AMO - Advanced Modeling and Optimization, Volume 11, Number 3, 2009

Interval Tree and its Applications[1]

Anita Pal* and Madhumangal Pal†

*Department of Mathematics,
National Institute of Technology Durgapur,
Durgapur-713209, India.
e-mail: anita.buie@gmail.com

†Department of Applied Mathematics with Oceanology and Computer Programming,
Vidyasagar University, Midnapore – 721 102, India.
e-mail: mmpalvu@gmail.com

Abstract. Interval graph is a very important subclass of intersection graphs and perfect graphs. It has many applications in different real life situations. The problems on interval graph are solved by using different data structures among them interval tree is very useful. During last decade this data structure is used to solve many problems on interval graphs due to its nice properties. Some of its important properties are presented here. Here we introduced some problems on interval graphs which are solved by using the data structure interval tree. A brief review of interval graph is also given here.

Keywords: Design and analysis of algorithms, interval graph, interval tree, diameter, all-pairs shortest paths, tree 3-spanner, k-covering problem.

AMS Subject Classifications: 68Q22, 68Q25, 68R10.

1 Introduction

An undirected graph $G = (V, E)$ is said to be an *interval graph* if the vertex set V can be put into one-to-one correspondence with a set I of intervals on the real line such that two vertices are adjacent in G iff their corresponding intervals have non-empty intersection. *i.e.*, there is a bijective mapping $f : V \to I$.

The set I is called an interval representation of G and G is referred to as the interval graph of I [9].

Interval graphs arise in the process of modelling many real life situations, specially involving time dependencies or other restrictions that are linear in nature. This graph and various subclass thereof arise in diverse areas such as archeology, molecular biology, sociology, genetics, traffic planning, VLSI design, circuit routing, psychology, scheduling, transportation etc. Recently, interval graphs have found applications in protein sequencing [13], macro substitution [15],

[1]AMO-Advanced Modelling and Optimization, ISSN 1841-4311

circuit routine [19], file organization [4], job scheduling [4], routing of two points nets [11] and so on. An extensive discussion of interval graphs is available in [9]. In addition to these, interval graphs have been studied intensely from both the theoretical and algorithmic point of view.

In the following an application of interval graph to scheduling is presented.

Let $C = \{C_1, C_2, \ldots, C_n\}$ be a collection of courses offered by a University. Let T_i be the time interval during which course C_i is to take place. We would like to assign courses to class rooms so that no two courses meet in the same room at the same time [9].

This problem can be solved by properly colouring the vertices of the graph $G = (C, E)$ where

$$(C_i, C_j) \in E \Leftrightarrow T_i \cap T_j \neq \phi.$$

Each colour corresponds to a different classroom. The graph G is obviously an interval graph, since it is represented by time intervals. This problem can be solved using only $O(n)$ time [27].

2 Interval Graphs

Interval graphs satisfy a lot of interesting properties. The first one is the *hereditary property.*

Lemma 2.1 *An induced subgraph of an interval graph is an interval graph [9].*

The next property of interval graphs is also a hereditary property, called *triangulated graph property*, which is stated below.

Every simple cycle of length strictly greater than 3 possesses a chord.

The graphs which satisfy this property are called triangulated graph. So we have the following lemma.

Lemma 2.2 *An interval graph satisfies the triangulated graph property [10].*

But, the converse of this lemma is not true as the graphs of Figure 1 (b), (c), (d) and (e) are all triangulated but they are not interval graphs.

Another important property on graphs is transitive orientation property stated below:

Each edge can be assigned a one-way direction in such a way that the resulting oriented graph (V, E) satisfies the following condition:

$$(u, v) \in E \text{ and } (v, w) \in E \Rightarrow (u, w) \in E,\ u, v, w \in V.$$

The following result is due to Ghouila-Houri [6].

Lemma 2.3 *The complement of an interval graph satisfies the transitive orientation property.*

The following theorem posed by Gilmore and Hoffman [7] establishes the position of the interval graphs in the world of perfect graphs.

Theorem 2.1 *Let G be an undirected graph. The following statements are equivalent.*
(i) G is an interval graph
(ii) G contains no chordless cycle of length 4 and its complement $\overline{G}$ is a comparability graph.
(iii) The maximal cliques of G can be linearly ordered such that, for every vertex u of G, the maximal cliques containing u occur consecutively.

Statement (iii) of this theorem has an interesting matrix formulation. A matrix whose entries are zeros and ones, is said to have the *consecutive 1's property for columns* if its rows can be permuted in such way that the 1's in each column occur consecutively.

The maximal cliques versus vertices incidence matrix of a graph G is called *clique matrix.*

The following theorem given by Fulkerson and Gross [5], is useful to recognize an interval graph.

Theorem 2.2 *An undirected graph G is an interval graph if and only if its clique matrix M has the consecutive 1's property for columns.*

Another important characterization of interval graph proposed by Lekkerkerker and Boland [14], is given below.

Theorem 2.3 *An undirected graph G is an interval graph if and only if the following two conditions are satisfied:*
(i) G is a triangulated graph, and
(ii) any three vertices of G can be ordered in such a way that every path from the first vertex to the third vertex passes through a neighbour of the second vertex.

The necessary and sufficient condition that a graph is an interval graph is stated below:

Theorem 2.4 *[14] A graph is an interval graph if and only if it contains none of the graphs shown in Figure 1 as an induced subgraph.*

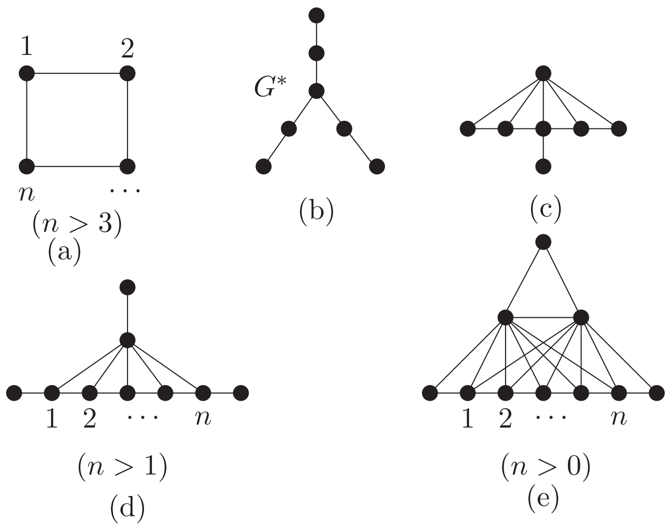

Figure 1: Forbidden structures for interval graphs.

