Interval Graphs with Side (and Size) Constraints

Itsik Pe'er and Ron Shamir*

Department of Computer Science,
Sackler faculty of Exact Sciences,
Tel Aviv University, Tel-Aviv, 69978 Israel
email: {izik,shamir}@math.tau.ac.il.

Abstract. We study problems of determining whether a given interval graph has a realization which satisfies additional given constraints. Such problems occur frequently in applications where entities are modeled as intervals along a line (events along a time line, DNA segments along a chromosome, etc.). When the additional information is order constraints on pairs of disjoint intervals, we give a linear time algorithm. Extant algorithms for this problem (known also as *seriation with side constraints*) required quadratic time. When the constraints are bounds on distances between endpoints, and the graph admits a unique clique order, we show that the problem is polynomial. However, we show that even when the lengths of all intervals are precisely predetermined, the problem is NP-complete. We also study unit interval satisfiability problems, which are concerned with the realizability of a set of unit intervals along a line, subject to precedence and intersection constraints. For all possible restrictions on the types of constraints, we either give polynomial algorithms or prove their NP-completeness.

1 Introduction

A graph $G(V, E)$ is an *interval graph* if one can assign to each vertex v an interval I_v on the real line, so that two intervals have a non-empty intersection if and only if their vertices are adjacent. The set of intervals $\{I_v\}_{v \in V}$ is then called a *realization* of G. Interval graphs have been intensively studied, due to their central role in many applications (cf. [32, 16, 10]). They arise in numerous practical problems which require the construction of a time line where each particular event or phenomenon corresponds to an interval representing its duration. Among the applications are planning [3], scheduling [19, 30], archaeology [24], and circuit design [33]. There are also non-temporal applications in genetics [5] and behavioral psychology [8]. In the Human Genome Project, a central problem which bears directly on interval graphs is the physical mapping of DNA [7, 23]: It calls for the reconstruction of a map (a realization) for a collection of DNA segments, based on partial information on the relations between pairs of segments.

* Research supported in part by a grant from the Ministry of Science and the Arts, Israel.

The question whether a given graph has an interval realization (interval graph recognition) can be answered in linear time [6, 26, 20]. Surprisingly little is known about realization problems when the input contains also additional constraints on the realization. The problems which we study here are concerned with the realizability of a set of intervals along a line, subject to various types of constraints. The constraints may be order constraints, distances between interval endpoints or sizes of the intervals. Such problems arise naturally in many of the applications mentioned above, and we shall mention additional motivation for specific problems.

1.1 Interval Graphs with Side Order Constraints

When two intervals x, y on the real line intersect, we denote their relation by $x \cap y$ (read 'x intersects y'). If x is completely to the left of y we write $x \prec y$ or, equivalently, $y \succ x$. Suppose we are given a graph $G(V, E)$ and a set C of ordered pairs from $\overline{E}$. Is there an interval realization of G which satisfies $I_x \prec I_y$ whenever $(x, y) \in C$? This problem contains as special cases the interval graph and the interval order recognition problems. The problem (called also *seriation with side constraints*) was originally motivated by problems in archaeology and in consecutive retrieval with access priorities [25]. It arises also in temporal reasoning [17], and in physical mapping, when order information on some pairs of segments is available (see, e.g., [27]).

Golumbic and Shamir [17] give a simple $O(n^3)$ algorithm for the problem on a graph with n vertices, and Korte and Möhring [25] give an $O(n^2)$ algorithm using MPQ-trees. We give here a new linear-time algorithm of complexity $O(n + |E| + |C|)$. This improvement is substantial when the interval graph is sparse. Such sparse graphs are typical to many applications, including physical mapping (see, e.g., [22]). Our algorithm builds on a new interval graph recognition algorithm by Hsu and Ma [20].

1.2 Interval Graphs with Difference Constraints

In this problem the input is an interval graph and a set of difference constraints on the endpoints of the intervals. A difference constraint has the form $x - y < c$ or $x - y \leq c$. The goal is to determine if there is a realization in which the endpoints of the intervals satisfy all the constraints. Note that this problem contains as special cases the previous one, and the problem in which every constraint involves two endpoints of the same interval, which we shall discuss later. Fishburn and Graham[11] studied a special case of the problem in which the lengths of all intervals are bounded by *the same* pair of integers $p \leq |I_v| \leq q$. Their characterization implicitly gives a $n^{O(pq)}$ time algorithm for the realization problem. Isaak [21] gave a polynomial algorithm for the problem when the bounds are integers, the input is an interval order, and the realization must be discrete, i.e., all endpoints must be integers. He also posed the more general question which we study here.

