Extending Partial Representations of Proper and Unit Interval Graphs

Pavel Klavík¹ · Jan Kratochvíl² · Yota Otachi³ · Ignaz Rutter²,⁴ · Toshiki Saitoh⁵ · Maria Saumell⁶ · Tomáš Vyskočil⁷

Received: 19 March 2015 / Accepted: 8 February 2016 / Published online: 25 February 2016
© Springer Science+Business Media New York 2016

Abstract The recently introduced problem of extending partial interval representations asks, for an interval graph with some intervals pre-drawn by the input, whether the partial representation can be extended to a representation of the entire graph. In this

The conference version of this paper appeared in SWAT 2014 [23].

✉ Pavel Klavík
klavik@iuuk.mff.cuni.cz
Jan Kratochvíl
honza@kam.mff.cuni.cz
Yota Otachi
otachi@jaist.ac.jp
Ignaz Rutter
rutter@kit.edu
Toshiki Saitoh
saitoh@edept.kobe-u.ac.jp
Maria Saumell
saumell@kma.zcu.cz
Tomáš Vyskočil
tomas.vyskocil@rutgers.edu

¹ Computer Science Institute, Charles University in Prague, Malostranské náměstí 25, 118 00 Prague, Czech Republic
² Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University in Prague, Malostranské náměstí 25, 118 00 Prague, Czech Republic
³ School of Information Science, Japan Advanced Institute of Science and Technology, Asahidai 1-1, Nomi, Ishikawa 923-1292, Japan
⁴ Faculty of Informatics, Karlsruhe Institute of Technology, Am Fasanengarten 5, 76128 Karlsruhe, Germany
paper, we give a linear-time algorithm for extending proper interval representations and an almost quadratic-time algorithm for extending unit interval representations. We also introduce the more general problem of bounded representations of unit interval graphs, where the input constrains the positions of some intervals by lower and upper bounds. We show that this problem is NP-complete for disconnected input graphs and give a polynomial-time algorithm for the special class of instances, where the ordering of the connected components of the input graph along the real line is prescribed. This includes the case of partial representation extension. The hardness result sharply contrasts the recent polynomial-time algorithm for bounded representations of proper interval graphs (Balko et al. in 2013). So unless P = NP, proper and unit interval representations have vastly different structure. This explains why partial representation extension problems for these different types of representations require substantially different techniques.

**Keywords** Intersection representation · Partial representation extension · Bounded representations · Restricted representation · Proper interval graph · Unit interval graph · Linear programming

1 Introduction

Geometric intersection graphs, and in particular intersection graphs of objects in the plane, have gained a lot of interest for their practical motivations, algorithmic applications, and interesting theoretical properties. Undoubtedly the oldest and the most studied among them are interval graphs (INT), i.e., intersection graphs of intervals on the real line. They were introduced by Hájos [18] in the 1950’s and the first polynomial-time recognition algorithm appeared already in the early 1960’s [16]. Several linear-time algorithms are known, see [6,11]. The popularity of this class of graphs is probably best documented by the fact that Web of Knowledge registers over 300 papers with the words “interval graph” in the title. For useful overviews of interval graphs and other intersection-defined classes, see textbooks [17,36].

Only recently, the following natural generalization of the recognition problem has been considered [26]. The input of the partial representation extension problem consists of a graph and a part of the representation and it asks whether it is possible to extend this partial representation to a representation of the entire graph. Klavík et al. [26] give a quadratic-time algorithm for the class of interval graphs and a cubic-time algorithm for the class of proper interval graphs. Two different linear-time algorithms are known for interval graphs [5,25]. In [27], it is shown that the partial representation extension problems are polynomially solvable for k-nested interval graphs (generalizing proper interval graphs) and NP-hard for k-length interval graphs (generalizing

---

5 Graduate School of Engineering, Kobe University, Rokkodai 1-1, Nada, Kobe 657-8501, Japan
6 Department of Mathematics and European Centre of Excellence NTIS (New Technologies for the Information Society), University of West Bohemia, Univerzitní 22, 306 14 Plzeň, Czech Republic
7 Department of Computer Science, Rutgers, The State University of New Jersey, 110 Frelinghuysen Road, Piscataway, NJ 08854-8019, USA

Springer
unit interval graphs), even when $k = 2$. There are also polynomial-time algorithms for function and permutation graphs [22], circle graphs [8], and proper circular-arc graphs [4]. Chordal graph representations as intersection graphs of subtrees of a tree [24] and intersection representations of planar graphs [7] are mostly hard to extend. The paper [28] characterizes minimal obstructions which make partial interval representations non-extendible.

A related line of research is the complex of simultaneous representation problems, pioneered by Jampani and Lubiw [19,20], where one seeks representations of two (or more) input graphs such that vertices shared by the input graphs are represented identically in each of the representations. Although in some cases the problem of finding simultaneous representations generalizes the partial representation extension problem, e.g., for interval graphs [5], this connection does not hold for all graph classes. For example, extending a partial representation of a chordal graph is $NP$-complete [24], whereas the corresponding simultaneous representation problem is polynomial-time solvable [20]. While a similar reduction as the one from [5] works for proper interval graphs, we are not aware of a direct relation between the corresponding problems for unit interval graphs.

In this paper, we extend the line of research on partial representation extension problems by studying the corresponding problems for proper interval graphs (PROPER INT) and unit interval graphs (UNIT INT). Roberts’ Theorem [33] states $PROPER \text{ INT} = UNIT \text{ INT}$. It turns out that specific properties of unit interval representations were not much investigated since it is easier to work with combinatorially equivalent proper interval representations. It is already noted in [26] that partial representation extension behaves differently for these two classes; see Fig. 1a. This is due to the fact that for proper interval graphs, in whose representations no interval is a proper subset of another interval, the extension problem is essentially topological and can be treated in a purely combinatorial manner. On the other hand, unit interval representations, where all intervals have length one, are inherently geometric, and the corresponding algorithms have to take geometric constraints into account.
It has been observed in other contexts that geometric problems are sometimes more difficult than the corresponding topological problems. For example, the partial drawing extension of planar graphs is linear-time solvable [1] for topological drawing but NP-hard for straight-line drawings [29]. Together with Balko et al. [2], our results show that a generalization of partial representation extension exhibits this behavior already in 1-dimensional geometry. The bounded representation problem is polynomial-time solvable for proper interval graphs [2] and NP-complete for unit interval graphs. From a perspective of representations, this result separates proper and unit interval graphs. We show that, unless \( P = NP \), the structure of all proper interval representations is significantly different from the structure of all unit interval representations; see Fig. 1b.

Next, we formally introduce the problems we study and describe our results.

1.1 Classes and Problems in Consideration

For a graph \( G \), an intersection representation \( \mathcal{R} \) is a collection of sets \( \{ R_u : u \in V(G) \} \) such that \( R_u \cap R_v \neq \emptyset \) if and only if \( uv \in E(G) \); so the edges of \( G \) are encoded by the intersections of the sets. An intersection-defined class \( \mathcal{C} \) is the class of all graphs having intersecting representations with some specific type of sets \( R_u \). For example, in an interval representation each \( R_u \) is a closed interval of the real line. A graph is an interval graph if it has an interval representation.

1.1.1 Studied Classes

We consider two classes of graphs. An interval representation is called proper if no interval is a proper subset of another interval (meaning \( R_u \subseteq R_v \) implies \( R_u = R_v \)). An interval representation is called unit if the length of each interval is one. The class of proper interval graphs (PROPER INT) consists of all interval graphs having proper interval representations, whereas the class of unit interval graphs (UNIT INT) consists of all interval graphs having unit interval representations. Clearly, every unit interval representation is also a proper interval representation.

In an interval representation \( \mathcal{R} = \{ R_v : v \in V \} \), we denote the left and right endpoint of the interval \( R_v \) by \( \ell_v \) and \( r_v \), respectively. For numbered vertices \( v_1, \ldots, v_n \), we denote these endpoints by \( \ell_i \) and \( r_j \). Note that several intervals may share an endpoint in a representation. When we work with multiple representations, we use \( \mathcal{R}' \) and \( \mathcal{R} \) for them. Their intervals are denoted by \( R'_v = [\ell'_v, r'_v] \) and \( \bar{\mathcal{R}}_v = [\bar{\ell}_v, \bar{r}_v] \).

1.1.2 Studied Problems

The recognition problem of a class \( \mathcal{C} \) asks whether an input graph belongs to \( \mathcal{C} \); that is, whether it has a representation by the specific type of sets \( R_u \). We study two generalizations of this problem: The partial representation extension problem, introduced in [26], and a new problem called the bounded representation problem.

A partial representation \( \mathcal{R}' \) of \( G \) is a representation of an induced subgraph \( G' \) of \( G \). A vertex in \( V(G') \) is called pre-drawn. A representation \( \mathcal{R} \) extends \( \mathcal{R}' \) if \( R_u = R'_u \) for each \( u \in V(G') \).
Problem: RepExt(C) (Partial Representation Extension of C)

Input: A graph G with a partial representation R'.

Question: Does G have a representation R that extends R'? 

Suppose, that we are given two rational numbers lbound(vi) and ubound(vi) for each vertex vi. A representation R is called a bounded representation if lbound(vi) \leq \ell_i \leq ubound(vi).

Problem: BoundRep (Bounded Representation of UNIT INT)

Input: A graph G and two rational numbers lbound(vi) and ubound(vi) for each vi \in V(G).

Question: Does G have a bounded unit interval representation?

It is easy to see that BoundRep generalizes RepExt(UNIT INT) since we can just put lbound(vi) = ubound(vi) = \ell'_i for all pre-drawn vertices, and lbound(vi) = -\infty, ubound(vi) = \infty for the remaining vertices.

The bounded representation problem can be considered also for interval graphs and proper interval graphs, where the left and right endpoints of the intervals can be restricted individually. A recent paper of Balko et al. [2] proves that this problem is polynomially solvable for these classes. Note that for unit intervals, it suffices to restrict the left endpoint since ri = \ell_i + 1. The complexity for other classes, e.g. circle graphs, circular-arc graphs or permutation graphs, is open.

1.2 Contribution and Outline

In this paper we present five results. The first is a simple linear-time algorithm for RepExt(PROPER INT), improving over a previous O(nm)-time algorithm [26]; it is based on known characterizations, and we present it in Sect. 2.

Theorem 1.1 If the endpoints of the pre-drawn intervals are given sorted from left to right, the problem RepExt(PROPER INT) can be solved in time O(n + m).

Second, in Sect. 3, we give a reduction from 3-Partition to show that BoundRep is NP-complete for disconnected graphs.

Theorem 1.2 The problem BoundRep is NP-complete.

Third, in Sect. 4.1, we give a relatively simple quadratic-time algorithm for the special case of BoundRep where the order of the connected components along the real line is fixed. The running time is O(n^2r + nD(r)),\(^1\) where r is the total encoding length of the bounds in the input, and D(r) is the time required for multiplying or dividing two numbers whose binary representation has length r. The presence of r and D(r) in the running time is due to the fact that the numbers specifying the upper

\(^1\) The complexity depends greatly on the considered computational model. If the arithmetic machine model is used (where the cost of all arithmetic operations is constant, independent of the size of operands), then this algorithm runs in time O(n^2).
and lower bounds for the intervals can be quite close to each other, requiring that
the corresponding rationals have an encoding that is super-polynomial in $n$. The best
known algorithm achieves $D(r) = O(r \log r 2^{\log^* r})$ [14].

Fourth, in Sects. 4.2–4.5, we show how to reduce the dependency on $r$ to obtain a
running time of $O(n^2 + nD(r))$, which may be beneficial for instances with bounds
that have a long encoding.

**Theorem 1.3** The problem BOUNDREP with a prescribed ordering $\triangleleft$ of the connected
components can be solved in time $O(n^2 + nD(r))$, where $r$ is the size of the input
describing bound constraints.

Further, in Sects. 3.1, 4.2, and 4.3 we derive some structural results concerning unit
interval representations. In particular, we show that all representation of one con-
ected component form a semilattice. We believe that these results might be useful
in designing a faster algorithm, attacking other problems, and getting overall better
understanding of unit interval representations.

If the number of connected components is small, we can use Theorem 1.3 to test
all possible orderings $\triangleleft$.

