An Improved Algorithm for the p-Center Problem on Interval Graphs with Unit Lengths*

T.C.E. Cheng[1], Liying Kang[1,2], C.T. Ng[1]

[1]Department of Logistics, The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong

[2]Department of Mathematics, Shanghai University
Shanghai 200444, China

Abstract

The p-center problem is to locate p facilities in a network of n demand points so as to minimize the longest distance between a demand point and its nearest facility. We consider this problem by modelling the network as an interval graph whose edges all have unit lengths. We present an $O(n)$ time algorithm for the problem under the assumption that the endpoints of the intervals are sorted, which improves on the existing best algorithm for the problem that has a run time of $O(pn)$.

Key words: p-center problem; interval graph; location

AMS subject classification: 05C85; 90C27

1 Introduction

Network location problems are concerned with finding the right locations to place one or more facilities in a network of demand points, i.e., customers represented by nodes in the network, that optimize a certain objective function related to the distance between the facilities and the demand points. Usually the facilities to be located are desirable, i.e., customers prefer to have the facilities located as close to them as possible. For example, services such as police and fire stations, hospitals, schools, and shopping centers are typical desirable facilities.

*This research was supported in part by The Hong Kong Polytechnic University under a grant from the *Area of Strategic Development in China Business Services*. The second author was also supported by the National Natural Science Foundation of China.

The p-center problem is to locate p facilities in a network so as to minimize the longest distance between a set of n demand points and the p facilities. This problem is central to the field of location theory and logistics, and has been subject to extensive research [4, 8, 15, 18, 19]. Applications of the p-center problem include the location of industrial plants, warehouses, distribution centers, and public service facilities in transportation networks, as well as the location of various service facilities in telecommunication networks [4, 8, 15, 18, 19].

We model the network as a graph $G = (V, E)$, where V is the vertex set with $|V| = n$ and E is the edge set with $|E| = m$. We assume that the demand points coincide with the vertices, and restrict the location of the facilities to the vertices. We also assume that each edge of E has a unit length. The *distance* $d(u, v)$ between two vertices u, v is the number of edges on a shortest u-v path from u to v in G. We write $d(u, U) = \min_{v \in U} d(u, v)$ for a set $U \subset V$. The p-center problem then becomes that of finding a set $X \subset V$ such that $|X| = p$ and for which $\max\{d(v_i, X) : v_i \in V(G)\}$ is minimum.

A graph $G = (V, E)$ is called an *intersection graph* for a finite family $\mathcal{F}$ of a nonempty set if there is a one-to-one correspondence between $\mathcal{F}$ and V such that two sets in $\mathcal{F}$ have nonempty intersection if and only if their corresponding vertices in V are adjacent to one another. We call $\mathcal{F}$ an *intersection model* of G. For an intersection model $\mathcal{F}$, we use $G(\mathcal{F})$ to denote the intersection graph for $\mathcal{F}$. If $\mathcal{F}$ is a family of intervals on a real line, then G is called an *interval graph* for $\mathcal{F}$, and $\mathcal{F}$ is called an *interval model* of G. If $\mathcal{F}$ is a family of arcs on a circle, then G is called a *circular-arc* graph for $\mathcal{F}$ and $\mathcal{F}$ is called a *circular-arc model* of G. A set D of the vertices of G is a *dominating set* of G if every vertex in $V - D$ is adjacent to some vertex in D. A dominating set that induces a path in G is called a *dominating path* of G.

Interval graphs arise in many application areas, such as archeology, biology, logistics, scheduling, traffic control, and VLSI design [9, 12]. The problems of recognizing interval and circular-arc graphs are known to be solvable in $O(m + n)$ time (see, e.g., [3]). We assume that an interval model $\mathcal{F}$ for G is available. In the rest of this paper, we assume that the endpoints of the intervals in $\mathcal{F}$ are already sorted.

The p-center problem is known to be NP-complete [7]. Olariu [17] provided an $O(n)$ time algorithm for locating a single facility on an interval graph that minimizes the maximum distance to a demand point. Frederickson [6] showed how to solve this problem for trees in optimal linear time (without necessarily restricting the location of the facilities to the vertices of the tree) using parametric search. Bespamyatnikh et al. [2] gave an $O(pn)$ time algorithm for the p-center problem on circular-arc graphs. It is easy to see that the p-center problem on general interval graphs (with unit edge lengths) is a special case of circular-arc graphs, and therefore it can be solved in $O(pn)$ time. Hsu et al. [11] presented a general p-facility location problem on

the real line with unimodal distance functions. For this specially structured problem, they gave an $O(pn^2)$ algorithm. Kariv and Hakimi [13] presented the results for the p-center problem on general graphs; however, since the problem is known to be NP-complete, they were able to give only an $O(n^{2p+1} \log n/(p-1)!)$ time algorithm. Tamir [18] showed that the weighted and unweighted p-center problems in networks can be found in $O(n^p m^p \log^2 n)$ and $O(n^{p-1} m^p \log^3 n)$ time, respectively. In addition, some work has been done on approximating the p-center solution (see, e.g., [1, 10]).

