Unit Interval Graphs, Properties and Algorithms

Sayyed Bashir Sadjad Hamid Zarrabi-Zadeh

School of Computer Science
University of Waterloo
{bssadjad, hzarrabi}@uwaterloo.ca

February 19, 2004

1 Introduction

In this article we address some of the results on *unit interval graphs*. In short, a unit interval graph is an interval graph in which all intervals have the same length. These graphs have many applications in bioinformatics, databases, scheduling, measurement theory, etc. [KS96, BFMY83, PY79, Gol80]. We will study some of the problems that arises from such applications later in this article, after addressing properties of this class and available recognition algorithms.

There are some other classes of graphs with different definitions which are in fact the same as the class of unit interval graphs. We introduce them in Section 2 and mention theorems that prove their equivalence. There are some efficient algorithms to check whether a graph is unit interval graph. Some of these algorithms are addressed in Section 3. A method for constructing efficient representations of unit interval graphs is given in Section 4. Finally, in Section 5 we investigate some problems on unit interval graphs that have faster algorithms than the general case. We also mention an interesting application of this class in bioinformatics in that section.

2 Definitions and Properties

We start this section by giving three different definitions of three graph classes and then mention theorems showing that all these three classes are the same. When there are short proofs for the claims, we will give details of them, but for long proofs we only give references. Throughout this article we assume that the graphs are simple (with no loops or multiple edges) and connected with at least two vertices. To show an edge between vertices v and w we use (v, w) whether G is directed or undirected. So, when it is clear from the context that G is undirected, (v, w) and (w, v) are the same edges.

Definition 2.1 *An undirected graph G is an* **interval graph** *if there is a mapping f from vertices of G to a set I of real intervals such that $(v, w) \in E(G)$ if and only if $f(v) \cap f(w) \neq \phi$. Such a set I is called an* **interval model** *for G.*

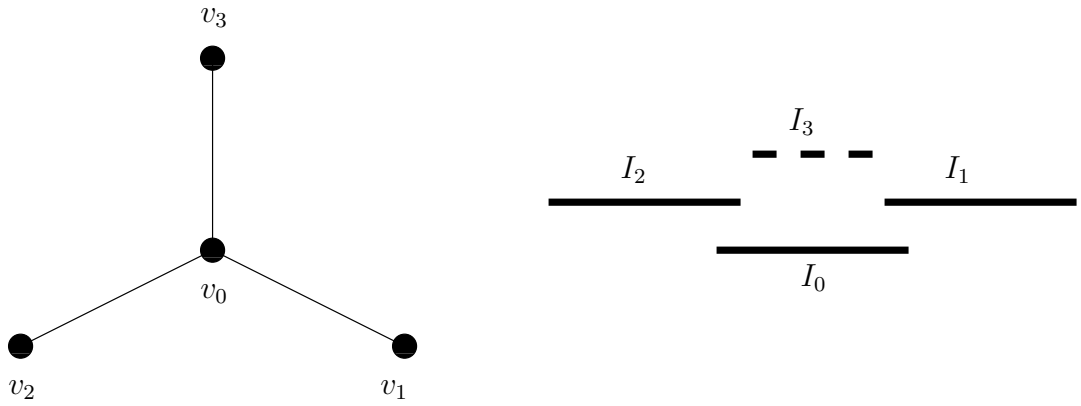

Figure 1: A proper or unit interval graph cannot contain a $K_{1,3}$.

An interval graph is both *chordal* and *co-comparability* [Gol80]. On the other hand, if a graph is both chordal and co-comparability, then it is an interval graph. Therefore a nice characterization of the interval graphs is the following theorem:

Theorem 2.2 (Golumbic [Gol80]) *A graph G is an interval graph if and only if G is chordal and $\bar{G}$ is a comparability graph.*

Definition 2.3 *An interval graph G is a* **unit interval graph** *if there exists an interval model I for G in which all intervals are of equal size. In this case, I is called a* **unit interval model** *for G.*

Definition 2.4 *An interval graph G is a* **proper interval graph** *if there exists an interval model I for G in which no interval properly contains another interval. In this case, I is called a* **proper interval model** *for G.*

It is not hard to show that the class of proper interval graphs is the same as the class of unit interval graphs. First of all, we show that a graph in any of these classes could not contain a $K_{1,3}$ as an induced subgraph.

Theorem 2.5 *If G is a unit interval graph or a proper interval graph then G contains no $K_{1,3}$ as an induced subgraph.*

Proof: The proof is given in Figure 1. In this figure, a $K_{1,3}$ is shown on the left and an interval model for it given on the right (I_i is the interval of v_i). As it is shown, I_3 should be a proper subset of I_0, so, $K_{1,3}$ is not a proper interval graph. Moreover, it is clear that I_0 and I_3 cannot be of equal size. Therefore, $K_{1,3}$ is not a unit interval graph. □

The following theorem shows that this condition is also sufficient and implies that the class of unit interval graphs is the same as proper interval graphs. This theorem is from [BW99] but an extended version of it, which will be addressed later, is in [Gol80, BLS99, Rob69].