Corollary 2.1 *A tree is an interval graph if and only if it does not contain G^* (see Figure 1) as an induced subgraph.*

Proof. Among the five forbidden structures of Theorem 2.4, only one of them can be an induced subgraph of a tree G^*. □

Corollary 2.2 *A tree is a circular-arc graph if and only if it is an interval graph.*

Proof. Let G be a circular-arc graph which is a tree and suppose that it is not an interval graph. Therefore, by Theorem 2.4, G contain some of the graphs shown in Figure 1. Since G is a tree, this induced subgraph can only be G^*. But this graph is not a circular-arc graph, which is a contradiction. The converse is true because interval graphs are a subclass of circular-arc graphs. □

3 Interval Tree

Let $G = (V, E)$, where $|V| = n, |E| = m$ be a simple (i.e., there is no self loop or parallel edges) connected graph, where vertices are numbered as 1, 2, $\ldots, n$. Let $I = \{I_1, I_2, \ldots, I_n\}$ be the interval representation of an interval graph G, where a_r is the left end point and the b_r is the right endpoint of the interval I_r, i.e., $I_r = [a_r, b_r]$ for all $r = 1, 2, \ldots, n$. Without any loss of generality we assume the following:

1. The intervals in I are indexed by increasing right endpoints i.e., $b_1 < b_2 < \cdots < b_n$.
2. The intervals are closed, i.e., contains both of its endpoints and that no two intervals share a common endpoint.
3. Vertices of the interval graph and the intervals on the real line are one and the same thing.
4. The interval graph G is connected and the list of sorted endpoints is given.

Considering the location of $2n$ endpoints of the n intervals on the real line in increasing order an array $e = \{e_1, e_2, \ldots, e_{2n}\}$ is formed. For each element e_i of e, three fields, $e_i.val$, $e_i.int$ and $e_i.type$ are defined as follows.

$$\begin{aligned} e_i.val &= \text{the value on the real line of the } i\text{th endpoint } e_i, \\ e_i.int &= k, \text{ if } e_i \text{ is the endpoint of the interval } I_k, \end{aligned}$$

$$e_i.type = \begin{cases} a, & \text{if the endpoint } e_i \text{ is left endpoint} \\ b, & \text{if the endpoint } e_i \text{ is right endpoint.} \end{cases}$$

It is shown, in [33], that the set intervals of every interval graph can be ordered in a non-decreasing order of their right endpoints and this ordering is referred as IG ordering. In this section, the vertices are labelled as IG ordering. The IG ordering is obviously unique when a representation by a set of intervals is provided and fixed.

The following lemma is a powerful result on interval graph. The most of the algorithms developed on interval graphs are based on this result.

Lemma 3.1 *If the vertices $u, v, w \in V$ are such that $u < v < w$ in the IG ordering and $(u, w) \in E$, then $(v, w) \in E$.*

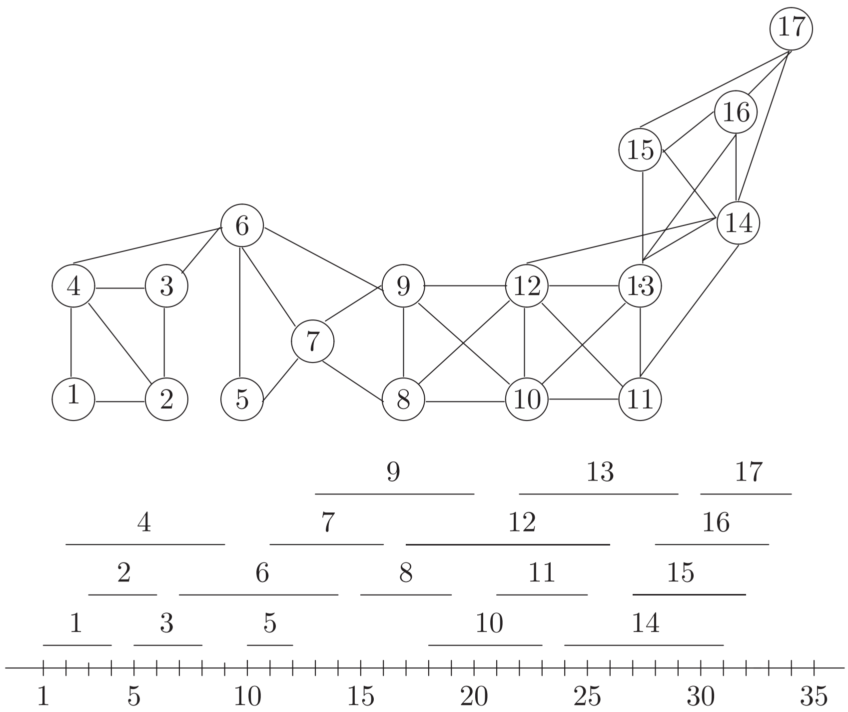

Figure 2: An interval graph and its interval representation.

v	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$H(v)$	4	4	6	6	7	9	9	12	12	13	14	14	16	17	17	17	17

Table 1: The array H of the graph of Figure 2.

An interval graph and its interval representation are illustrated in Figure 2.

Now, we introduce a very important data structure of interval graph called *interval tree (IT)* which is used to solve several problems on interval graphs. In the next section the definition of IT and its properties are presented.

3.1 Definition of interval tree

For each vertex $v \in V$ let $H(v)$ and $L(v)$ represent respectively the highest and the lowest numbered adjacent vertices of v. It is assumed that $(v, v) \in V$ is always true. So, if no adjacent vertex of v exist with higher (or lower) IG order than v then $H(v)$ (or $L(v)$) is assumed to be v. In other words,

$$\begin{aligned} H(v) &= \max\{u : (u, v) \in E, u \geq v\}, \text{ and} \\ L(v) &= \min\{u : (u, v) \in E, u \leq v\}. \end{aligned}$$

The array $H(1 : n)$ of the graph of Figure 2 is shown in Table 1.

