Interval Graph Representation with Given Interval and Intersection Lengths[☆]

Johannes Köbler[a], Sebastian Kuhnert[a,1,*], Osamu Watanabe[b,2]

[a] *Humboldt-Universität zu Berlin, Institut für Informatik, Germany*
[b] *Tokyo Institute of Technology, Dept. of Mathematical and Computing Sciences, Japan*

Abstract

We consider the problem of finding interval graph representations that additionally respect given interval lengths and/or pairwise intersection lengths, which are represented as weight functions on the vertices and edges, respectively. Pe'er and Shamir proved that the problem is NP-complete if only the former are given [SIAM J. Discr. Math. 10.4, 1997]. For the case when both are given, Fulkerson and Gross gave an $\mathcal{O}(n^2)$ time algorithm [Pacif. J. Math. 15.3, 1965]; we improve this to $\mathcal{O}(n+m)$ time and supplement it with a logspace algorithm. For the case when only the latter are given, we give both an $\mathcal{O}(nm)$ time algorithm and a logspace algorithm. In all these bounds, n is the number of vertices and m is the number of edges in the input graph.

Complementing their hardness result, Pe'er and Shamir give a polynomial-time algorithm for the case that the input graph has a unique interval ordering of its maximal cliques. For such graphs, their algorithm computes an interval representation (if it exists) that respects a given set of distance inequalities between the interval endpoints. We observe that deciding if such a representation exists is NL-complete.

Keywords: Constrained graph representation, intersection graph, interval graph, linear time, logspace

1. Introduction

The *interval representation problem* asks for a given graph $G = (V, E)$, if G is an interval graph, and if so, to compute an interval representation ρ for G, i.e., ρ determines for each vertex $u \in V$ an interval $\rho(u)$ such that $E = \big\{\{u, v\} \,\big|\, \rho(u) \cap \rho(v) \neq \varnothing\big\}$. Booth and Lueker [BL76] solve this problem in linear time, introducing the widely used concept of a PQ-tree to succinctly encode all *interval orderings* of the (inclusion-)maximal cliques of G (i.e., all orderings of the maximal cliques such that each vertex appears in consecutive maximal cliques). Hsu and Ma [HM99] give a simpler linear-time algorithm that relies on modular decomposition instead. Corneil et al. [COS09] show a further simplification, avoiding ordering the maximal cliques, by using lexicographic breadth first search. Klein [Kle96] gave a parallel AC^2 algorithm. Köbler et al. [KKLV11] show that the interval representation problem is complete for logspace.

Algorithmic aspects of interval graphs have been the subject of ongoing research for several decades, stimulated by their numerous applications; see e.g. [Gol04]. In some applications, interval representations with special properties are required. In scheduling, for example, tasks can have certain durations that should be reflected by the lengths of their intervals; and two consecutive tasks can require a certain handover period that determines by how much their intervals should intersect. Another early application of interval graphs is

[☆] An extended abstract of this article appeared in the proceedings of ISAAC 2012.
[*] Corresponding author
Email addresses: koebler@informatik.hu-berlin.de (Johannes Köbler), kuhnert@informatik.hu-berlin.de (Sebastian Kuhnert), watanabe@is.titech.ac.jp (Osamu Watanabe)
[1] Supported by DFG grants KO 1053/7–1 and KO 1053/7–2
[2] Supported in part by the ELC project (MEXT KAKENHI No. 24106008).

genome assembly; also here the length of the observed subsequences is known as well as the lengths of the overlaps between them. Throughout the paper, we use ℓ to denote a given weight function prescribing for each vertex $u \in V$ the length $\ell(u)$ of the interval $\rho(u)$, and s to denote a given weight function prescribing for each edge $\{u, v\} \in E$ the length $s(u, v)$ of the intersection $\rho(u) \cap \rho(v)$ of the intervals $\rho(u)$ and $\rho(v)$. An interval representation ρ is called *ℓ-respecting* if each interval $\rho(u)$ has the prescribed length $\ell(u)$, it is called *s-respecting* if each intersection $\rho(u) \cap \rho(v)$ has the prescribed length $s(u, v)$, and *(ℓ, s)-respecting* if it satisfies both conditions.

Fulkerson and Gross [FG65] gave an algorithm that finds (ℓ, s)-respecting interval representations in $\mathcal{O}(n^2)$ time. Pe'er and Shamir [PS97] showed that it is NP-complete to decide if a graph G admits an ℓ-respecting interval representation. For the restricted case that the given graph G is a UCO graph (i.e., up to reflection G has a unique interval ordering on its maximal cliques), the same authors gave a polynomial-time algorithm that can also handle more general constraints on differences between interval endpoints. The problem of finding s-respecting interval representations has also been investigated by Yamamoto [Yam07].

Our contribution

Using techniques from [KKLV11] we describe algorithms for constructing (ℓ, s)-respecting interval representations in linear time or alternatively in logspace, and s-respecting interval representations in $\mathcal{O}(nm)$ time or alternatively in logspace. Since computing ℓ-respecting interval representations is NP-hard, our result illustrates that the additional restriction imposed by the function s is quite helpful.

The first step towards our algorithms is to show that all interval representations ρ of the appropriate type have the same inclusion and overlap relationships between their intervals. For example, the equality $s(u, v) = \ell(u)$ enforces the inclusion $\rho(u) \subseteq \rho(v)$. We also show how these relations can be efficiently computed when G, s (and ℓ) are given as input.

In the next step we focus on graphs with overlap-connected representations and show that up to reflection, these graphs have a unique (ℓ, s)-respecting representation. For graphs with several overlap components we arrange these components into a tree, and combine their (ℓ, s)-respecting interval representations into one for the whole graph. We also show that all (ℓ, s)-respecting interval representations are isomorphic (meaning that arbitrary intersections of intervals or complemented intervals must have a fixed length that only depends on ℓ and s). This provides an alternative proof for [FG65, Theorem 2.1].

In order to compute s-respecting interval representations efficiently, we repeatedly use our algorithm for computing an (ℓ, s)-respecting interval representation as a subroutine. We prove that the lengths of the pairwise intersections uniquely determine the interval lengths if we require that the resulting s-respecting interval representation is minimal, i.e., it contains a minimum number of points. We also show that all minimal s-respecting interval representations are isomorphic.

Furthermore, we consider two variants of the interval representation problem for which Pe'er and Shamir [PS97] gave polynomial time algorithms for the case that the input graph G is restricted to be a UCO graph. The first variant is called BIG (short for *bounded interval graph recognition*) and allows the specification of lower and/or upper bounds on the lengths of the intervals representing the vertices of G. In the second variant (called DCIG, short for *distance-constrained interval graph*) it is even possible to prescribe lower and/or upper bounds on the distances between arbitrary interval endpoints. We observe that the decision versions of both problems (i.e., to decide whether there is an interval representation for a given UCO graph G fulfilling all given lower and upper bounds) are NL-complete, provided that these bounds are polynomially bounded. Hence, it is unlikely that these generalizations of the ℓ-respecting interval representation problem are solvable in deterministic logspace even for the class of UCO graphs. This contrasts with our logspace algorithm for s-respecting interval representation of general graphs.

