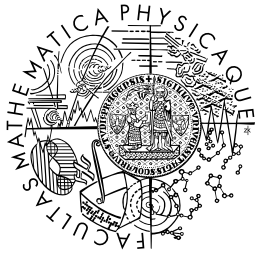


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Daniel Onduš

Partial representation extension for subclasses of interval graphs

Department of Applied Mathematics

Supervisor of the master thesis: prof. RNDr. Jan Kratochvíl, CSc.

Study programme: Mathematics

Study branch: Mathematical Structures

Prague 2021

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to thank my supervisor prof. Kratochvíl for all his little (and sometimes also bigger) remarks, suggestions and nudges that helped me shape this thesis and, most importantly, understand it thoroughly. I appreciate countless hours he spent listening to my unfinished ideas and looking for counterexamples and improvements. Finally, I would like to apologize for my numerous grammatical mistakes and incorrect definitions he had to correct.

Title: Partial representation extension for subclasses of interval graphs

Author: Daniel Onduš

Department: Department of Applied Mathematics

Supervisor: prof. RNDr. Jan Kratochvíl, CSc., Department of Applied Mathematics

Abstract: The problem of extending partial representations for an interval graph asks, whether it is possible to extend a given representation of some vertices to a valid representation of the entire graph. In this thesis we extend the recent result of Klavík et al. [5] who proved REPEXT can be decided for proper and unit interval graphs in polynomial time.

We describe properties of $\mathcal{PI}^{\pm}$ and $\mathcal{U}^{\pm}$ graphs and their representations and present algorithms deciding REPEXT for these classes in polynomial time.

In the process, we characterize relations between the $K_{1,3}$'s in a graph and show that we can decide the open vertex of every $K_{1,3}$. We also define notions of representation of the same order type and locally similar representations as well as intervals forced and locally forced to be closed (open) that are essential for extending partial representations when multiple types of intervals can occur in the same representation. We characterize intervals forced and locally forced to be closed (open) in a $\mathcal{U}^{\pm}$ graph using integer gaps in the pre-representation and we construct lower bounds for the rightmost endpoint of a component in polynomial time.

Keywords: graph, interval graph, computational complexity, partial representation

Contents

Introduction

Interval graphs and their various subclasses have been studied quite extensively as they play an important role in scheduling in computer science but interestingly they also have found applications in various unrelated fields such as genetics and bioinformatics.

We say a graph is an *interval graph* if each vertex can be represented by an interval on the real line such that two intervals have a nonempty intersection if and only if the corresponding vertices are adjacent.

Properties of these graphs allow us to construct many efficient algorithms that cannot be used in the general case. Most notably, the maximal independent set can be solved in polynomial time by a simple greedy algorithm as presented in [4] while being NP-complete for graphs in general.

Probably the two best studied subclasses are *proper interval graphs*, where no interval is a proper subset of another, and *unit interval graphs*, where all the intervals are of length one. As proven by Roberts in [10] these two classes coincide as long as all the intervals are closed. This result was generalized by Frankl and Maehara [3] who proved that the class of unit interval graphs where all the intervals are closed is the same as the class of unit interval graphs where all the intervals are open. If we allow different types of intervals in the same representation, we get several new classes as described in [2]. The most general class is the class of *mixed unit interval graphs* where closed, open and both types of half-closed intervals can be used in the same representation. Despite this being slightly misleading, for historical reasons when talking about unit interval graphs, we refer to the class of unit interval graphs where all the intervals are closed instead of the class of mixed unit interval graphs.

This thesis focuses on classes $\mathcal{PI}^{\pm}$ and $\mathcal{U}^{\pm}$ that are superclasses of proper and unit interval graphs respectively where we allow both closed and open intervals in the same representation. A graph is a $\mathcal{PI}^{\pm}$ graph if it has an interval representation such that no closed interval is a proper subset of another closed interval and each open interval is contained in a closed interval with the same endpoints. A graph is a $\mathcal{U}^{\pm}$ graph if all the intervals are of length one and they are either closed or open. As shown in [8], these two classes coincide for twin-free graphs and graphs belonging to these classes can be recognized in linear time as described in [7].

The partial representation extension problem is a natural generalization to the recognition problem. In this problem we are given a graph G belonging to some class $\mathcal{C}$ with some vertices already being pre-represented. We ask whether it is possible to extend the given pre-representation to a $\mathcal{C}$ representation for the entire graph. This can be done in polynomial time for interval graphs as proven in [6] and also for proper and unit interval graphs as proven in [5].

In this thesis we modify the algorithms presented in [5] and prove that the partial representation extension problem can be solved in polynomial time also for $\mathcal{PI}^{\pm}$

and $\mathcal{U}^{\pm}$ graphs. Furthermore, we construct a lower bound for the rightmost endpoint of a representation of a $\mathcal{U}^{\pm}$ graph in polynomial time. Since this bound cannot always be attained, we present an algorithm minimizing this endpoint subject to an ε-grid in polynomial time.

1. Preliminaries

In this chapter we are going to establish notation used throughout this thesis and some fundamental concepts.

1.1 Notation

For a vertex v from graph G, we denote the interval representing it by $I(v)$, the left endpoint of this interval by l_v and the right endpoint by r_v. In certain situations, where we have vertices indexed by $i \in I$ (say v_i or u_i) and if there is no danger of confusion, we will use l_i and r_i for the sake of brevity. If we know that an interval $I(v)$ is closed (open, respectively), we will call the vertex v closed (open) as well. Existence of an interval in a representation implies existence of a vertex in a graph and vice versa.

Two vertices u and v are called *twins* if $N_G[u] = N_G[v]$ and *false twins*, if $N_G(u) = N_G(v)$ (hence adding an edge connecting false twins makes them twins and vice versa). It is straightforward to see that if we swap the names of false twins (or twins) our graph is going to stay exactly the same (in other words, any permutation of false twins, as well as any permutation of twins, is an automorphism of the graph). Twins are also referred to as indistinguishable vertices in related literature.

1.2 Studied classes and problems

Definition 1. *We say a graph is an $\mathcal{I}^{\pm}$ graph if it has an interval representation such that each interval is either closed or open. We call each such representation an $\mathcal{I}^{\pm}$ representation.*

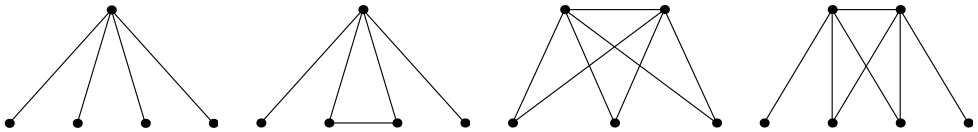
Definition 2. *We say a graph G is a $\mathcal{PI}^{\pm}$ graph if it has an $\mathcal{I}^{\pm}$ representation such that*

- *(proper condition) there are no two distinct vertices u, v in $V(G)$ such that $I(u)$ and $I(v)$ are closed and $I(u) \subsetneq I(v)$ and*
- *(master vertex condition) for every vertex $u \in V(G)$ such that $I(u)$ is open, there is a vertex $v \in V(G)$ such that $I(v)$ is closed and has the same endpoints as $I(u)$.*

Definition 3. *We say a graph is a $\mathcal{U}^{\pm}$ graph if it has an $\mathcal{I}^{\pm}$ representation such that each interval is of length one.*

Theorem 1 (Rautenbach and Szwarcfiter [8])**.** *For a twin-free graph G, the following statements are equivalent.*

- *G is a* $K_{1,4}$*,* $K^*_{1,4}$*,* $K^*_{2,3}$*,* $K^*_{2,4}$ *free interval graph.*
- *G is a* $\mathcal{PI}^{\pm}$ *graph.*
- *G is a* $\mathcal{U}^{\pm}$ *graph.*



(a) $K_{1,4}$ (b) $K^*_{1,4}$ (c) $K^*_{2,3}$ (d) $K^*_{2,4}$

Figure 1.1: Forbidden induced subgraphs for twin-free $\mathcal{U}^{\pm}$ graphs

Now we look at the problem we want to solve. A *pre-representation* (or a partial representation) Pr of a graph G is a representation of some induced subgraph of G. In other words we are given intervals representing some vertices of G. We call these vertices *pre-represented*.

Problem : REPEXT($\mathcal{C}$) - Partial representation extension of $\mathcal{C}$
Input : A graph G with a $\mathcal{C}$ pre-representation Pr
Question : Does G have a $\mathcal{C}$ representation R extending Pr?

Our general approach is to describe conditions that a valid representation R extending the pre-representation must satisfy and construct it if it exists. Therefore many propositions silently assume the pre-representation can be extended.

1.3 $K_{1,3}$'s

Induced copies of the claw $K_{1,3}$ in our graph play an important role as they distinguish proper and unit interval graphs from $\mathcal{PI}^{\pm}$ and $\mathcal{U}^{\pm}$ graphs. Moreover, it is also proven in [8], that if no vertices are pre-represented, then all the vertices that have to be represented by open intervals come from $K_{1,3}$'s. We will prove a stronger version of this fact in Lemma 13.

As proven in [2] and [8], in every $\mathcal{PI}^{\pm}$, as well as in every $\mathcal{U}^{\pm}$, representation of $K_{1,3}$ the vertex of degree three is represented by a closed interval and one of the leaves is represented by an open one. We call the vertex of degree three the *middle vertex* of this $K_{1,3}$, the leaf represented by an open interval the *open vertex* of this $K_{1,3}$ and the remaining two vertices represented by closed intervals together with the middle vertex *closed vertices* of this $K_{1,3}$.

Observation. Twins of vertices belonging to $K_{1,3}$ in $\mathcal{PI}^{\pm}$ and $\mathcal{U}^{\pm}$ representations have to be represented by identical intervals.

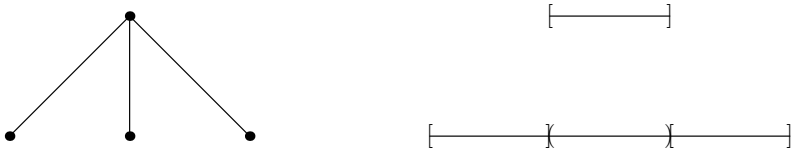


Figure 1.2: the claw $K_{1,3}$ and its $\mathcal{U}^{\pm}$ representation

Proof. Suppose we have represented a $K_{1,3}$ where u is the middle vertex, v is the open vertex and x and w are the remaining closed vertices. Twins of u clearly have to be represented by identical intervals. Twins of v cannot be represented by intervals shorter or longer than $I(v)$ because it would violate conditions in definition of $\mathcal{PI}^{\pm}$ and $\mathcal{U}^{\pm}$ graphs. Intervals representing twins of the remaining closed vertices x and w share only the endpoint with $I(u)$ and cannot be shorter or longer than $I(x)$ or $I(w)$ respectively because it would violate conditions in definitions again. □

We will refer to a $K_{1,3}$ that can be obtained from another $K_{1,3}$ by replacing some of its vertices by their twins as a *twin* $K_{1,3}$.

It is easy to see that all $K_{1,3}$'s in a graph can be found in polynomial time by going through all the quartets of vertices. For each $K_{1,3}$ determining the middle vertex is easy as well because it has three neighbours in this $K_{1,3}$. Now we show we can decide also the open vertex for a given $K_{1,3}$.