**Corollary 1.4** For $c$ connected components, BOUNDREP can be solved in $O(c!(n^2 +
n D(r)))$ time.

Finally, we note that every instance of REPEXT(UNIT INT) is an instance of
BOUNDREP. In Sect. 5, we show how to derive for these special instances a suit-
able ordering $\triangleleft$ of the connected components, resulting in an efficient algorithm for
REPEXT(UNIT INT).

**Theorem 1.5** REPEXT(UNIT INT) can be solved in time $O(n^2 + nD(r))$, where $r$ is
the size of the input describing positions of pre-drawn intervals.

All the algorithms described in this paper are also able to certify the extendibility
by constructing the required representations.

### 1.3 Notation and Preliminaries

As usual, we reserve $n$ for the number of vertices and $m$ for the number of edges of the
data graph $G$. We denote the set of vertices by $V(G)$ and the set of edges by $E(G)$. For a
vertex $v$, we denote the closed neighborhood of $v$ by $N[v] = \{x : vx \in E(G)\} \cup \{v\}$. We also reserve $r$ for the size of the input describing either bound constraints (for the
BOUNDREP problem) or positions of pre-drawn intervals (for REPEXT(UNIT INT)).

This value $r$ is for the entire graph $G$, and we use it even when we deal with a single
component of $G$. We reserve $c$ for the number of components of $G$ (maximal connected
subgraphs of $G$).

#### 1.3.1 (Un)located Components

Unlike the recognition problem, REPEXT cannot generally be solved independently
for connected components. A connected component $C$ of $G$ is located if it contains at
least one pre-drawn interval and unlocated if it contains no pre-drawn interval.
Let $R$ be any interval representation. Then for each component $C$, the union $\bigcup_{u \in C} R_u$ is a connected segment of the real line, and for different components we get disjoint segments. These segments are ordered from left to right, which gives a linear ordering $\prec$ of the components. So we have $c$ components ordered $C_1 \prec \cdots \prec C_c$.

2 Extending Proper Interval Representations

In this section, we describe how to extend partial representations of proper interval graphs in time $O(m + n)$. We also give a simple characterization of all extendible instances.

As we will see, the extendibility of a proper interval representation depends only on the left-to-right ordering of the endpoints of the pre-drawn intervals. Therefore, we assume that the input encodes a partial representation just by this left-to-right ordering, where some endpoints may have the same position. If rational number positions sorted from left to right are given, we can compute this ordering in time $O(n + m + r)$, where $r$ is the total encoding length of the positions of endpoints.

2.1 Indistinguishable Vertices

Vertices $u$ and $v$ are called indistinguishable if $N[u] = N[v]$. The vertices of $G$ can be partitioned into groups of (pairwise) indistinguishable vertices. Note that indistinguishable vertices may be represented by the same intervals (and this is actually true for general intersection representations). Since indistinguishable vertices are not very interesting from the structural point of view, if the structure of the pre-drawn vertices allows it, we want to prune the graph to keep only one vertex per group.

Suppose that we are given an instance of $\text{REPEXT(PROPER INT)}$. We compute the groups of indistinguishable vertices in time $O(n + m)$ using the algorithm of Rose et al. [34]. Let $u$ and $v$ be two indistinguishable vertices. If $u$ is not pre-drawn, or both vertices are pre-drawn with $R'_u = R'_v$, then we remove $u$ from the graph, and in the final constructed representation (if it exists) we put $R_u = R_v$. For the rest of the section, we shall assume that the input graph and partial representation are pruned.

An important property is that for any representation of a pruned graph, it holds that all intervals are pairwise distinct. So if two intervals are pre-drawn in the same position and the corresponding vertices are not indistinguishable, then we stop the algorithm because the partial representation is clearly not extendible.

2.2 Left-to-Right Ordering

Roberts [32] gave the following characterization of proper interval graphs:

**Lemma 2.1** (Roberts) A graph is a proper interval graph if and only if there exists a linear ordering $v_1 \prec v_2 \prec \cdots \prec v_n$ of its vertices such that the closed neighborhood of every vertex is consecutive.
Fig. 2 Two proper interval representations $\mathcal{R}_1$ and $\mathcal{R}_2$ with the left-to-right orderings $v_1 \lhd v_2 \lhd v_3 \lhd v_4 \lhd v_5 \lhd v_6 \lhd v_7 \lhd v_8$ and $v_2 \lhd v_1 \lhd v_3 \lhd v_4 \lhd v_5 \lhd v_7 \lhd v_6 \lhd v_8$.

This linear order $\lhd$ corresponds to the left-to-right order of the intervals on the real line in some proper interval representation of the graph. In each representation, the order of the left endpoints is exactly the same as the order of the right endpoints, and this order satisfies the condition of Lemma 2.1. For an example of $\lhd$, see Fig. 2.

How many different orderings $\lhd$ can a proper interval graph admit? In the case of a general unpruned graph possibly many, but all of them have a very simple structure. In Fig. 2, the graph contains two groups $\{v_1, v_2, v_3\}$ and $\{v_6, v_7\}$. The vertices of each group have to appear consecutively in the ordering $\lhd$ and may be reordered arbitrarily. Deng et al. [12] proved the following:

Lemma 2.2 (Deng et al.) For a connected (unpruned) proper interval graph, the ordering $\lhd$ satisfying the condition of Lemma 2.1 is uniquely determined up to local reordering of groups of indistinguishable vertices and complete reversal.

This lemma is key for partial representation extension of proper interval graphs. Essentially, we just have to deal with a unique ordering (and its reversal) and match the partial representation on it. Notice that in a pruned graph, if two vertices are indistinguishable, then their order is prescribed by the partial representation.

We want to construct a partial ordering $\prec$ which is a simple representation of all orderings $\lhd$ from Lemma 2.1. There exists a proper interval representation with an ordering $\lhd$ if and only if $\lhd$ extends either $\prec$ or its reversal. According to Lemma 2.2, $\prec$ can be constructed by taking an arbitrary ordering $\lhd$ and making indistinguishable vertices incomparable. For the graph in Fig. 2, we get

$$(v_1, v_2, v_3) \prec v_4 \prec v_5 \prec (v_6, v_7) \prec v_8,$$

where groups of indistinguishable vertices are put in brackets. This ordering is unique up to reversal and can be constructed in time $O(n + m)$ [10].

2.3 Characterization of Extendible Instances

We give a simple characterization of the partial representation instances that are extendible. We start with connected instances. Let $G$ be a pruned proper interval graph and $\mathcal{R}'$ be a partial representation of its induced subgraph $G'$. Then intervals in $\mathcal{R}'$ are in some left-to-right ordering $\prec_{\mathcal{R}'}$. (Recall that the pre-drawn intervals are pairwise distinct.)

Lemma 2.3 The partial representation $\mathcal{R}'$ of a connected graph $G$ is extendible if and only if there exists a linear ordering $\lhd$ of $V(G)$ such that:

\[ x \prec y \iff x \prec \lhd \prec y \prec \lhd \prec x, \]

\[ (x, y) \notin E(G) \iff x \lhd y \prec \lhd x \prec y. \]
The order of the left and right endpoints: $ell_1 < ell_2 < r_1 < ell_3 < ell_4 < r_2 < ell_5 < r_4 < ell_6 < r_5 < r_6$. The endpoints of the pre-drawn intervals split the segment into several subsegments. We place the remaining endpoints in this order and, within every subsegment, distributed equidistantly.

(1) The ordering $<$ extends $<^{R'}$, and either $<$ or its reversal.

(2) Let $R_u'$ and $R_v'$ be two pre-drawn touching intervals, i.e., $r_u = ell_v$, and let $w$ be any vertex distinct from $u$ and $v$. If $uw \in E(G)$, then $w < v$, and if $vw \in E(G)$, then $u < w$.

Proof: If there exists a representation $\mathcal{R}$ extending $\mathcal{R}'$, then it is in some left-to-right ordering $<$. Clearly, the pre-drawn intervals are placed in the same position, so $<$ has to extend $<^{\mathcal{R}'}$. According to Lemma 2.2, $<\|$ extends $<$ or its reversal. As for (2), clearly $v$ has to be the right-most neighbor of $u$ in $\mathcal{R}$: If $R_w$ is on the right of $R_v$, it would not intersect $R_u$. Similarly, $u$ is the left-most neighbor of $v$.

Conversely, let $v_1 < \cdots < v_n$ be an ordering from the statement of the lemma. We construct a representation $\mathcal{R}$ extending $\mathcal{R}'$ as follows. We compute a common left-to-right common order $< \|$ is uniquely determined by $<$. Since $<\|$ extends $<^{\mathcal{R}'}$, it is compatible with the partial representation (the pre-drawn endpoints are ordered as in $<\|$). To construct the representation $\mathcal{R}$, we place the non-pre-drawn endpoints between neighboring pre-drawn endpoints (or to the left or right of $\mathcal{R}'$) as in the ordering $<\|$. It is important that, if two pre-drawn endpoints $ell_i$ and $r_j$ share their position, then according to condition (2) there is no endpoint placed in between of $ell_i$ and $r_j$ in $<$ (otherwise one of the two implications would not hold, depending whether a left endpoint is intersected in between, or a right one). See Fig. 3 for an example.

We argue correctness of the constructed representation $\mathcal{R}$. First, it extends $\mathcal{R}'$, since the pre-drawn intervals are not modified. Second, it is a correct interval representation: Let $v_i$ and $v_j$ be two vertices with $v_i < v_j$, and let $v_k$ be the right-most neighbor of $v_i$ in $<\|$. If $v_iv_j \in E(G)$, then $ell_i < ell_k < r_i$ and, by consecutivity of $N[u]$ in $<$, we have $ell_j < ell_k$. Therefore, $R_{v_i}$ and $R_{v_j}$ intersect. If $v_iv_j \notin E(G)$ and $v_j \neq v_{k+1}$, then $r_i < ell_{k+1} < r_j$, so $R_{v_i}$ and $R_{v_j}$ do not intersect. If $v_iv_j \notin E(G)$ and $v_j = v_{k+1}$, then $r_i < ell_{k+1}$ and $R_{v_i}$ and $R_{v_j}$ do not intersect. Finally, we argue that $\mathcal{R}$ is a proper interval representation. In $<\|$ the order of the left endpoints is the same as the order of the right-endpoints, since $r_{i+1} < r_i$ is always placed on the right of $r_i$ in $<\|$. □

Now, we are ready to characterize general solvable instances.

---

Notice that, in the partial representation, some intervals may share position. But if two endpoints $ell_i$ and $r_j$ share the position, then $v_iv_j \in E(G)$ and we break the tie by setting $ell_i < r_j$. Springer
Lemma 2.4  A partial representation $\mathcal{R}'$ of a graph $G$ is extendible if and only if

1. for each component $C$, the partial representation $\mathcal{R}'_C$ consisting of the pre-drawn intervals in $C$ is extendible, and
2. pre-drawn vertices of each component are consecutive in $\preceq_{\mathcal{R}'}$.

Proof  The necessity of (1) is clear. For (2), if some component $C$ did not have its pre-drawn vertices consecutive in $\preceq_{\mathcal{R}'}$, then $\bigcup_{u \in C} R_u$ would not be a connected segment of the real line (contradicting existence of $\blacklozenge$ from Preliminaries).

Now, if the instance satisfies both conditions we can construct a correct representation $\mathcal{R}$ extending $\mathcal{R}'$ as follows. Using (2), the located components are ordered from left to right and we order the unlocated components arbitrarily on the right of all located components; see Fig. 4. This gives an ordering $\blacklozenge$ and we concatenate extending representations of all components in this ordering, constructed in Lemma 2.3.

We prove that $\text{RepExt}({\text{PROPER INT}})$ can be solved in time $O(n + m)$:

Proof (Theorem 1.1)  We use the characterization by Lemma 2.4. Condition (2) can be easily checked in time $O(n + m)$, because by our assumption the endpoints of pre-drawn intervals are given sorted from left to right. For condition (1), we use Lemma 2.3, and we check for each component both constraints (1) and (2). To check (2), we compute for $\prec$ and its reversal the unique orderings $\prec$. We test for each of them whether each touching pair of pre-drawn intervals is placed in $\prec$ according to (2). This can also be done in time $O(n + m)$.

If necessary, a representation $\mathcal{R}$ can be constructed in the same running time since the proofs of Lemmas 2.3 and 2.4 are constructive.