Theorem 2.6 *For a simple graph G the following three statements are equivalent:*

1. *G is a unit interval graph.*
2. *G is an interval graph with no $K_{1,3}$ as induced subgraph.*
3. *G is a proper interval graph.*

Proof: The proof is by going from (2) to (3) and then from (3) to (1) (we know that (1) to (2) holds). To show that (2)⇒(3), we choose an interval model I for G, in which every vertex v of G is represented by an interval I_v of I. Starting from left to right, for any proper inclusion we move the interval with start point at right, to right without violating any edge condition. Suppose that there are two intervals $I_x = [a, b]$ and $I_y = [c, d]$ such that $a < c \leq d \leq b$ (that is, I_x properly contains I_y). Since there is no induced $K_{1,3}$ in G, in at least one of intervals $[a, c]$ and $[d, b]$ there is no endpoint of an interval I_z that does not intersect I_y. So, we can change the intervals I_x and I_y, without violating any edge condition, so that there is no proper inclusion. A similar idea works for going from (3) to (1). Here for each interval, we do some expansion and shrinking to make it unit and continue with the interval on the right. □

Now, we define a third class of graphs which comes from measurement theories (attributed in [BW99] and [PD03] to [SS58]) and has close relations to unit and proper interval graphs. We have already seen a close connection between comparability graphs and interval graphs. We will see a closer relation between unit interval graphs and semiorders, which are a sub-class of partial orders. The following definition of semiorders is from [Gol80].

Definition 2.7 *A partial order $S = (V, P)$ with elements V and a binary order relation P (on V) is* **semiorder***, if there is a function $f : V \to \mathbb{R}$ from V to real numbers s.t. $(u, v) \in P$ if and only if $f(u) \geq f(v) + 1$.*

In the above definition, there could be any $\delta > 0$ instead of 1 in the condition $f(u) \geq f(v) + 1$. In Theorem 2.8, it is shown that all these definitions are equivalent. Historically, this is the root of unit interval graphs and Theorem 2.8 was proved in [Rob69] many years before [BW99].

Theorem 2.8 (Roberts [Rob69]) *Let $G = (V, E)$ be an undirected graph, then the following statements are equivalent:*

1. *There exists a function $f : V \to \mathbb{R}$ s.t.*

$$(u, w) \in E(G) \Leftrightarrow |f(u) - f(w)| < 1.$$

2. *There is a semiorder on $S(V, P)$ on the vertices of V such that*

$$(u, w) \in P \Leftrightarrow (u, w) \notin E(G).$$

3. *$\bar{G}$ is a comparability graph and every transitive orientation of $\bar{G}$ is a semiorder.*
4. *G is an interval graph with no induced $K_{1,3}$.*
5. *G is a proper interval graph.*

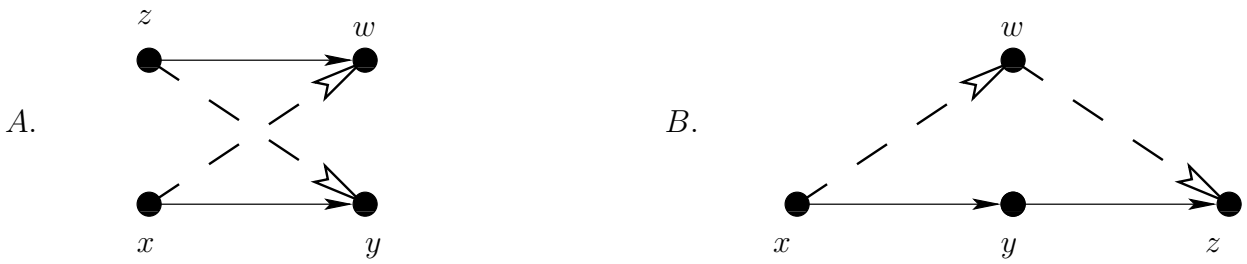

Figure 2: A representation of Theorem 2.9

6. G is a unit interval graph.

In [Rob69] it is mentioned that the (4)⇒(2) part was first observed by Marley through a personal communication with Roberts. Also, from historical point of view, the following theorem of Scott and Suppes [SS58] on semiorders is interesting since it is exactly the condition of having no $K_{1,3}$ or induced C_k (for $k \geq 4$) in the undirected complement of the relation. This is from [Gol80] but is attributed to [SS58].

Theorem 2.9 (Scott and Suppes [SS58]) *A partial order relation S defined on V is a semiorder if the following conditions hold:*

1. *S is irreflexive.*
2. *$(x, y) \in S$ and $(z, w) \in S$ imply $(x, w) \in S$ or $(z, y) \in S$.*
3. *$(x, y) \in S$ and $(y, z) \in S$ imply $(x, w) \in S$ or $(w, z) \in S$.*

To see why these conditions are the same as the previous condition (interval graph with no $K_{1,3}$), it is easy to check that condition (2) is equivalent to the condition that there is no C_k (for $k \geq 4$) in $\bar{G}$, where (u, w) is an edge in $G = (V, E)$ if and only if neither (u, w) nor (w, u) is in S. This is shown in part A of Figure 2. Notice that in this figure, if there is no other directed edges (relation) between x, y, w and z, then in G there will be a C_4 with no chord. On the other hand, notice that any other relation (i.e. other than xSy and zSw) then because of transitivity one of the white arrows should be present in S as well. So this is the necessary and sufficient condition for G to be chordal. Since S is a partial order, $\bar{G}$ is both chordal and co-comparability, and hence G is interval.