Algorithm 1: Deciding the open vertex for a $K_{1,3}$ using the neighbourhood condition

---

input : vertices u, v, w, x forming an induced $K_{1,3}$, where u is the middle vertex, in a $\mathcal{U}^{\pm}$ or $\mathcal{PI}^{\pm}$ graph G with a pre-representation Pr

output: vertex that will be represented by an open interval or FALSE if Pr cannot be extended

if $N_G[v] = N_G[u] \setminus \{w, x\}$ *up to twins of* w *and* x **and** *neither* v *nor any of its twins is pre-represented by a closed interval* **and** *neither* w *nor* x *is pre-represented by an open interval* **then**
  return v *is open*
else if $N_G[w] = N_G[u] \setminus \{v, x\}$ *up to twins of* v *and* x **and** *neither* w *nor any of its twins is pre-represented by a closed interval* **and** *neither* v *nor* x *is pre-represented by an open interval* **then**
  return w *is open*
else if $N_G[x] = N_G[u] \setminus \{v, w\}$ *up to twins of* v *and* w **and** *neither* x *nor any of its twins is pre-represented by a closed interval* **and** *neither* v *nor* w *is pre-represented by an open interval* **then**
  return x *is open*
else return *FALSE*;

Lemma 2 (Neighbourhood condition)**.** *Algorithm 1 chooses the open vertex of a* $K_{1,3}$ *in polynomial time. Extendibility of the pre-representation doesn't change by renaming vertices* v*,* w *and* x*.*

Proof. Each $K_{1,3}$ necessarily has to have exactly one open vertex. In case it doesn't, the pre-representation isn't extendible. Suppose this open vertex is v. Since $I(u)$ has the same endpoints as $I(v)$ in each representation, the only place, where u can get new neighbours are its endpoints. But if two non-twin closed intervals were to end on the same endpoint of $I(u)$, they would need to have different lengths and thus would violate the proper condition or the unit condition, hence we have exactly one for each endpoint coming from $K_{1,3}$ up to twins. This means the open vertex satisfies the condition $N_G[v] = N_G[u] \setminus \{w, x\}$ (up to twins). We will refer to this as to the *neighbourhood condition.*

If there is only one vertex satisfying the neighbourhood condition we are done. Suppose there are at least two, say, v and w. At first, assume our graph is twin-free. Then $N_G[v] = N_G[u] \setminus \{w, x\}$ and $N_G[w] = N_G[u] \setminus \{v, x\}$. Thus $N_G(v) = N_G(w)$ so v and w are false twins. Therefore if neither of them is pre-represented, we can pick arbitrarily because by swapping their names our graph stays exactly the same.

In case our graph is not twin-free, then open neighbourhoods of v and w differ only by their twins. Since those twins have to be represented by the identical intervals, we can conclude that if none of them is pre-represented we can pick arbitrarily again. □

We have proven, that whenever we are given a $K_{1,3}$, we can decide its open vertex and extendibility of the pre-representation doesn't depend on the order in which we process vertices of this $K_{1,3}$. If multiple candidates for the open vertex are available, then we say each such candidate can be decided both open and closed. Otherwise we say the open and closed vertices are given uniquely for this $K_{1,3}$.

Lemma 3. *Assume the pre-representation is extendible. If two $K_{1,3}$'s share a vertex that can be decided both open and closed by Algorithm 1 for at least one of them, they must be twin $K_{1,3}$'s.*

Proof. Suppose we can choose from vertices u and v to be open for the first $K_{1,3}$ and v is given uniquely for the second $K_{1,3}$. If we can choose from 3 vertices in the first $K_{1,3}$, the proof is almost identical. Assume v is a closed vertex of the second $K_{1,3}$ and cannot be decided open. If v is the middle vertex of this $K_{1,3}$, we can replace it with u to obtain a $K_{1,3}$ where u is the middle vertex. Therefore both $I(u)$ and $I(v)$ have to be closed but one of them has to be open for the first $K_{1,3}$ which is a contradiction.

We can apply the same argument if v is a non-middle closed vertex of the second $K_{1,3}$. In case v is the open vertex of the second $K_{1,3}$, we get a contradiction where both $I(u)$ and $I(v)$ have to be open.

Now we assume v is among vertices that can be decided open for both $K_{1,3}$'s. We will distinguish cases based on how many vertices the first and the second $K_{1,3}$ have in common.

- In case these $K_{1,3}$'s have exactly one vertex in common, let it be v such

that we can decide from u and v to be open for the first and from v and w for the second $K_{1,3}$. Since u and v are false twins and also v and w are false twins, also u and w are false twins. Let x be the middle vertex of the second $K_{1,3}$ and y its remaining vertex. Then x with u, v, w and y form an induced $K_{1,4}$. However, this is a forbidden subgraph for $\mathcal{U}^{\pm}$ and $\mathcal{PI}^{\pm}$ twin-free graphs by Theorem 1 and an induced $K_{1,4}$ includes no twins.

- In case they also have in common one non-middle vertex, that is uniquely given to be closed, we get a $K_{1,4}$ as above as well.

- In case these $K_{1,3}$'s have exactly two vertices in common that both can be decided to be open for both $K_{1,3}$'s, they are false twins therefore they are connected to both middle vertices. Now if those two middle vertices are not adjacent we get a graph that is not an interval graph, if they are adjacent we get an induced $K^*_{2,4}$ which is again a forbidden subgraph by Threorem 1. In a special case when those middle vertices are twins, we get an induced $K_{1,4}$ which is forbidden as well.

- In case these $K_{1,3}$'s have in common all three non-middle vertices, then, regardless of how many of them can be decided open, once we represent these three vertices, both middle vertices must be represented by the same interval hence we have twin $K_{1,3}$'s.

- In case these $K_{1,3}$'s share the middle vertex, then, regardless of how many other vertices they share, we get twin $K_{1,3}$'s since once we represent the first $K_{1,3}$, the second with the same middle interval has to be represented by identical intervals.

- In case the middle vertex x of the first $K_{1,3}$ is a non-middle closed vertex of the second $K_{1,3}$ and they also share vertex v that can be decided open or closed in both $K_{1,3}$'s. Now v and x are adjacent in the first $K_{1,3}$ and non-adjacent in the second one which is a contradiction.

□

Corollary 4. *Open vertices for non-twin $K_{1,3}$'s can be decided independently.*

Once all the vertices of $K_{1,3}$'s have been decided, we need to check whether the pre-representation was valid. We say it is invalid if there is a vertex decided open pre-represented by a closed interval or vice versa or there is a vertex decided both open and closed for different $K_{1,3}$'s. In such case we also know it cannot be extended.

Since we proved the choice of open intervals for $K_{1,3}$'s doesn't affect extendibility of the pre-representation, from now on we will only consider representations respecting this choice.

Algorithm 2: Deciding open intervals for all $K_{1,3}$'s

input : a $\mathcal{U}^{\pm}$ or $\mathcal{PI}^{\pm}$ graph G with a pre-representation Pr
output: vertex that will be represented by an open interval for each $K_{1,3}$ or FALSE if Pr cannot be extended

find all $K_{1,3}$'s;
foreach *group of twin* $K_{1,3}$*'s* **do**
  choose one $K_{1,3}$;
  find its middle vertex;
  run Algorithm 1 for this $K_{1,3}$ and its middle vertex;
  if *it returns the open vertex* **then** decide open vertices for all $K_{1,3}$'s in this group the same way;
  else return *FALSE*;
end
if *the pre-representation is not valid* **then return** *FALSE*;

1.4 Orders

Orders of vertices for interval graphs have been studied quite extensively and there are several results that are useful for us.

Lemma 5 (Roberts [9], Deng et al. [1])**.** *A graph is a proper (and unit) interval graph if and only if there exists linear ordering* $\lhd$ *of its vertices such that the closed neighbourhood of each vertex is consecutive in this order. For a connected graph this ordering is given uniquely up to reordering of groups of indistinguishable vertices and the complete reversal.*

A component is called *located* if it is has pre-represented at least one vertex. For located components we have a left-to-right order $\blacktriangleleft$ which is given uniquely. Unlocated components can be represented on the right in an arbitrary order.

Since $\lhd$ is not given uniquely when comparing twins, we restrict it to comparing groups of twins and vertices from different groups. When we say u with v are consecutive in $\lhd$ it means their groups of twins are consecutive.

Example. Let v_1, v_2, v_3, and v_4, v_5 and v_6, v_7 be three groups of twins. Then let $(v_1, v_2, v_3) \lhd (v_4, v_5) \lhd (v_6, v_7)$. Now we see v_2 and v_5 are consecutive in $\lhd$ since their groups of twins are.

In our case, we will often first remove the open vertices of $K_{1,3}$'s to obtain a proper (unit) interval graph for which we can calculate the order $\lhd$ and its reversal for each component. This order is computed independently on the pre-representation. We want $\lhd$ to be a left-to-right order for our representation, so if there are two non-twin vertices with different endpoints already pre-represented within a component, we can determine whether $\lhd$ is a left-to-right order. In case

it is not, we use its reversal. Since $\triangleleft$ was calculated independently on the pre-representation, we can assume it is left-to-right. In this case we say $\triangleleft$ is given uniquely for this component.

If only twins of one vertex or vertices with the same endpoints are pre-represented within a component, we need to discuss the case when $\triangleleft$ is left-to-right as well as the case when its reversal is left-to-right.

Furthermore, we can extend $\triangleleft$ to an ordering $<$. For twins we say $u < v$ if both u, v are pre-represented and $l_u < l_v$ holds. If $l_u = l_v$ we order them by indices. We order non-pre-represented twins by indices and insert them into $<$ immediately to the right from the leftmost pre-represented twin in this group. We see that if $\triangleleft$ is given uniquely then so is $<$ and it is a left-to-right order able to compare any two vertices within a component. For vertices from located components we say $u <' v$ if $u \in C_u$, $v \in C_v$ and $C_u \blacktriangleleft C_v$ or $u < v$ if they belong to the same component.

Definition 4. *We say representations R and R' extending the pre-representation are* of the same order type*, denoted $R \sim_{ord} R'$, if they have the same left-to-right order $\triangleleft_t$ for each located component C_t.*

1.5 Forced and locally forced intervals

Definition 5. *Assume the pre-representation is extendible.*

We say an interval $I(v)$ (and vertex v) is forced to be closed (open) *if for each representation R extending the pre-representation, $I(v)$ is closed (open). We might call this interval also* forced closed (open)*.*

Intervals $I(u)$ and $I(v)$, where u and v are not twins, form a forced open-closed pair *if for each representation R extending the pre-representation, one of them is open, the other one is closed and they have the same endpoints.*

Interval $I(u)$ is forced to be in a given position *if for each representation R extending the pre-representation, it is in this position.*

A given position for an interval means it has specified the value of the left (and also the right) endpoint. However, this doesn't restrict the interval to being open or closed.

Definition 6. *We say representations R and R' extending the pre-representation are* locally similar*, denoted $R \sim_{sim} R'$, if $R \sim_{ord} R'$ and for each vertex $v \in V(G)$, $I(v)$ in R and R' are of the same type.*

Definition 7. *We say an interval $I(v)$ is* locally forced to be closed (open) *if there exists a representation R extending the pre-representation such that $I(v)$ is closed (open) and there is no representation $R' \sim_{ord} R$ such that $I(v)$ is open (closed) and all the other intervals are of the same type as in R,*

Intervals $I(u)$ and $I(v)$, where u and v are not twins, form a locally forced open-closed pair *if there exists a representation R extending the pre-representation such that one of them is closed, the other one is open and in each $R' \sim_{sim} R$ intervals $I(u)$ and $I(v)$ have the same endpoints.*

We say something is only locally forced *when it is locally forced but it is not forced.*

Observation. Every interval forced to be closed (open) is also locally forced to be closed (open).

2. Extending representations for $\mathcal{PI}^{\pm}$ graphs

As we already mentioned, REPEXT is solvable for proper interval graphs in polynomial time. Our goal is to use this algorithm to present an algorithm solving REPEXT for $\mathcal{PI}^{\pm}$ graphs in polynomial time. We will do it in the following way. First, we find $K_{1,3}$'s in our graph G and remove their vertices decided open in order to obtain a proper interval graph G'. Then we need to deal with pre-represented open intervals as for $\mathcal{PI}^{\pm}$ graphs each open interval $I(v)$ is included in some closed interval $I(u)$ with identical endpoints. Finally, we show that if the pre-representation for this proper interval graph G' can be extended, we can extend also $\mathcal{PI}^{\pm}$ pre-representation for graph G and vice versa.

2.1 Open intervals

First, let us have a look at how can open intervals occur in our representation and how do we deal with it. Although not mentioned when defining the class of $\mathcal{PI}^{\pm}$ graphs, we make an observation regarding proper subsets of open intervals.

Observation (Proper conditions for open intervals). If $I(v)$ is an open interval, then there is no interval $I(w)$ such that it is a proper subset of $I(v)$. Also, if an open interval $I(v)$ is a subset of a closed interval $I(u)$, then $I(u)$ has the same endpoints as $I(v)$.

Proof. Let $I(u)$ be a closed interval with the same endpoints as $I(v)$. Then if $I(w)$ is a proper subset of $I(v)$, it is either closed or there exists $I(x)$ closed with the same endpoints as $I(w)$, but this is a contradiction with the proper condition for closed intervals.

For the second part, take a closed interval $I(u')$ that is superset of $I(v)$. It must have the same endpoints as $I(u)$ otherwise $I(u)$ would be its strict subset. □

2.1.1 Open intervals of $K_{1,3}$'s

As discussed in section 1.3, $K_{1,3}$'s can be found and their open vertices decided in polynomial time. To obtain a proper interval graph it is sufficient to remove these open vertices from our graph. We also need to remove their pre-represented intervals if there were any. In case we remove the pre-represented open interval of a $K_{1,3}$, we represent the corresponding middle vertex by a closed interval with the same endpoints.

