Satisfiability problems on intervals and unit intervals*

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

January 1995, revised November 1995

Abstract

For an interval graph with some additional order constraints between pairs of non-intersecting intervals, we give a linear time algorithm to determine if there exists a realization which respects the order constraints. Previous algorithms for this problem (known also as ***seriation with side constraints***) required quadratic time. This problem contains as subproblems interval graph and interval order recognition. On the other hand, it is a special case of the interval satisfiability problem, which is concerned with the realizability of a set of intervals along a line, subject to precedence and intersection constraints. We study such problems for all possible restrictions on the types of constraints, when all intervals must have the same length. We give efficient algorithms for several restrictions of the problem, and show the NP-completeness of another restriction.

1 Introduction

Two intervals x, y on the real line may either intersect or one of them is completely to the left of the other. When they intersect we denote their relation by $x \cap y$ (read 'x intersects y'), and if x

*Portions of this paper were presented at the Third European Symposium on Algorithms, Corfu, Greece, 1995 [31]

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

is completely to the left of y we write $x \prec y$ or, equivalently, $y \succ x$. These simple relations are the starting point for an amazing richness of combinatorial, algorithmic and complexity questions, with applications in vast and numerous areas.

The problems which we shall address here are concerned with the realizability of a set of intervals along a line, subject to constraints. They can be stated as generalizations of interval graphs, of interval orders, or as temporal reasoning problems. We have chosen to use the latter representation just for the sake of having a common language for all our 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 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 the permitted relations prescribed by the input. In case the answer is positive, we would also like to construct such a solution, or ***realization***.

Example: The events are x, y, z, and the permitted relations are: between x and y: $x \prec y$; between x and z: $x \prec z$ or $x \succ z$; between y and z: $y \succ z$ or $y \cap z$.
By convention, the converse relations ($y \succ x$, $z \succ x$ or $z \prec x$, $z \prec y$ or $z \cap y$) are implied and not stated explicitly. Two possible realizations are given in figure 1.

Figure 1: Two possible realizations for the example.

To save on writing, we shall denote the permitted relations between a pair of intervals by concatenating these relations. 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 $x \succ z$", and $x \prec\cap\succ y$ means there is no restriction on the relation of the two intervals (all relations are permitted).

Interval Satisfiability problems 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. Hajös [17] originally defined interval graphs in order to study the intersection data of time intervals. Other applications include seriation in archaeology [22, 23], behavioral psychology [8], scheduling [30], circuit design [36] and combinatorics [33]. In artificial intelligence, a lot of such work has been done in temporal reasoning [2], planning [3] and medical diagnosis [28]. There are also non-temporal applications: In genetics, arrangement of genetic material along a linear chain motivated Benzer to study similar problems [5]. A central challenge in modern molecular biology and the Human Genome Project is physical mapping of DNA [7, 21]: It calls for the reconstruction of a map (a realization) for a collection of DNA segments, based on partial information on intersection and precedence relations between pairs of segments.

An important notion in studying interval satisfiability problems is the ***domain***: It is the collection of 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, but in that case their order is not prescribed. These are the ***only*** types of input restrictions. Clearly ISAT restricted to this domain is equivalent to the well-known ***interval graph*** recognition problem (on the graph with vertices corresponding to events and edges corresponding to intersecting pairs of events), cf. [15, 27]. Another example is the domain $\{\cap, \prec, \succ\}$: Here the only permitted relations are either intersection or order. ISAT on this domain is equivalent to the ***interval order*** recognition problem [11]. By allowing other combinations of permitted relations, one gets various interesting generalizations of interval graphs and interval orders. One can thus study ISAT for all possible domains. There are seven non-empty permitted relations ($\{\prec, \cap, \succ, \prec\!\succ, \prec\!\cap, \cap\!\succ, \prec\!\cap\!\succ\}$) but two pairs ($\prec$ and $\succ$, $\prec\!\cap$ and $\cap\!\succ$) are converses, so if one is included in a domain then the other is also implicitly included. Hence, there are 5 distinct relations which form (as non-empty subsets) 31 possible domains. Golumbic and Shamir [16] have studied ISAT on restricted domains and classified the complexity (i.e., determined the polynomiality or NP-completeness) of all but four domains, which they conjectured to be NP-complete. Recently, Webber [38] has completed the classification by proving that conjecture.

One of our goals in this paper is to perform such a classification ***with the additional constraint that all intervals must be of the same length.*** We call this problem the ***Unit Interval Satisfiability*** (UISAT) problem. In section 4 we study restricted domains of UISAT and provide a complexity classification for all possible domains. While we build on the techniques developed by [16] and [38], the known results for ISAT do not imply anything for UISAT, due to the additional length restrictions on intervals. In the final analysis, 21 of the domains are shown to have polynomial solutions, and 10 are NP-complete.

The motivation to studying UISAT is similar to that for ISAT, as in many applications non-containment 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 (e.g., when the fragments originate from a clone library of cosmids or lambda, cf. [37, 14]). In these cases, partial information on intersection and precedence of events gives rise to UISAT problems.

Another problem which we study here is the problem of ***seriation with side constraints***: Order a given set of events subject to an interval graph G (representing all pairs of events which must intersect) a partial order P (representing some - but not necessarily all - non-intersecting pairs which must appear in a certain order). This is indeed ISAT restricted to the domain $\{\cap, \prec\!\succ, \prec, \succ\}$. Note that this problem contains as special cases the interval graph and the interval order recognition problems. An equivalent order-theoretic statement of the problem is: Given a (cocomparability) graph $G = (V, E)$ and a partial orientation R_0 of the edges of its complement $\bar{G} = (V, \bar{E})$, (i.e., $R_0 \subseteq \bar{E}, R_0 \cap R_0^{-1} = \emptyset$), does there exist an interval order orientation F of $\bar{G}$ that respects R_0 (i.e., $R_0 \subseteq F$)? Korte and Möhring [24] give applications of this problem to seriation in archaeology and to consecutive retrieval with access priorities. This problem arises also in physical mapping experiments, when order information on some pairs of segments is available (see, e.g., [26]).

Golumbic and Shamir [16] give a simple $O(n^3)$ algorithm for the problem on a graph with n vertices, and Korte and Möhring [24] give an $O(n^2)$ algorithm using MPQ-trees. We give here an algorithm of complexity $O(n+m+l)$, where n, m and l are the number of events, intersections and ordered pairs in the input, respectively (or, in the order-theoretic formulation above, $|V|$, $|E|$ and $|R_0|$). Hence, the algorithm is linear, and provides a substantial improvement over the previous algorithms in situations where interval graph is sparse (and the interval order is dense). Such sparse graphs are typical to many applications (see, e.g., [20] for the discussion of sparse graphs in physical mapping). Our algorithm builds on the new interval graph recognition algorithm by Hsu and Ma [19].

Section 2 contains basic definitions and background. Section 3 describes the linear algorithm for interval graph with side constraints. Section 4 provides a complete complexity classification of unit interval satisfiability problems on all possible domains.

2 Preliminaries and Background

All graphs in this paper are undirected, finite, and simple unless noted otherwise. Let $G = (V, E)$ be a graph. We use $V(G)$ for the vertex set of G and $E(G)$ for its edge set. We use both uv and (u, v) to denote the edge between u and v. For a vertex $u \in V$, the set $N(u) = \{v \in V | uv \in E\}$ is called the ***neighborhood*** of u. For a set of vertices $U \subseteq V$, the ***neighborhood*** of U is $N(U) = \{v \in V \setminus U | \exists u \in U \ s.t. \ uv \in E\}$. The subgraph ***induced by*** U is the graph $G[U] = (U, E[U])$ where $E[U] = \{uv | u \in U \text{ and } v \in U\}$.

A set $M \subseteq V$ is called a ***module*** in $G = (V, E)$ if for each $x, y \in M$, and for each $u \notin M$: $xu \in E \Leftrightarrow yu \in E$. Surely, V is a module, and for each $v \in V$, $\{v\}$ is a module. Such modules are called ***trivial***. If all modules in G are trivial, then G is called ***prime***. 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. We classify the 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.

A graph $G = (V, E)$ is called an ***intersection graph*** of a family of sets $S = \{I_v\}_{v \in V}$, if $I_v \cap I_u \neq \emptyset \Leftrightarrow vu \in E$. G is called an ***interval graph*** if it is an intersection graph of a family $S = \{I_v\}_{v \in V}$ of intervals on the real line. In that case, S is then called a ***realization*** of G. Depending on the convention, all intervals may be either closed or open, with no loss of generality. For each interval $I = (l, r)$, the length of I, denoted $|I|$, is $r - l$. If G has a realization in which all the intervals are of equal length, then it is called a ***unit interval graph***. If G has a realization in which no interval in S properly contains another, then G is called a ***proper interval graph***. Roberts [32] proved that a finite graph is a proper interval graph if and only if it is a unit interval graph. Interval graphs are recognizable in linear time ($O(n + m)$ time, where $n = |V|$ and $m = |E|$) [6, 25, 19], and so are unit interval graphs [9].

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 [13], 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 a ***canonical realization*** of G.