We prove here that the problem is polynomial when the interval graph has a unique clique order, up to complete reversal (we call such graphs *UCO graphs*). This class of graphs properly contains the prime interval graphs, and is recognizable in linear time. We also show that the problem of solving a system of difference constraints is *linearly equivalent* to solving the realization problem on UCO interval graphs with difference constraints. On the other hand, the problem is NP-complete in general, as will follow from the next result.

1.3 Interval Graphs with Exact Given Sizes

Suppose we are given an interval graph and for each interval its exact length in the realization is prescribed. How hard is it to decide if a realization exists? This problem arises naturally when both intersection and size data is given. See [18] for an example of a physical mapping experiment in which such information is generated. If all lengths are the same then the graph is a unit-interval graph [32] which can be recognized in linear time [9]. We prove that the problem is NP-complete. This implies that the realizability problems for interval graphs with difference constraints, and for interval graphs with individual upper and lower bounds on interval sizes, are also NP-complete. On the other hand, the problem is polynomial on UCO graphs.

1.4 Unit Interval Satisfiability Problems

In an *Interval Satisfiability* (ISAT) problem we are given a set of *events*, and for each pair of events a set of *permitted relations* between them. The set is any subset of $\{\prec, \cap, \succ\}$. The question is whether one can assign to each event an interval on the real line so that for each pair of events, their intervals satisfy one of their permitted relations. We denote the permitted relations between a pair of intervals by their concatenation. Hence, $x{\prec}{\cap}y$ is short for "$x{\prec}y$ or $x{\cap}y$", $x{\prec}{\succ}z$ is short for "$x{\prec}z$ or $x{\succ}z$", etc. In addition to the applications discussed above, ISAT problems arise in artificial intelligence, in the context of temporal reasoning [2] and medical diagnosis [28].

An important notion in studying interval satisfiability problems is the *domain*: It is the collection of possible sets of permitted relations in the input. For example, in the domain $\{\cap, \prec\succ\}$, for every two events either the input requires that they intersect or it requires that they should not intersect. These are the *only* types of input restrictions. Clearly, ISAT restricted to this domain is equivalent to the interval graph recognition problem. On the domain $\{\cap, \prec, \succ\}$, ISAT is equivalent to the interval order recognition problem [10]. By allowing other combinations of permitted relations, one gets various interesting generalizations of interval graphs and interval orders. Golumbic and Shamir [17] and Webber [35] have recently determined the polynomiality or NP-completeness of ISAT on all possible restricted domains.

We study here ISAT problems *with the additional constraint that all intervals must be of the same length.* We call this problem the *Unit Interval Satisfiability*

(UISAT) problem. We provide a complexity classification for all possible domains. While we build on the techniques developed by [17] and [35], the known results for ISAT do not imply anything for UISAT, due to the additional length restrictions on intervals.

The motivation to studying UISAT is similar to that for ISAT, as in many applications equal length (or, equivalently, non-containment [31]) of intervals is a natural condition. For example, in some planning and scheduling situations all events must have the same length. In many experiments of physical mapping, all the DNA fragments involved have nearly identical length, due to the recombination technique used to generate them (cf. [34, 15]). In these cases, partial information on intersection and precedence of events gives rise to UISAT problems.

Section 2 describes the linear algorithm for interval graph with side order constraints. Section 3 describes the polynomial algorithm for UCO graphs with constraints on differences between endpoints. Section 4 outlines the NP-completeness proof of interval graphs with exact given sizes. Section 5 provides a complexity classification of unit interval satisfiability problems on all possible domains. For space reasons, many details in the proofs and algorithms are necessarily omitted. Basic terminology on graphs and orders can be found in [32, 16, 10].

2 Interval Graphs with Side Order Constraints

The input for this problem is $J = (G, C)$ where $G = (V, E)$ is a graph and $C = \{(x, y)|x \prec y\}$ is a set of additional order constraints on pairs of non-adjacent vertices. We assume without loss of generality that all the intervals are open, and that G is connected. We denote by l_v, r_v the left and right endpoints of interval I_v.

Our algorithm builds on the new linear time algorithm for recognizing interval graphs of Hsu and Ma [20]. We decompose G into modules and develop new ideas to handle the order constraints. We first note some properties of the decomposition and introduce some definitions:

A set $M \subseteq V$ is called a *module* in $G = (V, E)$ if for each $x, y \in M$ and $u \notin M$: $xu \in E \Leftrightarrow yu \in E$. G is called *prime* if its only modules are the singleton vertices and V. For $M \subseteq V$, $G[M] = (M, E[M])$ is the subgraph of G induced on M. Define $N(M) = \{x \notin M|ux \in E, u \in M\}$. For a module M in the graph G, denote by $G[v/M]$ the graph $G' = (V', E')$, where $V' = V \setminus M \cup \{v\}$, and $E' = E[V \setminus M] \cup \{uv|u \in N(M)\}$. We say that $G[v/M]$ is obtained from G by *contracting M to v*. Following [20], we classify non-singleton modules into three categories: *type I* modules, which are cliques, *type II* modules, which are connected, but not cliques, and *type III* modules, which are not connected.