3 Bounded Representations of Unit Interval Graphs

In this section, we deal with bounded representations. An input of $\text{BOUNDREP}$ consists of a graph $G$ and, for each vertex $v_i$, two rational numbers called a lower bound $\text{lb}ound(v_i)$ and an upper bound $\text{ub}ound(v_i)$. (We allow $\text{lb}ound(v_i) = -\infty$ and $\text{ub}ound(v_i) = +\infty$.) The problem asks whether there exists a unit interval representation $\mathcal{R}$ of $G$ such that $\text{lb}ound(v_i) \leq \ell_i \leq \text{ub}ound(v_i)$ for each interval $v_i$. Such a representation is called a bounded representation.
Since unit interval representations are proper interval representations, most properties of proper interval representations described in Sect. 2 hold, in particular the existence and uniqueness of the orderings $\preceq$ and $\prec$. The key difference is that we cannot work only with the left-to-right ordering of the endpoints. Whether a bounded representation exists or not depends on precise rational number positions of bounds; see Fig. 1a.

3.1 Representations in $\varepsilon$-Grids

Endpoints of intervals can be positioned at arbitrary rational numbers. For the purpose of the algorithm, we want to work with representations drawn in limited resolution. For a given instance of the bounded representation problem, we want to find a lower bound for the required resolution such that this instance is solvable if and only if it is solvable in this limited resolution.

More precisely, we want to represent all intervals so that their endpoints correspond to points on some grid. For a value $\varepsilon = \frac{1}{K} > 0$, where $K$ is an integer, the $\varepsilon$-grid is the set of points $\{k\varepsilon : k \in \mathbb{Z}\}$. For a given instance of $\text{BOUNDREP}$, we ask which value of $\varepsilon$ ensures that we can construct a representation having all endpoints on the $\varepsilon$-grid. So the value of $\varepsilon$ is the resolution of the drawing.

If there are no bounds, every unit interval graph has a representation in the grid of size $\frac{1}{n}$ [10]. In the case of $\text{BOUNDREP}$, the size of the grid has to depend on the values of the bounds. Consider all values $\text{lbounds}(v_i)$ and $\text{ubounds}(v_i)$ distinct from $\pm\infty$, and express them as irreducible fractions $p_1/q_1, p_2/q_2, \ldots, p_b/q_b$. Then we define:

$$\varepsilon' := \frac{1}{\text{lcm}(q_1, q_2, \ldots, q_b)}, \quad \text{and} \quad \varepsilon := \frac{\varepsilon'}{n},$$

(1)

where $\text{lcm}(q_1, q_2, \ldots, q_b)$ denotes the least common multiple of $q_1, \ldots, q_b$. It is important that the size of this $\varepsilon$ written in binary is $O(r)$. We show that the $\varepsilon$-grid is sufficient to construct a bounded representation:

**Lemma 3.1** If there exists a bounded representation $R'$ for an input of the problem $\text{BOUNDREP}$, there exists a bounded representation $R$ in which all intervals have endpoints on the $\varepsilon$-grid, where $\varepsilon$ is defined by (1).

**Proof** We construct an $\varepsilon$-grid representation $R$ from $R'$ in two steps. First, we shift intervals to the left, and then we shift intervals slightly back to the right. For every interval $v_i$, the sizes of the left and right shifts are denoted by $\text{LS}(v_i)$ and $\text{RS}(v_i)$ respectively. The shifting process is shown in Fig. 5.

In the first step, we consider the $\varepsilon'$-grid and shift all the intervals to the left to the closest grid-point (we do not shift an interval if its endpoints are already on the grid). Original intersections are kept by this shifting, since if $x$ and $y$ are two endpoints satisfying $x \leq y$ before the left-shift, then $x \leq y$ also holds after the left-shift. So if

---

3 If $\varepsilon$ was not of the form $\frac{1}{K}$, then the grid could not contain both left and right endpoints of the intervals. We reserve $K$ for the value $\frac{1}{\varepsilon}$ in this paper.
In the first step, we shift intervals to the left to the $\varepsilon'$-grid. The left shifts of $v_1,\ldots,v_5$ are $(0, 0, \frac{1}{2}\varepsilon', \frac{1}{3}\varepsilon', 0)$. In the second step, we shift to the right in the refined $\varepsilon$-grid. Right shifts have the same relative order as left shifts: $(0, 0, 2\varepsilon, \varepsilon, 0)$.

The second step shifts the intervals to the right in the refined $\varepsilon$-grid to remove the additional intersections created by the first step. The right-shift is a mapping $RS : \{v_1,\ldots,v_n\} \rightarrow \{0, \varepsilon, 2\varepsilon, \ldots, (n - 1)\varepsilon\}$ having the right-shift property: For all pairs $(v_i, v_j)$ with $r_i = \ell_j$, $RS(v_i) \geq RS(v_j)$ if and only if $v_i v_j \in E$. So the right-shift property ensures that $RS$ fixes wrongly represented touching pairs created by $LS$.

To construct such a mapping $RS$, notice that if we relax the image of $RS$ to $[0, \varepsilon')$, the reversal of $LS$ has the right-shift property, since it produces the original correct representation $R'$. But the right-shift property depends only on the relative sizes of the shifts and not on the precise values. Therefore, we can construct $RS$ from the reversal of $LS$ by keeping the shifts in the same relative order. If $LS(v_i)$ is one of the $k$th smallest shifts, we set $RS(v_i) = (k - 1)\varepsilon$.\footnote{In other words, for the smallest shifts we assign the right-shift 0; for the second smallest shifts, we assign $\varepsilon$; for the third smallest shifts, $2\varepsilon$; and so on.}

Additionally, Lemma 3.1 shows that it is always possible to construct an $\varepsilon$-grid representation having the same topology as the original representation, in the sense that overlapping pairs of intervals keep overlapping, and touching pairs of intervals only touch; for an example, see vertices $v_2$ and $v_4$ in Fig. 5.

Next we look at the bound constraints. If, before the shifting, $v_i$ was satisfying $\ell_i \geq l\text{bound}(v_i)$, then this is also satisfied after $LS(v_i)$ since the $\varepsilon'$-grid contains the value $l\text{bound}(v_i)$. Obviously, the inequality is not broken after $RS(v_i)$. As for the upper bound, if $LS(v_i) = 0$ and $RS(v_i) = 0$, then the bound is trivially satisfied. Otherwise, after $LS(v_i)$ we have $\ell_i \leq u\text{bound}(v_i) - \varepsilon'$, so the upper bound still holds after $RS(v_i)$. \hfill $\square$

Additionally, Lemma 3.1 shows that it is always possible to construct an $\varepsilon$-grid representation having the same topology as the original representation, in the sense that overlapping pairs of intervals keep overlapping, and touching pairs of intervals only touch; for an example, see vertices $v_2$ and $v_4$ in Fig. 5.
keep touching. Also notice that both representations $R$ and $R'$ have the same order of the intervals.

In the standard unit interval graph representation problem, no bounds on the positions of the intervals are given, and we get $\varepsilon' = 1$ and $\varepsilon = \frac{1}{n}$. Lemma 3.1 proves in a particularly clean way that the grid of size $\frac{1}{n}$ is sufficient to construct unrestricted representations of unit interval graphs. Corneil et al. [10] show how to construct this representation directly from the ordering $<$, whereas we use some given representation to construct an $\varepsilon$-grid representation.

### 3.2 Hardness of BOUNDREP

In this subsection we focus on hardness of bounded representations of unit interval graphs. We prove Theorem 1.2, stating that BOUNDREP is $NP$-complete.

We reduce the problem from 3-Partition. An input of 3-Partition consists of natural numbers $k$, $M$, and $A_1, \ldots, A_{3k}$ such that $\frac{M}{4} < A_i < \frac{M}{2}$ for all $i$, and $\sum A_i = kM$. The question is whether it is possible to partition the numbers $A_i$ into $k$ triples such that each triple sums to exactly $M$. This problem is known to be strongly $NP$-complete (even if all numbers have polynomial sizes) [15].

**Proof (Theorem 1.2)** According to Lemma 3.1, if there exists a representation satisfying the bound constraints, then there also exists an $\varepsilon$-grid representation with this property. Since the length of $\varepsilon$ given by (1), written in binary, is polynomial in the size of the input, all endpoints can be placed in polynomially-long positions. Thus we can guess the bounded representation and the problem belongs to $NP$.

Let us next prove that the problem is $NP$-hard. For a given input of 3-Partition, we construct the following unit interval graph $G$. For each number $A_i$, we add a path $P_{2A_i}$ (of length $2A_i - 1$) into $G$ as a separate component. For all vertices $x$ in these paths, we set bounds

$$\text{lbound}(x) = 1 \quad \text{and} \quad \text{ubound}(x) = k \cdot (M + 2).$$

In addition, we add $k + 1$ independent vertices $v_0, v_1, \ldots, v_k$, and make their positions in the representation fixed:

$$\text{lbound}(v_i) = \text{ubound}(v_i) = i \cdot (M + 2).$$

See Fig. 6 for an illustration of the reduction. Clearly, the reduction is polynomial.

We now argue that the bounded representation problem is solvable if and only if the given input of 3-Partition is solvable. Suppose first that the bounded representation problem admits a solution. There are $k$ gaps between the fixed intervals $v_0, \ldots, v_k$ each of which has space less than $M + 1$. (The length of the gap is $M + 1$ but the endpoints are taken by $v_i$ and $v_{i+1}$.) The bounds of the paths force their representations to be inside these gaps, and each path lives in exactly one gap. Hence the representation induces a partition of the paths.

Now, the path $P_{2A_i}$ needs space at least $A_i$ in every representation since it has an independent set of the size $A_i$. The representations of the paths may not overlap and
We consider the following input for 3-PARTITION: \( k = 2, M = 7, A_1 = A_2 = A_3 = A_4 = 2 \) and \( A_5 = A_6 = 3 \). The associated unit interval graph is depicted on top, and at the bottom we find one of its correct bounded representations, giving 3-partitioning \{\( A_1, A_3, A_6 \)\} and \{\( A_2, A_4, A_5 \)\}.

Thus the obtained partition solves the 3-PARTITION problem.

Conversely, every solution of 3-PARTITION can be realized as a bounded representation in this way.

4 Bounded Representations of Unit Interval Graphs with Prescribed Ordering

In this section, we deal with the BOUNDREP problem when a fixed ordering \( \prec \) of the components is prescribed. First we solve the problem using linear programming. Then we describe additional structure of bounded representations, and using this structure we construct an almost quadratic-time algorithm that solves the linear programs.

4.1 LP Approach for BOUNDREP

According to Lemma 2.2, each component of \( G \) can be represented in at most two different ways, up to local reordering of groups of indistinguishable vertices. Unlike the case of proper interval graphs, we cannot arbitrarily choose one of the orderings, since neighboring components restrict each other’s space. For example, only one of the two orderings for the component \( C_1 \) in Fig. 7 makes a representation of \( C_2 \) possible.

The positions of the vertices \( u \) and \( v \) are fixed by the bound constraints. The component \( C_1 \) can only be represented with \( u \) being the right-most interval, since otherwise \( C_1 \) would block space for the component \( C_2 \).
In the algorithm, we process components \( C_1 \leftarrow C_2 \leftarrow \cdots \leftarrow C_c \) from left to right and construct representations for them. When we process a component \( C_i \), we want to represent it on the right of the previous component \( C_{i-1} \), and we want to push the representation of \( C_i \) as far to the left as possible, leaving as much space for \( C_{i+1}, \ldots, C_c \) as possible.

Now, we describe in details, how we process a component \( C_i \). We calculate by the algorithm of Corneil et al. the partial ordering \( < \) described in Sect. 2 and its reversal. The elements that are incomparable by these partial orderings are vertices of the same group of indistinguishable vertices. For these vertices, the following holds:

**Lemma 4.1** Suppose there exists some bounded representation \( \mathcal{R} \). Then there exists a bounded representation \( \mathcal{R}' \) such that, for every indistinguishable pair \( v_i \) and \( v_j \) satisfying \( \text{lbound}(v_i) \leq \text{lbound}(v_j) \), it holds that \( \ell'_i \leq \ell'_j \).

