Computation of inverse 1-centre location problem on the weighted interval graphs

Biswanath Jana*

Department of Applied Mathematics with
Oceanology and Computer Programming,
Vidyasagar University,
721102, Midnapore, India
Email: biswanathjana2012@gmail.com
*Corresponding author

Sukumar Mondal

Department of Mathematics,
Raja N. L. Khan Women's College,
Gope Palace, 721102
Paschim Medinipur, India
Email: sm5971@rediffmail.com

Madhumangal Pal

Department of Applied Mathematics with
Oceanology and Computer Programming,
Vidyasagar University,
721102, Midnapore, India
Email: mmpalvu@gmail.com

Abstract: Let T_{IG} be the tree corresponding to the weighted interval graph $G = (V, E)$. In an inverse 1-centre location problem the parameter of an interval tree T_{IG} corresponding to the weighted interval graph $G = (V, E)$, like vertex weights have to be modified at minimum total cost such that a pre-specified vertex $s \in V$ becomes the 1-centre of the interval graph G. In this paper, we present an $O(n)$ time algorithm to find an inverse 1-centre location problem on the weighted tree T_{IG} corresponding to the weighted interval graph, where the vertex weights can be changed within certain bounds and n is the number of vertices of the graph G.

Keywords: tree-networks; centre location; 1-centre location; inverse 1-centre location; inverse optimisation; tree; interval graphs.

Reference to this paper should be made as follows: Jana, B., Mondal, S. and Pal, M. (2017) 'Computation of inverse 1-centre location problem on the weighted interval graphs', *Int. J. Computing Science and Mathematics*, Vol. 8, No. 6, pp.533–541.

Biographical notes: Biswanath Jana received his Bachelor of Science degree with honours in Mathematics from Vidyasagar University, West Bengal, India in 2002 and he has passed Master of Science (Applied Mathematics) in

Mathematics with first class from the same university in 2004. He received his Bachelor of Education degree in 2006 from the same university. Currently, he is an Assistant Teacher in Mathematics of Jhargram Bikash Bharati Sikshayatan, Jhargram, Paschim Medinipur, West Bengal, India.

Sukumar Mondal received his BSc (Hons) and MSc in Applied Mathematics (rank second) from Vidyasagar University, West Bengal, India in 1992 and in 1994 from the same university. He was also awarded an individual research fellowship by CSIR, Govt. of India in 1997. He was awarded Post Doctorial Research Award (2006–2009) by UGC, India. Currently, he is an Associate Professor of Department of Mathematics of Raja N.L. Khan Women's College, Medinipur, West Bengal, India. He has guided two research scholars for PhD degrees, has published more than 22 articles in international journals, and published three books for honours students.

Madhumangal Pal is a Professor of Applied Mathematics at Vidyasagar University, India. He has published more than 250 articles in international and national journals. His specialisations include computational and fuzzy graph theory, fuzzy matrices and fuzzy algebra. He is the Editor-in-Chief of *Journal of Physical Sciences* and *Annals of Pure and Applied Mathematics*. He has written eight books for UG and PG students. He successfully guided 26 PhDs.

1 Introduction

An undirected graph $G = (V, E)$ is an *interval graph* if the vertex set V can be put into one-to-one correspondence with a set of intervals I on the real line R such that two vertices are adjacent in G if and only if their corresponding intervals have non-empty intersection. The set I is called an *interval representation* of G and G is referred to as the *intersection graph* of I (Golumbic, 2004). Let $I = \{i_1, i_2, \ldots, i_n\}$, where $i_c = [a_c, b_c]$ for $1 \leq c \leq n$, be the interval representation of the graph G, a_c is the left endpoint and b_c is the right end point of the interval i_c.

If the intervals have common end points then the Algorithm CONVERT (Pal and Bhattacharjee, 1995) may be used to convert the intervals of I into intervals of distinct end points. Here we consider the weighted interval graphs, i.e., corresponding to each interval i, we put a positive weight $w_i > 0$.

An interval graph and its interval diagram are shown in Figure 1(a) and Figure 1(b) respectively.

In vertex weighted tree $T = (V, E)$, the eccentricity $e(v)$ of the vertex v is defined as the sum of the weights of the vertices from v to the vertex farthest from $v \in T$, i.e.,

$$e(v) = \max\left\{d_w\left(v, v_i\right), \text{for all } v_i \in T\right\},$$

where $d_w(v, v_i)$ is the sum of the weights of the vertices on the path between v and v_i.