For every two intervals x, y on the line, exactly one of the following ***atomic relations*** holds:

- $x \prec y$: The interval x is completely to the left of y.
- $x \succ y$: The interval x is completely to the right of y.
- $x \cap y$: The intervals x and y have a non-empty intersection.

Let $P = (V, <)$ be a partial order. Call $<$ an ***interval order*** if there exists a set of intervals

$S = \{I_v\}_{v \in V}$ such that $v < u$ if and only if $I_v \prec I_u$. S is called a ***realization*** for P. Call $G = (V, E)$ the ***incomparability graph*** of P, if for each $u, v \in V$, $uv \in E$ if and only if u and v are incomparable in P, i.e., $u \not< v$ and $v \not< u$. Hence, G is an interval graph if and only if it is the incomparability graph of some interval order. Interval orders are recognizable in linear $(O(|V| + |<|))$ time [30, 4]. One can analogously define unit (proper) interval orders. The polynomiality of recognizing such orders follows from their characterization in [11, chapter 8].

Denote by $v(I, I')$ the atomic relation between the intervals I, I' on the line. Golumbic and Shamir [16] defined the ***interval satisfiability problem*** (ISAT) as follows: An ***instance*** J of the problem consists of a set V of n events, and for each pair of events, x,y, a ***relation set*** $D(x, y)$ of permitted atomic relations between them. A set of intervals $\{I_x\}_{x \in V}$, in which for each $x, y \in V$, $v(I_x, I_y) \in D(x, y)$ is called a ***consistent realization*** for J. If such a set exists, we also say that the instance J is ***satisfiable***.

The relation set $D(x, y)$ represents a disjunction of (one or more) possible atomic relations permitted between events x and y in any realization of the instance. Denote, for short, the possible non-empty sets of atomic relations $\{\prec, \cap\}$, $\{\prec, \succ\}$, $\{\cap, \succ\}$, and $\{\prec, \cap, \succ\}$, as $\prec\!\cap$, $\prec\!\succ$, $\cap\!\succ$, and $\prec\!\cap\!\succ$, respectively. We also write $xD(x, y)y$ instead of $v(I_x, I_y) \in D(x, y)$. In this notation the non-atomic relation sets are:

- $x \prec\!\cap y$: $x \prec y$ or $x \cap y$: interval x is not completely to the right of interval y.
- $x \cap\!\succ y$: $x \cap y$ or $x \succ y$: interval x is not completely to the left of interval y.
- $x \prec\!\succ y$: $x \prec y$ or $x \succ y$: the intervals do not intersect, but may appear in any order.
- $x \prec\!\cap\!\succ y$: $x \prec y$ or $x \cap y$ or $x \succ y$: no restrictions apply.

Each subset $\Delta \subseteq \{\prec, \succ, \cap, \prec\!\cap, \prec\!\succ, \cap\!\succ, \prec\!\cap\!\succ\}$ is called a ***domain***. ISAT(Δ) denotes the ISAT problem restricted to the case where for every pair of events x, y: $D(x, y) \in \Delta$. Note that the permitted relations must be in the domain for ***all*** pairs of events, including those which are perhaps not explicitly stated. A complete complexity classification of ISAT on all restricted domains has been given by [16] and [38]. Define the ***unit interval satisfiability problem*** (UISAT) as follows : An instance J of the problem has input identical to that of ISAT. The goal is to decide if there exists a realization for J in which all intervals have equal length. In Section 4, we study the complexity of UISAT(Δ) for all possible domains Δ.

We use here the algebra of three relations defined by [16], and not the thirteen-relation algebra of Allen [2]. Like in [16], this is done in order to simplify the analysis (there are $2^{2^{13}-1}$ domains for that larger algebra). The choice of the small algebra is motivated by body of knowledge on interval graphs and orders, as well as by the problem of physical mapping of

DNA, in which the input corresponds to the simpler algebra.

We shall always use n to denote the number of events in our instance. For a relation set D we let $N^{D} = |\{x, y|D(x, y) = D\}|$. Note that for any domain $\Delta = \{D_1, \ldots D_k\}$, $\sum_{i=1}^{k} N^{D_i} = n(n-1)$, but one can assume that only $k-1$ relation sets are given explicitly in the input, so subquadratic times in n are possible. Table 1 summarizes some of the best known results on restricted domains of ISAT and UISAT problems.

Restricted Domain	Class	complexity	Reference
$ISAT(\{\prec\!\!\succ, \cap\})$	Interval graphs	$O(n + N^{\cap})$	[13, 12, 6, 25]
$ISAT(\{\prec, \succ, \cap\})$	Interval orders	$O(n + N^{\prec})$	[10, 30, 4]
$ISAT(\{\prec\!\!\succ, \cap, \prec\cap\succ\})$	Interval sandwich	NP-complete	[16]
$UISAT(\{\prec\!\!\succ, \cap\})$	Unit Interval graphs	$O(n + N^{\cap})$	[32, 9]
$UISAT(\{\prec\!\!\succ, \cap, \prec\cap\succ\})$	Unit Interval sandwich	NP-complete	[14]

Table 1: Five cases of interval and unit interval satisfiability problems and their complexity.

3 Interval Graph with Side Order Constraints

In this section we study the interval satisfiability problem restricted to the domain $\Delta_2 = \{\cap, \prec, \succ, \prec\!\!\succ\}$. In other words, we are given a set of n events, and for each pair of events we know if their intervals should intersect or should not. For some non-intersecting pairs we know in addition which one must appear to the left of the other. We let $m = N^{\cap}$ and $l = N^{\prec} + N^{\succ}$.

This problem contains as special cases the recognition of interval graphs (where $l = 0$) and interval orders (where $N^{\prec\!\!\succ} = 0$). We give here an $O(n + m + l)$ time algorithm, improving over the extant algorithms of [24, 16]. Our algorithm is based on the new linear algorithm of Hsu and Ma for recognizing interval graphs [19].

We assume that only the pairs of events with relation sets $\cap$ and $\prec$ are given explicitly, and the $\prec\!\!\succ$-related pairs are implicit. More precisely, we assume that the input is given as $J = (G, C)$ where $G = (V, E)$, V is the set of events, $E = \{x, y|x \cap y\}$ and $C = \{(x, y)|x \prec y\}$ is the set of additional constraints, presented as ordered pairs. Hence, the algorithm is linear, and when $m, l = o(n^2)$ this is an improvement over the previous algorithms. Note that the case $m = o(n^2)$ and in fact even $m = \Theta(n)$ occur quite often in applications (cf. [20]). Thus, the algorithm is especially suitable for those cases where G is sparse and there are relatively few additional order constraints. Denote $Size(J) = m + n + l$.

3.1 Decomposition into Compliant Modules

Modular decomposition (also known as substitution decomposition) has recently been used by Hsu and Ma to devise a new linear time algorithm for recognizing interval graphs [19]. To solve $ISAT(\Delta_2)$ in linear time, we shall build on that algorithm and introduce new ideas to handle the order constraints. We assume without loss of generality that all the intervals are open, and that G is connected.

We first need some simple lemmas on the structure of modules in an interval graph. The following property [19] is immediate by the fact that a chordal graph does not contain an induced cycle of length ≥ 4 :

Lemma 3.1 *(Hsu and Ma, 1991) If S is a module in a chordal graph then either S is a clique or $N(S)$ is a clique.*

Lemma 3.2 *Let $G(V, E)$ be an interval graph and let H be a connected subgraph in G. In a realization $\{(l_v, r_v)\}_{v \in V}$, if $v, u \in H$ and $l_v < r_u$, then every point $x \in (l_v, r_u)$ is contained in some interval of a vertex in H.*

Proof. Since H is connected, it contains a path $v = v_0, \ldots, v_p = u$. Let k be the largest index such that $l_{v_k} < x$, and let $w = v_k$. Note that k is properly defined, since $l_v < x$. If $w = u$ then surely $l_w < x < r_w$. If $w \neq u$ then $l_w < x \leq l_{v_{k+1}}$ due to the maximality of k. But $l_{v_{k+1}} < r_w$, because v_{k+1} and w must intersect, yielding $l_w < x < r_w$. ∎

Lemma 3.3 *Every connected component of a module is a module.*

Proof. Let M be a module in G, and let M' be a connected component in $G[M]$, the subgraph of G induced by M. Let $v_1, v_2 \in M'$ and $u \in V \setminus M'$. If $u \in M \setminus M'$, then since M' is a connected component, $(u, v_1) \notin E$, and also $(u, v_2) \notin E$. Otherwise $u \in V \setminus M$, then since M is a module, $(u, v_1) \in E \Leftrightarrow (u, v_2) \in E$. ∎

Definition 3.4 *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 $I_t \mathrel{-\!\!\diamond\!\!-} I^{in}$.*

The following lemma justifies why we defined shattering as a property of vertices and not of intervals in a particular realization:

Definition 3.5 *We say that a module M is* good *if it is connected, or if M is maximal and $N(M) \neq \emptyset$.*

Lemma 3.6 *In a realization of an interval graph, no good module is shattered.*

Proof. Suppose t shatters the good module M. If M is a clique then t is 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 3.2 I_t meets some interval from M but not all, a contradiction. If M is not connected then for every $x \in N(M)$, $I_x \supseteq I^{in}$. Define $S = \{x \in V | I_x \subseteq I^{in}\}$. By assumption, $t \in S \neq \emptyset$. We shall show that $M' = M \cup S$ is a module, a contradiction to the maximality of M.