**Proof** Given a representation \( \mathcal{R} \), we call a pair \((v_i, v_j)\) bad if \( v_i \) and \( v_j \) are indistinguishable, \( \text{lbound}(v_i) \leq \text{lbound}(v_j) \) and \( \ell_i > \ell_j \). We describe a process which iteratively constructs \( \mathcal{R}' \) from \( \mathcal{R} \), by constructing a sequence of representations \( \mathcal{R} = \mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_k = \mathcal{R}' \), where the positions in a representation \( \mathcal{R}_s \) are denoted by \( \ell_s^i \) ’s.

In each step \( s \), we create \( \mathcal{R}_s \) from \( \mathcal{R}_{s-1} \) by fixing one bad pair \((v_i, v_j)\): we set \( \ell_s^i = \ell_{s-1}^j \) and the rest of the representation remains unchanged. Since \( v_i \) and \( v_j \) are indistinguishable and \( \mathcal{R}_{s-1} \) is correct, the obtained \( \mathcal{R}_s \) is a representation. Regarding bound constraints,

\[
\text{lbound}(v_i) \leq \text{lbound}(v_j) \leq \ell_{s-1}^i = \ell_s^i < \ell_{s-1}^j \leq \text{ubound}(v_j),
\]

so the bounds of \( v_i \) are satisfied.

Now, in each \( \mathcal{R}_s \), the set of all left endpoints is a subset of the set of all left endpoints of \( \mathcal{R} \). In each step, we move one left-endpoint to the left, so each endpoint is moved at most \( n-1 \) times. Hence the process terminates after \( O(n^2) \) iterations and produces a representation \( \mathcal{R}' \) without bad pairs as requested. \( \square \)

For \( < \) and its reversal, we use Lemma 4.1 to construct linear orderings \( \triangleleft \): If \( v_i \) and \( v_j \) belong to the same group of indistinguishable vertices and \( \text{lbound}(v_i) < \text{lbound}(v_j) \), then \( v_i \triangleleft v_j \). If \( \text{lbound}(v_i) = \text{lbound}(v_j) \), we choose any order \( \triangleleft \) between \( v_i \) and \( v_j \).

We obtain two total orderings \( < \), and we solve a linear program for each of them. Let \( v_1 \triangleleft v_2 \triangleleft \cdots \triangleleft v_k \) be one of these orderings. We denote the right-most endpoint of a representation of a component \( C_i \) by \( E_i \). Additionally, we define \( E_0 = -\infty \). Let \( \epsilon \) be defined as in (1). We modify all lower bounds by putting \( \text{lbound}(v_i) = \max\{\text{lbound}(v_i), E_{i-1} + \epsilon\} \) for every interval \( v_i \), which forces the representation of \( C_i \) to be on the right of the previously constructed representation of \( C_{i-1} \). The linear program has variables \( \ell_1, \ldots, \ell_k \), and it minimizes the value of \( E_i \). We solve:

**Minimize:** \( E_i := \ell_k + 1 \),

**subject to:**

\[
\ell_i \leq \ell_{i+1}, \quad \forall i = 1, \ldots, k - 1, \quad (2)
\]

\[
\ell_i \geq \text{lbound}(v_i), \quad \forall i = 1, \ldots, k, \quad (3)
\]
We solve the same linear program for the other ordering of the vertices of \( C_t \). If none of the two programs is feasible, we report that no bounded representation exists. If at least one of them is feasible, we take the solution minimizing \( E_t \).

**Lemma 4.2** Let the representation of \( C_{t-1} \) be fixed. Every bounded \( \varepsilon \)-grid representation of the component \( C_t \) with the left-to-right order \( v_1 \prec \cdots \prec v_k \) which is on the right of the representation of \( C_{t-1} \) satisfies constraints (3)–(6).

**Proof** Constraints of types (3) and (4) are satisfied, since the representation is bounded and on the right of \( C_{t-1} \). Constraints of type (5) correspond to a correct representation of intersecting pairs of intervals. The non-intersecting pairs of an \( \varepsilon \)-grid representation are at distance at least \( \varepsilon \), which makes constraints of type (6) satisfied.

Now, we are ready to show:

**Proposition 4.3** The BOUNDREP problem with prescribed \( \prec \) can be solved in polynomial time.

**Proof** Concerning the running time, it depends polynomially on the sizes of \( n \) and \( \varepsilon \), which are polynomial in the size of the input \( r \). It remains to show correctness.

Suppose that the algorithm returns a candidate for a bounded representation. The formulation of the linear program ensures that it is a correct representation: Constraints of type (2) make the representation respect \( \prec \). Constraints of type (3) and (4) enforce that the given lower and upper bounds for the positions of the intervals are satisfied, force the prescribed ordering \( \prec \) on the representation of \( G \), and force the drawings of the distinct components to be disjoint. Finally, constraints of type (2), (5) and (6) make the drawing of the vertices of a particular component \( C_t \) be a correct representation.

Suppose next that a bounded representation exists. According to Lemma 3.1 and Lemma 4.1, there also exists an \( \varepsilon \)-grid bounded representation \( R' \) having the order in the indistinguishable groups as defined above. So for each component \( C_t \), one of the two orderings \( \prec \) constructed for the linear programs agrees with the left-to-right order of \( C_t \) in \( R' \).

We want to show that the representation of each component \( C_t \) in \( R' \) gives a solution to one of the two linear programs associated to \( C_t \). We denote by \( E'_t \) the value of \( E_t \) in the representation \( R' \), and by \( E_{t}^{\min} \) the value of \( E_t \) obtained by the algorithm after solving the two linear programming problems associated to \( C_t \). We show by induction on \( t \) that \( E_{t}^{\min} \leq E'_t \), which specifically implies that \( E_{t}^{\min} \) exists and at least one of the linear programs for \( C_t \) is solvable.

We start with \( C_1 \). As argued above, the left-to-right order in \( R' \) agrees with one of the orderings \( \prec \), so the representation of \( C_1 \) satisfies the constraints (2). Since \( E_0 = -\infty \), the lower bounds are not modified. By Lemma 4.2, the rest of the constraints are also satisfied. Thus the representation of \( C_1 \) gives a feasible solution for the program and gives \( E_{1}^{\min} \leq E'_1 \).
Assume now that, for some $C_t$ with $t \geq 1$, at least one of the two linear programming problems associated to $C_t$ admits a solution, and from induction hypothesis we have $E_t^{\min} \leq E'_t$. In $R'$, two neighboring components are represented at distance at least $\varepsilon$. Therefore for every vertex $v_i$ of $C_{t+1}$, it holds $\ell_i \geq E'_t + \varepsilon \geq E_t^{\min} + \varepsilon$, so the modification of the lower bound constraints is satisfied by $R'$. Similarly as above using Lemma 4.2, the representation of $C_{t+1}$ in $R'$ satisfies the remaining constraints. It gives some solution to one of the programs and we get $E_{t+1}^{\min} \leq E'_{t+1}$.

In summary, if there exists a bounded representation, for each component $C_t$ at least one of the two linear programming problems associated to $C_t$ admits a solution. Therefore, the algorithm returns a correct bounded representation $R$ (as discussed in the beginning of the proof). We note that $R$ does not have to be an $\varepsilon$-grid representation since the linear program just states that non-intersecting intervals are at distance at least $\varepsilon$. To construct an $\varepsilon$-grid representation if necessary, we can proceed as in the proof of Lemma 3.1. \hfill \Box

We note that it is possible to reduce the number of constraints of the linear program from $O(k^2)$ to $O(k)$, since neighbors of each $v_i$ appear according to Lemma 2.1 consecutively in $\prec$. Using the ordering constraints (2), we can replace constraints (5) and (6) by a linear number of constraints as follows. For each $v_j$, there are two cases. If $v_j$ is adjacent to all vertices $v_i$ such that $v_i \prec v_j$, then we only state the constraint (5) for $v_1$ and $v_j$. Otherwise, let $v_i$ be the rightmost vertex such that $v_i \prec v_j$ and $v_i v_j \notin E$. Then we only state the constraint (5) for $v_i + 1$ and $v_j$, and the constraint (6) for $v_i$ and $v_j$. This is equivalent to the original formulation of the problem.

In general, any linear program can be solved in $O(n^{3.5} r^2 \log r \log \log r)$ time by using Karmarkar’s algorithm [21]. However, our linear program is special, which allows to use faster techniques:

**Proposition 4.4** The $\text{BoundRep}$ problem with prescribed $\prec$ can be solved in time $O(n^2 r + nD(r))$.

**Proof** Without loss of generality, we assume that the upper and lower bounds restrict the final representation (if it exists) to lie in the positive real line. For a given $i$, let $j_i$ be the index such that $v_{j_i}$ is the rightmost neighbor of $v_i$ in $\prec$. Let $h_i$ be the index such that $v_{h_i}$ is the rightmost vertex such that $v_{h_i} \prec v_i$ and $v_{h_i} v_i \notin E$. (Notice that $h_i$ might not be defined, in which case we ignore inequalities containing it.)

We replace the variables $\ell_1, \ldots, \ell_k$ by $x_0, \ldots, x_k$ such that $\ell_i = x_i - x_0$. We want to solve the following linear system:

Minimize: 

$$E'_t := x_k - x_0 + 1,$$

subject to:

$$x_i - x_{i+1} \leq 0, \quad \forall i = 1, \ldots, k - 1,$$

$$x_0 - x_i \leq -\text{lboun}(v_i), \quad \forall i = 1, \ldots, k,$$

$$x_i - x_0 \leq \text{uboun}(v_i), \quad \forall i = 1, \ldots, k,$$

$$x_{j_i} - x_i \leq 1, \quad \forall i = 1, \ldots, k,$$

$$x_{h_i} - x_i \leq -1 - \varepsilon, \quad \forall i = 1, \ldots, k.$$
The obtained linear program is a system of difference constraints, since each inequality has the form \( x_i - x_j \leq b_{i,j} \).

Following [9, Chapter 24.4], if the system is feasible, a solution, which is not necessarily optimal, can be found as follows. We define a weighted digraph \( D \) having vertex set \( V(D) = \{s, u_0, u_1, \ldots, u_k\} \), where \( u_i \) corresponds to \( x_i \) and \( s \) is a special vertex. For the edges \( E(D) \), we first have an edge \((s, u_i)\) of weight zero for every \( u_i \). Then for every constraint \( x_i - x_j \leq b_{i,j} \), we add the edge \((u_j, u_i)\) of weight \( b_{i,j} \). See Fig. 8.

As proved in [9, Chapter 24.4], there are two possible cases. If \( G \) contains a negative-weight cycle, then there is no feasible solution for the system. If \( G \) does not contain negative-weight cycles, then we define \( \delta(s, u_i) \) as the weight of the minimum-weight path connecting \( s \) to \( u_i \) in \( G \). Then we put \( x_i = \delta(s, u_i) \) for each \( i \) which defines a feasible solution of the system. Moreover, this solution minimizes the objective function \( \max\{x_i\} - \min\{x_i\} \). We next show that this function is equivalent to the objective function in our linear program.

Suppose that we have a solution of our system, satisfying the constraints but not necessarily optimizing the objective function. Because of our assumption that the representation lies in the positive real line, we know that \( \ell_i > 0 \) for all \( i \). Therefore, \( x_i > x_0 \). So \( \min\{x_i\} \) is always attained by \( x_0 \), while \( \max\{x_i\} \) is always attained by \( x_k \). So minimization of the objective function \( \max\{x_i\} - \min\{x_i\} \) is equivalent to the original minimization of \( E_t = x_k - x_0 + 1 \).

In order to find a negative-weight cycle in \( D \) or, alternatively, compute the weight of the minimum-weight paths from \( s \) to all the other vertices of \( D \), we use the Bellman-Ford algorithm. Notice that Dijkstra’s algorithm cannot be used in this case, since some edges of \( D \) have negative weight. We next analyze the running time of the whole procedure.

We assume that the cost of arithmetic operations with large numbers is not constant. The algorithm computes the value \( \varepsilon \) in the beginning which can be clearly done in
time $O(nD(r))$. (Instead of the least common multiple we can simply compute the product of $q_i$’s.)

Afterwards, we compute the weights of the edges of $D$ as multiples of $\varepsilon$, which takes time $O(kD(r))$. Then each step of the Bellman-Ford algorithm requires time $O(r)$, and the algorithm runs $O(k^2)$ steps in total. The total time to solve each linear program is therefore $O(k^2r + kD(r))$. Finally, the total time of the algorithm is $O(n^2r + nD(r))$.