A vertex with minimum eccentricity in the tree T is called a centre of that tree T, i.e., if $e(s) = \min\{e(v)$, for all $v \in V\}$, then s is the 1-centre. It is clear that every tree has either one or two centres. The eccentricity of a centre in a tree is defined as the radius of the tree and is denoted by $\rho(T)$, i.e.,

$$\rho(T) = \left\{\min_{v \in T} e(v)\right\}.$$

For the weighted tree T with n vertices and $n - 1$ edges of the corresponding weighted interval graphs, the inverse 1-centre problem on weighted tree T is concerned with modifying parameter, like vertex weight, at minimum total cost within certain modification bounds such that a pre-specified vertex becomes the 1-centre.

1.1 Survey and applications of the problem

As shown in Marlow (1983), Zhang and Liu (1996) and Zhang and Ma (1996), the inverse problems of many combinatorial/network optimisation problems can be solved by strongly or weakly polynomial algorithms. In the context of location problems Cai et al. (1999) proved that the inverse 1-centre location problem with edge length modification on general un-weighted directed graphs is NP-hard, while the underlying centre location problem is solvable in polynomial time. Recently, Yang and Zhang (2008) proposed an $O(n^2 \log n)$ time solution method for the inverse vertex centre problem on a tree provided that the modified edge lengths always remain positive. Recently, Alizadeh and Burkard (2009) have designed an algorithm for inverse 1-centre location problem with edge length augmentation on trees in $O(n \log n)$ time, using a set of suitably extended AVL-search trees. In Alizadeh and Burkard (2011), they have designed a combinatorial algorithm for inverse absolute on trees in $O(n^2)$ time when topology not allowed and $O(n^2r)$ time when topology allowed. Recently, Jana et al. (2012) have designed a linear time algorithm to compute inverse 1-centre location problem on the edge weighted trees.

In this paper, we propose an algorithm to compute inverse 1-centre location problem on weighted interval graphs in $O(n)$ time, where n is the number of vertices of the graph.

For instance, an important application comes from geophysical sciences and concerns predicting the movements of earthquakes. To achieve this aim, geologic zones are discretised into a number of cells. Adjacency relations can be modelled by arcs in a corresponding network. Although some estimates for the transmission times are known, precise values are hard to obtain. By observing an earthquake and the arrival times of the resulting seismic perturbations at various points and assuming that earthquakes travel along shortest paths, the problem is to refine the estimates of the transmission times between the cells. This is just an inverse shortest path problem.

2 Construction of the tree

Let i be pre-specified vertex which to be inverse 1-centre. Our target is to form a spanning tree corresponding to the weighted interval graph with two branches. Let the vertex i be the root of the tree. Then we find all adjacent vertices to i and set them as child (leaves) of i. To form the spanning trees we have the following two cases:

Case 1 If number of adjacent of i is one, i.e., $\deg(i) = 1$, then we can not construct a tree with root i and two branches. Therefore, vertex i is not inverse 1-centre of the weighted interval tree.

Case 2 If number of adjacent vertices to the vertex i are more than one, i.e., $\deg(i) > 1$, then three possibilities arises and we try to form a tree with two longest branches.

a When i is the starting vertex in G, i.e., $i = 1$.

In this case we find all adjacent vertices to the vertex 1 and set them as child (leaves) of 1 and marked them. Next we consider the vertices k and j whose right end points of the corresponding intervals are maximum and next maximum respectively. Next find all unmarked adjacent vertices to the vertices k and j respectively. If there is no common adjacent vertices to k and j, then find m_1, interval whose right end point is maximum among all adjacent to k and all unmarked adjacent are placed as the child of k and marked them else m_1' as child of j and marked. This process is continued until all intervals right to 1 are marked.

b When i is the end vertex in G, i.e., $i = n$.

In this case we find all adjacent vertices to the vertex n and set them as child (leaves) of n and marked them. Next we consider the vertices j' and k' whose left end points of the corresponding intervals are minimum and next minimum respectively. Next find all unmarked adjacent vertices to the vertices j' and k' respectively. If there is no common adjacent vertices to j' and k', then find m_1, interval whose left end point is minimum among all adjacent to j' and unmarked adjacent are placed as the child of j' and marked them else m_1' as child of k'. This process is continued until all intervals left to n are marked.

c When i is the vertex between 1 and n, i.e., $1 < i < n$.