On the other hand, condition (3) is equivalent to the condition that there is no induced $K_{1,3}$ in G (Figure 2 part B). Notice that since S is a partial order it is transitive so if xSy and ySz then xSz and the precondition of (3) means that x, y, z forms a clique in $\bar{G}$ and the postcondition is that any vertex w in V should be connected to at least one of them in $\bar{G}$ which means G does not contain any induced $K_{1,3}$.

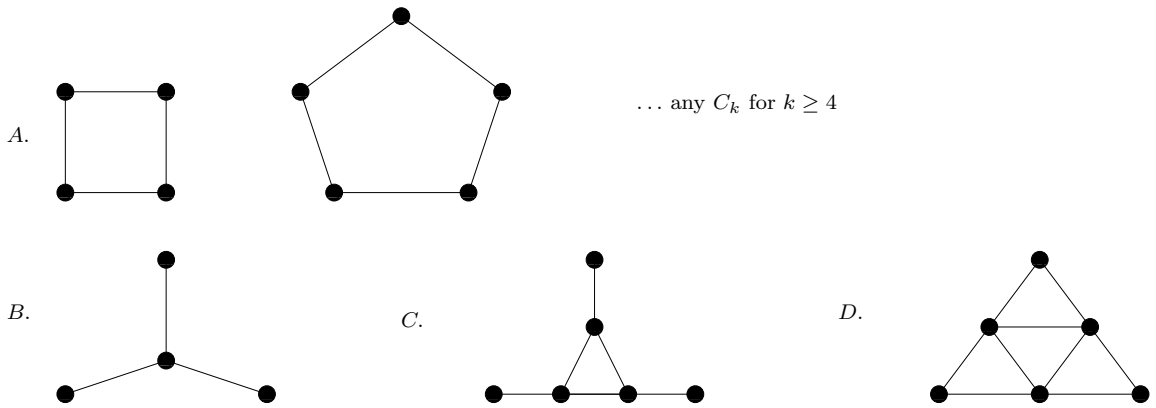

Figure 3: A graph G is unit interval iff it does not contain any of these graphs as an induced subgraph.

Another interesting historical observation is that in his paper, Roberts [Rob69] directly classifies proper interval graphs, rather than using conditions on chordal and co-comparability graphs. Starting from a graph G, define two vertices v and w to be *similar* if all neighbors of v and w are the same. It is clear that similarity is an equivalence relation. The graph G^* is the same as G with all similar vertices united in one vertex, i.e. each equivalence relation in similarity relation has one representative. The first representation theorem of Roberts is the following (translated in our context):

Theorem 2.10 (Roberts [Rob69]) *A graph G is unit interval graph if and only if G does not contain any of graphs in Figure 3 as induced sub-graph.*

Notice that no chordal graph contains an induced sub-graph of type A in Figure 3; graph B is $K_{1,3}$ and graphs C and D are not co-comparability.

3 Recognition Algorithms

There are several linear-time algorithms for recognizing interval graphs. The first such algorithm proposed by Booth and Lueker [BL76], employs a special data structure called PQ-tree to recognize interval graphs. Later, Korte and Möhring [KM89] proposed a modified version of PQ-trees called MPQ-trees, and present a simpler linear-time algorithm for interval graph recognition that works in an incremental fashion. Corneil, Olariu and Stewart [COS98] presented another linear-time recognition algorithm without using PQ-trees. Their algorithm recognizes interval graphs by five iterations of lexBFS, where different rules for tie breaking are applied in each iteration.

For unit interval graph recognition, one can first check whether the given graph is an interval graph, and then recognize it as a unit interval graph using special properties of unit interval graphs. The linear-time recognition algorithm proposed by Looges and Olariu [LO92] (attributed in [CKN$^+$95]) is an example of this approach. However, the resulting algorithm is not so simple,

because algorithms for interval graph recognition are not simple, especially those that involve PQ-trees.

There are, on the other hand, some linear-time algorithms that directly recognize a unit interval graph, without going through the interval graph recognition step. The first such algorithm (and in fact the simpler one) is proposed by Corneil et al. [CKN$^+$95]. Their algorithm is very simple and can be implemented easily, without any special data structure. In the following subsection, we give more details of this algorithm.