In the next subsections, we improve the time complexity of the BoundRep problem with prescribed $\sqsubseteq$ to $O(n^2 + nD(r))$. Our algorithm makes use of several properties of the set of all representations. We note that structural properties of the polyhedron of our linear program, in the case where all lower bounds equal zero and there are no upper bounds, have been considered in several papers in the context of semiorders [3,30].

4.2 The Partially Ordered Set $\mathcal{R}$

Let the graph $G$ in consideration be a connected unit interval graph. We study structural properties of its representations. Suppose that we fix one of the two partial left-to-right orders $<$ of the intervals from Sect. 2, so that only indistinguishable vertices are incomparable. We also fix some positive $\varepsilon = \frac{1}{K}$. For most of this section, we work only with lower bounds.

We define $\mathcal{R}$ as the set of all $\varepsilon$-grid representations satisfying the lower bounds and in some left-to-right ordering that extends $<$. We define a very natural partial ordering $\leq$ on $\mathcal{R}$: We say that $\mathcal{R} \leq \mathcal{R}'$ if and only if $\ell_i \leq \ell'_i$ for every $v_i \in V(G)$; i.e., $\leq$ is the Cartesian ordering of vectors $(\ell_1, \ldots, \ell_n)$. In this section, we study properties of the poset $(\mathcal{R}, \leq)$.

If $\varepsilon \leq \frac{1}{n}$, then $\mathcal{R} \neq \emptyset$. The reason is that the graph $G$ is a unit interval graph, and thus there always exists an $\varepsilon$-grid representation $\mathcal{R}$ far to the right satisfying the lower bound constraints.

4.2.1 The Semilattice Structure

Let us assume that $\text{lboun}(v_i) > -\infty$ for some $v_i \in V(G)$. Let $S$ be a subset of $\mathcal{R}$. The infimum $\inf(S)$ is the greatest representation $\mathcal{R} \in \mathcal{R}$ such that $\mathcal{R} \leq \mathcal{R}'$ for every $\mathcal{R}' \in S$. In a general poset, infimums may not exist, but if they do, they are always unique. For $\mathcal{R}$, we show:

**Lemma 4.5** Every non-empty $S \subseteq \mathcal{R}$ has an infimum $\inf(S)$.

**Proof** We construct the requested infimum $\mathcal{R}$ as follows:

$$\ell_i = \min \{\ell'_i : \mathcal{R}' \in S\}, \quad \forall v_i \in V(G).$$

Notice that the positions in $\mathcal{R}$ are well-defined, since the position of each interval in each $\mathcal{R}'$ is bounded and always on the $\varepsilon$-grid. If $\mathcal{R}$ is a correct representation, it is the infimum $\inf(S)$. It remains to show that $\mathcal{R} \in \mathcal{R}$. 

© Springer
Clearly, all positions in \(\mathcal{R}\) belong to the \(\varepsilon\)-grid and satisfy the lower bound constraints. Let \(v_i\) and \(v_j\) be two vertices. The values \(\ell_i\) and \(\ell_j\) in \(\mathcal{R}\) are given by two representations \(R_1, R_2 \in S\), that is, \(\ell_i = \ell^1_i\) and \(\ell_j = \ell^2_j\). Notice that the left-to-right order in \(\mathcal{R}\) has to extend \(<\): If \(v_i < v_j\), then \(\ell_i = \ell^1_i \leq \ell^2_i < \ell^2_j = \ell_j\), since \(R_1\) minimizes the position of \(v_i\) and the left-to-right order in \(R_2\) extends \(<\).

Concerning correctness of the representation of the pair \(v_i\) and \(v_j\), we suppose that \(\ell_i = \ell^1_i \leq \ell^2_j = \ell_j\); otherwise we swap \(v_i\) and \(v_j\).

– First we suppose that \(v_i v_j \in E(G)\). Then \(\ell^2_j \leq \ell^1_j\), since \(R_2\) minimizes the position of \(v_j\). Since \(R_1\) is a correct representation, \(\ell^1_j - 1 \leq \ell^1_i\). So \(\ell_j - 1 \leq \ell_i \leq \ell_j\), and the intervals \(v_1\) and \(v_2\) intersect.

– The other case is when \(v_i v_j \notin E(G)\). Then \(\ell^1_i \leq \ell^2_i \leq \ell^2_j - 1 - \varepsilon\), since \(R_1\) minimizes the position of \(v_i\), \(R_2\) is a correct representation and \(v_i < v_j\) in both representations. So \(v_i\) and \(v_j\) do not intersect in \(\mathcal{R}\) as requested.

Consequently, \(\mathcal{R}\) represents correctly each pair \(v_i\) and \(v_j\), and hence \(\mathcal{R} \in \text{Rep}\). \(\Box\)

A poset is a \((\text{meet})\)-semilattice if every pair of elements \(a, b\) has an infimum \(\inf(\{a, b\})\). Lemma 4.5 shows that the poset \((\text{Rep}, \leq)\) forms a \((\text{meet})\)-semilattice. Similarly as \(\text{Rep}\), we could consider the poset set of all \((\varepsilon\)-grid\) representations satisfying both the lower and upper bounds. The structure of this poset is a complete lattice, since all subsets have infimums and supremums. Lattices and semilattices are frequently studied, and posets that are lattices satisfy very strong algebraic properties.

4.2.2 The Left-Most Representation

We are interested in a specific representation in \(\text{Rep}\), called the left-most representation; a similar idea appears independently in [13]. An \(\varepsilon\)-grid representation \(\mathcal{R} \in \text{Rep}\) is the left-most representation if \(\mathcal{R} \leq \mathcal{R}'\) for every \(\mathcal{R}' \in \text{Rep}\); so the left-most representation is left-most in each interval at the same time. We note that the notion of left-most representation does not make sense if we consider general representations (not on the \(\varepsilon\)-grid). The left-most representation is the infimum \(\inf(\text{Rep})\), and thus by Lemma 4.5 we get:

Corollary 4.6 The left-most representation always exists and it is unique.

There are two algorithmic motivations for studying left-most representations. First, in the linear program of Sect. 4.1 we need to find a representation minimizing \(E_t\). Clearly, the left-most representation minimizes \(E_t\) and, in addition, it also minimizes the rest of the endpoints. The second motivation is that we want to construct a representation satisfying the upper bounds as well, so it seems reasonable to try to place every interval as far to the left as possible. The left-most representation is indeed a good candidate for a bounded representation:

Lemma 4.7 There exists a representation \(\mathcal{R}'\) satisfying both the lower and upper bound constraints if and only if the left-most representation \(\mathcal{R}\) satisfies the upper bound constraints.
4.3 Left-Shifting of Intervals

Suppose that we construct some initial $\varepsilon$-grid representation that is not the left-most representation. We want to transform this initial representation in $\mathcal{R}_{\text{Rep}}$ into the left-most representation of $\mathcal{R}_{\text{Rep}}$ by applying a sequence of the following simple operation called left-shifting. The left-shifting operation shifts one interval of a representations by $\varepsilon$ to the left such that this shift maintains the correctness of the representation; for an example see Fig. 9a. The main goal of this section is to prove that by left-shifting we can always produce the left-most representation.

\textbf{Proposition 4.8} For $\varepsilon = \frac{1}{K}$ and $K \geq \frac{n}{2}$, an $\varepsilon$-grid representation $\mathcal{R} \in \mathcal{R}_{\text{Rep}}$ is the left-most representation if and only if it is not possible to shift any single interval to the left by $\varepsilon$ while maintaining correctness of the representation.

Before proving the proposition, we describe some additional combinatorial structure of left-shifting. An interval $v_i$ is called fixed if it is in the left-most position and cannot ever be shifted more to the left, i.e., $\ell_i = \min\{\ell'_j : \mathcal{R}' \in \mathcal{R}_{\text{Rep}}\}$. For example, an interval $v_i$ is fixed if $\ell_i = \text{lbound}(v_i)$. A representation is the left-most representation if and only if every interval is fixed.

\subsection{Obstruction Digraph}

An interval $v_i$, having $\ell_i > \text{lbound}(v_i)$, can be left-shifted if it does not make the representation incorrect, and the incorrectness can be obtained in two ways. First, there could be some interval $v_j$, such that $v_j \prec v_i$, $v_iv_j \notin E(G)$ and $\ell_j + 1 + \varepsilon = \ell_i$; we call $v_j$ a left obstruction of $v_i$. Second, there could be some interval $v_j$, such that...
If $v_i < v_j$, $v_iv_j \in E(G)$ and $\ell_i + 1 = \ell_j$ (so $v_i$ and $v_j$ are touching); then we call $v_j$ a right obstruction of $v_i$. In both cases, we first need to move $v_j$ before moving $v_i$.

For the current representation $R$, we define the obstruction digraph $H$ on the vertices of $G$ as follows. We put $V(H) = V(G)$ and $(v_i, v_j) \in E(H)$ if and only if $v_j$ is an obstruction of $v_i$. For an edge $(v_i, v_j)$, if $v_j < v_i$, we call it a left edge; if $v_i < v_j$, we call it a right edge. As we apply left-shifting, the structure of $H$ changes; see Fig. 9b.

**Lemma 4.9** An interval $v_i$ is fixed if and only if there exists a directed path in $H$ from $v_i$ to an interval $v_j$ such that $\ell_j = lbound(v_j)$.

**Proof** Suppose that $v_i$ is connected to $v_j$ by a path in $H$. By the definition of $H$, $v_xv_y \in E(H)$ implies that $v_y$ has to be shifted before $v_x$. Thus $v_j$ has to be shifted before moving $v_i$, which is not possible because $\ell_j = lbound(v_j)$.

On the other hand, suppose that $v_i$ is fixed, i.e., $\ell_i = \inf\{\ell'_i : \forall R'\}$. Let $H'$ be the induced subgraph of $H$ of the vertices $v_j$ such that there exists a directed path from $v_i$ to $v_j$. If for all $v_j \in H'$, $\ell_j > lbound(v_j)$, we can shift all vertices of $H'$ by $\epsilon$ to the left, which constructs a correct representation and contradicts the fact that $v_i$ is fixed. Therefore, there exists $v_j \in H'$ having $\ell_j = lbound(v_j)$. \qed

For example, in Fig. 9 (left), if $\ell_4 = lbound(v_4)$, then the intervals $v_3, v_4, v_5$ and $v_7$ are fixed. Also, we can prove:

**Lemma 4.10** If $\epsilon = \frac{1}{K}$ and $K \geq \frac{n}{2}$, the obstruction digraph $H$ is acyclic.

**Proof** Suppose for contradiction that $H$ contains some cycle $u_1, \ldots, u_c$. This cycle contains $a$ left edges and $b$ right edges. Recall that if $(u_i, u_{i+1})$ is a left edge, then $\ell_{ui+1} = \ell_{ui} - 1 - \epsilon$, and if it is a right edge, $\ell_{ui+1} = \ell_{ui} + 1$ (and similarly for $(u_c, u_1)$). If we go along the cycle from $u_1$ to $u_1$, the initial and the final positions have to be the same. Therefore, $a(1 + \epsilon) = b$. If this equation holds, then $a$ has to be a multiple of $K$. Therefore $a \geq K$ and $b \geq K + 1$, and thus $n \geq c = a + b \geq 2K + 1$, which is not possible. \qed

We note that the assumption $K \geq \frac{n}{2}$ is necessary and tight. For every $\epsilon = \frac{1}{K}$, there exists a representation of a graph with $2K + 1$ vertices having a cycle in $H$. The graph contains two cliques $v_0, \ldots, v_{K-1}$ and $w_0, \ldots, w_K$ such that $v_i$ is also adjacent to $w_0, \ldots, w_i$. Then the assignment $\ell_{v_0} = 0$, $\ell_{v_i} = \ell_{v_0} + i\epsilon$ and $\ell_{w_i} = \ell_{v_0} + 1 + i\epsilon$ is a correct representation. Observe that $H$ contains the cycle $w_Kv_{K-1}w_{K-2}w_{K-2} \cdots v_1w_1v_0w_0w_k$. See Fig. 10 for $K = 3$.

