Algorithmic study of 2-interval graphs

Alexandre Simon

A thesis presented for the degree of
Master of Science

Thesis committee :	Dr. ir. L.J.J. van Iersel	TU Delft
	Dr. M.M. de Weerdt	TU Delft
	Dr. Stéphane Vialette	University Gustave Eiffel
	Dr. Florian Sikora	University Paris Dauphine

Faculty of Electrical Engineering, Mathematics and Computer Science
TU Delft

Contents

1 Acknowledgements

First and foremost, I am extremely grateful to my supervisors, Dr. Stéphane Vialette and Dr. Florian Sikora for their supervision throughout the year, their availability to discuss regularly, their support and their advice for my future career.

Second of all, I would also like to thank Dr. Leo van Iersel for following the advancement of my thesis and for his advice to improve it.

In addition, I would like to thank Dr. Mathijs de Weerdt for assessing my thesis.

Lastly, I would like to thank my family and my friends for their support and their company.

2 Introduction

2.1 Motivation

Interval graphs play an important role in graph theory and have intensively been studied for over sixty years [21]. One main reason for researchers to be interested in interval graphs is that many real-world problems can be modeled as interval graphs in a natural way. Their applications are very diverse and go from modelling food webs [10] to studying scheduling problems [5, 11] and assembling contiguous subsequences in DNA mapping [35]. Let us, for example, suppose that various university clubs need a room to offer an activity. Each activity begins at a certain time and ends at an other time. The university administration can only allocate one room and wants to satisfy the maximum number of activities, how can they do ? A very efficient solution would actually be to model each activity by an interval whose endpoints correspond to the beginning and end hours of the activity and build the corresponding interval graph as shown in Figure 1.

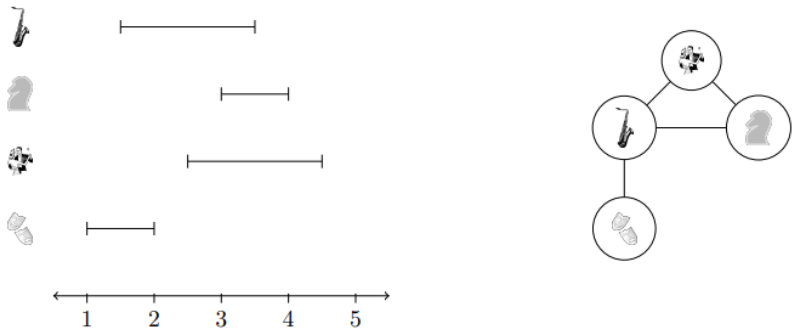

Figure 1: University clubs time slots with the associated interval graph

Then, they can find the maximum set of vertices that are all pairwise non adjacent. This therefore corresponds to solving the Maximum Independent Set problem on this resulting interval graph. Let us now suppose that the university administration has enough rooms for all the university clubs but that they want to allocate the least number of rooms to satisfy all activities. They can then reuse the exact same interval graph and find the maximum number of colours that can be allocated to each vertex so that no two vertices with the same colour are adjacent. In other words, they solve the Colouration problem on it.

In addition to their wide range of applications, interval graphs stand out from general graphs due to the fact that the most studied NP-hard problems on general graphs often become solvable in linear time on interval graphs. This is, for example, the case for Colourability [18], Independent Set [13] as we just mentioned but also for Feedback Vertex Set [26] that we will introduce later. Finally, the class of interval graphs is also very convenient as its recognition can also be realised in linear time and various characterisations have been developed [25, 14, 17]. Nevertheless, the class of interval graphs is also very restricted as any interval graph is chordal and therefore it is not possible to represent a cycle of length at least 4 by an interval representation.

To overcome this drawback, several generalisations of interval graphs have also been considered. In the following, we will mainly focus on 2-interval graphs, on 2-track graphs and

more generally on t-interval graphs and d-track graphs. These classes naturally generalise interval graphs where instead of associating one interval to each vertex we associate d intervals on either the same real line or on d parallel tracks. More recently, these classes have found some real-world applications since two intervals can, for example, model two associated tasks in scheduling [6] or two complementary segments of RNA for RNA secondary structure prediction and comparison.

As explained in [32], like DNA, RNA (ribonucleic acid) is assembled as a chain of nucleotides, but unlike DNA, RNA is found in nature as a single strand folded onto itself by complementary base pairing, rather than a paired double strand. This particular secondary structure is generally characterized by helices (contiguous base pairs), and various kinds of loops (unpaired nucleotides surrounded by helices). Furthermore, the formation of such stable secondary structure by complementary base pairing largely determines the structural stability and function of non-coding RNA (ncRNA). This then induces that ncRNA genes across different species are most similar in the pattern of nucleotide complementary rather than in the genomic sequence. Each helix of the RNA secondary structure can then be modeled as a 2-interval as they usually needs to be disjoint in the structure. This then allows to naturally model the secondary structure by a 2-interval graph as shown in Figure 2 below.

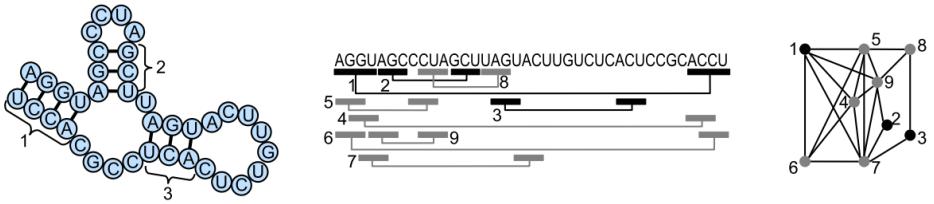

Figure 2: Helices in a RNA secondary structure (a) can be modeled as a set of balanced 2-intervals among all 2-intervals corresponding to complementary and inverted pairs of letter sequences (b), or as an independent subset in the balanced associated 2-interval graph (c).

In [12], it was even suggested to model each helix with 2 intervals of the same length: a balanced 2-interval. This then allows to define balanced 2-interval graphs and more generally balanced t-interval graphs and balanced d-track graphs. This is also motivated by a natural application from scheduling where we can model two associated and disjoint tasks of the same duration by a balanced 2-interval. In [6], they also considered that all tasks in their framework have the same duration which motivates the use of unit 2-interval graphs and more generally unit t-interval graphs and unit d-track graphs.

Nevertheless, most problems that are NP-hard on general graphs remain NP-hard on d-track or t-interval graphs (for $d \geq 2$ and $t \geq 2$) even if we restrict the length of the intervals. In the following, we will study the links between the different classes of multiple interval graphs and also show that Vertex Cover, Clique Cover and Biclique Cover are NP-complete and Feedback Vertex Set is APX-hard on unit 2-track graphs.

Finally, we will also introduce the class of boxicity d graphs which also is a generalisation of interval graphs where instead of considering an intersection model of intervals, we use boxes of dimension d. Seemingly similar to d-track graphs, we will see how they are related.

2.2 Preliminaries

Let us first introduce the definition of a graph :

Definition 1. *A graph is a pair $G = (V, E)$, where V is a set whose elements are called vertices, and E is a set of paired vertices, whose elements are called edges. The graph is said to be simple and undirected if there can be at most one edge between two vertices and if all the edges are bidirectional*

We will now only consider simple and undirected graphs. Let us now introduce the definition of interval graphs.

Definition 2. *A graph $G = (V, E)$ with $V = \{v_1, v_2, ..., v_n\}$ is an interval graph if there exists a family of closed intervals $\mathscr{I} = \{I_1, I_2, ..., I_n\}$ (interval representation) associated to the vertices such that $I_i \cap I_j \neq \emptyset \Leftrightarrow (v_i, v_j) \in E$. In other words, a graph is an interval graph if it has an intersection model consisting of intervals on a straight line.*

We give an example of an interval graph with its associated intervals in Figure 3.

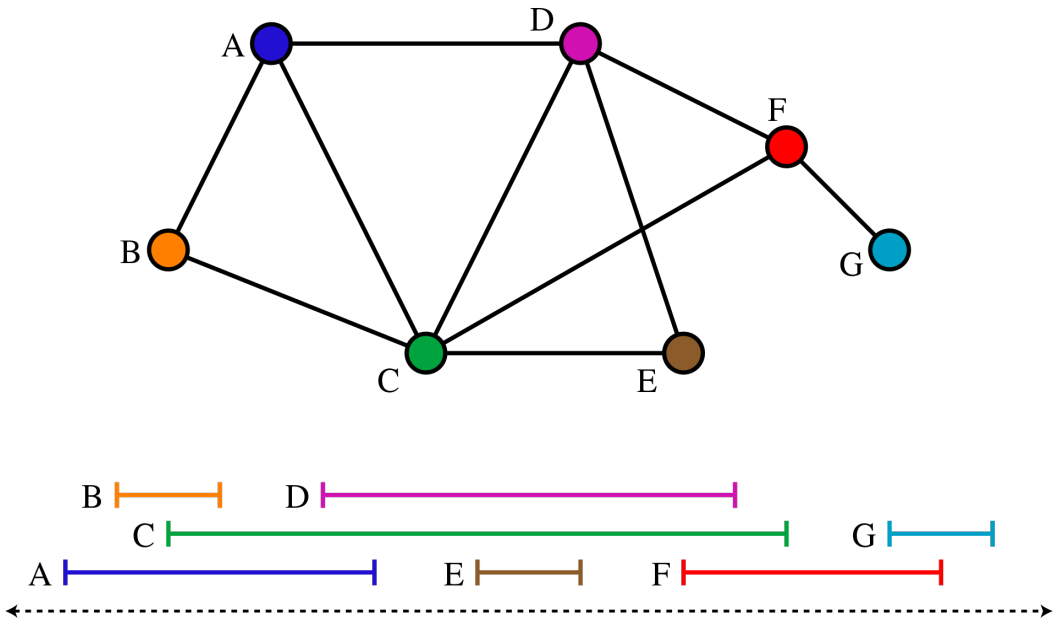

Figure 3: An interval graph with its interval representation

Let us also note that the following theorem from [25] allows to redefine the class of interval graphs in terms of forbidden subgraphs :

Theorem 1 ([25])**.** *A graph G is an interval graph if and only if it does not contain C_{n+4}, T_2, X_{31}, XF_2^{n+1} or XF_3^n as an induced subgraph.*

We illustrate, in Figure 4, the graphs C_{n+4}, T_2, X_{31}, XF_2^{n+1} and XF_3^n.

Two subclasses of interval graphs that can be directly obtained by restricting the interval representation are the classes of unit interval graphs and proper interval graphs defined below.

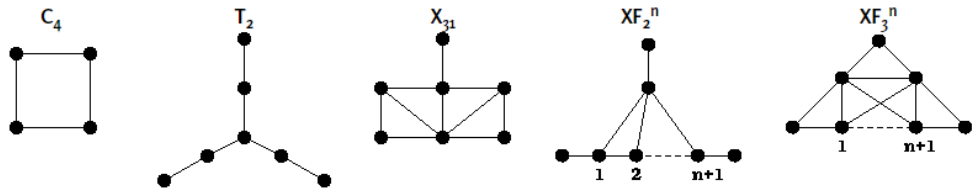

Figure 4: Forbidden subgraphs in an interval graph

Definition 3. *An interval graph is a proper interval graph if it has an interval representation where no interval is included in another.*

Definition 4. *An interval graph is a unit interval graph if it has an interval representation where the length of each interval is one.*

Also, note that it is only needed that the intervals of a unit interval graphs have the same length as we can always re-scale them all. We say that a graph $G = (V, E)$ is bipartite if its vertex set V can be partitioned into two disjoint and independent sets A and B. We thus have that $V = A \cup B$ and every edge of G has one endpoint in A and the other one in B. We say that a graph is complete if every pair of vertices is connected by a unique edge. Finally a graph $G = (V, E)$ is complete bipartite if it is bipartite such that $V = A \cup B$ and every pair of vertices $a \in A$ and $b \in B$ is connected by a unique edge. We will denote by $K_{m,n}$ the complete bipartite graph whose vertex set is partitioned into A and B with $|A| = m$ and $|B| = n$. We will later call the graph $K_{1,3}$ a claw (see Figure 5) and say that a graph is claw-free if it does not contain $K_{1,3}$ as an induced subgraph.

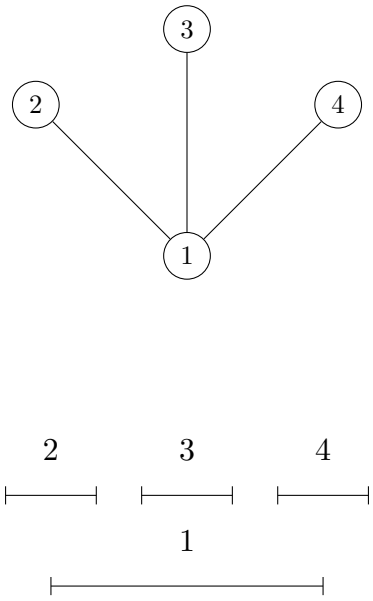

Figure 5: $K_{1,3}$ and an interval representation of it

In addition, the following theorem from [28] claims that unit interval graphs and proper interval graphs coincide. Furthermore, it gives a simple relationship between interval graphs and unit interval graphs.

Theorem 2 ([28])**.** *The following statements are equivalent when G is a simple graph :*

1. *G is a unit interval graph*
2. *G is an interval graph with no induced $K_{1,3}$*
3. *G is a proper interval graph*

Below, we give a short proof of this theorem from [7].

Proof. It is easy to see that 1. $\rightarrow$ 2. (or 3. $\rightarrow$ 2.) as in any interval representation of $K_{1,3}$, the intervals of the three leaves must be pairwise disjoint but they all need to intersect the interval of the central vertex. The interval of the central vertex must then contain the interval of one of the leaves implying that the representation cannot be proper and that the interval of the central vertex must be longer that the one it contains.
Let us now prove that 2. $\rightarrow$ 3.. We therefore consider a claw-free interval graph $G = (V, E)$ and an interval representation that assigns to each $u \in V$ an interval I_u. We now show how we can modify this representation so it becomes proper. Since G is claw-free, there is no pair of vertices $x, y \in V$ such that $I_y \subset I_x$ and I_x intersects some intervals on the left and on the right of I_y that are disjoint from I_y. If $I_x = [a, b]$ and $I_y = [c, d]$ then if $a < c \leq d < b$ it means that either $[a, c]$ or $[d, b]$ does not contain any endpoint of intervals that do no intersect I_y. We can therefore extend I_y past the end of I_x without changing the graph obtained by the interval representation to make it proper. We can thus repeat this operation for each interval $I_y \subset I_x$.
What remains to prove now is that 3. $\rightarrow$ 1., i.e., we can obtain a unit interval representation of any graph G given a proper interval representation of it. When no interval includes another, the right endpoints have the same order as the left endpoints. We can then process the representation from left to right and adjust the length of each interval to 1. At each step until all have been adjusted, let $I_x = [a, b]$ be the unadjusted interval that has the leftmost left endpoint. Let $\alpha = a$ unless I_x contains the right endpoint of some other interval, in which case let α be the largest such right endpoint. Such an endpoint would belong to an interval that has already been adjusted to have length 1; thus $\alpha < \min\{a+1, b\}$. Now, adjust the portion of the representation in $[a, \infty)$ by shrinking or expanding $[\alpha, b]$ to $[\alpha, a + 1]$ and translating $[b, \infty)$ to $[a + 1, \infty)$. The order of endpoints does not change, intervals earlier than I_x still have length 1, and I_x also now has length 1. Iterating this operation produces the unit interval representation. □

Moreover, we can also generalize the class of interval graphs in the two following natural ways.

Definition 5. *A graph $G = (V, E)$ is a d-track graph if, given d parallel tracks (real lines), there exist d families of closed intervals $\mathscr{I}^{(k)} = \{I_1^{(k)}, I_2^{(k)}, ..., I_n^{(k)}\}$ for $1 \leq k \leq d$ (d-track representation) associated to the vertices such that $I_i^{(k)} \cap I_j^{(k)} \neq \emptyset$ on any track k $\Leftrightarrow (v_i, v_j) \in E$.*

An alternative definition is the following :

Definition 6. *A graph $G = (V, E)$ is a d-track graph if there exist d interval graphs $G_1, G_2, ..., G_d$ with $V(G) = V(G_i)$ for $1 \leq i \leq d$ and satisfying $E(G) = E(G_1) \cup E(G_2) \cup ... \cup E(G_d)$*

It can easily be seen from the definition that the 1-track graphs are the interval graphs and for any integer d, the class of d-track graphs is included in the class of $(d+1)$-track graphs as we can realise no intersection on the last track.