Let $y \notin M'$. If $y \in N(M)$ then since $I_y \supseteq I^{in}$, y is adjacent to every vertex in M'. If $y \notin N(M)$, it follows that y is not adjacent to any vertex in M'. Hence, M' is a module. Moreover, since $N(M) \neq \emptyset$, $M' \subset V$, so M' is non-trivial. ■

Definition 3.7 *Let $J = (G, C)$ be an instance of $ISAT(\Delta_2)$ where $G = (V, E)$. 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 $u \prec v_2$. In this case we also say that M* complies *with C.*

Corollary 3.8 *In a satisfiable instance, every good module is compliant.*

Proof. Suppose a realization of the instance has $x \prec y \prec z$ where x, z belong to a module M and y does not. Since y shatters M, lemma 3.6 implies that M cannot be good. ■

Note that if the module M is not connected, then the assertion above fails (consider, for example, the module $\{x, z\}$ with constraints $x \prec y \prec z$).

Our algorithm decomposes the instance into modules and we shall need to decompose the set of constraints accordingly. We introduce some definitions to describe the products of the decomposition process, in terms of the graph and the constraints:

Definition 3.9 *Let $J = (G, C)$ be an instance of $ISAT(\Delta_2)$, and let M be a compliant module in it. 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. Note that 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.*** $J[v/M]$ ***will be called the instance obtained by contracting*** M ***to*** v ***in*** J***.***

The algorithm will perform several phases of contractions and decompositions. Below we show that a single contraction preserves the satisfiability of an instance. Let us first define a notion of equivalence between instances:

Definition 3.10 ***For instances*** J, J' ***we say that*** J ***is*** equivalent ***to*** J' ***if*** J ***is satisfiable if and only if*** J' ***is. For instances*** $J, J_1, \ldots, J_k$ ***we say that*** J ***is*** equivalent ***to*** $\{J_1, \ldots, J_k\}$ ***if*** J ***is satisfiable if and only if every*** J_i, $1 \leq i \leq k$, ***is satisfiable. Similarly we can define the equivalence of two sets of instances. Note that this relation is indeed a set theoretic equivalence.***

Proposition 3.11 ***If*** M ***is a good, compliant module in the instance*** $J = (G, C)$, ***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 $I_{v_j} = (l_j, r_j)$. 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 satisfiable.

Define the interval for the vertex v (which is a contraction of M) to be $I_v = I^{out}(M)$. We claim that the set of intervals $(S \setminus S_M) \cup \{I_v\}$ is a realization of $J[v/M]$:

- Let $G[v/M] = (V_{v/M}, E_{v/M})$. Since intervals in $S \setminus S_M$ are left unchanged, their intersection graph and order are unchanged. Hence, intervals $I_u, I_{u'} \in S \setminus S_M$, intersect if and only if $(u, u') \in E$ if and only if $(u, u') \in E_{v/M}$. Moreover, all constraints between vertices in $S \setminus S_M$ remain satisfied.
- If $(v, v_j) \in E_{v/M}$ then since M is a module, for every $u \in M$, $(u, v_j) \in E$. Hence, $I_{v_j} \cap I_u$ and since $I_u \subseteq I_v$, it follows that $I_{v_j} \cap I_v$.
- Let $v_j \notin M$, and suppose $I_{v_j} \cap I_v$. If M is a clique then I_{v_j} must intersect $\cap_{x \in M} I_x$ or else M is not a module. Otherwise I_{v_j} must properly contain I^{in}, by lemma 3.6. In both cases we get that $(v, v_j) \in E_{v/M}$.

- Let C_i be a constraint in $C[v/M]$ involving v. W.l.o.g., C_i is the constraint $v \prec x$, where $x \in V \setminus M$. Let the corresponding constraint in C, from which C_i was generated, be $y \prec x$ for some $y \in M$. Since C is satisfied by S, $I_y \prec I_x$, so $I_v \prec\cap I_x$. If I_x intersects I_v but does not intersect I_y, then x shatters M, a contradiction. Therefore $I_v \prec I_x$. It follows that every constraint in $C[v/M]$ is satisfied by $(S \setminus S_M) \cup \{I_v\}$.

($\Leftarrow$): Suppose that $S = \{I_{v_i} | v_i \in S \setminus M\} \cup \{I_v\}$ 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, then examine the set of intervals S', generated from S by replacing I_v with $|M|$ copies of it (with the same endpoints). It is clear that S' is a realization of G, and since M is compliant, S' satisfies C.

If M is not a clique, then by lemma 3.1, $N(M)$ is a clique, so the set $I = \cap_{u \in N(M)} I_u$ is an interval. Moreover, since M is a module, $I \cap I_v$. Let $I' = I \cap I_v$. 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', as described in figure 2. We claim that $S \setminus \{I_v\} \cup S'$ realizes J: The relative positions of intervals inside M are preserved by the mapping, and for each $I'_u \in S'$ the intervals intersecting it from $S \setminus \{I_v\}$ are exactly $N(M)$. Therefore $S' \cup S \setminus \{I_v\}$ is a realization of G. Finally, since M is compliant, and $C[v/M]$ is satisfied by $J[v/M]$, every constraint in C is satisfied. ■

Our algorithm for ISAT(Δ_2) will first eliminate (contract) all clique modules in J, and then form a decomposition into type II modules. On each of the resulting instances it then reduces type III modules and examines the satisfiability of the resulting instance in which the graph is prime. A failure in any step causes the failure of the whole algorithm. We now describe in detail each of the phases.

3.2 Reducing Type I Modules

We first describe an algorithm which, on a given instance J, either decides that J is unsatisfiable, or generates a new equivalent instance J' with no type I modules. The size of J' will not exceed the size of J.

Procedure Reduce-Modules-I(J, J')

Input: instance $J = (G, C)$.

Output: either *FALSE*, or an equivalent instance $J' = (G', C')$, without type I modules.

1. Find all maximal type I modules, $M_1, M_2, \ldots, M_{n'}$, including singletons. For each $v \in V(G)$, let $M_{i(v)}$ be the maximal type I module which contains v.

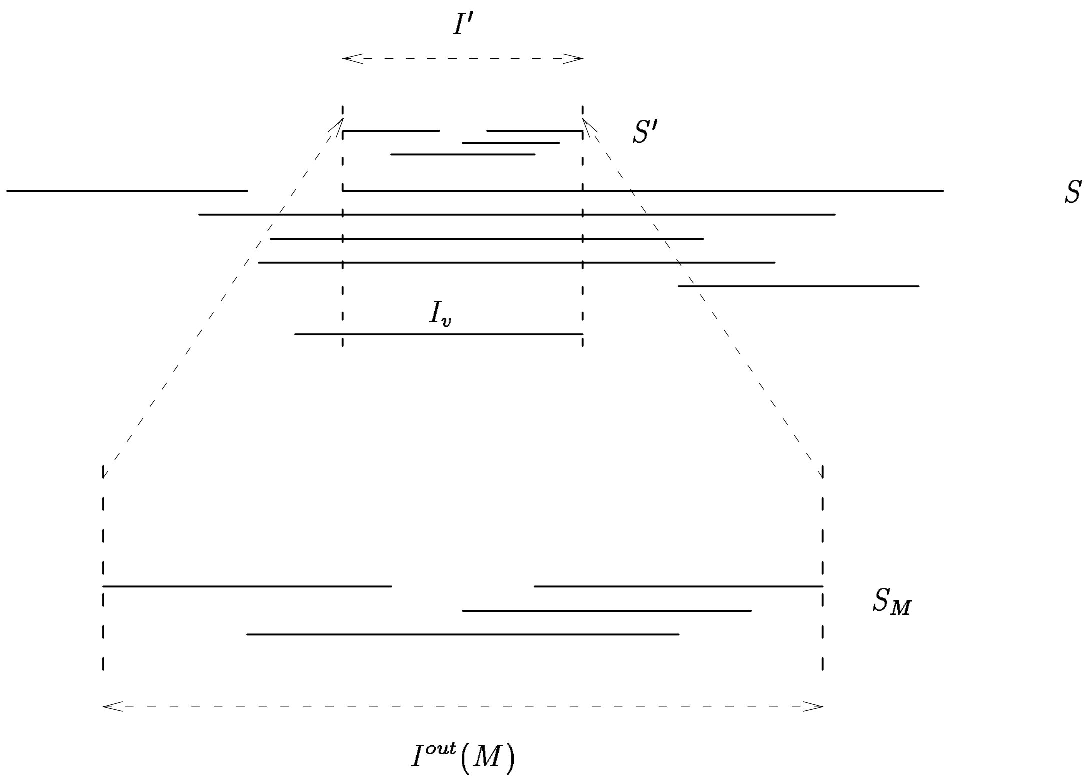

Figure 2: Construction of a realization for G

2. Contract every module M_i to a vertex v'_i. Call the resulting graph G'. Set $C' \leftarrow \emptyset$
3. For each constraint $v \prec u$: If there is a constraint $v'_{i(v)} \succ v'_{i(u)}$ in C', then return *FALSE*. Otherwise add the constraint $v'_{i(v)} \prec v'_{i(u)}$ to C', unless it already exists.

