Interval graph representation with given interval and intersection lengths

Johannes Köbler[1], Sebastian Kuhnert[1★], and Osamu Watanabe[2]

[1] Humboldt-Universität zu Berlin, Inst. für Informatik
[2] Tokyo Institute of Technology, Dept. of Mathematical and Computing Sciences

Abstract. We consider the problem of finding interval representations of graphs 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]. We give both a linear-time and a logspace algorithm for the case when both are given, and both an $\mathcal{O}(n \cdot m)$ time and a logspace algorithm when only the latter are given. We also show that the resulting interval systems are unique up to isomorphism.

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 maxcliques. For such graphs, their algorithm computes an interval representation that respects a given set of distance inequalities between the interval endpoints (if it exists). We observe that deciding if such a representation exists is NL-complete.

1 Introduction

Algorithmic aspects of interval graphs have been the subject of ongoing research for several decades, stimulated by their numerous applications; see e.g. [Gol04].

The *interval representation problem* asks, given a graph G, if G is an interval graph, and if so, to compute an interval representation for it. Booth and Lueker [BL76] solve this problem in linear time, introducing the widely used concept of PQ-trees to efficiently encode all possible orderings of the maximal cliques. Hsu and Ma [HM99] give a simpler linear-time algorithm that relies on modular decomposition instead. Corneil, Olariu, and Stewart [COS09] show a further simplification, avoiding ordering the maximal cliques, by using lexicographic breadth first search. Klein gave a parallel AC^2 algorithm [Kle96]. Köbler et al. [KKLV11] show that the interval representation problem is complete for logspace.

In this paper, we consider the problems whether a graph with a weight function ℓ on its vertices and/or a weight function s on its edges admits ℓ-respecting interval representations (where for each vertex v, its weight $\ell(v)$ prescribes the length of its interval), s-respecting interval representations (where for each

★ Supported by DFG grant KO 1053/7–1.

edge $\{u, v\}$, its weight $s(\{u, v\})$ prescribes the length of the intersection of the intervals of u and v), and (ℓ, s)-respecting interval representations (which are required to fulfill both these restrictions). Pe'er and Shamir showed that it is NP-complete to decide if a graph G admits an ℓ-respecting interval representation [PS97]. The problem of finding s-respecting interval representations was introduced in [Yam07].

Our results. We show how to construct (ℓ, s)-respecting interval representations in linear time or alternatively in logspace, and s-respecting interval representations in $\mathcal{O}(n \cdot m)$ time or alternatively in logspace. Since computing ℓ-respecting interval representations is NP-hard, our result illustrates that the information on interval intersections is quite helpful.

The first step towards our algorithms is to show that all interval representations of the appropriate type have the same inclusion and overlap relationships, and that these relations can be computed efficiently when G, ℓ (and s) are given as input. This is described in Section 3.

To obtain our results on (ℓ, s)-respecting interval representations (which are in Section 4), we first focus on graphs with overlap-connected representations. We show that these representations are unique up to reflection and can be computed efficiently (if they exist). 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.

In Section 5 we show how to compute s-respecting interval representations efficiently. To obtain our result, 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 already determine the interval lengths (up to insertion of points that are only present in a single interval). The resulting s-respecting interval representation is minimal, i.e., it contains no superfluous points. We also show that all minimal s-respecting interval representations are isomorphic.

In Section 6, we consider the variant of the interval representation problem for which Pe'er and Shamir gave a polynomial time algorithm [PS97]: On the one hand, the input graph is required to have a unique interval ordering of its inclusion-maximal cliques (up to reflection); on the other hand, general lower and upper bounds on distances between interval endpoints are allowed. We observe that this variant is in fact NL-complete. That is, it is unlikely that this generalization of the ℓ-respecting interval representation problem is solvable in deterministic logspace even for the restricted input 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 $\|A\|$.

For a graph $G = (V, E)$, the set of neighbors of a vertex $v \in V$ is denoted by $N(v)$. G is an *interval graph* if there is a system $\mathcal{I}$ of nonempty intervals over $\mathbb{N}$ (we allow $\mathcal{I}$ to be a multiset) and a bijection $\rho\colon V \to \mathcal{I}$ such 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 $\rho(G)$.