Finally, let I_1, I_2,..., I_t be t disjoint intervals, we say that I is a t-interval if $I = I_1 \cup I_2 \cup .. \cup I_t$. Furthermore, we say that two t-intervals I and J intersect if one of the t intervals of I intersect one of the t intervals of J. We can therefore introduce a second natural generalisation of interval graphs below.

Definition 7. *A graph $G = (V, E)$ with $V = \{v_1, v_2, ..., v_n\}$ is an t-interval graph if there exists a family of t-intervals $\mathscr{I} = \{I_1, I_2, ..., I_n\}$ (t-interval representation) associated to the vertices such that $I_i \cap I_j \neq \emptyset \Leftrightarrow (v_i, v_j) \in E$. In other words, a graph is a t-interval graph if it has an intersection model consisting of t-intervals on a straight line.*

We can thus notice that the 1-interval graphs are the interval graphs and that for any integer t, the class of t-interval graphs is included in the class of $(t+1)$-interval graphs as we can let an interval of each t-interval be disjoint from any other interval. Furthermore, it can be seen that the class of t-track graphs is included in the class of t-interval graphs as we can merge the t tracks by placing them one after the other. It thus forms t-intervals on a single real line.

Finally, as for interval graphs, we can restrict all the intervals to have the same length in a d-track or a t-interval representation to obtain the classes of *unit t-interval* graphs and *unit d-track* graphs.

Furthermore, if only the intervals of each t-interval have the same length then we say that the t-interval graph is *balanced*. Similarly, if only the intervals that represent the same vertex on each track have the same length then we also say that the d-track graph is *balanced*.

Finally, let us also introduce the notion of depth for all the introduced classes so far. The depth of a family of d-intervals (resp. d-track intervals) is the maximum number of intervals that share a common point.

We can also define the track number $t(G)$ or the interval number $i(G)$ of a graph G as the minimum integer d such that G is a d-track or a d-interval graph. Analogously, we define the unit track number $t_u(G)$ or the unit interval number $i_u(G)$ of a graph G as the minimum integer d such that G is a unit d-track or a unit d-interval graph. In [23], some upper bounds on the unit track number and the track number of any graph are given :

Theorem 3. *Let G be a graph on n vertices, m edges and maximum degree Δ. Then $t(G) \leq \lfloor \frac{n}{2} \rfloor$ and $t_u(G) \leq \lceil \frac{m}{2} \rceil$*

Proof. The bounds are proved by induction. First let us prove that $t(G) \leq \lfloor \frac{n}{2} \rfloor$. The bound clearly holds for the base case $n \leq 1$ since this results in a graph with no edge and thus no tracks are needed. Let us now assume that the graph has some edges (otherwise no tracks are needed). Let (u, v) be an edge of the graph and let us consider the subgraph of G with the vertices u and v deleted. By the induction hypothesis, this subgraph can be represented on $\lfloor \frac{n-2}{2} \rfloor$ tracks. We can now expand the representation by adding two disjoint intervals on each track for u and v. Finally, we open a new track and put two overlapping (but not included in one another) intervals for u and v. Finally for each vertex intersecting u or v (or both) we can add an interval that overlaps u or v (or both). For each vertex that does

not intersect u nor v, we add a disjoint interval. We therefore obtain a representation of G on $\lfloor \frac{n-2}{2} \rfloor + 1 = \lfloor \frac{n}{2} \rfloor$ tracks.
Let us now prove that $t_u(G) \leq \lceil \frac{m}{2} \rceil$. Similarly, the base case $m \leq 1$ is obvious since we may just add two overlapping and disjoint intervals on one track to realise one edge. Let us now assume that $m \geq 2$ and assume that a vertex u is adjacent to at least two vertices v and w. Otherwise, if every vertex has degree at most one, the edges of the graph form a matching and this can be realised on one track by pairs of overlapping intervals. By induction the subgraph of G with the edges (u, v) and (u, w) deleted can be realised on $\lceil \frac{m-2}{2} \rceil$ tracks with unit intervals. Let us now open a new track and represent the vertices u, v and w by three unit intervals U, V and W such that U intersects both V and W. We finally add a disjoint unit interval for each remaining vertex of the graph. We therefore obtain a unit representation of G on $\lceil \frac{m-2}{2} \rceil + 1 = \lceil \frac{m}{2} \rceil$ tracks. □

Similarly, it is known that $i(G) \leq \lceil \frac{n+1}{4} \rceil$ [19] and $i_u(G) \leq \lceil \frac{n-1}{2} \rceil$ [2], $i(G) \leq \lceil \frac{\sqrt{m}}{2} \rceil + 1$ [4] and $i_u(G) \leq \lceil \frac{m}{2} \rceil$ [2] , and $i(G) \leq i_u(G) \leq \lceil \frac{\Delta+1}{2} \rceil$ [20, 33].

Last but not least, we will also mention the class of boxicity d graph defined thereafter.

Definition 8. *A graph $G = (V, E)$ with $V = \{v_1, v_2, ..., v_n\}$ is a boxicity d graph if there exists a family of boxes of dimension d $\mathscr{B} = \{B_1, B_2, ..., B_n\}$ (box representation) associated to the vertices such that $B_i \cap B_j \neq \emptyset \Leftrightarrow (v_i, v_j) \in E$. In other words, a graph is a boxicity d graph if it has an intersection model consisting of boxes of dimension d.*

An example of a boxicity 2 graph is given below in Figure 6.

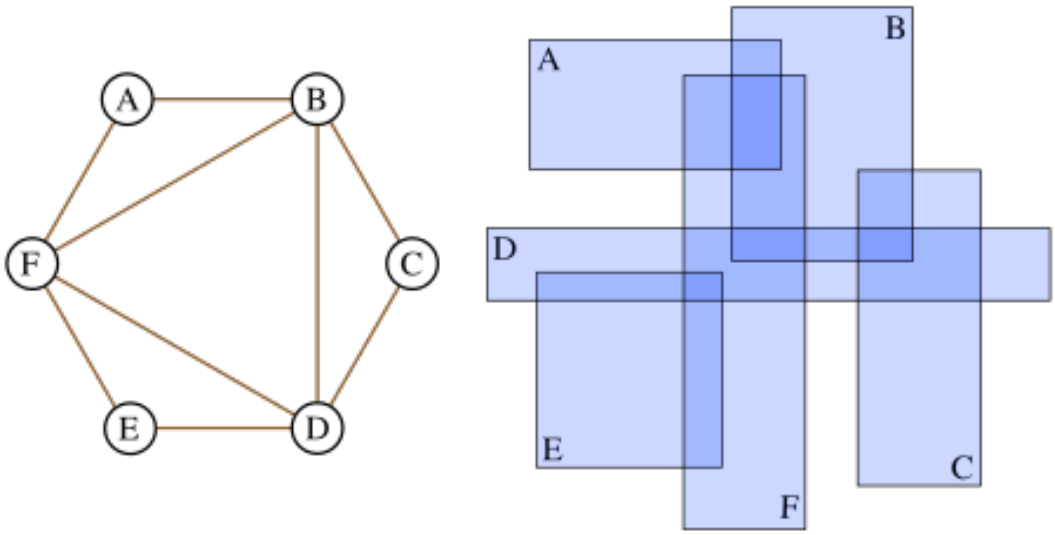

Figure 6: A boxicity 2 graph with a box representation of it

We can again notice that the graphs of boxicity 1 are the interval graphs and that for any integer d, the class of graphs of boxicity d is included in the class of graphs of boxicity $d + 1$ as we can project a dimension $d + 1$ representation onto one of dimension d. Finally, if a graph G has a box representation where the boxes are hypercubes (the boxes have unit length) of dimension d then we say that the resulting graph has cubicity d.

3 Inclusions between the multiple interval classes

In this section we will study the different inclusions between the introduced classes.

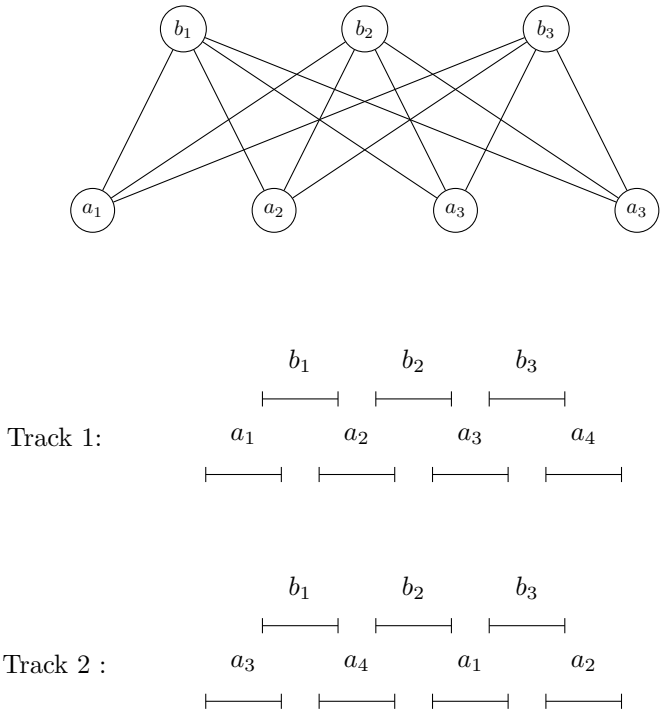

Figure 7: a 2-track representation of $K_{4,3}$

Property 1. *Any 2-track realisation of $K_{4,3}$ is contiguous on the two tracks*

Proof. By modifying the proof in [34], we can also introduce the definition of t-tight graphs on t tracks. As $K_{m,n}$ is triangle free, it can only be realized by depth 2 t-track representations. Therefore, when we read the representation from left to right, on each track, we obtain at most one new edge at the left endpoint of each interval. However, no new edge arises if the left endpoint of an interval is not included in another interval such as with the leftmost interval on each track. We can therefore realise at most $(m+n-1)$ edges on each track and therefore $|E| \leq (m+n-1)t$. We then get that we need at least $t \geq \frac{mn}{m+n-1}$ tracks to represent $K_{m,n}$. Therefore, when $\frac{mn}{m+n-1}$ is an integer t we say that $K_{m,n}$ is t-tight on t tracks. This then implies that any t-track realisation of a t-tight graph is contiguous on t tracks since an intersection of intervals must occur at the left endpoint of every interval after the first one. Finally, by setting $m = 4$ and $n = 3$, we have that $\frac{mn}{m+n-1} = \frac{12}{6} = 2$. Therefore $K_{4,3}$ is 2-tight on 2 tracks and a realisation of it is displayed in Figure 7. □

Let us now give a compact representation of $K_{4,3}$ that will be useful to prove the next Property. (Figure 8)

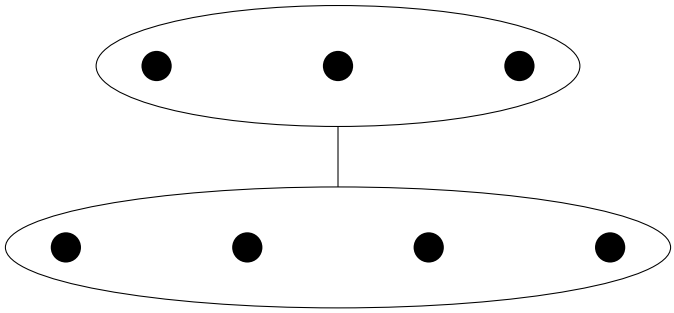

Figure 8: A compact representation of $K_{4,3}$

Property 2. *The class of balanced 2-track graphs is strictly included in the class of 2-track graphs*

Proof. The inclusion is trivial, any balanced 2-track representation is a 2-track representation. Let us now build a 2-track graph that is not balanced 2-track. For that, we will take the example of a 2-interval graph that is not balanced 2-interval constructed in [15] and make some modifications. We then consider a chain of five $K_{4,3}^{(i)}$ linked so that $a_4^{(i)}$ is adjacent to $a_1^{(i+1)}$ and $a_2^{(i)}$ is adjacent to $a_3^{(i+1)}$. This will therefore give a collection of intervals that overlap on both tracks. Finally, we add two vertices I^1, I^2 such that I^1 is adjacent to $a_4^{(1)}$, all the vertices of $K_{4,3}^{(2)}$, $a_1^{(3)}$ and $b_2^{(4)}$ and I^2 is adjacent to $b_2^{(2)}$, $a_4^{(3)}$, all the vertices of $K_{4,3}^{(4)}$ and $a_1^{(5)}$. This graph then has a 2-track representation but no balanced 2-track ones as illustrated in the following Figure 9.

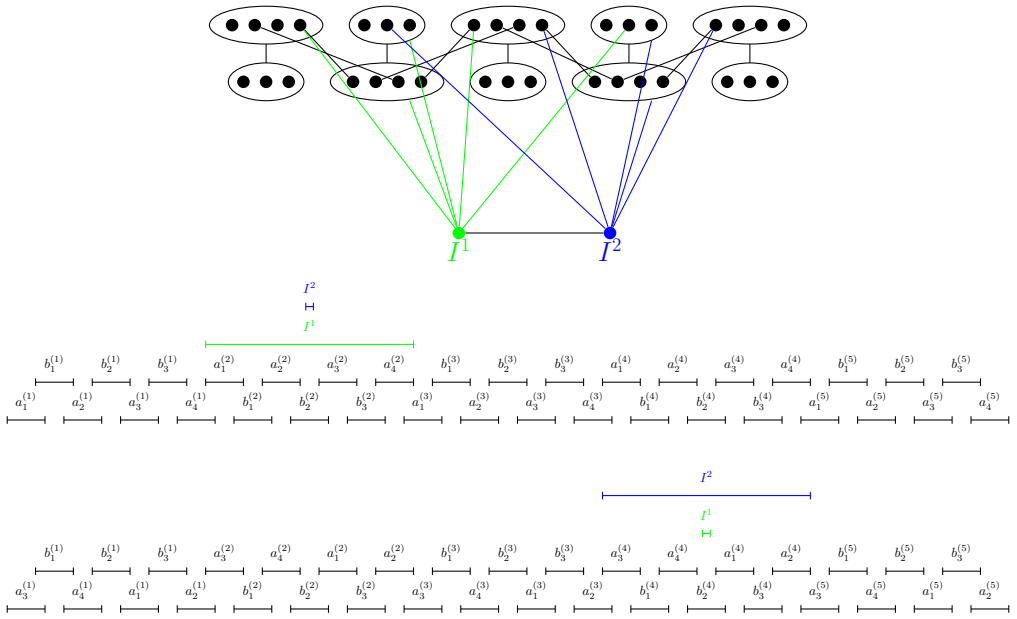

Figure 9: An example of a 2-track graph that is not balanced 2-track

By construction, on track 1, the length of I^1 needs to be strictly greater than the length of I^2 whereas, on track 2, the length of I^2 needs to be strictly greater than the length of I^1. Therefore, even if we set the length of the intervals from the $K_{4,3}^{(i)}$ so that both intervals of I^1 have the same length, it is not possible to do so for I^2.

□

Also notice that similar graphs can be built to prove the more general case that the class of balanced d-track graphs is strictly included in the class of d-track graphs. The following two properties also naturally hold for $d > 2$ tracks.

Property 3. *The class of unit 2-track graphs is strictly included in the class of balanced 2-track graphs.*

Proof. The inclusion is trivial, since a unit 2-track representation is a balanced 2-track representation. To show that the inclusion is strict, we can consider the graph $K_{1,5}$. It can easily be represented on two tracks such that the interval of the central vertex has the same length on both tracks. However, one of the tracks will be composed of the interval of the central vertex and three non intersecting intervals that intersect it, i.e., an interval representation of a claw. Therefore, $K_{1,5}$ cannot be a unit 2-track graph. □

By reusing the gadget $K_{4,4}-e$ introduced in [15] we can also prove the following property.

Property 4. *The class of unit 2-track graphs is strictly included in the class of unit 2-interval graphs. The class of balanced 2-track graphs is strictly included in the class of balanced 2-interval graphs.*

Proof. The two inclusions are trivial since we can put one track after the other to obtain a 2-interval representation. Let us consider the graph $K_{4,4} - e$ that is a $K_{4,4}$ with one edge deleted. Since $K_{4,4}$ is triangle-free, it is only representable by a t-track representation of depth 2. However, in this case we have $E \leq (m + n - 1)t$ since each interval can at most intersect one interval on its left endpoint except for the first interval. As $K_{4,4} - e$ has 15 edges then $t \geq \frac{15}{7} > 2$ and thus we need at least 3 tracks to represent it. Therefore $K_{4,4}-e$ is not a 2-track graphs although it is a unit 2-interval graph and thus also a balanced 2-interval graph as shown in [15]. □

Let us also introduce the class of linear arboricity ≤ 2 graphs that are the graphs that can be decomposed into at most 2 forests whose connected components are paths.