Theorem 3.12 *The algorithm terminates correctly in* $O(l+m+n)$ *time.*

Proof. Correctness: Suppose the algorithm returns *FALSE*. Then there exist maximal type I modules M_i, M_j where $x, y \in M_i$, $z, w \in M_j$ with constraints $x \prec z$, $y \succ w$ in C, and $x \neq y$ or $z \neq w$. As every such module is a clique, no realization is possible, since $I_x \prec (I_z \cap I_w) \prec I_y$.

Suppose the algorithm returns *TRUE*. We shall prove that J is satisfiable if and only if J' is. Consider the following sequence of instances $J_0, J_1, \ldots, J_k$: Let $J_0 = J = (G, C)$. For each $i > 0$, if the module M_i is compliant in $J_{i-1} = (G_{i-1}, C_{i-1})$, then $J_i = (G_i, C_i)$ is constructed from J_{i-1} by contracting M_i to the vertex v'_i, i.e., $J_i = J_{i-1}[v'_i/M_i]$.

We claim that $k = n'$ and, moreover, for each i, M_i is compliant in every J_t, $t < i$. Suppose not. If $k < n'$, then M_k is not compliant in J_{k-1}. For $0 \leq i \leq k$, let t be the least index such that M_i is not compliant in J_t. Then there exist $u_1, u_2 \in M_i$ and $v \notin M_i$ s.t. the constraints $u_1 \prec v$ and $v \prec u_2$ are in C_t. Due to the minimality of t, either $t \geq 1$ and at least one of these constraints is not in C_{t-1}, or else $t = 0$. In the former case $v = v'_t$ and there exist $v_1, v_2 \in M_t$ s.t. the constraints $u_1 \prec v_1$ and $v_2 \prec u_2$ are in C_{t-1} and therefore in C_0 (because maximal type I modules cannot be nested). In the case $t = 0$, there exist $u_1, u_2 \in M_1$ and $v \notin M_1$ s.t. the constraints $u_1 \prec v$ and $v \prec u_2$ are in C_0. In both cases, step 3 of the algorithm must detect a contradicting pair of constraints, and return *FALSE*.

In particular, M_i is compliant also in J_{i-1}, thus J_i is well defined for all $i \leq n'$. Furthermore, $J[M_i]$ is satisfiable, since $G[M_i]$ is a clique and $C[M_i]$ is empty. Hence, by proposition 3.11, J_{i-1} is equivalent to J_i, and by induction over i, J_i is equivalent to $J = J_0$. In particular $J_{n'} = J'$ is equivalent to J.

Complexity: Step 1 requires $O(m+n)$ time, as shown by Hsu and Ma [19]. Step 2 takes $O(m+n)$ time, while step 3 requires $O(l)$ ■

3.3 Decomposing Type II Modules

We now describe an algorithm which, on a given instance J, with no type I modules, either decides that J is unsatisfiable, or generates a new set of instances $\mathbb{J} = \{J_1, \ldots, J_k\}$, with the following properties: (1) Each J_i contains no connected non-trivial modules. (2) $\sum_i Size(J_i) \leq Size(J)$. (3) J is equivalent to $\mathbb{J}$.

We first need some definitions: Let G be a graph without type I modules. let $M_1, \ldots, M_k$ be the connected non-singleton modules in G. Form a rooted tree T with nodes $n_1, \ldots n_k$ so that the arc $n_i n_j$ is in T if and only if M_j contains M_i as a maximal submodule. The root of T, denoted n_1, corresponds to V, i.e., $M_1 = V$. Let S_i be the set of vertices in M_i which do not belong to any proper submodule of M_i. We call S_i the ***skeleton*** of M_i. T is called a ***type II decomposition tree*** of G.

For each node n_i of T, we define a graph G_i as follows: If $n_{i_1}, \ldots n_{i_t}$ are the children of n_i in T, then G_i is the graph obtained from $G[M_i]$ by contracting each module to a single vertex v'_{i_j} for $1 \leq j \leq t$. Note that G_i contains also the original vertices of S_i. We say that v'_{i_j} is the ***representative*** of every vertex $x \in M_{i_j}$ in G_i, and that each vertex in S_i is its own ***representative*** in G_i. Finally, the ***lowest common ancestor*** of nodes n_i, n_j in T is the first node $x \in T$ which the paths from n_i and n_j to the root have in common. Note that every node above x has the same representative for vertices $x_i \in M_i$ and $x_j \in M_j$, though they have different representatives in x.

We are now ready to describe the algorithm:

Procedure Decompose-Modules-II($J, \mathbb{J}$)

Input: instances $J = (G, C)$ with no type I modules.

Output: either *FALSE*, or an equivalent set of instances $\mathbb{J} = \{J_1, \ldots, J_k\}$, where each $J_i = (G_i, C_i)$ has no type I and type II modules.

1. Construct the type II decomposition tree T for G, with nodes $n_1, \ldots, n_k$.
2. Construct $G_1, \ldots, G_k$. Set $C_i = \phi$, $i = 1, \ldots, k$.
3. For each constraint $v_i \prec v_j$ in C, do as follows: Let $v_i \in S_i$, $v_j \in S_j$. Let n_t be the lowest common ancestor of n_i and n_j in T, and let x and y be the representatives of n_i and n_j, respectively, in G_t. If C_t already contains the constraint $x \succ y$, then return *FALSE*. Otherwise, add the constraint $x \prec y$ to C_t, unless it is already there.

Theorem 3.13 *The algorithm terminates correctly in $O(l + m + n)$ time.*

Proof. Correctness: Suppose that the algorithm returns *TRUE*. Property (1) holds, since by definition each G_i contains no connected module. To prove property (2), consider the operation of contracting a single type II module M_i in G. The products of the contraction are $G[M]$ and $G[v/M]$. The total number of vertices in the resulting graphs is $n + 1$. On the other hand, the total number of edges must be at most $m-|M|+1$, since there must be a vertex in $V \setminus M$ which is adjacent to every vertex in M, by the assumption that G is connected. By repeating this argument for every node of T, we get that the total number of vertices and edges in $G_1, \ldots, G_k$ is at most $m+n$. Moreover, every constraint in C appears in at most one instance J_i. Hence, $\sum_j Size(J_i) \leq Size(J)$.

Let us now prove property (3): Consider a leaf n_r in the decomposition tree T. Since n_r is a leaf, $S_r = M_r$. Define J'_r to be the instance $(G_r, C_r = C[M_r])$, and $\tilde{J} = J[v/M]$. By proposition 3.11, J is equivalent to $\{J'_r, \tilde{J}\}$. By iterating this argument with $J = \tilde{J}$ we finally obtain that J is equivalent to $\{J'_1, \ldots, J'_k\}$. Moreover, $J'_i = J_i$, since every constraint between nodes in M_i appears in J_i. Hence J is equivalent to $\mathbb{J}$.

Suppose that the algorithm returns *FALSE*. Then there exist nodes n_i, n_j with lowest common ancestor n_t, with $x, y \in M_i$, $z, w \in M_j$, and $x \prec z$, $y \succ w$ in C, and $x \neq y$, or $z \neq w$. Suppose in the iterated leaf contraction argument above M_i was contracted before M_j, into a vertex v in G_t. Then in the corresponding instance $\tilde{J}$, v shatters M_j. Thus, M_j is not compliant and by corollary 3.8 J is not satisfiable.

Complexity: The complexity of constructing the type II modular decomposition tree is $O(m+n)$ [18]. Every G_i is isomorphic to a subgraph of G, induced by S_i and one representative of every child submodule of n_i. Hence, all G_i's can be computed from T in $O(m+n)$ time. The processing done for every constraint takes constant time, besides finding the lowest common ancestor of the vertices participating in it. This latter part can be achieved for all l lowest common ancestor queries, on a tree of size k, in $O(l+k)$ time [35]. Since $k \leq n$, the total complexity is $O(l+m+n)$. ■

3.4 Reducing Type III Modules

Our next algorithm handles disconnected modules. Let J be an instance with no connected (non-singleton) modules. Note that this implies that every module in J forms an independent set, by lemma 3.3. Either the algorithm decides that J is unsatisfiable, or it generates an equivalent instance $J' = (G', C')$ such that G' is prime, and the size of J' does not exceed the size of J. The algorithm and its correctness proof are very similar to those for type I modules.

Procedure Reduce-Modules-III(J, J')

Input: instance $J = (G, C)$ with no connected modules.

Output: either *FALSE*, or an equivalent instance $J' = (G', C')$, where G' is prime.