In this case we find all adjacent vertices to the vertex i and set them as child (leaves) of i and marked them. Next, we consider the vertex k whose right end point of the corresponding interval among all adjacent vertices to i is maximum. Corresponding to the vertex k we find all unmarked vertices adjacent to k and put them as the child of k. Continuing this process on the right side of the interval diagram until all vertices corresponding to the intervals right of i are marked. Similarly, on the left side of i, we find j', the unmarked adjacent to i, and put them as child in left branch. Then same procedure is applied on left side until all intervals left of i are marked.

Now we propose a combinatorial algorithm to construct the tree T_{IG}. Our proposed algorithm is as follows:

```
Algorithm INT-TREE
Input:    Weighted interval graph $G$ with interval representation $I = [i_1, i_2, \dots, i_n]$, $i_j = [a_j, b_j]$
          and weight $w_j$, $j = 1, 2, \dots, n$.
Output:   The rooted tree $T_{IG}$ with two branches of the interval graph $G$.
Step 1    Set root = $i$ and compute $N(i)$ = the open neighbourhood of $i = \{v : (v, i) \in E\}$.
Step 2    If $|N(i)| = 1$, then end.
            If $|N(i)| > 1$ and $i$ is the starting interval, i.e., $i = 1$, then goto step 3.
            If $|N(i)| > 1$ and $i$ is the end interval, i.e., $i = n$, then goto step 4.
            If $|N(i)| > 1$ and $i$ is an interval between 1 and $n$, i.e., $1 < i < n$, then goto step 5.
Step 3    Set $N(i)$ as the child of the root $i$ and marked them.
            Step 3.1    Set $k = \max\{b_k: (k, i) \in E\}$, $j = \max\{b_j: (j, i) \in E, k \neq j$ and $b_j < b_k\}$.
```

```
        Step 3.2   Find unmarked adjacent of $j$ and k and if $N(j) \cap N(k) = \phi$, then
                   $m_1 = \max\{b_{m_1} : (m_1, k) \in, m_1 \in N(k)\}$ and set all unmarked $N(k)$ as the
                   child of $k$ and marked them.
        else $m_1' - \max\{bm_1' \in N(k) \cap N(j)\}$ set as child of $j$ and $\{N(k) \cup N(j) - \{m_1'\}\}$
        as child of k and marked and find $m_1'' = \max\{N(k) \cup N(j) - \{m_1'\}\}$.
        Step 3.3   This process is continued until all intervals are marked.
        Step 3.4   Compute the interval tree $T_{IG}$.
Step 4  Set $N(i)$ as the child of the root $i$ and marked them.
        Step 4.1   Set $j' = \min\{a_{j'} : (j', i) \in E\}$,
                   $k' = \min\{a_{k'} : (k', i) \in E, k' \neq j'$ and $a_j' < a_k'\}$.
        Step 4.2   Find unmarked adjacent of $j'$ and $k'$ and if $N(j') \cap N(k') = \phi$, then
                   $m_1 = \min\{a_{m_1} : (m_1, j')m_1' \in N(k') \cap N(j')\}$ and set all unmarked
                   $N(j')$ as the child of $j'$ and marked them.
                   else $m_1' = \min\{a_{m_1'} : m_1' \in N(k') \cap N(j')\}$ set as child of $k'$ and
                   $\{N(k') \cup N(j') - \{m_1'\}\}$ as child of $j'$ and marked and find
                   $m_1'' = \min\{a_{m_1''} : m_1'' \in \{N(k') \cup N(j') - \{m_1'\}\}\}$.
        Step 4.3   This process is continued until all intervals are marked.
        Step 4.4   Compute the interval tree $T_{IG}$.
Step 5  Set $N(i)$ as the child of the root $i$ and marked them.
        Step 5.1   Set $p = \max\{b_p : (p, i) \in E\}$, $q = \min\{a_q : (q, i) \in E\}$ and $p \neq q$.
        Step 5.2   Set $p' = \max\{b_{p'} : (p', p) \in E, p' \in N(p)\}$ and set all unmarked $N(p)$ as
                   the child of $p$ and marked.
        Step 5.3   Set $q' = \min\{a_{q'} : (q', q) \in E, q' \in N(q)\}$ and set all unmarked $N(q)$ as
                   the child of $q$ and marked.
        Step 5.4   This process is continued until all intervals are marked.
        Step 5.5   Compute the interval tree $T_{IG}$.
Step 6  Put weight $w_j$ to the vertex $j$ in $T_{IG}$ corresponding to the interval $j$ of the interval
        graph $G$.
end INT-TREE
```

The weighted tree T_{IG} with root as the vertex 1 of the interval graph G is shown in Figure 2 and we have the following important observation on T_{IG}.