The key to the algorithm is the fact that modules behave to some extent like single vertices (see Fig. 1): We show that for certain modules, order constraints between all module vertices and any non-module vertex must be consistent.

Moreover, order constraints among vertices in the same module must be consistent. These properties facilitate recursive reduction of the problem, by contracting appropriate modules and passing over the constraints to the new instance. In order to do this efficiently, we handle each type of modules separately.

Lemma 1. *([20]) If S is a module in a chordal graph then either S is a clique or $N(S)$ is a clique.*

Lemma 2. *Let $G(V, E)$ be an interval graph and let $U \subseteq V$ be a set of vertices inducing a connected subgraph $G[U]$. In a realization $\{(l_v, r_v)\}_{v \in V}$, if $v, u \in U$ and $l_v < r_u$, then every point $x \in (l_v, r_u)$ is contained in some interval of a vertex in U.*

Let S be a set of vertices in an interval graph G, and let $\{I_x\}_{x \in V}$ be an open realization of G. Define $I^{out}(S) = I^{out} = (\min_{x \in S} l_x, \max_{x \in S} r_x)$. Define $p = \min_{x \in S} r_x$ and $q = \max_{x \in S} l_x$. Note that $p > q$ if and only if S is a clique. When $p \leq q$ we define $I^{in}(S) = I^{in} = [p, q]$. Pick a vertex $t \notin S$. If S is a clique we say that t *shatters* S if I_t intersects I^{out} but does not intersect $\cap_{x \in S} I_x$. If S is not a clique we say that t *shatters* S if either I_t has at least one endpoint in I^{in}, or $I_t \cap I^{out}$ and $I_t \prec\!\succ I^{in}$.

Lemma 3. *In a realization of a graph, no module which is maximal or connected is shattered.*

Proof. Suppose t shatters the module M. If M is a clique then t in adjacent to some but not all of the vertices in S so M cannot be a module. Hence, M is not a clique. If $I_t \cap I^{out}$ and $I_t \prec\!\succ I^{in}$, or I_t has exactly one endpoint in I^{in}, then again we get a contradiction to M being a module.

The only remaining case is when $I_t \subseteq I^{in}$. If M is connected then by Lemma 2 I_t meets some interval from M but not all, a contradiction. If M is not connected, define $S = \{x \in V | I_x \subseteq I^{in}\}$, and let $M' = M \cup S$. Clearly $t \in M' \setminus M$. It can be shown that M' is a module, contradicting the maximality of M. $\square$

A module M of G is called *compliant* in J if there exist no $v_1, v_2 \in M$ and $u \in V \setminus M$ such that C contains the constraints $v_1 \prec u$ and $u \prec v_2$. By Lemma 3, we can conclude:

Corollary 4. *In a realizable instance, every maximal or connected module is compliant.* $\square$

For a compliant module M in J, denote by $C[M]$ the set of constraints between vertices in M. Denote by $C[v/M]$ the set of constraints obtained from $C \setminus C[M]$, by substituting v instead of every $u \in M$ and deleting repetitions. The compliance of M assures us that the new set will contain no contradicting pairs of constraints (say, $x \prec y$ and $y \prec x$). Denote by $J[v/M]$ and $J[M]$ the instances $(G[v/M], C[v/M])$ and $(G[M], C[M])$ respectively (see Fig. 1). We say that $J[v/M]$ is obtained from J by *contracting M to v*.