This class contains the class of maximum degree 3 graphs [22] and is contained in the class of maximum degree 4 graphs and in the class of unit 2-track graphs. As a matter of fact, we can prove a stronger statement :

Property 5. *Linear arboricity $\leq$ 2 graphs are exactly the depth 2 unit 2-track graphs*

Proof. Suppose a graph has linear arboricity ≤ 2, then it can be decomposed into at most 2 forests of paths. On the first track, we realise one of the two forests by adding a unit interval for each vertex of the forest and we realise the paths by a succession of overlapping intervals. On the second track, we then realise the other forest in the same way. The representation obtained is then of depth 2, since paths are triangle free. Now suppose we are given a depth 2 unit 2-track representation. Then, on each track, since the representation is unit, the representation is proper and thus no interval is included in an other and therefore an interval only intersects other intervals that contain one of its extremities. Furthermore, as

the representation is of depth 2, the extremity of each interval can only be contained in one other interval. We thus have that any interval can only intersect two other intervals, it can therefore only form a succession of overlapping intervals. Each contiguous succession of intervals then represents a path so each track realises a forest of paths. As there are two tracks, the graph is of linear arboricity ≤ 2.

□

We can then compile the following inclusions in Figure 10.

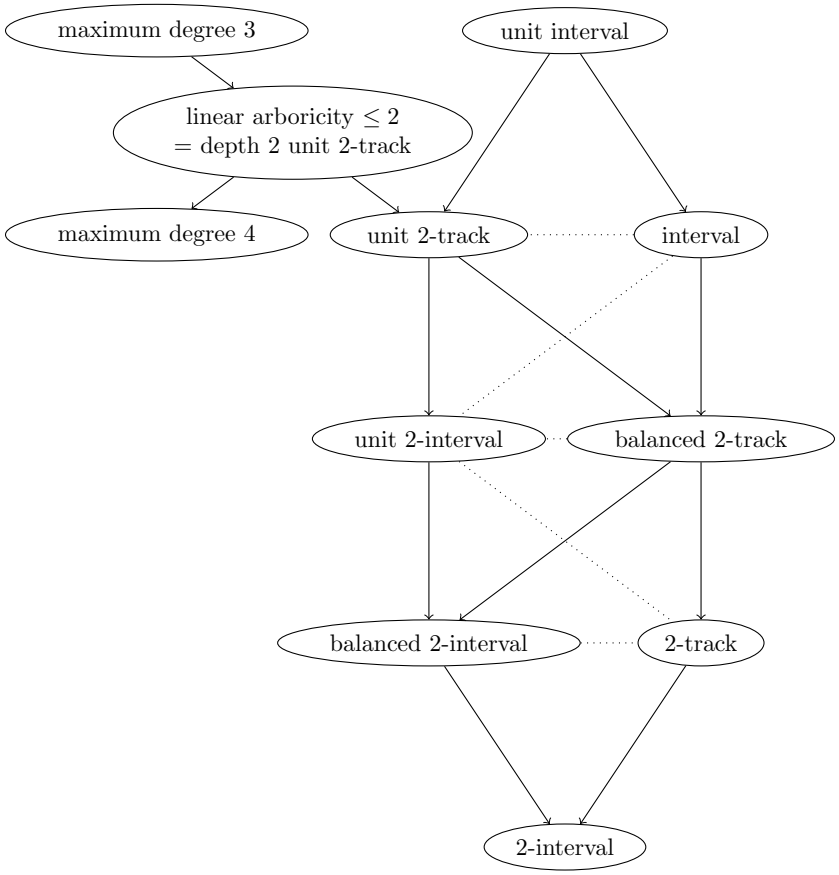

Figure 10: Map of inclusions

Let us note that unit 2-track graphs and interval graphs are not related since C_4 is unit 2-track but is not interval but $K_{1,5}$ is interval but is not unit 2-track. Similarly, unit 2-interval graphs and interval graphs are not related. We also cannot include unit 2-interval graphs in (balanced) 2-track graphs (or the other way around) since $K_{4,4} - e$ is a unit 2-interval graphs but is not a 2-track graph and $K_{1,5}$ is a balanced 2-track graph but is not a unit 2-interval graph. Finally, balanced 2-interval graphs and 2-track graphs are not related as we can notice that the graph built in Property 2 is not balanced 2-interval.

4 Bound on the unit track number

Let us now see how to modify an interval representation of a graph G to obtain a unit $\psi(G)$-track representation where $\psi(G)$ is the claw-number of G. Furthermore, we will give a bound on the unit track number of any graph given its track number. We define the track number $t(G)$ of a graph G as being the smallest integer d such that G is d-track. Similarly, we define the unit track number $t_u(G)$ of a graph G as being the smallest integer d such that G is unit d-track. Finally, for any graph G, we define the claw number $\psi(G)$ as being the maximum n such that $K_{1,n}$ is an induced subgraph of G.

We will then give an upper bound on the unit track number with respect to the track number. For that, we will first introduce an algorithm to transform an interval representation of a $K_{1,d+1}$-free graph into a unit d-track representation.

```
1: procedure MaxIndependentSet(𝒥)
2:     Let S = ∅
3:     while 𝒥 ≠ ∅ do
4:         Select I as the interval that ends the first in the representation 𝒥
5:         Put I in S
6:         Remove all intervals in the representation 𝒥 that intersect I
7:     return S
```

We will denote by S_i the elements of S.

Algorithm 1

1: Let S = MaxIndependentSet($\mathscr{I}$)
2: Open $|S|$ tracks
3: Put S_i and S_{i+1} on the track T_i
4: Put all the intervals that intersect S_i or S_{i+1} (or both) on T_i
5: Remove all the intervals from T_i that end in S_i or that end after S_i but do not intersect an interval disjoint from S_i
6: **for** $0 \leq i < \psi(G)$ **do**
7: Put all tracks T_j such that $j \equiv i \pmod{\psi(G)}$ one after the other
8: Add, on each track, a disjoint interval for each vertex that is not represented on the track

Lemma 4. *Let S be the set of intervals returned by the procedure MaxIndependentSet($\mathscr{I}$). Then the vertices associated to the intervals of S form a maximum independent set of G.*

Proof. First of all, let us show that all the intervals of S are disjoint. The procedure first selects the interval I of the representation that has the smallest right endpoint. It then removes all the intervals that intersect I and then repeats the operation. After removal, the representation only consists of intervals that are disjoint from I. Therefore the next selected interval will also be disjoint from I. Inductively, all selected intervals in S are disjoint. Now let us show that S is maximal. Suppose it is not and that S' is a maximal independent set. Let $S = \{S_1, S_2, ..., S_n\}$ and $S' = \{S'_1, S'_2, ..., S'_{n+1}\}$. First notice that S'_1 cannot end before S_1 and S'_{n+1} cannot start after the end of S_n otherwise they would have been selected by the procedure. As S and S' are different then there must be a first interval S'_k which is not

in S and we can suppose that all the intervals S'_j, $j < k$ coincide with the first ones of S. As S'_k is not in S then S'_k must end after the end of S_k. However if we select S_k instead of S'_k then $\{S_1, .., S_k, S'_{k+1}, .., S'_{n+1}\}$ would also be a maximal independent set. We can therefore replace all the intervals of S' by the ones of S. It is therefore not possible that S' is bigger than S. □

As the set of intervals represents a maximum independent set then we have that any interval in the representation intersects at least one interval of S.

Lemma 5. *Algorithm 1 returns a valid $\psi(G)$-track representation.*

Proof. To prove that the representation is indeed valid we need to make sure that Algorithm 1 never puts twice the same interval on a track. It is easy to see that before merging the tracks it only puts an interval once. Let us now prove that when we merge two tracks $T^{(i)}$ and $T^{(\psi(G)+i)}$ there was no interval in common and therefore no interval is duplicated. Suppose there was an interval I on both tracks. Then since I is on $T^{(i)}$ and on $T^{(\psi(G)+i)}$, it must at least intersect $S^{(i+1)}$ and $S^{(\psi(G)+i)}$. Furthermore, I must end after $S^{(\psi(G)+i)}$ and intersect another interval J disjoint from $S^{(\psi(G)+i)}$. However in this case I intersects $\psi(G)+1$ disjoint intervals in the original representation : $S^{(i+1)}$, .. , $S^{(\psi(G)+i)}$ and J. This is impossible as this would form a $K_{1,\psi(G)+1}$. Therefore the final representation is indeed a valid $\psi(G)$-track representation. □

Lemma 6. *Algorithm 1 returns a $\psi(G)$-track representation of G.*

Proof. What we need to prove is that, for each intersection of an interval I and an interval J in the original interval representation, there is a track on which the intervals I and J intersect. As S is a maximal independent set and the graph G has claw number $\psi(G)$, then I and J must intersect at least one interval of S and at most $\psi(G)$ intervals of S. Suppose both of them intersect the same interval S_k. Then I and J intersect on the track T_{k-1} as this track contains S_k and all the intervals that intersect S_k. So let us suppose I and J do not intersect any same interval of S. In this case neither I nor J is included in the other. So let us suppose I begins before J. As they are intersecting there should be an interval S_k and an interval S_{k+1} in S such that I intersects S_k and J intersects S_{k+1}. Therefore the intersection between I and J is realised on the track T_k since it contains S_k, S_{k+1} and all the intervals that intersect one of them or both except the ones that end in S_k or the ones that ends after S_k but does not intersect a disjoint interval from S_k. However I ends after S_k since it intersects J and J is disjoint from S_k. Therefore I and J should be intersecting on T_k. □

Theorem 7. *The $\psi(G)$-track representation of G returned by Algorithm 1 is unitary.*

Proof. We should now prove that, on each track, the interval representation does not contain any interval that intersects 3 disjoint intervals (as this would form a claw). First notice that when we merge two tracks together, we do not create any intersection so we just need to check that before merging the tracks, none of them contained any interval that intersects 3 disjoint intervals. It is indeed the case as on each track we only put two disjoint intervals S_k and S_{k+1} and some intervals that intersect them. Furthermore we do not add the intervals that end in S_k. Therefore if there is an interval that intersects 3 disjoint intervals then we can assume without loss of generality that the leftmost one is S_k and the two others are intersecting S_{k+1} (there cannot be a disjoint interval between S_k and S_{k+1}). However the

central interval should then be disjoint from S_k and ends before the end of S_{k+1}, which is impossible otherwise we would have selected this interval as S_{k+1}. □

Let us now use this algorithm to obtain an upper bound on the unit track number of any graph.

Lemma 8. *Let G be a graph and let $G_1, G_2, \ldots, G_j$ be graphs such that $V(G) = V(G_p)$ for $1 \leq p \leq j$ and $E(G) = E(G_1) \cup E(G_2) \cup \ldots \cup E(G_j)$. Then $t_u(G) \leq t_u(G_1) + t_u(G_2) + \ldots + t_u(G_j)$*

Proof. Realise each G_i on $t_u(G_i)$ tracks and put all the tracks in parallel. This indeed gives a unit $t_u(G_1) + t_u(G_2) + \ldots + t_u(G_j)$ track representation of G therefore $t_u(G) \leq t_u(G_1) + t_u(G_2) + \ldots + t_u(G_j)$ □

Lemma 9. *Let $r(\psi)$ denote the largest real number such that there exists a graph G with claw number ψ such that $t_u(G) = r(\psi)t(G)$. Then, there exists an interval graph G' such that $t_u(G') = r(\psi)$*

Proof. Let G be a graph with claw number ψ such that $t(G) = b$ and $t_u(G) = b \cdot r(\psi)$. Then, there exist interval graphs $G_1, G_2, \ldots, G_b$ such that $V(G_i) = V(G)$ for $1 \leq i \leq b$ and $E(G) = E(G_1) \cup E(G_2) \cup \ldots \cup E(G_b)$. By Lemma 8, $r(\psi) \cdot b = t_u(G) \leq \sum_{i=1}^{b} t_u(G_i)$. It follows that there exists at least one i, $(1 \leq i \leq b)$ such that $t_u(G_i) \geq r(\psi)$. Recalling that G_i is a interval graph and thus $t(G_i) = 1$ we have $t_u(G_i) \geq r(\psi) \cdot t(G_i)$. From the definition of $r(\psi)$, it follows that $t_u(G_i) = r(\psi) \cdot t(G_i) = r(\psi)$, as required. □

Theorem 10. *For any graph G of claw number ψ, $t_u(G) \leq \psi t(G)$*

Proof. Let G be a graph of claw number ψ, then according to lemma 9, there exists an interval graph G' such that $t_u(G') = r(\psi)$. However by running Algorithm 1 on G', we obtain that $t_u(G') \leq \psi$ and therefore $r(\psi) \leq \psi$. We thus get that $t_u(G) = r(\psi)t(G) \leq \psi t(G)$ □

5 Problems on these classes

As already introduced, most problems that are NP-hard on general graphs remain NP-hard on d-track and t-interval (for $d \geq 2$ and $t \geq 2$) graphs. In the table below, we sum up some of these results. Here "P" means that the problem is in P, "NP" that the problem has been shown to be NP-hard and "GI" that it is GI-hard. When the problem is NP-hard and that the parameterized complexity of it has been studied for these classes of graphs then we denote by "FPT" the fact that the problem is FPT and by "W[1]" that it is W[1]-hard.

	Interval		2-track			2-interval		
	Unit	Unres.	Unit	Balanced	Unres.	Unit	Balanced	Unres.
Recognition	P	P	NP	NP	NP	?	NP	NP
3-Colourability	P	P	NP	NP	NP	NP	NP	NP
Clique	P	P	P	P	P	FPT	FPT	FPT
Clique Cover	P	P	NP	NP	NP	NP	NP	NP
Colourability	P	P	NP	NP	NP	NP	NP	NP
Domination	P	P	W[1]	W[1]	W[1]	W[1]	W[1]	W[1]
Feedback Vertex Set	P	P	NP	NP	NP	NP	NP	NP
Hamiltonian Cycle	P	P	NP	NP	NP	NP	NP	NP
Hamiltonian Path	P	P	NP	NP	NP	NP	NP	NP
Independent Dominating Set	P	P	W[1]	W[1]	W[1]	W[1]	W[1]	W[1]
Independent Set	P	P	W[1]	W[1]	W[1]	W[1]	W[1]	W[1]
Maximum Cut	?	NP	NP	NP	NP	NP	NP	NP
Monopolarity	P	P	NP	NP	NP	NP	NP	NP
Polarity	P	P	NP	NP	NP	NP	NP	NP
Graph Isomorphism	P	P	?	GI	GI	GI	GI	GI
Vertex Cover	P	P	NP	NP	NP	NP	NP	NP

Figure 11: Table summing up the complexity of some problems on the classes of interval, 2-track and 2-interval graphs

5.1 Vertex Cover

Introduced in [24], the Vertex Cover problem is one of the 21 Karp's problems that have been proven to be NP-complete on general graphs. We give the definition of Vertex Cover below :

Vertex Cover
Input: An undirected graph $G = (V, E)$, a positive integer $K \leq \|V\|$
Question: Is there a vertex cover of size K or less, that is, a subset $V' \subseteq V$ such that $\|V'\| \leq K$ and, for each edge $(u, v) \in E$, at least one of u and v belongs to V' ?

Let us now show that Vertex Cover remains NP-complete on unit 2-track graphs (and thus on (unit) d-track and (unit) d-interval graphs for any $d \geq 2$).

Note, however, that this result can also be inferred as VERTEX COVER is already NP-complete on graphs of maximum degree 3 [16].

Theorem 11. VERTEX COVER *is NP-complete on depth 2 unit 2-track graphs*

Proof. As VERTEX COVER is in NP for general graphs then we also have that VERTEX COVER is in NP for depth 2 unit 2-track graphs. Let us now prove that VERTEX COVER is NP-hard on depth 2 unit 2-track graphs. To show it, we will consider 3-SAT with at most 3 occurrences per variable and 2 or 3 variables per clauses (that has been shown NP-complete in [30]) to VERTEX COVER. Note that we can assume without loss of generality that in this case each literal can appear at most twice (otherwise we can set the variable linked to the literal to the value that will satisfy the three clauses in each the literal appears) Let us consider an instance of this problem with variable set $X = \{x_1, x_2, .., x_n\}$ and clause set $C = \{c_1, c_2, .., c_m\}$ where $c_i = \{l_i^{(1)}, l_i^{(2)}, l_i^{(3)}\}$ if c_i is a 3-clause or $c_i = \{l_i^{(1)}, l_i^{(2)}\}$ if c_i is a 2-clause.

