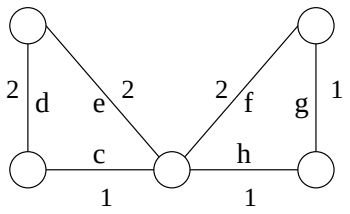


Figure 12: An Eulerian graph. The weight and label of each edge is given on the figure.

The answer of the algorithm now depends on the intial choice of the Eulerian cycle. Let us choose the word $cdefgh$. Clearly the three consecutive edges def can not be decomposed into paths of length 3. However with another Eulerian cycle, the word $cdehgf$ admits a decomposition into cd, eh and gf. It follows that the algorithm should test the decomposition of all Eulerian cycles to deliver a negative answer. Since the number of different Eulerian cycles in a graph may

be exponentially large, the algorithm fails to solve the decomposition problem in polynomial time. In fact we have the following result:

Lemma 6.1 *The (a, b)-decomposition problem on unweighted graph polynomially reduces to the (a, b)-decomposition problem on weighted Eulerian graph*

Proof. Consider an instance of the (a, b)-decomposition problem, composed of an unweighted graph $G = (V, E)$. Since the number of odd-degree nodes is even in any graph, we can choose a perfect matching $M \subset V \times V$ between these nodes. From graph G we then construct the (multi) Eulerian weighted graph $G' = (V, E \cup M)$. For all edges of E, we assign a weight of 1. The edges in M have a weight of b. The values of a and b are unchanged.

If G is (a, b)-decomposable, then G' is (a, b)-decomposable: use the same paths to decompose E, plus one path per edge in M. Conversely if G' is (a, b)-decomposable, since each edge of E' has a weight of b, it has not been appended to another edge. Thus the decomposition defines an (a, b)-decomposition of the partial graph G. □

Leading to the following corollary, using theorem 4.3:

Corollary 6.1 *The $(3, 3)$-decomposition problem on weighted Eulerian graph is $\mathcal{NP}$-complete*

7 Conclusion

We studied the problems of graph decomposition into paths: given a graph $G = (V, E)$, the problem is to determine whether the edge-set G can be decomposed into partial subgraphs $G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k)$ such that $(E_i)_{i=1,k}$ defines a partition of E, $a \leq |E_i| \leq b$ and G_i admits an Eulerian path. In this paper we investigate the complexity of the (a, b)-decomposition problem. We give a characterization of $(2, b)$-decomposable graphs, from which it is possible to decide in linear time the $(2, b)$-decomposition problem. We prove the $\mathcal{F}_{\exists}$-decomposition problem is $\mathcal{NP}$-complete, even on bipartite graphs. However we identify some polynomial cases for the (a, b)-decomposition when $a \geq 3$: Eulerian graphs, trees if $a = b$, complete bipartite for the $(3, 3)$-decomposition. We have extended some results for the (a, b)-decomposition problem on weighted graph. The $(2, b)$-decomposition problem remains polynomial contrary to the (a, b)-decomposition on weighted Eulerian graph.

We conjecture in this paper that the $\mathcal{H}$-decomposition problem into a given family $\mathcal{H} = \{H_1, \ldots, H_n\}$ of connected graphs is $\mathcal{NP}$-complete if and only if all H_i-decomposition problems are $\mathcal{NP}$-complete. It would be also interesting to extend the completeness result of the $(3, 3)$-decomposition problem on (bipartite) graphs to any (k, k)-decomposition problems.

The results of this paper maybe extended to directed graphs. On one hand it is quite immediate to adapt proof of theorem 4.3 to show that the $(3, 3)$-decomposition problem remains $\mathcal{NP}$-complete. On the other hand, lemma 2.2 does not hold for directed graphs. For instance graph of figure 13 is $(2, 3)$-decomposable, but a decomposition uses necessarilly 2 directed paths of length 3. Hence the $(2, b)$-decomposition problem on directed graphs is open. If it happens to be polynomial, one can wonder if a nice characterization exists for $(2, b)$-decomposable directed graphs. By the way does a characterization exists even for $(2, 2)$-decomposable directed graphs?