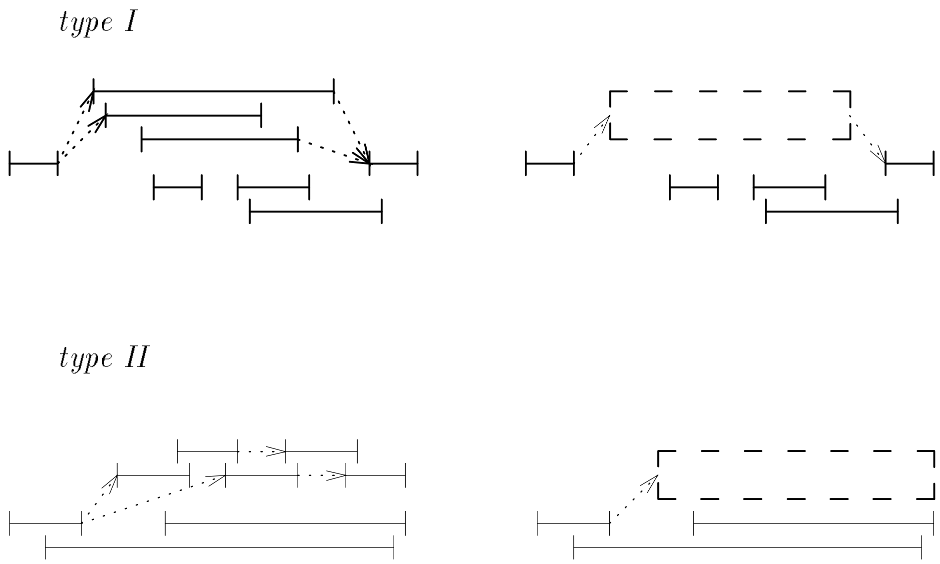

Fig. 1. Compliant modules and their contraction: An instance is represented by a realization of the interval graph, plus dotted arcs denoting the side order constraints. The boxes on the right are single intervals which replace the respective modules in the contraction and inherit the module's constraints. For type I and type II modules, the constraints between all module vertices and each non-module vertex (left) are consistent, therefore the compliant module can be contracted (right). For a type II module, order constraints within the module must also be consistent (this is verified in the sub-instance of that module in a lower level of the decomposition tree).

The algorithm will perform several phases of contractions and decompositions. Below we show that a single contraction of an appropriate module preserves the realizability of an instance. For instances $J, J_1, \ldots, J_k$ we say that J is *equivalent* to $\{J_1, \ldots, J_k\}$ if J is has a realization if and only if every J_i, $1 \leq i \leq k$, has one.

Proposition 5. *If M is a compliant module in J, which is connected or maximal, then J is equivalent to $\{J[v/M], J[M]\}$.*

Proof. ($\Rightarrow$): Suppose that $S = \{I_{v_1}, \ldots, I_{v_n}\}$ is a realization of J where for $1 \leq j \leq n$: $I_{v_j} = (l_j, r_j)$. The set of intervals $S_M = \{I_u | u \in M\}$ is an interval realization of $G[M]$. Moreover, S_M satisfies every constraint in $C[M]$, because S satisfies C. Hence, $J[M]$ is realizable.

Define the interval for the vertex v (which is a contraction of M) to be $I_v = I^{out}(M)$. The set of intervals $(S \setminus S_M) \cup \{I_v\}$ is a realization of $J[v/M]$: The intervals in $S \setminus M$ are left unchanged, hence, they satisfy both the appropriate intersection properties and the appropriate order constraints. It can also be shown, that I_v satisfies the intersection and order properties required from v by the definition of $I^{out}(M)$.

($\Leftarrow$): Suppose that $S = \{I_{v_i} | v_i \in V \setminus M\} \cup \{I_v\}$ and $S_M = \{I_{v_i} | v_i \in M\}$ are sets of intervals realizing $J[v/M]$ and $J[M]$, respectively. If M is a clique, let S' be a set of intervals containing $|M|$ copies of I_v (with the same endpoints). Otherwise, by Lemma 1, $N(M)$ is a clique, so the set $I = \cap_{u \in N(M)} I_u$ is an interval. Moreover, since M is a module, $I \cap I_v \neq \emptyset$. Construct the set of intervals S' by mapping each interval $I_u \in S_M$ to I'_u using the monotone linear transformation that maps $I^{out}(M)$ to $I \cap I_v$. In both cases, it can be proven that $S \setminus \{I_v\} \cup S'$ realizes J. □

By the above proposition, the following scheme solves our problem: Detect a module M which is connected or maximal. If it is not compliant in the instance J, answer 'no'. Otherwise continue recursively on each of the two instances $J[v/M]$ and $J[M]$. Upon reaching an instance with a prime or empty graph, solving the problem is immediate. Answer "yes" if and only if the answers to all the sub-instances were "yes". In order to implement this scheme in linear time, the algorithm proceeds in phases, as shown schematically below:

Do the following including checking compliance of the relevant modules in each step:

1. Contract all clique modules.
2. Form a decomposition tree for the Type-II modules and construct a sub-instance for each node.
3. For every sub-instance, contract all Type-III modules, solving realizability problems induced on each module and on the contracted instance.
4. Return $TRUE$ if and only if all sub-instances were true.