We will then revisit the classical reduction from 3-SAT to VERTEX COVER in [16] by introducing a new gadget for the 2-clauses in our variation of 3-SAT.

For each variable $x_i \in X$, we build the following truth-setting component $T_i = (V_i, E_i)$ where $V_i = \{x_i, \overline{x_i}\}$ and $E_i = \{(x_i, \overline{x_i})\}$. (two vertices for x_i and its complement joined by an edge) (see Figure 12)

Figure 12: Gadget T_i for each variable

For each 3-clause $c_j \in C$, we build the following satisfaction testing component $S_j = (V'_j, E'_j)$ where $V'_j = \{l_j^{(1)}, l_j^{(2)}, l_j^{(3)}\}$ and $E'_j = \{(l_j^{(1)}, l_j^{(2)}), (l_j^{(1)}, l_j^{(3)}), (l_j^{(2)}, l_j^{(3)})\}$ (a triangle of the three vertices for the literals) (See Figure 13)

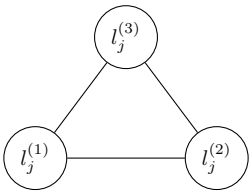

Figure 13: Gadget S_j for each 3-clause

Finally for each 2-clause $c_j \in C$, we build the following satisfaction testing component $S'_j = (V'_j, E'_j)$ where $V'_j = \{l_j^{(1)}, l_j^{(2)}, u_j, v_j\}$ and $E'_j = \{(l_j^{(1)}, l_j^{(2)}), (l_j^{(1)}, u_j, (l_j^{(2)}, u_j), (u_j, v_j)\}$ (a triangle of the three vertices for the 2 literals and the dummy vertex u_j and a pendant vertex v_j adjacent to u_j) (see Figure 14)

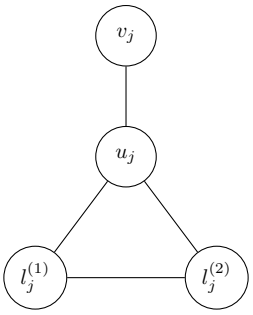

Figure 14: Gadget S'_j for each 2-clause

Furthermore, for each clause, we also build the communicating edges E''_j between the vertices of V'_j representing the literals and the vertices of V_i representing the variables such that an edge between a literal and a variable is drawn if they are equal. In other words, let us assume, for example that $l_j^{(1)} = x_k$, $l_j^{(2)} = \overline{x_l}$ and $l_j^{(3)} = \overline{x_m}$ for some $1 \leq k \leq n, 1 \leq l \leq n, 1 \leq m \leq n$ then $E''_j = \{(l_j^{(1)}, x_k), (l_j^{(2)}, \overline{x_l}), (l_j^{(3)}, \overline{x_m})\}$.

For example, let $X = \{x_1, x_2, x_3, x_4\}$ and $C = \{c_1, c_2\}$ such that $c_1 = (x_1, x_2, \overline{x_3})$ and $c_2 = (\overline{x_1}, \overline{x_3}, \overline{x_4})$. We then build the following graph :

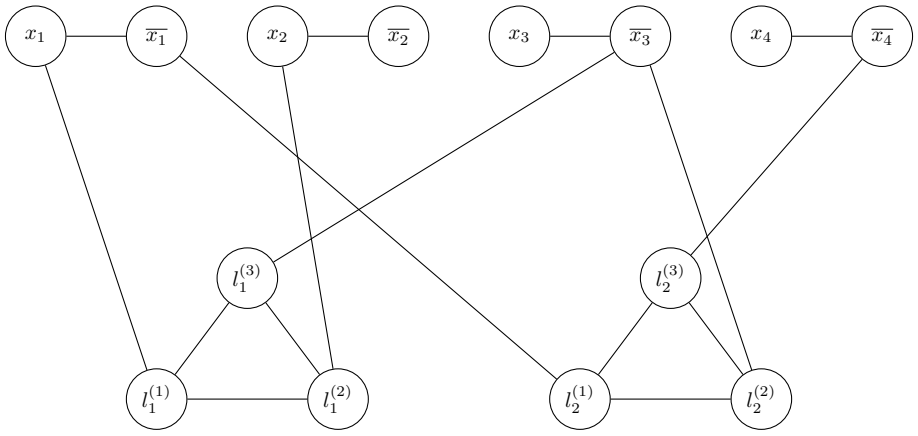

Figure 15: Example of construction of the full gadget

Let us introduce another example. Let $X = \{x_1, x_2, x_3\}$ and $C = \{c_1, c_2\}$ such that $c_1 = (x_1, x_2, \overline{x_3})$ and $c_2 = (\overline{x_1}, \overline{x_3})$. We then build the following graph :

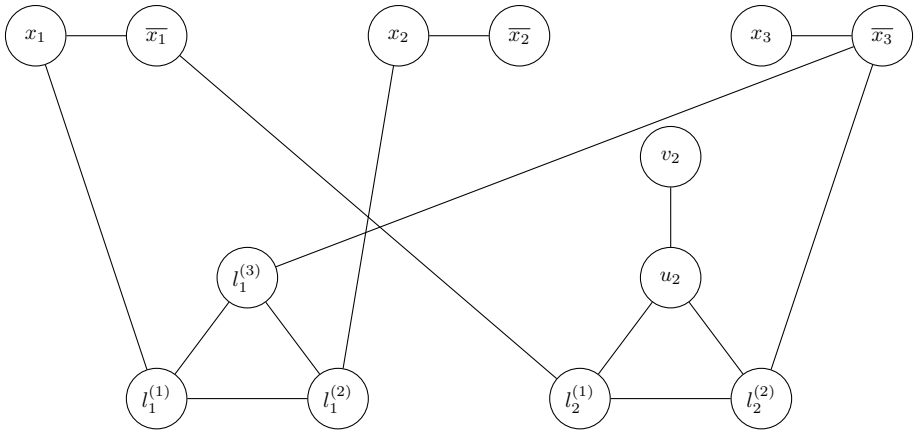

Figure 16: Example of construction of the full gadget

Finally, the construction of the instance of Vertex Cover is completed by setting $K = n+2m$ and $G = (V, E)$ where $V = (\bigcup_{i=1}^{n} V_i) \cup (\bigcup_{j=1}^{m} V'_j)$ and $E = (\bigcup_{i=1}^{n} E_i) \cup (\bigcup_{j=1}^{m} E'_j) \cup (\bigcup_{j=1}^{m} E''_j)$. It is easy to see that the construction can be realised in polynomial time.

Let us now show that C is satisfiable if and only if G has a vertex cover of size K or less. Suppose $V' \subseteq V$ is a vertex cover of G with $|V'| \leq K$. By construction V' must contain at least one vertex from each T_i to cover the single edge in E_i and at least two vertices from each S_j to cover the three edges in E'_j. As this implies that $|V'| \geq n + 2m = K$ then V' contains exactly one vertex from each T_i and two vertices of each S_j. As it has exactly either x_i or $\overline{x_i}$ then we obtain a truth assignment $\tau : X \to \{T, F\}$ such that $\tau(x_i) = T$ if $x_i \in V'$ and $\tau(x_i) = F$ if $\overline{x_i} \in V'$. As V' has two vertices from each S_j (the gadget of the 3-clauses), it covers the three edges in E'_j and at most two edges of E''_j. We should then show that the last edge of each E''_j is covered by a vertex of some V_i that belongs to V'. But that therefore implies that the corresponding literal x_i or $\overline{x_i}$ is true under τ and satisfies the clause c_j. Similarly, as V' has two vertices from each S'_j (the gadget of the 2-clauses), one of the two vertices must be u_j (otherwise it is not possible to cover all the edges in E'_j) and the other vertex must either be $l_j^{(1)}$ or $l_j^{(2)}$ to cover the edge $(l_j^{(1)}, l_j^{(2)})$. However, the other vertex is not in the vertex cover and so its corresponding communicating edge must therefore be covered by a vertex of some V_i that belongs to V'. So again, that therefore implies that the corresponding literal x_i or $\overline{x_i}$ is true under τ and satisfies the clause c_j. As this holds for every clause, τ is a satisfying truth assignment for C.

Conversely, let $\tau : X \to \{T, F\}$ be a truth assignment for C. We can build a vertex cover V' that contains one vertex from each T_i, two vertices from each S_j and two vertices from each S'_j. For each T_i, we include x_i if $\tau(x_i) = T$ or $\overline{x_i}$ otherwise. This ensures that at least one of the two or three edges from each set E''_j is covered since τ satisfies every clause. For each S_j, we then only need to include the endpoints of the two other edges in E''_j. For each S'_j, we include u_j and the endpoint of the other edge in E''_j. This will then give a vertex cover of size K.

Let us now show that the graph G is a unit 2-track graph of depth 2. For each truth setting component T_i, we represent x_i and $\overline{x_i}$ by two unit intervals X_i and $\overline{X_i}$. Let us put all the pairs X_i and $\overline{X_i}$ on the track 1 and on the track 2. If both x_i and $\overline{x_i}$ are adjacent to

at most one vertex $l_j^{(2)}$ for some j (i.e. the corresponding variable appears as a second literal for some clause at most once) then let us make the right endpoint of X_i intersect the left endpoint of $\overline{X_i}$ on track 1 and be disjoint on track 2. Otherwise let us make them be disjoint on track 1 but intersect in the same way on track 2. (if one of them is a second literal of some clause twice, say x_i, then x_i cannot be in any other clause and $\overline{x_i}$ can be in at most one other clause so we can make them intersect on track 2). For each satisfaction testing component S_j (resp. S'_j), we represent $l_j^{(1)}, l_j^{(2)}$ and $l_j^{(3)}$ (resp. u_j) by three unit intervals $L_j^{(1)}, L_j^{(2)}$ and $L_j^{(3)}$ (resp. U_j) on track 1 such that the right endpoint of $L_j^{(1)}$ intersects the left endpoint of $L_j^{(3)}$ (resp. U_j) and the right endpoint of $L_j^{(3)}$ (resp. U_j) intersects the left endpoint of $L_j^{(2)}$. Furthermore, let us suppose that the vertex $l_j^{(2)}$ is adjacent to x_i with interval X_i. Then we make the right endpoint of $L_j^{(2)}$ intersect the left endpoint of X_i. However, as a literal may appear twice then if the left endpoint of X_i is already intersecting some $L_{j'}^{(2)}$ then we can revert the intervals $L_j^{(1)}, L_j^{(2)}$ and $L_j^{(3)}$ (resp. U_j) and make the left endpoint of $L_j^{(2)}$ intersect the right endpoint of X_i instead. Similarly, on track 2, we put $L_j^{(1)}$ and $L_j^{(2)}$ such that the right endpoint of $L_j^{(1)}$ intersects the left endpoint of $L_j^{(2)}$. Let us now suppose that the vertex $l_j^{(1)}$ is adjacent to $x_{i'}$ with interval $X_{i'}$. Then we make the left endpoint of $L_j^{(1)}$ intersect the right endpoint of $X_{i'}$. Again, if needed we can revert the intervals. Finally, on track 2, let us assume that $l_j^{(3)} = x_{i''}$ (in the case of a 3-clause) with interval $X_{i''}$ we add $L_j^{(3)}$ (resp. U_j) so that its right endpoint intersect the left endpoint of $X_{i''}$ (resp. V_j that represents v_j). If needed again, the left endpoint of $L_j^{(3)}$ may intersect the right endpoint of $X_{i''}$ instead. We then add an other disjoint interval for each other vertex not represented on one of the tracks to make the representation valid.

By construction, each interval intersects at most two other disjoint intervals, always at their extremities only and always at most one interval for each extremity is intersected. The construction is therefore of depth 2. Furthermore, the maximum degree of the graph is 3 since the vertices of the truth setting components V_i have degree at most 3 (the other vertex of V_i and at most two other vertices from V'_j since a literal appears at most twice) and the vertices of the satisfaction testing components V'_j have degree at most 3 (the two others vertices in the triangle they form and either a vertex from V_i or the pendant vertex v_j)

(We can also notice that this implies that all intervals are displayed)

For example, let us consider the same example $X = \{x_1, x_2, x_3, x_4\}$ and $C = \{c_1, c_2\}$ such that $c_1 = (x_1, x_2, \overline{x_3})$ and $c_2 = (\overline{x_1}, \overline{x_3}, \overline{x_4})$ whose gadget graph is displayed on figure 16. Let us then build the corresponding depth 2 unit 2-track representation of this graph :

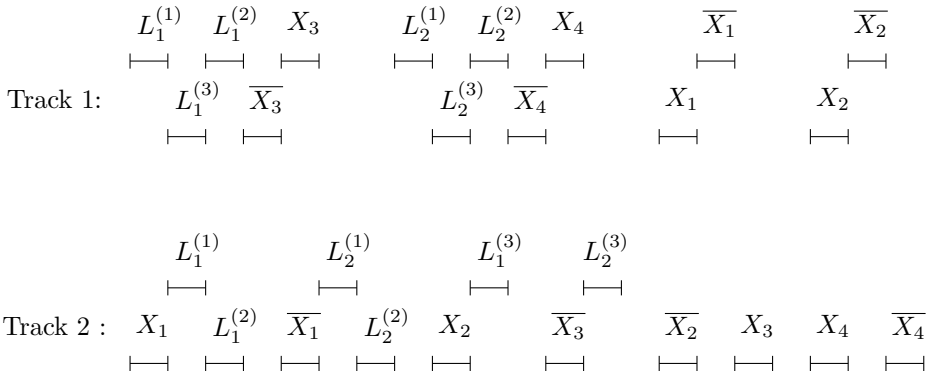

Figure 17: Depth 2 unit 2-track representation of the graph given in the first example

□

5.2 Feedback Vertex Set

Let us now consider the Feedback Vertex Set problem. This also is one of the 21 Karp's problems that have been proved to be NP-complete on general graphs. We give the definition of Feedback Vertex Set below :

Feedback Vertex Set
Input: An undirected graph $G = (V, E)$, a positive integer $K \leq \|V\|$
Question: Is there a subset $V' \subseteq V$ with $\|V'\| \leq K$ such that V' contains at least one vertex from every cycle in G ?

In [27], it was shown that Feedback Vertex Set was APX-complete on graphs with maximum degree 4 by reducing Vertex Cover in cubic graphs to Feedback Vertex Set with maximum degree 6 and then introducing some gadgets to reduce the maximum degree down to 4. We will actually show that the resulting graph in their proof is even a depth 2 unit 2-track graph (linear arboricity ≤ 2) which is strictly contained in the class of maximum degree 4 graphs (K_5 for example has maximum degree 4 but is not depth 2 unit 2-track)

Theorem 12. Feedback Vertex Set *is APX-complete on depth 2 unit 2-track graphs*

Proof. Let us recall the usual reduction of Vertex Cover to Feedback Vertex Set in [24]. Suppose we have a graph $G = (V, E)$. Let $G' = (V', E')$ be a copy of G but for each edge $(u, v) \in E$ we add a vertex $w_{uv} \in V'$ and two edges so that $(u, w_{uv}) \in E'$ and $(v, w_{uv}) \in E'$. (See Figure 18)

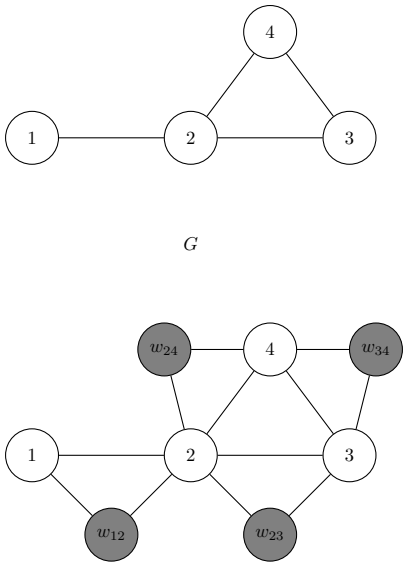

Figure 18: An example graph G with the corresponding transformation G'

This reduction clearly takes polynomial time.