2.1.2 Open intervals forced by the pre-representation

Lemma 6. *If an interval is locally forced to be open, it is either pre-represented open or it is the open interval of some $K_{1,3}$.*

Proof. Let us suppose that for a non-pre-represented vertex v, $I(v)$ is forced to be open. Using the master vertex condition, there must be a closed vertex u such that $I(u)$ has the same endpoints as $I(v)$. Vertex u is either a twin of v, but then v could be changed to closed, or there is a closed vertex ending at an endpoint of $I(u)$. In the latter case, assume it is ending at the right endpoint. If there is no closed interval ending at the left endpoint of $I(u)$, there is also no open interval ending at the left endpoint of $I(u)$, hence we can move $I(v)$ slightly to the left and change it to a closed interval which is a contradiction. If there is a closed interval ending at the right endpoint of $I(u)$, v is the open vertex of a $K_{1,3}$. □

Corollary 7. *Each interval locally forced to be open is forced to be open. If there exists a representation R extending the pre-representation, there also exists a representation $R' \sim_{ord} R$ such that all the intervals not forced to be open are closed.*

Proof. We know that open intervals of $K_{1,3}$'s as well as pre-represented open intervals are forced to be open, hence all the intervals locally forced to be open are forced to be open by the previous lemma.

Assume we have intervals $I(v_1), \ldots, I(v_f)$ that are represented as open in R but they are not forced to be open. We will change the representation R into R' in f steps. In the i-th step we change the open interval $I(v_i)$ to a closed one to obtain R_i. If we cannot change it, it would mean $I(v_i)$ was locally forced to be open by R_{i-1}. Since this cannot happen we have $R' = R_f$ in which all of them are closed. □

Now we look at the pre-represented open intervals. Let v, that is not the open vertex of a $K_{1,3}$, be pre-represented by an open interval $I(v)$. Then there must be a closed interval $I(u)$ with the same endpoints. By the same argument as above we conclude that u is either a twin of v or has exactly one extra neighbour (up to twins). Thus if we cannot find such u in our graph, the pre-representation cannot be extended.

2.1.3 Dealing with open intervals

For a vertex v pre-represented by an open interval $I(v)$, we know that there will be a closed interval $I(u)$ with identical endpoints. Therefore we would like to find the vertex u that will be represented by this closed interval, remove the open vertex v from the graph as well as $I(v)$ from the pre-representation and pre-represent closed $I(u)$ at its place instead.

Lemma 8. *Assume the pre-representation is extendible. Then for pre-represented open intervals $I(v_1),\ldots,\ I(v_m)$ we can find vertices $u_1,\ldots,\ u_m$ such that there exists a representation R extending the pre-representation where $I(u_i)$ is closed and has the same endpoints as $I(v_i)$ for every $i \in \{1,\ldots,\ m\}$ in polynomial time.*

Proof. As for open vertices of $K_{1,3}$'s, the vertex, that will be represented by a closed interval can be determined uniquely, it is the middle vertex of corresponding $K_{1,3}$ and no open vertex can belong to two or more non-twin $K_{1,3}$'s. If several twins of the open interval of a $K_{1,3}$ are pre-represented they have to be in the same position since, as we proved, twin $K_{1,3}$'s have to be represented by identical intervals.

For a pre-represented open vertex v that doesn't belong to any $K_{1,3}$ there must be a vertex u that is either a twin of v or has exactly one extra neighbor (up to twins) as explained above.

In case v has some non-pre-represented twins and there are no pre-represented intervals sharing one endpoint with $I(v)$, then we may simply use some twin of v as vertex u. In case there is a closed interval $I(w)$ sharing an endpoint with $I(v)$, we clearly cannot use its twin, because since $I(v)$ is open, v is not a neighbour of w.

In case we cannot use a twin, we need to find a vertex u with exactly one more neighbour (up to twins). This u will be represented by a closed interval, thus it cannot be the open vertex of any $K_{1,3}$. It cannot be the middle vertex either, since at this point we assume $I(v)$ is not the open vertex of any $K_{1,3}$. Since after removing open vertices of $K_{1,3}$'s our graph is a proper interval graph, for each component there is a linear order $\lhd$ unique up to reversal and reordering of groups of twins. It is clear that any suitable u is consecutive with v in $\lhd$. This means for each pre-represented open interval $I(v)$ there are (up to twins) at most two vertices in the graph that can be represented by a closed interval with the same endpoints.

Suppose there are two such vertices u_1 and u_2 with extra neighbours x_1 and x_2 respectively. We claim that if there is no pre-represented interval sharing an endpoint with $I(v)$ we can pick arbitrarily and it will not change the result of REPEXT. Let R_1 be a representation extending the pre-representation where $I(u_1)$ has the same endpoints as $I(v)$ and $I(x_1)$ starts at the right endpoint of $I(u_1)$. This means x_2 has to be represented entirely on the left from $I(v)$. Therefore the left endpoint of $I(u_2)$ ends on the left from $I(v)$ and x_2 is the greatest vertex in $\lhd$ non-adjacent with v such that $x_2 \lhd v$.

Now let us construct a representation R_2 extending the pre-representation such that $I(u_2)$ has the same endpoints as $I(v)$. Since the next interval on the right from $I(x_1)$ has nonzero distance from $I(v)$ then we can simply move $I(u_1)$ to the right and shorten $I(x_1)$ a bit. Then we move $I(u_2)$ and lengthen $I(x_2)$ to reach the left endpoint of $I(v)$. This can be done as every vertex y such that $x_2 \lhd y \lhd u_2$ is adjacent with all v, x_2 and u_2.

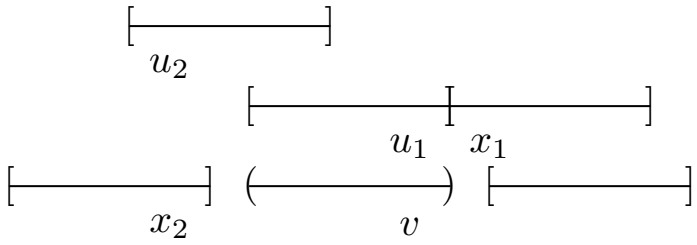


Figure 2.1: the representation R_1

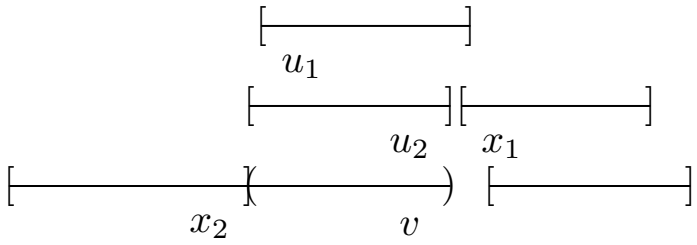


Figure 2.2: the representation R_2

If there is a pre-represented interval $I(x)$ sharing an endpoint with $I(v)$, assume $v \triangleleft x$ and u_1 with u_2 are possible candidates for u. Now we pick u satisfying $v \triangleleft u \triangleleft x$. This condition is necessary as u has to be adjacent with x but v is not.

If $I(u)$ is pre-represented in a different position than $I(v)$, we obviously cannot choose it.

To construct the representation R we first take care of open vertices of $K_{1,3}$'s and then we approach the remaining vertices one by one always adding $I(u_i)$ to the representation. Since we proved that we do not change the extendibility of the pre-representation when we make a choice, the representation can still be extended after the m-th step.

As for the running time, the only non-trivial operation is calculating $\triangleleft$ and $<$ for each component which can be done in polynomial time. Since we perform each operation at most polynomially many times, we are done. □

2.1.4 Obtaining a proper interval graph

Now we sum up the process described in the previous subsection.

For each pre-represented open interval of a $K_{1,3}$, we first pre-represent the middle vertex of the same $K_{1,3}$ by a closed interval in its position instead. Then we remove all open vertices of $K_{1,3}$'s and their pre-represented intervals and we get a proper interval graph, hence we can calculate orders $\triangleleft$ and $<$ for each component.

Using these orders, for each non-$K_{1,3}$ vertex pre-represented by an open interval we can decide the corresponding vertex that we pre-represent by a closed interval with the same endpoints, if it exists.

Now we can finally remove all pre-represented open intervals and their corre-

sponding vertices.

This way we obtain a proper interval graph G' with a proper interval pre-representation Pr'. As mentioned before, all these operations can be done in polynomial time.

2.2 Deciding REPEXT by adding open vertices to the representation

Once we solve REPEXT for the proper interval graph G' we have two options. Either it returns FALSE, which means also REPEXT for $\mathcal{PI}^{\pm}$ should return FALSE as the graph G' was a proper interval graph and while obtaining this graph and the pre-representation Pr' we couldn't change extendibility of representation as proven by Lemmas 2 and 8.

If it returns TRUE, that means there exists a representation R' extending Pr', and we want to modify it to R extending Pr that works for the $\mathcal{PI}^{\pm}$ graph G.

Lemma 9. *A representation R' for a proper interval graph G' extending Pr' obtained as above can be modified into a representation R for a $\mathcal{PI}^{\pm}$ graph G extending the original pre-representation Pr.*

Proof. For each open vertex v we need to add $I(v)$ to the representation. We have already chosen a closed vertex u such that $I(u)$ has the same endpoints. Therefore we can just put $I(v)$ into this place. Now this is either already a valid representation or v has more or less neighbours than it should.

We know that u has (up to twins) either one or two more neighbours than v. In case v has more neighbours than it should, there is an interval $I(x)$ that should end at the endpoint of $I(u)$ but it ends inside. It can be easily moved to the endpoint. The only thing left to check is that x doesn't lose any neighbours. But each neighbour of x must end on the endpoint or outside of $I(u)$ on one side, otherwise it would violate the proper condition.

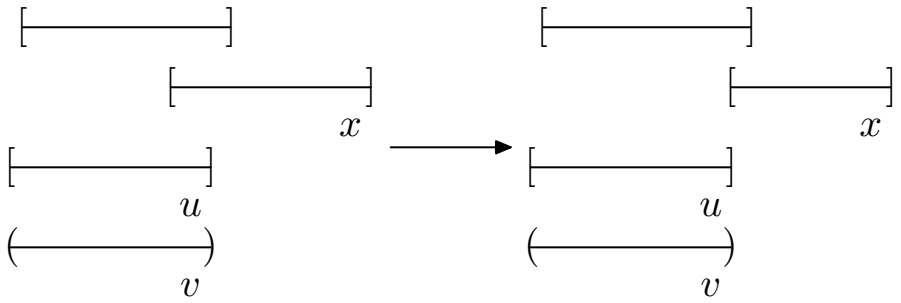


Figure 2.3: removing an extra neighbour of an open interval

If $I(x)$ is a pre-represented interval, we shorten $I(u)$ accordingly. In a special case both $I(x)$ and $I(u)$ are pre-represented, then either v is the open vertex of

a $K_{1,3}$ but this would lead to a contradiction as $I(u)$ and $I(x)$ can be touching only with the endpoints, or $I(v)$ is pre-represented open in a position different to $I(u)$ thus we couldn't choose it.

In case v has less neighbours than it should, it means there is an interval $I(x)$ that ends at the endpoint of $I(u)$ but it should end inside. We can lengthen it without obtaining new neighbours as there has to be non-zero distance between the endpoint of $I(u)$ and the endpoint of the closest (non-twin) interval inside it.