We write $[l, r]$ to denote the interval $\{i \in \mathbb{N} \mid l \leq i \leq r\}$. With the *length* of an interval we denote the number of points in it.[3] For an interval model $\mathcal{I}$ we always suppose $\bigcup_{I \in \mathcal{I}} I = [1, k]$ for some k, i.e., we disallow shifting and gaps between connected components. $\mathcal{I}$ can be regarded as hypergraph with nodes $[1, k]$ and hyperedges $\mathcal{I}$. Two interval models $\mathcal{I}$ 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 graph G isomorphic if $\rho_1(G)$ and $\rho_2(G)$ are isomorphic. The *slots* of $\mathcal{I}$ are the equivalence classes on $[1, k]$ w.r.t. containment in the intervals in $\mathcal{I}$. That is, two vertices are in the same slot, if all hyperedges contain either both or none of them.

For functions $\ell\colon V \to \mathbb{N}$ and $s\colon E \to \mathbb{N}$, an interval representation $\rho\colon V \to \mathcal{I}$ of $G = (V, E)$ 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 $\left\|\bigcup_{v \in V} \rho'(v)\right\| < \left\|\bigcup_{v \in V} \rho(v)\right\|$.

As usual, L is the class of all languages decidable by Turing machines with a read-only input tape using only $\mathcal{O}(\log N)$ space on the working tapes, where N is the input size. FL is the class of all functions computable by such machines that additionally have a write-only output tape. Note that FL is closed under composition: To compute $f(g(x))$ 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.

This closure property can also be used 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 is possible in logspace as iterative integer multiplication is in $\mathsf{DLOGTIME}$-uniform TC^0 [HAB02].

3 Deriving structural information

Let $G = (V, E)$ be a graph, let $n = \|V\|$ and $m = \|E\|$, and let $\ell\colon V \to \mathbb{N}$ and $s\colon E \to \mathbb{N}$ specify the desired interval and intersection lengths. For convenience, we

[3] This does not coincide with the usual notion of length $r - l$. However, if we use the real interval $(l - 0.5, r + 0.5)$, then both measures coincide.

write $s(u, v)$ instead of $s(\{u, v\})$ for $\{u, v\} \in E$; for $\{u, v\} \notin E$ we let $s(u, v) = 0$. Using this convention, we define two relations $R_{\ell,s}, R_s \subseteq V^2$:

$$\begin{aligned}(u, v) \in R_{\ell,s} &\Leftrightarrow \{u, v\} \in E \wedge \ell(u) > s(u, v)\\ (u, v) \in R_s &\Leftrightarrow \{u, v\} \in E \wedge \exists w \in V \setminus \{u, v\} : s(w, u) > \min\{s(w, v), s(u, v)\}\end{aligned}$$

By the following lemma, these relations characterize a structural property that *all* (ℓ, s)-respecting (resp., minimal s-respecting) interval representations of G have in common.

Lemma 1.
(a) *Let $\rho\colon V \to \mathcal{I}$ be any (ℓ, s)-respecting interval representation of G, and let $\{u, v\} \in E$. Then $\rho(u) \setminus \rho(v) \neq \varnothing$ if and only if $(u, v) \in R_{\ell,s}$.*
(b) *Let $\rho\colon V \to \mathcal{I}$ be any minimal s-respecting interval representation of G, and let $\{u, v\} \in E$. Then $\rho(u) \setminus \rho(v) \neq \varnothing$ if and only if $(u, v) \in R_s$.*

Proof. Part (a) follows directly from the definitions.

We now show part (b). By definition, $(u, v) \in R_s$ means that there is a $w \in V$ such that $s(w, u) > s(w, v)$ or $s(w, u) > s(u, v)$. 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 direction, consider a point $p \in \rho(u) \setminus \rho(v)$. By minimality of ρ, there is a vertex $w \in V \setminus \{u\}$ with $p \in \rho(w)$. Note that $w \neq v$ by choice of p. If $\rho(w) \supset \rho(u) \cap \rho(v)$, it follows that $s(w, u) > s(u, v)$. Otherwise $\rho(w) \cap \rho(u) \supsetneq \rho(w) \cap \rho(v)$ and thus $s(w, u) > s(w, v)$. □

Lemma 2. *$R_{\ell,s}$ and R_s can be enumerated in time $\mathcal{O}(m)$ and $\mathcal{O}(n \cdot m)$, respectively, and both can be enumerated in logspace.*

Proof. The logspace part is obvious. To enumerate $R_{\ell,s}$ in linear 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, 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 1, these relations describe the situation in any appropriate representation of G, e.g. 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.