The array H is monotonic non-decreasing, which is proved in the following lemma.

Lemma 3.2 *If $u, v \in V$ and $u < v$ then $H(u) \leq H(v)$.*

Proof. If possible let $H(u) > H(v)$ for $u < v$. From definition of $H(v)$ it follows that $v < H(v)$. Thus we have $u < v \leq H(v) < H(u)$ which implies $u < v < H(u)$. This implies $(v, H(u)) \in E$ (by Lemma 3.1). Therefore, $H(v) = H(u)$, which contradicts $H(u) > H(v)$. Hence $H(u) \leq H(v)$. □

For a given interval graph G let a spanning subgraph $G' = (V, E')$ be defined as

$$E' = \{(u, v) : u \in V \text{ and } v = H(u),\ u \neq n\}.$$

The following lemma establishes that this subgraph G' is a tree and it is unique for a given interval representation.

Lemma 3.3 *The subgraph G' for a connected interval graph G is a tree.*

Proof. By the definition of G', G' has n vertices and $n-1$ edges. Also, $H(v) \geq v$, by the definition of H. For the sake of contradiction, we assume that $H(v) = v$, for some $v \neq n$. Let $u \in V$ be a vertex such that $u < v$. Since $H(v) = v$ by hypothesis and $H(u) \leq H(v)$, by Lemma 3.2, it follows that $H(u) \leq v$. In other words, the vertex u is not adjacent to a vertex which is greater than v. Also, since $H(v) = v$ the vertex v is not adjacent to a vertex which is greater than v. Thus the subgraphs induced by the vertices $\{1, 2, \ldots, v\}$ and $\{v+1, \ldots, n\}$ are disconnected in G. Hence, G is disconnected. Therefore, the assumption $H(v) = v$ is not true, i.e., $H(v) > v$ for all $v \in V$, $(v \neq n)$. Thus, G' has no self loop and consequently G' is a tree. □

Since the subgraph G' is built from the vertex set V and the edge set E', where $E' \subseteq E$, G' is a spanning tree of G. In what follows the subgraph G' is referred to as *interval tree* and it is denoted by $T_I(G)$. The existence and uniqueness of interval tree are proved in the following lemma.

Lemma 3.4 *The interval tree $T_I(G)$ of a connected interval graph G exists and is unique for a given interval representation.*

Proof. The existence of $T_I(G)$ follows from the definition of interval tree and proof of Lemma 3.3.

Since the given IG order of a vertex $v \in V$ is unique, $H(v)$ is also unique. Thus the tree $T_I(G)$ is unique for any interval graph G. □

The interval tree $T_I(G)$ of the interval graph of Figure 2 is shown in Figure 3.

The *level* of a tree is defined recursively as follows:
We take the root of the tree as n and the level of the root as 0. The level of each child of the root is 1. If the level of a vertex is l then the level of each of its child is $l+1$. The level of a vertex u in the interval tree is denoted by $level_I(u)$. Let N_i be the set of vertices which are at a distance i from the vertex n, i.e., N_i is the set of vertices at level i. Thus $N_i = \{u : \delta_G(u, n) = i\}$ and N_0 is the singleton set $\{n\}$. It may be noted that if $u \in N_i$ then $level_I(u) = i$. Let k be the maximum length of a shortest path from the vertex n to any other vertex in G. It is easy to see that N_k is non-empty while N_{k+1} is empty.

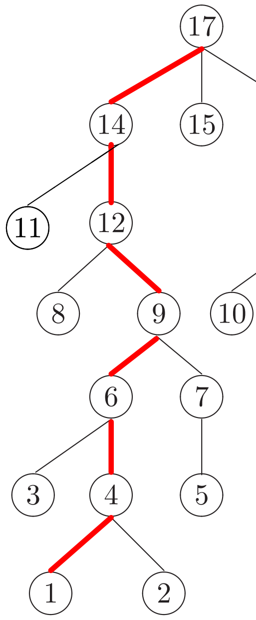

Figure 3: The interval tree of the graph of Figure 2.

3.2 Properties of the interval tree

Let $\min(N_i)$ and $\max(N_i)$ represent the minimum and maximum numbered vertices of the set N_i. That is, $\min(N_i) = \min\{u : u \in N_i\}$ and $\max(N_i) = \max\{u : u \in N_i\}$. The vertices of N_i satisfy the following result.

Lemma 3.5 *The vertices of N_i are consecutive integers and* $\max(N_{i+1}) = \min(N_i) - 1$ *for all i.*

Proof. From the definition of interval tree it follows that the vertices in N_1 are $L(n), L(n) + 1, \ldots, n-1$. Therefore, the lemma is true for $i = 1$.

Let the lemma be true for $i = k$. Therefore the vertices in N_k are consecutive integers and $\max(N_{k+1}) = \min(N_k) - 1$. By definition of interval tree, it follows that if $u \in N_{k+1}$ then $H(u) \in N_k$. If v is equal to $\min(N_{k+1})$ then by Lemma 3.1, $v, v+1, \ldots, \max(N_{k+1})$ and also $\max(N_{k+1}) + 1, \max(N_{k+1}) + 2, \ldots, H(v) - 1$ are all adjacent to $H(v)$. Since, $\max(N_{k+1})$ is the maximum vertex in N_{k+1} so, $v, v+1, \ldots, \max(N_{k+1}) \in N_{k+1}$. Thus the vertices in N_{k+1} are consecutive integers. Since v is the minimum vertex in N_{k+1} therefore $v - 1 \notin N_{k+1}$, but $v - 1 \in N_{k+2}$. That is, the lemma is true for $i = k+1$ if the lemma is true for $i = k$. Hence the lemma follows by induction. $\square$

From the above lemma it follows that if u is a vertex of interval tree at level i with $L(u) = \min(N_i)$ then the vertices at level $(i+1)$ of the interval tree are $L(u), L(u)+1, \ldots, v-1$, where v is the minimum vertex at level i. From this observation we have the following lemma.