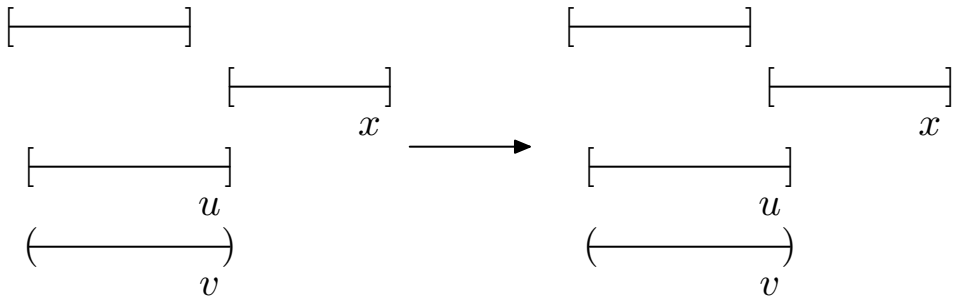


Figure 2.4: adding a missing neighbour of an open interval

If $I(x)$ is a pre-represented interval, we lengthen $I(u)$ accordingly. If we cannot, there has to be an open $I(w)$ with the same endpoints as $I(x)$ non-adjacent with u. $I(w)$ either comes from a $K_{1,3}$ or was pre-represented, but since open v is a neighbour of x it is also a neighbour of w and so is u. A special case when both $I(x)$ and $I(u)$ are pre-represented leads to a contradiction as well. □

Theorem 10. *REPEXT for $\mathcal{PI}^{\pm}$ graphs is decidable in polynomial time.*

Proof. This is a corollary of the previous lemma together with the facts that all operations we performed before running REPEXT for proper interval graphs can be done in polynomial time and REPEXT itself runs in time $\mathcal{O}(n+m)$. □

We summarize the presented algorithm in the following pseudo-code display.

Algorithm 3: REPEXT for $\mathcal{PI}^{\pm}$ graphs

input : $\mathcal{PI}^{\pm}$ graph G and a pre-representation Pr
output: TRUE if Pr can be extended and FALSE otherwise

run Algorithm 2;
if *it returns FALSE* **then return** *FALSE*;
foreach *pre-represented open vertex* v *of a* $K_{1,3}$ **do**
  represent $I(u)$ with the same endpoints where u is the corresponding middle vertex to v and remove $I(v)$ from Pr
end
foreach *open vertex* v *of a* $K_{1,3}$ **do** remove it from G;
obtain order of vertices $\triangleleft$ for each located component;
foreach *pre-represented open interval* $I(v)$ **do**
  find a vertex u such that $I(u)$ is closed with the same endpoints as $I(v)$ and represent it;
  if u *cannot be found* **then return** *FALSE* ;
end
foreach *vertex* v *pre-represented by open interval* $I(v)$ **do** remove v from G and $I(v)$ from Pr;
run proper interval REPEXT for G' and Pr';
if *it returns TRUE* **then return** *TRUE*;
else return *FALSE*;

3. Extending representations for $\mathcal{U}^{\pm}$ graphs

Theorem 1 proves a twin-free graph is a $\mathcal{U}^{\pm}$ graph if and only if it is a $\mathcal{PI}^{\pm}$ graph. Representations however, might differ significantly.

3.1 Linear programming

When extending representations for $\mathcal{U}^{\pm}$ graphs, we slightly modify the approach of Klavík et al. from [5], where this is done by minimizing the right endpoint of each component using linear programming.

To do this we first need the notion of an ε-grid. For practical purposes we restrict our attention to instances with rational endpoints of pre-represented intervals. Suppose for each pre-represented vertex v_i, where $i \in \{1, 2, \ldots, b\}$, the left endpoint ℓ_i of the interval $I(v_i)$ is expressed as an irreducible fraction $\frac{p_i}{q_i}$. Let

$$\varepsilon' \equiv \frac{1}{lcm(q_1, \ldots, q_b)}$$

and

$$\varepsilon \equiv \frac{\varepsilon'}{n}.$$

Now the ε-grid is the set of points $\{k\varepsilon \ ; \ k \in \mathbb{Z}\}$.

Lemma 11. *If the pre-representation can be extended into some representation R', it can be also extended into a representation R, in which all intervals have endpoints on the ε-grid for ε defined as above, such that $R \sim_{sim} R'$.*

Proof. We proceed similarly as in [5].

We construct the representation R in two steps. The first step is the left shifting where we move the left endpoint of each interval to the closest ε'-grid point to the left. By this step we might introduce some additional intersections and also lose some existing ones for open intervals.

The second step is the right shifting where we move l_i to $l_i + RS(v_i) \cdot \varepsilon$, where RS is a mapping from $V(G)$ to $\{0, 1, \ldots, n-1\}$. If we relaxed image of RS to $[0, n)$, we could obtain a viable representation by reversing the left shifting. However, correctness of a representation doesn't depend on the exact value of the shifts but on their relative order. Suppose $0 = s_1 < s_2 < \cdots < s_t$ for $t \leq n$ were the different values of shifts in the left shifting. Then for each v such that it was shifted by value s_i, we set $RS(v) = (i-1)$.

This means we have exactly the same intersections as in R'. Finally, all pre-represented intervals are kept in place as their left endpoints lie on the ε'-grid and we didn't change any interval from open to close or vice-versa. □

Next requirement for running the linear program is the order of vertices $\triangleleft$ extended to $<$ for each component C_t that can be obtained only for unit interval graphs. A convenient way to approach this is to first find the $K_{1,3}$'s and remove their open vertices from the graph G, by doing so we get a unit interval graph G'. If any of them were pre-represented, we represent the corresponding middle vertices in their places to keep this information. At this point we are almost ready to run the linear program, however we need to change certain inequalities to account for the open intervals. Therefore we need to find them, which is discussed in the rest of this chapter.

As already mentioned, located components are ordered from left to right by $\blacktriangleleft$ and we process them in this order. Assume all components up to C_{t-1} have been minimized. The open vertices of $K_{1,3}$'s have been removed before obtaining the order. Thus we insert each open vertex v of a $K_{1,3}$ in C_t into the order $\triangleleft$ in a way that for the corresponding middle vertex u, $u \triangleleft v$ and u with v are consecutive in $\triangleleft$. We modify $<$ accordingly.

For each pre-represented interval $I(v_i)$ we need to set the value $lbound(l_i)$ to l_i. For other vertices in C_t we set $lbound(l_j) = E_{t-1} + \varepsilon$ where E_{t-1} is the rightmost endpoint of the previous component. We change this value to E_{t-1} if either v_j or each interval in C_{t-1} reaching E_{t-1} is open. All these operations are included when we talk about running the linear program. Our program has variables $l_1, \ldots, l_k$ representing the left endpoints of vertices in C_t.

$$\begin{aligned}
\min \quad & E_t \equiv l_k + 1 && && (3.1a)\\
\text{s.t.} \quad & l_i \le l_{i+1}, && \forall i \in \{1, \ldots, k-1\}, && (3.1b)\\
& l_i = lbound(l_i) && \forall v_i \text{ pre-represented}, && (3.1c)\\
& l_i \ge lbound(l_i) && \text{otherwise}, && (3.1d)\\
& l_i \ge l_j - 1 && \forall v_i v_j \in E(G), v_i < v_j, \text{ if both are closed}, && (3.1e)\\
& l_i \ge l_j - 1 + \varepsilon && \forall v_i v_j \in E(G), v_i < v_j, \text{ if at least one is open}, && (3.1f)\\
& l_i + \varepsilon \le l_j - 1 && \forall v_i v_j \notin E(G), v_i < v_j, \text{ if both are closed}, && (3.1g)\\
& l_i \le l_j - 1 && \forall v_i v_j \notin E(G), v_i < v_j, \text{ if at least one is open}, && (3.1h)\\
& l_k = l_j + 1 = l_i + 2 && \forall v_i < v_j < v_k \text{ closed of the same } K_{1,3}, && (3.1i)\\
& l_i = l_j && \forall v_i, v_j \text{ open and middle of the same } K_{1,3} && (3.1j)
\end{aligned}$$

Lemma 12. *Let the representation of C_{t-1} be fixed such that its rightmost endpoint has value E_{t-1}. Each correct ε-grid representation of the component C_t with left-to-right order $v_1 < \cdots < v_k$ which is on the right of C_{t-1} satisfies the*

constraints (3.1c) - (3.1j). Conversely if the program returns a representation, it is correct.

Proof. Constraints of type (3.1b) make the representation respect $<$. Constraints (3.1c) and (3.1d) must be satisfied as they make sure the pre-represented intervals remain in their positions and others are represented on the right from the previous component. Constraints (3.1e) and (3.1f) make sure that the intervals representing adjacent vertices are at least touching, or intersecting if one of them has open endpoints. Conversely (3.1g) and (3.1h) keep intervals of non-adjacent vertices at least ε away or let them at most touch in case at least one is open. Finally (3.1i) and (3.1j) guarantee the $K_{1,3}$'s are represented in the only possible way. □

We restrict ourselves to ε-grid representations when using linear programming. However, in the following sections we talk about and prove results for general $\mathcal{U}^{\pm}$ representations.

3.2 Open intervals

3.2.1 Open intervals of $K_{1,3}$'s

Identification of open intervals of $K_{1,3}$'s is described in Chapter 1. For the rest of this chapter we assume we have removed the open vertices of $K_{1,3}$'s from G to obtain a $K_{1,3}$-free unit interval graph G' and calculated the order $\triangleleft$ (and its reversal) for each component. If any of them were pre-represented, we represent the corresponding middle vertices in their places to keep this information.

3.2.2 Open intervals forced by the pre-representation

Unlike in $\mathcal{PI}^{\pm}$ distances matter in this case. Especially the lengths of gaps between two represented intervals. For two non-intersecting represented intervals $I(u)$ and $I(v)$ such that $I(u)$ is on the left from $I(v)$, we say the *length of the gap* formed by these intervals is $l_v - r_u$

Definition 8. *If the length of a gap between two represented intervals is an integer, we call this gap an* integer gap*. We call intervals forming this gap* boundary intervals*.*

We say an integer gap is split *by already represented interval within this gap if it has integer distances from the boundary intervals.*

It might happen that there are some intervals already represented inside this gap but they don't have integer distance from boundary intervals. In case they do,

we will split the gap into smaller ones and focus mainly on unsplit ones. We are going to discuss cases depending on whether the pre-representation contains integer gaps or not.

Lemma 13. *In case there is no integer gap between any two pre-represented intervals, the intervals locally forced to be open are only those already pre-represented and the open intervals of $K_{1,3}$'s.*

Proof. Suppose a vertex v is not the open vertex of any $K_{1,3}$ and it is not pre-represented and yet it is locally forced to be open. If $I(v)$ has no closed intervals ending at its endpoints we can change it to closed. Therefore we assume there is a closed interval $I(u)$ ending, say, at its right endpoint. If there is no interval ending at the left endpoint of $I(v)$, we can move it slightly to the left and change it to closed. If there is no interval ending at the right endpoint of $I(u)$, we can move it slightly to the right and change $I(v)$ to closed. Therefore we assume there is $I(x_1)$ at the right endpoint of $I(u)$ and $I(x_2)$ at the left endpoint of $I(v)$.

We could keep on constructing new intervals at the endpoints and obtain an infinite sequence x_n, but since our graph is finite, we either run out of vertices, which means we can slightly shift a sequence of intervals and change $I(v)$ to closed, or we reach endpoints of pre-represented vertices in both directions. However, the latter means these pre-represented intervals have an integer distance which is a contradiction.

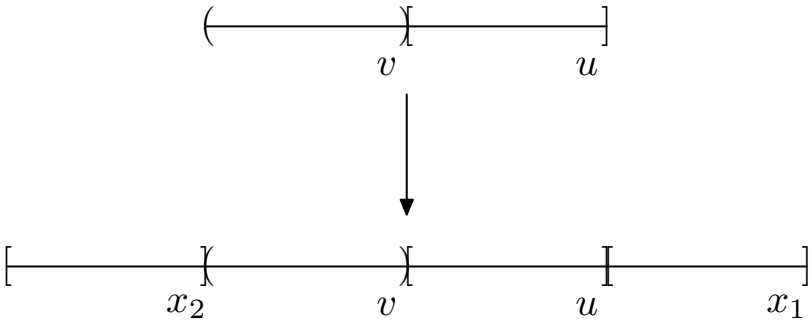


Figure 3.1: constructing a sequence of intervals

□

3.3 Integer gaps and independent sets

We say that a set of intervals covers a distance d in a representation R if d is the difference of the rightmost and the leftmost endpoints of intervals in this set. In this sense each interval covers the distance 1.

Lemma 14 (Minimal length of independent set)**.** *Each independent set of size k covers at least distance k in a representation. If it covers distance exactly k, any two consecutive intervals must share an endpoint and at least one of the intervals in each such pair must be open.*

Proof. Since the length of every interval is one and they can share at most endpoints this condition must hold. □

Definition 9. *We say a k-tuple $I(v_1)$, $I(v_2)$, ..., $I(v_k)$ is* filling an integer gap *of size k formed by boundaries $I(b_l)$ and $I(b_r)$ in a representation R, if $l_i = l_{b_l} + i$ for every $i \in \{1, 2, \ldots, k\}$.*

We say an interval participates in filling an integer gap *if it belongs to a k-tuple filling this gap.*

If the pre-representation is extendible, we say an integer gap g is forced to be filled *in a given way if for each representation R extending the pre-representation there exists a k-tuple filling g in this way, such that this k-tuple fills this gap also for each $R' \sim_{sim} R$.*

A given way for filling an integer gap refers to types of intervals participating in filling this gap, e.g. a gap might be forced to be filled by (only) closed or open intervals, or using at least one closed or open interval.

