Maximal Neighborhood Search and Rigid Interval Graphs

Article *in* Journal of Graph Algorithms and Applications · January 2013

DOI: 10.7155/jgaa.00293

CITATIONS
2

READS
150

2 authors, including:

Yaokun Wu
Shanghai Jiao Tong University
56 PUBLICATIONS **353** CITATIONS

SEE PROFILE

MAXIMAL NEIGHBORHOOD SEARCH AND RIGID INTERVAL GRAPHS

PENG LI[†] AND YAOKUN WU[†‡]

Abstract. A rigid interval graph is an interval graph which has only one clique tree. In 2009, Panda and Das show that all connected unit interval graphs are rigid interval graphs. Generalizing the two classic graph search algorithms, Lexicographic Breadth-First Search (`LBFS`) and Maximum Cardinality Search (`MCS`), Corneil and Krueger propose in 2008 the so-called Maximal Neighborhood Search (`MNS`) and show that one sweep of `MNS` is enough to recognize chordal graphs. We develop the `MNS` properties of rigid interval graphs and characterize this graph class in several different ways. This allows us obtain linear time 4-sweep and 3-sweep `MNS` algorithms for recognizing rigid interval graphs and unit interval graphs, respectively, generalizing a corresponding 3-sweep `LBFS` algorithm for unit interval graph recognition designed by Corneil in 2004. For unit interval graphs, we even present a new linear time 2-sweep `MNS` certifying recognition algorithm.

Key words. certifying algorithm, clique path, clique tree, interval graph, Lexicographic Breadth-First Search, Maximal Neighborhood Search, `MNS` ordering, `MNS` type algorithm, recognition algorithm, rigid interval graph, unit interval graph

AMS subject classifications. 05C62, 05C85, 68R10, 68W40

1. Introduction.

All graphs considered in this paper are finite, nonempty, simple and undirected. For any nonnegative integer n, we use the notation $[n]$ for the set of first n positive integers. Let $\mathcal{S}$ be a family of sets $S_1, \ldots, S_n$. The *intersection graph* of $\mathcal{S}$ is the graph G with $[n]$ as vertex set and i and j are adjacent if and only if $i \neq j$ and $S_i \cap S_j \neq \emptyset$. We call $\mathcal{S}$ an *intersection model* of G provided G is the intersection graph of $\mathcal{S}$. It is known that every graph has an intersection model [67, Theorem 1.1]. A *unit interval graph* (UIG) is the intersection graph of a set of intervals of the same length on the real line. An *interval graph* is the intersection graph of a set of intervals on the real line. A *chordal graph* is the intersection graph of a set of subtrees of a common host tree. It is apparent that {unit interval graphs} $\subseteq$ {interval graphs} $\subseteq$ {chordal graphs}. One usual task in algorithmic graph theory [11, 34, 67, 82] is to do some data mining to tell if a given graph can arise from an intersection model of some specific form (recognition problem) and even reconstruct an intersection model of some required form when such a model exists (representation problem). Especially, there have been intense study on recognizing/representing unit interval graphs [4, 15, 18, 26, 28, 39, 40, 41, 42, 49, 52, 61, 68, 70], interval graphs [8, 24, 25, 36, 44, 45, 46, 47, 48, 50, 53, 54, 57, 74], chordal graphs [79, 81, 85, 86] and some other important graph classes.

Interval graphs and unit interval graphs model the overlapping structure of a set of intervals of the real line. They are related to lots of applications involving overlapping and consecutiveness and are simple and beautiful enough to garner a good deal of attention which leads to the discovery of endlessly many nice combinatorial properties of them [1, 2, 6, 29, 33, 34, 35, 37, 62, 64, 69, 72, 75]. Accordingly, many recognition algorithms for interval graphs are developed one after another. Most of them involve some cleverly designed data structure, say PQ tree [8], PC tree [48], MPQ tree [53], modular decomposition [44, 45, 47], manufacture of a clique path from a clique tree [36], etc.. Many algorithms for recognizing unit interval graphs

[†]Department of Mathematics, Shanghai Jiao Tong University, Shanghai, 200240, China.
[‡]Corresponding author (`ykwu@sjtu.edu.cn`).

are either based on interval graph recognition [61] or need some procedure of sorting vertices according to their degrees [18, 26, 28].

Graph searching [20] is fundamental for graph algorithms, including graph recognition algorithms. Before going further, let us expound a bit on the multi-sweep graph search algorithms.

For any set S of cardinality n, an *ordering* of S is just a bijection from $[n]$ to S and two ordering σ and σ' of S are said to be *reversal* of each other if $\sigma(i) = \sigma'(n+1-i)$ for any $i \in [n]$. Let G be a graph on n vertices. A *selection rule* $\mathfrak{S}$ is a map from $2^{V(G)} \setminus \{V(G)\}$ to $2^{V(G)} \setminus \{\emptyset\}$ such that $T \cap \mathfrak{S}(T) = \emptyset$ for any $T \subsetneq V(G)$. A *graph search algorithm* with selection rule $\mathfrak{S}$ determines an ordering τ of $V(G)$ by selecting $\tau(1) \in \mathfrak{S}(\emptyset)$ and then going on inductively to pick $\tau(i+1)$ from $\mathfrak{S}(\{\tau(1), \ldots, \tau(i)\})$ for any $i \in [n-1]$. We call $\mathfrak{S}(\{\tau(1), \ldots, \tau(i)\})$ a *slice at time* $i+1$ in the process of generating the ordering τ with the selection rule $\mathfrak{S}$, or merely a slice of τ, and denote it by the simplified notation $S_\tau(i+1)$, hoping that the selection rule is clear from context. Note that many different orderings may be generated by the same selection rule as there are many ways of breaking ties by choosing one element from a given slice. If $\mathfrak{S}'$ is a new selection rule for $V(G)$ satisfying $\mathfrak{S}'(T) \subseteq \mathfrak{S}(T)$ for any $T \in 2^{V(G)} \setminus \{V(G)\}$, we may think that we are introducing additional rule of breaking ties into the original search algorithm corresponding to $\mathfrak{S}$ and then restrict the output vertex ordering to a smaller range. For any graph search algorithm $\mathcal{A}$ with a selection rule $\mathfrak{S}$, any graph G and $u \in V(G)$ satisfying $u \in \mathfrak{S}(\emptyset)$, we write $\mathcal{A}(G, u)$ for the search algorithm applied to G with a selection rule $\mathfrak{S}'$ such that $\mathfrak{S}'(\emptyset) = \{u\}$ and $\mathfrak{S}' = \mathfrak{S}$ elsewhere.

A *multi-sweep graph search algorithm* generates several orderings $\tau_1, \tau_2, \ldots$ of the vertex set of a graph in turn with selection rules $\mathfrak{S}_1, \mathfrak{S}_2, \ldots$ in each sweep respectively. Usually, the rule $\mathfrak{S}_i$ will rely on the previous orderings $\tau_1, \ldots, \tau_{i-1}$. It is expected that these orderings will be better and better in some sense and the final ordering will provide us what we want, say a good certificate for either the membership or the nonmembership of the graph in certain graph class.

Generalizing both the Maximal Cardinality Search (`MCS`) algorithm [84, 85] and the Lexicographic Breadth-First Search (`LBFS`) algorithm [79], Corneil and Krueger propose the *Maximal Neighborhood Search* (`MNS`) algorithm [20] as a greedy algorithm for displaying the tree structure of graphs. The idea behind `MNS` is quite natural: at each step, pick a vertex whose set of neighbors already explored is maximal with respect to set inclusion. We give below the psudocode for the algorithm `MNS`(G, u), which is the generic `MNS` algorithm that chooses the specific vertex u as the starting vertex.

```
MNS(G, u)
 1  ▷ Input a graph G and a vertex u of G
 2  ▷ Output an ordering τ of V(G)
 3  label[u] ← {u}
 4  for v ∈ V(G) \ {u}
 5      do label[v] ← ∅
 6  i ← 1
 7  while exists an unvisited vertex
 8      do pick an unvisited vertex v such that label[v] is maximal under set inclusion
 9         set v as a visited vertex
10         τ(i) ← v
```

```
11          i ← i + 1
12          for each unvisited vertex w in N_G(v)
13              do label[w] ← label[w] ∪ {v}
14  return τ
```

Any ordering τ produced by the MNS algorithm is called an MNS *ordering*. A graph search algorithm is an MNS *type algorithm* if all possible output vertex orderings of the algorithm are MNS orderings. Note that both LBFS [34, 36, 79] and MCS [84, 85] are MNS type algorithms and have simple linear time implementations. Another quite useful and easily implementable MNS type algorithm is the so-called Lexicographic Depth-First Search (LDFS) [20], which can run with a log factor off linear [55] (a more complicated version has a loglog factor [17, 83]). A generalization of MCS, called Lexicographic Maximum Cardinality Search (LMCS), is also an MNS type algorithm and is proposed to study chordal powers of graphs [9]. Surely, MNS itself is an MNS type algorithm. It worths mentioning that LBFS is a special breadth-first search while MNS may not be any breadth-first search.

For the purpose of devising certain multi-sweep LBFS algorithms, Ma [63] and Simon [80] independently propose a special kind of LBFS, called LBFS+. In the same fashion, for any graph search algorithm $\mathcal{A}$ with selection rule $\mathfrak{S}$, let us propose here the algorithm $\mathcal{A}+$, which is adapted from $\mathcal{A}$ by introducing some tie-breaking rule according to a given vertex ordering. Note that each output of $\mathcal{A}+(G,\tau)$ must be an output of $\mathcal{A}(G,\tau(|V(G)|))$.

$\mathcal{A}+(G,\tau)$:
{ Input: a graph G and an ordering τ of $V(G)$;
Output : an ordering σ of $V(G)$.}
For any $i \in [|V(G)|]$, after $\sigma(1),\ldots,\sigma(i-1)$ are determined, choose $\sigma(i)$ to be the vertex in $\mathfrak{S}(\{\sigma(1),\ldots,\sigma(i-1)\})$ that appears last in τ.

We note that LDFS+ can be implemented in the same time as LDFS [17]. Additionally, LBFS+ [15, 36] can be executed in linear time as with LBFS itself.

In 2004, Corneil [15] presents a 3-sweep LBFS algorithm for the recognition of unit interval graphs and argues that it is the most easily implementable unit interval graph recognition algorithm thus known. In 2009, Corneil, Olariu and Stewart [22, 24] establish a huge theory of the LBFS properties of interval graphs, based on which they employ six passes of Lexicographic Breadth-First Search with certain tie-breaking rules to get a very simple recognition algorithm for general interval graphs. Referring to this 6-sweep LBFS algorithm, Spinrad gives the following interesting comments [82, p. 194]:

> The word 'simple' has many different meanings when applied to an algorithm, some of which are illustrated nicely in this case. I believe that the algorithm below would be quite simple to implement, and I include it for that reason. However, the correctness proof is extraordinarily long (much longer than the proof of the algorithm based on PQ-trees), ... Examples like this show us that the God of graph theory may be different from the God of graph algorithms; which would you include in a justification of the statement that interval graphs can be recognized in linear time?

As with the multi-sweep graph search recognition of (unit) interval graphs, it is largely believed that LBFS is the correct graph traversal algorithm to be used. When discussing their interval graph recognition algorithm, Korte and Möhring [53, p. 74] make the comments that "BFS together with lexicographical tie breaking are essential for these results". Also, after establishing some structural theorem, Simon [81] asserts that "Probably this theorem explains why lexBFS has to be preferred to maximum cardinality search in some applications like recognizing interval graphs".

Following the quite elegant 3-sweep LBFS algorithm devised by Corneil [15], maybe a bit surprisingly due to the comments in last paragraph, this paper will show that there is a 3-sweep MNS algorithm for recognizing unit interval graphs, in which any MNS type algorithm can be adopted. Note that our algorithm basically only uses $\mathcal{A}$ and $\mathcal{A}+$ for some MNS type algorithm $\mathcal{A}$ and involves no additional processing procedures. The key to get to such an algorithm is the formulation of the concept of rigid interval graphs (see §2.3), a graph class lying between unit interval graphs and interval graphs, and a realization of the fact that many earlier results on unit interval graphs should be understood as results for rigid interval graphs. The main part of this paper is to develop structural and algorithmic properties of rigid interval graphs, especially its properties related to MNS type algorithms. We will propose a simple 4-sweep MNS algorithm for recognizing rigid interval graphs, along the same line of developing the 3-sweep MNS algorithm for recognizing unit interval graphs. As we will summarize in §5, after putting the original Corneil's 3-sweep LBFS algorithm for recognizing unit interval graphs into the framework of multi-sweep MNS algorithms, we will gain quite intuitive understanding of the role of each sweep in the recognition algorithm.

Building on some deep results from [24], we are able to design a simple linear time 4-sweep LBFS algorithm for recognizing interval graphs and will report it elsewhere [57]. It is very unlikely that there exists a corresponding multi-sweep MNS algorithm for recognizing general interval graphs. Compared with the work in [24, 57] on establishing multi-sweep LBFS algorithms for recognizing general interval graphs, our algorithms for recognizing rigid/unit interval graphs in this paper can work with much wider MNS type algorithms, the algorithm analysis is much easier and each step of the algorithm can be much more transparent to the users – all these are mainly due to the intrinsic simplicity of the rigid interval graphs from the viewpoint of clique tree representation.