Lemma 3.6 *If $level_I(u) < level_I(v)$ then $u > v$.*

The height of a tree, T, is defined as

$$h(T) = \max\{level_I(v) : v \in V\}.$$

The maximum value of $level_I(v)$ is $h(T_I(G))$ and the minimum value of $level_I(v)$ is 0. This minimum occurs when $v = n$. But, if $v = n$ then $d(u, n) = level_I(u) \leq h(T_I(G))$. Thus, $\delta_G(u, v)$ is maximum when $level_I(v) = 1$ and $level_I(u) = h(T_I(G))$ and the maximum distance is $h(T_I(G)) + 1$.

The following lemma is obvious.

Lemma 3.7 *$level_I(1) = h(T_I(G))$ and $level_I(n) = 0$.*

From Lemma 3.7, it is easy to note that the path from the vertex 1 to the vertex n in the interval tree $T_I(G)$ is the longest path among the paths ending at n. This path is referred as *main path*. The main path of the graph of Figure 2 is shown by thick (red) lines.

We denote the shortest distance between the vertices u and v in G by $\delta_G(u, v)$. If two vertices have same level then the distance in G between them is either 1 or 2. This result is given in the following lemma.

Lemma 3.8 *[29] For $u, v \in V$ if $level_I(u) = level_I(v)$ then*

$$\delta_G(u, v) = \begin{cases} 1, & \text{if } (u, v) \in E(G) \\ 2, & \text{otherwise.} \end{cases}$$

But, if $level_I(u) = level_I(v), u, v \in V$ then $\delta_{T_I(G)}(u, v)$ is not necessarily 1 or 2, it may even be more than 3 units. For example, for the interval graph of Figure 3, $level_I(8) = level_I(10) = 3$ and $\delta_{T_I(G)}(8, 10) = 6$.

If level of the vertex v is j then it should be adjacent in G only to the vertices at level $j - 1$, j and $j + 1$. This observation is proved in the following lemma.

Lemma 3.9 *[29] If $u, v \in V$ and $|level_I(v) - level_I(u)| > 1$ then $(u, v) \notin E(G)$.*

We denote u_l as a vertex of level l and u_l^* a vertex of the same level on the main path. Let X_l be the set of vertices at level l of IT which are greater than u_l^*, i.e.,

$$X_l = \{v : v > u_l^* \text{ and } v \in N_l\}.$$

Similarly, Y_l be the set of vertices at level l of IT which are less than u_l^*, i.e.,

$$Y_l = \{v : v < u_l^* \text{ and } v \in N_l\}.$$

It may be noted that $X_l \cap Y_l = \phi$ and $N_l = X_l \cup Y_l \cup \{u_l^*\}$. Since the vertices of N_l are consecutive integers, the vertices of X_l and Y_l are also consecutive integers.

Lemma 3.10 *If v be any member of $\bigcup_{i=0}^{2} X_{l+i}$ then $\delta_G(v, u_l^*) \leq 2$.*

Proof. From definition of X_l it follows that $u_l^* < v$ for all $v \in X_l$, and for all l.

Let v_1 be any vertex of X_{l+2}. Then $u_{l+2}^* < v_1 < u_{l+1}^*$. Since $(u_{l+2}^*, u_{l+1}^*) \in E$, by Lemma 3.1 $(v_1, u_{l+1}^*) \in E$. Therefore, $\delta_G(v_1, u_l^*) = 2$ (as $u_l^* \to u_{l+1}^* \to v_1$). If v_1'' be any vertex of X_{l+1} then $u_{l+1}^* < v_1'' < u_l^*$. Since $(u_{l+1}^*, u_l^*) \in E$, $(v_1'', u_l^*) \in E$ (by Lemma 3.1) and hence $\delta_G(u_l^*, v_1'') = 1$.

Again, if $v \in X_l$ then $\delta_G(v, u_l^*) \leq 2$ (by lemma 3.8). Thus $\delta_G(u_l^*, v) \leq 2$ for all $v \in \bigcup_{i=0}^{2} X_{l+i}$. $\square$

Lemma 3.11 *If v be any member of $\bigcup_{i=0}^{2} Y_{l+i}$ then either $\delta_G(v, u_l^*) \leq 2$ or $\delta_G(v, u_{l+3}^*) \leq 2$.*

Proof. Let t_1 and t_1' be any two vertices of Y_{l+2} and Y_{l+1} respectively. Let u_l' be any vertex at level l. There are two cases. Case I. $u_l' = u_l^*$ and Case 2. $u_l' \neq u_l^*$.

Case I. $u_l' = u_l^*$. In this case $\delta_G(u_l^*, t_1') = 1$ and $\delta_G(u_l^*, t_1) = 2$. Also, $\delta_G(u_l^*, v) \leq 2$ (by Lemma 3.8) for all $v \in Y_l$. Therefore, $\delta_G(u_l^*, v) \leq 2$ for all $v \in \bigcup_{i=0}^{2} Y_{l+i}$.

Case II. $u_l' \neq u_l^*$. Without loss of generality we assume that parent$(t_1) = t_1'$ and parent$(t_1') = u_l'$. Since parent$(t_1) = t_1'$, i.e., $H(t_1) = t_1' < u_{l+1}^*, (t_1, u_{l+1}^*) \notin E$. Similarly, $H(t_1') = u_l' < u_l^*$ implies $(t_1', u_l^*) \notin E$. Thus, $\delta_G(u_l^*, t_1') = 2$ and $\delta_G(u_l^*, t_1) = 3$ (as $u_l^* \to u_l' \to t_1'$ or $u_l^* \to u_{l+1}^* \to t_1' \to t_1$).