As Vertex Cover in cubic graphs is already APX-hard it implies that Feedback Vertex Set with maximum degree 6 is APX-hard too as argued in [27]. We will now reuse their gadgets to reduce the maximum degree of the graph to 4 and call G'_4 the resulting graph.

Let us first consider the gadget to replace every degree 6 vertex of the graph to obtain a graph of maximum degree 5:

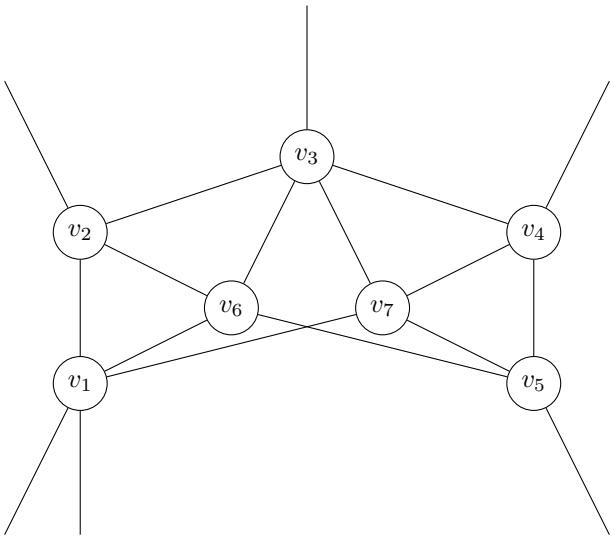

Figure 19: Gadget to reduce degree 6 vertices to degree 5

It can be seen that the original graph has a FVS K of size t if and only if the resulting graph with the degree 6 vertices replaced by the above gadget has a FVS K' of size $t + 2n_6$ where n_6 is the number of nodes of degree 6. Let us suppose u is a degree 6 vertex, if u is in a FVS K then we can add v_1, v_3 and v_5 in K' as they intersect all the internal cycles but also all the other cycles that visit a vertex of the gadget. If now, u is not in a FVS then we can add v_6 and v_7 in K' to delete all the internal cycles. Conversely, we can argue that the smallest possible FVS K' needs to contain either v_6 and v_7 or v_1, v_3 and v_5 in which case we can build easily a FVS K for the original graph. From the graph it is clear that we cannot pick only one vertex to delete all the internal cycles. If we only pick two then it can only be v_6 and v_7 but that only deletes the internal cycles but none of the external ones, this will thus correspond to not picking the replaced vertex in G. If now, K' is minimal it contains at most 3 vertices since it is enough to delete all the internal cycles and the ones that visit a vertex of the gadget : this corresponds to picking the replaced vertex in G. Therefore we always obtain a FVS of size $K = K' - 2n_6$

Let us now give a unit 2-track representation of this gadget that we decompose into several blocks :

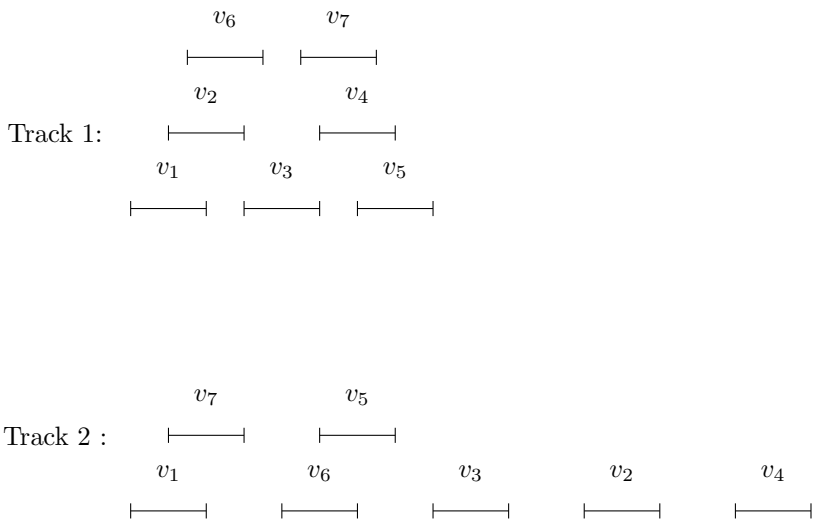

Figure 20

Let us then introduce the gadget used in [27] to reduce the maximum degree of the graph from 5 to 4. For that we will replace all the degree 5 vertices of the graph (each v_1 and v_3) by the following graph :

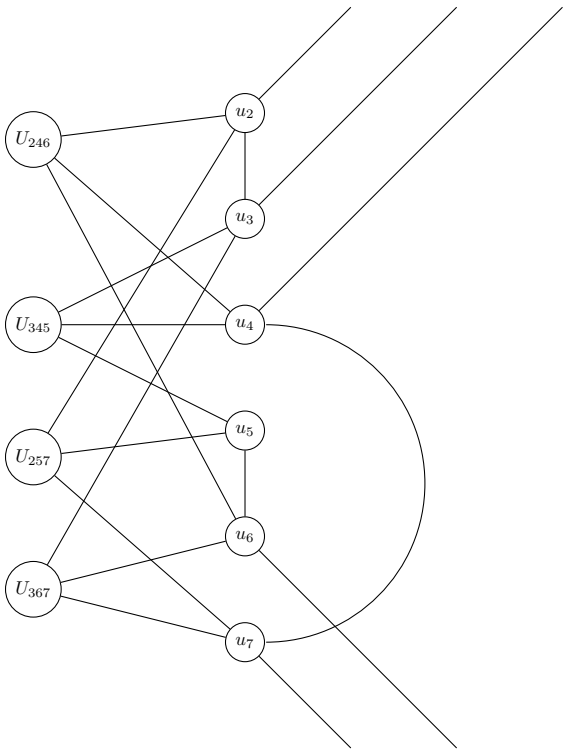

Figure 21: Gadget to reduce degree 5 vertices

It can then be seen that the original graph contains a FVS K of size t if and only if the resulting graph with the vertices of degree 5 replaced by the above gadget has a FVS K' of size $t+3n_5$ where n_5 is the number of degree 5 vertices. This comes from the fact that, if a vertex v is in K then we can add v_2, v_3, v_4 and v_7 in K' since they intersect all the internal cycles but also each cycle that visits a node of the gadget. If u is not in K' then we can add U_{345}, U_{257} and U_{367} as they intersect all the internal cycles. Conversely, we can also notice that K' needs to have at least these three vertices for each gadget and at most four and thus build K from K'.

Let us now give a unit 2-track representation of this gadget that we decompose into several blocks :

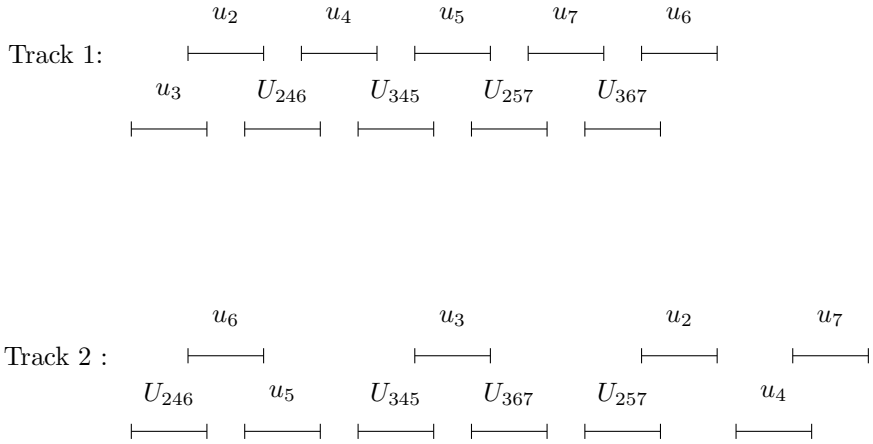

Figure 22

Let us now first replace v_1 and v_3 from the first gadget by the second gadget to obtain a full gadget to replace any vertex of degree 6 and get a graph G'_4 of maximum degree 4. We will denote $u_k^{(1)}$ the vertices of the gadget that replaces v_1 and $u_k^{(3)}$ the vertices of the gadget that replaces v_3. For that we replace v_3 by the second gadget and we connect $u_2^{(3)}$ to v_4, $u_3^{(3)}$ to v_6, $u_4^{(3)}$ to v_2 and $u_6^{(3)}$ to v_7. Furthermore, we replace v_1 by the second gadget and we connect $u_2^{(1)}$ to v_6, $u_6^{(1)}$ to v_2 and $u_7^{(1)}$ to v_7.

Let us then now show that we can represent the resulting graph on two unit tracks. Let us first give a unit 2-track representation of the full gadget :

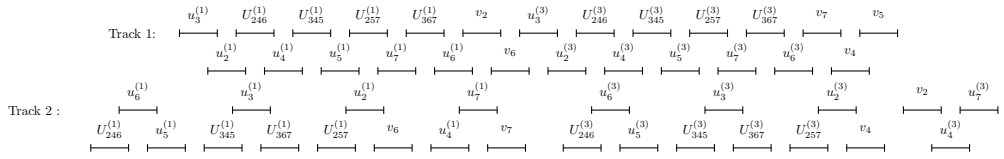

Figure 23

Let us now build a unit 2-track representation of the full graph G'_4 with maximum degree 4. Let us note that, originally, the graph G is cubic and therefore unit 2-track and that G' has degree 6 because for each edge we added a vertex w_{uv}. Let us first omit the vertices

w_{uv} and thus first focus on realising G and the gadgets. As G is cubic it has a unit 2-track representation of depth 2. Therefore, in any unit 2-track representation of G, if an interval intersects two other intervals on one track, it intersects exactly one on the other. Let us consider a unit 2-track representation of depth 2 of G and let us replace every interval I on the first track by the intervals of the full gadget on the track 1 so that $u_3^{(1)}$ always intersect an interval (we take the symmetry of the path of intervals of the gadget if we replace an interval I that did not intersect any other on its left extremity). Therefore, if I used to intersect two intervals on the first track then I only intersects one interval on the second track, we then replace I by the block of intervals $u_4^{(1)}$, $u_7^{(1)}$ and v_7 such that $u_4^{(1)}$ intersects the interval I used the intersect (we take the symmetry if I used to intersect an interval on its right extremity). Now, if I used to intersect only one interval on the first track then I intersects two intervals on the second track, we then replace I again by the block of intervals $u_4^{(1)}$, $u_7^{(1)}$ so that $u_4^{(1)}$ intersects one of the intervals I used the intersect and by the block v_5, v_6 so that v_5 intersects the other interval I used to intersect since the extremity of v_5 is free on track 1 (as I used to intersect only one interval on the first track). This therefore gives us a representation of G'_4 with the w_{uv} omitted. We can now use the intervals v_2, $u_7^{(3)}$ and v_4 for each variable to realise the edges with the vertices w_{uv}.

For that we will first go through each interval I of the representation of G from left to right from the first track to the second.

If I intersects two intervals on the first track, then we add the block of intervals v_2 , $u_4^{(3)}$ and $u_7^{(3)}$ for I on the second track. If now I only intersects one interval on the first track, then we add the block of intervals $U_{257}^{(3)}$, $u_2^{(3)}$, v_4 on the second track. Similarly, if I is on the second track. We will therefore have a series of intervals, one after the other, in the order they appear in the 2-track representation of G, on the second track.

We can then add the intervals w_{HI} to intersect the block of intervals representing H and the one representing I for each intersection of intervals H and I in the representation of G. We also need to make sure that v_4 intersects w_{HI} if the block of intervals $U_{257}^{(3)}$, $u_2^{(3)}$, v_4 is used to represent I (We can just take the symmetry of the block).

We therefore realise all the edges with the w_{uv} and thus clearly obtain a depth 2 unit 2-track representation of G'_4

□

5.3 Clique Cover

Similarly, Clique Cover also is NP-complete on general graphs. Let us first introduce the problem and prove that it remains NP-complete on unit 2-track graphs.

Clique Cover
Input: An undirected graph $G = (V, E)$, a positive integer $K \leq \|V\|$
Question: Can the vertices of G be partitioned into $k \leq K$ disjoint sets $V_1, .., V_k$ such that, for $1 \leq i \leq k$ the subgraph induced by V_i is a complete graph ?

Let us note that this result can be inferred as Clique Cover already is NP-complete on graphs with maximum degree 3 as shown in [8] and that the class of maximum degree 3

graphs is actually included in the class of unit 2-track graphs.

Theorem 13. Clique Cover *is NP-complete on depth 2 unit 2-track graphs*

Proof. As Clique Cover is in NP for general graphs then we also have that Clique Cover is in NP for depth 2 unit 2-track graphs. Let us now prove that Clique Cover is NP-hard on depth 2 unit 2-track graphs. To show it, we will reduce LSAT to Clique Cover. We recall that an LSAT formula is a 3-SAT formula in which each clause (viewed as a set of literals) intersects at most one other clause, and, moreover, if two clauses intersect, then they have exactly one literal in common [3]. Let us then consider an instance of this problem with variable set $X = \{x_1, x_2, .., x_n\}$ and clause set $C = \{c_1, c_2, .., c_m\}$.
From the proof of [3], we can also notice that if two clauses are intersecting then these clauses only have two literals.

For each variable $x_i \in X$ we build the following truth-setting component $T_i = (V_i, E_i)$ where $V_i = \{x_i, \overline{x_i}, u_i\}$ and $E_i = \{(x_i, u_i), (\overline{x_i}, u_i)\}$ (See Figure 24)

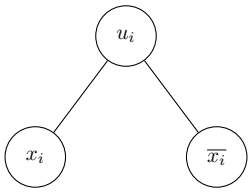

Figure 24: Gadget T_i for each variable

For each clause $c_j \in C$, we create a vertex c_j and we connect it to the vertices that represent its literals. Furthermore, if two clauses c_i and c_j intersect, we also add an edge between the two vertices c_i and c_j. Note that, each vertex c_j may only be adjacent to one other vertex c_i. We can now set $K = 2n$ and notice that this construction can be realised in polynomial time.

For example, let $X = \{x_1, x_2, x_3, x_4\}$ and $C = \{c_1, c_2, c_3\}$ such that $c_1 = (x_1, \overline{x_3})$ and $c_2 = (\overline{x_1}, \overline{x_3})$ and $c_3 = (\overline{x_2}, x_3, x_4)$ We then build the following graph :

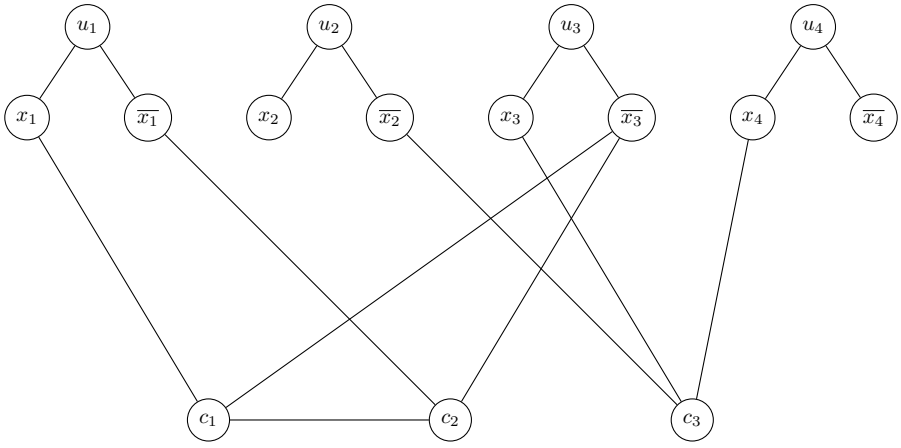

Figure 25: Example of construction of the full gadget

A satisfying truth assignment for this example would be $\tau(x_1) = T$, $\tau(x_2) = F$ $\tau(x_3) = F$, $\tau(x_4) = T$. This then gives a clique cover of G with 8 cliques such that $K_1 = \{\overline{x_1}, u_1\}$, $K_2 = \{x_2, u_2\}$, $K_3 = \{x_3, u_3\}$, $K_4 = \{\overline{x_4}, u_4\}$, $K_5 = \{\overline{x_3}, c_1, c_2\}$, $K_6 = \{x_4, c_3\}$, $K_7 = \{x_1\}$ and $K_8 = \{\overline{x_2}\}$

Let us now prove that C is satisfiable if and only if G has a clique cover of size K or less. Suppose G can be partitioned into K cliques or less. Since, in each truth-setting component T_i, x_i and $\overline{x_i}$ are not adjacent then at least two cliques are needed to cover them. Then G needs at least $2n = K$ cliques to be covered. Since G is covered by $2n$ cliques then there cannot be any truth setting component T_i where x_i, $\overline{x_i}$ and u_i are in three different cliques. Therefore there should be a clique K_i with vertex set $\{x_i, u_i\}$ or $\{\overline{x_i}, u_i\}$ for each component. We can then define a truth assignment $\tau : X \to \{T, F\}$ with $\tau(x_i) = T$ if $\overline{x_i} \in K_i$ and $\tau(x_i) = F$ if $x_i \in K_i$. Furthermore, as n cliques are required for all the K_i's then it implies that any other clique of the cover must contain one vertex x_i or its complement $\overline{x_i}$ not covered by K_i. As G is entirely covered by cliques then any vertex c_j is in a clique that therefore also contains a vertex x_i or $\overline{x_i}$. However, if a vertex c_j is in the same clique as a vertex x_i or $\overline{x_i}$ then they are adjacent in G. By construction, this therefore means that the literal x_i or $\overline{x_i}$ is in c_j. By definition of our truth assignment τ, x_i or $\overline{x_i}$ is therefore true since it is not in K_i and thus c_j is satisfied. Since all the c_j are put in a clique with some x_i then all the clauses are satisfied.
Conversely, let $\tau : X \to \{T, F\}$ be a truth assignment for C. Then, if $\tau(x_i) = T$ then we add $\overline{x_i}$ and u_i in a clique, otherwise we add x_i and u_i. Now for each vertex x_i or $\overline{x_i}$ that is not in a clique yet, if it is in two different clauses then we can put it with the vertices corresponding to the clauses in which it appears in the same clique. Finally we only need to consider the variables that appear in only one clique. Then, we just need to find a perfect matching between the vertices of the clauses and the ones of the variables and this can be done in polynomial time since this resulting graph is bipartite. By construction, as τ is a satisfying assignment then such a matching should exist. (Some variables may not be matched with a clause since the clause can be satisfied by some other variables, in this case we can just add a clique that only contains the vertex of this variable). This therefore gives a cover of G with $2n$ cliques.