For the practical use of an algorithm, two important issues are how easy it is to implement the algorithm and how easy the user can be sure of the correctness of the output of the algorithm without having to trust the algorithm and the implementation of the algorithm. We have spent some time discussing the first issue. For the second issue, we should mention the so-called *certifying algorithm* [7, 66], namely an algorithm whose output contains not only an answer of acceptance/rejection but also a certificate or witness for the user to easily convince himself that the particular output is correct or something buggy happens in the implementation or design of the algorithm. As mentioned in [54], "the certificate is useful because it allows trust in a well-known theorem to be substituted for trust in a program of dubious origin".

The only known linear time certifying recognition algorithm for interval graphs is reported by Kratsch et al. [54]. It uses as a subroutine the recognition algorithm of Korte and Möhring [53], which in turn applies the MPQ tree data structure as mentioned above. For unit interval graphs, some simpler certifying algorithms are known. Meister proposes a special breadth-first search, called min-LexBFS, and design a certifying recognition algorithm for unit interval graph which basically consists of

three sweeps of `min-LexBFS` [68, Theorem 16]. Note that `min-LexBFS` is incomparable with `LBFS` and even incomparable with `MNS`. Hell and Huang [41] modify Corneil's 3-sweep `LBFS` algorithm for recognizing unit interval graphs into a certifying algorithm. Since we generalize the 3-sweep `LBFS` algorithm of Corneil [15], it is no strange that we also get a certifying recognition algorithm for unit interval graphs – indeed our certifying algorithm will comprise of two passes of `MNS` and one additional sweep to search for a certificate of membership/nonmenbership following the output ordering of the second `MNS` sweep. It may worth pointing out that the nonmembership certificate of our algorithm is the same with that of the algorithm of Meister [68] and is different with that used in the algorithm of Hell and Huang [41].

The paper proceeds as follows. In §2, we prepare some background for our research as well as some preliminary work and notation, including the introduction of rigid interval graph, the key object of this paper. The 2-sweep `MNS` certifying recognition algorithm for unit interval graphs, as well as an interesting by-product of it (Corollary 3.12), are presented in §3. We demonstrate in §4 how to use simple sweeps of `MNS` to recognize unit interval graphs and rigid interval graphs. What come to our aid for this purpose will be some characterizations of rigid interval graphs. Note that at the end of §4 we indicate a connection between the work in §3 and §4. The last section, §5, concludes the paper with some remarks.

2. Background and notations. Let G be a graph. A sequence of distinct vertices $v_1, v_2, \ldots, v_\ell$ of G is a *path of length* $\ell - 1$, or an $(\ell-1)$*-path*, in G if each two consecutive vertices in the sequence form an edge of the graph and we denote this path by $[v_1, \ldots, v_\ell]$. We also call the path $[v_1, \ldots, v_\ell]$ a path connecting v_1 and v_ℓ, or simply a v_1, v_ℓ-path. A cyclic sequence of distinct vertices $v_1, v_2, \ldots, v_\ell$ is a *cycle of length* ℓ, or an ℓ*-cycle*, in G if each two consecutive vertices in the cyclic sequence form an edge, noting that v_1 and v_ℓ is regarded as consecutive, and we denote this cycle by $\langle v_1, \ldots, v_\ell \rangle$. A path/cycle is *chordless* if nonconsecutive vertices on the path/cycle are not adjacent in the graph. The *neighbors* of a vertex v are all those vertices which are adjacent to v in G. We reserve the notation $N_G(v)$ for this set of neighbors of v and write $N_G[v]$ for $N_G(v) \cup \{v\}$. The set $N_G[v]$ is often called the *closed neighborhood* of the vertex v in G.

An *interval representation* $\mathcal{I}$ of a graph G consists of two mappings $\ell_\mathcal{I}$ and $r_\mathcal{I}$ from $V(G)$ to the reals such that $\ell_\mathcal{I}(v) \leq r_\mathcal{I}(v)$ for any $v \in V(G)$ and $[\ell_\mathcal{I}(v), r_\mathcal{I}(v)] \cap [\ell_\mathcal{I}(w), r_\mathcal{I}(w)] \neq \emptyset$ if and only if $vw \in E(G)$. If $r_\mathcal{I} - \ell_\mathcal{I}$ takes the constant value 1, the interval representation $\mathcal{I}$ is named a *unit interval representation*. An interval representation $\mathcal{I}$ of G can also be thought of as a map from $V(G)$ to the intervals of reals with $\mathcal{I}(v) = [\ell_\mathcal{I}(v), r_\mathcal{I}(v)]$ for each $v \in V(G)$. An interval representation $\mathcal{I}$ is *proper* if different vertices are mapped by $\mathcal{I}$ to intervals that no one contains the other. Given an interval representation $\mathcal{I}$ of G, there is naturally a partial order on $V(G)$, denoted $<_\mathcal{I}$, such that for any $u, v \in V(G)$ we have $u <_\mathcal{I} v$ if and only if $r_\mathcal{I}(u) < \ell_\mathcal{I}(v)$. More generally, for any two subsets M and N of $V(G)$, we write $M <_\mathcal{I} N$ if $u <_\mathcal{I} v$ for any $u \in M$ and $v \in N$.

A graph admitting an interval representation, a unit interval representation and a proper interval representation, respectively, is called an *interval graph*, a *unit interval graph* and a *proper interval graph*. It is known that the class of unit interval graphs coincides with the class of proper interval graphs [31, 59, 77, 78].

2.1. Vertex orderings and membership certificates. Let G be a graph on n vertices and let τ be an ordering of $V(G)$. For any $j, k \in [n]$, we define

$$N_{G,\tau}[j] = \{i \in [n] : \tau(i) \in N_G[\tau(j)]\}$$

and

$$N^{\geq k}_{G,\tau}[j] = N_{G,\tau}[j] \setminus [k-1],\ N^{\leq k}_{G,\tau}[j] = N_{G,\tau}[j] \cap [k].$$

For each $j \in [n]$, set

$$\ell_{G,\tau}(j) = \min\{i :\ i \in N_{G,\tau}[j]\},\ \ r_{G,\tau}(j) = \max\{i :\ i \in N_{G,\tau}[j]\}.$$

Note that $\ell_{G,\tau}(j) \leq j \leq r_{G,\tau}(j)$. It is easy to see that an ordering τ of the vertex set of a graph G is an MNS ordering if and only if it obeys the MNS *rule*, namely we cannot find $1 \leq k < j \leq |V(G)|$ such that $N^{\leq k-1}_{G,\tau}[k] \subsetneq N^{\leq k-1}_{G,\tau}[j]$ happens. For any nonnegative integers s and t, we often use $[s, t]$ for the interval of integers $[t] \setminus [s-1]$. The same notation will also represent the interval of reals in some places but it should always be clear from context which usage is adopted.

Generalizing the concept of a leaf of a tree, a *simplicial vertex* of a graph is a vertex whose neighbors form a clique in the graph. A *perfect elimination ordering* (PEO) of a graph on n vertices is an ordering σ of its vertex set such that $\sigma(i)$ is simplicial in the subgraph induced by $\sigma(1), \ldots, \sigma(i)$ for any $i \in [n]$. The next result says that being a chordal graph forces the existence of simplicial vertices.

THEOREM 2.1. *[30] A graph is chordal if any only if it has a PEO.*

An LBFS *ordering* of a graph G is an ordering τ such that for any $1 \leq i < j \leq |V(G)|$, the smallest positive number k from $N^{\leq i-1}_{G,\tau}[i] \triangle N^{\leq i-1}_{G,\tau}[j]$, if any, must come from the set $N_{G,\tau}[i] \setminus N_{G,\tau}[j]$. The graph search paradigm to get just all possible LBFS orderings is just the aforementioned LBFS *algorithm* [16, 79]. One evidence that LBFS may be especially suitable for exploring interval graphs is that any LBFS ordering of an interval graph is a common PEO for all its powers [10, p. 41] [27, Corollary 6].

If τ is an MNS ordering of a graph G on n vertices, we say that $\tau(n)$ is an MNS *end-vertex* of G.

THEOREM 2.2. *[20, §2.6] The* MNS *end-vertices of a chordal graph are simplicial. Equivalently, all* MNS *orderings of a chordal graph are PEOs.*

Let G be a graph and σ an ordering of $V(G)$. We call σ an *I-ordering* if $\sigma(i)\sigma(j) \in E(G)$ implies $\sigma(i)\sigma(k) \in E(G)$ for any $1 \leq i < k < j \leq |V(G)|$ [24]. We say that the ordering σ fulfils *the neighborhood condition*, also called the *3-vertex condition*, provided $N_G[v]$ is consecutive in this ordering for any vertex v [15]. In this case, we also call σ a *UI-ordering*. Theorem 2.1 is an ordering characterization of chordal graphs. Similar ordering characterizations for interval graphs and unit interval graphs are also available. They are suggested by a very natural ordering of any family of n intervals on the real line, namely an ordering σ such that the left end-points of these intervals will never decrease when the ordered intervals appear from $\sigma(1)$ to $\sigma(n)$ in turn.

THEOREM 2.3. *[73, 76] A graph is an interval graph if and only if there is an I-ordering of its vertex set.*

THEOREM 2.4. *[78] A graph is a unit interval graph if and only if there is a UI-ordering of its vertex set.*

REMARK 2.5. *Suppose that G is a unit interval graph with n vertices and σ a UI-ordering of $V(G)$. Deng, Hell and Huang [26] propose the following simple*

construction of a proper interval representation $\mathcal{I}$ of G: $\mathcal{I}(\sigma(i)) = [i, r_{G,\sigma}(i) + 1 - \frac{1}{i}]$ for any $i \in [n]$. We also report that there are efficient means of transforming a proper interval representation into a unit interval representation of the same unit/proper interval graph [31, 59].

An equivalent formulation of Theorem 2.4 is that a graph is a unit interval graph if and only if it has a vertex ordering such that the vertices of each maximal clique of the graph appear consecutively in the ordering [78]. Finally, we mention that some other characterizations of unit (proper) interval graphs can be found in [26, 51, 64, 71, 87].

2.2. Certificates of non-membership. The following result explains why the intersection graphs of subtrees are called chordal graphs. Meanwhile, it provides us good certificates for non-membership in the class of chordal graphs.

THEOREM 2.6. *[67, Theorem 2.4] A graph is chordal, namely has a clique tree, if and only if it has no chordless cycles other than 3-cycles.*

An *asteroidal triple* (AT) is a set of three vertices such that each pair of vertices is joined by a path that avoids the closed neighborhood of the third. A graph is *asteroidal triple-free* (AT-free) provided it does not contain any AT. The family of AT-free graphs share some nice linear behaviors [14, 21]. Being AT-free prohibits a chordal graph from growing in three directions at once [21]. As described by Golumbic, in the world of an interval graph "every shipment from a supplier to the consumer must pass by the middle man" [34, p. 174].

THEOREM 2.7. *[56] A graph is an interval graph if and only if it is AT-free and chordal.*

Another characterization of interval graphs is essentially the famous **2**+**2** characterization of interval orders [29].

THEOREM 2.8. *[32, Theorem 1, Theorem 2] A graph is an interval graph if and only if it has no chordless 4-cycle and is a cocomparability graph.*

Let G be a graph and let a, b, c, d be four vertices in G. We say that $\{a, b, c, d\}$ is a *claw centered at* a in G provided b, c and d are three independent vertices from the neighbors of a in G. Being a unit interval graph prohibits an interval graph from having a claw; in other words, a claw certifies an interval graph for not being a unit interval graph.

THEOREM 2.9. *[77] An interval graph is a unit interval graph if and only if it is claw-free.*

2.3. Clique orderings. A *module* in a graph is a set of vertices such that each vertex outside of it is either adjacent to all vertices in it or adjacent to no vertex of it. A graph is *prime* if it only possesses those trivial modules, the empty set, the singleton sets and the whole vertex set. Hsu [45] observes that every prime interval graph has a unique maximal clique arrangement in certain sense. Regarding the task of recognizing interval graphs and more generally, circular-arc graphs, Hsu [45] points out that "The difficulty in dealing with circular-arc graphs lies largely in the fact that even a simple circular-arc graph (or interval graph) can have an exponential number of arc representations." Hsu and his coauthors [44, 46, 47] conquer this difficulty by putting forward a two-step recognition algorithm of interval graphs: firstly they do a modular tree decomposition to reduce the problem to the recognition of prime interval graphs and then they show that the prime graph assumption permits some smart recognition algorithm.

The problem in which we are interested in is to locate certain class of interval graphs with not so many representations in some sense so that some simple multi-sweep graph searches suffice to expose their linear structures. We propose to measure

the simplicity of an interval graph from the number of ways of ordering its maximal cliques nicely and we will thus formulate a subclass of interval graphs, which we call rigid interval graphs, in this subsection.

For chordal graphs, those special vertex orderings called PEOs play a crucial role in many algorithmic/combinatorial problems. It is worth mentioning that for some problems we had better look at some corresponding good orderings of the maximal cliques, so-called perfect sequences of chordal graphs [38, 43, 65]. We do not really need the concept of perfect sequences here but let us emphasize that it is very helpful to examine those chordal graphs from the tree-like or path-like/linear arrangement of their maximal cliques.