3.1 The algorithm of Corneil et al.

The algorithm of Corneil et al. simply uses two runs of Breath-First-Search to recognize a unit interval graph. To describe the algorithm, we need to define some and notations terms here: The *open neighborhood* of a vertex v, denoted by $N(v)$, is the set of all vertices adjacent to v. The *closed neighborhood* of v is defined as $N[V] = N(v) \cup \{v\}$. Two vertices v and u are *indistinguishable* if $N[v] = N[u]$. For a BFS of G, we denote by L_k the set of all vertices at level k of the BFS tree. For a vertex $v \in L_k$, we denote by $d^-(v)$ the number of neighbors of v at level $k-1$, and by $d^+(v)$ the number of neighbors of v at level $k+1$.

Given an interval v, we denote by v_L and v_R the coordinates of the left and right endpoints of v, respectively. Let $I = \{v_1, v_2, \ldots, v_n\}$ be a unit interval model for G. The sequence $v_1, v_2, \ldots, v_n$ is a *natural labeling* for I, if $v_{iL} \leq v_{(i+1)L} (1 \leq i < n)$. A vertex v of G is called a *left anchor*, if it can be labeled as v_1 in a natural labeling for some unit interval model of G.

Note that if $v_1, v_2, \ldots, v_n$ is a natural labeling for I, then v_n can be considered as a left anchor, too. This is because the mirror of model I (which can be obtained by replacing each interval $[a, b]$ by $[-b, -a]$) is a valid unit interval model that has v_n as its first interval.

The following property helps us to find a left anchor of given unit interval graph G:

Theorem 3.1 (Corneil et al. [CKN$^+$95]) *Let I be a unit interval model and $v_1, v_2, \ldots, v_n$ be a natural labeling for I. Let t be the last nonempty level in the BFS starting from v_1. Then $v_n \in L_t$, and among all intervals in L_t, v_n has minimum degree. Furthermore, every vertex z in L_t such that $\deg(z) = \deg(v_n)$ is indistinguishable form v_n.*

Using the above theorem and the fact that in a natural labeling, v_n is also a left anchor, the following procedure gives a left anchor for a given unit interval graph G:

Procedure GET-LEFT-ANCHOR(G)

1. perform a BFS on G, starting from an arbitrary vertex $v_s \in V(G)$.
2. let t be the last level of the BFS.
3. find a vertex $z \in L_t$ with minimum degree among all vertices in L_t.
4. **return** z

Theorem 3.1 is not applicable directly to the BFS on G in the above algorithm (because it starts from an arbitrary vertex v_s), However, if I is a unit interval model for G with a natural labeling $v_1, v_2, \ldots, v_n$, then v_s is a left anchor for each of two sets $I' = \{v_1, v_2, \ldots, v_s\}$ and $I'' = \{v_s, v_{s+1}, \ldots, v_n\}$. By applying Theorem 3.1 on both I' and I'', Chan et al. showed that the output of procedure GET-LEFT-ANCHOR, z, is indistinguishable form either v_1 or v_n, so, it is a left anchor [CKN$^+$95]. Finally, they gave the following algorithm for unit interval graph recognition:

Algorithm RECOGNIZE (G)

1. $z \leftarrow$ GET-LEFT-ANCHOR(G)
2. perform a BFS on G starting from z.
3. order vertices of G as $v_1, v_2, \ldots, v_n$ based on
   i) their level in the BFS, and
   ii) among the vertices in the same level, on increasing order of $d^+(v) - d^-(v)$.
4. **for** all $v \in V$ **do**
5. let $\alpha(v) = \min\{i : v_i \in N[v]\}$ and $\gamma(v) = \max\{i : v_i \in N[v]\}$.
6. **if** $\gamma(v) - \alpha(v) \neq deg(v)$ **then**
7. **return** false
8. **return** $v_1, v_2, \ldots, v_n$

The algorithm is based on the following characterization of unit interval graphs:

Theorem 3.2 (Roberts [Rob71]) *A graph G is a unit interval graph if and only if there is an order on vertices such that for each vertex v, the closed neighborhood of v is a set of consecutive vertices in that order.*

The order defined in Theorem 3.2 is called a *consecutive order* of G. Chan et al. showed that if G is a unit interval graph, then the order found in line 3 of algorithm RECOGNIZE must be a consecutive order [CKN+95]. Theorem 3.2 expresses an important characterization of unit interval graphs, which is the base of some other unit interval graph recognition algorithms, such as [dFMdM95].

3.2 Other Recognition Algorithms

There are some other linear-time algorithms for unit interval graph recognition. In 1995, de Figueiredo, Meidanis and de Mello [dFMdM95] gave another linear-time recognition algorithm based on Theorem 3.2. Their algorithm finds a consecutive order of vertices (which is called *indifference order* in their paper) using just one iteration of lexBFS. Although their algorithm is not simpler than what presented in previous section, it constructs a compact structure for representing all indifference orders of the given unit interval graph. This representation, in fact, provides a way for solving the isomorphism problem on this class of graphs.

Another linear-time recognition algorithm is proposed by Deng, Hell and Huang [DHH96]. Their algorithm is mainly proposed to recognize and represent proper circular-arc graphs, by characterizing proper circular-arc graphs with local tournaments. However, since their algorithm needs to recognize parts of input as proper interval graph, it also yields a recognition algorithm for proper interval graphs. The resulting algorithm is linear and based on a new orientation characterization of proper interval graphs.