Related work

There are several related notions of constrained interval representation problems, many of which are motivated by scheduling and artificial intelligence applications. In an influential paper, Allen [All83] classified the possible temporal relations between intervals, which give rise to an *interval algebra* consisting of all unions of these relations. In a series of results, constraint satisfaction problems over various subalgebras of this

interval algebra have been considered. See [KJJ03] for the final classification into tractable and NP-complete cases, which also contains a survey of previous results. While Allan's interval algebra and its subalgebras focus on ordering the intervals, several extensions have been studied that additionally allow constraints on interval lengths. Krokhin et al. [KJJ04] study the case where disjunctions of linear relations on the intervals are given and each of these disjunctions contains at most one relation other than $\neq$. They characterize which subalgebras of Allen's interval algebra remain tractable after this addition.

Skrien [Skr84] considers a variant of the interval representation problem where the endpoints of the desired representation must be placed according to a given partial order, giving an $\mathcal{O}(n^3)$ time algorithm that finds such a representation or detects that none exists. Jampani and Lubiw [JL10] give an $\mathcal{O}(n^2 \log n)$ time algorithm that computes *simultaneous interval representations* for two given graphs, i.e., vertices that are present in both graphs must be mapped to the same interval in both representations. Klavík et al. [KKV11] describe an $\mathcal{O}(n^2)$ time algorithm for *extending partial interval representations,* where an interval representation ρ_0 of an induced subgraph of G is given as additional input and the resulting interval representation of G is required to agree with ρ_0 on the vertices of this subgraph. Bläsius and Rutter [BR13] give linear time algorithms for both simultaneous interval representation and for extending partial interval representations. They obtain these algorithms by giving linear time reductions from the latter to the former and from the former to a special case of *simultaneous PQ-ordering*, i.e., the problem of finding a linear order that is compatible with a given set of PQ-trees. They give a linear time algorithm for this special case and also show that simultaneous PQ-ordering is NP-complete in general.

An interval graph is called *proper* if it admits an interval representation where no interval contains another one. Such representations can be found in linear time using algorithms by Deng et al. [DHH96] and by Hell et al. [HSS01]. Köbler et al. [KKLV11] give a logspace algorithm for the same task. A generalization of proper interval graphs are the *interval matrices* studied by McConnell [McC03], which prescribe the desired relation between each pair of intervals as either disjoint, contained, containing, or overlapping. McConnell [McC03] shows how representations of interval matrices can be computed in linear time; a logspace algorithm is given by Köbler et al. [KKV13].

Generalizing the work on extending partial interval representations, Balko et al. [BKO13] give a linear time algorithm for the *bounded interval representation problem*, where each interval endpoint must be placed in a prescribed range. They also give an $\mathcal{O}(n^2)$ time algorithm for the case that the interval representation is additionally required to be proper. Klavík et al. [KKO$^+$14] show that the bounded interval representation problem is NP-complete for unit interval representations, i.e., when all intervals must have the same length. They also give a linear time algorithm for extending partial proper interval representations and an algorithm for extending unit interval representations that runs in $\mathcal{O}(n^2)$ time if the endpoints of the prescribed intervals have the same denominator.

Organization of the paper

In Section 3 we show that all (ℓ, s)-respecting and all minimal s-respecting interval representations have the same inclusion and overlap relationships between their intervals. In Section 4 we describe our algorithm for computing (ℓ, s)-respecting interval representations and in Section 5 we show how to compute s-respecting interval representations efficiently. In Section 6, we show that the problems BIG and DCIG are both NL-complete on UCO graphs.

2. Preliminaries

We say that two sets A and B *overlap* and write $A \between B$, if $A \cap B \neq \varnothing$, $A \setminus B \neq \varnothing$, and $B \setminus A \neq \varnothing$. The cardinality of a finite set A is denoted by $|A|$. By $\mathbb{Z}^+$ we denote the set of positive integers.

For a graph $G = (V, E)$, the number of its vertices is denoted by n, the number of its edges is denoted by m, and the set of neighbors of a vertex $v \in V$ is denoted by $N(v)$. The graph G is called *interval* if there is a system $\mathcal{I}$ of nonempty intervals over $\mathbb{Z}^+$ (we allow $\mathcal{I}$ to be a multiset) and a bijection $\rho\colon V \to \mathcal{I}$ such that for all $u, v \in V$ with $u \neq v$, it holds that $\{u, v\} \in E \Leftrightarrow \rho(u) \cap \rho(v) \neq \varnothing$. In this case, ρ is called an *interval representation* of G, and $\mathcal{I}$ is called an *interval model* of G. The latter is also denoted by $\rho(G)$.

For $a, b \in \mathbb{Z}$ with $a \leq b$, we write $[a, b]$ to denote the interval $I = \{i \in \mathbb{Z} \mid a \leq i \leq b\}$ of integers. With the *length* $|I|$ of I we denote the number of points $i \in I$.[3] For an interval system $\mathcal{I}$ we always suppose $\bigcup_{I \in \mathcal{I}} I = [1, k]$ for some k, i.e., we disallow gaps and shifting the whole system. An interval system $\mathcal{I}$ can be regarded as a hypergraph with nodes $[1, k]$ and hyperedges $\mathcal{I}$. Two interval systems $\mathcal{I}$ and $\mathcal{I}'$ with points $[1, k]$ are *isomorphic* if they are isomorphic as hypergraphs, i.e., if there is a permutation $\pi\colon [1, k] \to [1, k]$ of the points that induces a bijection between the intervals of $\mathcal{I}$ and $\mathcal{I}'$ (preserving multiplicities). We call two interval representations ρ_1 and ρ_2 of a simple graph G isomorphic if the resulting interval models $\rho_1(G)$ and $\rho_2(G)$ are isomorphic. The *slots* of $\mathcal{I}$ are the equivalence classes on $[1, k]$ with respect to containment in the intervals in $\mathcal{I}$. That is, two points are in the same slot, if each interval in $\mathcal{I}$ contains either both or none of them.

Let $\ell\colon V \to \mathbb{Z}^+$ and $s\colon E \to \mathbb{Z}^+$ be weight functions for a graph $G = (V, E)$. For convenience, we write $s(u, v)$ instead of $s(\{u, v\})$ for $\{u, v\} \in E$; for $\{u, v\} \notin E$ we let $s(u, v) = 0$. An interval representation $\rho\colon V \to \mathcal{I}$ of G is called *ℓ-respecting* if $|\rho(v)| = \ell(v)$ for all $v \in V$, *s-respecting* if $|\rho(u) \cap \rho(v)| = s(u, v)$ for all $\{u, v\} \in E$, and *(ℓ, s)-respecting* if both conditions hold. An s-respecting interval representation ρ of G is called *minimal* if there is no s-respecting interval representation ρ' of G that uses fewer points, i.e., that satisfies $\left|\bigcup_{v \in V} \rho'(v)\right| < \left|\bigcup_{v \in V} \rho(v)\right|$.

Logspace computations

As usual, L is the class of all languages decidable by deterministic Turing machines using only $\mathcal{O}(\log N)$ space on the work tapes, where N is the input size. The read-only input tape is exempt from this space bound. The class NL is defined using non-deterministic Turing machines that otherwise obey the same restrictions. The class FL contains all functions computable by deterministic Turing machines with the same restrictions; the result of the function is placed on an additional write-only output tape, which is, like the input tape, exempt from the space bound. Note that FL is closed under composition: To compute $f\big(g(x)\big)$ for $f, g \in \mathsf{FL}$, simulate the Turing machine for f and keep track of the position of its input head. Every time this simulation needs a character from f's input tape, simulate the Turing machine for g on input x until it outputs the required character. Note also that g can first output a copy of its input x and afterwards compute additional information to be used by f. This construction can be iterated a constant number of times, still preserving the logarithmic space bound. We will utilize this closure property in our logspace algorithms by employing pre- and post-processing steps.