Let G be a graph. Let $\mathscr{C}(G)$ denote the set of all maximal cliques of G. The *maxclique-vertex matrix* of a graph G is the matrix whose rows are labelled by $\mathscr{C}(G)$ and whose columns are labelled by $V(G)$ and the (C, v)-entry is 1 if $v \in C \in \mathscr{C}(G)$ and is 0 otherwise. A *clique tree* T of G is a tree with $\mathscr{C}(G)$ as its vertex set and for each vertex v of G the nodes of the tree which contain v induce a subtree of T. A clique tree which is a path is called a *clique path*. By virtue of the Helly property of subtrees of a host tree, it is no hard to see that a graph has a clique tree if and only if it is chordal [67, Theorem 2.1]. Similarly, an important observation is that a graph has a clique path if and only if it is an interval graph [30] [67, Theorem 3.1], which relates the "sequential structure of interval graphs to a consecutiveness property of their maximal cliques" [53, p. 68]. A rephrasing of this fact is that a graph is an interval graph if and only if its maxclique-vertex matrix has the consecutive ones property for columns [67, Corollary 3.9]. Note that our comments on Theorem 2.4 at the end of §2.1 just say that the maxclique-vertex matrix of a graph G has the consecutive ones property for rows if and only if G is a unit interval graph.

The *clique graph* of a graph is the intersection graph of its maximal cliques with each edge weighted by the size of the intersection of the two cliques at both ends of the edge. It is known that the clique trees of a chordal graph correspond to exactly those maximum weight spanning trees of its clique graph [67, Theorem 2.3].

A graph is a *rigid interval graph* if it has a unique clique tree, namely its clique graph has a unique maximum weight spanning tree, and that tree is a path. If $[C_1, C_2, \ldots, C_m]$ is the unique clique tree of a graph G, we call the orderings μ and μ' of $\mathscr{C}(G)$ its *canonical orderings*, where $\mu(i) = C_i$ and $\mu'(i) = C_{m+1-i}$ for any $i \in [m]$. Rigid interval graphs include connected unit interval graphs [71, Lemma 2.9] (also see Theorem 4.3) and those connected chordal graphs having at most two maximal cliques that contain simplicial vertices [3, Theorem 4.3]. In some sense, we can say that rigid interval graphs are those graphs which can be arranged in an intrinsic linear fashion. We direct the reader to [29, Section 3.6] for more discussions of the unique linear orderings of the maximal cliques of interval graphs.

As we have seen, both unit interval graphs and interval graphs have vertex ordering characterizations. Likewise, we will develop in this paper vertex ordering characterizations for rigid interval graphs; see Theorem 4.10 and Theorem 4.18.

EXAMPLE 2.10. *Let G be the graph with exactly three maximal cliques $A = \{1, 2, 3\}, B = \{2, 3, 4, 5\}$ and $C = \{3, 5, 6\}$. It is easy to see that G is an interval graph but not any unit interval graph (by Theorem 2.9 and the fact that $\{3, 1, 4, 6\}$ is a claw in G). Note that the only clique tree of G is the path $[A, B, C]$.*

EXAMPLE 2.11. *Let G be the graph with exactly four maximal cliques, $\{4, 1\}$, $\{5, 2\}$, $\{6, 3\}$ and $\{1, 2, 3\}$. Then G has only one clique tree but G is not any interval graph.*

EXAMPLE 2.12. *The graph $K_{1,m}$ is an interval graph but any tree on its m maximal cliques can be its clique tree. Especially, when $m \geq 3$, it does not have a unique clique tree.*

EXAMPLE 2.13. *Let G be the interval graph with $V(G) = \{a_i : i \in [6]\}$ and $E(G) = \{a_1a_2, a_2a_3, a_3a_4, a_4a_5, a_3a_6\}$. It is clear that G is prime and G is not any rigid interval graph.*

REMARK 2.14. *Let G be a rigid interval graph such that there are no two different vertices sharing the same close neighborhood and there is no vertex adjacent to all other vertices. Then one can show that G is a prime graph.*

In the remainder of this section, we will tackle a bit the relationship among clique ordering, interval representation and vertex ordering.

If G has an interval representation $\mathcal{I}$, then, by the Helly property of intervals, each maximal clique $C \in \mathscr{C}(G)$ corresponds to a nonempty interval $\cap_{v\in C}\mathcal{I}(v)$, which we denote by $\mathcal{I}(C)$. It is obvious that $\mathcal{I}(C)\cap\mathcal{I}(C') = \emptyset$ for any two different maximal cliques C and C' of G. This then implies that there is an ordering μ, called the *$\mathcal{I}$-ordering* of $\mathscr{C}(G)$, such that $\mathcal{I}(\mu(1)) < \mathcal{I}(\mu(2)) < \cdots < \mathcal{I}(\mu(|\mathscr{C}(G)|))$ holds. This ordering is determined up to reversal by the graph G itself when G is a connected unit interval graph (Theorem 4.3) and may depend on the representation $\mathcal{I}$ when G is a general interval graph. It is no hard to check that $[\mu(1), \mu(2), \ldots, \mu(m)]$ is a clique path of the given interval graph.

Let G be a graph and let μ be an ordering of $\mathscr{C}(G)$. For any $v \in V(G)$, define

$$\ell_\mu(v) = \min_{v\in\mu(i)} i, \quad r_\mu(v) = \max_{v\in\mu(i)} i.$$

For any $v, w \in V(G)$, we write $v \prec_\mu w$ provided

$$\ell_\mu(v) < \ell_\mu(w) \tag{2.1}$$

or

$$\ell_\mu(v) = \ell_\mu(w),\ r_\mu(v) < r_\mu(w). \tag{2.2}$$

An ordering τ of $V(G)$ is *compatible* with an ordering μ of $\mathscr{C}(G)$ if τ is a linear extension of the partial order $\prec_\mu$. We say that an ordering τ of $V(G)$ is *left-compatible* with the ordering μ of $\mathscr{C}(G)$ provided it respects Eq. (2.1), that is,

$$1 = \ell_\mu(\tau(1)) \leq \ell_\mu(\tau(2)) \leq \cdots \leq \ell_\mu(\tau(n)) = |\mathscr{C}(G)|,$$

and we call τ *right-compatible* with μ provided

$$1 = r_\mu(\tau(1)) \leq r_\mu(\tau(2)) \leq \cdots \leq r_\mu(\tau(n)) = |\mathscr{C}(G)|.$$

The following observations highlight the links among good vertex ordering/traversal, good clique ordering/traversal and the `MNS` type algorithms and we omit their routine verifications.

REMARK 2.15. *Let G be an interval graph with a clique path $[\mu(1), \ldots, \mu(m)]$. If an ordering τ of $V(G)$ is left-compatible with μ then τ is both an* `MNS` *ordering and an I-ordering.*

REMARK 2.16. *Let G be an interval graph with n vertices. Then an ordering τ of $V(G)$ is an I-ordering if and only if G has an interval representation $\mathcal{I}$ so that $\ell_\mathcal{I}(\tau(1)) < \ell_\mathcal{I}(\tau(2)) < \cdots < \ell_\mathcal{I}(\tau(n))$. Let the $\mathcal{I}$-ordering of $\mathscr{C}(G)$ be μ. Then the*

ordering μ and the corresponding clique path is determined by τ. Moreover, τ is left-compatible with μ.

REMARK 2.17. *Let G be a unit interval graph with n vertices. Then an ordering τ of $V(G)$ is a UI-ordering if and only if G has an interval representation $\mathcal{I}$ so that $\ell_{\mathcal{I}}(\tau(1)) < \ell_{\mathcal{I}}(\tau(2)) < \cdots < \ell_{\mathcal{I}}(\tau(n))$ and $r_{\mathcal{I}}(\tau(1)) < r_{\mathcal{I}}(\tau(2)) < \cdots < r_{\mathcal{I}}(\tau(n))$. Denote by μ the $\mathcal{I}$-ordering of $\mathscr{C}(G)$. Then the ordering μ and the corresponding clique path is determined by τ. In addition, τ is left-compatible, right-compatible and even compatible with μ.*

REMARK 2.18. *Let G be a connected unit interval graph and hence a rigid interval graph. Then the set of all UI-orderings of G are exactly all those vertex orderings which are compatible with at least one of the two canonical orderings of $\mathscr{C}(G)$.*

3. Two-sweep MNS certifying algorithm for recognizing unit interval graphs. We shall first prove a succession of results related to (MNS) orderings and then prepare two results on AT in a graph, culminating in the proof of Theorem 3.9. Based on Theorem 3.9, we will be able to design our two-sweep MNS certifying algorithm for recognizing unit interval graphs. This section is presented in a way so that we can establish the correctness of the certifying algorithm as quick as possible. We defer to next section some closer look at MNS type algorithms which may provide a bit more understandings of why the algorithm is indeed a natural one.

LEMMA 3.1. *Let G be a graph on n vertices and let $i \in [n]$. Let τ be an ordering of $V(G)$ such that*

$$N_{G,\tau}^{\geq j+1}[j] \subseteq N_{G,\tau}[j+1] \tag{3.1}$$

for every $j \in [i-1]$. Then we have

$$N_{G,\tau}^{\leq \min\{i,t\}}[t] = [\ell_{G,\tau}(t), \min\{i,t\}] \tag{3.2}$$

for any $t \in [n]$.

Proof. Both sides of Eq. (3.2) are $\emptyset$ when $\ell_{G,\tau}(t) > i$. Therefore, we need only consider the case that $\ell_{G,\tau}(t) \leq i$. In this case, it is clear that $\ell_{G,\tau}(t)$ is the minimum element of both sides of Eq. (3.2). To get Eq. (3.2), it thus suffices to show that

$$j+1 \in N_{G,\tau}^{\leq \min\{i,t\}}[t] \tag{3.3}$$

as a consequence of

$$j \leq \min\{i,t\} - 1 \tag{3.4}$$

and

$$j \in N_{G,\tau}^{\leq \min\{i,t\}}[t]. \tag{3.5}$$

Note that Eqs. (3.4) and (3.5) gives $t \in N_{G,\tau}^{\geq j+1}(j)$ and hence from Eqs. (3.1) and (3.4) we deduce that $t \in N_{G,\tau}[j+1]$. This together with Eq. (3.4) guarantees Eq. (3.3), as wanted. □

LEMMA 3.2. *Let G be a graph on n vertices. Let $i \in [n]$ and let τ be an* MNS *ordering of G. Suppose that Eq. (3.1) holds for each $j \in [i-1]$. Then it holds for any $q \in [n]$ that*

$$N_{G,\tau}^{\leq i}[q] = [\ell_{G,\tau}(q), \min\{i, r_{G,\tau}(q)\}]. \tag{3.6}$$

Proof. If it occurs $q \geq i$, then Eq. (3.2) for $t = q$ is the same with Eq. (3.6) and so we can apply Lemma 3.1 for $t = q$ to derive the result. We still need to address the case of

$$q < i. \tag{3.7}$$

In view of Eqs. (3.2) and (3.7), our task is to show that $p \in N_{G,\tau}[q]$, or equivalently,

$$q \in N_{G,\tau}[p] \tag{3.8}$$

on the condition that p is an integer satisfying

$$q < p, \tag{3.9}$$

$$p \leq i \tag{3.10}$$

and

$$p \leq r_{G,\tau}(q). \tag{3.11}$$

Utilizing Lemma 3.1 for $t = r_{G,\tau}(q)$ and $t = p$, respectively, we get to

$$N_{G,\tau}^{\leq \min\{i, r_{G,\tau}(q)\}}[r_{G,\tau}(q)] = [\ell_{G,\tau}(r_{G,\tau}(q)), \min\{i, r_{G,\tau}(q)\}] \tag{3.12}$$

and

$$N_{G,\tau}^{\leq \min\{i,p\}}[p] = [\ell_{G,\tau}(p), \min\{i,p\}]. \tag{3.13}$$

Eq. (3.10) and Eq. (3.13) conspire to tell us

$$\emptyset \neq [\ell_{G,\tau}(p), p] = N_{G,\tau}^{\leq \min\{i,p\}}[p] \subseteq N_{G,\tau}^{\leq i}[p]. \tag{3.14}$$

So, if we assume for a contradiction that Eq. (3.8) does not hold, then Eq. (3.14) implies either $q > p$ or

$$q < \ell_{G,\tau}(p). \tag{3.15}$$

By Eq. (3.9), only the latter case, namely Eq. (3.15), can happen. We are now ready to deduce

$$\begin{array}{rcl} N_{G,\tau}^{\leq p-1}[p] & \subseteq & [\ell_{G,\tau}(p), p-1] \\ & \subsetneq & [q, p-1] \quad \text{(by Eqs. (3.9) and (3.15))} \\ & \subseteq & [\ell_{G,\tau}(r_{G,\tau}(q)), p-1] \quad \text{(by } \ell_{G,\tau}(r_{G,\tau}(q)) \leq q\text{)} \\ & = & N_{G,\tau}^{\leq p-1}[r_{G,\tau}(q)]. \quad \text{(by Eqs. (3.10), (3.11) and (3.12))} \end{array}$$

Comparing the above formula with Eq. (3.11) says that τ cannot be any MNS ordering of G, which is a contradiction, as wanted. □

Corollary 3.3. *A graph G on n vertices is a unit interval graph if and only if it has an* MNS *ordering τ such that Eq. (3.1) is fulfilled for all $j \in [n-1]$.*

Proof. The backward direction is a consequence of Theorem 2.4 and Lemma 3.2. We now address the other direction. Let the unit interval graph G have a unit interval

representation $\mathcal{I}$ and let τ be an ordering of $V(G)$ such that $\ell_{\mathcal{I}}(\tau(1)) \leq \ell_{\mathcal{I}}(\tau(2)) \leq \cdots \leq \ell_{\mathcal{I}}(\tau(n))$. It is easy to see that τ is what we want, finishing the proof. □

LEMMA 3.4. *Let G be a graph on n vertices. Let $u \in V(G)$ be simplicial and non-isolated. Let τ be an* MNS *ordering with $\tau(1) = u$. Suppose that there is $i \in [n]$ such that Eq.* (3.1) *is valid for any $j \in \{2, 3, \ldots, i-1\}$. Then Eq.* (3.6) *holds for any $q \in [n]$. Especially, if $i = n$, then τ is a UI-ordering of $V(G)$.*

Proof. Since $\tau(1) = u$ is a simplicial vertex of G and $\deg_G(u) > 1$, we find that $\tau(2)\tau(1) \in E(G)$ and $N^{\geq 2}_{G,\tau}[1] \subseteq N_{G,\tau}[1] \subseteq N_{G,\tau}[2]$. Accordingly, we know that Eq. (3.1) indeed holds for all $j \in [i-1]$. By now, we can directly read from Lemma 3.2 that Eq. (3.6) holds for any $q \in [n]$. When $i = n$, Eq. (3.6) just claims that $N_{G,\tau}[q] = [\ell_{G,\tau}(q), \min\{n, r_{G,\tau}(q)\}] = [\ell_{G,\tau}(q), r_{G,\tau}(q)]$ for any $q \in [n]$. This says nothing but that τ is a UI-ordering of $V(G)$. □

For any MNS type algorithm $\mathcal{A}$ with selection rule $\mathfrak{S}$, we can introduce new tie-breaking rule to favor vertices of smallest degree and get the following algorithm $\mathcal{A}^{\triangle}$, which is also of MNS type.