Finding the clique modules in G can be done in linear time [20]. Given these modules, contracting them in J is simple. Constructing a decomposition tree for the type II modules is possible in $O(m + n)$ time [20]. It can be shown that each node x in the type II decomposition tree gives rise to a sub-instance of our problem, the resulting set of sub-instances is equivalent to the original instance, and the total size of the resulting sub-instances can only decrease. Moreover, the graph of each sub-instance can be obtained from the type II decomposition, and the constraints in it are those between pairs of vertices whose lowest common ancestor is x. Each of these sub-instances contains no type II modules. The assignment of constraints to sub-instances can also be done in linear time, by answering the appropriate lowest common ancestor queries.

For an instance with only type III modules, these modules can be identified in $O(n+m)$ time [20]. Given these modules, they are simple to contract. Contracting these modules yields instances with prime or empty graph. These simple cases are easily solvable in linear time. In summary:

Theorem 6. *Deciding if a graph has a realization which satisfies additional order constraints can be done in linear time.* □

3 UCO Interval Graphs with Metric Constraints

We call an interval graph *uniquely clique-orderable* (UCO) if its maximal cliques have the same linear order, up to complete reversal, in every realization. UCO graphs can be recognized in linear time by applying the PQ-tree algorithm of Booth and Lueker [6], and noting that G is UCO if and only if the final tree contains a single Q-node. In this section we consider the following problem: Let the input be a UCO graph $G = (V, E)$, and a set S of linear inequalities on the variables $\{l_v, r_v\}_{v \in V}$ (which are the intervals endpoints), in which every inequality has the either the form $x - y < C_{xy}$, or the form $x - y \leq C_{xy}$, for variables x, y and constant C_{xy}. The question asked is whether there is a realization $\{[l_v, r_v]\}_{v \in V}$ for G, whose intervals endpoints solve S.

We reduce this problem to deciding whether there is a feasible solution for a system of inequalities, each of which is either of the form $x - y < C$ or $x - y \leq C$. Such a system is called *a system of difference constraints.*

Let $\prec_G \subseteq E(\bar{G})$ be the unique interval order admitted by G. It is known how to determine if G is UCO, and if so, how to compute $\prec_G$ in $O(m + n)$ time [6, 26, 20]. We construct two systems T and $\bar{T}$ of difference constraints on the variables $\{l_v, r_v\}_{v \in V}$, as follows: Both systems include all inequalities in S. In addition, for each $x, y \in V$: If $x \prec_G y$ then T contains an inequality $r_x < l_y$, and $\bar{T}$ contains an inequality $r_y < l_x$. If $xy \in E$ then both T and $\bar{T}$ contain an inequality $r_x \geq l_y$ (and $r_y \leq l_x$). With these definitions one can prove the following:

Lemma 7. *P has a realization if and only if either T or $\bar{T}$ have a feasible solution.*

Hence, we can solve our problem by deciding whether system T or $\bar{T}$ is feasible. We shall prove now that a system S of weak and strict difference constraints on n variables is reducible in linear time to a system S' of weak difference constraints, with numbers only $O(n)$ larger. (Standard transformation techniques [13] would give numbers $O(2^L)$ larger for binary input length L.) Assume all constants in S to be integral, and fix $\epsilon < \frac{1}{n}$. Define S' to include every weak inequality $x - y \leq c$ in S, and a weak inequality $x - y \leq c - \epsilon$ for every strict inequality $x - y < c$ in S.

Lemma 8. *S has a feasible solution if and only if S' has one.*

Proof. The 'if' direction is trivial, since a feasible solution to S' also satisfies S. The proof of the 'only if' uses the notion of the distance graph for the system of difference constraints (cf. [1, p. 103]), generalized to handle both strict and weak inequalities: For such a system T, construct a directed weighted graph $D(T)$ with vertex set V. For every constraint $x - y \leq C_{xy}$ or $x - y < C_{xy}$ add an arc (y, x) to $D(T)$ with weight C_{xy} and label the arc $\leq$ or $<$, respectively. Bellman [4] has shown that when all inequalities in T are weak, T is feasible if and only if $D(T)$ contains no negative cycles (see also [1, p. 103]).