1. Find all maximal type III modules, $M_1, \ldots, M_k$. For each $v \in V(G)$, let $M_{i(v)}$ be the maximal type III module (perhaps a trivial one) which contains v.
2. For each M_i, if the order constraints in $C[M_i]$ contain a directed cycle, return *FALSE*.
3. Contract every M_i to a vertex v'_i, to obtain G'. Set $C' \leftarrow \emptyset$.
4. For each constraint $v \prec u$ s.t. $i(v) \neq i(u)$: If there is a constraint $v'_{i(v)} \succ v'_{i(u)}$ in C', then return *FALSE*. Otherwise add the constraint $v'_{i(v)} \prec v'_{i(u)}$ to C', unless it is already exists.

Note that in the above procedure $M_{i(v)}$ may be a trivial module: If $E = \emptyset$ then $M_{i(v)} = V$ for all v, and if $E \neq \emptyset$ and v is not contained in any bigger type III module, then $M_{i(v)} = \{v\}$.

Theorem 3.14 *The algorithm terminates correctly in* $O(l + m + n)$ *time.*

Proof. Correctness: Since for each module M_i the graph $G[M_i]$ contains no edges, the instance $J[M_i]$ is satisfiable if and only if there is a complete order on the vertices in M_i, satisfying all the constraints in $C[M_i]$. This holds if and only if $C[M_i]$ contains no directed cycle $x \prec y \prec \ldots \prec x$. Therefore if step 2 returns ***FALSE*** then for some i, $J[M_i]$ is not satisfiable, and so J is not satisfiable.

If all instances $J[M_i]$ are satisfiable, then the correctness proof is completely analogous to the proof of theorem 3.12.

Complexity: For each M_i, let n_i, m_i and l_i be the numbers of vertices, edges, and constraints in $(G[M_i], C[M_i])$, respectively. According to [19], all maximal type III modules can be found in $O(m + n)$ time. Creating $C[M_i]$ requires $O(l_i)$ time, and checking the acyclicity of the constraints (by depth-first search), takes $O(n_i + l_i)$. Since $\sum_i (l_i + n_i + m_i) \leq n + m + l$, the overall time complexity of this algorithm is $O(n + m + l)$. ■

3.5 Testing Prime Graphs

There is a one-to-one correspondence between consecutive orders of the maximal cliques in an interval graph G and transitive orientations of its complement $\bar{G}$. Since in a prime interval graph this order is unique, up to complete reversal [19], there can be only two possible orientations to $\bar{G}$, one the reverse of the other. In a prime graph, take the orientation induced by an arbitrary realization of G: to be satisfiable, either all constraints in C must follow that orientation, or all of them must oppose it. This justifies the following simple algorithm:

Procedure Solve-Prime(J)

Input: instance $J = (G, C)$ with prime G.

Output: either *FALSE*, or a legal exit if J has a realization.

1. Find a realization S of G.
2. Count the number p of order constraints satisfied by S.
3. If $0 < p < |C|$ return *FALSE*.

Theorem 3.15 *The algorithm terminates correctly in* $O(l + m + n)$ *time.*

Proof. Correctness follows from the discussion above. For the complexity: Step 1 takes $O(m + n)$ time, for example, by finding a consecutive order [6], and using it to construct the canonical realization. Clearly, step 2 takes $O(l)$ time. ■

3.6 The Complete Algorithm

We can now describe an overview of a linear time algorithm for $ISAT(\Delta_2)$. The algorithm terminates and returns *FALSE* if any of its procedures returns *FALSE*:

Algorithm ISAT-Δ_2(J)

Input: instance J.

Output: either *TRUE* or *FALSE*.

1. Reduce-Modules-I(J,J^1).
2. Decompose-Modules-II(J^1,$\{J_i^2\}_{i=1}^k$).
3. For each $1 \leq i \leq k$ do
4. Reduce-Modules-III(J_i^2, J_i^3).
5. Solve-Prime(J_i^3).
6. Return *TRUE*

The correctness and linearity of each step follow from theorems 3.12, 3.13, 3.14 and 3.15, and together we conclude:

Theorem 3.16 $ISAT(\Delta_2)$ *can be solved in* $O(m + n + l)$ *time.*

With a slight modification to the above algorithm, one can also produce a consistent realization for J in the same time complexity, if one exists.

4 Unit Interval Satisfiability

In this section we study UISAT(Δ) for each possible domain Δ. We use the names of domains given by Golumbic and Shamir [16], and Webber [38]. We refer the reader to Table 1 for some special cases of the problem that have been studied previously.

4.1 The domain $\Delta_1 = \{\cap, \prec, \succ, \prec\!\cap, \cap\!\succ, \prec\!\cap\!\succ\}$

In [16], it was shown that solving an instance J of ISAT(Δ_1) is equivalent to deciding whether a system of linear inequalities has a feasible solution. Each permitted relation (except $\prec\!\cap\!\succ$) is translated into one or two inequalities on the intervals endpoints, where the inequalities may be weak ($\leq$) or strict ($<$). We use that formulation, and add an equation $l_v + 1 = r_v$, for each $v \in V$, to force the solution to form a unit interval graph. Substituting for the r_v-s, we get the following system (P) of linear inequalities with the variables $\{l_v\}_{v \in V}$: Let $A = \{(x,y) | x \prec\!\cap y \textit{ or } x \cap y\}$ and $B = \{(x,y) | x \prec y\}$.

$$\begin{aligned} l_x - l_y &\leq 1 && (x,y) \in A \\ l_x - l_y &< -1 && (x,y) \in B \end{aligned} \tag{P}$$

Hence, (P) has a feasible solution if and only if J is satisfiable. Fix some positive $\epsilon < 1/n$, and define another system (Q) of weak inequalities with the same variables:

$$\begin{aligned} l_x - l_y &\leq 1 && (x,y) \in A \\ l_x - l_y &\leq -1 - \epsilon && (x,y) \in B \end{aligned} \tag{Q}$$

For a system (S) of inequalities of the form $l_x - l_y \leq c_{xy}$, define a directed weighted graph with vertices corresponding to variables and an arc of weight $w^S_{x,y} = c_{xy}$ from l_y to l_x. The resulting graph $G(S)$ is called the ***constraint graph*** of the system (S). It is well known that (S) has a feasible solution if and only if $G(S)$ contains no cycle of total negative weight (see, e.g., [1, p. 103]).

Lemma 4.1 *(P) has a feasible solution if and only if $G(Q)$ contains no negative cycle.*

Proof. If $G(Q)$ contains no negative cycle then by the above (Q) has a feasible solution, which is also a solution of (P). Suppose $G(Q)$ contains a negative cycle $C = (l_0, \ldots l_k, l_0)$ and that

(P) has a solution $\tilde{l_x}$. By our assumption $\sum_{i=0}^{k} w_{i,i+1}^{Q} < 0$ (indices are mod $k+1$). Every arc in that cycle corresponds to an inequality in (Q) and to a (weak or strict) inequality in (P) with the same variables. In the subsystem of inequalities corresponding to that cycle in (P), at least one inequality is strict, since otherwise all arcs have positive weight. Let n_A and n_B be the number of arcs in C from the sets A and B, respectively. Summing the inequalities in this subsystem, we get:

$$0 = \sum_{i=0}^{k} (\tilde{l}_i - \tilde{l}_{i+1}) < \sum_{\substack{0 \le i \le k \\ (i,i+1) \in A}} 1 - \sum_{\substack{0 \le i \le k \\ (i,i+1) \in B}} 1 = n_A - n_B$$

Hence, $n_A > n_B$ and $n_A \geq n_B + 1$. On the other hand,

$$0 > \sum_{i=0}^{k} w_{i,i+1}^{Q} = n_A - (1+\epsilon) n_B \geq n_A - n_B - n\epsilon > 0,$$

a contradiction. ■

Let $m = N^{\prec\cap} + N^{\prec} + 2N^{\cap}$ be the total number of inequalities in (P). One can detect if $G(Q)$ contains a negative cycle in $O(\sqrt{n}m \log n)$ time, using the algorithm of [29]. In summary:

Theorem 4.2 *$UISAT(\Delta_1)$ can be solved in $O(\sqrt{n}m \log n)$ time.* ■

4.2 The domain $\Delta_2 = \{\cap, \prec, \succ, \prec\!\!\succ\}$

Recall that a graph with four vertices is called $K_{1,3}$ if it contains a vertex adjacent to the other three, which form an independent set. The following classical characterization of unit interval graphs [32] will be used :

Theorem 4.3 *(Roberts, 1969) An interval graph is a unit interval graph if and only if it contains no induced $K_{1,3}$.*

Let J be an instance of $UISAT(\Delta_2)$. We shall use the linear algorithm for $ISAT(\Delta_2)$ which was described in section 3. Applying that algorithm to our instance J, if the algorithm returns ***FALSE***, then J has no interval realization, and in particular, 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. The following result is from [11, chapter 8]:

Theorem 4.4 *(Fishburn, 1985) Let $H = (V, <)$ be an interval order. H is a proper interval order if and only if its incomparability graph G does not contain an induced $K_{1,3}$.*

By the theorem, as we have already established that P is an interval order, it suffices to check if G contains an induced $K_{1,3}$. A fast way to check this is based on the following lemma:

Lemma 4.5 *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 adjacent vertices $v_i, v_j \in V$.*

Proof. ($\Rightarrow$) Suppose that G is not a unit interval graph. Then, by theorem 4.4 it contains an induced $K_{1,3}$ subgraph. Suppose, w.l.o.g., that $v_iv_j, v_iv_k, v_iv_l \in E$, while $\{v_j, v_k, v_l\}$ is an independent set in G. Then the intervals $[L_k, R_k], [L_j, R_j], [L_l, R_l]$ must be disjoint, and w.l.o.g, suppose that $R_k < L_j \leq R_j < L_l$. Since the interval $[L_i, R_i]$ intersects both $[L_k, R_k]$ and $[L_l, R_l]$, then $L_i \leq R_k < L_j \leq R_j < L_l \leq R_i$. In particular, $L_i < L_j$ and $R_i < R_j$.