$\mathcal{A}^{\triangle}(G, u)$:
{ Input: a graph G with n vertices and a vertex u of G;
Output : an ordering τ of $V(G)$.}
Take $\tau(1) = u$. For any $i \in [n-1]$, after $\tau(1), \ldots, \tau(i)$ are determined, choose $\tau(i+1)$ to be a vertex in $\mathfrak{S}(\{\tau(1), \ldots, \tau(i)\})$ that has minimum degree.

LEMMA 3.5. *Let u be a simplicial vertex of a connected graph G. Let $\mathcal{A}$ be an* MNS *type algorithm. Do $\mathcal{A}^{\triangle}(G, u)$ to produce an $\mathcal{A}^{\triangle}$ ordering τ. Let $n = |V(G)|$ and take $i \in [n]$. Assume that Eq.* (3.1) *holds for every $j \in \{2, 3, \ldots, i-1\}$. Let α and β be two integers such that $\alpha \in [i+1]$ and $\alpha < \beta \leq n$. Then we have*

$$\ell_{G,\tau}(\alpha) \leq \ell_{G,\tau}(\beta). \tag{3.16}$$

In addition, if the equality in Eq. (3.16) *holds, then* $\deg_G(\tau(\alpha)) \leq \deg_G(\tau(\beta))$ *and*

$$N^{\leq \alpha-1}_{G,\tau}[\alpha] = N^{\leq \alpha-1}_{G,\tau}[\beta] = [\ell_{G,\tau}(\alpha), \alpha - 1]. \tag{3.17}$$

Proof. We observe that $\alpha - 1 < \alpha \leq r_{G,\tau}(\alpha)$, $\alpha - 1 < \alpha < \beta \leq r_{G,\tau}(\beta)$, $\alpha - 1 \leq i$. Thus applying Lemma 3.4 yields

$$\begin{cases} N^{\leq \alpha-1}_{G,\tau}[\alpha] = [\ell_{G,\tau}(\alpha), \min\{\alpha - 1, r_{G,\tau}(\alpha)\}] = [\ell_{G,\tau}(\alpha), \alpha - 1]; \\ N^{\leq \alpha-1}_{G,\tau}[\beta] = [\ell_{G,\tau}(\beta), \min\{\alpha - 1, r_{G,\tau}(\beta)\}] = [\ell_{G,\tau}(\beta), \alpha - 1]. \end{cases} \tag{3.18}$$

A quick consequence of Eq. (3.18) is that Eq. (3.17) is true whenever the equality in Eq. (3.16) holds. Moreover, as τ is an $\mathcal{A}^{\triangle}$ ordering and $\alpha < \beta$, Eq. (3.17) certainly guarantees $\deg_G(\tau(\alpha)) \leq \deg_G(\tau(\beta))$ when $\alpha > 1$. On the other hand, if $\alpha = 1$ and $\ell_{G,\tau}(\alpha) = \ell_{G,\tau}(\beta)$, recalling that $\tau(1) = u$ is simplicial, we have $N_G[\tau(1)] \subseteq N_G[\tau(\beta)]$ and henceforth $\deg_G(\tau(\alpha)) = \deg_G(\tau(1)) \leq \deg_G(\tau(\beta))$ also follows.

The remainder of the proof is to establish Eq. (3.16). It is trivial for $\alpha = 1$ as $\ell_{G,\tau}(1) = 1$. We now look at the case of $\alpha > 1$. The connectedness of G and the MNS rule imply that $N^{\leq \alpha-1}_{G,\tau}[\alpha]$ must be nonempty and so Eq. (3.18) shows us that

$$N^{\leq \alpha-1}_{G,\tau}[\alpha] \subsetneq N^{\leq \alpha-1}_{G,\tau}[\beta] \tag{3.19}$$

whenever $\ell_{G,\tau}(\alpha) > \ell_{G,\tau}(\beta)$. Since τ is an MNS ordering and $\alpha < \beta$, Eq. (3.19) is impossible and so we arrive at Eq. (3.16) again, as claimed. □

LEMMA 3.6. *Let G be a connected graph on n vertices and let σ be an* MNS *ordering of $V(G)$. If $G[V(G) \setminus N_G[\sigma(n)]]$ has at least two different connected components, say A and B, then there exist $c \in N_G(\sigma(n))$, $a' \in N_G(c) \cap A$ and $b' \in N_G(c) \cap B$ such that $\{a', b', c, \sigma(n)\}$ is a claw in G centered at c.*

Proof. Set $p = \min_{a \in A} \sigma^{-1}(a)$ and $q = \min_{b \in B} \sigma^{-1}(b)$. Without loss of generality, assume that $p < q$. Observe that $N_G[\sigma(q)] \subseteq N_G(\sigma(n)) \cup B$ and hence, as $\sigma(q)$ is the earliest vertex in B visited by σ, we conclude that $N_{G,\sigma}^{\leq q-1}[n] \supseteq N_{G,\sigma}^{\leq q-1}[q]$. Considering the fact that $n > q$, the rule of MNS then enables us assert that

$$N_{G,\sigma}^{\leq q-1}[n] = N_{G,\sigma}^{\leq q-1}[q]. \tag{3.20}$$

Furthermore, the connectedness of G and the rule of MNS ensure that $\{\sigma(1), \sigma(2), \ldots, \sigma(q)\}$ induces a connected subgraph H of G. Since $p < q$, $\{\sigma(q)\} = V(H) \cap B$ and $\sigma(p) \in V(H) \cap A$, there is an induced path $[b', c_1, \ldots, c_k, a']$ in H such that $a' \in V(H) \cap A$, $b' = \sigma(q)$, and $c_1, \ldots, c_k \in N_{G,\sigma}^{\leq q-1}[n]$. Subsequently, we appeal to Eq. (3.20) to get that $k = 1$ and then we can check that the four vertices $\sigma(n), a', b'$ and $c = c_1$ form a desired claw in G, finishing the proof. □

LEMMA 3.7. *Let z be a simplicial vertex of a graph G. Let a and b be two neighbors of z in G such that there exist $v \in N_G[a] \setminus N_G[b]$ and $u \in N_G[b] \setminus N_G[a]$. Then the following hold:*

(i) *Neither u nor v belongs to $N_G[z]$.*

(ii) *If u and v are connected by a path P in $G[V(G) \setminus N_G[z]]$, then either $\langle a, b, u, v\rangle$ is a chordless 4-cycle in G or $\{u, v, z\}$ is an AT in G with three certificating paths $[v, a, z]$, $[z, b, u]$ and P.*

Proof. On account of the fact that z is simplicial, $za, zb \in E(G)$, and $au, bv \notin E(G)$, it is easy to deduce that $u, v \notin N_G[z]$ and $ab \in E(G)$. If $uv \in E(G)$, then clearly $\langle a, b, u, v\rangle$ is a chordless 4-cycle of G. If $uv \notin E(G)$, we see that v misses the path $[z, b, u]$ and u misses the path $[v, a, z]$, and hence $\{u, v, z\}$ is a required AT in G with the asserted certificating paths. □

LEMMA 3.8. *Let G be a graph without any induced 4-cycle. Let a, b, u, v, w, z be six different vertices of G such that $ab, bz, za, zu, av, bw \in E(G)$ and uv, ua, ub, uw, vz, vb, wa, $wz \notin E(G)$. Then $\{u, v, w\}$ is an AT of G with the certificate paths $[v, a, z, u], [u, z, b, w]$ and $[w, b, a, v]$.*

Proof. It suffices to verify that $vw \notin E(G)$. But, if it were not the case, then $\langle v, w, b, a\rangle$ will be a chordless 4-cycle in G, violating our assumption. □

THEOREM 3.9. *Let G be a connected graph without chordless 4-cycles. Let u be both a simplicial vertex and an* MNS *end-vertex of G. Let $\mathcal{A}$ be an* MNS *type algorithm and let τ be an ordering produced by $\mathcal{A}^{\triangle}(G, u)$. Then either τ is a UI-ordering of $V(G)$ or G contains either a claw or an AT.*

Proof. Let $n = |V(G)|$.

CASE 1. $N_{G,\tau}^{\geq i+1}[i] \subseteq N_{G,\tau}[i+1]$ for $i = 2, \ldots, n-1$. We conclude from Lemma 3.4 that τ is a UI-ordering of $V(G)$.

CASE 2. Among $\{2, 3, \ldots, n-1\}$, there is a minimum i such that $N_{G,\tau}^{\geq i+1}[i] \nsubseteq N_{G,\tau}[i+1]$. Let $p = \ell_{G,\tau}(i)$, $q = \ell_{G,\tau}(i+1)$, and take

$$s \in N_{G,\tau}^{\geq i+1}[i] \setminus N_{G,\tau}[i+1] = N_{G,\tau}^{\geq i+2}[i] \setminus N_{G,\tau}[i+1]. \tag{3.21}$$

Lemma 3.4 says $N_{\widetilde{G},\tau}^{\leq i}[i+1] = [q, i]$ while the connectedness of G gives $N_{\widetilde{G},\tau}^{\leq i}[i+1] \neq \emptyset$. This then leads to

$$i \in N_{\widetilde{G},\tau}^{\leq i}[i+1]. \tag{3.22}$$

Setting $\alpha = i+1$ and $\beta = s$, Lemma 3.5 asserts that

$$q \leq \ell_{G,\tau}(s). \tag{3.23}$$

Applying Lemma 3.5 again for $\alpha = i$ and $\beta = i+1$ ensures $p \leq q$.

Case 2.1. $p < q$. Eq. (3.23) combined with $p < q$ gives $p \notin N_{G,\tau}[s]$. The definition of q along with $p < q$ guarantees that $p \notin N_{G,\tau}[i+1]$. The fact that $s \notin N_{G,\tau}[i+1]$ comes from Eq. (3.21). In light of the choice of p and s, it is direct that $\tau(i)\tau(p), \tau(i)\tau(s) \in E(G)$. Summarizing all these as well as Eq. (3.22), the set $\{\tau(i), \tau(i+1), \tau(s), \tau(p)\}$ is a claw in G with $\tau(i)$ being its center.

Case 2.2. $p = q$. In view of Lemma 3.5, we see that $\deg_G(\tau(i)) \leq \deg_G(\tau(i+1))$ and $N_{\widetilde{G},\tau}^{\leq i-1}[i] = N_{\widetilde{G},\tau}^{\leq i-1}[i+1] = [p, i-1]$. Based on this information, Eq. (3.21) and Eq. (3.22), we can thus conclude that there exists

$$t \in N_{\widetilde{G},\tau}^{\geq i+2}[i+1] \setminus N_{G,\tau}[i]. \tag{3.24}$$

We remark that, by Lemma 3.5 for $(\alpha, \beta) = (i, s)$ and (i, t), it happens

$$p \leq \min\{\ell_{G,\tau}(s), \ell_{G,\tau}(t)\}. \tag{3.25}$$

Case 2.2.1. $p = q > 1$. As G is connected, we can find an element

$$k \in N_{\widetilde{G},\tau}^{\leq p-1}[p].$$

Note that Eq. (3.25) immediately tells us that $\{s, t\} \cap N_{G,\tau}[k] = \emptyset$. Also, it follows from $k < p = \ell_{G,\tau}(i) = \ell_{G,\tau}(i+1)$ that $\{i, i+1\} \cap N_{G,\tau}[k] = \emptyset$. Lastly, recall that $\{\tau(i), \tau(i+1), \tau(k)\} \subseteq N_G(\tau(p))$.

Case 2.2.1.1. $s \in N_{G,\tau}(p)$. Notice that Eq. (3.21) asserts $s \notin N_{G,\tau}[i+1]$. We are then in a position to check that $\{\tau(p), \tau(i+1), \tau(s), \tau(k)\}$ is a claw in G centered at $\tau(p)$.

Case 2.2.1.2. $t \in N_{G,\tau}(p)$. For this case we need to recall from Eq. (3.24) that $t \notin N_{G,\tau}[i]$ and then we can verify that $\{\tau(p), \tau(i), \tau(t), \tau(k)\}$ is a claw with $\tau(p)$ at the center.