Assume S' is not feasible. Then $D(S')$ contains a negative-weight cycle c. Let $w(c)$ and $w'(c)$ be the total weight of c in $D(S)$ and $D(S')$, respectively. Suppose $w(c) = w'(c) < 0$. Summing all inequalities along c in $D(S')$ we get $w'(c) \geq 0$, a contradiction. Otherwise, c contains an arc labeled $<$, $w'(c) < 0$, and since $|w(c) - w'(c)| \leq n\epsilon < 1$, it follows that $w(c) < 1$. Hence, $w(c) \leq 0$, since $w(c)$ is integer. Summing all inequalities along c in $D(S)$ we get $w(c) > 0$, a contradiction. □

By Lemma 7, and Lemma 8, solving an instance of our problem linearly reduces into determining if at least one of two systems of difference constraints is feasible. The feasibility of such a system with M weak inequalities on N variables, with sum of constants C, can be decided in $O(\min(NM, \sqrt{N}M \log NC))$ time [29, 12]. In our instance (G, S) there are n vertices, so $N = 2n$, $M = \Theta(n^2)$. Hence:

Theorem 9. *The realization problem on UCO interval graph with difference constraints can be solved in* $O(\min(n^3, n^{2\frac{1}{2}} \log nC))$ *time.* □

Interestingly, we cannot do any better: We can show a linear time reduction in the opposite direction, which enables us to decide the feasibility of a system of difference constraints, by determining whether a given UCO interval graph has a realization with intervals whose lengths match given bounds, and this is a special case of UCO interval graphs with metric constraints.

4 Interval graphs with exact given sizes

We consider here the most restricted version of difference constraints, in which an exact size is prescribed for each interval. Formally the problem is the following:

> METRIC INTERVAL GRAPH RECOGNITION (MIG):
> **INSTANCE:** A graph $G = (V, E)$ and function $s : V \to \mathbb{Q}^+$, assigning a *size* $s(v)$ for each vertex.
> **QUESTION:** Is there a realization $\{I_v\}_{v \in V}$ of G with $|I_v| = s(v)$ for every $v \in V$?

The problem arises naturally whenever a combination of intersection data and specific interval (event) lengths are known. Previous solved special cases were mentioned in the introduction.

Theorem 10. *MIG is NP-complete.*

The proof is by reduction from 3-Coloring. Unfortunately, the current proof we have is rather lengthy (15 pages) so we cannot describe it here. It is interesting to note that in this problem, there *is* loss of generality by assuming that all intervals are closed or open. We first construct a reduction to the problem with the additional designation of each interval as closed or open, and then show that although in general there may not be an all-closed realization whenever there is one with a prespecified designation, this is true for our reduction. Interestingly, the interval graph generated by our reduction has a decomposition tree of depth 2 only, so the problem is hard even in that case. Moreover, the length of all intervals involved is polynomial in $|V|$, so the problem is also Strongly NP-complete. By the results of the previous section, MIG is polynomial on UCO and hence, on prime interval graphs.

5 Unit Interval Satisfiability

In this section we study UISAT restricted to each possible domain. Recall that every domain is a collection of subsets of $\{\prec, \cap, \succ\}$. In a restricted problem only relations from the domain are allowed in the input. The problem restricted to domain Δ is denoted UISAT(Δ). Previously studied cases include UISAT($\{\prec\!\succ, \cap\}$), which is equivalent to unit interval graph recognition (a linear-time solvable problem) [31, 9], and UISAT(Δ_0) with $\Delta_0 = \{\prec\!\succ, \cap, \prec\!\cap\!\succ\}$, which is the NP-complete unit interval graph sandwich problem [15]. The version of UISAT(Δ_0) with the additional restriction that the maximum clique has size $\leq k$ was recently shown to be polynomial for fixed k but hard when k is a parameter [22].

We use the names of domains from [17] and [35]. We denote by N^D the number of occurrences of relation set D in the input. We assume here without loss of generality that all interval are closed.

5.1 The Domain $\Delta_1 = \{\cap, \prec, \succ, \prec\!\cap, \cap\!\succ, \prec\!\cap\!\succ\}$

Golumbic and Shamir [17] have shown that ISAT(Δ_1) reduces to deciding whether a system of difference constraints (with variables corresponding to interval endpoints) has a feasible solution. The requirement of unit length to all intervals is expressible as a set of additional linear constraints on the endpoints, and substituting for the variables of all right endpoints of intervals, one gets a system of difference constraints. Using arguments similar to those in Sect. 3, the problem reduces to deciding if a distance graph (in which all arc weights are integers between $-2n-1$ and $2n$) contains no negative cycle. Let $m = N^{\prec\cap} + N^{\prec} + 2N^{\cap}$. Using the algorithm of [29] to detect a negative cycle, we conclude:

Theorem 11. *UISAT(Δ_1) can be solved in $O(\sqrt{n}m \log n)$ time.* □

5.2 The Domain $\Delta_2 = \{\cap, \prec, \succ, \prec\!\succ\}$

Let J be an instance of UISAT(Δ_2). We first apply the linear algorithm for ISAT(Δ_2) which was described in Sect. 2. If the algorithm returns *FALSE*, then J has no proper interval realization. Otherwise, the algorithm gives a realization of J and, implicitly, an interval order $P = (V, <)$ which respects all the order constraints, but is not guaranteed to be a proper interval order. Form the interval graph $G = (V, E)$, where $uv \in E$ if and only if $v \cap u$ in J. Fishburn and Graham [11] have shown that an interval order is a proper interval order if and only if its incomparability graph G does not contain an induced $K_{1,3}$. By that result, as we have already established that P is an interval order, it suffices to check if G contains an induced $K_{1,3}$. To check this in linear time we recall some properties of interval graphs: Let $C_1, \ldots, C_k$ be the maximal cliques in a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$. The *clique matrix* of G is the $n \times k$ zero-one matrix $C(G) = (m_{ij})$ where $m_{ij} = 1$ if and only if $v_i \in C_j$. If the columns in $C(G)$ can be permuted so that the ones in each row are consecutive, then we say that $C(G)$ has the *consecutive ones* property, and we call such a permutation of the columns a *consecutive (clique) order*. According to Gilmore and Hoffman [14], G is an interval graph if and only if $C(G)$ has the consecutive ones property. For $C(G)$ in a consecutive order, let L_i and R_i be the column indices of the leftmost and rightmost 1's, respectively, in the i-th row of $C(G)$. Clearly, $\{[L_i, R_i]\}_{v_i \in V}$ is a realization of G. Such realization is called *canonical*. One can now prove the following:

Lemma 12. *An interval graph $G = (V, E)$ is a unit interval graph if and only if no canonical realization $\{[L_i, R_i]\}_{v_i \in V}$ for G satisfies $L_i < L_j$ and $R_i > R_j$ for any adjacent vertices $v_i, v_j \in V$.*

Let $m = |E| = N^{\cap}$. The clique matrix $C(G)$ of an interval graph and a consecutive clique order can be computed in $O(n + m)$ time (see, e.g., [6]). The consecutive permutation on the columns can also be performed in $O(n + m)$ time. The indices L_i and R_i can be computed, for all vertices v_i, in a total time of $O(n + m)$, and the check, for each edge $v_i v_j$ whether $L_i < L_j$ and $R_i > R_j$, takes $O(m)$ time for all edges. Therefore:

Theorem 13. *UISAT(Δ_2) can be solved in $O(n + N^{\cap} + N^{\prec})$ time.* □

5.3 The Domain $\boldsymbol{\Delta}_3 = \{\prec\!\succ, \prec, \succ, \prec\!\cap\!\succ\}$

It is easy to see that an instance of UISAT(Δ_3) is satisfiable if and only if it is satisfiable as an instance of ISAT(Δ_3). The results of [17] on ISAT(Δ_3), together with the above observation imply:

Theorem 14. *UISAT(Δ_3) can be solved in $O(n + N^{\prec})$ time.* □

5.4 The Domain $\boldsymbol{\Delta}_5 = \{\cap\!\succ, \prec\!\cap, \prec\!\succ\}$

This problem appears to be technically the hardest in this section. In fact, the corresponding problem for ISAT was the only problem left open in the classification of ISAT domains of [17]. Weber [35] subsequently proved it is NP-complete.

Theorem 15. *UISAT(Δ_5) is NP-complete.*

We omit the proof which is based on a reduction from Not-All-Equal Satisfiability. Unlike Webber's proof for ISAT, we use a symmetric six-interval gadget for each clause. It is interesting to note that the construction generates a directed graph whose diameter is at most four, irrespective of the size of the formula.

5.5 Summary

Every possible domain Δ is either contained in one of $\Delta_1, \Delta_2, \Delta_3$, and in this case UISAT(Δ) is polynomial, or it contains one of Δ_0, Δ_5, in which case UISAT(Δ) is clearly NP-hard. Thus, we have resolved the complexity of UISAT(Δ) for all possible domains Δ. In the final analysis, there are 21 polynomial domains and 10 NP-complete ones. (The symmetry of some relation sets reduces the number of non-isomorphic nonempty domains to 31.) As noted by [17], one of the byproducts of this classification is the ability to speed up enumerative solution of problems on NP-complete domains, by finding the polynomial domain which contains the largest number of relations in the input, thereby minimizing the size of enumeration needed.