By applying this closure property it is easy to generalize our logspace results to the case where the prescribed lengths are rational: Bring all lengths to a common denominator and use the resulting numerators. This transformation can be performed in logspace as iterated integer multiplication is in $\mathsf{DLOGTIME}$-uniform TC^0 [HAB02].

It should be stressed that linear-time bounds are formally incomparable with logspace bounds. On the other hand, the membership of a computational problem in L implies the existence of logarithmic time parallel algorithms for this problem.

3. Deriving structural information

Let $G = (V, E)$ be a graph and let $\ell\colon V \to \mathbb{Z}^+$ and $s\colon E \to \mathbb{Z}^+$ specify the prescribed interval and intersection lengths. Let $\{u, v\} \in E$ be an edge. We say that vertex u *exceeds* vertex v in an interval representation ρ of G if $\rho(u) \setminus \rho(v) \neq \varnothing$. The following two relations $R_{\ell,s}, R_s \subseteq V \times V$ allow us to determine for each (ℓ, s)-respecting interval representation of G and for each minimal s-respecting interval representation of G, whether u exceeds v or not:

$$R_{\ell,s} = \big\{(u, v) \in V \times V \;\big|\; \{u, v\} \in E \wedge s(u, v) < \ell(u)\big\},$$
$$R_s = \big\{(u, v) \in V \times V \;\big|\; \{u, v\} \in E \wedge \exists w \in N(u) \setminus \{v\} : \min\{s(w, v), s(u, v)\} < s(w, u)\big\}.$$

The following lemma shows that these relations indeed have the intended meaning.

[3] This does not coincide with the usual notion of length $|I| = b - a$. However, if we replace I by the interval $(a - 0.5, b + 0.5)$ of reals, then both measures coincide.

Lemma 3.1. *Let ρ be an (ℓ, s)-respecting interval representation and let ρ' be a minimal s-respecting interval representation of a graph $G = (V, E)$. Then for each edge $\{u, v\} \in E$,*

(i) *$(u, v) \in R_{\ell,s}$ if and only if $\rho(u) \setminus \rho(v) \neq \varnothing$ and*
(ii) *$(u, v) \in R_s$ if and only if $\rho'(u) \setminus \rho'(v) \neq \varnothing$.*

Proof. Let $\{u, v\} \in E$ be an edge. Part (i) easily follows from the following equivalences:

$$s(u, v) < \ell(u) \Leftrightarrow |\rho(u) \cap \rho(v)| < |\rho(u)| \Leftrightarrow \rho(u) \setminus \rho(v) \neq \varnothing \,.$$

To show Part (ii), we first assume that $(u, v) \in R_s$. Then there is a vertex $w \in N(u) \setminus \{v\}$ such that $s(w, v) < s(w, u)$ or $s(u, v) < s(w, u)$. Either way, there must be a point $p \in \rho'(w) \cap (\rho'(u) \setminus \rho'(v))$, implying $\rho'(u) \setminus \rho'(v) \neq \varnothing$.

For the backward implication, consider a point $p \in \rho'(u) \setminus \rho'(v)$. By minimality of ρ', there is a vertex $w \in N(u)$ with $p \in \rho'(w)$. Note that $w \neq v$ by the choice of p. If $\rho'(w) \supset \rho'(u) \cap \rho'(v)$ (see Fig. 1(a)), it follows that $s(w, u) > s(u, v)$. Otherwise (see Fig. 1(b)), $\rho'(w) \cap \rho'(u) \supsetneq \rho'(w) \cap \rho'(v)$, and thus $s(w, u) > s(w, v)$. □

Figure 1: The two cases in the proof of Lemma 3.1(ii).

Lemma 3.2. *Given G, ℓ and s, the relations $R_{\ell,s}$ and R_s can be enumerated in time $\mathcal{O}(m)$ and $\mathcal{O}(nm)$, respectively, and both can be enumerated in logspace.*

Proof. Both $R_{\ell,s}$ and R_s are defined by first order formulas; these can easily be evaluated in logspace. Iterating over all pairs $(u, v) \in V \times V$ gives logspace enumeration.

To enumerate $R_{\ell,s}$ in $\mathcal{O}(m)$ time, loop over all edges $\{u, v\} \in E$ (considering both orientations) and output (u, v) if $\ell(u) > s(u, v)$.

To enumerate R_s in $\mathcal{O}(nm)$ time, loop over all edges $\{w, u\} \in E$ (again, considering both orientations) and all nodes $v \in V \setminus \{w, u\}$, and output (u, v) if $s(w, u) > \min\{s(w, v), s(u, v)\}$. □

We write $u \between_{\ell,s} v$ if $(u, v) \in R_{\ell,s} \wedge (v, u) \in R_{\ell,s}$, and $u \subseteq_{\ell,s} v$ if $\{u, v\} \in E \wedge (u, v) \notin R_{\ell,s}$. The relations $\between_s$ and $\subseteq_s$ are defined analogously using R_s. By Lemma 3.1, these relations describe the situation in any appropriate representation of G. For example, we have $u \between_{\ell,s} v \Leftrightarrow \rho(u) \between \rho(v)$ in any (ℓ, s)-respecting interval representation ρ of G, and $u \between_s v \Leftrightarrow \rho'(u) \between \rho'(v)$ in any minimal s-respecting interval representation ρ' of G.

Given two intervals $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ with $I_1 \between I_2$, we say that I_1 overlaps I_2 *from the left* if $a_2 \in I_1$, and *from the right* if $b_2 \in I_1$.

Lemma 3.3. *Let ρ be any s-respecting interval representation of G. For any three vertices $v, w_1, w_2 \in V$ such that $\rho(w_1) \between \rho(v) \between \rho(w_2)$, the intervals $\rho(w_1)$ and $\rho(w_2)$ overlap $\rho(v)$ from the same side if and only if $s(w_1, w_2) > \min\{s(w_1, v), s(w_2, v)\}$.*

Note that this condition can be decided both in constant time and in logspace.

Proof. See Figure 2 for an illustration. If $\rho(w_1)$ and $\rho(w_2)$ overlap $\rho(v)$ from the same side, then $\rho(w_1)$ and $\rho(w_2)$ contain at least one common point outside $\rho(v)$, making their intersection larger than the minimum of $|\rho(w_1) \cap \rho(v)|$ and $|\rho(w_2) \cap \rho(v)|$.

Now suppose to the contrary that $\rho(w_1)$ and $\rho(w_2)$ overlap $\rho(v)$ from different sides. In this case $(\rho(w_1) \cap \rho(v)) \setminus \rho(w_2)$ and $(\rho(w_2) \cap \rho(v)) \setminus \rho(w_1)$ are both non-empty, implying that $|\rho(w_1) \cap \rho(w_2)|$ is smaller than both $|\rho(w_1) \cap \rho(v)|$ and $|\rho(w_2) \cap \rho(v)|$. □

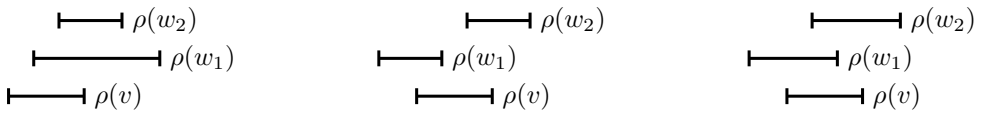

Figure 2: Three different ways how $\rho(w_1)$ and $\rho(w_2)$ can overlap $\rho(v)$, illustrating the proof of Lemma 3.3.