The complete graph is an interval graph. The p-dominating set problem on a general graph is trivially reducable to a p-center problem on a complete graph with edge lengths equal to 1 or 2. The dominating set problem is known to be NP-complete [7]. So the p-center problem on interval graphs with general edge lengths is NP-hard. In this paper we present an $O(n)$ algorithm for the p-center problem on interval graphs with unit lengths under the assumption that the endpoints of the intervals are sorted.

In the next section we introduce some useful notation and some results pertinent to interval graphs. Section 3 presents an $O(n)$ time algorithm for the p-center problem under study in this paper.

2 Preliminaries

Let $\mathcal{F}$ be the set of n intervals in the interval model for G. Every interval $I_i \in \mathcal{F}$ is defined by its left endpoint a_i and right endpoint b_i, i.e., $I_i = [a_i, b_i]$, with $a_i \leq b_i$. Without loss of generality, we assume that no two distinct intervals in $\mathcal{F}$ have the same endpoints and that $a_i < b_i$ for every $1 \leq i \leq n$. Let L be the sorted array containing the endpoints of the intervals in $\mathcal{F}$. Let $L = (p_1, p_2, \ldots, p_{2n})$, where every p_i is either a_j or b_j for some j. Note that from L we can easily obtain the list of all the intervals in $\mathcal{F}$ that are sorted by the a_i's (respectively, b_i's). Hence, we assume that the intervals in $\mathcal{F}$ have been labelled in such a way that $b_i < b_j$ if and only if $i < j$. Such a labelling is easily obtained from L in $O(\log n)$ time using $n/\log n$ processors by parallel prefixing [14, 16]. Let s (respectively, t) be the smallest left endpoint (respectively, largest right endpoint) among those in L. Then, $s = a_l$ for some $l \in \{1, 2, \ldots, n\}$ and $t = b_n$. Without loss of generality, we assume that $\cup_{i=1}^{n} I_i$ is equal to the interval $[s, t]$, i.e., the union of the intervals in $\mathcal{F}$ is one connected component $[s, t]$ on the real line. This is because if it is not the case, then G is not connected, and we can consider the connected components of G.

Chen et al. [5] defined a successor function on intervals to solve the all-pair shortest path

problem on interval and circular-arc graphs. Bespamyatnihk et al.[2] used Chen's idea to define a *right successor function* and a *left successor function* of an integer q as follows: For every point q on the real line, $RSUC(q) = I_i \in \mathcal{F}$ if and only if $b_i = \max\{b_j \mid I_j \text{ contains } q\}$, and $LSUC(q) = I_i \in \mathcal{F}$ if and only if $a_i = \min\{a_j \mid I_j \text{ contains } q\}$. For an interval I_i, $RSUC(I_i) = RSUC(b_i)$ and $LSUC(I_i) = LSUC(a_i)$. Note that, because $\cup_{i=1}^{n} I_i$ is a connected component, $RSUC(I_i) = I_i$ implies that $i = n$. For example, in Figure 1, $RSUC(I_1) = I_4$,

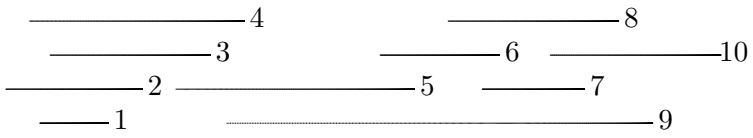

Figure 1.