Recently, Panda and Das [PD03] have proposed a linear-time recognition algorithm which is based on finding a special ordering of vertices called bicompatible elimination ordering. A perfect elimination ordering of graph G is called a *bicompatible elimination ordering (BCO)*, if its reverse is also a perfect elimination ordering. They use a characterization of unit interval graphs that says a graph G is a unit interval graph, if and only of it has a BCO. To check whether graph G is a unit

interval graph, their algorithm tries to find a BCO in G in linear time. Although their algorithm is not simpler than Corneil et al.'s one, the BCO resulted from their algorithm provides a new way to find a Hamiltonian cycle in the given unit interval graph. We will address this result in Section 5.3.

4 Representing Unit Interval Graphs

We know that an interval graph with n vertices can be represented by an interval model in which all endpoints are distinct integers between 1 and $2n$. We call this a *standard interval model*. Proper interval graphs can be represented in standard interval model, however, such representation is not possible, in general, for unit interval graphs. Suppose, for example, that in a unit interval model, the leftmost interval v_1 is starting at position 1 and the second interval v_2 is starting at position 2. Now, if we have two nonadjacent intervals that are both adjacent to v_2 but not to v_1, there is no integer position to start both of them, because at least one of them should be started exactly between v_{1R} and v_{2R}. One solution to this problem is to add a new endpoint halfway between v_{1R} and v_{2R}, and then scale the model in order to make all endpoints integer. Repeating this operation, however, may lead to intervals with exponential lengths that do not obey the notion of *efficient representation*, as specified in [Spi03].

The recognition algorithm of Corneil et al [CKN$^+$95] provides an intelligent method for obtaining a representation in which all endpoints are integers and the length of all intervals are at most n. This is indeed an efficient representation, because intervals have polynomial (in this case $O(n)$) length. We now briefly explain the method.

Suppose that the ordering $v_1, v_2, \ldots, v_n$ is produced by algorithm RECOGNIZE(G) by performing a BFS starting from a left anchor z of G (see Section 3.1). We make a tree T from this ordering by putting an edge between each vertex v_i (excluding z) and the earliest neighbor of v_i (i.e. $v_{\alpha(v_i)}$). Now, we perform a postorder traversal on T, breaking ties by choosing smaller numbered neighbor first, that is, if v_i is adjacent to two unvisited vertices v_j and v_k so that $j < k$, then we proceed postorder traversal from v_i to vj, before going from v_i to v_k; therefore, in this case, we have postorder(v_j)<postorder(v_k).

Now we can construct the unit interval model as follows: for every vertex v in level k of BFS, we create an interval with two endpoints $v_L = kn+$ postorder(v) and $v_R = v_L + n$. It is obvious that all intervals created in this way are of equal size and all endpoints of them are integers not exceeding n^2. Corneil et al. [CKN$^+$95] showed that the model obtained from this method is a valid unit interval model.

5 Problems on Unit Interval Graphs

In this section we briefly address some problems that are intractable for general graphs, but efficient algorithms for them exist on unit interval graphs.

5.1 Clique, Coloring and Independent Set

We know that the class of interval graphs, and consequently the class on unit interval graphs, are subclasses of the class of chordal graphs [Gol80]. Therefore, the algorithms available for chordal graphs can be suitably used for unit interval graphs; especially those that are linear-time.

To solve Clique, Coloring and Independent Set problems, therefore, it is sufficient to find a perfect elimination order (PEO) in the given unit interval graph G using lexBFS or MCS (Maximum Cardinality Search) in linear time [Gol80]. Coloring problem, then, can be solved in linear time by applying the well-known greedy algorithm for vertex coloring on the PEO. This algorithm also finds a cliques of size $\chi(G)$, simultaneously, and since G can not have any bigger clique, Clique problem is also solved at the same time. Independent Set problem is solvable in linear time by applying the greedy algorithm for independent sets on the reverse order of the PEO [Gol80].

5.2 Dominating Set

Let $G = (V, E)$ be a graph and A be a subset of V. A is called a *dominating set* of G if $N[A] = V$, where $N[A] = \cup_{v \in A} N[v]$. In other words, A is a dominating set of G, if every vertex in $V - A$ is adjacent to at least one vertex in A. The problem of finding a dominating set with minimum number of vertices is NP-Complete [GJ79]. There are some other variations of Dominating Set problem: Let A be a dominating set of G. A is called an *independent dominating set*, if $G[A]$ is an independent set; it is called a *connected dominating set*, if $G[A]$ is connected; and it is called a *total dominating set*, if $G[A]$ has no isolated vertex. If the vertices of G are assigned a real weight, then the weight of a set of vertices is the sum of the weights of all vertices in that set. The Weighted Domination Set problem is then the problem of finding a dominating set of minimum weight in a weighted graph. Weighted independent, connected and total dominating set problems are defined similarly.