Now, $u_{l+3}^* < t_1 < u_{l+2}^* < t_1', (u_{l+3}^*, u_{l+2}^*) \in E$ and $(t_1, t_1') \in E$ implies $(t_1, u_{l+2}^*) \in E$ and $(u_{l+2}^*, t_1') \in E$. Thus, $\delta_G(u_{l+3}^*, t_1) \leq 2$ and $\delta_G(u_{l+3}^*, t_1') = 2$. Hence, either $\delta_G(u_l^*, v) \leq 2$ or $\delta_G(u_{l+3}^*, v) \leq 2$ for all $v \in \bigcup_{i=0}^{2} Y_{l+i}$. $\square$

4 Applications of Interval Tree

4.1 Construction of tree 3-spanner

A *t-spanner* of a graph G is a spanning subgraph $H(G)$ in which the distance between every pair of vertices is at most t times their distance in G, i.e., $\delta_H(u, v) \leq t\ \delta_G(u, v)$, for all $u, v \in V$. The parameter t is called the *stretch factor*.

The *minimum t-spanner* problem is to find a t-spanner H with the fewest possible edges for fixed t. The spanning subgraph H is called a minimum t-spanner of G and it is denoted by $H_t(G)$. A *spanning tree* of a connected graph G is an acyclic (cycle free) connected spanning subgraph of G. A *tree spanner* of a graph is a spanning tree that approximates the distance between the vertices in the original graph. In particular, a spanning tree T is said to be a *tree t-spanner* of a graph G if the distance between any two vertices in T is at most t times their distance in G, i.e., $\delta_T(u, v) \leq t\ \delta_G(u, v)$ for all $u, v \in V$. It is obvious that if G is connected then $|E(H_t(G))| \geq n - 1$, equality holds iff G admits a tree t-spanner.

The t-spanner problems

The minimum t-spanner problem is of two types - decision version and optimization version.

The decision version of the problem is stated below:

Input: A graph $G = (V, E)$ and $k \geq 0$ are given.
Question: Whether G has a t-spanner with k or fewer edges,
i.e., $|E(H_t(G))| \leq k$.

The optimization version of the problem is defined in the following:
Input: A graph $G = (V, E)$.
Problem: Find a t-spanner with the fewest possible edges for a fixed t.

In this section, the optimization version of the problem is considered.

It can be shown by examples that the interval tree may or may not be a tree 3-spanner of the corresponding interval graph.

Lemma 4.1 *The interval tree may or may not be a tree 3-spanner.*

But, the tree 3-spanner can be constructed by modification of the interval tree. The modification process and details of construction of tree 3-spanner is discussed in the next section.

The tree 3-spanner, $T_{3S}(G)$, of G can be constructed from the interval tree by rearranging the parent vertex of some vertices.

The method is described below:
Let w_l^* and w_{l+1}^* be two vertices on the main path at levels l and $l+1$ respectively. Then we assign parent of each vertex u, satisfying $w_{l+1}^* < u < w_l^*$ as u_l^* i.e., $parent_I(u) = w_l^*$, where $parent_I(u)$ is the parent of the vertex u in the interval tree $T_I(G)$. This process is repeated for all vertices of all levels l, $l = 1, 2, \ldots, h(T_I(G)) - 1$. In other words, if $N_l = \{x_1, x_2, \ldots, x_{i-1}, w_l^*, x_{i+1}, \ldots, x_p\}$ and $N_{l+1} = \{y_1, y_2, \ldots, y_{j-1}, w_{l+1}^*, y_{j+1}, \ldots, y_q\}$, where $p = |N_l|$ and $q = |N_{l+1}|$ then parent of $y_{j+1}, \ldots, y_q$ and $x_1, x_2, \ldots, x_{i-1}$ are w_l^*.

If $N_l = \{x_1, x_2, \ldots, x_{i-1}, w_l^*, x_{i+1}, \ldots, x_p\}$, where $p = |N_l|$, then let $N_l' = \{x_1, x_2, \ldots, x_{i-1}\}$ and $N_l'' = \{x_{i+1}, \ldots, x_p\}$. That is, N_l' (respectively, N_l'') is the subset of N_l whose vertices are less than w_l^* (respectively, greater than w_l^*). We denote the set of vertices at level l of the tree $T_{3S}(G)$ by N_l^*. Then from the construction of $T_{3S}(G)$ we have $N_{l+1}^* = \{w_{l+1}^*\} \cup N_{l+1}'' \cup N_l', l = 0, 1, \ldots, h(T_I(G)) - 1$.

The tree $T_{3S}(G)$, for the interval graph of Figure 2 is shown in Figure 4.

Lemma 4.2 *If w_l^* and w_{l+1}^* be two vertices on the main path at level l and $l+1$ respectively and u be any vertex such that $w_{l+1}^* < u < w_l^*$ then (u, w_l^*) is an edge in G as well as in $T_{3S}(G)$.*

From the construction of tree 3-spanner and also from the Figure 4, it is easy to observe that a vertex not in the main path is adjacent to exactly one vertex in the main path.

Lemma 4.3 *Let $u, v \in V$ and $level_S(u) = level_S(v)$ then $\delta_{T_{3S}(G)}(u, v) = 2$.*

Proof. Let $level_S(u) = level_S(v) = l$. Then by Lemma 4.2, (u, w_{l-1}^*) and (v, w_{l-1}^*) are two distinct edges in $T_{3S}(G)$. Hence there exists only one path $u \to w_{l-1}^* \to v$ of length 2, and consequently $\delta_{T_{3S}(G)}(u, v) = 2$. □

Let u be a vertex of $T_{3S}(G)$ at level l then there may be an edge $(u, v) \in E(G)$ if v is at level $l-1, l$ or $l+1$. This fact is justified in the following.

Lemma 4.4 *If $|level_S(u) - level_S(v)| > 1$ for the vertices u and v of $T_{3S}(G)$ then $(u, v) \notin E(G)$.*

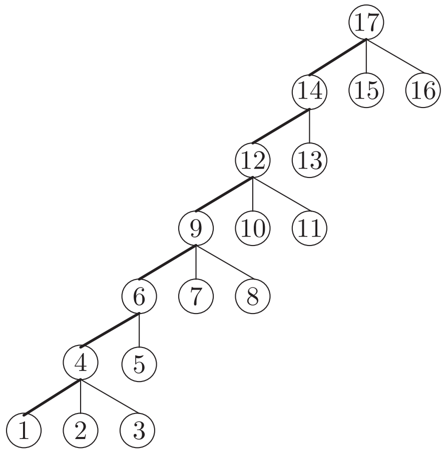

Figure 4: The tree 3-spanner, $T_{3S}(G)$, for the graph of Figure 2.