Lemma 1: The tree T_{IG} formed by the Algorithm **INT-TREE** is a spanning tree.

Theorem 1: The time complexity of the Algorithm **INT-TREE** is $O(n)$, where n is the number of vertices of the tree.

Proof: step 1 takes $O(n)$ time, since the intervals are sorted and the root is selected from n intervals. Similarly step 2 can be computed in $O(n)$ time. Since the end points of the intervals are sorted and intersection of two finite sets of n elements can be executed in $O(n)$ time, so step 3, or step 4, or step 5 can be computed in $O(n)$ time. Also step 6 takes $O(n)$ time. Hence overall time complexity of our proposed **Algorithm INT-TREE** is $O(n)$ time.

3 Inverse 1-centre, algorithm and its complexity

Here, we introduce some notations for our algorithmic purpose.

R_i is the longest weighted path right to the vertex i; L_i is the longest weighted path left to the vertex i does not contain any vertex of the path R_i; $w(R_i)$, $w(L_i)$ are sum of weights of the vertices of the path R_i, L_i, respectively; $w^*(R_i)$; $w^*(L_i)$ are sum of weights of the vertices of the path R_i, L_i after modification, respectively, $w_{low} = \min\{w(L_i), w(R_i)\}$, $w_{high} = \max\{w(L_i), w(R_i)\}$; k is the number of vertices in such path between L_i, R_i whose weight is maximum, except the vertex i; $w_{low}(v) = \min\{w(v) : v \in T_{IG}\} = w'$; $w_{upp}(v) = \max\{w(v) : v \in T_{IG}\}$; T_{IG} is the weighted interval tree corresponding to the interval graph G; T'_{IG} is the modified tree of the tree T_{IG} corresponding to the interval graph G.

To find inverse 1-centre we discuss following two cases:

Case 1 if sum of weights of one side of the root i is equal to the sum of weights of other side, i.e., $w(L_i) = w(R_i)$, then i is the centre as well as the inverse 1-centre of the graph.

Case 2 If $w(L_i) \neq w(R_i)$, then we have following three subcases:

Case 2.1 When $w_{low} = kw'$.

Case 2.2 When $w_{low} > kw'$.

Case 2.3 When $w_{low} < kw'$.

Under above conditions we modify the tree T_{IG} with the help of following non-linear semi-infinite (or nonlinear) optimisation model:

$$\text{Minimize} \sum_{v \in V(T_{IG})} \left\{ c^+(w(v))x(w(v)) + c^-(w(v))y(w(v)) \right\}$$

subject to

$$\max_{v\in V(T_{IG})} d_{\bar{w}}(v,i) \le \max_{v\in V(T_{IG})} d_{\bar{w}}(v,p), \text{ for all } p \in V\left(T_{IG}\right),$$

$$\bar{w}(v) = w(v) + x\{w(v)\} - y\{w(v)\} \text{ for all } v \in V\left(T_{IG}\right),$$

$$x\{w(v)\} \le w^+\{w(v)\} \text{ for all } v \in V\left(T_{IG}\right),$$

$$y\{w(v)\} \le w^-\{w(v)\} \text{ for all } v \in V\left(T_{IG}\right),$$

$$x\{w(v)\}, y\{w(v)\} \ge 0 \text{ for all } v \in V\left(T_{IG}\right),$$

where $\bar{w}(v)$ be the modified vertex weight, $w^+\{w(v)\} = w_{upp}(v) - w(v)$ and $w^-\{w(v)\} = w(v) - w_{low}(v)$ are the maximum feasible amounts by which $w(v)$ can be increased and reduced respectively, i.e., $w_{low}(v) \le \bar{w}(v) \le w_{upp}(v)$, $x\{w(v)\}$ and $y\{w(v)\}$ are the amounts by which the vertex weight $w(v)$ is increased and reduced respectively, $c^+(w(v))$ is the non negative cost if $w(v)$ is increased by one unit and $c^-(w(v))$ is the non negative cost if $w(v)$ is reduced by one unit. Every feasible solution (x, y) with $x = \{x(w(v)) : v \in V(T_{IG})\}$ and $y = \{y(w(v)) : e \in V(T_{IG})\}$ is also called a feasible modification of the inverse 1-centre location problem.