Corollary 15. *A non-pre-represented interval locally forced to be open that is not the open interval of any $K_{1,3}$, must participate in filling an integer gap.*

At this point it is good to realize what does filling an integer gap mean.

Definition 10. *If two closed intervals form a gap of the length k it means the shortest path between them must consist of at least $k+2$ vertices (including themselves). We call each such path of the length $k+2$ a* gap path.

Observation. If intervals forming a gap belong to different components, they cannot have a gap path.

Lemma 16. *An integer gap formed by two closed intervals is forced to be filled by closed intervals if and only if it has a gap path.*

Lemma 17. *If an integer gap g formed by boundary intervals $I(b_l)$ and $I(b_r)$ is filled by k open intervals, it means these intervals correspond to an independent set of k vertices. Furthermore also boundary vertices are independent with respect to them, therefore we have an independent set of size $k+2$.*

Consider a set $X_g \equiv \{x \in V(G'); b_l \leq' x \leq' b_r\}$. As mentioned in Chapter 1, since we can order vertices only within components, $x \leq' y$ means either $x \leq y$ within a component or $C_x \blacktriangleleft C_y$ where $x \in C_x$ and $y \in C_y$.

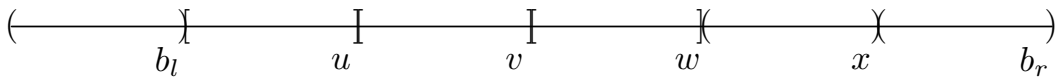
However, for some components the order $\triangleleft$ (and also $<$) is not given uniquely and we can choose from two options. Assume $b_l \in C_l$ and $b_r \in C_r$. Then for all vertices $x \in C_x$ such that $C_l \blacktriangleleft C_x \blacktriangleleft C_r$, we have $b_l \leq' x \leq' b_r$. But if both $b_l \in C_l$ and $x \in C_l$, we either have $b_l \leq x$ or $x \leq b_l$ depending on whether $\triangleleft$ or its reversal is the left-to-right order for this component.

This means if $\triangleleft$ is given uniquely (or fixed) for both C_l and C_r, the set X_g is given uniquely, otherwise it might not be.

Definition 11. *Assume g is an integer gap of the length k formed by boundary intervals $I(b_l) \in C_l$ and $I(b_r) \in C_r$. For the fixed $\triangleleft_l$ for C_l and $\triangleleft_r$ for C_r, we call each independent set of size $k+2$ consisting of vertices from the set X_g defined as above a* gap independent set.

For the rest of this section we assume the left-to-right order $\triangleleft$ is fixed for each located component.

Example. Let us consider the situation displayed below. If the vertices b_l and b_r are pre-represented with a gap of the length 4 and u, v, w are the closed vertices of a $K_{1,3}$, $x > w$ and x and w, x and b_r, u and b_l are non-adjacent, this is the only possible representation.



In this case we have the gap filled with both open and closed intervals and we have an interval forced to be open even without the gap independent set of size 6, as long as the open vertex of this $K_{1,3}$ has already been removed.

From Lemma 14 we know if there is a gap independent set for some integer gap, we need some of the vertices in this set to be represented by open intervals. Our decision is to let all of them be open unless they are locally forced to be closed. This is justified by the following lemma.

Lemma 18. *If there exists a representation R extending the pre-representation, there also exists a representation $R' \sim_{ord} R$ such that all the intervals not locally forced to be closed are open.*

Proof. Assume we have intervals $I(v_1), \ldots, I(v_f)$ that are represented as closed in R but they are not locally forced to be closed. We will change the representation R into R' in steps. In the i-th step we change the closed interval $I(v_i)$ into an open one to obtain R_i. If we cannot change it, it would mean $I(v_i)$ was locally forced to be closed by R_{i-1}. Since this cannot happen we have $R' = R_f$ in which all of them are open. □

Another issue is that while three closed vertices of a $K_{1,3}$ don't form an independent set of size three and we removed the open ones from graph, they cover distance three in every representation. Therefore we need to check if this is the only special case that can happen and to modify our graph before looking for gap independent sets.

3.3.1 Forced closed intervals

We remind that an interval forced to be closed is always locally forced to be closed as well. Thus if we prove some conditions hold for an interval locally forced to be closed, they also hold for an interval forced to be closed.

Lemma 19. *An interval is locally forced to be closed only if at least one of the following holds:*

1. *it is pre-represented as a closed interval,*
2. *it is a closed interval of a $K_{1,3}$,*
3. *it is touching a locally forced open-closed pair and it is adjacent with the closed vertex of this pair and non-adjacent with the open one,*
4. *it lies on a gap path for an integer gap formed by closed boundaries.*

If an interval satisfies condition 1, 2, 4 or it is touching a forced open-closed pair in condition 3, it is forced to be closed.

Proof. Suppose $I(u)$ is locally forced to be closed and the first two conditions do not hold. For a representation R satisfying definition, there must be a closed $I(w)$ touching one of its endpoints because otherwise we can change $I(u)$ to open, let it be the right endpoint. If we can move $I(u)$ to the right, we may change it to open. If we cannot move it, there is either an open $I(v)$ with the same endpoints as $I(w)$ or some $I(x)$ closed holding the left endpoint of $I(u)$.

The former gives us the third option, because if there is a representation $R' \sim_{sim} R$ where $I(w)$ and $I(v)$ do not have the same endpoints, it necessarily means either $I(v)$ is strictly on the right from $I(u)$ or $I(w)$ has a nontrivial intersection with $I(u)$. Hence we can move $I(u)$ slightly to the right if needed and change it to open unless there is a closed $I(x)$ holding the left endpoint of $I(u)$. This either gives us the next option if both $I(w)$ and $I(x)$ are touching the endpoints of $I(u)$ for each such R' or it means each of $I(w)$ and $I(x)$ can have a nontrivial intersection with $I(u)$ while the other one is touching its endpoint. Thus the distance between $I(x)$ and $I(w)$ can be less than one hence we can find a representation R'' such that both $I(x)$ and $I(w)$ have nontrivial intersections with $I(u)$ and we may change $I(u)$ to open.

Finally, suppose $I(u)$ has closed intervals ending at both endpoints that we cannot move. Since $I(u)$ is not a closed interval of a $K_{1,3}$, it doesn't contain any open intervals preventing them from moving. This means either one of the closed vertices touching its endpoints was pre-represented or something is holding them, which leads, analogically as in the prove of Lemma 13, to a sequence we can shift unless there is an integer gap filled by closed intervals.

Conversely, interval $I(u)$ satisfying conditions 1 and 2 is obviously forced to be closed. As for condition 3, for a forced open-closed pair $I(v)$ open $I(w)$ closed, we know $I(u)$ satisfying it is touching only an endpoint of its neighbour $I(w)$, hence it is forced to be closed. We will see later that each locally forced open-closed pair is actually a forced open-closed pair. Finally, all intervals forming a gap path touch only with endpoints thus all of them are forced to be closed. □

Definition 12. *A* special P_4 *is an induced P_4 that consists of two forced open-closed pairs.*

For each induced P_4 we distinguish its end and middle vertices. For a special P_4, its end vertices are represented by open intervals and middle vertices by closed intervals.

Lemma 20. *If there exists a representation R extending the pre-representation in which a P_4 is represented by two open-closed pairs formed by intervals $I(x)$, $I(u)$, $I(w)$ and $I(v)$ in this order with u and w being the middle vertices, x with u and w with v are consecutive in $\lhd$. We also have $N_G[x] \subsetneq N_G[u]$ and $N_G[v] \subsetneq N_G[w]$ up to twins.*

Lemma 21. *If an open $I(v)$ and a closed $I(u)$ form a forced open-closed pair, u, v either belong to a $K_{1,3}$ or a special P_4. If an interval $I(x)$ is forced to be closed implied by this pair, then x belongs to a $K_{1,3}$ or a special P_4 as well.*

Furthermore, every special P_4 is implied either by touching pre-represented open intervals, a $K_{1,3}$ or by open intervals participating in filling an integer gap.

Proof. We know there must be a closed interval touching an endpoint of $I(u)$ otherwise we can change it to open. Let's call it $I(w)$ and assume it is touching the right endpoint of $I(u)$. Now to prevent $I(u)$ from moving to the right we either have to hold it from the left side as well, which gives us a $K_{1,3}$, or there must be an open interval forming a forced open-closed pair with $I(w)$ which gives us a special P_4.

If $I(x)$ is forced to be closed implied by this pair, it means x is a neighbour of u and not of v so it has to touch an endpoint of $I(u)$. If $I(v)$ was the open interval of a $K_{1,3}$, $I(u)$ is the middle and $I(x)$ is a closed interval of the same $K_{1,3}$. If u and v belong to a special P_4, either $I(x)$ plays the role of $I(w)$, which means it belongs to a special P_4, or it is touching the other endpoint of $I(u)$ which creates a $K_{1,3}$ together with $I(w)$.

Finally, a special P_4 contains two intervals that are forced to be open and Lemma 13 and Corollary 15 tell us how it can be done. □

Definition 13. *We say an induced P_4 is a* locally special P_4 *if there exists a representation R extending the pre-representation such that for each representation $R' \sim_{sim} R$, this P_4 is represented by two touching open-closed pairs.*

Lemma 22. *Each locally forced open-closed pair that is not a forced open-closed pair belongs to a locally special P_4. Each interval only locally forced to be closed by this pair also belongs to a locally special P_4. Each open interval of a locally special P_4 is locally forced to be open.*

Proof. Assume an open $I(v)$ and a closed $I(u)$ form a locally forced open-closed pair. Take some representation R from the definition. There must be a closed interval touching one of its endpoints, otherwise v and u are twins. Assume it is the right endpoint and let us call this interval $I(w)$. Now we need to prevent $I(u)$ from moving to the right. The first option is a closed interval holding $I(u)$ from the right, which means $I(u)$ is the middle interval of a $K_{1,3}$ and thus our pair is a forced open-closed pair. The second option is an open interval with the same

endpoints as $I(v)$, which means we have a P_4 represented in the required way in R. If we can move any of those four intervals without moving the others to obtain a representation $R' \sim_{sim} R$, we can also move the open-closed pair which is a contradiction, hence this P_4 is locally special.

From the proof of lemma above we can see if we have a locally forced closed interval touching this pair, it again belongs to either a $K_{1,3}$, which means it is even forced to be closed, or to a locally special P_4.

Finally we prove an open interval of a locally special P_4 is locally forced to be open. Let $I(x)$ be the right open interval of this P_4 in R. If we can move representation only in a way that whenever $I(x)$ can be changed to closed it can remain open at that position, we have a contradiction since we created a locally similar representation where a locally special P_4 is not represented by two touching open-closed pairs.

Therefore we suppose there is a representation R'' such that $I(x)$ cannot remain open at its position and all the other intervals are of the same type as in R. Hence there is a closed interval $I(y)$ ending at an endpoint of $I(x)$. We can move $I(x)$ slightly towards $I(y)$ unless $I(x)$ is being held or there is also an open interval $I(z)$ with the same endpoints as $I(y)$. The former leads to a gap path, implying $I(x)$ is forced to be closed, which is a contradiction with $I(x)$ being open in R. The latter means whenever $I(y)$ and $I(z)$ do not have the same endpoints we can change $I(x)$ to the open and this can happen only if it lies in its P_4 but in such case we cannot keep it closed. Therefore if $I(x)$ is closed, $I(y)$ and $I(z)$ have the same endpoints. Thus $I(y)$ with $I(z)$ form a locally forced open-closed pair and, since $I(x)$ is touching this pair and it cannot belong to a $K_{1,3}$, $I(x)$ must be a closed interval of a locally special P_4 in R''.

We know $I(x)$ is the right open interval of a locally special P_4 in R, let $I(u)$ be the right closed interval of this P_4. Then $N_G[x] \subsetneq N_G[u]$. In case x is the left middle vertex of a locally special P_4 in R'', the left end vertex of this P_4 is consecutive with x in $\lhd$ and to its left, but we already know from R it is (a twin of) u. Thus $N_G[u] \subsetneq N_G[x]$ which is a contradiction.