Case 2.2.1.3. Neither s nor t lies in $N_{G,\tau}(p)$. We employ Lemma 3.8 for $v = \tau(s)$, $w = \tau(t)$, $u = \tau(k)$, $z = \tau(p)$, $a = \tau(i)$ and $b = \tau(i+1)$ and then find that $\{\tau(s), \tau(t), \tau(k)\}$ is an AT of G with three certificate paths $[\tau(s), \tau(i), \tau(p), \tau(k)]$, $[\tau(k), \tau(p), \tau(i+1), \tau(t)]$ and $[\tau(t), \tau(i+1), \tau(i), \tau(s)]$. Notice that both $s \notin N_{G,\tau}[i+1]$ (Eq. (3.21)) and $t \notin N_{G,\tau}[i]$ (Eq. (3.24)) are used in the above checking process.

Case 2.2.2. $p = q = 1$. Because $\tau(1) = u$ is simplicial, Lemma 3.7 (i) says that neither s nor t lies in $N_{G,\tau}[1]$. Let A and B be the connected components of $G[V(G) \setminus N_G[\tau(1)]]$ such that $\tau(s) \in A$ and $\tau(t) \in B$.

Case 2.2.2.1. $A = B$. This means that $\tau(s)$ and $\tau(t)$ can be connected by a path P which misses $\tau(1)$. Utilizing Lemma 3.7 (ii) for $z = \tau(1), a = \tau(i), b = \tau(i+1), v = \tau(s), u = \tau(t)$, we see that $\{\tau(s), \tau(t), \tau(1)\}$ is an AT and the three certificate paths are $[\tau(s), \tau(i), \tau(1)]$, $[\tau(1), \tau(i+1), \tau(t)]$ and the path P.

Case 2.2.2.2. Now assume that $A \neq B$. Recall that $\tau(1) = u$ is an MNS end-vertex. By Lemma 3.6, there exist $c \in N_G(\tau(1))$, $a' \in A$ and $b' \in B$ such that $\{\tau(1), c, a', b'\}$ is a claw centered at c, completing the proof. □

Given any MNS type algorithm $\mathcal{A}$, we now present an algorithm which we call T-MNS$^{\mathcal{A}}$, or the Two-Sweep MNS UIG Certifying Algorithm based on $\mathcal{A}$. Note that Theorems 2.1, 2.2, 2.4, 2.7, 2.9 and 3.9 validate the correctness of this algorithm. In the following pseudocodes of T-MNS$^{\mathcal{A}}$, we use annotations to indicate the correspondence between the parts of the algorithm with Theorem 2.2 and the different cases distinguished in the proof of Theorem 3.9.

T-MNS$^{\mathcal{A}}(G)$

▷ **Input** a connected graph G with $|V(G)| = n$
▷ **Output** an answer that G is a UIG as well as one of its UI-ordering or
▷ a certificate showing that G is not a UIG.
Generate an MNS ordering σ of $V(G)$
if σ is not a PEO
  then return an induced cycle of length at least four
    ▷ Theorem 2.2
Do $\mathcal{A}^{\triangle}(G, \sigma(n))$ to yield an ordering τ
$N^{\geq i+1}_{G,\tau}[i] \leftarrow \{k : \ k \geq i+1 \text{ and } \tau(k) \in N_G[\tau(i)]\}$
for $i \leftarrow 2$ **to** $n-1$
  do if $N^{\geq i+1}_{G,\tau}[i] \not\subseteq \tau^{-1}(N_G[\tau(i+1)])$
    then $p \leftarrow \min\{j : \ \tau(j) \in N_G(\tau(i))\}$
      $q \leftarrow \min\{j : \ \tau(j) \in N_G(\tau(i+1))\}$
      ▷ we must have $p \leq q$ by Lemma 3.5
      Pick an $s \in N^{\geq i+1}_{G,\tau}[i] \setminus \tau^{-1}(N_G[\tau(i+1)])$
      if $p < q$
        then return $\{\tau(p), \tau(i), \tau(i+1), \tau(s)\}$ is a claw
          ▷ Case 2.1
      Choose a t so that $\tau(t) \in N_G[\tau(i+1)] \setminus N_G[\tau(i)]$
      if $p = q > 1$
        then pick a vertex $\tau(k)$ such that $k < p$ and $\tau(k) \in N_G(\tau(p))$
          if $\tau(s) \in N_G(\tau(p))$
            then return $\{\tau(k), \tau(p), \tau(i+1), \tau(s)\}$ is a claw
              ▷ Case 2.2.1.1
          if $\tau(t) \in N_G(\tau(p))$
            then return $\{\tau(k), \tau(p), \tau(i), \tau(t)\}$ is a claw
              ▷ Case 2.2.1.2
          return $\{\tau(k), \tau(s), \tau(t)\}$ is an AT with certificate paths $[\tau(s), \tau(i),$
          $\tau(p), \tau(k)], [\tau(k), \tau(p), \tau(i+1), \tau(t)]$ and $[\tau(t), \tau(i+1), \tau(i), \tau(s)]$
          ▷ Case 2.2.1.3
      ▷ We now have $p = q = 1, \tau(s), \tau(t) \in V(G) \setminus N_G[\tau(1)]$
      $A \leftarrow$ the connected component of $G[V \setminus N_G[\tau(1)]]$ containing $\tau(s)$
      $B \leftarrow$ the connected component of $G[V \setminus N_G[\tau(1)]]$ containing $\tau(t)$
      if $A = B$
        then $P \leftarrow$ a path in A connecting $\tau(s)$ and $\tau(t)$
          return $\{\tau(s), \tau(t), \tau(1)\}$ is an AT with three certificate paths
          $P, [\tau(s), \tau(i), \tau(1)], [\tau(t), \tau(i+1), \tau(1)]$
          ▷ Case 2.2.2.1

39 **else** choose $u \in N_G(A) \cap N_G(B) \cap N_G(\tau(1))$
40 choose $x \in A \cap N_G(u), y \in B \cap N_G(u)$
41 **return** $\{\tau(1), u, x, y\}$ is a claw
42 ▷ Case 2.2.2.2
43 **return** τ is a UI-ordering of the UIG G
44 ▷ Case 1

We point out that the T-MNS$^{\mathcal{A}}$ algorithm can be implemented easily in linear time as long as there is good linear time implementation of $\mathcal{A}^{\triangle}$, say if $\mathcal{A}$ is LBFS. Indeed, Step 6 can be done in linear time using the algorithm of [86]. Step 4 can be done with any easily implementable MNS type algorithm, say LBFS, MCS, or even LDFS.

It follows from the correctness of the above T-MNS$^{\mathcal{A}}$ algorithm that the following algorithm correctly recognizes unit interval graphs.

Two-sweep MNS algorithm for recognizing unit interval graphs based on an MNS type algorithm $\mathcal{A}$: { Input: a connected graph G on n vertices; Output: a statement declaring whether or not G is a unit interval graph.} Step 1. Generate an MNS ordering σ of $V(G)$; Step 2. Do $\mathcal{A}^{\triangle}(G, \sigma(n))$ to yield sweep τ; Step 3. If τ is a UI-ordering of G, then conclude that G is a unit interval graph; else, conclude that G is not any unit interval graph.

REMARK 3.10. *We point out that UI-ordering is very easy to test [15, p. 377] and a unit interval representation can be constructed from a UI-ordering directly by [18, Theorem 3.2] or indirectly by Remark 2.5. Thus, choosing $\mathcal{A}$ to be* LBFS*, the above algorithm can be easily executed in linear time and can even be used to reconstruct a unit interval representation in linear time when the input is a unit interval graph.*

As a consequence of the correctness of their certifying 3-sweep LBFS algorithm for recognizing unit interval graphs, Hell and Huang [41] deduce again the famous characterization of unit interval graphs first discovered by Wegner [87]. In the last part of this section, let us show that Theorem 3.9 also implies similar side-result.

Let G be a graph. A vertex v of G is *admissible* if there are no vertices x and y such that there is an x, v-path avoiding the closed neighborhood of y and there is an y, v-path avoiding the closed neighborhood of x. The set of vertices of G which are noth simplicial and admissible is denoted by $AS(G)$.

THEOREM 3.11. *[19, Theorem 3.1] Let G be an arbitrary graph. Then any vertex from $AS(G)$ must be an* LBFS *end-vertex.*

COROLLARY 3.12. *Let G be a connected AT-free graph such that none of its induced subgraph is a 4-cycle or a claw. If $AS(G) \neq \emptyset$, then G is a unit interval graph. In particular, G cannot contain any induced 5-cycle.*

Proof. By Theorem 3.11, any vertex from $AS(G)$ must be both simplicial and an MNS end-vertex. For now, we see that Theorem 3.9 implies this result, as wanted. □

It is interesting to compare Corollary 3.12 with Theorem 2.8. Note that the condition that $AS(G) \neq \emptyset$ cannot be relaxed to the existence of simplicial vertices, as can be seen from the 5-cycle with one new vertex joining to a vertex on the cycle. We also mention that each LBFS end-vertex of an AT-free graph is admissible [23, Theorem 4.1], which, combined with Theorem 2.2 and 2.7, asserts that $AS(G) \neq \emptyset$

for any interval graph G and so Corollary 3.12 can be turned into a characterization of unit interval graphs.

4. Characterizations and recognition algorithm of rigid interval graphs. This section aims to give very simple multi-sweep MNS algorithms, in which we do not need to sort vertices according to degrees, for recognizing unit interval graphs and rigid interval graphs, respectively. Before we launch into a discussion of those algorithms, it is useful to examine the structural properties of rigid interval graphs.

LEMMA 4.1. *Let G be a graph possessing a clique path $P = [C_1, C_2, \ldots, C_m]$. Suppose that for every $i \in [m-2]$ it holds*

$$(C_i \cap C_{i+1}) \setminus C_{i+2} \neq \emptyset \quad \text{and} \quad (C_{i+1} \cap C_{i+2}) \setminus C_i \neq \emptyset. \tag{4.1}$$

Then P is the only clique tree of G and hence G is a rigid interval graph.

Proof. Let T be any clique tree of G and we want to show that $T = P$. Take any two maximal cliques C_p and C_q which are adjacent in T. Without loss of generality, assume that $p < q$ and then our task is to derive $p = q - 1$. Let us assume $p < q - 1$ and aim to get a contradiction.

Pick $x \in C_p \setminus C_q$ and $y \in C_q \setminus C_p$. Since $C_pC_q \in E(T)$ and T is a clique tree of G, we know that $C_p \cap C_q$ is an x, y-separator in G. On the other hand, from Eq. (4.1) we find that there exists $v_i \in (C_i \cap C_{i+1}) \setminus C_{i+2} \subseteq (C_i \cap C_{i+1}) \setminus C_q$ for any $i \in [p, q-2]$ and there exists $v_{q-1} \in (C_{q-1} \cap C_q) \setminus C_{q-2} \subseteq (C_{q-1} \cap C_q) \setminus C_p$. In view of $p < q - 1$, it is clear that $x, v_p, v_{p+1}, \ldots, v_{q-1}, y$ is a vertex sequence such that every two consecutive vertices in it are adjacent in $G[V(G) \setminus (C_p \cap C_q)]$, contracting with the fact that $C_p \cap C_q$ is an x, y-separator in G. This finishes the proof. □

LEMMA 4.2. *Let T be a clique tree of a connected chordal graph G. Let C_1, C_2, C_3 be three maximal cliques of G so that $\mathrm{Dist}_T(C_1, C_3) = \mathrm{Dist}_T(C_2, C_3) + 1 = \mathrm{Dist}_T(C_2, C_3) + \mathrm{Dist}_T(C_1, C_2)$. If either $C_1 \cap C_3 = \emptyset$ or $C_2 \setminus (C_1 \cup C_3) = \emptyset$ holds, then $(C_1 \cap C_2) \setminus C_3 \neq \emptyset$.*

Proof. Note that C_2 is the disjoint union of $X = C_2 \cap C_1 \cap C_3$, $Y = (C_2 \cap C_1) \setminus C_3$, $Z = C_2 \setminus (C_1 \cup C_3)$ and $W = (C_2 \cap C_3) \setminus C_1$. Our claim is just $Y \neq \emptyset$.

Suppose that $Y = \emptyset$. This gives $Z = Y \cup Z = C_2 \setminus C_3 \neq \emptyset$. It then comes from our assumption that $C_1 \cap C_3 = \emptyset$ and so $X = \emptyset$ follows. But $C_1 \cap C_2 = X \cup Y = \emptyset$ together with $C_1C_2 \in E(T)$ implies that $\emptyset$ is the minimal separator for any vertex in C_1 and any vertex in C_2, contradicting with the connectedness of G. □

THEOREM 4.3. *[71, Lemma 2.9] A connected unit interval graph must be a rigid interval graph.*

Proof. Let G be a connected unit interval graph with m maximal cliques. Let $\mathcal{I}$ be a unit interval representation of G and let μ be the $\mathcal{I}$-ordering of $\mathscr{C}(G)$. Note that the path $P = [\mu(1), \mu(2), \ldots, \mu(m)]$ is a clique tree of G. For any $i \in [m-2]$, if there are $v \in \mu(i) \cap \mu(i+2)$ and $w \in \mu(i+1) \setminus (\mu(i) \cup \mu(i+2))$, then we will find $\mathcal{I}(v) \supsetneq \mathcal{I}(w)$, contradicting with the assumption that $\mathcal{I}$ is a unit interval representation. Consequently, applying Lemmas 4.1 and 4.2 yields the asserted result. □

THEOREM 4.4. *Let G be a graph having a clique path $P = [C_1, C_2, \ldots, C_m]$. Then P is the unique clique tree of G if and only if Eq. (4.1) holds for every $i \in [m-2]$.*

Proof. The backward direction is just Lemma 4.1 and so our concern is the other implication.