4. Given interval and intersection lengths

Let $G = (V, E)$ be a graph, and let $\ell\colon V \to \mathbb{Z}^+$ and $s\colon E \to \mathbb{Z}^+$ specify the prescribed interval and intersection lengths, respectively. In this section, we give linear-time and logspace algorithms that construct an (ℓ, s)-respecting interval representation of G, or detect that such a representation does not exist.

We define $E_{\ell,s} = \big\{\{u, v\} \in E \bigm| u \between_{\ell,s} v\big\}$ and $G_{\ell,s} = (V, E_{\ell,s})$, and call the connected components of $G_{\ell,s}$ the *overlap components* of G. As a first step, we consider overlap-connected graphs.

Lemma 4.1. *Given a graph $G = (V, E)$ and the functions $\ell\colon V \to \mathbb{Z}^+$ and $s\colon E \to \mathbb{Z}^+$ such that $G_{\ell,s}$ is connected, it is possible to compute in $\mathcal{O}(n + m)$ time (resp., in logspace) an (ℓ, s)-respecting interval representation ρ of G, or to detect that none exists. Moreover, ρ is unique up to reflection.*

Proof. We first compute the relations $R_{\ell,s}$, $\between_{\ell,s}$ and $\subseteq_{\ell,s}$; this can be done efficiently by Lemma 3.2. Let $v_1, v_2, \ldots, v_N$ be a walk in $G_{\ell,s}$ that visits every vertex at least once; such a walk can be constructed in linear time using depth first search or in logspace using Reingold's [Rei08] universal exploration sequences. The following algorithm computes an interval representation ρ of G by moving along this walk (which we assume has been computed in a pre-processing step). At each step of the walk, we compute an interval $I_i = [p_i, p_i + \ell(v_i) - 1]$ for v_i. If v_i is visited for the first time (i.e., there is no $j < i$ with $v_j = v_i$), we set $\rho(v_i) = I_i$. For the first two vertices of the walk, we define $p_1 = 1$ and $p_2 = \ell(v_1) - s(v_1, v_2) + 1$. Note that after I_1 has been placed, there are only two possibilities for I_2 that respect (ℓ, s); see Fig. 3 for an illustration. After that, all further intervals are uniquely determined because Lemma 3.3 allows to detect whether the next interval I_i has to overlap I_{i-1} from the same side as I_{i-2}. This allows to define

$$p_i = \begin{cases} p_{i-1} - \ell(v_i) + s(v_{i-1}, v_i) & \text{if } I_i \text{ overlaps } I_{i-1} \text{ from the left, and} \\ p_{i-1} + \ell(v_{i-1}) - s(v_{i-1}, v_i) & \text{otherwise.} \end{cases}$$

Note that I_i can be computed from the walk and the functions ℓ and s, remembering only the two previous intervals, so these intervals can be enumerated in logspace.

In a post-processing step, we check that ρ is indeed an (ℓ, s)-respecting interval representation of G, i.e., that $|\rho(v)| = \ell(v)$ for all $v \in V$ and $|\rho(u) \cap \rho(v)| = s(u, v)$ for all $\{u, v\} \in E$, and that $\rho(u) \cap \rho(v) \neq \varnothing$ implies $\{u, v\} \in E$. To verify the latter in linear time, the algorithm iterates over the endpoints of the interval model $\rho(V)$ from left to right, maintaining a set of vertices whose intervals contain the current point, and verifies that each vertex whose left endpoint is the current point has edges to all vertices in this set.

In a second post-processing step, we shift the resulting intervals such that 1 becomes the smallest point.

If there is an (ℓ, s)-respecting interval representation ρ_0 of G at all, an easy induction along the walk shows that ρ is a (possibly shifted and reflected) copy of ρ_0, and thus also an (ℓ, s)-respecting interval representation of G. This shows that the algorithm always finds an appropriate interval representation if there is one, and proves also the uniqueness part of the lemma. □

The next step is to generalize Lemma 4.1 to the case that $G_{\ell,s}$ is not connected. We can assume that there are no vertices v and v' such that both $v \subseteq_{\ell,s} v'$ and $v' \subseteq_{\ell,s} v$ hold; otherwise we compute an (ℓ, s)-respecting interval representation ρ for the graph $G' = G \setminus \{v' \mid \exists v < v' : v \subseteq_{\ell,s} v' \wedge v' \subseteq_{\ell,s} v\}$ and extend it to G afterwards. Let $\mathcal{C} = \{G_1, \ldots, G_k\}$ be the connected components of $G_{\ell,s}$. We write $G_i \leq_{\ell,s} G_j$ if $i = j$ or if there are vertices u in G_i and v in G_j such that $u \subseteq_{\ell,s} v$. The latter implies that, for any (ℓ, s)-respecting interval representation ρ of G, the interval $\bigcup_{u \in G_i} \rho(u)$ is contained in some slot $S \subseteq \rho(v)$ of the interval system $\rho(G_j)$, because otherwise there would be an overlap-path between $\rho(G_i)$ and $\rho(G_j)$. Thus $\leq_{\ell,s}$ is a

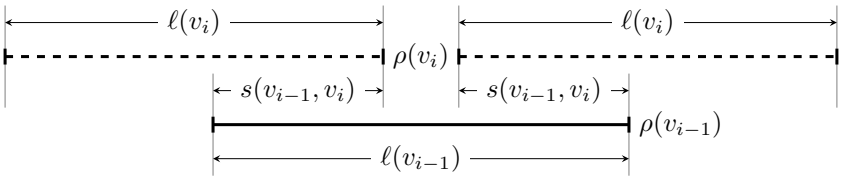

Figure 3: If $v_i \between_{\ell,s} v_{i-1}$, and if $\rho(v_{i-1})$ is already determined, there remain only the two dashed possibilities for $\rho(v_i)$. The algorithm in the proof of Lemma 4.1 is based on this observation.

partial order on the overlap components of G, and each overlap component has a unique smallest successor. If G is connected, $(\mathcal{C}, \leq_{\ell,s})$ is also connected; by removing reflexive and transitive edges, we obtain a rooted tree $T_{\ell,s}$, which we call the *overlap component tree* of G.

Theorem 4.2. *Given a graph $G = (V, E)$ and the length functions $\ell\colon V \to \mathbb{Z}^+$ and $s\colon E \to \mathbb{Z}^+$, it is possible to compute in $\mathcal{O}(n + m)$ time (resp., logspace) an (ℓ, s)-respecting interval representation ρ of G, or to detect that none exists. Moreover, ρ is unique up to isomorphism.*

Proof. We assume that G is connected, otherwise we consider its connected components separately and concatenate their representations afterwards.