References

1. R. K. Ahuja, T. A. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.
2. J. F. Allen. Maintaining knowledge about temporal intervals. *Comm. ACM*, 26:832–843, 1983.
3. J. F. Allen. *Reasoning about plans*. Morgan Kaufman, 1991.
4. R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.
5. S. Benzer. On the topology of the genetic fine structure. *Proc. Nat. Acad. Sci. USA*, 45:1607–1620, 1959.
6. K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and planarity using PQ-tree algorithms. *J. Comput. Sys. Sci.*, 13:335–379, 1976.
7. A. V. Carrano. Establishing the order of human chromosome-specific DNA fragments. In A. D. Woodhead and B. J. Barnhart, editors, *Biotechnology and the Human Genome*, pages 37–50. Plenum Press, 1988.
8. C. H. Coombs and J. E. K. Smith. On the detection of structures in attitudes and developmental processes. *Psych. Rev.*, 80:337–351, 1973.
9. X. Deng, P. Hell, and J. Huang. Linear time representation algorithms for proper circular arc graphs and proper interval graphs. Technical report, School of Computing Science, Simon Fraser University, 1993.
10. P. Fishburn. *Interval Orders and Interval Graphs*. Wiley, New York, 1985.
11. P. Fishburn and R. L. Graham. Classes of interval graphs under expanding length restrictions. *J. Graph Theory*, 9:459–472, 1985.
12. H. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18:1013–1036, 1989.
13. P. Gács and L. Lovász. Khachiyan's algorithm for linear programming. *Math. Prog. Study*, 14:61–68, 1981.
14. P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canad. J. Math.*, 16:539–548, 1964.
15. P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. Technical report, Computer Science Dept., Tel Aviv University, April 1993. To appear in *Journal of Computational Biology*.

16. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, New York, 1980.
17. M. C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *J. ACM*, 40:1108–1133, 1993.
18. E. D. Green and M. V. Olson. Chromosomal region of the Cystic Fibrosis gene in yeast artificial chromosomes: a model for human genome mapping. *Science*, 250:94–98, 1990.
19. G. Hajös. Über eine art von graphen. *Intern. Math. Nachr.*, 11, 1957. problem 65.
20. W-L. Hsu and T-H Ma. Substitution decomposition on chordal graphs and applications. In W-L. Hsu and R. C. T. Lee, editors, *Proc. 2nd Int. Symp on Algorithms (ISA '91)*, pages 52–60. Springer-Verlag, 1991. LNCS 557.
21. G. Isaak. Discrete interval graphs with bounded representation. *Discrete Applied Mathematics*, 33:157–183, 1993.
22. H. Kaplan and R. Shamir. Pathwidth, bandwidth and completion problems to proper interval graphs with small cliques. Technical report, CS Department, Tel Aviv University, November 1993. To appear in *SIAM J. Comput.*
23. R. M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *Proc. 25th STOC*, pages 278–285. ACM Press, 1993.
24. D. G. Kendall. Incidence matrices, interval graphs, and seriation in archaeology. *Pacific J. Math.*, 28:565–570, 1969.
25. N. Korte and R. H. Möhring. Transitive orientation of graphs with side constraints. In H. Noltemeier, editor, *Proc. of the International workshop on Graph-theoretic concepts in Computer Science (WG '85)*, pages 143–160, Linz, 1985. Universitätsverlag Rudolf Trauner.
26. N. Korte and R. H. Möhring. An incremental linear time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18:68–81, 1989.
27. P. Lichter et al. High-resolution mapping of human chromosome 11 by in situ hybridization with cosmid clones. *Science*, 247:64–69, 1990.
28. K. Nökel. *Temporally Distributed Symptoms in Technical Diagnosis.* Lecture Notes in Artificial Intelligence 517. Springer Verlag, 1991.
29. J. B. Orlin and R. K. Ahuja. New scaling algorithms for the assignment and minimum cycle mean problems. *Mathematical Programming*, 54:41–56, 1992.
30. C. Papadimitriou and M. Yannakakis. Scheduling interval ordered tasks. *SIAM J. Comput.*, 8:405–409, 1979.
31. F. S. Roberts. Indifference graphs. In F. Harary, editor, *Proof Techniques in Graph Theory*, pages 139–146. Academic Press, New York, 1969.
32. F. S. Roberts. *Discrete Mathematical Models, with Applications to Social Biological and Environmental Problems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.
33. S. A. Ward and R. H. Halstead. *Computation Structures.* MIT Press, 1990.
34. J.D. Watson, M. Gilman, J. Witkowski, and M. Zoller. *Recombinant DNA.* W.H. Freeman, New York, 2nd edition, 1992.
35. A.B. Webber. Proof of an interval satisfiability conjecture. manuscript, Wesstern Illinois University, 1994.

This article was processed using the LaTeX macro package with LLNCS style