($\Leftarrow$) Suppose that v_i and v_j are adjacent and that $L_i < L_j$ and $R_i > R_j$. Since C_{L_i} is a maximal clique, there exist a vertex $u \in C_{L_i} \setminus C_{L_j}$. Since C_{R_i} is a maximal clique, there exist a vertex $w \in C_{R_i} \setminus C_{R_j}$. Since $L_i < L_j \leq R_j < R_i$, and since the 1's in M are consecutive, $uw \notin E$, $uv_j \notin E$, $wv_j \notin E$. Since $u, v_i \in C_{L_i}$ we have $uv_i \in E$. Since $w, v_i \in C_{R_i}$ we have $wv_i \in E$. Hence, $\{u, w, v_i, v_j\}$ induce a $K_{1,3}$ in G, so G is not a unit interval graph. ∎

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_iv_j whether $L_i < L_j$ and $R_i > R_j$, takes $O(m)$ time for all edges. Therefore:

Theorem 4.6 *UISAT(Δ_2) can be solved in $O(n + N^{\cap} + N^{\prec})$ time.* ∎

4.3 The domain $\Delta_3 = \{\prec\succ, \prec, \succ, \prec\cap\succ\}$

Lemma 4.7 *An instance of $UISAT(\Delta_3)$ is satisfiable if and only if it is satisfiable as an instance of $ISAT(\Delta_3)$.*

Proof. The 'only if' direction is trivial. For the 'if', according to [16], if an instance J of $ISAT(\Delta_3)$ is satisfiable then it has an interval realization in which all events are disjoint. Such a realization is obviously proper, so J is also satisfiable as an instance of $UISAT(\Delta_3)$. ∎

The results of Golumbic and Shamir [16] on $ISAT(\Delta_3)$, together with the above observation imply:

Theorem 4.8 *$UISAT(\Delta_3)$ can be solved in $O(n + N^{\prec})$ time.* ■

4.4 The domain $\Delta_5 = \{\cap\succ, \prec\cap, \prec\succ\}$.

In this domain, for each two intervals $I = (l, r), I' = (l', r')$ the set of permitted relations between them is one of the following two (see figure 3):

- $I \prec\succ I'$, namely, either $r < l'$ or $r' < l$. We represent this relation by the two dashed arrows, as in figure 3(a). To satisfy the relation $I \prec\succ I'$, intervals I,I' must be placed so that one of the dashed arrows should head in the positive direction.

Figure 3: Representations of the permitted relations in ISAT(Δ_5). (a): $I \prec\succ I'$; (b): $I \prec\cap I'$

- $I \prec\cap I'$, namely, $l < r'$. The case $I \cap\succ I'$ is symmetric.

Webber [38] recently proved that $ISAT(\Delta_5)$ is NP-complete. We use here a different, more symmetric construction, inspired by Webber's reduction, and adapted to the additional constraints of unit-length intervals.

Lemma 4.9 *Given six intervals $\{I_j\}_{j=1}^{6}$, the chain of relations $I_1 \prec I_2 \prec\cap I_3 \prec I_4 \prec\cap I_5 \prec I_6 \prec\cap I_1$ is unsatisfiable.*

Proof. Let l_j, r_j be the endpoints of interval I_j. Then $r_1 < l_2 \leq r_3 < l_4 \leq r_5 < l_6 \leq r_1$, yielding a contradiction. ■

Theorem 4.10 *$UISAT(\Delta_5)$ is NP-complete.*

Proof. Clearly $UISAT(\Delta_5)$ is in NP. We will show a reduction to ISAT(Δ_5) from Not-All-Equal (NAE) Satisfiability restricted to positive formulas, which is defined as follows: The input to the problem is a boolean formula $\Phi = \wedge_{i=1}^{m} C^i$ in the set of variables $X = \{x^1, \ldots, x^n\}$. Each clause is of the form $C^i = y_0^i \vee y_1^i \vee y_2^i$, where y_j^i is a literal representing a variable in X for all i, j. Note that there are no negations of variables. The question is whether there exists a truth assignment to the variables, such that in every clause either one or two variables are true. Schaefer [34] has shown that this problem is NP-complete.

A simple gadget which we use repeatedly in the construction is a ***switch***: It consists of two events I_1, I_2 related as $I_1 \prec\!\!\succ I_2$ (see figure 4). The whole construction consists of $n + 3m + 1$ switches: One for each variable, three for each clause (one per literal), and one special (which we call ***pivot***) switch. Each switch and its two events will be indexed for identification. A ***variable component*** will be just a switch $V(x)$ for each variable $x \in X$.

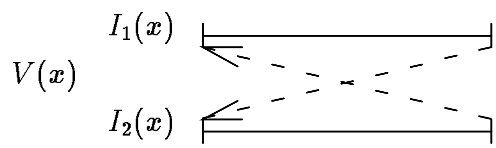

Figure 4: A switch which is a variable component

A ***clause component*** will consist of three switches, whose events are related in a particular way. Specifically, for each clause $C = y_0 \vee y_1 \vee y_2$, the clause component, $U(C)$ will consist of three switches $V(y_0), V(y_1), V(y_2)$, where the permitted relations between them will be (as in figure 5):

- $I_k(y_j) \prec\!\cap I_k(y_{j+1}), j = 0, 1, 2, k = 1, 2.$
- $I_k(y_j) \prec\!\cap I_{3-k}(y_{j-1}), j = 0, 1, 2, k = 1, 2.$

Here and throughout, indices are modulo 3.

The instance $P_\Phi = (V, D)$ will be constructed as follows:

1. Number variables arbitrarily. For each variable $x^j \in X$, introduce a variable component $V(x^j)$. For each j, j', k, k', where $j < j'$, set $I_k(x^j) \prec\!\cap I_{k'}(x^{j'})$.
2. Number clauses arbitrarily. For each clause $C^i \in C$, introduce a clause component $U(C^i)$. For each i, i', j, j', k, k', where $i < i'$ set $I_k(y_j^i) \prec\!\cap I_{k'}(y_{j'}^{i'})$.
3. Add a special switch $V(p)$, called the ***pivot***. For each literal y_j^i, and for each k, k', set $I_k(y_j^i) \prec\!\cap I_{k'}(p)$. For each variable $x^j \in X$, and for each k, k', set $I_k(x^j) \cap\!\succ I_{k'}(p)$.

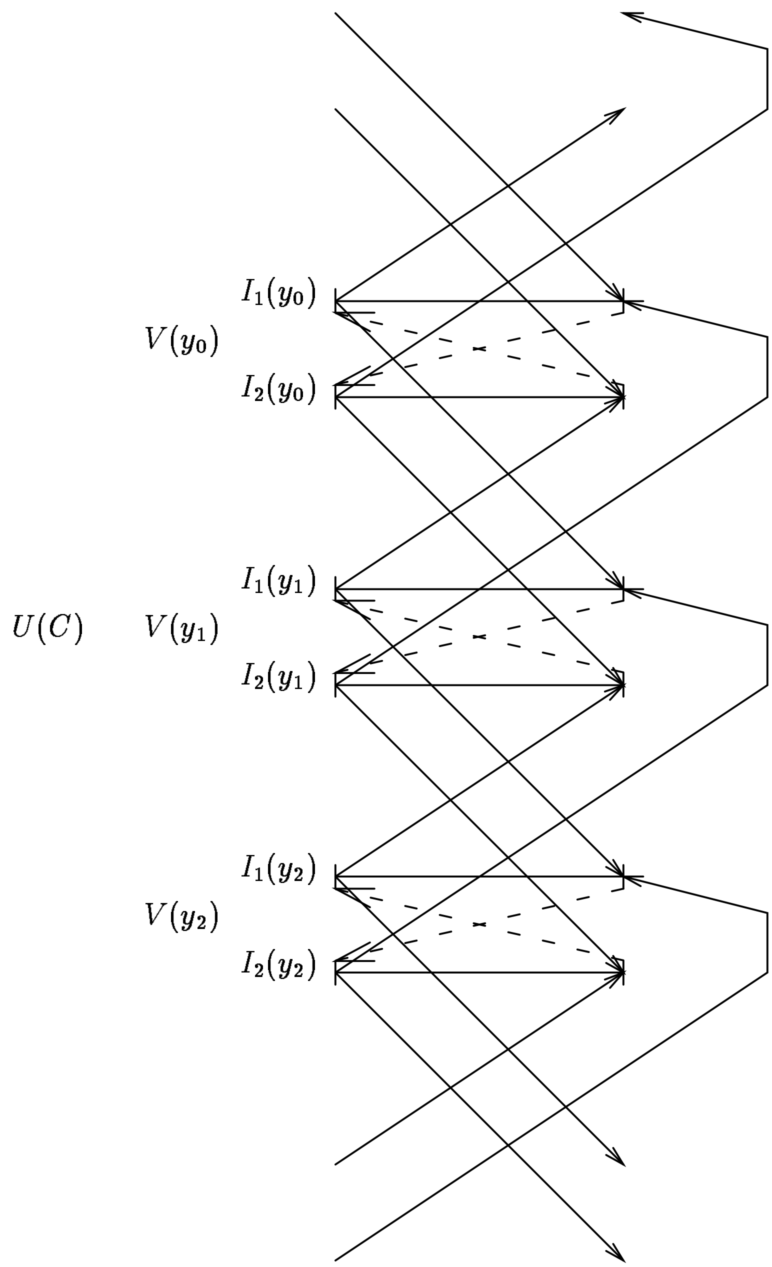

Figure 5: The clause component. Edges between the top and bottom switches wrap around.