The algorithm works as follows: As pre-processing steps, it computes the relations $R_{\ell,s}$, $\between_{\ell,s}$ and $\subseteq_{\ell,s}$, the connected components $G_1, \ldots, G_k$ of $G_{\ell,s}$, an (ℓ, s)-respecting interval representation ρ_i for each of them, and the overlap component tree $T_{\ell,s}$. The main part of the algorithm constructs an (ℓ, s)-respecting interval representation ρ of G by combining appropriately shifted copies of the representations of the overlap components. This is done in a depth-first traversal of the overlap component tree. Each overlap component G_i is assigned an offset o_i indicating the value of the smallest point in $\rho(G_i)$. The representation of the root component G_r is not shifted, i.e., $o_r = 1$. For each other overlap component G_i, compute the offset o_i as the sum $o_i = o_j + n_i + s_i$, where

- o_j is the offset of the parent G_j of G_i,
- n_i is the number of points in $\rho_j(G_j)$ that are to the left of the slot S of G_j containing G_i, and
- s_i is the sum of representation sizes of all previously visited siblings of G_i that are contained in the same slot S of G_j as G_i.

The latter are available from their interval representations. Figure 4 shows an example. For a vertex $v \in V$, let G_{i_v} be the overlap component that contains v. Then the algorithm checks whether $\rho(v) = \rho_{i_v}(v) + o_{i_v}$ is an (ℓ, s)-respecting interval representation of G (as described in the proof of Lemma 4.1) and outputs ρ if this is the case.

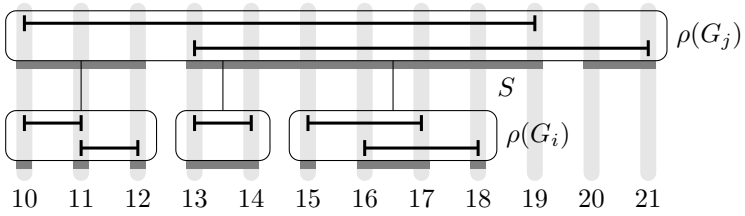

Figure 4: The offset o_i for the overlap component G_i is the sum of the offset $o_j = 10$ of its parent G_j, the number n_i of points in $\rho_j(G_j)$ left of the slot $S = [4, 10]$ of G_j that contains G_i (i.e., $n_i = 3$), and the sum s_i of representation sizes of previously handled siblings of G_i that are contained in the same slot S (i.e., $s_i = 2$), resulting in $o_i = 15$.

If G admits an (ℓ, s)-respecting interval representation, then this algorithm will find one: Each component has a unique representation up to reflection by Lemma 4.1, implying that they all have the same length;

and in every (ℓ, s)-respecting interval representation of G, each overlap component must be placed in the appropriate slot of its parent overlap component without intersecting the other overlap components placed in this slot. In the construction of the representation ρ, the only arbitrary choices are the precise placement of overlap components belonging to the same slot S of their parent component, the order of the connected components of G, and whether the representations of the individual overlap components are reflected. All these choices can be transformed into one another by isomorphisms of the resulting interval system, so ρ is unique up to isomorphism.

To finish the proof, we show that the algorithm can be implemented in linear time or logspace. The relations can be computed efficiently by Lemma 3.2. Connected components can be found in linear time using depth first search, and in logspace using Reingold's [Rei08] connectivity algorithm. The (ℓ, s)-respecting representations of the components of $G_{\ell,s}$ can be computed using Lemma 4.1. The construction of the overlap component tree $T_{\ell,s}$ can easily be implemented in logspace. To obtain it in linear time, compute $\leq_{\ell,s}$ by iterating over the edges of G, and remove reflexive and transitive arcs; see [HMR93, Proposition 3.6] for how the latter is possible in linear time. Computing the offsets for shifting is clearly possible in linear time. To compute them in logspace, the algorithm traverses the overlap component tree (see e.g. [Lin92] for how to do this in logspace). During this traversal, it only needs to store the offset o_i for the current overlap component G_i; this is sufficient because the offset $o_j = o_i - n_i - s_i$ of the parent component G_j of G_i can be recovered from o_i by recomputing the numbers n_i and s_i. □

5. Given intersection lengths

Let $G = (V, E)$ be a graph and let $s\colon E \to \mathbb{Z}^+$ specify the prescribed intersection lengths. In this section, we give a Turing reduction from the problem of finding a minimal s-respecting interval representation of G to the problem of finding an (ℓ, s)-respecting interval representation.

In particular, we show that the lengths of the intervals in a minimal s-respecting representation ρ are determined by G and s, and can be computed efficiently. Notice that if ρ is not required to be minimal we can always increase the length of some intervals by adding new points to an interval containing the largest (or smallest) point of $\rho(G)$ or by duplicating points that are contained in a single interval of $\rho(G)$.

Lemma 5.1. *Let $G = (V, E)$ be an interval graph with the length function $s\colon E \to \mathbb{Z}^+$, and let ρ be any minimal s-respecting interval representation of G. Then the interval lengths $\ell(v) = |\rho(v)|$ do not depend on the choice of ρ and can be computed from G and s in logspace; or in $\mathcal{O}(n + m)$ time, if the relation R_s is given as additional input.*

Recall that R_s can be computed in $\mathcal{O}(nm)$ time or in logspace by Lemma 3.2.

Proof. We first describe the algorithm. For each $v \in V$, it distinguishes these cases:

1. If $N(v) = \varnothing$, the algorithms sets $\ell(v) := 1$.
2. If $\exists w \in N(v) : v \subseteq_s w$, it sets $\ell(v) := s(v, w)$.
3. Else, if $\exists w_1, w_2 \in N(v)$ such that $v \between_s w_1 \between_s w_2 \between_s v$ and $s(w_1, w_2) < \min\{s(w_1, v), s(w_2, v)\}$ (i.e., w_1 and w_2 overlap each other, and overlap v from different sides, cf. Lemma 3.3), it sets $\ell(v) := s(w_1, v) + s(w_2, v) - s(w_1, w_2)$.
4. Otherwise, we consider the subgraph $G[N(v)]$ and define $\ell_v\colon N(v) \to \mathbb{Z}^+$ by $\ell_v(w) = s(w, v)$ for all $w \in N(v)$. Additionally, we define $s_v\colon \left(E \cap \binom{N(v)}{2}\right) \to \mathbb{Z}^+$ by $s_v(w_1, w_2) = \min\{s(w_1, v), s(w_2, v)\}$ if w_1 and w_2 overlap v from the same side (see Lemma 3.3), and $s_v(w_1, w_2) = s(w_1, w_2)$ otherwise. The algorithm computes an (ℓ_v, s_v)-respecting interval representation $\rho_v\colon N(v) \to \mathcal{I}_v$ of $G[N(v)]$, and sets $\ell(v) := |\bigcup_{I \in \mathcal{I}_v} I|$.

Next, we show that the computed function ℓ satisfies $\ell(v) = |\rho(v)|$ for each $v \in V$. For an isolated vertex v, as considered in Case 1, we have $|\rho(v)| = 1$ by minimality of ρ, so $\ell(v) = 1$ is correct. By Lemma 3.1(b) and the definitions of $\between_s$ and $\subseteq_s$, we have $u \between_s v \Leftrightarrow \rho(u) \between \rho(v)$ and $u \subseteq_s v \Leftrightarrow \rho(u) \subseteq \rho(v)$. In Case 2, this immediately implies $\ell(v) = s(v, w) = |\rho(v) \cap \rho(w)| = |\rho(v)|$.