Proof. Let $N_l = \{x_1, x_2, \ldots, x_{i-1}, w_l^*, x_{i+1}, \ldots, x_p\}$ where $p = |N_l|$.
Then $N_l' = \{x_1, x_2, \ldots, x_{i-1}\}$ and $N_l'' = \{x_{i+1}, \ldots, x_p\}$.

By the definition of $T_I(G)$, $H(x_j) \leq w_{l-1}^*, x_j \in N_l'$ and $(x_j, v) \notin E(G)$ for $x_j \in N_l'$ and $v \in N_{l-1}''$. But, there may be an edge in G between the vertices of N_l' and N_{l-1}' (by Lemma 3.9). The parents of the vertices of N_l' are changed to w_l^* in $T_{3S}(G)$. Similarly, the parents of the vertices of N_{l-1}' become w_{l-1}^* in $T_{3S}(G)$.

Hence $level_I(u) = level_S(u) + 1$ for all $u \in N_l'$ and for all l. But, the level of the vertices in $T_{3S}(G)$ of N_l'' remain same as in $T_I(G)$, i.e., $level_I(u) = level_S(u)$ for all $u \in N_l''$ and for all l.

Thus, by Lemma 3.9, if $|level_S(u) - level_S(v)| > 1$ then $(u, v) \notin E(G)$. $\square$

The following lemma gives distance between two vertices in $T_{3S}(G)$ at two consecutive levels.

Lemma 4.5 *If u and v be two vertices such that $|level_S(u) - level_S(v)| \leq 1$ then $\delta_{T_{3S}(G)}(u, v) =$ 2 or 3.*

Proof. Case I. $|level_S(u) - level_S(v)| = 0$.
Without loss of generality we assume that $level_S(u) = l = level_S(v)$. Then there is a path $u \to w_{l-1}^* \to v$ between u and v of length 2. Thus $\delta_{T_{3S}(G)}(u, v) = 2$.

Case II. $|level_S(u) - level_S(v)| = 1$.
Let $level_S(v) = l$ and $level_S(u) = l + 1$. Then the path between u and v in $T_{3S}(G)$ is $u \to w_l^* \to w_{l-1}^* \to v$ which is of length 3. Thus, $\delta_{T_{3S}(G)}(u, v) = 3$. $\square$

The following lemma is the combination of the lemmas 4.4 and 4.5.

Lemma 4.6 *The tree $T_{3S}(G)$ is a tree 3-spanner of the interval graph G.*

The interval tree exits and is unique for a given interval representation of an interval graph (Lemma 3.4). The tree 3-spanner $T_{3S}(G)$ is obtained by rearranging the parents of the vertices of interval tree $T_I(G)$. So one may conclude the following result.

Theorem 4.1 *Every connected interval graph has a tree 3-spanner and it is unique for a given interval representation.*

Theorem 4.2 *A tree 3-spanner of an interval graph with n vertices can be constructed, in sequential, in $O(n)$ time, if the sorted intervals are given.*

Theorem 4.3 *The tree 3-spanner of an interval graph can be constructed in parallel using $O(\log n)$ time and $O(n/\log n)$ processors on an EREW PRAM, where n represents the number of vertices of the interval graph.*

4.2 Computation of diameter

Let $G = (V, E)$ be a graph and $\delta_G(u, v)$ be the shortest distance between the vertices u and v. The eccentricity of the vertex u is denoted by $ecen(u)$ and is defined as

$$ecen(u) = \min_{v \in V}\{\delta_G(u, v)\}.$$

The radius $(\rho(G))$ and diameter $(diam(G))$ of a graph G are defined as

$$\begin{aligned} \rho(G) &= \min_{u \in V}\{ecen(u)\} \\ diam(G) &= \max_{u \in V}\{ecen(u)\}. \end{aligned}$$

The diameter of an interval graph G and the height of the corresponding interval tree $T_I(G)$ of G are related by the following relation.

Lemma 4.7 *[28] Let $v_1^* \in N_1$ be the vertex on the main path. If all $v_1 \in N_1$ are adjacent to v_1^* in G then $diam(G) = h(T_I(G))$, otherwise $diam(G) = h(T_I(G)) + 1$.*

The level of the vertices on the main path of the both trees $T_I(G)$ and $T_{3S}(G)$ remain unchanged, the heights of $T_{3S}(G)$ and $T_I(G)$ are also same. Thus the following lemma directly follows from Lemma 4.7.

Lemma 4.8 *The height of the tree 3-spanner is either $diam(G)$ or $diam(G)-1$, i.e., $diam(G) = h(T_{3S}(G))$ or $h(T_{3S}(G)) + 1$.*

4.3 All-pairs shortest distances

According to the lemma 3.8, the shortest distance between the vertices u and v, when $level_I(u) = level_I(v)$, is either 1 or 2. But, if their levels are different then the distance between two vertices may be 1 or 2 or more. In this case the distance between any two vertices can also be computed easily with the help of interval tree. The technique is described below.

By Lemma 3.9, to compute the distance between u and v, $u < v$, we check the adjacency of the vertex v with the vertices at levels $level(v) + 1$, $level(v)$ and $level(v) - 1$. Hence the distance $\delta_G(u, v)$ between any two vertices $u, v \in V$ can be computed using the following lemma.

Lemma 4.9 *[29] Given $u, v \in V$, let z_1 be the vertex at level $level(v)+1$ on the path marked* $\min(u)$ *and $z_2 = H(z_1)$. If $level(u) > level(v)$, then*

$$\delta_G(u,v) = \begin{cases} level(u) - level(v), & \text{if } (z_1, v) \in E \\ level(u) - level(v) + 1, & \text{if } (z_1, v) \notin E, \text{ and } (z_2, v) \in E \\ level(u) - level(v) + 2, & \text{otherwise.} \end{cases}$$

Using the above lemma the all-pairs shortest distances can be computed for an interval graph. The time complexity is presented below.