Let us now give a unit 2-track representation of this graph. For each clause we add a unit interval on the first track. For each pair of intersecting clauses, we add a unit interval for the literal that they share (there can only be one shared literal in LSAT) that intersects both. Furthermore, at the other extremity, we add a unit interval for another literal that is in the clause. Note that this literal cannot be in any other clause (since a clause may intersect only one other clause and they can only share one literal) so we cannot add the same interval twice. For each clause that do not intersect any other clause, we just add two unit intervals that represent two of its literals at the extremities of the interval that represents the clause. Finally, on the other track, we add three unit intervals for x_i, $\overline{x_i}$ and u_i so that u_i intersects both x_i and $\overline{x_i}$ at its extremities. For each pair of intersecting clauses, we add two unit intervals that are intersecting. Finally, we can add a unit interval for each clause and make it intersect its remaining literal if needed. (note that this is needed if the clause is a 3-clause, which cannot be the case if the clause intersects another one)

Let us then build the corresponding unit 2-track representation of the example :

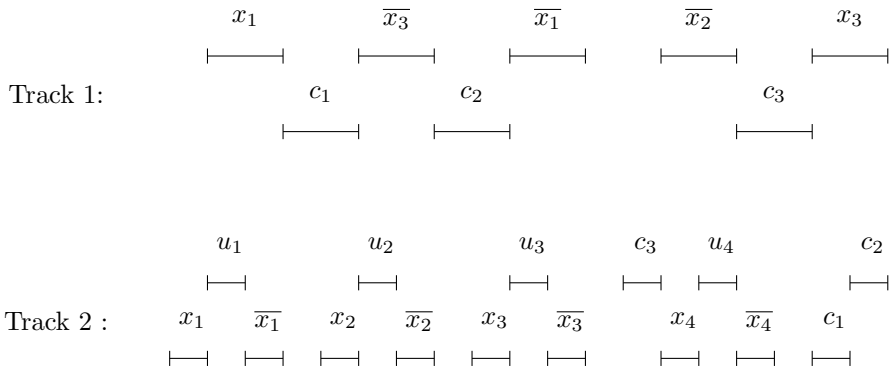

Figure 26: unit 2-track representation of the graph given in the first example

□

5.4 Biclique Cover

We can now consider a similar problem where instead of partitioning the vertex sex of a graph with as few cliques as possible, we partition it with as few bicliques as possible. Let us recall that a biclique is a complete bipartite graph, i.e. a graph $G = (V, E)$ whose vertex set V can be partitioned into two independent sets A and B ($V = A \cup B$) and such that every vertex of A is adjacent to every vertex of B ($\forall a \in A, \forall b \in B, (a, b) \in E$) We can then formulate the Biclique Cover problem below :

Biclique Cover
Input: An undirected graph $G = (V, E)$, a positive integer $K \leq \|V\|$
Question: Can the vertices of G be partitioned into $k \leq K$ disjoint sets $V_1, .., V_k$ such that, for $1 \leq i \leq k$ the subgraph induced by V_i is a complete bipartite graph ?

Again, this result can be inferred as Biclique Cover already is NP-complete on graphs with maximum degree 3 as shown in [9].

Theorem 14. Biclique Cover *is NP-complete on depth 3 unit 2-track graphs*

Proof. As Biclique Cover is in NP for general graphs then we also have that Biclique Cover is in NP for depth 3 unit 2-track graphs. Let us now, again, reduce LSAT to Biclique Cover on depth 3 unit 2-track graphs. Let us then consider an instance of LSAT with variable set $X = \{x_1, x_2, ..., x_n\}$ and clause set $C = \{c_1, c_2, ..., c_m\}$

For each variable $x_i \in X$ we build the following truth setting component $T_i = (V_i, E_i)$ where $V_i = \{x_i, \overline{x_i}, u_i, v_i, \overline{v_i}\}$ and $E_i = \{(v_i, u_i), (\overline{v_i}, u_i), (v_i, \overline{v_i}), (v_i, x_i), (\overline{v_i}, \overline{x_i})\}$ (See Figure 27)

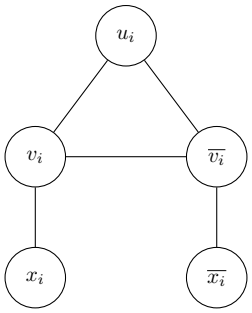

Figure 27: Gadget T_i for each variable

For each clause $c_j \in C$ we create a vertex c_j and we connect it to the vertex x_i if x_i is in c_j or to $\overline{x_i}$ if $\overline{x_i}$ is in c_j. We can now set $K = 2n$ and notice that this construction can be realised in polynomial time.

For example, let $X = \{x_1, x_2, x_3, x_4\}$ and $C = \{c_1, c_2, c_3\}$ such that $c_1 = (x_1, x_2, \overline{x_3})$ and $c_2 = (\overline{x_1}, \overline{x_3}, \overline{x_4})$ and $c_3 = (\overline{x_2}, x_3, x_4)$. We then build the following graph :

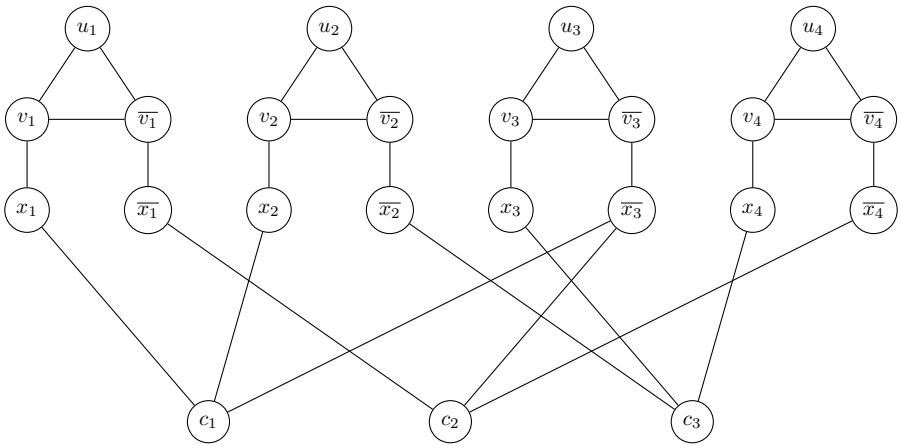

Figure 28: Example of construction of the full gadget

Let us now prove that C is satisfiable if and only if G has a biclique cover of size K or less. Suppose G can be partitioned into K bicliques or less. Since each truth setting component contains a triangle and thus not a biclique then at least two bicliques are needed to cover it and thus at least $2n$ bicliques to cover G. Since G is covered by $2n$ bicliques then there cannot be any truth setting component T_i where its vertices are in three different bicliques. Therefore each T_i must be covered with two bicliques and therefore one of the two bicliques, say K_i should either be $\{x_i, u_i, v_i\}$ or $\{\overline{x_i}, u_i, \overline{v_i}\}$. We can then define a truth assignment $\tau : X \to \{T, F\}$ with $\tau(x_i) = T$ if $\overline{x_i} \in K_i$ and $\tau(x_i) = F$ if $x_i \in K_i$. Furthermore, as n bicliques are required for all the K_i's then it implies that any other biclique of the cover must contain at least one vertex x_i or its complement $\overline{x_i}$ not covered by K_i. As, G is entirely covered by bicliques, then any vertex c_j is in the same biclique as a vertex x_i or $\overline{x_i}$. However by construction, if a vertex c_j is in the same biclique as x_i or $\overline{x_i}$, it means

that the literal x_i or $\overline{x_i}$ is in c_j. By definition of our truth assignment τ, x_i or $\overline{x_i}$ is therefore true since it is not in K_i and thus c_j is satisfied. Since all the c_j are put in a biclique with some x_i, then all the clauses are satisfied.
Conversely, let $\tau : X \to \{T, F\}$ be a truth assignment for C. Then, if $\tau(x_i) = T$ then we add $\overline{x_i}$, u_i and $\overline{v_i}$ in a biclique, otherwise we add x_i, u_i and v_i. Now for each vertex x_i (or $\overline{x_i}$) that is not in a biclique yet, if it is in two different clauses then we can put x_i and v_i (or $\overline{x_i}$ and $\overline{v_i}$) with the vertices corresponding to the clauses in which it appears in the same biclique. Finally we only need to consider the variables that appear in only one clause. Then, we just need to find a perfect matching between the vertices of the clauses and the vertices of the literals and this can be done in polynomial time since this resulting graph is bipartite. By construction, as τ is a satisfying assignment then such a matching should exist. (Some variables may not be matched with a clause since the clause can be satisfied by some other variables, in this case we can just add a biclique that only contains the vertex of this variable and its corresponding v_i). For each pair of matched vertices (x_i, c_j), we add x_i, c_j and v_i to a biclique. This therefore gives a cover of G with $2n$ bicliques.

Let us now show that this graph is unit 2-track. On the first track, we add a unit interval for each clause. For each clause c_j, we add two unit intervals that represent two of its literals at the extremities such that none of this literals are shared. Then, we add the corresponding interval v_i or $\overline{v_i}$ at the extremity of each interval x_i or $\overline{x_i}$. (We may add a unit interval for the literal that has not yet been chosen to intersect the interval of a clause)
On the second track, we add three unit intervals for each component T_i, such that v_i, $\overline{v_i}$ and u_i are intersecting. Then, we add an interval for each c_j and we make it intersect their remaining literal's interval if needed.

Let us then build the corresponding unit 2-track representation of the example :

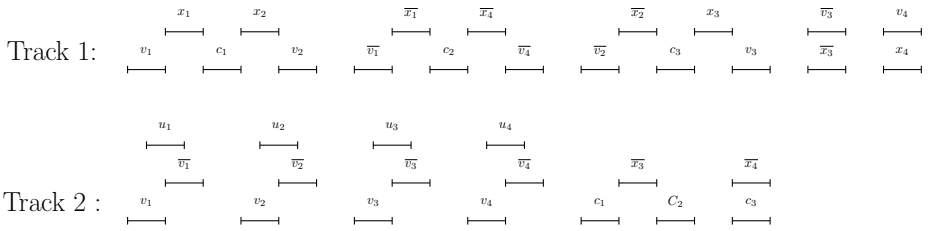

Figure 29: unit 2-track representation of the graph given in the first example

□

6 Boxicity and Track number

Boxicity d graphs and d track graphs seem to have a very close definition. In [29], it was actually shown that a boxicity d graph can be seen as the intersection of d tracks, whereas a d-track graph is, by definition, the union of d tracks. More formally, they proved the following lemma :

Lemma 15 ([29])**.** *Given a graph G, the minimum positive integer d such that there exist interval graphs $G_1, G_2, ..., G_d$ with $V(G) = V(G_i)$ for $1 \leq i \leq d$ and satisfying $E(G) = E(G_1) \cap E(G_2) \cap ... \cap E(G_d)$ is equal to the boxicity of G.*

However, we will show that we cannot bound the track number of a graph by its boxicity.

Theorem 16. *For any integer $m \geq 2$, the complete bipartite graph $K_{m,m}$ has boxicity 2 and track number $\lceil \frac{m^2}{2m-1} \rceil$.*

Proof. Let us first notice that for all $m \geq 2$ $K_{m,m}$ has a representation of boxicity 2. Consider m rectangles R_i $(1 \leq i \leq m)$ defined by the 4 points A_i, B_i, C_i and D_i such that $A_i = (0, 2(i-1))$, $B_i = (2m-1, 2i(i-1))$, $C_i = (2m-1, 2i-1)$ and $D_i = (0, 2i-1)$. These m rectangles represent the m vertices of the same independent set and are such that they are all identical and disjoint but such that all the lines (A_i, B_i) and (C_i, D_i) are parallel. Similarly, we represent the other independent set of $K_{m,m}$ by m identical and disjoint rectangles R'_i $(1 \geq i \geq m)$ defined by the 4 points A'_i, B'_i, C'_i and D'_i such that $A'_i = (2(i-1), 0)$, $B'_i = (2i-1, 0)$, $C'_i = (2i-1, 2m-1)$ and $D'_i = (2(i-1), 2m-1)$. Again, all the lines (A'_i, D'_i) and (B'_i, C'_i) are parallel. Furthermore, each rectangle R_i intersects all the rectangles R'_i. This is therefore a valid boxicity 2 representation of $K_{m,m}$. We give, for example, a boxicity 2 representation of $K_{4,4}$ below :

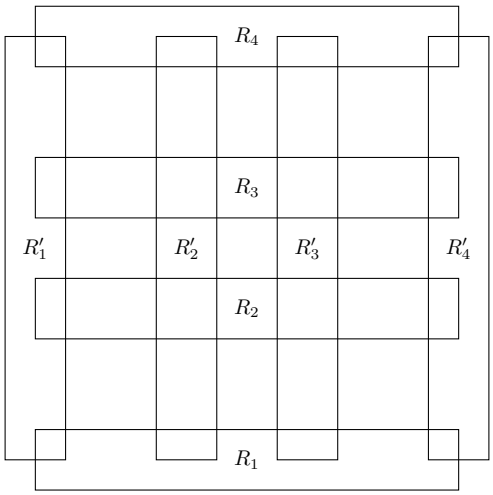

Figure 30: A boxicity 2 representation of $K_{4,4}$ (some rectangles have been shifted for clarity reasons)

Let us now prove that $K_{m,m}$ can only be realised on at least $\lceil \frac{m^2}{2m-1} \rceil$ tracks. As $K_{m,m}$ is triangle-free, it can only be realised by a t-track representation of depth 2. However, in

this case, we obtain at most one new edge at the left endpoint of each interval, except the first one of each track and therefore we can realise at most $|E| \leq (2m-1)t$ edges on t tracks. Since $K_{m,m}$ has m^2 edges we then need at least $t \geq \lceil \frac{m^2}{2m-1} \rceil$ tracks to realise all the edges. □

It would also be interesting to see whether we can bound the boxicity of a graph by its track number. The next theorem also shows that it is not possible. Let us first define the join of two graphs G and H, denoted $G \oplus H$, as being the graph formed by adding all the possibles edges between a vertex of G and a vertex of H. In other words, if $G = (V_G, E_G)$ and $H = (V_H, G_H)$ then $G \oplus H = (V_G \cup V_H, E_G \cup E_H \cup \{(u,v) | u \in G, v \in H\})$. In the next theorem, we use the graph G_n built in [31] in the following way : G_1 is the graph consisting of two non adjacent vertices and $G_n = G_{n-1} \oplus G_1$.