Lemma 3. *Let $\rho\colon V \to \mathcal{I}$ 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. 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 $\|\rho(w_2) \cap \rho(v)\|$. □

4 Given interval and intersection lengths

Let $G = (V, E)$ be a graph, and let $\ell\colon V \to \mathbb{N}$ and $s\colon E \to \mathbb{N}$ specify the desired 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 $E_{\ell,s} = \{\{u, v\} \in E \mid u \between_{\ell,s} v\}$ 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. *Given $G = (V, E)$, ℓ and s, such that $G_{\ell,s}$ is connected, it is possible in linear time (resp., in logspace) to compute an (ℓ, s)-respecting interval representation $\rho\colon V \to \mathcal{I}$ of G, or to detect that none exists. Moreover, if existent, ρ is unique up to reflection.*

Proof. 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 universal exploration sequences [Rei08]. The following algorithm computes an interval representation $\rho\colon V \to \mathcal{I}$ of G by moving along this walk (which we assume has been computed in a pre-processing step). It computes an interval I_i for v_i at each step and outputs $\rho(v_i) = I_i$, if there is no $j < i$ with $v_j = v_i$. Define $I_1 = [1, \ell(v_1)]$ and $I_2 = [\ell(v_1)-s(v_1, v_2)+1, \ell(v_1)-s(v_1, v_2)+\ell(v_2)]$. Note that after I_1 has been placed, there are only two possibilities for I_2 that respect (ℓ, s); see Fig. 1 for an illustration. After that, all further intervals are completely determined because of Lemma 3, and can be computed from the walk, ℓ and s, remembering only the two previous intervals.

In a post-processing step, check that ρ is (ℓ, s)-respecting. Additionally, shift the resulting intervals such that 1 becomes the smallest point.

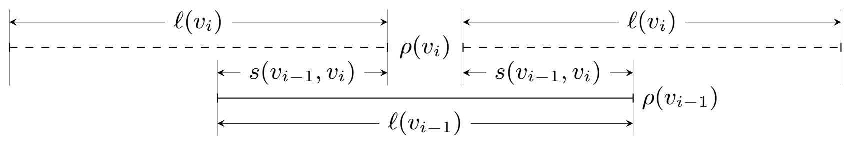

Fig. 1. Proof of Lemma 4: If $v_i \between_{\ell,s} v_{i-1}$, and if $\rho(v_{i-1})$ is already determined, there remain only the two dashed possibilities for $\rho(v_i)$.

The uniqueness up to reflection follows from the fact that the only arbitrary decision (except shifting) was to place $\rho(v_2)$ right of $\rho(v_1)$. $\square$

The next step is to generalize Lemma 4 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 compute an (ℓ, s)-respecting interval representation for $G \setminus \{v'\}$ and extend it by $v' \mapsto \rho(v)$ 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 $v \subseteq_{\ell,s} u$. 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 $\rho(G_j)$, because otherwise there would be an overlap-path from $\rho(G_i)$ to $\rho(G_j)$. Thus $\leq_{\ell,s}$ is a partial order on the overlap components of G. 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 5. *Given $G = (V, E)$, ℓ and s, it is possible in linear time (resp., logspace) to compute an (ℓ, s)-respecting interval representation $\rho\colon V \to \mathcal{I}$ of G, or to detect that none exists. Moreover, ρ is unique up to isomorphism.*

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

The algorithm works as follows: As pre-processing steps, compute the connected components $G_1, \ldots, G_k$ of $G_{\ell,s}$, an (ℓ, s)-respecting interval representation 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 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. The representation of the root component is not shifted. The representations of the other components are shifted to the appropriate slot of their parent component; if several child components are contained in the same slot, they are placed beside each other in the order in which they are encountered. It remains to check that the result is indeed an (ℓ, s)-respecting interval representation of G.