$RSUC(I_4) = I_9$, $RSUC(I_9) = I_{10}$ and $LSUC(I_{10}) = I_9$, $LSUC(I_9) = I_4$, $LSUC(I_4) = I_2$. Furthermore, the *ith* iterated successor $RSUC(q, i)$ of an integer q is defined as $RSUC(RSUC(\ldots RSUC(q, 0)))$, where $RSUC$ appears $i + 1$ times and $RSUC(q, 0) = q$. Similarly, we can define $LSUC(q, i)$. For any interval I_j, we define $RSUC(I_j, i)$ to be $RSUC(b_j, i)$, and $LSUC(I_j, i)$ to be $LSUC(a_j, i)$. In Figure 1, for example, $RSUC(I_1, 1) = I_4, RSUC(I_1, 2) = I_9, RSUC(I_1, 3) = I_{10}$ and $LSUC(I_{10}, 1) = I_9, LSUC(I_{10}, 2) = I_4, LSUC(I_{10}, 3) = I_2$. Observe that the $RSUC$ function actually gives rise to a tree structure T (RSUC-TREE) whose nodes correspond to the intervals in $\mathcal{F}$. We define, for every $I_i, 1 \leq i < n$, that $RSUC(I_i)$ is the parent node in T corresponding to node I_i. Using the tree data structure based on the successor function, Chen et al. [5] were able to compute iterated successors in constant time.

Lemma 2.1 *After an $O(n)$ time preprocessing step, given interval $I_q \in \mathcal{F}$ and $i \in [1 \ldots n]$, $RSUC(I_q, i)$ or $LSUC(I_q, i)$ can be computed in constant time.*

Chen et al. [5] made further use of their tree structure to obtain the following result.

Lemma 2.2 *After an $O(n)$ time preprocessing step, given two intervals $I_i \in \mathcal{F}$ and $I_j \in \mathcal{F}$, the distance between I_i and I_j in $G(\mathcal{F})$ can be computed in constant time.*

By the definition of the successor function, it is easy to see that for any $I_i \in \mathcal{F}$, there exists an integer $l(I_i)$ such that $RSUC(I_i, l(I_i) - 1) = I_n$. Let $RSUC(I_1, 1) = I_{i_2}, RSUC(I_1, 2) =$

$I_{i_3}, \ldots, RSUC(I_1, l(I_1)-1) = I_{i_{l(I_1)}} = I_n$. Clearly, $\{I_1 = I_{i_1}, I_{i_2}, \ldots, I_{i_{l(I_1)}} = I_n\}$ is a dominating set of G, and $(I_1 = I_{i_1}, I_{i_2}, \ldots, I_{i_{l(I_1)}} = I_n)$ is a dominating path of G. We denote this path by $P(I_1)$. In fact, $P(I_1)$ is the unique path with length $l(I_1)-1$ from I_1 to I_n in T (RSUC-TREE). For each $I_q \in V(G) - V(P(I_1))$, if I_q is adjacent to at least two vertices on $P(I_1)$, then all the vertices on $P(I_1)$ adjacent to I_q must be consecutive vertices of $P(I_1)$. Furthermore, we have the following result.

Lemma 2.3 *For any $I_q \in V(G)-V(P(I_1))$, I_q is adjacent to at most three consecutive vertices on $P(I_1)$ in $G(\mathcal{F})$.*

Proof. Suppose to the contrary that I_q is adjacent to four consecutive vertices on $P(I_1)$. Denote the four vertices by $I_{i_k}, I_{i_{k+1}}, I_{i_{k+2}}$ and $I_{i_{k+3}}$. From the definition of the right successor function, it must be the case that $a_{i_{k+3}} > b_{i_{k+1}}$. Moreover, both $I_q I_{i_k} \in E(G)$ and $I_{i_{k+1}} = RSUC(I_{i_k})$ imply that $b_{i_{k+1}} > b_q$. Hence $a_{i_{k+3}} > b_q$, contradicting our assumption that $I_q I_{i_{k+3}} \in E(G)$. The lemma follows. ■

For notational convenience, we define $c = \lceil l(I_1)/p \rceil$, $d = \lceil (c-1)/2 \rceil$. If p divides $l(I_1)$, we define

$$C(I_1) = \begin{cases} \{I_{i_{d+1}}, I_{i_{(d+1)+2d}}, \ldots, I_{i_{(d+1)+2(p-1)d}}\} & \text{if } c \text{ is even,} \\ \{I_{i_{d+1}}, I_{i_{(d+1)+(2d+1)}}, \ldots, I_{i_{(d+1)+(p-1)(2d+1)}}\} & \text{if } c \text{ is odd.} \end{cases}$$