Theorem 17. *For any integer $n \geq 1$, the graph G_n has track number 2 but boxicity n.*

Proof. In [31], the lemma 7 shows that G_n has boxicity n, we therefore just need to show that G_n can be represented on two tracks. To do that, let us first notice that given a 2-track representation of G_{n-1} we obtain G_n by adding two intervals, I and J on each track such that I intersects all the intervals of the representation of G_{n-1} on the first track and J intersects all the intervals of the representation on the second track. Then on the first track we can just add J next to I so that it does not intersect any interval. Similarly, on the second track, we add I next to J so that it does not intersect any interval. Let us denote by $V = \{v_1, v_2, ...v_{2n}\}$ the vertices of G_n and associate the interval I_k for each v_k. On the first track, for each $k \in \{1, ..., n\}$ we add $I_k = [0, 3^{k-1}]$ and $I_{k+1} = [2 \cdot 3^{k-1}, 3^k]$. On the second track, for each $k \in \{1, ..., n\}$ we add $I_k = [2 \cdot 3^{k-1}, 3^k]$ and $I_{k+1} = [0, 3^{k-1}]$. We therefore obtain a 2-track representation of G : the intervals I_k and I_{k+1} are clearly not intersecting for each $k \in \{1, ..., n\}$. Furthermore, on the first track I_k intersects all the intervals $I_{k'}$ with $k' < k$ and on the second track I_{k+1} intersects all the intervals $I_{k'}$ with $k' < k$. We give, for example, a 2-track representation of G_3 below :

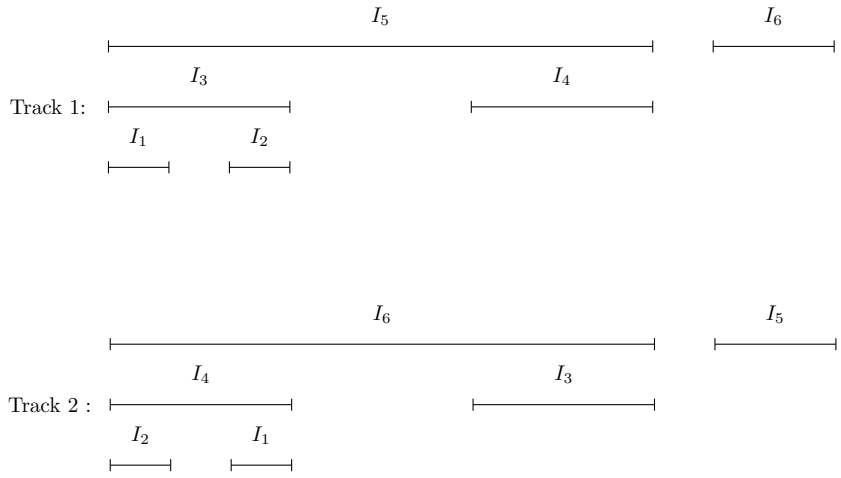

Figure 31: A 2-track representation of G_3 (some intervals have been shortened for clarity reasons)

□

Also notice that the graph G_3 in the last example allows us to show that the class of unit 2-track graphs is not included in the class of boxicity 2 graphs and thus cubicity 2 graphs. As a matter of fact, we can also represent G_3 with two unit tracks as follows :

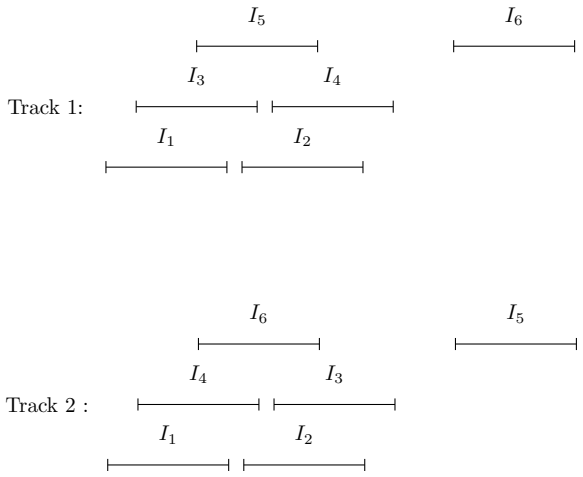

Figure 32: A unit 2-track representation of G_3 (some intervals have been shortened for clarity reasons)

It would, nonetheless, be interesting to investigate the relationship between unit d-track graphs and cubicity d graphs further. As a matter of fact, it is possible that the class of cubicity 2 graphs is included in the class of unit 2-track graphs. The main argument comes from the fact that $K_{1,5}$ is neither a cubicity 2 graph nor a unit 2-track graph even though $K_{1,4}$ both is. Furthermore, a cubicity 2 representation seems to be more restricted than a unit 2-track one since it has some local structure where a square can only intersect other squares that are close to it, whereas with two tracks, an interval on the second track can intersect some intervals that are far from it on the first one. Moreover, by noticing that a cubicity 2 graph can be built by considering the intersection of two tracks, it also seems that it is more restrictive than the union of two tracks. Finally, finding a graph that is not unit 2-track but that is of cubicity 2 seems to be harder than expected. In [1], they could not find any interval graph of claw number 3 or 4 that had a cubicity of 3, which could have also been a good example since it may also be the case that any interval graph of claw number 4 or less is unit 2-track.

However, we can notice that cubicity 3 graphs are not included in unit 3-track graphs since $K_{1,8}$ has cubicity 3 (we can intersect a cube on each of the 8 vertices of a cube). Therefore, taking the intersection of 3 tracks is somehow not more restrictive that the union of 3 tracks. We can also realise that a cubicity d graph can have a claw number of at most 2^d whereas a d'-track graph can have a claw number of at most $2d'$.

7 Characterization of unit 2-interval graphs

Finally, we can notice that a natural generalisation of the result in [28], that the class of $K_{1,3}$-free interval graphs is equal to the class of unit interval graphs, would be to prove that $K_{1,5}$-free 2-interval graphs are unit 2-interval. However this is not true as shown in the following theorem :

Theorem 18. *The class of $K_{1,5}$-free 2-interval graphs is not included in the class of unit 2-interval graphs.*

Proof. Let us consider the graph G composed of three claws whose central vertices u, v and w are connected and thus form a triangle. Respectively, the three vertices that also are connected to u are denoted u_1, u_2 and u_3, the ones connected to v are v_1, v_2 and v_3 and the ones connected to w are w_1, w_2 and w_3. This graph is clearly $K_{1,5}$-free and 2-interval but is not unit 2-interval. If we try to realise the triangle $\{u, v, w\}$ with one of their 2 intervals then one of these intervals will be trapped, say w, and cannot be used to intersect any interval w_1, w_2 and w_3 and they will thus need to be intersected with the second interval of w, but then this would form a claw and thus the representation is not unit. Now if we use several intervals for the triangle, say one interval of u intersects one interval of v. Then the second interval of u and the second intervals of v are used to intersect an interval of w then again this interval of w will have no free extremity and thus its second interval will have to intersect w_1, w_2 and w_3. For any vertex u, we will use u_L and u_R to denote the left interval and the right interval of u. Let us then give a 2-interval representation of G in Figure 33.

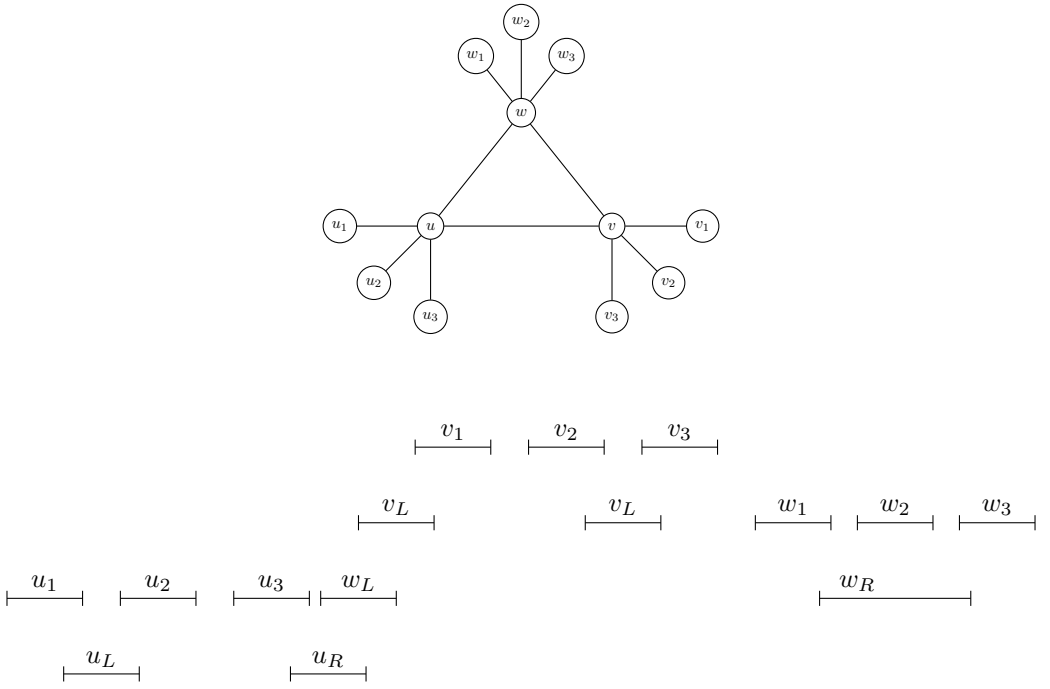

Figure 33: The graph G with a 2-interval representation of it

□

Nevertheless, we can make the conjecture that any $K_{1,5}$-free interval graph is unit 2-interval. This also comes as a natural generalization of [28]. It can also easily be seen that unit 2-interval graphs are $K_{1,5}$-free. Even though we were not able to prove it, we noticed that the difficulty of transforming an interval representation of a $K_{1,5}$-free graph into a unit 2-interval representation comes from the maximal $K_{1,3}$'s of the graph due to their asymmetry. We say that a graph has a maximal $K_{1,3}$ if there is a central vertex of a $K_{1,3}$ that is not a central vertex of a $K_{1,4}$. An interval representation of $K_{1,3}$ will have a long interval that intersects three non overlapping intervals such that the one in the middle is included in the long interval. However it is not clear how to cut the interval to make it become a 2-interval and how to dispatch the other intervals. Surprisingly, it is not difficult to fix the problem with the $K_{1,4}$'s of the graph and we can then give an algorithm for that. We can also notice that the real difficulty comes from the fact that in an interval representation of $K_{1,3}$ only one interval may be included in the long interval corresponding to the central vertex of the claw. In the case where two intervals are included then we can similarly use the same cutting rules as in the following algorithm.

We assume that $I = [i_1, i_2]$,$A = [a_1, a_2]$ and so on

```
1: procedure InnerIntervals(I, 𝒥)
2:     𝒥 = {J ∈ 𝒥 |J and I are intersecting}
3:     A = argmin_{J∈𝒥}{j_2}
4:     D = argmax_{J∈𝒥}{j_1}
5:     𝒥_A = {J ∈ 𝒥 |j_1 > a_2}
6:     𝒥_D = {J ∈ 𝒥 |j_2 < d_1}
7:     B = argmin_{J∈𝒥_A}{j_2}
8:     C = argmax_{J∈𝒥_D}{j_1}
9:     return B,C
```

Algorithm 2 Cutting Algorithm

```
1: 𝒜 = ∅
2: M = max_{I∈𝒥}(i_2) + 1
3: for each I ∈ 𝒥 do
4:     if IsCentralVertex(n_I,G) then
5:         B, C = InnerIntervals(I,𝒥)
6:         i_L = [i_1,b_2]
7:         i_R = [c_1,i_2]
8:         I' = i_L ∪ i_R
9:     else
10:        I' = I ∪ [M, M + 1]
11:        M = M + 2
       𝒜 = 𝒜 ∪ I'
12: return 𝒜
```

IsCentralVertex checks if the vertex associated to the current interval is the central vertex of a claw (this can be done by checking if the complement of the neighbourhood of n_I has a triangle for example). If that is the case then by assumption n_I is the central vertex of a 4-claw and thus I has 4 disjoint intervals intersecting it. As it can have more than a 4-tuple of disjoint intervals InnerIntervals makes sure to select to following tuple (A,B,C,D) such

that A and B is the left most couple of disjoint intervals intersecting I and C and D the right most one. Notice also than B and C are necessary included in I and then B is included in i_L and C in i_R. Let us give an example of how the algorithm works in Figure 34.

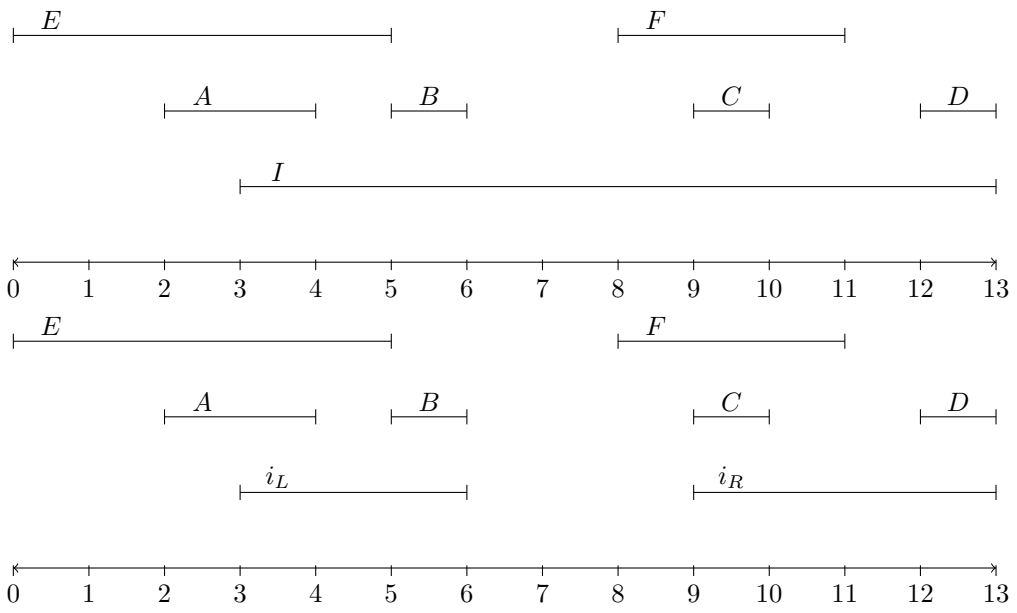

Figure 34: Example of how the algorithm cuts I

Let us first introduce a couple of definitions. For each interval $I \in \mathscr{I}$, Algorithm 2 checks if it intersects at least three disjoint intervals, if that is the case then we say that I is *cut* into two *partial* intervals i_L and i_R. Otherwise, I remains *complete* and a dummy interval is added so it forms a 2-interval. Finally we say that an interval $J = [j_1, j_2]$ is *before* $I = [i_1, i_2]$ if $j_2 < i_1$ and that J is *after* I if $i_2 < j_1$. If J is after I but before K then we say that J is *between* I and K. We also say that a graph is maximal-3-claw-free if for any vertex u of G, u is the central vertex of $K_{1,3} \Leftrightarrow u$ is the central vertex of $K_{1,4}$.

Theorem 19. *Let G be a $K_{1,5}$-free interval graph. Then if G is maximal-3-claw-free, G is unit 2-interval. Furthermore, given an interval representation of G, a unit 2-interval representation of G can be computed in polynomial time*

To prove theorem 19, let us show that Algorithm 2 computes, for any maximal-3-claw-free and $K_{1,5}$-free graph G, a unit 2-interval representation of G.