If G admits an (ℓ, s)-respecting interval representation, then this algorithm will find it: The representations of the components are unique up to reflection by Lemma 4, implying that they have the same length in all representations; 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. In the construction of the representation, the only arbitrary choices are the precise placement of overlap components within their containing slot, 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. Connected components can be found in linear time using depth first search, and in logspace using Reingold's connectivity algorithm [Rei08]. The

(ℓ, s)-respecting representations of the components of $G_{\ell,s}$ can be computed using Lemma 4. 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, and also in logspace if during the tree traversal (see e.g. [Lin92] for how to do this in logspace) a current shift-offset is maintained. □

5 Given intersection lengths

Let $G = (V, E)$ be a graph and let $s\colon E \to \mathcal{I}$ prescribe the desired intersection lengths. In this section, we reduce finding a minimal s-respecting interval representation of G to finding (ℓ, s)-respecting interval representations. In particular, we show that the lengths of the intervals in a minimal s-respecting representation are determined by G and s, and can be computed efficiently.

Note that we need minimality here, in contrast to the case of (ℓ, s)-respecting representations. The reason is that adding a point to an interval of an (ℓ, s)-respecting representation always destroys this property, while in an s-respecting representation, we can always duplicate points that are contained in a single interval.

Lemma 6. *Let $G = (V, E)$ be an interval graph with length function $s\colon E \to \mathbb{N}$, and let $\rho\colon V \to \mathcal{I}$ be an arbitrary 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 R_s is given as additional input.*

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

1. If $N(v) = \varnothing$, set $\ell(v) := 1$.
2. If $\exists w \in N(v) : v \subseteq_s w$, then set $\ell(v) := s(v, w)$.
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 $s(w_1, w_2) < \min\{s(w_1, v), s(w_2, v)\}$, then set $\ell(v) := s(w_1, v) + s(w_2, v) - s(w_1, w_2)$.
4. Otherwise, consider the subgraph $G[N(v)]$ and define $\ell_v\colon N(v) \to \mathbb{N}$ by $\ell_v(w) = s(w, v)$ for all $w \in N(v)$. Additionally, define $s_v\colon \left(E \cap \binom{N(v)}{2}\right) \to \mathbb{N}$ 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, and $s_v(w_1, w_2) = s(w_1, w_2)$ otherwise. Compute an (ℓ_v, s_v)-respecting interval representation $\rho_v\colon N(v) \to \mathcal{I}_v$ of $G[N(v)]$, and set $\ell(v) := \left\|\bigcup_{I \in \mathcal{I}_v} I\right\|$.

Next, we show that the computed 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 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, $\rho(w_1)$ and $\rho(w_2)$ cover $\rho(v)$, overlapping it from different sides (the latter is true by Lemma 3), so we have the situation depicted in Fig. 2.

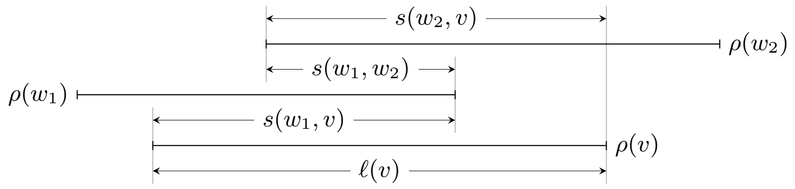

Fig. 2. Proof of Lemma 6, case 3: $\rho(w_1)$ and $\rho(w_2)$ cover $\rho(v)$, overlapping it from different sides.

Thus, $\ell(v) = s(w_1, v) + s(w_2, v) - s(w_1, w_2) = \|\rho(w_1) \cap \rho(v)\| + \|\rho(w_2) \cap \rho(v)\| - \|\rho(w_1) \cap \rho(w_2)\| = \|(\rho(w_1) \cup \rho(w_2)) \cap \rho(v)\| = \|\rho(v)\|$.

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 5, this model is unique up to isomorphism; in particular, its length is uniquely determined, implying $\|\rho(v)\| \geq \ell(v)$. By minimality of ρ, both values are equal.

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 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 require $w_1 \in W_1$ and $w_2 \in W_2$. For the linear-time implementation of case 4, 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 that case 4 is reached for v_2. □

The following is a consequence of Theorem 5 and Lemmas 2 and 6.