The problems of finding minimum-cardinality and minimum-weight dominating set are NP-Complete for all variations of dominating set problem [GJ79]. The problem remains NP-Complete even for chordal graphs and some sub-classes of chordal graphs, like split graphs [BCD98]. However, linear-time algorithm exists for Dominating Set problem on interval graphs. Ramalingam and Pandu Rangan [RR88] proposed linear-time algorithms for solving all dominating-like problems on interval graphs in a unified approach. This approach is applicable for unit interval graphs, as well. Later, Chang [Cha98] proposed a new unified approach that solves Weighted Dominating Set problem as well as independent and connected versions of the problem in $O(n)$ time, provided a standard interval model of G is given (i.e. an interval model in which all endpoints are integers between 1 to $2n$). Notice that such a model is not available for unit interval graphs, as stated in section 4. Therefore, to use the algorithms of Chang on a unit interval graph G, we should use a proper interval model of G, instead.

5.3 Hamiltonian Path and Hamiltonian Cycle

The problem of deciding whether a graph has a Hamiltonian cycle is well known to be NP-Complete [GJ79]. It remains NP-Complete for many classes of graphs, including chordal graphs [Mul96]. (In fact, [Mul96] showed that Hamiltonian Cycle problem is NP-Complete even for some more restricted subclasses of chordal graphs, such as strongly chordal split graphs). The problem of deciding whether a graph has a Hamiltonian path is also NP-Complete.

Surprisingly, Both Hamiltonian Cycle and Hamiltonian Path problems are polynomial solvable for the class of interval graphs and unit interval graphs. Keil introduced a linear-time algorithm for Hamiltonian Cycle problem in interval graphs [Kei85]. Solving Hamiltonian problems is much easier for the class of unit interval graphs. Bertossi [Ber83] proved that a unit interval graph has a Hamiltonian path if and only if it is connected. He also proved a condition under which a

unit interval graphs has a Hamiltonian cycle. He then gave an $O(n \log n)$ algorithm for finding a Hamiltonian cycle in a unit interval graph. Chen, Chang and Chang [CCC97] derived the following more general condition for Hamiltonian problems on unit interval graphs:

Theorem 5.1 (Chen et al. [CCC97]) *Let $G = (V, E)$ be a unit interval graph of $n \geq 4$ vertices, then:*

1. *G has a Hamiltonian path, if and only if G is 1-connected;*
2. *G has a Hamiltonian cycle, if and only if G is 2-connected;*
3. *G is Hamiltonian-connected, if and only if G is 3-connected.*

(Recall that a graph G is Hamiltonian-connected if there is a Hamiltonian path between any two distinct vertices of G).

Since connectivity and biconnectivity properties of graphs can be tested in linear time using some variations of Depth-First-Search [CLRS01], Theorem 5.1 immediately leads to linear-time algorithms for testing whether a unit interval graph has a Hamiltonian path or cycle. Chen et al. [CCC97] also provide simple methods for retrieving a Hamiltonian path and a Hamiltonian cycle in a given unit interval graph G, provided a consecutive order of G is available. (such an order can be obtained by linear-time algorithms of [CKN$^+$95] and [dFMdM95]).

Alternative algorithms for finding Hamiltonian path and Hamiltonian cycle in unit interval graphs has been recently proposed by Panda and Das [PD03]. As we mentioned in Section 3.2, the algorithm of Panda and Das recognize a unit interval graph by finding a bicompatible elimination ordering (or BCO) of it. The resulting BCO immediately provides a Hamiltonian path, according to the following Theorem:

Theorem 5.2 (Panda and Das [PD03]) *If $v_1, v_2, \ldots, v_n$ is a BCO of a connected unit interval graph G, then $P = (v_1, v_2, \ldots, v_n)$ is a Hamiltonian path of G.*

Moreover, if the input graph G is biconnected, the following theorem can be used to find a Hamiltonian cycle in G:

Theorem 5.3 (Panda and Das [PD03]) *Let G be a biconnected unit interval graph and let $\alpha = (v_1, v_2, \ldots, v_n)$ be a BCO of G. Then $deg_{G_i}(v_i) \geq 2$ for all i ($1 \leq i \leq n-2$), where $G_i = G[v_i, v_{i+1}, \ldots, v_n]$.*

Using the above theorem, the following algorithm finds a Hamiltonian cycle C in the given biconnected unit interval graph G:

Algorithm HAMILTONIAN-CYCLE (G)

1. Let $\alpha = (v_1, v_2, \ldots, v_n)$ be a BCO of G
2. $C \leftarrow \{v_1v_2, v_2v_3, \ldots, v_{n-1}v_n\}$
3. $r \leftarrow 1$
4. $found \leftarrow$ false

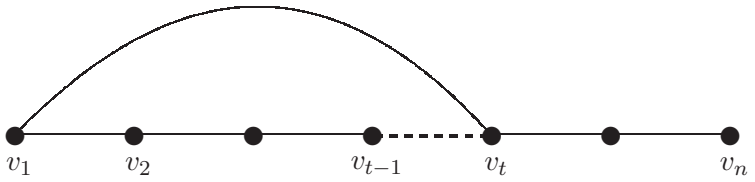

Figure 4: Illustrating the first iteration of HAMILTONIAN-CYCLE algorithm.