In case x is the right middle vertex of a locally special P_4 in R'', we take a look at the left closed interval of the P_4 in R and call it $I(w)$. Since it is not a neighbour of x we know for the left end vertex w'' of the locally special P_4 in R'' we have $w \leq w''$. However, for every such vertex w'' except for (twins of) w, $I(w'')$ has a nontrivial intersection with $I(x)$ in R. Thus w'' must be w but then $I(w)$ is open in R'' which is a contradiction because it should be of the same type as in R. □

These three lemmas give us important knowledge to find the intervals locally forced to be closed. However, finding an interval forced to be closed, especially if we are able to represent it, might lead to another forced closed intervals. For example in case one of the boundary vertices of an integer gap is open, there will be no forced closed intervals implied by a gap path, but if we find a forced closed interval with the same endpoints, there might be. This means we have to repeatedly look for forced closed intervals until no new intervals are found or

represented.

Algorithm 4: Finding intervals forced to be closed using conditions 1, 2 and 4 from Lemma 19

input : A graph G' with the pre-representation, the $K_{1,3}$'s found and their open vertices removed

output: A set $C \subseteq V(G)$ of some vertices forced to be closed and a representation for some of them

```
repeat
    if I(v) is already represented and closed then add v to C;
    if v is a closed vertex of a K_{1,3} then
        add v to C;
        represent all the closed intervals of this K_{1,3} if it can be done
         uniquely;
    end
    foreach unchecked (possibly split) integer gap with closed endpoints
     within the same component do
        if it has a gap path then
            add vertices on this path to C;
            fill this gap by closed intervals corresponding to these vertices;
        end
    end
until no new intervals are forced to be closed or represented;
```

Lemma 23. *Algorithm 4 runs in polynomial time, all the vertices in C are forced to be closed and each interval it represents is forced to be in this position.*

Proof. We can see this algorithm checks conditions 1, 2 and 4 of Lemma 19 in a loop. It makes only linearly many loops, since in each loop at least one interval must be either forced or represented. Conditions 1 and 2 can be checked in linear time since the $K_{1,3}$'s have already been found. Condition 4 can be checked in polynomial time as well since for a gap path to exist both boundary vertices must lie within the same component hence we have the order $\triangleleft$ given uniquely for this component and now finding the shortest path between them can be done in polynomial time.

As proven in Lemma 19 these conditions are sufficient hence all the vertices in C are forced to be closed. Finally, we represent the closed vertices of a $K_{1,3}$ only if we have the order $\triangleleft$ for its component given uniquely and a gap path for a given gap has a unique representation. □

Lemma 24. *If an integer gap g of the length k has a gap path, then all the open vertices participating in filling g come from $K_{1,3}$'s.*

Proof. As we have shown, a gap path is forced to be represented by $k+2$ closed intervals touching with endpoints. Suppose $I(v)$ is an open interval participating in filling g. Then there exists a closed interval $I(u)$ with the same endpoints as

$I(v)$ and closed intervals $I(w)$ and $I(x)$ touching it from either side forming a $K_{1,3}$. □

As we can see from Lemma 21 an interval (locally) forced to be closed implied by condition 3 of Lemma 19 is either a closed interval of a $K_{1,3}$, which are checked by Algorithm 4, or a closed interval of a (locally) special P_4. These P_4's are again implied either by $K_{1,3}$'s which we have already checked, by two touching pre-represented open intervals which we can check again easily, or by intervals locally forced to be open that participate in filling integer gaps.

Algorithm 5: Finding forced closed intervals using condition 3 from Lemma 19

input : A graph G', an unsplit integer gap g of the length k formed by $I(b_l)$ and $I(b_r)$, an independent set I_S of size $k+2$ consisting of vertices satisfying $b_l \leq' x \leq' b_r$, a vertex $u \in I_S$

output: TRUE if u is belongs to a (locally) special P_4 forced by g and FALSE otherwise

$w_1 \equiv$ the greatest neighbour of u in $\lhd$;
$x_1 \equiv$ the vertex immediately to the left from u in $\lhd$;
$v_1 \equiv$ the vertex immediately to the right from w_1 in $\lhd$;
$y_1 \equiv$ the smallest vertex in $\lhd$ in I_S such that $u \leq' y_1 \leq' b_r$;
if v_1, w_1, x_1, y_1 *are defined* **and** $N_G[x_1] \subseteq N_G[u] \setminus w_1$ **then**
  $I_S^1 \equiv I_S \setminus \{u, y_1\} \cup \{v_1, x_1\}$;
  if I_S^1 *is independent set* **then return** *TRUE*;
end

$w_2 \equiv$ the smallest neighbour of u in $\lhd$;
$x_2 \equiv$ the vertex immediately to the right from u in $\lhd$;
$v_2 \equiv$ the vertex immediately to the left from w_2 in $\lhd$;
$y_2 \equiv$ the greatest vertex in $\lhd$ in I_S such that $b_r \leq' y_2 \leq' u$;
if v_2, w_2, x_2, y_2 *are defined* **and** $N_G[x_2] \subseteq N_G[u] \setminus w_2$ **then**
  $I_S^2 \equiv I_S \setminus \{u, y_2\} \cup \{v_2, x_2\}$;
  if I_S^2 *is independent set* **then return** *TRUE*;
end
else return *FALSE*;

Lemma 25. *Algorithm 5 runs in polynomial time and it correctly determines whether vertex u belongs to a special P_4 implied by an unsplit integer gap g.*

Proof. Suppose u belongs to a (locally) special P_4 implied by the gap g and it is its left middle vertex. This corresponds to the case I_S^1, we get the case I_S^2 if u is the right middle vertex. Now let an open $I(v)$ and a closed $I(w)$ be the (locally) forced open-closed pair to its right and $I(x)$ be the open vertex forming a (locally) forced open-closed pair with $I(u)$. Since they belong to an induced P_4, x, u, w and v must lie within the same component thus they are comparable by $\lhd$ for this component. It is clear that x with u and w with v are consecutive in $\lhd$ because they have the same endpoints and they cannot be twins. Also w is the greatest neighbour of u in $\lhd$ since it is touching only its endpoint. Since $I(x) \subset I(u)$

and x is not a neighbour of w also $N_G[x] \subseteq N_G[u] \setminus w$ holds. It is clear that by replacing u and y in a gap independent set we get another gap independent set as x and u formed an open-closed pair. Also $v \leq' y$ holds because v is the first non-neighbour of u on its right. This means u satisfies all the conditions and thus is found by the algorithm.

Conversely, suppose we found vertices v, w and x satisfying conditions in the algorithm. Since $I(u)$ and $I(y)$ are consecutive in a gap independent set they have to touch with endpoints in each representation R extending the pre-representation. Therefore also $I(x)$ and $I(v)$ have to touch with endpoints since we can replace y by v and u by x and we get another gap independent set. From this $I(x)$ has the same endpoints as $I(u)$ and $I(v)$ as $I(y)$. Finally, also $I(w)$ has the same endpoints as $I(v)$ because $I(x) \triangleleft I(w) \triangleleft I(v)$ (or reversed) and x and w are non-adjacent. Then both $I(u)$ and $I(w)$ are forced to be closed as they are adjacent but touching only with endpoints and $I(x)$ and $I(v)$ are forced to be open, which finishes the proof of correctness.

As for the time complexity, every operation can be performed in at most linear time. □

Corollary 26. *An interval is locally forced to be closed if and only if it is forced to be closed.*

Proof. We know intervals only locally forced to be closed have to satisfy condition 3 in Lemma 19, thus they are touching only locally forced open-closed pairs. To avoid being forced, those pairs and also locally forced closed intervals have to come from locally special P_4's. For each locally special P_4 there is a representation which makes it satisfy conditions of Algorithm 5. However, each 4-tuple satisfying those conditions is already a special P_4, hence its closed intervals are forced to be closed. □

3.3.2 Subgraphs with a fixed length of representation

We need to find all the subgraphs contributing distance into filling an integer gap without contributing the corresponding amount of vertices into an independent set. The contributed distance must come from forced closed intervals, because open intervals contribute into an independent set. This knowledge will provide us with a natural extension of Lemma 14.

The most important example are three closed vertices of a $K_{1,3}$ that contribute to an independent set only with two vertices since we removed the open vertex of each $K_{1,3}$ but they can still participate in filling a gap covering distance three. We save the information they belong to a $K_{1,3}$ but we remove the edges between the closed vertices of the same $K_{1,3}$. This way we can still use them as a part of an independent set and this modification doesn't affect other vertices. The same goes for boundary vertices if they are closed vertices of some $K_{1,3}$'s. Important note is that we keep the order $<$ calculated before this operation.

Lemma 27. *The only subgraphs contributing to filling an integer gap without a gap path that do not contribute to an independent set with the number of vertices corresponding to the length of representation are $K_{1,3}$'s.*

Proof. For the length 2 forced closed intervals come only from special P_4's but they include independent set of size 2. For the length 3 and greater we can always shrink two closed intervals closer together unless one of them contains an open interval with the same endpoints. Thus all such closed intervals must belong to $K_{1,3}$'s. □

For the following theorem we use a graph G'' obtained from G' by removing edges between closed vertices of the same $K_{1,3}$'s.

Theorem 28. *Given two intervals $I(b_l)$ and $I(b_r)$ forming an unsplit integer gap of the length $k \geq 2$ without a gap path, this gap is forced to be filled using at least one open interval if and only if it has a gap independent set in graph G''.*

If there is a representation R' extending the pre-representation, there is also a representation $R \sim_{ord} R'$ such that all the vertices forming this independent set except those forced to be closed are represented by open intervals. We call these open intervals marked open by g (also for a split gap).

In case there is an independent set of size $k + 3$ or greater, our representation cannot be extended.

Proof. An independent set of size $k + 2$ needs to fill at least distance $k + 2$ by Lemma 14. Since this distance is at least 4 we need to use at least one open interval. For $k = 1$ it might happen that this gap is filled by 3 closed intervals of a $K_{1,3}$, otherwise we must use at least one open interval as well. If there is an independent set of a greater size we have to represent it outside this gap which is a contradiction with the left-to-right order.

Conversely, we want to show that whenever we are forced to fill a gap of a given length using at least one open interval there is such an independent set. According to Lemma 19 forced closed intervals participating in filling this gap might be pre-represented, come from $K_{1,3}$'s, special P_4's or from a gap path. However, pre-represented intervals would split the gap, special P_4's contain two open intervals contributing to an independent set and we assumed our gap doesn't have a gap path. Hence we can assume all the closed intervals come from $K_{1,3}$'s.

Take a representation R and a k-tuple filling the gap including at least one open interval from the definition. The only way to lose independence is that two closed intervals share an endpoint. Since we allowed this for the closed intervals of the same $K_{1,3}$, they must come from different ones. If we can shrink them closer together, there is a representation $R' \sim_{sim} R$ in which this k-tuple is not filling the gap. If we cannot, they either belong to a gap path or the last closed interval of one of them includes an open interval with the same endpoints, which gives us a new $K_{1,3}$. □

Lemma 29. *Suppose a non-pre-represented interval $I(v)$ is marked open by some integer gap g and it is not the open vertex of any $K_{1,3}$. Then it is marked open by some unsplit integer gap.*

Proof. Assume $I(v)$ is marked open by a gap g, hence it participates in filling g. If g is unsplit then we are done. If g is split and contains no unsplit gaps, it means it is completely filled by intervals forming a gap path thus $I(v)$ is the open vertex of a $K_{1,3}$ by Lemma 24, otherwise g contains some unsplit gap g' that $I(v)$ is participating in filling, thus $I(v)$ is marked open by g'. □

Corollary 30. *If an interval is locally forced to be open, at least one of the following holds:*

- *it is pre-represented as open,*
- *it is the open interval of a $K_{1,3}$,*
- *it is marked open by an unsplit integer gap.*

Proof. Suppose an interval locally forced to be open doesn't satisfy the first two conditions. Then by Corollary 15 it must participate in filling an integer gap. This means it belongs to a k-tuple filling this gap and since it is locally forced to be open, this gap is forced to be filled using at least one open interval. Hence by Lemmas 28 and 29 it is marked open by an unsplit integer gap. □

From now on, if we refer to independent sets we will be talking about them in modified graph G'' unless stated otherwise. This also means $\alpha(G) \equiv \alpha(G'')$.