Corollary 7. *Given $G = (V, E)$ and s, it is possible in $\mathcal{O}(n \cdot m)$ time (resp., in logspace) to compute a minimal s-respecting interval representation $\rho\colon V \to \mathcal{I}$ of G, or to detect that none exists. Moreover, ρ is unique up to isomorphism.*

6 Interval graphs with unique maxclique ordering

As mentioned before, deciding if a graph has an ℓ-respecting interval representation is NP-complete [PS97]. However, if the input graph G is required to have a unique interval ordering of its inclusion-maximal cliques (up to reflection), even the more general problem DCIG (short for *distance constrained interval graph*) becomes tractable: Additionally to G, a system of difference inequalities of the form $x_i - x_j \geq c$ is given, where the variables are the left and right endpoints of the intervals (strict inequalities are allowed, too). The problem is to decide if G has an interval model that satisfies these inequalities. Pe'er and Shamir show that DCIG is linear-time equivalent to the problem NegCycle, i.e., deciding if a digraph has a negative cycle [PS97]. Based on the following facts, we observe that this problem is NL-complete.

Fact 8 NegCycle *is* NL*-complete.*

Proof. The problem is in NL, because one can check if a nondeterministically chosen path is a negative cycle, storing only the first vertex, the number of steps taken so far and the accumulated weight. 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: Let all arcs have weight 1, except (t, s), which is introduced if not yet present, and assigned the weight $-n$. □

Fact 9 *The linear-time reductions between* NegCycle *and* DCIG *for interval graphs with unique maxclique ordering can be implemented in logspace.*

Proof idea. For most steps of the reductions in [PS97] this is obvious, only computing the unique maxclique ordering requires the algorithm from [KKLV11]. □

We remark that the reduction from NegCycle to DCIG generates only lower and upper bounds on interval lengths, so NL-hardness holds for this special case, too.

Conclusion

We have shown how to compute (ℓ, s)- and s-respecting interval representations, giving a linear-time algorithm for the former, an $\mathcal{O}(n \cdot m)$ 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 in logspace, and it can also be done in linear time using the PQ-tree algorithms of [BL76].

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

Does the complexity of computing (ℓ, s)- and s-respecting interval representations increase, when the interval and intersection lengths are restricted only for some vertices? Our techniques are not directly applicable in this case, as the algorithm of Lemma 4 relies on the uniqueness of the representation, which is not necessarily preserved in the modified scenario.

Acknowledgement. We thank Oleg Verbitsky for interesting discussions about these results.

References

[BL76] Kellogg S. Booth and George S. Lueker. 'Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms'. In: *J. Comput. Syst. Sci.* 13.3 (1976), pp. 335–379.

[COS09] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. 'The LBFS Structure and Recognition of Interval Graphs'. In: *SIAM J. Discr. Math.* 23.4 (2009), pp. 1905–1953.

[Gol04] Martin Charles Golumbic. Algorithmic graph theory and perfect graphs. 2nd ed. *Annals of Discrete Mathematics* 57. Amsterdam: Elsevier, 2004.

[HAB02] William Hesse, Eric Allender, and David A. Mix Barrington. 'Uniform constant-depth threshold circuits for division and iterated multiplication'. In: *J. Comput. Syst. Sci.* 65.4 (2002), pp. 695–716.

[HM99] Wen-Lian Hsu and Tze-Heng Ma. 'Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs'. In: *SIAM J. Comput.* 28.3 (1999), pp. 1004–1020.

[HMR93] Michel Habib, Michel Morvan, and Jean-Xavier Rampon. 'On the calculation of transitive reduction—closure of orders'. In: *Discrete Math.* 111.1–3 (1993), pp. 289–303.

[KKLV11] Johannes Köbler, Sebastian Kuhnert, Bastian Laubner, and Oleg Verbitsky. 'Interval graphs: Canonical representations in logspace'. In: *SIAM J. Comput.* 40.5 (2011), pp. 1292–1315.

[Kle96] Philip N. Klein. 'Efficient parallel algorithms for chordal graphs'. In: *SIAM J. Comput.* 25.4 (1996), pp. 797–827.

[Lin92] Steven Lindell. 'A logspace algorithm for tree canonization. extended abstract'. In: *Proc. 24th STOC.* 1992, pp. 400–404.

[PS97] Itsik Pe'er and Ron Shamir. 'Realizing interval graphs with size and distance constraints'. In: *SIAM J. Discr. Math.* 10.4 (1997), pp. 662–687.

[Rei08] Omer Reingold. 'Undirected connectivity in log-space'. In: *J. ACM* 55.4 (2008), 17:1–17:24.

[Yam07] Naoki Yamamoto. 'Weighted interval graphs and their representations'. (In Japanese.) Master's Thesis. Tokyo Inst. of Technology, 2007.