5. **while** not $found$ **do**
6. Let t be the largest index such that $v_r v_t \in E(G)$
7. **if** $t \neq n$ **then**
8. $C \leftarrow C \backslash \{v_{t-1}v_t\} \cup \{v_r v_t\}$
9. $r \leftarrow t-1$
10. **else**
11. $C \leftarrow C \cup \{v_r v_n\}$
12. $found \leftarrow$ true
13. **return** C

Throughout the algorithm, C contains a Hamiltonian path in G, which starts from a vertex v_r (r stores the index of this starting vertex throughout the algorithm) and ends at the vertex v_n. In the beginning of the algorithm, we initialize C by a Hamiltonian path between v_1 and v_n (obtained by Theorem 5.2), and set $r = 1$. In the first iteration of the main loop, if v_1v_n exists (i.e. $t = n$ in line 6 of the algorithm), then we add v_1v_n to C and reach a Hamiltonian cycle, and exit the loop. Otherwise (i.e. $t < n$), we modify C to be a new Hamiltonian path between v_{t-1} and v_n, by removing $v_{t-1}v_t$ and adding $v_r v_t$ to C (see Figure 4). Then, we set $r = t-1$ and iterate. According to Theorem 5.3, for every vertex v_r, there is always an adjacent vertex v_t such that $t - r \geq 2$. It means that in each iteration of the algorithm, we move the start point of our Hamiltonian path, v_r, at least one point forward toward v_n. Therefore, after at most $n-2$ iterations, we reach a vertex v_r ($r \leq n-2$) that is adjacent to v_n. Adding $v_r v_n$ to C leads to a Hamiltonian cycle of the given graph.

Remark: A linear-time algorithm is proposed by Brandstädt, Dragan and Köhler [BDK00] for the Hamiltonian Path and Hamiltonian Cycle problems on the class of (claw-net)-free graphs, i.e. the class of graphs that have neither an induced claw nor an induced net (in Figure 3, graphs B and C illustrate a claw and a net, respectively). According to Theorem 2.10, every unit interval graph is a (claw-net)-free graph. However, the class of (claw-net)-free graphs is more general and contains a couple of other graph classes such as proper circular-arc graphs and claw-free AT-free graphs.

5.4 An Application in Molecular Biology

In this section we study the model of Kaplan and Shamir for solving some problems in molecular biology [KS96]. First we notice that the following problem is NP-Complete [GJ79].

Problem 5.4 (Interval Graph Completion Problem) *Given a graph $G = (V, E)$ and a non-negative integer k. The problem is to answer whether there is an interval super-graph $G' = (V, E')$ of G where $|E' - E| \leq k$.*

Since there are efficient algorithms to check whether a given graph is interval, this problem is solvable in polynomial time if k is a given constant. A similar problem can be defined for subgraphs of a given graph (i.e. find the largest subgraph of G which is interval). These problems can be defined for proper (or unit) interval graphs as well. According to [KS96] all these problems are NP-complete when k is not constant. The reason that these problems are important in molecular biology is the following application: To study a genome, usually several copies of that genome are broken in shorter segments (these segments are called *clone*), for further study these clones are preserved but all information about their position in the DNA sequence is lost. On the other hand in physical mapping of DNA's this order should be reconstructed and to do this, experimental data on overlapping of these clones is used. Imagine the graph of all clones in which two clones are adjacent if their overlap (we call this *overlap graph*). It is clear that this should be an interval graph given that the outcome of experiments are accurate and complete, i.e. there is an overlap in the original DNA between two clones if and only if in the experimental data this overlap has been observed. The fact is that in real applications the experimental data are not absolutely precise. In other words based on the type of the experiment there could be some false positive or false negative in overlap determination (we call the outcome graph of experiments *experiment graph*). So the goal is to find a subgraph or super-graph of the experiment graph which is interval. Notice if all clones are of the same size then what we are looking for is a unit interval graph.

There are two other facts that could help constructing the real DNA. Firstly, clones from one copy of the DNA can not overlap. So if we know which clone comes from which DNA then we can impose a coloring on the vertices in which all clones from a particular copy have the same color.

On the other hand, the maximum clique of overlap graph compared to the number of vertices (clones) is very small. The former is typically 5 to 15 while the latter is some thousands.

Based on this introduction Kaplan and Shamir define the following two problems in their paper [KS96] and show that both are solvable in polynomial time when k is constant.