Now, we have the following three important results.

Lemma 2: If $w_{low} = kw'$ in T_{IG}, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices except the vertex i, i.e., root i up to minimum weight maintaining the bounding condition in the path whose weight is maximum and i is the inverse 1-centre.

Lemma 3: If $w_{low} > kw'$ in T_{IG}, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of some vertices except the root i maintaining the bounding condition in the path whose weight is maximum and i is the inverse 1-centre.

Lemma 4: If $w_{low} < kw'$ in T_{IG}, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices up to minimum weight except the root i maintaining the bounding condition in the path whose weight is maximum and enhance the weights of some vertices except the root i in the path whose weight is minimum and i is the inverse 1-centre.

Our proposed algorithm to the inverse 1-centre location problem is as follows:

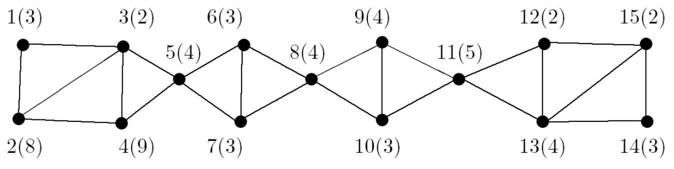
```
Algorithm 1-INV-INT-LOC-TREE
Input:   Weighted interval graph $G$ with interval representation $I = [i_1, i_2, \ldots, i_n]$, $i_j = [a_j, b_j]$, $j = 1, 2, \ldots, n$.
Output   Vertex $i$ as inverse 1-centre of the tree $T_{IG}$ and modified tree $T'_{IG}$.
Step 1   Construction of the tree $T_{IG}$ (as per Section 2) with root $i$.
Step 2   Compute the weighted path $R_i$ from $i$ to other vertex $v_j$ on the tree $T_{IG}$.
Step 3   Next compute weighted path $L_i$ from $i$ to the vertex $v_k$ does not contain any vertex of the path $R$ except $i$.
Step 4   Calculate weights of two paths $L_i$ and $R_i$, i.e., $w(L_i)$ and $w(R_i)$.
Step 5   //Modification of the tee $T_{IG}$//
    Step 5.1   If $w(L_i) = w(R_i)$, then $i$ is the vertex one centre as well as inverse 1-centre of $T_{IG}$.
    Step 5.2   If $w(L_i) \neq w(R_i)$, then
        Step 5.2.1   If $w_{low} = kw'$ in $T_{IG}$, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices except the vertex $i$, i.e., root $i$ up to minimum weight maintaining the bounding condition in the path whose weight is maximum, then go to step 5.3.
        Step 5.2.2   If $w_{low} > kw'$ in $T_{IG}$, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of some vertices except the root $i$ maintaining the bounding condition in the path whose weight is maximum, then go to step 5.3.
        Step 5.2.3   If $w_{low} < kw'$ in $T_{IG}$, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices except the root $i$ up to minimum weight maintaining the bounding condition in the path whose weight is maximum and enhance the weights of some vertices except the root $i$ in the path whose weight is minimum, then go to step 5.3.
    Step 5.3   $T'_{IG}$ = modified tree of the tree $T_{IG}$.
end 1-INV-INT-LOC-TREE.
```

Lemma 5: Algorithm **1-INV-INT-LOC-TREE** correctly computes the inverse 1-centre location on the weighted interval tree.

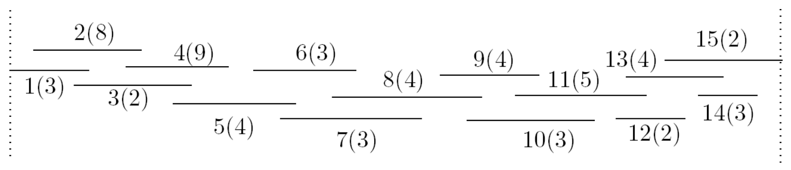
The modification of T_{IG} by this process, in general, is not unique but cost is always minimum. We have another important observation in the tree T'_{IG} given by the Algorithm **1-INV-INT-LOC-TREE.**

Lemma 6: The specified vertex i in the modified tree T'_{IG} is the inverse 1-centre.