Theorem 4.4 *[29] The all-pairs shortest distances of an interval graph with n vertices can be computed in $O(n^2/p + \log n)$ time using p processors on an EREW PRAM.*

4.4 The 2-neighbourhood-covering problem

The k-neighbourhood-covering (k-NC) problem is a variant of the domination problem. Domination is a natural model for location problems in operations research, networking, etc.

A vertex x k-dominates another vertex y if $\delta_G(x,y) \leq k$. A vertex z k-neighbourhood-covers an edge (x,y) if $\delta_G(x,z) \leq k$ and $\delta_G(y,z) \leq k$, i.e., the vertex z k-dominates both x and y. Conversely, if $\delta_G(x,z) \leq k$ and $\delta_G(y,z) \leq k$ then the edge (x,y) is said to be k-neighbourhood-covered by the vertex z. A set of vertices $C \subseteq V$ is a k-NC set if every edge in E is k-NC by some vertex in C. The k-NC number $\rho(G,k)$ of G is the minimum cardinality of all k-NC set.

This problem is NP-complete for general graph and also for chordal graph.

A linear time algorithm has been developed to solve 2-neighbourhood covering problem [16].

Let C be the minimum 2-neighbourhood-covering set of the given interval graph.

The main basic idea to compute C is described below. If there exists at least one vertex of N_1 which is not adjacent to u_l^*, we take u_1^* as a member of C otherwise we select the vertex u_2^* as a member of C. Let the first selected vertex (either u_1^* or u_2^*) be at level l. After selection of first member of C, we are to consider two vertices u_{l+3}^* or u_{1+4}^* (not both) will be a member of C. This selection is to be made according to some results, discussed in the following. After selection of second member of C, we set $l+3$ or l, if u_{l+3}^* is selected, otherwise we set $l+4$ to l. This selection is to be continued till new $l+3$ becomes greater than the height of the tree IT.

The condition to select u_1^* as a first member of C is obtained in the following lemma.

Lemma 4.10 *If there exists at least one vertex of N_1 which is not connected with u_1^* then u_1^* is a possible member of C.*

Proof. From the construction of IT it is clear that n is the parent of u_1^*. By hypothesis there exist at least one vertex at level 1, i.e., in N_1 which is not connected with u_1^*. Let v_1' be any such vertex. Then $\delta_G(u_1^*, v_1') = 2$ (as $u_1^* \to n \to v_1'$) and $\delta_G(u_1^*, n) = 1$, i.e., the vertex u_1^* is a 2-NC of the edge (v_1', n). If v_1'' be any vertex of N_1 connected with u_1^* then $\delta_G(v_1'', u_1^*) = 1$. As $\delta_G(n, u_1^*) = 1$, u_1^* is also a 2-NC of the edge (v_1'', n). Hence u_1^* is a 2-NC of (v_1, n) for each $v_1 \in N_1$. □

If u_1^* is connected with all vertices of N_1 then the vertex u_1^* may also be a member of C. But, in this case, the vertex u_2^* is to be selected as a member of C. This result is proved in the following lemma.

Lemma 4.11 *If u_1^* is connected with all vertices of N_1 then u_2^* is a possible member of C.*

Proof. Let u_1^* be connected with all vertices of N_1. Therefore, $\delta_G(u_1^*, v_1) = 1 = \delta_G(u_1^*, n)$ for all $v_1 \in N_1$. Hence the path from u_2^* to any $v_1, v_1 \in N_1$ is $u_2^* \to u_1^* \to v_1$ (since u_1^* is adjacent with all vertices of N_1), so $\delta_G(u_2^*, v_1) = 2$. But, u_2^* may be adjacent to some vertices of N_1. In this case, $\delta_G(u_2^*, v_1) = 1$. Hence $\delta_G(u_2^*, v_1) \leq 2$, for all $v_1 \in N_1$. Also, $\delta_G(u_2^*, n) = 2$. Thus, the edges $(n, v_1), v_1 \in N_1$ are 2-NC by u_2^*.

Again, if $v_2 \in N_2$ then $\delta_G(u_2^*, v_2) \leq 2$. Therefore, $\delta_G(u_2^*, v_1) \leq 2$ and $\delta_G(u_2^*, v_2) \leq 2$ for $v_1 \in N_1$ and $v_2 \in N_2$. Thus, each edge $(v_1, v_2) \in E$ is 2-NC by u_2^* may be selected as a member of C. □

The other members of the set C can be determined by using the lemmas 3.10 and 3.11.

Theorem 4.5 *The 2-neighbourhood covering set of an interval graph can be computed in $O(n)$ time.*

References

[1] Bera, D., Pal, M. and Pal, T.K., An efficient algorithm to generate all maximal cliques on trapezoid graphs, *International Journal of Computer Mathematics*, 79 (10)((2002) 1057-1065.

[2] Bera, D., Pal, M. and Pal, T.K., An efficient algorithm for finding all hinge vertices on trapezoid graphs, *Theory of Computing Systems*, 36 (1) (2003) 17-27.

[3] Bera, D., Pal, M. and Pal, T.K., An optimal PRAM algorithmfor a spanning tree on trapezoid graphs, *Journal of Applied Mathematics and Computing*, 12 (1-2) (2003) 21-29.

[4] Carlisle. M. C., Loyd. E. L., On the k-coloring of intervals, *LNCS, 497, ICCI'91*, (1991) 90-101.

[5] Fulkerson, D. R. and Gross, O. A., Incidence matrices and interval graphs, *Pacific J. Math.*, 15 (1965) 835-855.

[6] Ghouilo-Houri, A., Characterisation des graphs non orientes dont on peut orienter les arretes de maniere a obtenir le graphe d'une relation d'ordre. *C. R. Acad. Sci.*, Paris, 254 (1962) 1370-1371.

[7] Gilmore, P. C. and Hoffman, A. J., A characterization of comparability graphs and of interval graphs, *Canad. J. Math.*, 16 (1964) 539-548.