### 4.3.2 Predecessors of Poset $\mathcal{Rep}$

A representation $R' \in \mathcal{Rep}$ is a predecessor of $R \in \mathcal{Rep}$ if $R' < R$ and there is no representation $\bar{R} \in \mathcal{Rep}$ such that $R' < \bar{R} < R$. We denote the predecessor relation by $\prec$. In a general poset, predecessors may not exist. But they always exist for a poset of a discrete structure like ($\mathcal{Rep}$, $\leq$), since there are finitely many representations $\tilde{R}$ between any $R' < R$. Also, for any two representations $R' < R$, there exists a finite chain of predecessors $R' = \tilde{R}_0 < \tilde{R}_1 < \cdots < \tilde{R}_k = R$. For the poset ($\mathcal{Rep}$, $\leq$), the predecessor relation is easy to describe:

\[ \mathcal{Rep} \] Springer
Lemma 4.11  For \( \varepsilon = \frac{1}{K} \) and \( K \geq \frac{n}{2} \), the representation \( R' \) is a predecessor of \( R \) if and only if \( R' \) is obtained from \( R \) by applying one left-shifting operation.

Proof  Clearly, if \( R' \) is obtained from \( R \) by one left-shifting, it is its predecessor. On the other hand, suppose we have \( R' \prec R \). Let \( H \) be the obstruction digraph of \( R \) and \( \tilde{H} \) be the subgraph of \( H \) induced by the intervals having different positions in \( R \) and \( R' \). Then there are no directed edges from \( \tilde{H} \) to \( H \setminus \tilde{H} \) (otherwise \( R' \) would be an incorrect representation). According to Lemma 4.10, the digraph \( \tilde{H} \) is acyclic. Therefore, it contains at least one sink \( v_i \). By left-shifting \( v_i \) in \( R \), we create a correct representation \( \tilde{R} \in \mathcal{R} \). Clearly, \( R' \leq \tilde{R} \prec R \), and so \( R' \) is a predecessor of \( R \) if and only if \( R' = \tilde{R} \). \( \square \)

Again, the assumption on the value of \( \varepsilon \) is necessary. For example, in Fig. 10 the structure of \( \mathcal{R} \) is just a single chain where a predecessor of some representation is obtained by shifting all intervals by \( \varepsilon \) to the left.

4.3.3 Proof of Left-Shifting Proposition

The main proposition of this subsection is a simple corollary of Lemma 4.11.

Proof (Proposition 4.8)  The left-most representation \( R \) is \( \inf(\mathcal{R}) \), so it has no predecessors and nothing can be left-shifted. On the other hand, if \( \inf(\mathcal{R}) \prec R \), there is a chain of predecessors in between, which implies using Lemma 4.11 that it is possible to left-shift some interval. \( \square \)

4.4 Preliminaries for the Shifting Algorithm

Before describing the shifting algorithm, we present several results which simplify the graph and the description of the algorithm.

4.4.1 Pruned Graph

The obstruction digraph \( H \) may contain many edges since each vertex \( v_i \) can have many obstructions. But if \( v_i \) has many, say, left obstructions, these obstructions have to be in the same position. If two intervals \( u \) and \( v \) have the same position in a correct unit interval representation, then \( N[u] = N[v] \) and they are indistinguishable. Our goal is to construct a pruned graph \( G' \) which replaces each group of indistinguishable
vertices of $G$ by a single vertex. This construction is not completely straightforward since indistinguishable vertices may have different lower and upper bounds.

Let $\{\Gamma_1, \ldots, \Gamma_k\}$ be the partitioning of $V(G)$ by groups of indistinguishable vertices (the groups are ordered by $\prec$ from left to right). We construct a unit interval graph $G'$, where the vertices are $\gamma_1, \ldots, \gamma_k$ with $\text{lb}(\gamma_i) = \max\{\text{lb}(v_j) : v_j \in \Gamma_i\}$, and the edges $E(G')$ correspond to the edges between the groups of $G$.

Suppose that we have the left-most representation $R'$ of the pruned graph $G'$ and we want to construct the left-most representation $R$ of $G$. Let $\Gamma_\ell$ be a group. We place each interval $v_i \in \Gamma_\ell$ as follows. Let $\gamma_\ell \prec$ be the first non-neighbor of $\gamma_\ell$ on the left and $\gamma_\ell \rightarrow$ be the right-most neighbor of $\gamma_\ell$ (possibly $\gamma_\ell \rightarrow = \gamma_\ell$). We set

$$\ell_i = \max\{\text{lb}(v_i), \ell_{\gamma_\ell \prec} + 1 + \epsilon, \ell_{\gamma_\ell \rightarrow} - 1\},$$

and if $\gamma_\ell \prec$ does not exist, we ignore it in max. The idea of this formula is to place each interval as far to the left as possible while maintaining the structure of $R'$. Figure 11 contains an example of the construction of $R$.

Before proving the correctness of the construction of $R$, we show two general properties of the formula (7). The first lemma states that each interval $v_i \in \Gamma_\ell$ is not placed in $R$ too far from the position of $\gamma_\ell$ in $R'$.

**Lemma 4.12** For each $v_i \in \Gamma_\ell$, it holds

$$\ell_{\gamma_\ell \prec} - 1 \leq \ell_i \leq \ell_{\gamma_\ell \rightarrow}.$$

**Proof** Since $\ell_{\gamma_\ell \prec} \leq \ell_{\gamma_\ell \rightarrow}$, we have $\ell_{\gamma_\ell \prec} - 1 \leq \ell_{\gamma_\ell \rightarrow} - 1 \leq \ell_i$. Because $R'$ is a correct bounded representation, $\ell_{\gamma_\ell}$ is greater than or equal to each term in (7), and we obtain the second inequality. \hfill \qed

The second lemma states that the representations $R$ and $R'$ are intertwining each other. That is, if $R$ is drawn on top of $R'$, then the vertices of each group $\Gamma_\ell$ are between $\gamma_{\ell-1}$ and $\gamma_\ell$; see Fig. 11.

**Lemma 4.13** For each $v_i \in \Gamma_\ell$ and $\ell > 1$, it holds

$$\ell_{\gamma_{\ell-1}} < \ell_i \leq \ell_{\gamma_\ell}.$$

**Proof** The second inequality has been proved in the preceding lemma. For the first inequality, we distinguish two cases:
– Suppose first that $\gamma'_{\ell}$ is a neighbor of $\gamma_{\ell-1}$. Since $\gamma'_{\ell-1} \in E(G')$ and $R'$ is a correct representation, we have $\ell_{\gamma'_{\ell-1}} \leq \ell_{\gamma_{\ell}} + 1$. By (7), $\ell_{\gamma'_{\ell}} + 1 < \ell_{i}$.

– Next assume that $\gamma'_{\ell}$ is not a neighbor of $\gamma_{\ell-1}$. Since $\gamma_{\ell-1}$ and $\gamma_{\ell}$ are not indistinguishable, then necessarily $\gamma'_{\ell-1}$ is a non-neighbor of $\gamma_{\ell-1}$. Thus $\ell_{\gamma_{\ell-1}} < \ell_{\gamma'_{\ell}} - 1$. By (7), $\ell_{\gamma'_{\ell}} - 1 \leq \ell_{i}$.

Now, we are ready to show correctness of the construction of $R$.

**Proposition 4.14** From the left-most representation $R'$ of the pruned graph $G'$, we can construct the correct left-most representation $R$ of $G$ by placing the intervals according to (7).

**Proof** Let $v_i$ and $v_j$ be a pair of vertices of $G$. Let $v_iv_j \in E(G)$. If $v_i$ and $v_j$ belong to the same group $\Gamma^v$, they intersect each other at position $\ell_{\gamma_{\ell}}$ by (8). Otherwise let $v_i \in \Gamma_{\ell}$ and $v_j \in \Gamma_{\ell'}$, and assume that $\Gamma_{\ell} < \Gamma_{\ell'}$. Then $\ell_{i} \leq \ell_{\gamma_{\ell}} \leq \ell_{j}$ by the intertwining property (9). Also, $\ell_{i} \leq \ell_{\gamma_{\ell'}} \leq \ell_{\gamma'_{\ell'}} \leq \ell_{i} + 1$ since $\gamma'_{\ell'}$ is a right neighbor of $\gamma_{\ell}$ and (8). Therefore, $\ell_{i} \leq \ell_{j} \leq \ell_{i} + 1$ and $v_i$ intersects $v_j$ in $R$. Now, let $v_i / v_j \not\in E(G)$, $v_i \in \Gamma_{\ell}$, $v_j \in \Gamma_{\ell'}$ and $v_i < v_j$. Then $\ell_{i} \leq \ell_{\gamma_{\ell}} \leq \ell_{\gamma'_{\ell'}} \leq \ell_{j} - 1 - \epsilon$ by (7) and (8), so $v_i$ and $v_j$ do not intersect. So the assignment $R'$ is a correct representation of $G$.

It remains to show that $R$ is the left-most representation of $G$. We can identify each $\gamma_{\ell}$ with one interval $v_i \in \Gamma_{\ell}$ having $\text{lbound}(v_i) = \text{lbound}(\gamma_{\ell})$; for an example see Fig. 11. So $G'$ can be viewed as an induced subgraph of $G$. We want to show that the intervals of $G'$ are represented in the same position in $R$ and in $R'$. Since $R|_{G'}$ (which denotes $R$ restricted to $G'$) is some representation of $G'$ and $R'$ is the left-most representation of $G'$, we get $\ell'_{\gamma_{\ell}} \leq \ell_{\gamma_{\ell}}$ for every $\gamma_{\ell}$. By (8), we get $\ell'_{\gamma_{\ell}} = \ell_{\gamma_{\ell}}$.

We know that $R|_{G'}$ is the left-most representation, or in other words each interval of $G'$ is fixed in $R$. The rest of the intervals are placed so that they are either trivially fixed by $\ell_i = \text{lbound}(v_i)$, or they have as obstructions some fixed intervals from $G'$, in which case they are fixed by Lemma 4.9. Therefore, every interval of $G$ is fixed and $R$ is the left-most representation.

For the pruned graph $G'$, the obstruction digraph $H$ has in- and out-degree at most two. Each interval has at most one left obstruction and at most one right obstruction, and these obstructions are always the same intervals. More precisely, if $v_j$ is a left obstruction of $v_i$, then $v_j = v_{\ell-\epsilon}$, whereas if $v_j$ is a right obstruction of $v_i$, then $v_j = v_{\ell+\epsilon}$.

The pruning operation can be done in time $O(n + m)$, so we may assume that our graph $G$ is already pruned and contains no indistinguishable vertices. The structure of obstructions in $G$ can be computed in time $O(n + m)$ as well.

**4.4.2 Position Cycle**

For each interval in some $\epsilon$-grid representation, we can write its position in this form:

$$\ell_i = \alpha_i + \beta_i \epsilon, \quad \alpha_i \in \mathbb{Z}, \quad \beta_i \in \mathbb{Z}_K,$$

where $\epsilon = \frac{1}{K}$. In other words, $\alpha_i$ is the integer position of $v_i$ in the grid and $\beta_i$ describes how far this interval is from this integer position.
Concerning left-shifting, the values $\beta_i$ are more important. We can depict $\mathbb{Z}_K = \{0, \ldots, K-1\}$ as a cycle with $K$ vertices where the value decreases clockwise. The value $\beta_i$ assigns to each interval $v_i$ one vertex of the cycle. The cycle $\mathbb{Z}_K$ together with marked positions of $\beta_i$’s is called the position cycle. A vertex of the position cycle is called taken if some $\beta_i$ is assigned to it, and empty otherwise. The position cycle allows us to visualize and work with left-shifting very intuitively. When an interval $v_i$ is left-shifted, $\beta_i$ cyclically decreases by one, so $\beta_i$ moves clockwise along the cycle. For an illustration, see Fig. 12.

If $(v_i, v_j)$ is a left edge of $H$, then $\beta_j = \beta_i - 1$, and if $(v_i, v_j)$ is a right edge, then $\beta_i = \beta_j$. So if $v_j$ is an obstruction of $v_i$, $\beta_j$ has to be very close to $\beta_i$ (either at the same position or at the next clockwise position). If there is a big empty space in the clockwise direction from $\beta_i$, the interval $v_i$ can be left-shifted many times (or till it becomes fixed by $\ell_i = \text{lb}(v_i)$). Notice that if $\beta_i$ is very close to $\beta_j$, it does not mean that $\ell_i$ is very close to $\ell_j$ because the values $\alpha_i$ and $\alpha_j$ are ignored in the position cycle.