Figure 1 (a) An interval graph *G* (b) Interval matching diagram of the interval graph *G* of (a)



(a)



(b)

Figure 2 Tree T_{IG} of the interval graph G with longest branch on both sides by checking adjacency

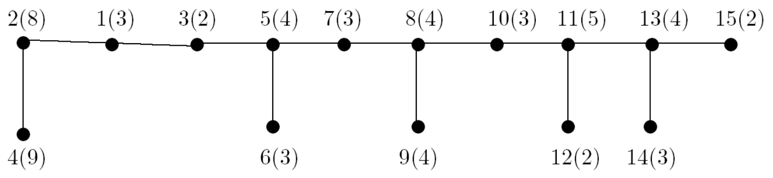
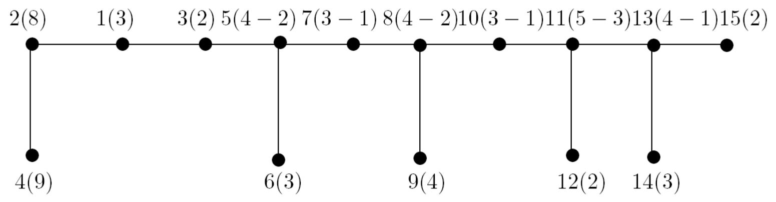


Figure 3 Modified tree T'_{IG} of the tree T_{IG}



Theorem 2: the time complexity to find inverse 1-centre problem on a given vertex weighted interval tree T_{IG} is $O(n)$, where n is the number of vertices of the tree.

Proof. Step 1 takes $O(n)$ time, since the adjacency relation of interval graph can be tested in $O(n)$ time. Step 2, i.e., longest weighted path from i to v_i can be computed in $O(n)$ time if T is traversed in a depth- first-search manner. Similarly, step 3 can be computed in

$O(n)$ time. Step 4 takes $O(n)$ time to compute he sum of the weights of the paths. Also, step 5 takes $O(n)$ time. Since comparing two numbers and distribution of the excess weight takes $O(n)$ time, so, step 5.1, step 5.2 and step 5.3 can be computed $O(n)$ time. Hence overall time complexity of our proposed **Algorithm 1-INV-INT-LOC-TREE** is $O(n)$ time, where n is the number of vertices of the interval graph.

4 Concluding remarks

In this article, we investigated the inverse 1-centre location problem with different vertex weights on the tree corresponding to the weighted interval graph G. We developed an exact combinatorial solution algorithm for the tree of interval graphs with fast running time $O(n)$, where n is the number of vertices of the interval graph.

Acknowledgements

The authors thank the anonymous referees for their thoughtful and valuable remarks which led to improvement of this paper.

References

Alizadeh, B. and Burkard, R.E. (2009) 'Inverse 1-center location problems with edge length augmentation on tree', *Computing*, Vol. 86, No. 4, pp.331–343.

Alizadeh, B. and Burkard, R.E. (2011) 'Combinatorial algorithms for inverse absolute and vertex 1-center location problems on trees', *Networks*, Vol. 58, No. 1, pp.190–200.

Cai, M.C., Yang, X.G. and Zhang, J.Z. (1999) 'The complex analysis of the inverse center location problem', *Journal of Global Optimization*, Vol. 15, No. 2, pp.213–218.

Golumbic, M.C. (2004) *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York.

Jana, B., Mondal, S. and Pal, M. (2012) 'Computation of inverse 1-center location problem on the weighted trees', *CiiT International Journal of Networking and Communication Engineering*, Vol. 4, No. 2, pp.70–75.

Marlow, B. (1983) 'Inverse problems', in Megiddo, N. (Ed.): 'Linear-time algorithms for linear programming in R^3 and related problems', *SIAM J. Comput.*, Vol. 12, No. 4, pp.759–776.

Pal, M. and Bhattacharjee, G.P. (1995) 'The parallel algorithms for determining edge-packing and efficient edge dominating sets in interval graphs', *Parallel Algorithms and Applications*, Vol. 7, No. 3, pp.193–207.

Yang, X. and Zhang, J. (2008) 'Inverse center location problem on a tree', *Journal of Systems Science and Complexity*, Vol. 21, No. 4, pp.651–664.

Zhang, J.Z. and Liu, Z.H. (1996) 'Calculating some inverse linear programming problems', *J. Computational and Applied Mathematics*, Vol. 72, No. 2, pp.261–273.

Zhang, J.Z. and Ma, Z.F. (1996) 'A network flow method for solving some inverse combinatorial optimization problems', *Optimization*, Vol. 37, No. 1, pp.59–72.