Problem 5.5 (Clique Bounded Proper Interval Super-graph) *Given a graph G and a constant k, is there any super-graph G' of G which is a proper interval graph and $\omega(G) \leq k$.*

Problem 5.6 (Colored Proper Interval Super-graph) *Given a graph G and a proper k-coloring c of G, is there any super-graph G' of G which is a proper interval graph and respects c (is properly colored by c).*

An interesting fact discovered in [KS96] is the relationship between the first problem and the bandwidth problem. The bandwidth of a graph is defined as follows:

Definition 5.7 *Given a graph $G = (V, E)$ where $|V| = n$, a* linear layout *of G is a mapping $L : V \to \{1, 2, \ldots n\}$. The bandwidth of L is defined as $bw_L = \max_{(u,v)\in E} |L(v) - L(w)|$. The bandwidth of G is the minimum bandwidth bw_L over all linear layouts.*

It is easy to show that if the *clique bounded* problem has a solution then the bandwidth of G is less than k. It is shown that these two problems are in fact equivalent, i.e. the answer of the first problem is positive if and only if the bandwidth of G is less than k. Using this fact some intractability results are proven for this problem. It is also shown that the second problem is not fixed parameter tractable or $P = NP$.

References

[BCD98] Andreas Brandstädt, Victor D. Chepoi, and Feodor F. Dragan. The algorithmic use of hypertree structure and maximum neighbourhood orderings. *Discrete Applied Mathematics*, 82:43–77, 1998.

[BDK00] Andreas Brandstädt, Feodor F. Dragan, and Ekkehard Köhler. Linear time algorithms for hamiltonian problems on (claw,net)-free graphs. *SIAM Journal on Computing*, 30(5):1662–1677, 2000.

[Ber83] Alan A. Bertossi. Finding hamiltonian circuits in proper interval graphs. *Information Processing Letters*, 17(2):97–101, 1983.

[BFMY83] Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30(3):479–513, 1983.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

[BLS99] Andreas Brandstadt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. Society for Industrial and Applied Mathematics, 1999.

[BW99] Kenneth P. Bogart and Douglas B. West. A short proof that 'proper = unit'. *DMATH: Discrete Mathematics*, 201, 1999.

[CCC97] Chiuyuan Chen, Chin-Chen Chang, and Gerard J. Chang. Proper interval graphs and the guard problem. *Discrete Mathematics*, 170(1-3):223–230, 1997.

[Cha98] Maw-Shang Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM Journal on Computing*, 27(6):1671–1694, 1998.

[CKN+95] Derek G. Corneil, Hiryoung Kim, Sridhar Natarajan, Stephan Olariu, and Alan P. Sprague. Simple linear time recognition of unit interval graphs. *Information Processing Letters*, 55(2):99–104, 1995.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Sten. *Introduction to Algorithms*. The MIT Press, McGraw-Hill Book Company, 2 edition, 2001.

[COS98] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. The ultimate interval graph recognition algorithm? In *Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '98)*, pages 175–180, 1998.

[dFMdM95] C.M.H. de Figueiredo, J. Meidanis, and C. P. de Mello. A linear-time algorithm for proper interval graph recognition. *Information Processing Letters*, 56(3):179–184, 1995.

[DHH96] Xiaotie Deng, Pavol Hell, and Jing Huang. Linear-time representation algorithms for proper circular-arc graphs and proper interval graphs. *Information Processing Letters*, 25(2):390–403, 1996.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, CA, 1979.

[Gol80] Martin C. Golumbic. *Algorithmic graph theory and perfect graphs*. Academic Press, New York, 1980.

[Kei85] J. Mark Keil. Finding hamiltonian circuits in interval graphs. *Information Processing Letters*, 20(4):201–206, 1985.

[KM89] Norbert Korte and Rolf H. Möhring. An incremental linear time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989.

[KS96] Haim Kaplan and Ron Shamir. Pathwidth, bandwidth, and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25(3):540–561, 1996.

[LO92] Peter J. Looges and Stephan Olariu. Optimal greedy recognition and coloring for indifference graphs. In *Computer Science and Operation Research: New Results at their Interfaces*, pages 127–137, 1992.

[Mul96] Haiko Muller. Hamiltonian circuits in chordal bipartite graphs. *Discrete Mathematics*, 156(1-3):291–298, 1996.

[PD03] B. S. Panda and Sajal K. Das. A linear time recognition algorithm for proper interval graphs. *Information Processing Letters*, 87(3):153–161, 2003.

[PY79] Christos H. Papadimitriou and Mihalis Yannakakis. Scheduling interval-ordered tasks. *SIAM Journal on Computing*, 8(3):405–409, 1979.

[Rob69] Fred S. Roberts. Indifference graphs. *Proof Tachniques in Graph Theory*, pages 139–146, 1969.

[Rob71] Fred S. Roberts. On the compatibility between a graph and a simple order. *Jour. of Combinatorial Theory*, 11:28–38, 1971.

[RR88] G. Ramalingam and C. Pandu Rangan. A unified approach to domination problems in interval graphs. *Information Processing Letters*, 27:271–274, 1988.

[Spi03] Jerry P. Spinrad. *Efficient Graph Representations*. American Mathematical Society (AMS), Providence, R.I., 2003.

[SS58] Dana S. Scott and Patrick Suppes. Foundational aspects of theories of measurement. *Journal of Symbolic Logic*, 23(2):113–128, 1958.