### 4.5 The Shifting Algorithm for BOUNDREP

We want to solve an instance of BOUNDREP with a prescribed ordering $\preceq$. We work with an $\varepsilon$-grid which is different from the one in Sect. 3.1. The new value of $\varepsilon$ is the value given by (1) refined $n$ times, so

$$\varepsilon = \frac{1}{n^2} \cdot \varepsilon'.$$

Lemma 3.1 applies for this value of $\varepsilon$ as well, so if the instance is solvable, there exists a solution which is on this $\varepsilon$-grid.

The algorithm solves the linear program of Sect. 4.1 in time $O(k^2 + kD(r))$, for a component with $k$ vertices. We assume that the input component is already pruned, otherwise we prune it and use Proposition 4.14 to complete the representation. We expect that the left-to-right order $\preceq$ of the vertices is given. The algorithm requires time $O(kD(r))$, since the bounds are given in the form $\frac{p_i}{q_i}$ and we need to perform arithmetic operations with these bounds.

\[\text{Springer}\]
4.5.1 Overview

The algorithm for solving one linear program works in three basic steps:

1. We construct an initial $\varepsilon$-grid representation (in the ordering $\prec$) having $\ell_i \geq \text{lbound}(v_i)$ for all intervals, using the algorithm of Corneil et al. [10].
2. We shift the intervals to the left while maintaining correctness of the representation until the left-most representation is constructed, using Proposition 4.8.
3. We check whether the left-most representation satisfies the upper bounds. If it does, this representation solves the linear program of Sect. 4.1 and minimizes $E_t$. Otherwise, the left-most representation does not satisfy the upper bound constraints. Thus by Lemma 4.7 no representation satisfies the upper bound constraints, and the linear program has no solution.

4.5.2 Input Size

Let $r$ be the size of the input describing bound constraints. A standard complexity assumption is that we can operate with polynomially large numbers (having $O(\log r)$ bits in binary) in constant time, to avoid the extra factor $O(\log r)$ in the complexity of most of the algorithms. However, the value of $\varepsilon$ given by (1) might require $O(r)$ digits when written in binary. The assumption that we can compute with numbers having $O(r)$ digits in constant time would break most of the computational models. Therefore, our computational model requires a larger time for arithmetic operations with numbers having $O(r)$ digits in binary. For example, the best known algorithm for multiplication/division in a Turing machine requires time $O(D(r))$.

A straightforward implementation of our algorithm working with the $\varepsilon$-grid would require time $O(k^2 r^c)$ for some $c$, instead of $O(k^2 + kD(r))$. There is an easy way out: Instead of computing with long numbers having $O(r)$ digits, we mostly compute with short numbers having only $O(\log r)$ digits. Instead of the $\varepsilon$-grid, we mostly work in a larger $\Delta$-grid, where $\Delta = \frac{1}{\sqrt{n}}$. The algorithm computes with long numbers only in two steps. First, some initial computations concerning the input are performed. Second, when the shifting makes some interval fixed, the algorithm finds the final $\varepsilon$-grid position of the interval. As explained later, all these computations can be done in total time $O(kD(r))$.

4.5.3 Left-Shifting

The basic operation of the algorithm is the LEFTSHIFT procedure, which we describe as follows. We deal separately with fixed and unfixed intervals (some intervals might be fixed from the beginning). Unfixed intervals are on the $\Delta$-grid and fixed intervals have precise positions calculated on the $\varepsilon$-grid. We place only unfixed intervals on the position cycle for the $\Delta$-grid. At any moment of the algorithm, each vertex of the position cycle is taken by at most one $\beta_i$; this is true for the initial representation and the shifting maintains this property.

The procedure LEFTSHIFT($v_i$) shifts $v_i$ from the position $\ell_i$ into a new position $\ell'_i$ such that the representation remains correct. It consists of two steps:
(1) Since \(v_i\) is unfixed, it has some \(\beta_i\) placed on the position cycle. Let \(k\) be such that the vertices \(\beta_i + 1, \ldots, \beta_i + k\) of the position cycle are empty, and the vertex \(\beta_i + k + 1\) is taken by some \(\beta_b\). Then the candidate for the new position of \(v_i\) is \(\bar{\ell}_i = \ell_i - k \Delta\).

(2) We need to ensure that this shift from \(\ell_i\) to \(\bar{\ell}_i\) is valid with respect to \(\text{lboud}(v_i)\) and the positions of the fixed intervals. Concerning the lower bound, we cannot shift further than \(\text{lboud}(v_i)\). Concerning the fixed intervals, the shift is limited by positions of fixed obstructions of \(v_i\). If \(v_j\) is a fixed left obstruction, we cannot shift further than \(\ell_j + 1 + \varepsilon\), and if \(v_j'\) a fixed right obstruction, we cannot shift further than \(\ell_j' - 1\).

The resulting position after applying \(\text{LeftShift} (v_i)\) is

\[
\ell'_i = \max\{\bar{\ell}_i, \text{lboud}(v_i), \ell_j + 1 + \varepsilon, \ell_j' - 1\}.
\]  

(11)

**Lemma 4.15** If the original representation \(R\) is correct, then the \(\text{LeftShift} (v_i)\) procedure produces a correct representation \(R'\).

**Proof** Clearly, the lower bound for \(v_i\) is satisfied in \(R'\). The shift of \(v_i\) from \(\ell_i\) to \(\ell'_i\) can be viewed as a repeated application of the left-shifting operation from Sect. 4.3. We just need to argue that each left-shifting operation can be applied till the position \(\ell'_i\) is reached.

If at some point, the left-shifting operation could not be applied, there would have to be some obstruction \(v_j\) of \(v_i\). There is no unfixed obstruction since all vertices of the position cycle \(\beta_i + 1, \ldots, \beta_i + k\) are empty. And \(v_j\) cannot be fixed as well since we check positions of both possible obstructions. So there is no obstruction \(v_j\). \(\Box\)

After \(\text{LeftShift} (v_i)\), if \(\bar{\ell}_i\) is not a strict maximum of the four terms in (11), the interval \(v_i\) becomes fixed; either trivially because \(\ell'_i = \text{lboud}(v_i)\), or by Lemma 4.9 since \(v_i\) becomes obstructed by some fixed interval. In such a case, we remove \(\beta_i\) from the position cycle.

### 4.5.4 Fast Implementation of Left-Shifting

Since we apply the \(\text{LeftShift}\) procedure repeatedly, we want to implement it in time \(O(1)\). In (11), the first term \(\bar{\ell}_i\) is a short number (on the \(\Delta\)-grid) and the remaining terms are long numbers (on the \(\varepsilon\)-grid). We first compare \(\bar{\ell}_i\) to the remaining terms, which can be done in time \(O(1)\) (Lemma 4.16). If \(\bar{\ell}_i\) is a strict maximum, we use it for \(\ell'_i\). Otherwise, we need to compute the maximum of the remaining three terms, which takes time \(O(D(r))\). But then the interval \(v_i\) becomes fixed, so this costly step is done exactly \(k\) times, and takes total time \(O(k D(r))\).

**Lemma 4.16** With total precomputation time \(O(k D(r))\), it is possible to compare \(\bar{\ell}_i\) to the remaining terms in (11) in time \(O(1)\) per \(\text{LeftShift}\) procedure.
Proof Initially, we do the following precomputation for the lower bounds. We have \( b \) lower bounds given by the input in the form \( \frac{p_1}{q_1}, \ldots, \frac{p_b}{q_b} \) as irreducible fractions. For each bound, we first compute its position \((\alpha_i, \beta_i)\) on the \( \epsilon \)-grid; see (10).

If \( \text{lbound}(v_i) \ll \text{lbound}(v_j) \) for some vertices \( v_i \) and \( v_j \), then \( \text{lbound}(v_i) \) is never achieved because the graph is connected and every representation takes space at most \( k \). Therefore we can increase \( \text{lbound}(v_i) \) without changing the solution of the instance. More precisely, let \( \alpha = \max \alpha_i \). Then we modify each bound by setting \( \alpha_i := \max \{ \alpha - k - 1, \alpha_i \} \). In addition, we shift all the bounds by subtracting a constant \( C \) such that \( \alpha_i - C \in [0, k + 1] \) for each \( \alpha_i \). Concerning \( \beta_i \), we round the position \((\alpha_i, \beta_i)\) down to a position \((\alpha_i, \overline{\beta_i})\) of the \( \Delta \)-grid. These precomputations can be done for all lower bounds in time \( O(kD(r)) \).

Suppose that we want to find out whether \( \tilde{\ell}_i \leq \text{lbound}(v_i) = \alpha_i + \beta_i \cdot \epsilon \), where \( \tilde{\ell}_i \) is in the \( \Delta \)-grid. Then it is enough to check whether \( \tilde{\ell}_i \leq \alpha_i + \overline{\beta_i} \Delta \), which can be done in constant time since both \( \alpha_i \) and \( \beta_i \) are short numbers. To compare \( \tilde{\ell}_i \) to the other terms in (11), notice that, when \( v_j \) or \( v_j' \) become fixed, their precise position is computed using (11). Then we also compute the values \( \ell_{j'} - 1 \) and \( \ell_j + 1 + \epsilon \) used in (11) and round them down to the \( \Delta \)-grid (since each interval becomes fixed only once, this rounding takes total time \( O(kD(r)) \)). Using these precomputed values, \( \tilde{\ell}_i \) can be compared to the remaining terms in (11) in time \( O(1) \).

Notice that the representation is built in a position shifted by \( C \), so after its construction it is necessary to shift it back.

4.5.5 Initial Representation

Recall that the position cycle has \( n^2 \) vertices and \( \Delta = \frac{1}{n^2} \). The algorithm of Corneil et al. [10] gives a representation in the \( \frac{1}{k} \)-grid. Using the proof of Lemma 3.1, we use this representation to construct the initial \( \Delta \)-grid representation. Then we shift it such that \( \ell_i \geq \text{lbound}(v_i) \) for each \( v_i \) and \( \ell_i \leq \text{lbound}(v_i) + \Delta \) for some \( v_i \). For this initial representation, each interval can be shifted to the left in total by at most \( O(k) \).

The initial representation places all intervals in such a way that \( \beta_i \)’s are positioned almost equidistantly in the position cycle; refer to the left-most position cycle in Fig. 13. For our purposes, it is enough that all \( \beta_i \)’s are placed in pairwise different vertices of the position cycle.
4.5.6 Shifting Phases

We repeatedly apply the \textsc{LeftShift} procedure in such a way that each interval is almost always shifted by almost one. When an interval becomes fixed, it is removed from the position cycle. The shifting of unfixed intervals proceeds in two phases:

- \textit{The first phase} creates one big gap by clustering all $\beta_i$’s in one part of the cycle. To do so, we apply the \textsc{LeftShift} procedure to each interval, in the order given by the position cycle. We obtain one big gap of size at least $n(n - 1)$. Refer to Fig. 13.

- \textit{In the second phase}, we use this big gap to shift intervals one by one, which also moves the cluster along the position cycle. The second phase finishes when each interval becomes fixed and the left-most representation is constructed. For an example, see Fig. 14.

4.5.7 Putting It All Together

First, we show correctness of the shifting algorithm and its complexity:

\begin{lemma}
For a component having $k$ vertices, the shifting algorithm constructs a correct left-most representation in time $O(k^2 + kD(r))$.
\end{lemma}

\begin{proof}
First, we argue correctness of the algorithm. The algorithm starts with an initial representation which is correct and satisfies the lower bounds. By Lemma 4.15, the \textsc{LeftShift} procedure maintains the correctness. Each interval can be shifted to the left by at most $O(k)$, and after each \textsc{LeftShift} procedure one interval is shifted by at least $\Delta$. Thus after finitely many applications of the \textsc{LeftShift} procedure, every interval becomes fixed, and we obtain the left-most representation.