In Case 3, the intervals $\rho(w_1)$ and $\rho(w_2)$ cover the interval $\rho(v)$, overlapping it from different sides (the latter is true by Lemma 3.3), so we have the situation depicted in Fig. 5. Thus, $\ell(v) = s(w_1, v)+s(w_2, v)-s(w_1, w_2) = |\rho(w_1) \cap \rho(v)| + |\rho(w_2) \cap \rho(v)| - |\rho(w_1) \cap \rho(w_2)| = |(\rho(w_1) \cup \rho(w_2)) \cap \rho(v)| = |\rho(v)|$.

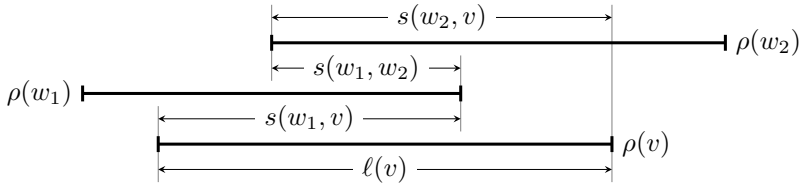

Figure 5: Proof of Lemma 5.1, Case 3: $\rho(w_1)$ and $\rho(w_2)$ cover $\rho(v)$, overlapping it from different sides.

In Case 4, the definitions of ℓ_v and s_v truncate the intervals of the vertices in $N(v)$ to include only their intersections with $\rho(v)$. We have $|\rho_v(u)| = |\rho(u)|$ for all $u \subseteq_s v$, and $|\rho_v(w)| = |\rho(w) \cap \rho(v)|$ for all $w \between_s v$. So truncating $\rho(G[N(v)])$ gives an (ℓ_v, s_v)-respecting model $\rho_v(G[N(v)])$ of $G[N(v)]$. By Theorem 4.2, this model is unique up to isomorphism; in particular, its length is uniquely determined, implying $|\rho(v)| \geq \ell(v)$. Indeed, both values are equal, for if $|\rho(v)| > \ell(v)$ then the part of $\rho(G)$ that is covered by $\rho(v)$ could be replaced by the shorter $\rho_v(G[N(v)])$, contradicting the minimality of ρ.

It is obvious that this algorithm can be implemented in logspace. To see that it is also possible in linear time, observe that in Case 3, Lemma 3.3 allows us to partition the $\between_s$-neighbors of v into two sets W_1 and W_2, where neighbors that overlap from the same side are in the same set, and that we can restrict the search by requiring $w_i \in W_i$ and $s(w_i, v) = \max\{s(w'_i, v) \mid w'_i \in W_i\}$.

For seeing that Case 4 can be implemented in linear time, observe that each vertex u of G can occur in at most three of the auxiliary graphs. Suppose to the contrary that there are vertices v_1, v_2, v_3, v_4 such that for each $i \in [1, 4]$, $u \in N(v_i)$ and Case 4 is reached for v_i. The latter implies that no $\rho(v_i) = [v_i^-, v_i^+]$ is contained in any other interval, and that none of them is covered by two overlapping intervals. Because Case 2 does not hold, there are no containments, so we can assume $v_1^- < v_2^- < v_3^- < v_4^-$ and $v_1^+ < v_2^+ < v_3^+ < v_4^+$. As Case 3 holds neither, it follows that $v_1^+ < v_3^-$ and $v_2^+ < v_4^-$. Now let $\rho(u) = [u^-, u^+]$. As u is a neighbor of all v_i, we know $u^- \leq v_1^+$ and $v_4^- \leq u^+$. But this implies that $\rho(u)$ either covers $\rho(v_2)$ alone or together with $\rho(v_1)$, contradicting the assumption that Case 4 is reached for v_2. We remark that Case 3 is subsumed by Case 4 and is only handled separately to obtain the linear time bound. □

The following is a consequence of Theorem 4.2 and Lemmas 3.2 and 5.1.

Corollary 5.2. *Given $G = (V, E)$ and $s\colon E \to \mathbb{Z}^+$, it is possible to compute in $\mathcal{O}(nm)$ time (resp., in logspace) a minimal s-respecting interval representation ρ of G, or to detect that none exists. Moreover, ρ is unique up to isomorphism.*

6. Interval graphs with a unique maxclique ordering

As mentioned in the introduction, deciding if a graph has an ℓ-respecting interval representation is NP-complete [PS97]. This changes however, if the input graph G is required to have a unique (up to reflection) interval ordering of its maximal cliques; such a graph is called *UCO* for unique clique order. On UCO graphs, even a more general problem becomes tractable: Along with G, a system D of inequalities of the form $x_i - x_j \leq c$ is given, where the variables refer to arbitrary endpoints of the intervals representing the vertices of G (strict inequalities are allowed, too). The decision version of this problem is called DCIG (short for *distance constrained interval graph*). It asks whether G has an interval representation satisfying all the inequalities given by D, where all intervals are closed and their endpoints may take arbitrary rational values. A more restricted problem is BIG (short for *bounded interval graph recognition*) which only allows the specification of lower and/or upper bounds on the lengths of the intervals corresponding to the vertices of G.

Pe'er and Shamir [PS97] show that when restricted to UCO graphs, DCIG and BIG are both linear-time equivalent to the problem NoNegCycle, i.e., checking whether a weighted digraph (V, E, w) with a weight function $w\colon E \to \mathbb{Z}$ does not have a negative cycle. We observe that the reductions given in [PS97] can be performed in logspace. Moreover, the constants occurring in the instances of DCIG and BIG and the weights occurring in the corresponding instances of NoNegCycle are polynomially related. Hence, as NoNegCycle with polynomially bounded weights is easily seen to be NL-complete, it follows that DCIG and BIG are both NL-complete on the class of UCO graphs when the constants of the inequalities are polynomially bounded.

As an intermediate step in their reductions, Pe'er and Shamir consider the problem DiffIneq, which asks whether a system of difference inequalities admits a solution over the rationals, i.e., the inequalities have the form $x_i - x_j \leq c_k$ or $x_i - x_j < c_k$, where the constants c_k are integral.

Lemma 6.1. *The disjunctive reduction from* DCIG *on UCO graphs to* DiffIneq *given by Pe'er and Shamir [PS97] can be implemented in logspace.*

Proof. We first describe the reduction. Let $G = (V, E)$ be the given UCO graph and let D be the distance constraints for the interval endpoints. The idea is to extend D with additional constraints to two systems A and A' of difference inequalities, so that any interval representation of G that satisfies D corresponds to a satisfying assignment for one of A and A', and vice versa.

Any interval ordering $\prec_G$ on the maximal cliques of G can also be viewed as the partial order on V that has $u \prec_G v$ if and only if $\prec_G$ places the interval of the maximal cliques containing u completely left of the interval of the maximal cliques containing v. For $v \in V$, let v^- and v^+ denote the variables for its left and right endpoint, respectively. Both A and A' contain all distance constraints in D. Additionally, for each pair of adjacent vertices u and v, both systems contain the constraints $u^- \leq v^+$ and $v^- \leq u^+$ to ensure that their intervals intersect. For $u \prec_G v$, the inequality $u^+ < v^-$ is added to A, and the inequality $v^+ < u^-$ is added to A'. By assumption $\prec_G$ is unique up to reversal, so the result is unique up to exchanging A and A'.