To determine whether a particular vertex belongs to any independent set of size $k+2$ we can remove its neighbours from the graph because then it is necessarily contained in the maximal independent set. If we do this for each vertex, we increase time by a linear factor, so it remains polynomial.

Before proceeding with the algorithm, we proof an important fact, that saves us a lot of time when determining the intervals forced to be closed using Lemma 19. We first run Algorithm 4. Then once we look for a gap independent set to obtain marked open intervals we can also find intervals forced to be closed determined by condition 3. Our lemma says we can stop here.

Lemma 31. *Assume we ran Algorithm 4, to obtain a representation R'. Then each forced closed interval $I(u)$ implied by an unsplit integer gap g such that u belongs to a gap independent set for some integer gap is found after running Algorithm 5 for the modified graph G'', the gap g and the vertex u.*

Proof. Suppose we have applied conditions 1, 2 and 4 as many times as needed and condition 3 once to a vertex u participating in filling an integer gap g and it wasn't labeled as forced to be closed yet.

It cannot be forced to be closed by conditions 1 and 2 anymore because it would have to be pre-represented or in a $K_{1,3}$ and this doesn't change.

In condition 3 vertices v, w, x and y are determined uniquely by the vertex u and the order $\lhd$. Therefore the only thing that can change is that they didn't satisfy conditions in Algorithm 5 and now they do. For such a thing to happen we need to split the original gap g existing in R' and this has to be done by condition 3 for some other vertex u'. But if v and x were in a gap independent set for a new gap g' they necessarily had to be also in a gap independent set for g, otherwise we get a contradiction with Lemma 14 or 28.

Condition 4 might find some intervals forced to be closed, because new represented closed intervals might create shorter gaps or gaps ending with closed intervals that ended with open intervals before. Hence we need to check only the intervals that lie both on a gap path and in a gap independent set.

If an interval lies on a gap path, it is forced to be closed and has intervals forced to be closed touching each of its endpoints, and since it is in a modified independent set it has intervals locally forced to be open touching each of its endpoints, otherwise it is in some $K_{1,3}$, but in such case we already know it is forced to be closed.

Suppose $I(u)$ is such a closed interval in a gap of the length 1. Then there are closed $I(w_1)$ and $I(w_2)$ and open $I(v_1)$ and $I(v_2)$ as mentioned. These form two locally forced open-closed pairs. They either have been pre-represented or some pair comes from a $K_{1,3}$ or a special P_4. In the first case $I(u)$ was already found on the first run of condition 4, in the second one $I(u)$ lies in a $K_{1,3}$ - either directly or implied by a special P_4 and a closed interval touching it.

If a gap has the length at least two then a gap path itself consists of at least four closed intervals, therefore at least one of the mentioned open intervals touching $I(u)$ doesn't have the same endpoints as the boundaries of this gap, thus it belongs to a $K_{1,3}$ hence also $I(u)$ does.

□

Algorithm 6: Determining intervals marked open by an integer gap

input : An unsplit integer gap g formed by $I(b_l)$ and $I(b_r)$, a set of vertices $X_g \equiv \{x \in V(G');\ b_l \leq' x \leq' b_r\}$ with the order $<'$ and a vertex $u \in X_g$, the representation R' obtained by running Algorithm 4

output: OPEN if $I(u)$ is marked open by g, CLOSED if it is forced to be closed implied by R' or g and NOT FORCED if neither holds

$k \equiv l_{b_r} - r_{b_l}$;
if $u \in C$ *from Algorithm 4* **then return** *CLOSED*;
modify G' into G'' by removing edges between the closed vertices of the same $K_{1,3}$'s;
foreach *neighbour of* u **do** remove it from G'' to obtain G''';
if *exists an independent set of size* $k+2$ *in* $G'''|_{X_g}$ **then**
    run Algorithm 5 for G'', g, u;
    if *it returns TRUE* **then return** *CLOSED*;
    else return *OPEN*;
end
else return *NOT FORCED*;

Lemma 32. *Algorithm 6 returns correct result and works in polynomial time.*

Proof. If our algorithm returns result it is correct as proven by Lemmas 19, 31 and 28. It is also clear it will always return some value. As for running time, each vertex has only linearly many neighbours, there are only polynomially many $K_{1,3}$'s and most importantly the maximal independent set in interval graphs can be solved in polynomial time. Therefore each operation takes at most polynomial time and is run only polynomially many times which finishes the proof. □

To sum this part up, we will represent a vertex u by an open interval if it is pre-represented such way or it is the open vertex of a $K_{1,3}$ or if it is marked open by some unsplit integer gap and it is not forced to be closed by any condition in Lemma 19. Otherwise we will represent it by a closed interval.

This means we should run Algorithm 6 for each unsplit integer gap g such that $u \in X_g$ and represent it by an open interval if and only if OPEN was returned at least for one gap and CLOSED for no gap.

3.4 Minimizing the rightmost endpoints of components

The process described above works in certain scenarios, particularly when both boundary vertices belong to the same component, because that determines its order $\lhd$, but if we are not able to determine the subset X_g in which we are looking for a gap independent set, we cannot use it. The key here seems to be to minimize the rightmost endpoints of located components from left to right without this knowledge. This way we either obtain a representation or we know the

pre-representation cannot be extended. We denote the rightmost endpoint of a component C_t by E_t. From now on we will be talking about ε-grid representations as we need to use linear programming.

Lemma 33. *A located component C such that $C \blacktriangleleft C_l$ where $b_l \in C_l$ and $I(b_l)$ is the left boundary of the leftmost integer gap doesn't contain any intervals locally forced to be open implied by integer gaps and its rightmost endpoint can be minimized by linear programming at this stage.*

Proof. Let $H = \bigcup_{C \blacktriangleleft C_l} C$. Then H contains no integer gaps and by Lemma 13 we can determine intervals forced to be open. This is along with the order $\triangleleft$ (and its reversal) for each component which we obtain after removing the open vertices of the $K_{1,3}$'s sufficient to run the linear program. □

Lemma 34. *If an integer gap exists between intervals of two components $C_l \blacktriangleleft C_r$ and it has a gap independent set for the orders $\triangleleft_l$ for C_l and $\triangleleft_r$ for C_r, then for each representation R extending the pre-representation with the orders $\triangleleft_l$ for C_l and $\triangleleft_r$ for C_r, the representation of each component C_x such that $C_l \blacktriangleleft C_x \blacktriangleleft C_r$ ends and starts with an interval belonging to this set and each two such consecutive components share endpoints.*

Proof. Since we have allocated exactly as much space as we need, in case there is any component C_x where the last interval is not participating in filling this gap it would necessarily overstep into the next gap-filling interval but this is already in the next component which is a contradiction. □

At this point we will use our knowledge about independent sets extended by section 3.3.2.

Observation (Relative position of integer gap). There are four possibilities how the boundaries of an integer gap can be positioned relative to a component:

1. Both b_l and b_r lie in C_t. This gives us the unique order $\triangleleft$ for C_t and we can run Algorithm 6 for each vertex within this gap.

2. Neither of them lies in C_t. In case this gap has a gap independent set, its representation within C_t has to start at the rightmost endpoint of the previous component as proven by the previous lemma, it cannot use more space than its size and has to end at the rightmost endpoint of C_t.

3. The left boundary b_l lies in some previous component and the right boundary $b_r \in C_t$. In case this gap has a gap independent set, its representation within C_t has to start at the rightmost endpoint of the previous component and fill the gap until b_r precisely.

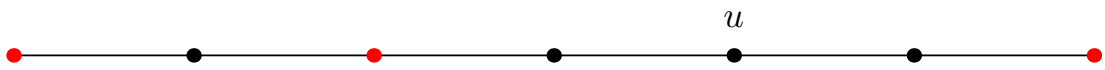
4. The left boundary $b_l \in C_t$ and the right boundary b_r lies in some next component. In case this gap has a gap independent set, its representation within C_t has to end precisely at the rightmost endpoint of C_t.

If possibility 2 happens at the same time as 3 or 4, then the integer gap given by possibility 2 can be split into smaller gaps therefore we don't need to discuss it. Also as proven by Lemmas 34 and 30 in possibility 3 the boundary interval in C_t is given uniquely, because it is the interval closest to the rightmost endpoint of the previous component with an integer distance from it. The same goes for possibility 4 but we don't know the position of the leftmost endpoint of the next component yet.

For each of those possibilities we can check whether there exist intervals forming integer gaps with given properties. If they don't, we can skip that possibility. However we cannot check existence of a gap independent set for possibilities 2 and 4, thus we will discus both the case it exists and the case it does not.

3.4.1 Finding lower bounds for the rightmost endpoint using independent sets

Example. Let C_t be a component as displayed below where the only pre-represented vertex is u with the interval $I(u) = [x, x+1]$. Assume $\triangleleft$ orders the vertices from left to right as in the picture.



We can see the red vertices form together with u an independent set of size 4. Depending on whether we use $\triangleleft$ or its reversal, we end up with a lower bound for E_t either $x+2$ or $x+3$ but we also take space on the left from $I(u)$. Let C_{t-1} be the previous component of this graph. In case (the minimal value for) E_{t-1} is greater than $x-2$ we can only use reversal of $\triangleleft$ and therefore our lower bound for E_t would be $x+3$.

For the general case, suppose we are minimizing the rightmost endpoint E_t of a component C_t with pre-represented vertices $u_1, u_2, \ldots, u_m$ where $I(u_i) = [x_i, x_i + 1]$ or $(x_i, x_i + 1)$ for each $i \in \{1, \ldots, m\}$. In case there are at least two non-twin intervals without the same endpoints pre-represented, we have $\triangleleft$ given uniquely for this component, otherwise we have to consider its reversal as well. Thus we can extend those two orders to $<_1$ and $<_2$.

Before proceeding, we need to know whether we have enough space to represent vertices lying on the left from each pre-represented interval for each order.

In case when the order isn't given uniquely, only twins of one vertex and intervals with the same endpoints are pre-represented, thus we take the leftmost pre-represented interval $I(u)$ and define $left_j \equiv \alpha\{v \in C_t; v \leq_j u\}$.

- In case $left_j > l_u - E_{t-1} + 1$, the order $<_j$ cannot be used because we don't have enough space to represent the independent set of size $left_j$ on the left from $I(u)$,

- in case $left_j = l_u - E_{t-1} + 1$ we have an integer gap formed by the last interval of C_{t-1} and $I(u)$ with a gap independent set,
- in case $left_j < l_u - E_{t-1} + 1$ there is no gap independent set for the gaps given by possibility 3 from the relative position observation.

In case when the order $\triangleleft$ for C_t is given uniquely, it means we have several non-twin vertices pre-represented. In this case we have to calculate size of a maximum independent set formed by vertices on the left from each pre-represented interval. From groups of intervals forming integer gaps within this component we can take only the leftmost one. Analogically as above, if we find any vertex with an independent set larger than allocated space, we know the pre-representation isn't extendible. There can be only one interval where the equality holds and it gives us a gap independent set.

Now for each $<_j$ with enough space on the left we calculate a maximal lower bound for E_t, where

$$maxbound_j \equiv \max_{i \in \{1,\ldots,m\}} (x_i + \alpha_i);\ \alpha_i = \alpha\{v \in C_t; u_i \leq_j v\}$$

so for each pre-represented interval $I(u_i)$ we find the size of a maximum independent set on its right. It is clear from Lemma 14 this really is a lower bound for E_t. For convenience we assume $maxbound_1 \leq maxbound_2$ whenever both are defined and if only one is defined let it be $maxbound_1$. If neither of them is defined the pre-representation isn't extendible. Clearly, if possibility 4 of the relative position observation was to happen, we need either $E_t = maxbound_1$ or $E_t = maxbound_2$.

Let $maxbound_{prev} \equiv E_{t-1} + \alpha(C_t)$, which is a lower bound given by the rightmost endpoint of the previous component. Therefore we need to respect this bound also if bounds given by orders are smaller. It is clear that if possibility 2 of the relative position observation was to happen, we need $E_t = maxbound_{prev}$.