4. For each clause $C^i = y_0 \vee y_1 \vee y_2$, where y_j represents the variable x^{m_j}, for each j, k, set $I_k(x^{m_j}) \prec\cap I_k(y_j)$.

5. For each two intervals $I_k(y^i_j)$ and $I_{k'}(x^{j'})$, that the permitted relations between them were not fixed in the previous stage, $I_k(y^i_j) \prec\cap I_{k'}(x^{j'})$.

The construction is clearly polynomial. We now prove that Φ has Not-All-Equal truth assignment if and only if P_Φ has a unit interval realization.

if: Suppose P_Φ has a unit interval realization S. Define a truth assignment $A : X \to \{T, F\}$ by setting $A(x^j) = T$ if and only if $I_1(x^j) \prec I_2(x^j)$. As the intervals of each switch are disjoint, A is well-defined. Let $I_L(p)$ and $I_R(p)$ be the left and right intervals in the realization of $V(p)$, respectively. We shall prove that A is NAE on every clause C.

By contrary, assume that for $C = y_0 \vee y_1 \vee y_2$, where y_j represents the variable x^{m_j}, and $A_C \equiv A(x^{m_0}) = A(x^{m_1}) = A(x^{m_2})$. If $A_C = T$, by definition of A, $I_1(x^{m_j}) \prec I_2(x^{m_j})$ for each $j = 0, 1, 2$. If for some j, $I_1(y_j) \succ I_2(y_j)$, then: $I_1(x^{m_j}) \prec I_2(x^{m_j}) \prec\cap I_2(y_j) \prec I_1(y_j) \prec\cap I_L(p) \prec I_R(p) \prec\cap I_1(x^{m_j})$ yielding a contradiction according to lemma 4.9. Hence, $I_1(y_j) \prec I_2(y_j)$, for each $j = 0, 1, 2$, and: $I_1(y_0) \prec I_2(y_0) \prec\cap I_1(y_2) \prec I_2(y_2) \prec\cap I_1(y_1) \prec I_2(y_1) \prec\cap I_1(y_0)$ a contradiction. The case $A_C = F$ is similar (simply exchange throughout I_1 and I_2 above).

only if: Suppose Φ is satisfiable by a NAE-assignment A. We will describe a unit interval realization S for P_Φ:

Choose positive values $\epsilon_1, \epsilon_2, \epsilon_3$, where $\epsilon_1 < \frac{1}{2(MAX(m,n)+2)}$, $\epsilon_2 < \frac{\epsilon_1}{6}$, and $\epsilon_3 < \frac{\epsilon_2}{2}$.

We first define a real value $place(x)$ for each switch $V(x)$ in our instance:

- For each variable x^j set: $place(x^j) \stackrel{def}{=} j\epsilon_1$.
- For each clause C^i, set $base(C^i) \stackrel{def}{=} -\frac{1}{2} + i\epsilon_1$. Let $C^i = y^i_0 \vee y^i_1 \vee y^i_2$, and let $first(i) \in \{0, 1, 2\}$ be an index such that $A(y^i_{first(i)-1}) = A(y^i_{first(i)})$. There exists exactly one such index, since A is NAE. Set, for $j = 0, 1, 2$, $place(y^i_{first(i)+j}) \stackrel{def}{=} base(C^i) + j\epsilon_2$. This definition assures that the literal assigned to the middle *place* has different truth value than the other two.
- Set $place(p) \stackrel{def}{=} 0$.

It is easy to see that for each x, $-\frac{1}{2} < place(x) < \frac{1}{2} - \epsilon_1$. For each switch $V(x)$, we define two unit intervals at distance ϵ_3 from $place(x)$, on each of its sides: Denote $IRIGHT(x) \stackrel{def}{=}$

$[place(x)+\epsilon_3, place(x)+\epsilon_3+1]$, $ILEFT(x) \stackrel{def}{=} [place(x)-\epsilon_3-1, place(x)-\epsilon_3]$. Extend the assignment A to $X \cup Y \cup \{p\}$ by setting $A(p) = \boldsymbol{TRUE}$, and $A(y^i_j) = A(x)$ if y^i_j represents x.

We are finally ready to give the realization: For each switch x, if $A(x)$ is $TRUE$, then $I_1(x) \stackrel{def}{=} ILEFT(x)$ and $I_2(x) \stackrel{def}{=} IRIGHT(x)$. Otherwise, then $I_1(x) \stackrel{def}{=} IRIGHT(x)$ and $I_2(x) \stackrel{def}{=} ILEFT(x)$.

We have to prove now that we have generated a unit interval realization of P_Φ. As all intervals were defined to be of unit length, we only need to verify that the realization satisfies all the relations.

Consider two intervals $I, I' \in S$:

- If I, I' belong to the same switch $V(x)$, then they are assigned to the disjoint intervals $ILEFT(x)$ and $IRIGHT(x)$.
- If $I = I_k(x^j)$, $I' = I_{k'}(p)$ then:
  $l(I') \leq \epsilon_3$ and $r(I) \geq \epsilon_1 - \epsilon_3 > \epsilon_3$ so $I' \prec\cap I$.
- If $I = I_k(y^i_j)$, $I' = I_{k'}(p)$ then:
  $r(I') \geq -\epsilon_3$ and $l(I) \leq -\frac{1}{2} + m\epsilon_1 + 2\epsilon_2 + \epsilon_3 < -\epsilon_3$ so $I' \cap\succ I$.
- If $I = I_k(x^j)$, $I' = I_{k'}(x^{j'})$ for $j < j'$ then:
  $place(x^{j'}) - place(x^j) \geq \epsilon_1$ therefore $r(I') - l(I) \geq \epsilon_1 - 2\epsilon_3 > 0$ so $I' \cap\succ I$.
- If $I = I_k(y^i_j)$, $I' = I_{k'}(y^{i'}_{j'})$ for $i < i'$ then:
  $base(C^{i'}) - base(C^i) \geq \epsilon_1$ therefore $r(I') - l(I) \geq \epsilon_1 - 2\epsilon_2 - 2\epsilon_3 > 0$ so $I' \cap\succ I$.
- Suppose I, I' belong to the same clause component, i.e., $I = I_k(y^i_j)$, $I' = I_{k'}(y^i_{j'})$, $j \neq j'$. The clause intervals are assigned so that the middle *place* position is given to the variable y^i_t whose truth value is different from that of the other two. The left and right positions are given to the preceding variable in the clause, y^i_{t-1} and to the succeeding variable in the clause, y^i_{t+1}, respectively (as usual, indices are modulo 3). Since the clause component is invariant with respect to cyclic renaming of its switches, it suffices to verify that all relations hold when $A(y^i_0) = A(y^i_2) \neq A(y^i_1)$, for the two possible truth values of $A(y^i_1)$. This is easily done by inspection (see figure 6).
- If $I = I_k(y^i_j)$, $I' = I_k(x)$ where y^i_j represents x, then, since $A(y^i_j) = A(x)$, either both $I = IRIGHT(y^i_j)$ and $I' = IRIGHT(x^{m^i_j})$ and both contain $\frac{1}{2}$, or both $I = ILEFT(y^i_j)$ and $I' = ILEFT(x^{m^i_j})$ and both contain $-\frac{1}{2}$. In both cases $I' \prec\cap I$.

- If $I = I_k(y^i_j)$, $I' = I_{k'}(x^{j'})$, where y^i_j does not represent $x^{j'}$, then $r(I') \geq 0$ and $l(I) \leq 0$ so $I' {\cap}{\succ} I$.

$\blacksquare$

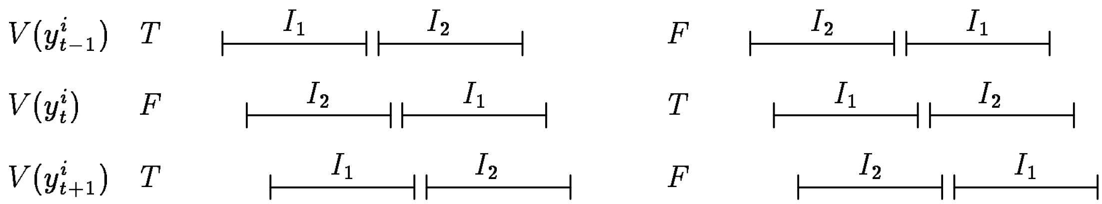

Figure 6: Possible truth assignments and relative positions of the intervals in a clause component.