Lemma 20. *Let G be an interval graph and $\mathscr{I}$ be its interval representation. Then Algorithm 2 computes a 2-interval representation of G.*

Proof. Let us first notice that Algorithm 2 outputs a collection $\mathscr{A}$ of 2-intervals. This is the case as we only add intervals I of the form $i_L \cup i_R$ with i_L and i_R disjoint. Let us now prove that the 2-interval representation output by Algorithm 2 is a representation of G. First of all, it does not create nor delete any vertex as $|\mathscr{A}| = |\mathscr{I}|$. Second of all, it does not omit any edge : either an interval remains complete and thus unchanged or an interval is cut into i_L and i_R. As $i_L = [i_1, b_2]$ or $i_R = [c_1, i_2]$ then if an edge is forgotten there should be an interval E included in (b_2, c_1) but if it is the case then this would form a $K_{1,5}$ in the original graph as A, B, C, D and E would be 5 disjoint intervals intersecting I. Finally, it does not create any new edge, since we either cut intervals or we create a series of intervals $[M, M+1]$ that do not intersect any interval. □

We must now show that the 2-interval representation obtained by Algorithm 2 is unit. As an interval graph is unit if and only if it is 3-claw-free, i.e., if it has an interval representation such that none of its intervals intersects three disjoints intervals then a 2-interval representation is unit if none of the intervals intersects three disjoints intervals. Now let us prove the following lemma :

Lemma 21. *At the end of Algorithm 2, if an interval $I \in \mathscr{I}$ remains complete, there cannot be three disjoint complete intervals intersecting it and if I is cut into i_L and i_R, there cannot be three disjoint complete intervals intersecting i_L nor i_R.*

Proof. First of all, it is trivial that if I remains complete it cannot intersect three disjoint complete intervals because this would form a claw in the original graph and we would have thus cut I. Furthermore, it is easy to see that there cannot be three disjoint complete intervals intersecting i_L by construction. Indeed, when I has been cut, Algorithm 2 chose two intervals A and B and so that they are disjoint and that b_2 is the smallest,i.e., Algorithm 2 chose the leftmost couple of disjoint intervals. Similarly, C and D were chosen so that they are disjoint and that c_1 is the greatest. If there were three disjoint intervals M, N and O intersecting i_L (resp. i_R) then B (resp. C) would have been N and thus i_L (resp. i_R) would not have intersected O (resp. M) (See Figure 35)

□

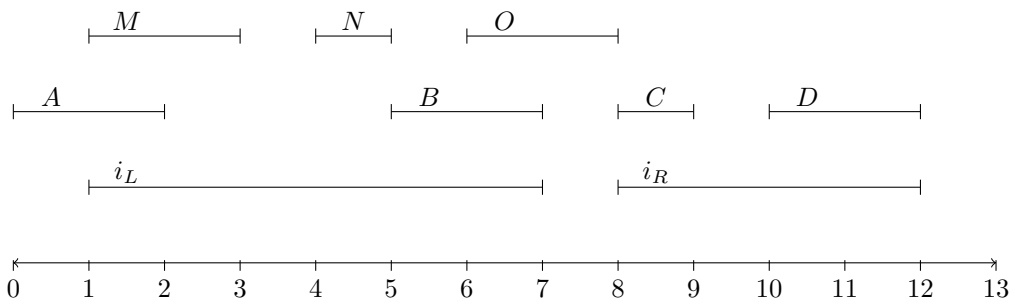

Figure 35: Algorithm 2 would have chosen B=N here as $n_1 > a_2$ and $n_2 < b_2$

We can now start to prove that there cannot be three disjoint intervals intersecting any interval at the end of Algorithm 2.

Lemma 22. *At the end of Algorithm 2, if an interval I intersects three disjoint intervals u, v, w (in this order) then either v is complete or we can find an interval V' included in v that is complete*

Proof. If V is partial then either $v = v_L$ or $v = v_R$. If $v = v_L$ then there is a complete interval B included in V. If $v = v_R$ then there is a complete interval C included in V. □

Lemma 23. *At the end of Algorithm 2, if an interval I intersects three disjoint intervals u, V, w (in this order) with V complete, then I must be partial*

Proof. According to lemma 21, if $w = W$ is complete, u must be partial. If $u = u_R$ then originally there was some complete interval U that has been cut into u_L and u_R, but then U, V and W would have cause I to be cut too : impossible. Therefore u=u_L but then there is a complete interval B included in u_L that shares the same end as u_L and hence intersects I. Thus B, V and W are three disjoint complete intervals : this would have force I to be cut too. Symmetrically, if $u = U$ is complete, w must be partial. But for the same reason, if $w = w_L$ then I would have been cut and if $w = w_R$ then there is a complete interval C included in w_R and sharing the same beginning as w_R intersecting I as well. U,V and C would then force I to be cut. Identically, if both u and w are partial and $u = u_R$ and $w = w_L$ then also originally there are two complete intervals U and W that would have cause I to be cut. If $u = u_R$ and $w = w_R$ then there is a complete interval C that shares the same beginning as w_R and originally there is a complete interval U that has been cut into u_L and u_R so I would have been cut. Symmetrically if $u = u_L$ and $w = w_L$. Finally, if $u = u_L$ and $w = w_R$, then there are a complete interval B and a complete interval C that would have caused I to be cut too. We illustrate this in the following two figures.

□

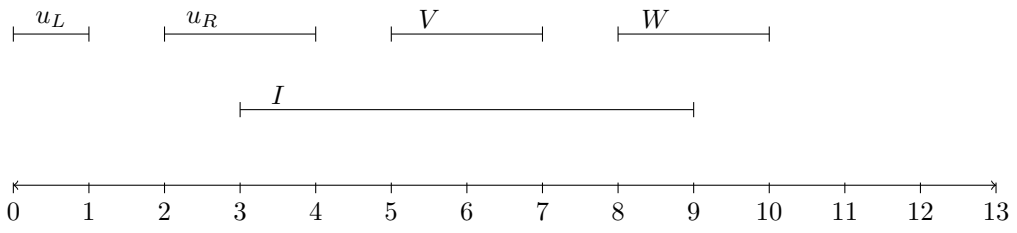

Figure 36: I would be cut as in the original representation there was an interval U intersecting I

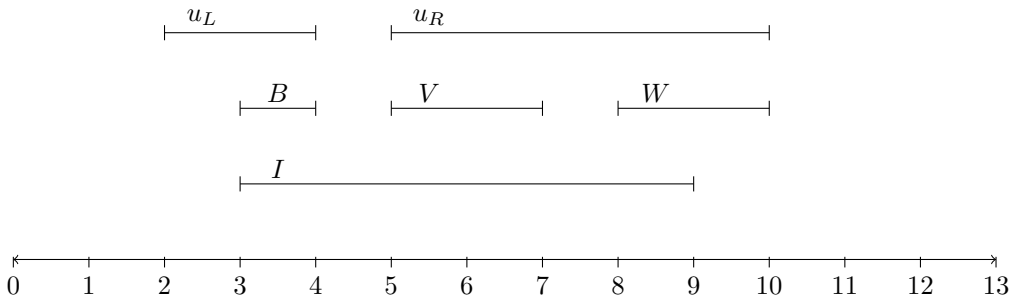

Figure 37: If U has been cut then there was a complete interval B that intersects I

Lemma 24. *At the end of Algorithm 2, a partial interval i cannot intersect three disjoint intervals u, V, w (in this order) with V complete*

Proof. As i is partial then either $i = i_L$ or i_R. We first assume that $i = i_L$. Then if u is complete then i_L would not intersect w since U and V is a couple of two disjoint complete intervals and i_L ends at the end of the second interval of the leftmost couple. Same if u is partial and $u = u_R$ then originally there is a complete interval U such that U and V were a couple of two disjoint complete intervals so i_L would not intersect w. We can assume that $u = u_L$ but then there is a complete interval B that shares the same end as u_L so B also intersects i_L. Therefore i_L would not intersect w since B and V would form a couple of disjoint intervals. Now let us assume that $i = i_R$, if w is complete that i_R would not intersect u since V and W is a couple of two disjoint complete intervals and i_R starts at the beginning of the first interval of the rightmost couple Similarly if $w = w_L$, there was a complete interval W cut into w_L and w_R that formed a couple of complete intervals so i_R would not begin before the beginning of V and thus would not intersect u. Finally if $w = w_R$ then there is a complete interval C that shares the same beginning as w_R and thus V and C is a couple of disjoint intervals so i_R does not intersect u either. □

Therefore according to lemma 22, if there are three disjoint intervals u, v and w intersecting I, we can find a complete interval V' such that u, V' and w are disjoint. However in this case, according to lemma 23, I is necessarily partial but then according to lemma 24 this is not possible. Therefore we cannot have three disjoint intervals on any interval I at the end of Algorithm 2.

8 Conclusion

Interval graphs have been tremendously studied and their real-life applications are diverse. Furthermore, most problems on interval graphs are easy as they can be solved efficiently by a polynomial time algorithm. However, the class of interval graphs is relatively restricted and some real-life problems need more intervals to be modeled. For this reason, t-interval graphs and d-track graphs have been introduced. Nonetheless, despite their wide range of applications, most problems on these classes unfortunately remain intractable for $t \geq 2$ and $d \geq 2$. As we have shown, it is the case for Vertex Cover, Feedback Vertex Set, Clique Cover and Biclique Cover but also for most problems that are NP-hard on general graphs. Even approximating them is difficult on these classes as we showed, for example, that Feedback Vertex Set was APX-hard on unit 2-track graphs. Finally, even within a parameterized framework, many problems are W[1]-hard on 2-track graphs such as Domination or Independent Set. It is also surprising to notice that restricting the length of the intervals in a d-track or a t-interval representation does not make any problem simpler. It would then be interesting to see if there is some problem where restricting the length allows to solve the problem significantly faster and to understand how we can take advantage of this restriction.

The different classes of t-interval and d-track graphs all are related and different algorithms to switch from one representation to the others have been introduced. we gave an algorithm that modifies an interval representation to obtain a unit ψ-track representation where ψ is the claw number of the associated graph. Similarly, we gave an algorithm to obtain a unit 2-interval representation of an interval graph of claw number 4 but with no maximal $K_{1,3}$.

Finally, despite having a close structure to d-track graphs at first sight, the class of boxicity d graphs contains graphs that cannot be represented on d tracks and vice versa. It would however be interesting to see whether the class of cubicity 2 graphs is strictly included in the class of unit 2-track graphs or not. This seems to be possible since we can at most represent $K_{1,4}$ but not $K_{1,5}$ with both a unit 2-track representation and a cubicity 2 representation. Furthermore, we would naturally believe that taking the intersection of two (or more) tracks is more restricting than taking their union. As shown, this is not true for d-track graphs and boxicity d graphs (for $d \geq 2$) but it may hold if we consider two unit tracks. However, we can notice that cubicity 3 graphs are not included in unit 3-track graphs since we can represent a $K_{1,8}$ with a cubicity 3 representation. Therefore, from $d = 3$, taking the intersection of unit tracks gives different graphs than taking the union. Furthermore, in [1], they gave an upper bound on the cubicity of an interval graph in regards to its claw number Ψ. However, as explained their upper bound may not be tight as they have not found any interval graph with cubicity greater than $\lceil log(\Psi) \rceil$. It therefore seems natural to conjecture that we can represent an interval graph of claw number Ψ be a cubicity $\lceil log(\Psi) \rceil$ representation as this would come as a generalization of the Theorem of Roberts stating that an interval graph is unit if and only if it does not contain an induced $K_{1,3}$. Similarly, we would also expect the unit interval number or even the unit track number of an interval graph with claw number Ψ to be $\lceil \frac{\Psi}{2} \rceil$ and thus improve the result obtained by our algorithms in section 4 and section 7.

Finally, as most problems remain NP-hard on t-interval graphs and d-track graphs (for $t \geq 2$ and $d \geq 2$) as illustrated in Section 5 (Feedback Vertex Set and Clique Cover for example), some interesting directions would be to study the parameterized complexity of these problems on the stated classes of graphs and see if we can use their structure to

design better FPT algorithms or to obtain polynomial kernels.

References

[1] Abhijin Adiga and L Sunil Chandran. Cubicity of interval graphs and the claw number. *Journal of Graph Theory*, 65(4):323–333, 2010.

[2] Thomas Andreae. On the unit interval number of a graph. *Discrete applied mathematics*, 22(1):1–7, 1988.

[3] Esther M Arkin, Aritra Banik, Paz Carmi, Gui Citovsky, Matthew J Katz, Joseph SB Mitchell, and Marina Simakov. Selecting and covering colored points. *Discrete Applied Mathematics*, 250:75–86, 2018.

[4] József Balogh and András Pluhár. A sharp edge bound on the interval number of a graph. *Journal of Graph Theory*, 32(2):153–159, 1999.

[5] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)*, 48(5):1069–1090, 2001.

[6] Reuven Bar-Yehuda, Magnús M Halldórsson, Joseph Naor, Hadas Shachnai, and Irina Shapira. Scheduling split intervals. *SIAM Journal on Computing*, 36(1):1–15, 2006.

[7] Kenneth P Bogart and Douglas B West. A short proof that"proper= unit". *arXiv preprint math/9811036*, 1998.

[8] Márcia R Cerioli, Luérbio Faria, Talita O Ferreira, Carlos AJ Martinhon, Fábio Protti, and B Reed. Partition into cliques for cubic graphs: Planar case, complexity and approximation. *Discrete Applied Mathematics*, 156(12):2270–2278, 2008.

[9] Sebastian M Cioaba. *The NP-completeness of some edge-partitioning problems*. PhD thesis, M. Sc. Thesis, Queen's University at Kingston, Canada, 2002.

[10] Joel E Cohen and David W Stephens. *Food Webs and Niche Space.(MPB-11), Volume 11*. Princeton University Press, 2020.

[11] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[12] Maxime Crochemore, Danny Hermelin, Gad M Landau, and Stéphane Vialette. Approximating the 2-interval pattern problem. In *European Symposium on Algorithms*, pages 426–437. Springer, 2005.

[13] András Frank et al. Some polynomial algorithms for certain graphs and hypergraphs. 1976.

[14] Delbert Fulkerson and Oliver Gross. Incidence matrices and interval graphs. *Pacific journal of mathematics*, 15(3):835–855, 1965.

[15] Philippe Gambette and Stéphane Vialette. On restrictions of balanced 2-interval graphs. In Andreas Brandstädt, Dieter Kratsch, and Haiko Müller, editors, *Graph-Theoretic Concepts in Computer Science, 33rd International Workshop, WG 2007, Dornburg, Germany, June 21-23, 2007. Revised Papers*, volume 4769 of *Lecture Notes in Computer Science*, pages 55–65. Springer, 2007. doi:10.1007/978-3-540-74839-7_6.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition, 1979. URL: http://www.amazon.com/Computers-Intractability-NP-Completeness-Mathematical-Sciences/dp/0716710455.

[17] Paul C Gilmore and Alan J Hoffman. A characterization of comparability graphs and of interval graphs. *Canadian Journal of Mathematics*, 16:539–548, 1964.

[18] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

[19] Jerrold R Griggs. Extremal values of the interval number of a graph, ii. *Discrete Mathematics*, 28(1):37–47, 1979.

[20] Jerrold R Griggs and Douglas B West. Extremal values of the interval number of a graph. *SIAM Journal on Algebraic Discrete Methods*, 1(1):1–7, 1980.

[21] György Hajós. Uber eine art von graphen. *Internationale mathematische nachrichten*, 1957.

[22] P Horak et al. A short proof of a linear arboricity theorem for cubic graphs. 1982.

[23] Minghui Jiang. Recognizing d-interval graphs and d-track interval graphs. *Algorithmica*, 66(3):541–563, 2013.

[24] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[25] C Lekkeikerker and J Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.

[26] Chin Lung Lu and Chuan Yi Tang. A linear-time algorithm for the weighted feedback vertex problem on interval graphs. *Information Processing Letters*, 61(2):107–111, 1997.

[27] Romeo Rizzi. Minimum weakly fundamental cycle bases are hard to find. *Algorithmica*, 53(3):402–424, 2009.

[28] Fred S Roberts. Indifference graphs. *Proof techniques in graph theory*, pages 139–146, 1969.

[29] Fred S Roberts. On the boxicity and cubicity of a graph. *Recent progress in combinatorics*, 1:301–310, 1969.

[30] Craig A Tovey. A simplified np-complete satisfiability problem. *Discrete applied mathematics*, 8(1):85–89, 1984.

[31] William T Trotter Jr. A characterization of robert's inequality for boxicity. *Discrete Mathematics*, 28(3):303–313, 1979.

[32] Stéphane Vialette. On the computational complexity of 2-interval pattern matching problems. *Theoretical Computer Science*, 312(2-3):223–249, 2004.

[33] Douglas B West. A short proof of the degree bound for interval number. *Discrete mathematics*, 73(3):309–310, 1989.

[34] Douglas B. West and David B. Shmoys. Recognizing graphs with fixed interval number is np-complete. *Discret. Appl. Math.*, 8(3):295–305, 1984. doi:10.1016/0166-218X(84)90127-6.

[35] Peisen Zhang, Eric A Schon, Stuart G Fischer, Eftihia Cayanis, Janie Weiss, Susan Kistler, and Philip E Bourne. An algorithm based on graph theory for the assembly of contigs in physical mapping of dna. *Bioinformatics*, 10(3):309–317, 1994.