We will refer to finding bounds as described in this section as to Algorithm 3.4.1

3.4.2 Finding a minimal representation using bounds

We need to discuss the following possibilities:

(i.) $maxbound_{prev} \leq maxbound_1 \leq maxbound_2$

(ii.) $maxbound_1 \leq maxbound_{prev} \leq maxbound_2$

(iii.) $maxbound_1 \leq maxbound_2 \leq maxbound_{prev}$

(iv.) $maxbound_{prev} \leq maxbound_1$

(v.) $maxbound_1 < maxbound_{prev}$

Cases (i.) and (iv.) are very similar. In such cases we have to determine whether there exists a representation with $E_t = maxbound_1$. Before running the linear program we have to determine (marked) open intervals and intervals forced to be closed given by Algorithm 6. Clearly, possibility 2 of the relative position observation cannot happen (if $maxbound_{prev} = maxbound_1$ we can split such a gap). Possibilities 1 and 3 can be checked easily since X_g is given uniquely for each gap g coming from them for both $<_1$ and $<_2$. Now $E_t = maxbound_1$ needs to hold only if possibility 4 holds, otherwise we could move E_t to the right. Since $maxbound_1$ is given by some $I(u_i)$ and an independent set on its right, integer gap g coming from possibility 4 needs to start at $I(u_i)$. When restricted to C_t we have $X_g|_{C_t} = \{v \in C_t;\ u_i \leq_1 v\}$ with $k = \alpha_i$ which is enough to run Algorithm 6.

The linear program might return several results. If we find a representation such that $E_t = maxbound_1$, we know it is a minimal one. If we don't find such a representation, this either means we found no representation or $E_t > maxbound_1$, thus the pre-representation isn't extendible or possibility 4 cannot hold for $<_1$ if it is. In such case we change the intervals marked open only by the gap from this possibility to closed, allow the intervals forced to be closed implied only by the gap from this possibility to be open if they were marked open by another gap and run the linear program again. If we found no representation again, we have to move to $<_2$ or we say the pre-representation isn't extendible if $maxbound_2$ is not defined. If we found some representation we compare new E_t with the previous one and keep the smaller one. If it is smaller than $maxbound_2$, we stop here.

If we moved to $maxbound_2$ we approach it in a similar way but if we don't find any representation either with open or closed intervals we say the pre-representation isn't extendible. If we find one, we keep the new E_t if it is smaller than the previous smallest E_t.

Next, let us have a look at case (v.). In such case we have the fixed order $<_1$. Again we can determine the intervals (marked) open and forced to be closed implied by gaps coming from possibilities 1 and 3 easily. Possibility 4 cannot happen here. If possibility 2 was to happen, we know that we need $E_t = maxbound_{prev}$. Then a gap g given by this possibility has $X_g|_{C_t} = C_t$ and $k = \alpha(C_t)$ which is enough to determine the intervals marked open and forced to be closed using Algorithm 6.

There are two possible outcomes. If we find a viable representation for C_t, where $E_t = maxbound_{prev}$, we know it is a minimal one. If such a representation doesn't exist or $E_t > maxbound_{prev}$, it either means the pre-representation isn't extendible or possibility 2 cannot hold if it is. In such case we change the intervals marked open only by the gap from this possibility to closed, allow intervals forced to be closed implied only by the gap from this possibility to be open if they were marked open by another gap and run the linear program again. If we found some representation, we compare new E_t with the previous one and keep the smaller one. If we found no representation again, we say the pre-representation isn't extendible.

Case (iii.) is similar to (v.), but in this case we run this algorithm for both $<_1$ and

$<_2$. If for either of them we obtain a representation where $E_t = maxbound_{prev}$, we have found a minimal representation, otherwise we take the minimum of possible outcomes or we say the pre-representation isn't extendible if we found no representation for either order.

Finally, case (ii.) is somehow a mix of these cases. We first run the algorithm for case (v.). Now if we already found a representation where $E_t = maxbound_{prev}$ or $E_t < maxbound_2$, we can stop here. Otherwise we run the algorithm used in cases (i.) and (iv.) for $<_2$ and keep the new E_t if it is smaller than the previous minimum. Again if we found no representation in any of those cases we say the pre-representation isn't extendible.

We will refer to this subsection as to the minimizing algorithm or Algorithm 3.4.2.

3.5 Deciding REPEXT

At this point we have already described the algorithm minimizing the rightmost endpoint of each component. We will prove that if we manage to represent all located components using this algorithm, the pre-representation is extendible and otherwise it is not. When talking about some E_t being minimal, it is in context of reaching lower bounds mentioned in the previous section with the minimum being the greater of values $maxbound_{prev}$ and $maxbound_1$.

Theorem 35. *A pre-representation is extendible if and only if the minimizing algorithm finds a representation for each located component.*

Proof. It is clear that if we find a representation for each located component we can represent unlocated components on the right which gives us a representation for the whole graph.

Suppose we didn't find any representation. In such case there must be the first component we couldn't represent. Let's denote it C_t. In case C_t cannot be represented on its own with unlimited free space on either side we know the pre-representation cannot be extended. Otherwise it necessarily means either the endpoint E_{t-1} is too far on the right or the endpoint E_t reaches some pre-represented vertex of C_{t+1} for some t. Since we can check the second problem when calculating a representation for C_{t+1}, we only have to deal with E_{t-1} being too far.

If our algorithm found E_{t-1} that was minimal, it means we cannot improve it just by changing a representation of C_{t-1}. Thus either the pre-representation isn't extendible or already E_{t-2} was too far on the right. Therefore we find the leftmost component C_s where E_s wasn't minimal. If we don't find any such component it means the pre-representation isn't extendible because we cannot improve E_{t-1}.

Suppose E_s is not minimal and let there be a representation of C_s with a smaller

value E_s^* (which is not smaller than the minimum) for which the pre-representation is extendible to the whole graph. Apart from trying both orders which we do whenever possible and needed, we can change the value of E_s only by the choice of intervals that are open. Since E_s^* gives us a valid representation and E_s doesn't, it means some intervals are locally forced to be open. Due to Lemma 15 they must participate in filling an integer gap, because pre-represented open intervals and open intervals of $K_{1,3}$'s are forced to be open, thus they couldn't be closed in the representation giving us E_s.

Let's have a look at the relative position observation. Gaps determined by possibilities 1 and 3 have been already taken care of because we know E_{s-1} was minimal and integer gaps within components determine (marked) open vertices in such way that their representation cannot affect extendibility of the pre-representation. That means those vertices have to be marked open by gaps coming from possibilities 2 and 4. If one those possibilities was to happen, E_s would need to be minimal. However our algorithm already checked we cannot achieve such an E_s. Finally, this means either one of possibilities 2 and 4 happened and in such case the pre-representation isn't extendible or none of them happened so we don't need to change any intervals to open, thus reaching value E_s^* don't help us, which is a contradiction. □

Corollary 36. *Let G be a connected $\mathcal{U}^{\pm}$ graph with some pre-represented vertices. Then the value of the rightmost endpoint calculated by our algorithm cannot be improved.*

Proof. Let E be the value calculated by our algorithm and $\delta > 0$ such that there is a representation of G with the rightmost endpoint $E - \delta$. Then let G' be a graph obtained from G by adding a component consisting of a single vertex u pre-represented by $I(u) = [E - \frac{\delta}{2}, E - \frac{\delta}{2} + 1]$. It is clear that for the value E the representation of G' isn't extendible while for $E - \delta$ it is. Hence we have a contradiction because we proved if the pre-representation is extendible, our algorithm will find a representation extending the pre-representation. □

Algorithm 7: REPEXT for $\mathcal{U}^{\pm}$ graphs

input : A $\mathcal{U}^{\pm}$ graph G and a pre-representation Pr with rational endpoints
output: TRUE if Pr can be extended and FALSE otherwise

compute ε;
run Algorithm 2;
if *it returns FALSE* **then return** *FALSE*;
foreach *open vertex* v *of a* $K_{1,3}$ **do**
  if $I(v)$ *was pre-represented* **then** represent the corresponding middle vertex at the same place;
  remove it from G to obtain G';
end
run Algorithm 4 for G';
c_l = number of located components;
for *t=1* **to** c_l **do**
  obtain $\triangleleft$ for C_t and extend it to $<_1$ and $<_2$;
  run Algorithm 3.4.1 for C_t;
  run Algorithm 3.4.2 for C_t;
end
if *Algorithm found a representation for each located component* **then**
  return *TRUE*;
else return *FALSE*;

Corollary 37. *REPEXT for* $\mathcal{U}^{\pm}$ *is decidable in polynomial time.*

Proof. As proven by Theorem 35 Algorithm 7 returns correct result for REPEXT. When it comes to running time, computing ε and operations with $K_{1,3}$'s can be done in polynomial time. Algorithm 4 runs in polynomial time as we already proved. Computing orders is done in polynomial time as well. In Algorithm 3.4.1 the only non-trivial operations are calculating independent set, linear programming and Algorithm 6 that all run in polynomial time and we run them only polynomially many times. The same holds for Algorithm 3.4.2 which finishes the proof. □

Conclusion

The goal of this thesis was to understand and describe properties of $\mathcal{PI}^{\pm}$ and $\mathcal{U}^{\pm}$ graphs and their representations and to use this knowledge to develop algorithms deciding REPEXT for these classes in polynomial time.

In Chapter 1 we described relations between the $K_{1,3}$'s in a graph which lead to Algorithm 2 that is able to decide open vertex of every $K_{1,3}$. We also defined notions of representations of the same order type and locally similar representations as well as intervals forced and locally forced to be closed (open) that are essential for extending partial representations when multiple types of intervals can occur in the same representation.

In Chapter 2 we showed how to decide vertices that will be represented by open intervals and modify our graph to use the algorithm deciding REPEXT for proper interval graphs presented in [5] to obtain Algorithm 3 deciding REPEXT for $\mathcal{PI}^{\pm}$ graphs in polynomial time.

The main result of this thesis is presented in Chapter 3. In this chapter we characterized intervals forced and locally forced to be closed (open) in a $\mathcal{U}^{\pm}$ graph using integer gaps in the pre-representation, we constructed lower bounds for the rightmost endpoint of a component and, most importantly, we presented Algorithm 7 deciding REPEXT for $\mathcal{U}^{\pm}$ graphs in polynomial time.

We believe that a connected $\mathcal{U}^{\pm}$ graph without a pre-representation can be represented within the distance $\alpha(G)+\delta$ for every $\delta > 0$. Further research might focus on proving this and showing that Algorithm 7 reaches lower bounds presented in Chapter 3 up to some $\delta > 0$. Another natural direction is deriving an algorithm deciding REPEXT also for the class of mixed unit interval graphs, or showing this problem is NP hard.

Bibliography

[1] Xiaotie Deng, Pavol Hell, and Jing Huang. Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *SIAM Journal on Computing*, 25(2):390–403, 1996. doi: 10.1137/S0097539792269095.

[2] Mitre C. Dourado, Van Bang Le, Fábio Protti, Dieter Rautenbach, and Jayme L. Szwarcfiter. Mixed unit interval graphs. *Discrete Mathematics*, 312(22):3357–3363, 2012. doi: 10.1016/j.disc.2012.07.037.

[3] P. Frankl and H. Maehara. Open-interval graphs versus closed-interval graphs. *Discrete Mathematics*, 63(1):97–100, 1987. doi: 10.1016/0012-365X(87)90156-7.

[4] U. I. Gupta, D. T. Lee, and J. Y.-T. Leung. Efficient algorithms for interval graphs and circular-arc graphs. *Networks*, 12(4):459–467, 1982. doi: 10.1002/net.3230120410.

[5] Pavel Klavík, Jan Kratochvíl, Yota Otachi, Ignaz Rutter, Toshiki Saitoh, Maria Saumell, and Tomáš Vyskočil. Extending partial representations of proper and unit interval graphs. *Algorithmica*, 77(4):1071–1104, 2017. doi: 10.1007/s00453-016-0133-z.

[6] Pavel Klavík, Jan Kratochvíl, Yota Otachi, Toshiki Saitoh, and Tomáš Vyskočil. Extending partial representations of interval graphs. *Algorithmica*, 78(3):945–967, 2017. doi: 10.1007/s00453-016-0186-z.

[7] Van Bang Le and Dieter Rautenbach. Integral mixed unit interval graphs. *Discrete Applied Mathematics*, 161(7-8):1028–1036, 2013. doi: 10.1016/j.dam.2012.09.013.

[8] Dieter Rautenbach and Jayme L. Szwarcfiter. Unit interval graphs of open and closed intervals. *Journal of Graph Theory*, 72(4):418–429, 2013. doi: 10.1002/jgt.21650.

[9] Fred S Roberts. *Representations of indifference relations*. PhD thesis, Department of Mathematics, Stanford University, 1968.

[10] Fred S Roberts. Indifference graphs. proof techniques in graph theory. In *Proceedings of the Second Ann Arbor Graph Conference, Academic Press, New York*, pages 139–146, 1969.

List of Figures