Concerning complexity, all precomputations, as well as the computation of the final positions of the intervals when they become fixed, take total time $O(kD(r))$. Using Lemma 4.16, each \textsc{LeftShift($v_i$)} procedure can be performed in time $O(1)$ unless $v_i$ becomes fixed. In the first phase, the \textsc{LeftShift} procedure is applied $k - 1$ times. In each iteration of the second phase, each interval is shifted by at least $\frac{n-1}{n}$ (unless it becomes fixed). Since each interval can be shifted by at most $O(k)$ from its initial position, the second phase applies the \textsc{LeftShift} procedure $O(k^2)$ times. So the total running time of the algorithm is $O(k^2 + kD(r))$. \hfill \Box
\end{proof}
We are ready to prove that $\text{BOUNDREP}$ with a prescribe ordering $\lhd$ can be solved in time $O(n^2 + nD(r))$:

**Proof (Theorem 1.3)** As in the algorithm of Sect. 4.1, we process the components $C_1 \lhd \cdots \lhd C_c$ from left to right, and for each of them we solve two linear programs. For each linear program, we find the left-most representation using Lemma 4.17, and we test for this representation whether the upper bounds are satisfied. According to Lemma 4.7, the linear program is solvable if and only if the left-most representation satisfies the upper bounds, and clearly the left-most representation minimizes $E_i$. The time complexity of the algorithm is $O(n^2 + nD(r))$. \hfill $\Box$

We finally present an FPT algorithm for $\text{BOUNDREP}$ with respect to the number of components $c$. The algorithm is based on Theorem 1.3.

**Proof (Corollary 1.4)** There are $c!$ possible left-to-right orderings of the components of $G$. For each of them, by Theorem 1.3 we can decide in time $O(n^2 + nD(r))$ whether there exists a bounded representation respecting the order. In total, we spend time $O((n^2 + nD(r))c!)$. \hfill $\Box$

## 5 Extending Unit Interval Graphs

The $\text{REPEXT(UNIT INT)}$ problem can be solved using Theorem 1.3. It is enough to show that this problem is a particular instance of $\text{BOUNDREP}$ in which the ordering $\lhd$ of the components can be derived:

**Proof (Theorem 1.5)** The graph $G$ contains unlocated and located components. As in Sect. 2, unlocated components can be placed far to the right and we can deal with them using a standard recognition algorithm.

Concerning located components $C_1, \ldots, C_c$, they have to be ordered in $\mathcal{R'}$ from left to right, which gives the required ordering $\lhd$. Then it is straightforward to construct the instance of $\text{BOUNDREP}$ with this ordering $\lhd$: For each pre-drawn interval $v_i$ at position $\ell_i$, we put $l\text{bound}(v_i) = u\text{bound}(v_i) = \ell_i$. For the rest of the intervals, we set no bounds. Clearly, this instance of $\text{BOUNDREP}$ is equivalent to the original $\text{REPEXT(UNIT INT)}$ problem, and we can solve it in time $O(n^2 + nD(r))$ using Theorem 1.3. \hfill $\Box$

## 6 Conclusions

### 6.1 Polyhedron Interpretation

Consider the linear program of Sect. 4.1. The described shifting algorithm has the following geometric interpretation. When the constraints (4) are omitted, all solutions of the linear program form an unbounded polyhedron. The initial solution is one point of the polyhedron and the left-most representation is the vertex of the polyhedron minimizing all values $\ell_i$. One application of the $\text{LEFTSHIFT}$ procedure corresponds to decreasing one variable while staying in the polyhedron. The algorithm computes a
Manhatten-like path from the initial solution to the left-most representation consisting of $O(n^2)$ shifts.

We believe that the polyhedron has some additional useful structure which might be exploited for constructing faster algorithms and might lead to discovering new properties of unit interval representations. It is also an interesting question whether some of our techniques can be generalized to other systems of difference constraints.

### 6.2 Simultaneous Representations

Let $G_1, \ldots, G_k$ be graphs having $V(G_i) \cap V(G_j) = I$ for each $i \neq j$. The SimRep($C$) problem asks whether there exists representations $R_1, \ldots, R_k$ of $G_1, \ldots, G_k$ (of class $C$) which assign the same sets to the vertices of $I$. This problem was considered in [19] and its relations to the partial representation extension problem were discussed in [5,26].

We believe that it might be possible to apply some of our techniques to solve the simultaneous representations problem for proper and unit interval graphs. A possible approach would be to construct simultaneous left-to-right orderings $<_1, \ldots, <_k$ having the same order on $I$, and then use linear programming/shifting to construct the simultaneous representation.

### 6.3 Open Problems

To conclude the paper, we present two open problems.

**Problem 6.1** Is it possible to solve the problem RepExt(Unit Int) in time faster than $O(n^2 + nD(r))$?

**Remark** Soulignac [35] recently announced a faster algorithm solving BoundRep with a prescribed ordering $\preceq$ of the connected components and RepExt(Unit Int) in time $O(n^2 + r)$, based on the work of Pierlot [30,31] and the approach of Sect. 4.1. It is open whether the problems can be solved in linear time $O(n + m + r)$.

We consider the next problem to be the major open problem concerning restricted representations of graphs. It involves circular-arc graphs (Circular-Arc), which is the class of intersection graphs of arcs of a circle (for references see [36]). We ask the following question:

**Problem 6.2** Can the problem RepExt(Circular-Arc) be solved in polynomial time?

We believe that solving this problem might lead to a better understanding of the class itself. All known polynomial-time recognition algorithms are quite complex, and construct specific types of representations called canonical representations. Further, many results concerning circular-arc graphs were later shown to be false; for instance recently the graph isomorphism problem of circular-arc graphs is again open. To solve RepExt(Circular-Arc), the structure of all representations needs to be better understood which could be a major breakthrough concerning this and other classes.
Acknowledgements  We would like to thank an anonymous reviewer for suggestions which improved understandability of Sect. 2. The first, second and sixth authors are supported by ESF Eurogiga project GraDR as GACR GIG/11/E023, the first two authors also by GACR 14-14179S and Charles University as GAUK 196213. The fourth author is supported by a fellowship within the Postdoc-Program of the German Academic Exchange Service (DAAD). The sixth author is supported by the project LO1506 of the Czech Ministry of Education, Youth and Sports, the project NEXLIZ—CZ.1.07/2.3.00/30.0038, which is co-financed by the European Social Fund and the state budget of the Czech Republic, and ESF EuroGIGA project ComPoSe as F.R.S.-FNRS—EUROGIGA NR 13604.

References

1. Angelini, P., Battista, G.D., Frati, F., Jelínek, V., Kratochvíl, J., Patrignani, M., Rutter, I.: Testing planarity of partially embedded graphs. ACM Trans. Algorithms 11(4), 32:1–32:42 (2015)
2. Balko, M., Klavík, P., Otachi, Y.: Bounded representations of interval and proper interval graphs. In: Algorithms and Computation—24th International Symposium, ISAAC 2013, Lecture Notes in Computer Science, vol. 8283, pp. 535–546. Springer, Berlin (2013)
3. Balof, B., Doignon, J.P., Fiorini, S.: The representation polyhedron of a semiorder. Order 30(1), 103–135 (2013)
4. Bang-Jensen, J., Huang, J., Zhu, X.: Completing Orientations of Partially Oriented Graphs. CoRR (2015). arXiv:1509.01301
5. Bläsius, T., Rutter, I.: Simultaneous PQ-ordering with applications to constrained embedding problems. In: SODA’13: Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1030–1043 (2013)
6. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms. J. Comput. Syst. Sci. 13, 335–379 (1976)
7. Chaplick, S., Dorbec, P., Kratochvíl, J., Montassier, M., Stacho, J.: Contact representations of planar graphs: extending a partial representation is hard. In: Graph-Theoretic Concepts in Computer Science—40th International Workshop, WG 2014, Lecture Notes in Computer Science, vol. 8747, pp. 139–151. Springer, Berlin (2014)
8. Chaplick, S., Fulek, R., Klavík, P.: Extending partial representations of circle graphs. In: Graph Drawing—21st International Symposium, GD 2013, Lecture Notes in Computer Science, vol. 8242, pp. 131–142. Springer, Berlin (2013)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
10. Corneil, D.G., Kim, H., Natarajan, S., Olariu, S., Sprague, A.P.: Simple linear time recognition of unit interval graphs. Inf. Process. Lett. 55(2), 99–104 (1995)
11. Corneil, D.G., Olariu, S., Stewart, L.: The LBFS structure and recognition of interval graphs. SIAM J. Discret. Math. 23(4), 1905–1953 (2009)
12. Deng, X., Hell, P., Huang, J.: Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. SIAM J. Comput. 25(2), 390–403 (1996)
13. Durán, G., Fernández Slezak, F., Grippo, L.N., Oliveira, F.d.S., Szwarcfiter, J.: On unit interval graphs with integer endpoints. In: LAGOS 2015 (2015). http://www.lia.ufc.br/lagos2015/docs/endm/ENDM50_74.pdf
14. Fürer, M.: Faster integer multiplication. SIAM J. Comput. 39(3), 979–1005 (2009)
15. Garey, M.R., Johnson, D.S.: Complexity results for multiprocessor scheduling under resource constraints. SIAM J. Comput. 4(4), 397–411 (1975)
16. Gilmore, P.C., Hoffman, A.J.: A characterization of comparability graphs and of interval graphs. Can. J. Math. 16, 539–548 (1964)
17. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. North-Holland, Amsterdam (2004)
18. Hajós, G.: Über eine Art von Graphen. Int. Math. Nachr. 11, 65 (1957)
19. Jampani, K.R., Lubiw, A.: Simultaneous interval graphs. In: Algorithms and Computation—21st International Symposium, ISAAC 2010, Lecture Notes in Computer Science, vol. 6506, pp. 206–217. Springer, Berlin (2010)
20. Jampani, K.R., Lubiw, A.: The simultaneous representation problem for chordal, comparability and permutation graphs. J. Graph Algorithms Appl. 16(2), 283–315 (2012)
21. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica 4(4), 373–395 (1984)
22. Klavík, P., Kratochvíl, J., Krawczyk, T., Walczak, B.: Extending partial representations of function graphs and permutation graphs. In: Algorithms—20th Annual European Symposium, ESA 2012, Lecture Notes in Computer Science, vol. 7501, pp. 671–682. Springer, Berlin (2012)
23. Klavík, P., Kratochvíl, J., Otachi, Y., Rutter, I., Saitoh, T., Saumell, M., Vyskočil, T.: Extending partial representations of proper and unit interval graphs. In: Algorithm Theory—14th Scandinavian Symposium and Workshops, SWAT 2014, Lecture Notes in Computer Science, vol. 8503, pp. 253–264. Springer, Berlin (2014)
24. Klavík, P., Kratochvíl, J., Otachi, Y., Saitoh, T.: Extending partial representations of subclasses of chordal graphs. Theor. Comput. Sci. 576, 85–101 (2015)
25. Klavík, P., Kratochvíl, J., Otachi, Y., Saitoh, T., Vyskočil, T.: Extending partial representations of interval graphs. CoRR (2013). arXiv:1306.2182
26. Klavík, P., Kratochvíl, J., Vyskočil, T.: Extending partial representations of interval graphs. In: Theory and Applications of Models of Computation—8th Annual Conference, TAMC 2011, Lecture Notes in Computer Science, vol. 6648, pp. 276–285. Springer, Berlin (2011)
27. Klavík, P., Otachi, Y., Šejnoha, J.: On the classes of interval graphs of limited nesting and count of lengths. CoRR (2015). arXiv:1510.03998
28. Klavík, P., Saumell, M.: Minimal obstructions for partial representations of interval graphs. In: Algorithms and Computation—25th International Symposium, ISAAC 2014, Lecture Notes in Computer Science, vol. 8889, pp. 401–413. Springer, Berlin (2014)
29. Patrignani, M.: On extending a partial straight-line drawing. Int. J. Found. Comput. Sci. 17(5), 1061–1070 (2006)
30. Pirlot, M.: Minimal representation of a semiorder. Theory Decis. 28, 109–141 (1990)
31. Pirlot, M.: Synthetic description of a semiorder. Discret. Appl. Math. 31(3), 299–308 (1991)
32. Roberts, F.S.: Representations of Indifference Relations. Ph.D. Thesis, Stanford University (1968)
33. Roberts, F.S.: Indifference graphs. In: Harary, F. (ed.) Proof Techniques in Graph Theory, pp. 139–146. Academic Press, London (1969)
34. Rose, D.J., Tarjan, R.E., Lueker, G.S.: Algorithmic aspects of vertex elimination on graphs. SIAM J. Comput. 5(2), 266–283 (1976)
35. Soulignac, F.J.: Bounded, minimal, and short representations of unit interval and unit circular-arc graphs. CoRR (2014). arXiv:1408.3443
36. Spinrad, J.P.: Efficient Graph Representations. Field Institute Monographs (2003)