We now suppose that P is the unique clique tree of G and will complete the proof via an induction on the number m of maximal cliques. Note that the result is trivially true when $m \leq 2$. Assume $m \geq 3$. If $(C_m \cap C_{m-1}) \setminus C_{m-2} = \emptyset$, then we replace the edge C_mC_{m-1} by the new edge C_mC_{m-2} and thus obtain from P a new clique tree

of G; If $(C_{m-2} \cap C_{m-1}) \setminus C_m = \emptyset$, then $[C_1, C_2, \ldots, C_{m-2}, C_m, C_{m-1}]$ is a clique tree of G other than P. So we are reduced to proving Eq. (4.1) for $i \in [m-3]$. Consider the graph G' with vertex set $\cup_{i\in[m-1]}C_i$ and edge set $\cup_{i\in[m-1]}\binom{C_i}{2}$. Note that $P' = [C_1, C_2, \ldots, C_{m-1}]$ is a clique path of G'. Furthermore, P' should be the unique clique path of G'. Otherwise, for any other clique tree T' of G', we can attach the vertex C_m to T' by adding an edge C_mC_{m-1} and then yield a clique tree of G which is different from P. Consequently, Eq. (4.1) for $i \in [m-3]$ becomes a corresponding claim for the graph G' with $m-1$ maximal cliques and this holds true by our induction assumption. □

The foregoing theorem suggests us search for some conditions which ensures the validity of Eq. (4.1). In order to present such conditions, we need to formalize some concepts.

Definition 4.5. *Let G be a graph with n vertices and let τ be an ordering of $V(G)$.*

- *If $\tau(p)\tau(p+1) \in E(G)$ for any $p \in [n-1]$, we call τ a* consecutive *ordering.*
- *For an ordering μ of $\mathscr{C}(G)$, we say that τ is* strongly left-compatible *with μ provided τ is consecutive and is left-compatible with μ. Recall from Remark 2.15 that τ must be both an* MNS *ordering and an I-ordering when $[\mu(1), \ldots, \mu(|\mathscr{C}(G)|)]$ is a clique path of G.*
- *An ordering τ of a graph G is a* perfect clique slice ordering*, or* PCSO *for short, if it is an* MNS *ordering and each of its* MNS *slices is a clique.*

Remark 4.6. *For an I-ordering of a graph, its* MNS *slice coincides with its* LBFS *(*MCS*) slice and, particularly, it is an* MNS *ordering if and only if it is an* LBFS *(*MCS*) ordering.*

Lemma 4.7. *Let G be a graph with m maximal cliques and μ an ordering of $\mathscr{C}(G)$ such that $[\mu(1), \ldots, \mu(m)]$ is a clique path of G. If an ordering τ of $V(G)$ is strongly left-compatible with μ, then $(\mu(i+1) \cap \mu(i+2)) \setminus \mu(i) \neq \emptyset$ for any $i \in [m-2]$.*

Proof. Pick $i \in [m-2]$ and let $s = \max\{r : \tau(r) \in \mu(i+1)\}$. Surely,

$$\tau(s) \in \mu(i+1). \tag{4.2}$$

Because τ is left-compatible with μ, we have

$$\tau(s) \notin \mu(i) \tag{4.3}$$

and

$$\ell_\mu(\tau(s+1)) \geq i+2. \tag{4.4}$$

But τ is a consecutive ordering and so $\tau(s)\tau(s+1) \in E(G)$. This together with Eq. (4.4) implies

$$r_\mu(\tau(s)) \geq i+2. \tag{4.5}$$

Since $[\mu(1), \ldots, \mu(m)]$ is a clique path, Eq. (4.2) and Eq. (4.5) show that

$$\tau(s) \in \mu(i+2). \tag{4.6}$$

Combining Eqs. (4.2), (4.3) and (4.6) yields $\tau(s) \in (\mu(i+1) \cap \mu(i+2)) \setminus \mu(i)$, which is what we want. □

Lemma 4.8. *Let G be a graph with m maximal cliques and μ an ordering of $\mathscr{C}(G)$ such that $[\mu(1), \ldots, \mu(m)]$ is a clique path of G. If G has a PCSO τ which is left-compatible with μ, then $(\mu(i) \cap \mu(i+1)) \setminus \mu(i+2) \neq \emptyset$ for any $i \in [m-2]$.*

Proof. Take $i \in [m-2]$ and choose $s = \min\{r : \tau(r) \in \mu(i+1) \setminus \mu(i+2)\}$. We assert that

$$\tau(s) \in (\mu(i) \cap \mu(i+1)) \setminus \mu(i+2), \tag{4.7}$$

from which the lemma follows. To establish Eq. (4.7), it suffices to check that

$$\tau(s) \in \mu(i). \tag{4.8}$$

If Eq. (4.8) were not true, then

$$\ell_\mu(\tau(s)) = r_\mu(\tau(s)) = i+1. \tag{4.9}$$

Pick $\tau(t) \in \mu(i+2) \setminus \mu(i+1)$. Owing to the fact that τ is left-compatible with μ, we see that $s < t$. In light of Eq. (4.9), $\mu(t)$ cannot be a neighbor of $\tau(s)$ and, further considering the minimality assumption on s, it holds $N_{G,\tau}^{\leq s-1}[s] \subseteq \tau^{-1}(\mu(i+1) \cap \mu(i+2)) \subseteq N_{G,\tau}^{\leq s-1}[t]$. We thus find that the two two nonadjacent vertices $\tau(s)$ and $\tau(t)$ both belong to $S_\tau[s]$. This is absurd as we already assume that τ is a PCSO, finishing the proof. □

Let us introduce another concept here, which prepares us for an ordering characterization of rigid interval graphs. Let τ be an ordering of the vertex set of a graph. We refer to τ as a *rigid interval ordering* provided it is both a consecutive I-ordering and a PCSO.

REMARK 4.9. *Let G be a rigid interval graph, let μ be a canonical ordering of $\mathscr{C}(G)$ and let τ be any ordering of $V(G)$ which is compatible with μ. It is no hard to check that τ is a rigid interval ordering.*

THEOREM 4.10. *Let G be a graph with n vertices. Then G is a rigid interval graph if and only if G has a rigid interval ordering τ.*

Proof. The "only if" part is simply Remark 4.9. The rest of the proof is devoted to the converse.

By Remark 2.16, the I-ordering τ determines an ordering μ of $\mathscr{C}(G)$ so that τ is left-compatible with μ and $[\mu(1), \ldots, \mu(m)]$ is a clique path of G, where $m = |\mathscr{C}(G)|$. Let $C_i = \mu(i)$ for $i \in [m]$. By Theorem 4.4, we need to check that Eq. (4.1) holds for every $i \in [m-2]$. But the latter half of Eq. (4.1) comes from Lemma 4.7 while the first half is a consequence of Lemma 4.8. This is the proof. □

EXAMPLE 4.11. *Let G be the graph with $V(G) = \{v_i : i \in [5]\}$ and $E(G) = \{v_1v_2, v_1v_3, v_2v_3, v_2v_4, v_3v_5\}$. Then the ordering v_1, v_2, v_3, v_4, v_5 is both a PCSO and an I-ordering. But G is not any rigid interval graph. Note that $v_4v_5 \notin E(G)$ and so this ordering is not consecutive.*

LEMMA 4.12. *Let G be a graph with m maximal cliques. Let μ' be an ordering of $\mathscr{C}(G)$ such that $[\mu'(1), \ldots, \mu'(m)]$ is a clique path of G and*

$$(\mu'(i) \cap \mu'(i+1)) \setminus \mu'(i+2) \neq \emptyset \tag{4.10}$$

for any $i \in [m-2]$. Then any `MNS` *ordering τ starting from an element in $\mu'(1) \setminus \mu'(2)$ must be left-compatible with μ' and hence is an I-ordering of G.*

Proof. Let q be the maximum number such that $\{\tau(1), \tau(2), \ldots, \tau(q)\}$ is a clique in G. By the `MNS` rule and the fact that the only maximal clique that contains $\tau(1)$ is $\mu'(1)$, we see that $\{\tau(1), \tau(2), \ldots, \tau(q)\} = \mu'(1)$. Let x and y be two vertices such that $\ell_{\mu'}(x) < \ell_{\mu'}(y)$. What we need to show is $r < p$ where $r = \tau^{-1}(x)$ and $p = \tau^{-1}(y)$.

When $\ell_{\mu'}(x) = 1$, we have $r \leq q < p$ and hence the claim is trivially true.

Take $i \in [m-2]$ and we now proceed to deal with the case of $\ell_{\mu'}(x) = i+1 > 1$ under the assumption that the claim is valid for smaller $\ell_{\mu'}(x)$. Without loss of generality, we assume that $p = \tau^{-1}(y) = \min_{\ell_{\mu'}(z)>i} \tau^{-1}(z)$. This assumption as well as the fact that P is a clique path tell us that $N_{G,\tau}^{\leq p-1}[p] \subseteq N_{G,\tau}^{\leq p-1}[r]$. Considering that τ is an MNS ordering, to get $r < p$ it is now sufficient to show the existence of an element from $N_{G,\tau}^{\leq p-1}[r] \setminus N_{G,\tau}^{\leq p-1}[p]$.

By virtue of Eq. (4.10), we can pick a vertex $w \in (\mu'(i) \cap \mu'(i+1)) \setminus \mu'(i+2) \subseteq (\mu'(i) \cap \mu'(i+1)) \setminus \mu'(\ell_{\mu'}(y))$. Since $\ell_{\mu'}(w) \leq i$, our induction assumption implies that $\tau^{-1}(w) < \tau^{-1}(y) = p$. This means that $\tau^{-1}(w) \in N_{G,\tau}^{\leq p-1}[r] \setminus N_{G,\tau}^{\leq p-1}[p]$ and hence we are done. □

EXAMPLE 4.13. *Let G be the graph with $V(G) = \{v_1, v_2, v_3, v_4, v_5\}$ and $E(G) = \{v_1v_2, v_2v_3, v_3v_4, v_4v_2, v_3v_5\}$. The* MNS *orderings starting at v_1 are just v_1, v_2, v_3, v_4, v_5 and v_1, v_2, v_4, v_3, v_5. It is easy to check that both of them are I-orderings, confirming the assertion of Lemma 4.12. Observe that the latter is even a UI-ordering but the former is not any UI-ordering.*

Moving just one step forward from Lemma 4.12, we get to the next simple theorem, which is the crux in our work to understand the MNS structure of rigid interval graphs.

THEOREM 4.14. *Let G be an interval graph on n vertices with the unique clique tree $P = [\mu(1), \mu(2), \ldots, \mu(m)]$. Let τ be an* MNS *ordering of $V(G)$. Then, there exists $s \in [m]$ such that*

(i) $\tau(1) \in \mu(s)$;
(ii) $\tau^{-1}(y) > \tau^{-1}(x)$ *for any* $x, y \in V(G)$ *satisfying* $s \leq \ell_\mu(x) < \ell_\mu(y)$;
(iii) $\tau^{-1}(y) > \tau^{-1}(x)$ *for any* $x, y \in V(G)$ *satisfying* $s \geq r_\mu(x) > r_\mu(y)$.

Proof. Let q be the maximum number such that $\{\tau(1), \tau(2), \ldots, \tau(q)\}$ is a clique in G. By the MNS rule, we see that $\{\tau(1), \tau(2), \ldots, \tau(q)\}$ must indeed form a maximal clique C of G. Take $s = \mu^{-1}(C)$ and then claim (i) is satisfied. By symmetry, we only need to verify (ii) in the sequel. By virtue of Lemma 4.4, we can utilize the same arguments as in the proof of Lemma 4.12 to do the job and we leave the routine verification to the readers. □

If we view MNS as traversing maximal cliques rather than vertices (a clique is traversed when its last vertex has been traversed), Theorem 4.14 says that by applying an MNS type algorithm on a rigid interval graph we simply explore outward from one maximal clique in two directions and "flood" the unique clique path with an expanding wave that grows and finally stops at an end-clique. We refer to [53, Lemma 3.2] for a similar result for general interval graph and LBFS.

Theorem 2.2 suggests that the MNS end-vertices of a chordal graph, especially of a tree, are somehow on the boundary of the graph. This is also confirmed by the result of Dragan et al. [27, Corollary 7] that the LBFS end-vertex of a connected interval graph has eccentricity equal to the diameter of the graph. Intuitively, a good transversal of an interval graph should walk through the graph from one side to the other side and so an MNS end-vertex may be a good starting point for a good vertex ordering. Note that Theorem 4.14 further supports the idea that an MNS end-vertex is on one extreme of the rigid interval graph – we refer the readers to [5, 12, 13, 19, 58] for more discussions on various graph extremities. On the other hand, Lemma 4.12, a special case of Theorem 4.14, supports the intuition that starting from an extreme vertex will be helpful in designing a good trip. To make our transversal even better, making reference to the experience of a pioneer will be a natural choice, as demonstrated in the following lemma.