Clearly, any solution to A (or to A') also satisfies D and specifies the endpoints of an interval representation of G. Conversely, any interval representation of G that satisfies D corresponds to $\prec_G$ or $\prec_G^{-1}$, and thus satisfies one of A and A' [PS97, Lemma 3.1]. It remains to observe that $\prec_G$ can be computed in logspace using the interval graph representation algorithm of [KKLV11]; the rest of the reduction is easily possible in logspace. □

Fact 6.2. *The problems* DiffIneq *and* NoNegCycle *are equivalent under logspace many-one reductions.*

Proof. Pe'er and Shamir [PS97, Lemma 3.2] prove that the general DiffIneq problem reduces to its special case where all inequalities are weak (i.e., of the form $x_i - x_j \leq c_k$). This is achieved by choosing $\epsilon = \frac{1}{n}$ (where n is the number of variables), replacing each strict inequality $x - y < c$ by $x - y \leq c - \epsilon$, and scaling the constants by n to restore integrality. As this transformation can be implemented in logspace, we can assume that all inequalities are weak.

It is not hard to see that a NoNegCycle instance (V, E, w) is equivalent to the DiffIneq instance (X, A) with variables $X = V$ and difference inequalities $A = \{x_i - x_j \leq w(x_j, x_i) \mid (x_j, x_i) \in E\}$; see e.g. [AMO93, pp. 103 ff.]. Note that this transformation can be performed in logspace and that it can easily be reversed to obtain a logspace reduction in the opposite direction.

For the reader's convenience we include a self-contained proof of the equivalence $(X, A) \in$ DiffIneq $\Leftrightarrow$ $(V, E, w) \in$ NoNegCycle. For each walk $(v_1, \ldots, v_k)$ of weight $W = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$ in (V, E, w), any assignment $\sigma\colon X \to \mathbb{Q}$ that satisfies all difference inequalities in A also satisfies $\sigma(v_k) \leq \sigma(v_1) + W$; this can be shown by an easy induction on k. If (V, E, w) contains a negative cycle $(v_1, \ldots, v_{k-1}, v_1)$, this implies $\sigma(v_1) < \sigma(v_1)$, contradicting the existence of a valid assignment.

On the other hand, if (V, E, w) does not have a negative cycle, fix an ordering $v_1, \ldots, v_n$ of V such that whenever there is a directed path from v_j to v_i for $i < j$, then there is also a directed path from v_i to v_j. Such an ordering can be obtained by finding the strongly connected components, contracting each of them to a single vertex, taking a topological ordering of the resulting acyclic digraph, and substituting each representative vertex with the vertices of the strongly connected component it stands for. To construct a valid

assignment $\sigma\colon X \to \mathbb{Z}$, let $\sigma(v_1) = 0$, and for $j > 1$ let $\sigma(v_j) = \min\{\sigma(v_i) + w(v_i, v_j) \mid v_i \in N^-(v_j) \wedge i < j\}$. This immediately satisfies all difference inequalities in A that correspond to edges in E that go forward with respect to the chosen ordering of V, and also the others as there are no negative cycles. □

Fact 6.3. *The problem* NoNegCycle *with polynomially bounded weight functions is* NL*-complete.*

Proof. As NL is closed under complementation [Imm88, Sze88], it suffices to show that the problem NegCycle, which is the problem of deciding whether a given weighted digraph contains a negative cycle, is NL-complete. To check if a weighted digraph $G = (V, E, w)$ with a polynomially bounded weight function $w\colon E \to \mathbb{Z}$ has a negative cycle, non-deterministically walk through G (along edges) for at most $|V|$ steps, i.e., guess a vertex $v_0 \in V$, and proceed from v_i by guessing $v_{i+1} \in N^+(v_i)$. During the walk, store the start vertex v_0, the current vertex v_i, the number i of steps taken so far, and the accumulated weight of the traversed edges; this can be done in logarithmic space. Accept if the guessed walk returns to the start vertex and has negative weight, otherwise reject. If G contains a negative cycle $v_0, \dots, v_k, v_0$, the computation that successively visits the vertices of this cycle accepts. On the other hand, any accepting computation witnesses a negative closed walk, which must contain a negative cycle. Thus NegCycle is in NL.

To prove the hardness, we reduce from the NL-complete problem s-t-Con to decide if there is a directed path from s to t in a given digraph. An instance (V, E, s, t) of s-t-Con is reduced to the NegCycle instance (V, E', w), where the digraph $G' = (V, E')$ is obtained from $G = (V, E)$ by adding the arc (t, s) if it is not yet present, and the weight function $w\colon E' \to \mathbb{Z}$ is given by $w(t, s) = -n$ and $w(u, v) = 1$ for all other arcs $(u, v) \in E' \setminus \{(t, s)\}$. If there is a path from s to t in G, it has length (and thus weight) at most $n - 1$. Combining this path with the arc (t, s) results in a negative cycle in G'. Conversely, if there is a negative cycle in G', it must contain the arc (t, s), as this is the only one with negative weight. Cutting (t, s) out of that cycle results in a path from s to t in G. □

Finally we show that also the problem BIG is NL-hard on the class of UCO graphs.

Lemma 6.4. *The reduction from* DiffIneq *to the problem* BIG *on UCO graphs that is given in [PS97, Section 3.2] can be implemented in logspace.*

Proof. Again, we first describe the reduction. Let A be a system of difference inequalities on the variables $X = \{x_1, \dots, x_n\}$. As argued in the proof of Fact 6.2, it can be assumed that all inequalities are weak, so write $A = \{x_{j_i} - x_{k_i} \le c_i \mid i = 1, \dots, m\}$. Fix $C > 1 + \sum_{i=1}^{m} |c_i|$ and let $c'_i = c_i + (j_i - k_i)C$.

The reduction maps A to the DCIG instance (G, D), where $G = (V, E)$ is the intersection graph of the interval system $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2 \cup \mathcal{I}_3$ (i.e., $V = \mathcal{I}$ and $\{u, v\} \in E \Leftrightarrow u \cap v \neq \varnothing$), defined by

$$\begin{aligned}
&\mathcal{I}_1 = \{a_i \mid i = 0, \dots, n\} && \text{where } a_i = [i, i+1] \\
&\mathcal{I}_2 = \{b_{i/2} \mid i = 0, \dots, 2n+1\} && \text{where } b_{i/2} = [i/2, i/2] \\
&\mathcal{I}_3 = \{w_i \mid i = 0, \dots, m\} && \text{where } w_i = [k_i, j_i] \text{ if } k_i < j_i \text{ and } w_i = [j_i + \tfrac{1}{4}, k_i - \tfrac{1}{4}] \text{ otherwise.}
\end{aligned}$$

For integral i, the constraints $b_i^+ - b_i^- = 0$ and $b_{i+1/2}^+ - b_{i+1/2}^- = 1$ are included in D. For $j_i > k_i$, the constraint $w_i^+ - w_i^- \le c'_i$, and for $j_i < k_i$, the constraint $w_i^+ - w_i^- \ge -c'_i - \epsilon$ with $\epsilon < 1/n$ is added to D.