4.5 Conclusion

For every domain Δ, which is a set of disjunctions of the atomic relations $\prec, \succ, \cap$, one of the following holds:

- Δ is contained in one of $\Delta_1, \Delta_2, \Delta_3$. In this case UISAT($\Delta$) is clearly polynomial.
- Δ contains one of Δ_0, Δ_5. In this case UISAT(Δ) is clearly NP-hard.

Thus, we have resolved the complexity of UISAT(Δ) for all possible domains Δ. Figure 7 summarizes the complexity in all domains of UISAT.

In summary, there are 21 polynomial domains and 10 NP-complete ones. Interestingly, the resulting classification of the domains is identical to that done by [16] and [38] on ISAT problems. As noted by [16], one of the byproducts of this classification is the ability to speed up the 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.

In practical applications it often happens that the lengths of the intervals are only roughly equal. This situation raises new interesting interval satisfiability problems similar to the ones discussed in this paper, in which intervals lengths vary between $1-\alpha$ and $1+\alpha$, for some small fixed α.

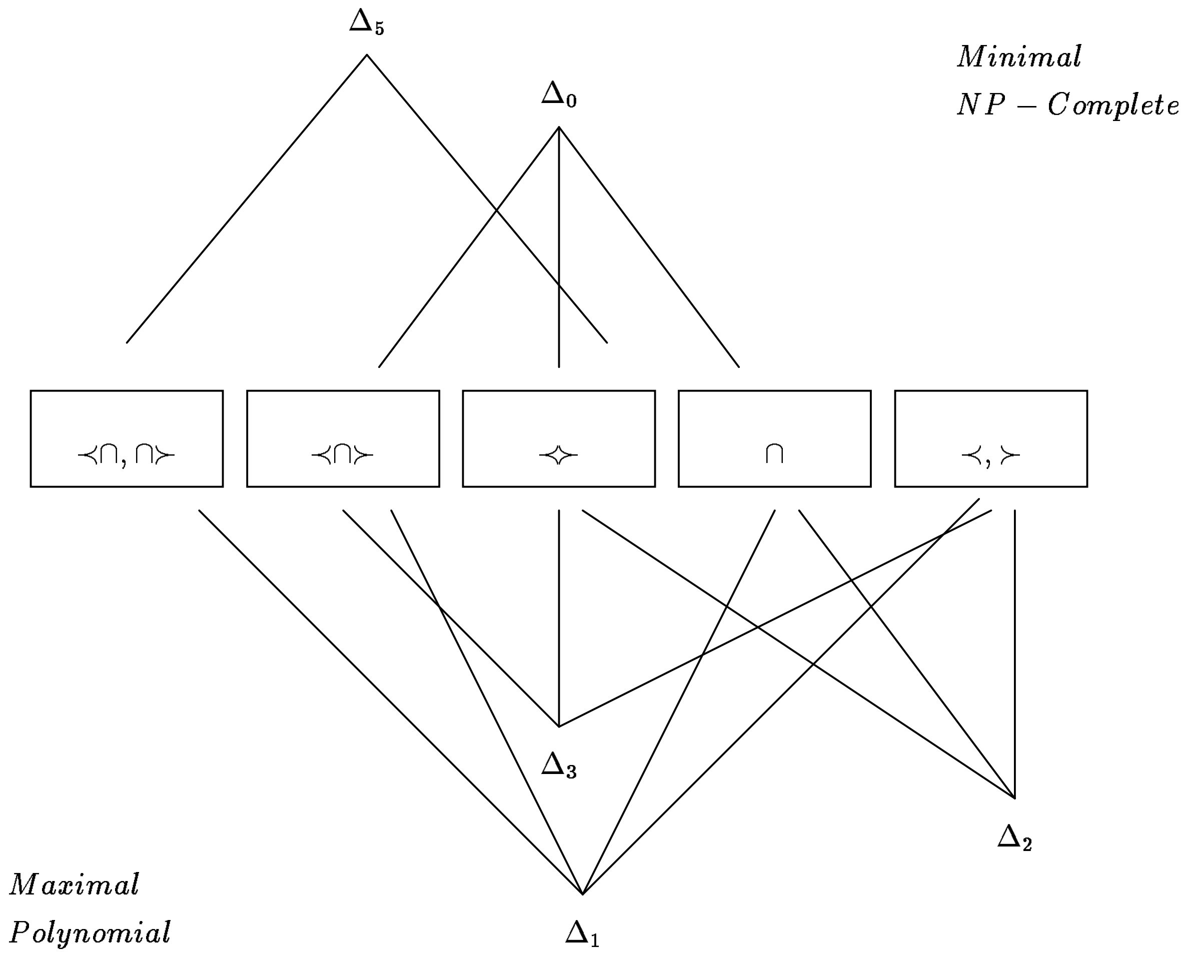

Figure 7: The complexity of unit interval satisfiability of different domains. The minimal NP-complete domains are marked at the upper side. The maximal polynomial domains are marked at the lower side. Every other domain is either a subset of one of Δ_1, Δ_2, Δ_3, and hence polynomial, or a superset of Δ_0 or Δ_5, and hence NP-complete.

Acknowledgments

We thank the referees of this paper for their helpful remarks.

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] P. Baldy and M. Morvan. A linear time and space algorithm to recognize interval orders. *Discrete Applied Mathematics*, 46:173–178, 1993.

[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, New York, 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. Intransitive indifference with unequal indifference intervals. *J. Math. Psych.*, 7:144–149, 1970.

[11] P. Fishburn. *Interval Orders and Interval Graphs.* Wiley, New York, 1985.

[12] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15:835–855, 1965.

[13] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Canad. J. Math.*, 16:539–548, 1964.

[14] P. W. Goldberg, M. C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. *Journal of Computational Biology*, 2(1):139–152, 1995.

[15] M. C. Golumbic. Interval graphs and related topics. *Discrete Math.*, 55:113–121, 1985.

[16] M. C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *J. ACM*, 40:1108–1133, 1993.

[17] G. Hajös. Über eine art von graphen. *Intern. Math. Nachr.*, 11, 1957. problem 65.

[18] W.-L. Hsu. A simple test for interval graphs. In W-L. Hsu and R. C. T. Lee, editors, *Proc. 18th Int. Workshop (WG '92), Graph-Theoretic Concepts in Computer Science*, pages 11–16. Springer-Verlag, 1992. LNCS 657.

[19] 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.

[20] H. Kaplan and R. Shamir. Pathwidth, bandwidth and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25(3):540–561, 1996.

[21] R. M. Karp. Mapping the genome: some combinatorial problems arising in molecular biology. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 278–285. ACM Press, New York, 1993.

[22] D. G. Kendall. Incidence matrices, interval graphs, and seriation in archaeology. *Pacific J. Math.*, 28:565–570, 1969.

[23] D. G. Kendall. Some problems and methods in statistical archaeology. *World Archaeol.*, 1:68–76, 1969.

[24] N. Korte and R. H. Möhring. Transitive orientation of graphs with side constraints. In H. Noltemeier, editor, *Proc. of the International workshop on Graphtheoretic concepts in Computer Science (WG '85)*, pages 143–160, Linz, 1985. Universitätsverlag Rudolf Trauner.

[25] N. Korte and R. H. Möhring. An incremental linear time algorithm for recognizing interval graphs. *SIAM J. Computing*, 18:68–81, 1989.

[26] P. Lichter et al. High-resolution mapping of human chromosome 11 by in situ hybridization with cosmid clones. *Science*, 247:64–69, 1990.

[27] R. H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. In I. Rival, editor, *Graphs and Order*, pages 41–101. Reidel, Dordrecht, 1985.

[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. Computing*, 8:405–409, 1979.

[31] I. Pe'er and R. Shamir. Interval graphs with side (and size) constraints. In *Proc. of the Third Annual European Symp. on Algorithms, (ESA 95) Corfu, Greece*, pages 142–154. Springer, 1995.

[32] F. S. Roberts. Indifference graphs. In F. Harary, editor, *Proof Techniques in Graph Theory*, pages 139–146. Academic Press, New York, 1969.

[33] F. S. Roberts. *Discrete Mathematical Models, with Applications to Social Biological and Environmental Problems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[34] T. J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symp. on Theory of Computing*, pages 216–226, 1978.

[35] B. Schieber and U. Vishkin. On finding lowest common ancestors: Simplifiction and parallelization. *SIAM J. Computing*, 17:1253–1262, 1988.

[36] S. A. Ward and R. H. Halstead. *Computation Structures.* MIT Press, 1990.

[37] J.D. Watson, M. Gilman, J. Witkowski, and M. Zoller. *Recombinant DNA.* W.H. Freeman, New York, 2nd edition, 1992.

[38] A.B. Webber. Proof of an interval satisfiability conjecture. manuscript, Wesstern Illinois University, 1994.