[8] Ghosh, P.K. and Pal, M. *An optimal algorithm to solve 2-neighbourhood covering problem on trapezoid graphs*, *Advanced Modeling and Optimization*, 9(1) (2007) 15-36.

[9] Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, (Academic Press, New York, 1980).

[10] Hajös, G., Uber eine Art von Graphen, Fifst posed the problem of characterizing interval graphs. *Intern. Math. Nachr.*, 11 (1957) problem 65.

[11] Hashimoto, A. and Stevens, J., Wire routing by optimizing channel assignment within large apertures, in Proc., *8th IEEE Design Automation Workshop*, (1971) 155-169.

[12] Hota, M., Pal, M. and Pal, T.K., An efficient algorithm for finding a maximum weight k-independent set on trapezoid graphs, *Computational Optimization & Applications*, 18 (2001) 49-62.

[13] Jungck, J. R., Dick, O. and Dick, A. G., Computer assisted sequencing, interval graphs and molecular evolution, *Biosystem*, 15 (1982) 259-273.

[14] Lekkerkerker, C. G. and Boland, J. C., Representation of a finite graph by a set of intervals on the real line, *Fund. Math.* 51 (1962) 45-64.

[15] Fabri, J., *Automatic Storage Optimization*, (UMI Press Ann Arbor, MI, 1982).

[16] Mondal, S., Pal, M. and Pal, T.K., An optimal algorithm to solve 2-neighbourhood covering problem on interval graphs, *International Journal of Computer Mathematics*, 79 (2) (2002) 189-204.

[17] Mondal, S., Pal, M. and Pal, T.K., An optimal algorithm to solve the all-pairs shortest paths problem on permutation graph, *Journal of Mathematical Modelling and Applications*, 2(1) (2003) 57-65.

[18] Nayeem, Sk. Md. Abu and Pal, M., Shortest path problem on a network with imprecise edge weight, *Fuzzy Optimization and Decision Making*, 4 (2005) 293-312.

[19] Ohtsuki. T., Mori. H., Khu. E. S., Kashiwabara. T., Fujisawa. T., One dimensional logic gate assignment and interval graph, *IEEE Trans. Circuits and Systems*, 26 (1979) 675-684.

[20] Pal, M. and Bhattacharjee, G. P., An optimal parallel algorithm for computing all maximal cliques of an interval graph and its applications, *J. of Institution of Engineers (India)*, 76 (1995) 29-33.

[21] Pal, M., *Some sequential and parallel algorithms on interval graphs*, Ph.D Thesis, Indian Institute of Technology, Kharagpur, India, 1995.

[22] Pal, M. and Bhattacharjee, G. P., Optimal sequential and parallel algorithms for computing the diameter and the centre of an interval graph, *Intern. J. Computer Maths.*, 59 (1995) 1-13.

[23] Pal, M. and Bhattacharjee, G. P., An improved algorithm for finding the maximum weight $k-$independent set on an interval graph, in Proc.: *5th National Seminar on Theoretical Computer Science,* Bombay, India, Aug. 1-4, (1995) 95-104.

[24] Pal, M. and Bhattacharjee, G. P., The parallel algorithms for determining edge-packing and efficient edge dominating sets in interval graphs, *Parallel Algorithms and Applications,* 7 (1995) 193-207.

[25] Pal, M. and Bhattacharjee, G. P., A sequential algorithm for finding a maximum weight k-independent set on interval graphs, *Intern. J. Computer Maths.*, 60 (1996) 439-449.

[26] Pal, M., An efficient parallel algorithm for computing a maximum-weight independent set of a permutation graph, in Proc.: *6th National Seminar on Theoretical Computer Science*, Banasthali Vidyapith, Rajasthan, India, Aug. 5-8 (1996) 276-285.

[27] Pal, M. and Bhattacharjee, G. P., An optimal parallel algorithm to color an interval graph, *Parallel Processing Letters*, 6(4) (1996) 439-449.

[28] Pal, M. and Bhattacharjee, G. P., A data structure on interval graphs and its applications, *J. Circuits, Systems, and Computer*, 7 (1997) 165-175.

[29] Pal, M. and Bhattacharjee, G. P., An optimal parallel algorithm for all-pairs shortest paths on unweighted interval graphs, *Nordic J. Computing*, 4 (1997) 342-356.

[30] Pal, M., Efficient algorithms to compute all articulation points of a permutation graph, *Korean J. of Computational and Applied Mathematics,* 5 (1998) 141-152.

[31] Pal, M., A parallel algorithm to generate all maximal independent sets on permutation graphs, *Intern. J. Computer Maths.*, 67 (1998) 261-274.

[32] Pal, M., Mondal, S., Bera, D. and Pal, T. K., An optimal parallel algorithm for computing cutvertices and blocks on interval graphs, *Intern. J. Computer Math.,* 75 (2000) 59-70.

[33] Ramalingam, G. and Pandu Rangan, C., A unified approach to domination problem in interval graphs, *Information Processing Letters*, 27 (1988) 271-274.

[34] Saha, Anita, Pal, M. and Pal, T.K., An optimal parallel algorithm to construct a tree 3-spanner on interval graphs, *International J. Computer Mathethics*, 82 (3) (2005) 259-274.

[35] Saha, Anita, Pal, M. and Pal, T.K., An optimal parallel algorithm for solving all-pairs shortest paths problem on circular-arc graph, *Journal of Applied Mathematics and COmputing*, 17 (1-2) (2005) 1-23.

[36] Saha, Anita and Pal, M. An algorithm to find a minimum feedback vertex set of an interval graph, *Advanced Modeling and Optimization (An Electronic International Journal)*, 7(1) (2005) 99–116.

[37] Saha, Anita, Pal, M. and Pal, T.K., An efficient PRAM algorithm for maximum weight independent set on permutation graphs, *Journal of Applied Mathematics and Computing*, 19 (1-2) (2005) 77-92.

[38] Saha, Anita, Pal, M. and Pal, T.K., Selection of programme slots of television channels for giving advertisement: A graph theoretic approach, *Information Sciences*, 177(2) (2007) 2480-2492.