All these constructions are easily possible in logspace. □

7. Conclusion

We have shown how to compute (ℓ, s)- and s-respecting interval representations, giving a linear-time algorithm for the former, an $\mathcal{O}(nm)$ time algorithm for the latter, and logspace algorithms for both. We remark that deciding whether a graph admits an (ℓ, s)- or s-respecting interval representation is L-complete: In the reduction proving that recognizing interval graphs is L-hard [KKLV11, Theorem 7.7], all generated *yes*-instances are paths; so these graphs have (ℓ, s)- and s-respecting interval representations if we let $\ell(v) = 2$ and $s(e) = 1$ for all vertices u and edges e.

We also have shown that (ℓ, s)- and minimal s-respecting interval representations are unique up to isomorphism. This implies that any algorithm that computes canonical interval representations of interval hypergraphs can be used to obtain *canonical* (ℓ, s)- and s-respecting interval representations. The algorithm given in [KKLV11, Theorem 4.6] solves this problem in logspace, and it can also be solved in linear time using the PQ-tree algorithms of Booth and Lueker [BL76].

Open questions

The bottleneck in our $\mathcal{O}(nm)$ time algorithm for computing s-respecting interval representations is the enumeration of R_s (see Lemma 3.2). Can this also be implemented in $\mathcal{O}(n+m)$ or at least $\mathcal{O}(n^2)$ time?

Does the complexity of computing s-respecting interval representations increase, if the intersection lengths are restricted only for some vertices? Our techniques are not directly applicable in this case, as the algorithm of Lemma 5.1 relies on the uniqueness of the representation, which is not necessarily preserved in the modified scenario.

Acknowledgements. We thank Oleg Verbitsky for interesting discussions about these results and for bringing the work of Fulkerson and Gross [FG65] to our attention. We also thank the anonymous referees for their valuable comments.

References

[All83] James F. Allen, *Maintaining knowledge about temporal intervals*, Communications of the ACM **26** (1983), no. 11, 832–843.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin, *Network flows*, Prentice Hall, Englewood Cliffs, N.J., 1993.

[BKO13] Martin Balko, Pavel Klavík, and Yota Otachi, *Bounded representations of interval and proper interval graphs*, Proceedings of 24th International Symposium on Algorithms and Computation (ISAAC), LNCS, no. 8283, Springer, 2013, pp. 535–546.

[BL76] Kellogg S. Booth and George S. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, Journal of Computer and System Sciences **13** (1976), no. 3, 335–379.

[BR13] Thomas Bläsius and Ignaz Rutter, *Simultaneous PQ-ordering with applications to constrained embedding problems*, Proceedings of 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2013, pp. 1030–1043.

[COS09] Derek G. Corneil, Stephan Olariu, and Lorna Stewart, *The LBFS structure and recognition of interval graphs*, SIAM Journal on Discrete Mathematics **23** (2009), no. 4, 1905–1953.

[DHH96] Xiaotie Deng, Pavol Hell, and Jing Huang, *Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs*, SIAM Journal on Computing **25** (1996), no. 2, 390–403.

[FG65] Delbert Ray Fulkerson and Oliver Gross, *Incidence matrices and interval graphs*, Pacific Journal of Mathematics **15** (1965), no. 3, 835–855.

[Gol04] Martin Charles Golumbic, *Algorithmic graph theory and perfect graphs*, 2nd ed., Annals of Discrete Mathematics, no. 57, Elsevier, Amsterdam, 2004.

[HAB02] William Hesse, Eric Allender, and David A. Mix Barrington, *Uniform constant-depth threshold circuits for division and iterated multiplication*, Journal of Computer and System Sciences **65** (2002), no. 4, 695–716.

[HM99] Wen-Lian Hsu and Tze-Heng Ma, *Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs*, SIAM Journal on Computing **28** (1999), no. 3, 1004–1020.

[HMR93] Michel Habib, Michel Morvan, and Jean-Xavier Rampon, *On the calculation of transitive reduction—closure of orders*, Discrete Mathematics **111** (1993), no. 1–3, 289–303.

[HSS01] Pavol Hell, Ron Shamir, and Roded Sharan, *A fully dynamic algorithm for recognizing and representing proper interval graphs*, SIAM Journal on Computing **31** (2001), no. 1, 289–305.

[Imm88] Neil Immerman, *Nondeterministic space is closed under complementation*, SIAM Journal on Computing **17** (1988), no. 5, 935–938.

[JL10] Krishnam Raju Jampani and Anna Lubiw, *Simultaneous interval graphs*, Proceedings of 21st International Symposium on Algorithms and Computation (ISAAC), LNCS, no. 6506, Springer, 2010, pp. 206–217.

[KJJ03] Andrei Krokhin, Peter Jeavons, and Peter Jonsson, *Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra*, Journal of the ACM **50** (2003), no. 5, 591–640.

[KJJ04] ———, *Constraint satisfaction problems on intervals and lengths*, SIAM Journal on Discrete Mathematics **17** (2004), no. 3, 453–477.

[KKLV11] Johannes Köbler, Sebastian Kuhnert, Bastian Laubner, and Oleg Verbitsky, *Interval graphs: Canonical representations in logspace*, SIAM Journal on Computing **40** (2011), no. 5, 1292–1315.

[KKO+14] Pavel Klavík, Jan Kratochvíl, Yota Otachi, Ignaz Rutter, Toshiki Saitoh, Maria Saumell, and Tomáš Vyskočil, *Extending partial representations of proper and unit interval graphs*, Proceedings of 14th Scandinavian Workshop on Algorithm Theory (SWAT), LNCS, no. 8503, Springer, 2014, pp. 253–264.

[KKV11] Pavel Klavík, Jan Kratochvíl, and Tomáš Vyskočil, *Extending partial representations of interval graphs*, Proceedings of 8th Annual Conference on Theory and Applications of Models of Computation (TAMC) (Berlin), Springer, 2011, pp. 276–285.

[KKV13] Johannes Köbler, Sebastian Kuhnert, and Oleg Verbitsky, *Helly circular-arc graph isomorphism is in logspace*, Proceedings of 38th International Symposium Mathematical Foundations of Computer Science (MFCS) (Berlin), Springer, 2013, pp. 631–642.

[Kle96] Philip N. Klein, *Efficient parallel algorithms for chordal graphs*, SIAM Journal on Computing **25** (1996), no. 4, 797–827.

[Lin92] Steven Lindell, *A logspace algorithm for tree canonization*, Proceedings of 24th Annual ACM Symposium on Theory of Computing (STOC) (New York), ACM, 1992, pp. 400–404.

[McC03] Ross M. McConnell, *Linear-time recognition of circular-arc graphs*, Algorithmica **37** (2003), no. 2, 93–147.

[PS97] Itsik Pe'er and Ron Shamir, *Realizing interval graphs with size and distance constraints*, SIAM Journal on Discrete Mathematics **10** (1997), no. 4, 662–687.

[Rei08] Omer Reingold, *Undirected connectivity in log-space*, Journal of the ACM **55** (2008), no. 4, 17:1–17:24.

[Skr84] Dale Skrien, *Chronological orderings of interval graphs*, Discrete Applied Mathematics **8** (1984), no. 1, 69–83.

[Sze88] Róbert Szelepcsényi, *The method of forced enumeration for nondeterministic automata*, Acta Informatica **26** (1988), no. 3, 279–284.

[Yam07] Naoki Yamamoto, *Weighted interval graphs and their representations*, Master's thesis, Tokyo Inst. of Technology, 2007.