LEMMA 4.15. *Let $\mathcal{A}$ be an* MNS *type algorithm. Let G be a graph with a unique clique tree $[\mu(1), \ldots, \mu(m)]$. Let σ be an* MNS *ordering of G with $\sigma(1) \in (\mu(1) \setminus \mu(2)) \cup (\mu(m) \setminus \mu(m-1))$ and let τ be the output of $\mathcal{A}+(G, \sigma)$. Then τ is compatible with the canonical clique ordering μ, and hence, by Remark 4.9, is a rigid interval ordering of G.*

Proof. Without loss of generality, suppose that $\sigma(1) \in \mu(m) \setminus \mu(m-1)$. It follows from Theorem 4.14 that $\sigma(n) \in \mu(1) \setminus \mu(2)$ and σ is left-compatible with μ', the reversal of μ. Applying Theorem 4.14 again yields τ is left-compatible with μ. It is now sufficient to show that for any $1 \leq i < j \leq n$ fulfilling $\ell_\mu(\tau(i)) = \ell_\mu(\tau(j))$ we have $r_\mu(\tau(i)) \leq r_\mu(\tau(j))$. The assumption $\ell_\mu(\tau(i)) = \ell_\mu(\tau(j))$ means that $\tau(j)$ lies in the MNS slice of τ when $\tau(i)$ is to be chosen. According to the rule of $\mathcal{A}+$, the reason that the $\mathcal{A}+$ slice of τ at time i consists of $\tau(i)$ itself can only be $\sigma^{-1}(\tau(i)) > \sigma^{-1}(\tau(j))$. Because σ is left-compatible with μ', we get $r_\mu(\tau(i)) = m + 1 - \ell_{\mu'}(\tau(i)) \leq m + 1 - \ell_{\mu'}(\tau(j)) = r_\mu(\tau(j))$ and so the proof is finished. □

Following the 3-sweep LBFS algorithm devised by Corneil [15] in recognizing unit interval graphs, we propose the following 3-sweep MNS algorithm.

Three-sweep MNS algorithm based on an MNS type algorithm $\mathcal{A}$: { Input: a graph G on n vertices; Output: a vertex ordering of G.} Step 1. Generate an MNS ordering δ of $V(G)$; Step 2. Do $\mathcal{A}(G, \delta(n))$ yielding sweep σ; Step 3. Do $\mathcal{A}+(G, \sigma)$ yielding sweep τ; Step 4. Output τ.

THEOREM 4.16. *The three-sweep* MNS *algorithm as described above outputs a rigid interval ordering when the input is a rigid interval graph and outputs a UI-ordering when the input graph is a connected unit interval graph.*

Proof. Let the unique clique tree of the input rigid interval graph G be $[\mu(1), \ldots, \mu(m)]$. It follows from Theorem 4.14 that $\sigma(1) = \delta(n) \in (\mu(1) \setminus \mu(2)) \cup (\mu(m) \setminus \mu(m-1))$. Now the result directly follows from Lemma 4.15 and Remark 2.18, as wanted. □

Recall Remark 3.10. Correspondingly, Theorem 2.4 and Theorem 4.16 together guarantees that the Three-sweep MNS algorithm can be used to recognize unit interval graphs and even reconstruct a unit interval representation when the input is a unit interval graph and all these can be done in linear time when $\mathcal{A}$ is taken to be LBFS. Let us provide a description of the linear time unit interval graph recognition algorithm where we assume that $\mathcal{A}$ and $\mathcal{A}+$ can be linearly implemented.

Three-sweep MNS algorithm for recognizing unit interval graphs based on an MNS type algorithm $\mathcal{A}$: { Input: a graph G on n vertices; Output: a statement declaring whether or not G is a unit interval graph.} Step 1. Generate an MNS ordering δ of $V(G)$; Step 2. Do $\mathcal{A}(G, \delta(n))$ yielding sweep σ; Step 3. Do $\mathcal{A}+(G, \sigma)$ yielding sweep τ; Step 4. If τ is a UI-ordering of G, then conclude that G is a unit interval graph; else, conclude that G is not a unit interval graph.

Theorem 4.10 along with Theorem 4.16 suggests that we can make use of the

Three-sweep MNS algorithm to recognize rigid interval graphs as long as we have a simple method to check whether or not a given vertex ordering is a rigid interval ordering. We do not pursue such a possible 3-sweep MNS recognition algorithm here; instead, let us give below a 4-sweep MNS recognition algorithm for rigid interval graphs.

Four-sweep MNS algorithm for recognizing rigid interval graphs based on an MNS type algorithm $\mathcal{A}$:
{ Input: a graph G on n vertices;
Output: a statement declaring whether or not G is a rigid interval graph.}
Step 1. Generate an arbitrary MNS ordering δ of $V(G)$;
Step 2. Do $\mathcal{A}(G, \delta(n))$ yielding sweep π;
Step 3. Do $\mathcal{A}+(G, \pi)$ yielding sweep σ;
Step 4. Do $\mathcal{A}+(G, \sigma)$ yielding sweep τ;
Step 5. If σ is a consecutive I-ordering of G and τ is consecutive, then conclude that G is a rigid interval graph; else, conclude that G is not any rigid interval graph.

The above algorithm can be easily implemented in linear time when taking $\mathcal{A}$ to be LBFS. Prior to proving the correctness of the algorithm for any MNS type algorithm $\mathcal{A}$, we prepare a simple lemma.

LEMMA 4.17. *Given as input a rigid interval graph G, the orderings σ and τ produced in executing the Four-sweep* MNS *algorithm as described above are both rigid interval orderings, and hence consecutive I-orderings, of $V(G)$.*

Proof. Applying Theorem 4.16 on the first three sweeps δ, π and σ shows that σ is a rigid interval ordering. Utilizing Theorem 4.16 again on the last three sweeps π, σ and τ we find that τ is also a rigid interval ordering. □

Here is another characterization of the rigid interval graphs, which in conjunction with Lemma 4.17 confirms the correctness of our 4-sweep MNS algorithm for recognizing rigid interval graphs.

THEOREM 4.18. *A graph G is a rigid interval graph if and only if it has two consecutive orderings σ and τ such that σ is an I-ordering, τ is an* MNS *ordering and $\tau(1) = \sigma(n)$.*

Proof. For the forward direction, we use the Four-sweep MNS algorithm to generate the orderings σ and τ of the vertex set of the rigid interval graph G and then Lemma 4.17 tells us that they are the required orderings.

We now turn to considering the backward direction. As observed in Remark 2.16, the I-ordering σ determines a clique path $[\mu(1), \ldots, \mu(m)]$ and σ is left-compatible with μ. Since σ is consecutive, it is indeed strongly left-compatible with μ and so we conclude from Lemma 4.7 that

$$(\mu(i+1) \cap \mu(i+2)) \setminus \mu(i) \neq \emptyset \tag{4.11}$$

for any $i \in [m-2]$. Let μ' be the the reversal of μ. Note that $\tau(1) = \sigma(n) \in \mu(m) \setminus \mu(m-1) = \mu'(1) \setminus \mu'(2)$ and thus we can derive from Lemma 4.12 that τ is left-compatible with μ'. Because τ is assumed to be consecutive, it is strongly left-compatible with μ' and so an application of Lemma 4.7 shows that

$$(\mu(m-i) \cap \mu(m-i-1)) \setminus \mu(m-i+1) = (\mu'(i+1) \cap \mu'(i+2)) \setminus \mu'(i) \neq \emptyset \tag{4.12}$$

for any $i \in [m-2]$. In view of Theorem 4.4, Eqs. (4.11) and (4.12) verify the validity of the backward direction, as desired. □

We note that unit interval graphs can be characterized by the existence of a so-called bicompatible elimination ordering [51, 71]. It would be interesting to investigate if there is any parallel vertex ordering characterization for rigid interval graphs, namely a characterization in terms of the properties of a vertex ordering and its reversal.

To conclude this section, let us try our hands at illustrating a connection between the Two-sweep MNS algorithm for recognizing unit interval graphs presented in §3 and the above Three-sweep MNS algorithm for recognizing unit interval graphs.

LEMMA 4.19. *Let G be a rigid interval graph on n vertices and let δ, σ, and τ be obtained by applying the Three-sweep* MNS *algorithm for recognizing unit interval graphs on G based on an* MNS *type algorithm $\mathcal{A}$. Then τ is an output of the algorithm $\mathcal{A}^{\triangle}(G, \sigma(n))$.*

Proof. By Lemma 4.15, τ is a rigid interval ordering and so we can assume that it is compatible with a canonical ordering μ of $\mathscr{C}(G)$. Take $i \in [2, n]$ and let $S = S_\tau(i)$ be the MNS slice of τ at time i. Let $\ell_\mu(\tau(i)) = p$ and $r_\mu(\tau(i)) = q$. Then, we have $\ell_\mu(x) = p$ and $r_\mu(x) \geq q$ for any $x \in S$. Consequently, we now get to

$$\deg_G(\tau(i)) = \min\{\deg_G(x) : \ x \in S\}.$$

Since this holds for all $i \geq 2$, it is clear that τ is an output of the algorithm $\mathcal{A}^{\triangle}(G, \sigma(n))$. □

5. Concluding remarks. In [15, p. 373] Corneil gives the following description of his 3-sweep LBFS algorithm for recognizing unit interval graphs:

> The first sweep is an arbitrary LBFS to find a "left-anchor". The following two LBFS sweeps, with a specific tie-breaking rule, provide an ordering of V that satisfies the "neighbourhood condition" if and only if G is a unit interval graph.

Our work here develops the idea of Corneil in several aspects. Firstly, we find that to get a UI-ordering of a unit interval graph, the first LBFS sweep in Corneil's algorithm can be replaced by an MNS sweep, the second LBFS+ sweep by another MNS sweep and the third LBFS+ sweep by an MNS+ sweep. Secondly, we find that the above multi-sweep algorithm applies essentially to rigid interval graphs, a superclass of unit interval graphs, where a rigid interval ordering should be in place of a UI ordering as a membership certificate (Theorem 4.16). Lastly, we have a better understanding of the role of each MNS sweep: Roughly speaking, the first sweep of MNS can capture an end-interval/ "left-anchor" (Theorem 4.14), a second sweep of MNS then sorts the left end-points of the intervals and gives us an I-ordering of those intervals (Lemma 4.12) and finally a third sweep of MNS sorts the right end-points for each group of intervals whose left end-points are nearby in certain sense and then reconstructs a required UI-ordering (Lemma 4.15). Note that we also earn more understanding of what is the "left-anchor" in a unit interval graph, a rigid interval graph, or even a general interval graph, and will report it among other results elsewhere [58].

A naive algorithm of recognizing rigid/unit interval graph is to peel off an MNS end-vertex once at a time, hoping this this process mimics removing each time either the left endpoint or the right endpoint of a line segment. This picture may be too far away from the reality as [28, Fig. 1] shows that removing an LBFS end-vertex from a rigid/unit interval graph may result in a graph which is not any rigid/unit interval graph any more.

All the multi-sweep graph search algorithms presented in this note can do the job by appealing to any MNS type algorithm and are thus quite flexible. But for an easy and efficient implementation of our algorithm, we indeed have to focus on those MNS

type algorithms with easy and/or linear implementation. It is interesting to discuss if there is any more natural MNS type algorithm besides LBFS which fulfils this criterion. Especially, we would like to know if there is any linear time implementation of LDFS, LDFS+ and LDFS$^{\triangle}$.

A two-sweep MNS certifying recognition algorithm for unit interval graphs is presented in §3. It may be interesting to further study those cases in which we get a claw as a certificate for nonmembership in unit interval graphs and seek for the possibility of modifying our algorithm to get a certifying algorithm for (rigid) interval graphs. This may call for a better understanding of rigid interval graphs and general interval graphs.

Interval graphs form a special class of circular-arc graphs. Much work has been done on recognizing circular-arc graphs and its subclasses [52, 60]. It may deserve to see if certain multi-sweep graph search algorithm can provide us simpler recognition algorithms for various circular-arc graphs.

Acknowledgments. The authors thank Derek Corneil, Wen-Lian Hsu, Jing Huang and Ziqing Xiang for their help and encouragement during the preparation of this paper. This work was supported by the National Natural Science Foundation of China (No. 10871128).