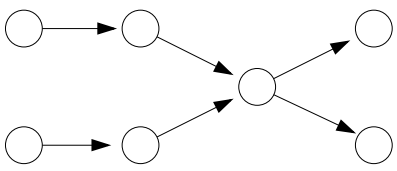


Figure 13: A graph $(2, 3)$-decomposable

We can also consider some optimisation version of the decomposition problem. The H-decomposition problem is related to the optimisation problem denoted by the maximum H-packing problem [CY97]. An H-packing of a graph G is a set $L = G_1, \ldots, G_s$ of edge-disjoint subgraphs of G, where each subgraph is isomorphic to H. The H-packing number of G is the

maximum cardinality of an H-packing of G. The maximisation of H-packing corresponds also to the minimisation of the number of edges that the H-packing does not cover. In [Yus04], Yuster studied the $\mathcal{F}$-packing number where $\mathcal{F}$ is a fixed finite or infinite family of graphs. For a graph G, the $\mathcal{F}$-packing number is the maximum number of pairwise edge-disjoint copies of elements of $\mathcal{F}$. The definition of the $\mathcal{F}$-packing number given by Yuster must be moderated, this favours the smallest graph in the family $\mathcal{F}$. If some graphs in $\mathcal{F}$ do not have the same number of edges, the $\mathcal{F}$-packing number of G can be associated with one $\mathcal{F}$-decomposition which does not cover all edges, even if the graph G admits a $\mathcal{F}$-decomposition.

For our industrial problem, it would be interesting to minimize the number of vehicles used. We suggest another optimisation problem : if G admits an (a, b)-decomposition, the *minimum* (a, b)-decomposition consists in minimizing the packing number. More genarally the minimum $\mathcal{F}$-decomposition is defined as minimizing the packing number for the family $\mathcal{F} \cup \{K_2\}$, i.e. adding one edge to family $\mathcal{F}$.

References

[Alo83] N. Alon. A note on the decomposition of graphs into isomorphic matchings. *Acta Mathematica Hungarica*, 42:221–223, 1983.

[Bei70] L.W. Beineke. Characterizations of derived graphs. *Journal of Combinatorial Theory*, 9:129–135, 1970.

[BJV00] C. Barnhart, H. Jin, and P.H. Vance. Railroad blocking : a network design application. *Operations Research*, 48(4):603–614, 2000.

[BR83] A. Bialostocki and Y. Roditty. $3k_2$-decomposition of a graph. *Acta Mathematica Hungarica*, 40:201–208, 1983.

[CGG03] T.G. Crainic, B. Gendron, and I. Ghamlouche. Cycle-based neighbourhoods for fixed-charge, capacitated multicommodity network design. Technical report, Centre de recherche sur les transports, Universit de Montral, 2003.

[CT91] E. Cohen and M. Tarsi. Np-completeness of graph decomposition problems. *Journal of Complexity*, 7(2):200–212, 1991.

[CY97] Y. Caro and R. Yuster. Packing graphs: The packing problem solved. *the electronic journal of combinatorics*, 4, 1997.

[DT97] D. Dor and M. Tarsi. Graph decomposition is np-complete : A complete proof of holyer's conjecture. *SIAM Journal on Computing*, 26(4):1166–1187, 1997.

[Edm65] J. Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[FLT85] O. Favaron, Z. Lonc, and M. Truszczynski. Decomposition of graphs into graphs with three edges. *Ars Combin.*, 20:125–146, 1985.

[GCF99] B. Gendron, T.G. Crainic, and A. Frangioni. *Telecommunications Network Planning*, chapter Multicommodity Capacitated Network Design, pages 1–19. Kluwer Academics Publisher, 1999.

[GJ79] M.R. Garey and M.D. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[HJ04] R. Häggkvist and R. Johansson. A note on edge-decompositions of planar graphs. *Discrete Mathematics*, 283(1-3):263–266, 2004.

[HLY99] K. Heinrich, J. Liu, and M. Yu. p_4-decompositions of regular graphs. *Journal of Graph Theory*, 31(2):135–143, 1999.

[Hol81] I. Holyer. The np-completeness of some edge partition problems. *SIAM Journal on Computing*, 10(4):713–717, 1981.

[KH78] D.G. Kirkpatrick and P. Hell. On the completeness of a generalized matching problem. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 240–245, 1978.

[KH83] D.G. Kirkpatrick and P. Hell. On the complexity of general graph factor problems. *SIAM Journal on Computing*, 12(3):601–609, 1983.

[Las75] M. Las Vergnas. A note on matchings in graphs. In *Colloque sur la Theorie des Graphes*, Cahiers Centre Etude Rech. Oper., pages 257–260, Paris, 1975.

[Lon97] Z. Lonc. Edge decomposition into copies of $sk_{1,2}$ is polynomial. *Journal of Combinatorial Theory, Series B*, 69(2):164–182, 1997.

[PT04] M. Priesler and M. Tarsi. On some multigraph decomposition problems and their computational complexity. *Discrete Mathematics*, 281(1-3):247–254, 2004.

[PT05] M. Priesler and M. Tarsi. Multigraph decomposition into stars and into multistars. *Discrete Mathematics*, 296(2-3):235–244, 2005.

[Sum74] D. Sumner. Graphs with 1-factors. In *American Mathematical Society*, volume 42, pages 8–12, 1974.

[Tar72] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[Yus04] R. Yuster. Integer and fractional packing of families of graphs. *Random Structures and Algorithms*, 26:110–118, 2004.