REFERENCES

[1] K. Asdre, S.D. Nikolopoulos, *A polynomial solution to the k-fixed-endpoint path cover problem on proper interval graphs*, Theoretical Computer Science, 411 (2010), pp. 967–975.
[2] K. Asdre, S.D. Nikolopoulos, *The 1-fixed-endpoint path cover problem is polynomial on interval graphs*, Algorithmica, 58 (2010), pp. 679–710.
[3] B. Aspvall, P. Heggernes, *Finding minimum height elimination trees for interval graphs in polynomial time*, BIT, 34 (1994), 484–509.
[4] J. Bang-Jensen, J. Huang, L. Ibarra, *Recognizing and representing proper interval graphs in parallel using merging and sorting*, Discrete Applied Mathematics, 155 (2007), pp. 442–456.
[5] A. Berry, J.R.S. Blair, J-P. Bordat, G. Simonet, *Graph extremities defined by search algorithms*, Algorithms, 3 (2010), pp. 100–124.
[6] R. van Bevern, C. Komusiewicz, H. Moser, R. Niedermeier, *Measuring indifference: Unit interval vertex deletion,* Lecture Notes in Computer Science, 6410 (2010), pp. 232–243.
[7] M. Blum, S. Kannan, *Designing programs that check their work*, Proceedings of the 21th Annual ACM Symposium on Theory of Computing (STOC' 89), (1989), pp. 86–97.
[8] K.S. Booth, G.S. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, Journal of Computer and System Sciences, 13 (1976), pp. 335–379.
[9] A. Brandstädt, V.D. Chepoi, F.F. Dragan, *Perfect elimination orderings of chordal powers of graphs*, Discrete Mathematics, 158 (1996), 273–278.
[10] A. Brandstädt, F.F. Dragan, F. Nicolai, *LexBFS-orderings and powers of chordal graphs*, Discrete Mathematics, 171 (1997), 27–42.
[11] A. Brandstädt, V.B. Le, J.P. Spinrad, *Graph Classes: A Survey*, SIAM Monographs on Discrete Mathematics and Applications, 3, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[12] J. Cáceres, O.R. Oellermann, *On 3-Steiner simplicial orderings*, Discrete Mathematics, 309 (2009), pp. 5828–5833.
[13] M.R. Cerioli, F.S. Oliveira, J.L. Szwarcfiter, *Extreme cliques in interval graphs*, Ars Combinatoria, 94 (2010), pp. 103–114.
[14] J-M. Chang, C-W. Ho, M-T. Ko, *LexBFS-ordering in asteroidal triple-free graphs*, Lecture Notes in Computer Science, 1741 (1999), pp. 163–172.
[15] D.G. Corneil, *A simple 3-sweep* LBFS *algorithm for the recognition of unit interval graphs*, Discrete Applied Mathematics, 138 (2004), pp. 371–379.
[16] D.G. Corneil, *Lexicographic Breadth First Search – A survey*, Lecture Notes in Computer Science, 3353 (2005), pp. 1–19.
[17] D.G. Corneil, *Private communication to Yaokun Wu*, August 9, 2011.

[18] D.G. Corneil, H. Kim, S. Natarajan, S. Olariu, A.P. Sprague, *Simple linear time recognition of unit interval graphs*, Information Processing Letters, 55 (1995), pp. 99–104.
[19] D.G. Corneil, E. Köhler, J-M. Lanlignel, *On end-vertices of Lexicographic Breadth First Searches*, Discrete Applied Mathematics, 158 (2010), pp. 434–443.
[20] D.G. Corneil, R.M. Krueger, *A unified view of graph searching*, SIAM Journal on Discrete Mathematics, 22 (2008), pp. 1259–1276.
[21] D.G. Corneil, S. Olariu, L. Stewart, *Asteroidal triple-free graphs*, SIAM Journal on Discrete Mathematics, 10 (1997), pp. 399–430.
[22] D.G. Corneil, S. Olariu, L. Stewart, *The ultimate interval graph recognition algorithm? (extended abstract),* Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 1998, pp. 175–180.
[23] D.G. Corneil, S. Olariu, L. Stewart, *Liear time algorithms for dominating pairs in asteroidal triple-free graphs*, SIAM Journal on Computing, 28 (1999), pp. 1284–1297.
[24] D.G. Corneil, S. Olariu, L. Stewart, *The* `LBFS` *structure and recognition of interval graphs*, SIAM Journal on Discrete Mathematics, 23 (2009), pp. 1905–1953.
[25] C. Crespelle, *Fully dynamic representations of interval graphs*, Lecture Notes in Computer Science, 5911 (2010), pp. 77–87.
[26] X. Deng, P. Hell, J. Huang, *Linear time representation algorithms for proper circular-arc graphs and proper interval graphs*, SIAM Journal on Computing, 25 (1996), pp. 390–403.
[27] F.F. Dragan, F. Nicolai, A. Brandstädt, `LexBFS`*-orderings and powers of graphs*, Lecture Notes in Computer Science, 1197 (1997), pp. 166–180.
[28] C.M.H. de Figueiredo, J. Meidanis, C.P. de Mello, *A linear-time algorithm for proper interval graph recognition*, Information Processing Letters, 56 (1995), pp. 179–184.
[29] P.C. Fishburn, *Interval Orders and Interval Graphs: A Study of Partially Ordered Sets*, John Wiley & Sons Inc., 1985.
[30] D.R. Fulkerson, O.A. Gross, *Incidence matrices and interval graphs*, Pacific Journal of Mathematics, 15 (1956), pp. 835–855.
[31] F. Gardi, *The Roberts characterization of proper and unit interval graphs*, Discrete Mathematics, 307 (2007), pp. 2906–2908.
[32] P.C. Gilmore, A.J. Hoffman, *A characterization of comparability graphs and of interval graphs,* The Canadian Journal of Mathematics, 16 (1964), pp. 539–548.
[33] M.C. Golumbic, *Reasoning about time*, in: *Mathematical Aspects of Artificial Intelligence*, F. Hoffman, ed., American Mathematical Society, Proc. Symposia in Appl. Math., 55, pp. 19–53, 1998.
[34] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, 2nd ed., Annals of Discrete Mathematics, 57, Elsevier, Amsterdam, The Netherlands, 2004.
[35] A. Gyárfás, *Combinatorics of intervals*, preliminary version, 2003.
[36] M. Habib, R.M. McConnell, C. Paul, L. Viennot, `Lex-BFS` *and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing*, Theoretical Computer Science, 234 (2000), pp. 59–84.
[37] P. Hanlon, *Counting interval graphs*, Transactions of the American Mathematical Society, 272 (1982), pp. 383–426.
[38] H. Hara, A. Takemura, *Boundary cliques, clique trees and perfect sequences of maximal cliques of a chordal graph*, arXiv:cs/0607055, 2006.
[39] P. Heggernes, D. Meister, C. Papadopoulos, *A new representation of proper interval graphs with an application to clique-width*, Electronic Notes in Discrete Mathematics, 32 (2009), pp. 27–34.
[40] P. Hell, J. Huang, *Lexicographic orientation and representation algorithms for comparability graphs, proper circular arc graphs, and proper interval graphs*, Journal of Graph Theory, 20 (1995), pp. 361–374.
[41] P. Hell, J. Huang, *Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs*, SIAM Journal on Discrete Mathematics, 18 (2005), pp. 554–570.
[42] P. Hell, R. Shamir, R. Sharan, *A fully dynamic algorithm for recognizing and representing proper interval graphs*, SIAM Journal on Computing, 31 (2001), pp. 289–305.
[43] J. Herzog, T. Hibi, X. Zheng, *Dirac's theorem on chordal graphs and Alexander duality*, European Journal of Combinatorics, 25 (2004), pp. 949–960.
[44] W.-L. Hsu, *A simple test for interval graphs*, Lecture Notes in Computer Science, 657 (1993), pp. 11–16. An updated version is available as: *A new test for interval graphs*, `http://iasl.iis.sinica.edu.tw/webpdf/paper-1992-A_New_Test_for_Interval_Graphs.pdf`
[45] W.-L. Hsu, $O(m \cdot n)$ *algorithms for the recognition and isomorphism problems on circular-arc graphs*, SIAM Journal on Computing, 24 (1995), pp. 411–439.
[46] W.-L. Hsu, *On-line recognition of interval graphs in* $O(m + n \log n)$ *time*, Lecture Notes in

Computer Science, 1120 (1996), pp. 27–38.
[47] W.-L. Hsu, T-H. Ma, *Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs*, SIAM Journal on Computing, 28 (1999), pp. 1004–1020.
[48] W.-L. Hsu, R.M. McConnell, *PC trees and circular-ones arrangements*, Theoretical Computer Science, 296 (2003), pp. 59–74.
[49] L. Ibarra, *A fully dynamic graph algorithm for recognizing proper interval graphs*, Lecture Notes in Computer Science, 5431 (2009), pp. 190–201.
[50] L. Ibarra, *A fully dynamic graph algorithm for recognizing interval graphs*, Algorithmica, 58 (2010), pp. 637–678.
[51] R.E. Jamison, R. Laskar, *Elimination orderings of chordal graphs*, in: K.S. Vijain, N.M. Singhi (Eds.), Proceedings of the Seminar on Combinatorics and Applications, ISI, Calcutta, 1982, pp. 192–200.
[52] H. Kaplan, Y. Nussbaum, *Certifying algorithms for recognizing proper circular-arc graphs and unit circular-arc graphs*, Discrete Applied Mathematics, 157 (2009), pp. 3216–3230.
[53] N. Korte, R.H. Möhring, *An incremental linear-time algorithm for recognizing interval graphs*, SIAM Journal on Computing, 18 (1989), pp. 68–81.
[54] D. Kratsch, R.M. McConnell, K. Mehlhorn, J.P. Spinrad, *Certifying algorithms for recognizing interval graphs and permutation graphs*, SIAM Journal on Computing, 36 (2006), pp. 326–353.
[55] R. Krueger, *Graph Searching*, Ph.D. thesis, University of Toronto, 2005.
[56] C.G. Lekkerkerker, J.C. Boland, *Representation of a finite graph by a set of intervals on the real line*, Fundamenta Mathematicae, 51 (1962), pp. 45–64.
[57] P. Li, Y. Wu, *A 4-sweep* `LBFS` *algorithm for recognizing interval graphs*, manuscript.
[58] P. Li, Y. Wu, *What is an extreme vertex in an interval graph?* manuscript.
[59] M.C. Lin, F.J. Soulignac, J.L. Szwarcfiter, *Short models for unit interval graphs*, Electronic Notes in Discrete Mathematics, 35 (2009), pp. 247–255.
[60] M.C. Lin, J.L. Szwarcfiter, *Characterizations and recognition of circular-arc graphs and subclasses: A survey*, Discrete Mathematics, 309 (2009), pp. 5618–5635.
[61] P.J. Looges, S. Olariu, *Optimal greedy algorithms for indifference graphs*, Computers & Mathematics with Applications, 25 (1993), pp. 15–25.
[62] W-F. Lu, W-L. Hsu, *A clustering algorithm for interval graph test on noisy data*, Lecture Notes in Computer Science, 2647 (2003), pp. 195–208.
[63] T. Ma, unpublished manuscript.
[64] H. Maehara, *On time graphs*, Discrete Mathematics, 32 (1980), pp. 281–289.
[65] Y. Matsui, R. Uehara, T. Uno, *Enumeration of the perfect sequences of a chordal graph*, Theoretical Computer Science, 411 (2010), pp. 3635–3641.
[66] R.M. McConnell, K. Mehlhorn, S. Näher, P. Schweitzer, *Certifying algorithms*, Computer Science Review, 5 (2011), pp. 119–161.
[67] T.A. McKee, F.R. McMorris, *Topics in Intersection Graph Theory*, SIAM Monographs on Discrete Mathematics and Applications, 2, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[68] D. Meister, *Recognition and computation of minimal triangulations for AT-free claw-free and co-comparability graphs*, Discrete Applied Mathematics, 146 (2005), pp. 193–218.
[69] R.H. Möhring, *Algorithmic aspects of comparability graphs and interval graphs*, in: I. Rival, editor, Graphs and Orders, pp. 41–101, D. Reidel, Boston, 1985.
[70] B.S. Panda, S.K. Das, *A linear time recognition algorithm for proper interval graphs*, Information Processing Letters, 87 (2003), pp. 153–161.
[71] B.S. Panda, S.K. Das, *A parallel algorithm for generating bicompatible elimination orderings of proper interval graphs*, Information Processing Letters, 109 (2009), pp. 1041–1046.
[72] I. Peér, R. Shamir, *Satisfiability problems on intervals and unit intervals*, Theoretical Computer Science, 175 (1997), pp. 349–372.
[73] G. Ramalingam, C.P. Rangan, *A uniform approach to domination problems on interval graphs*, Information Processing Letters, 27 (1988), pp. 271–274.
[74] G. Ramalingam, C.P. Rangan, *New sequential and parallel algorithms for interval graph recognition*, Information Processing Letters, 34 (1990), pp. 215–219.
[75] I. Rapaport, K. Suchan, I. Todinca, *Minimal proper interval completions*, Information Processing Letters, 106 (2008), pp. 195–202.
[76] A. Raychaudhuri, *On powers of interval and unit interval graphs*, Congressus Numerantium, 59 (1987), pp. 235–242.
[77] F.S. Roberts, *Indifference graphs*, F. Harary (Ed.), Proof Techniques in Graph Theory, Academic Press, New York, (1969), pp. 139–146.
[78] F.S. Roberts, *On the compatibility between a graph and a simple order*, Journal of Combina-

torial Theory, Ser. B, 11 (1971), pp. 28–38.
[79] D.J. Rose, R.E. Tarjan, G.S. Lueker, *Algorithmic aspects of vertex elimination on graphs*, SIAM Journal on Computing, 5 (1976), pp. 266–283.
[80] K. Simon, *A new simple linear algorithm to recognize interval graphs*, Lecture Notes in Computer Science, 553 (1992), pp. 289–308.
[81] K. Simon, *A note on lexicographic breadth first search for chordal graphs*, Information Processing Letters, 54 (1995), pp. 249–251.
[82] J.P. Spinrad, *Efficient Graph Representations*, Fields Institute Monographs, 19, American Mathematical Society, 2003.
[83] J.P. Spinrad, *Efficient implementation of lexicographic depth first search*, in preparation.
[84] R.E. Tarjan, *Maximum cardinality search and chordal graphs,* unpublished lecture notes for CS 259, Stanford University, 1976.
[85] R.E. Tarjan, M. Yannakakis, *Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM Journal on Computing, 13 (1984), pp. 566–579.
[86] R.E. Tarjan, M. Yannakakis, *Addendum: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM Journal on Computing, 14 (1985), pp. 254–255.
[87] G. Wegner, *Eigenschaften der nerven homologische-einfactor familien in R^n*, Ph.D. thesis, Universität Gottigen, 1967.