If p does not divide $l(I_1)$, let $l(I_1) = (c-1)p + r$ $(0 < r < p)$, and we define

$$C(I_1) = \begin{cases} \{I_{i_{d+1}}, I_{i_{(d+1)+2d}}, \ldots, I_{i_{(d+1)+2(r-1)d}}, I_{i_{(d+1)+2(r-1)d+2(d-1)+1}}, \\ \ldots, I_{i_{(d+1)+2(r-1)d+(p-r)[2(d-1)+1]}}\} & \text{if } c \text{ is even,} \\ \{I_{i_{d+1}}, I_{i_{(d+1)+(2d+1)}}, \ldots, I_{i_{(d+1)+(r-1)(2d+1)}}, I_{i_{(d+1)+(r-1)(2d+1)+2d}}, \\ \ldots, I_{i_{(d+1)+(r-1)(2d+1)+2(p-r)d}}\} & \text{if } c \text{ is odd.} \end{cases}$$

Theorem 2.1 *Let $OPT(\mathcal{F})$ be the value of the optimal solution for the p-center problem given an instance $\mathcal{F}$. Then, $d \le OPT(\mathcal{F}) \le d+1$.*

Proof. We first show that $OPT(\mathcal{F}) \le d+1$. From the construction of $C(I_1)$, it is easy to see that for each $I_q \in V(P(I_1))$, $d(I_q, C(I_1)) \le d$. Since $P(I_1)$ is a dominating path of G, it follows that $d(I_q, C(I_1)) \le d+1$ for each $I_q \in V(G)$. So, $OPT(\mathcal{F}) \le d+1$.

Suppose that S is an optimal solution for the problem. For each $I_q \in S \cap V(P(I_1))$, there exist at most $2OPT(\mathcal{F})+1$ vertices on $P(I_1)$ with distance no more than $OPT(\mathcal{F})$ to I_q. For any $I_q \in S - V(P(I_1))$, by Lemma 2.3, I_q is adjacent to at most three consecutive vertices on

$P(I_1)$, it follows that there exist at most $2OPT(\mathcal{F})+1$ vertices on $P(I_1)$ with distance no more than $OPT(\mathcal{F})$ to I_q. So, $(2OPT(\mathcal{F})+1)p \geq l(I_1)$.

We claim that $OPT(\mathcal{F}) \geq d$. Otherwise, $OPT(\mathcal{F}) \leq d-1$. If $c = \lceil l(I_1)/p \rceil$ is even, then $d = \lceil (c-1)/2 \rceil$ implies that $c-1 = 2(d-1)+1$. Hence $(2OPT(\mathcal{F})+1)p \leq (2(d-1)+1)p = (c-1)p < l(I_1)$, a contradiction. If $c = \lceil l(I_1)/p \rceil$ is odd, then $d = \lceil (c-1)/2 \rceil$ implies that $c-1 = 2d$. Hence $(2OPT(\mathcal{F})+1)p \leq (2(d-1)+1)p = (c-2)p < l(I_1)$, and we arrive at a contradiction again. So, $OPT(\mathcal{F}) \geq d$. ■

For the interval $RSUC(I_i, q)$ ($LSUC(I_i, q)$), we denote its left endpoint by $a(RSUC(I_i, q))$ ($a(LSUC(I_i, q))$) and its right endpoint by $b(RSUC(I_i, q))$ ($b(LSUC(I_i, q))$).

Lemma 2.4 *For any interval $I_i \in \mathcal{F}$ and an integer q, if $RSUC(I_i, q) = I_j$ with $j < n$, then for any interval I_k with $k > j$, $d(I_i, I_k) \geq q+1$.*

Proof. We apply induction on q. If $q = 1$, then $d(I_i, I_k) \geq 2$ for any $k > j$, for otherwise $I_k = RSUC(I_i, q)$. We now assume that the assertion holds for all positive integers $s < q$, i.e., if $RSUC(I_i, s) = I_j$ with $j < n$, then $d(I_i, I_k) > s+1$ holds for $k > j$. We consider the case $RSUC(I_i, q) = I_j$ with $j < n$. For $k > j$, it must be the case $a_k > b(RSUC(I_i, q-1))$, for otherwise $RSUC(I_i, q) = I_k$. Let $I_{k'}$ be the adjacent vertex of I_k on the shortest I_i-I_k path. $I_{k'}I_k \in E(G)$ implies that $b_{k'} > a_k$. Combining this with $a_k > b(RSUC(I_i, q-1))$, we have $b_{k'} > b(RSUC(I_i, q-1))$. Applying the inductive hypothesis, it follows that $d(I_i, I_{k'}) \geq q$. Thus $d(I_i, I_k) \geq q+1$, as required. ■

For every pair of intervals $I_i, I_k \in \mathcal{F}(i < k)$, we define $T_{i,k}$ as the set of intervals whose right endpoints are between b_i and b_k. The set $T_{i,k}$ can be partitioned into two subsets $T^1_{i,k}$ and $T^2_{i,k}$ such that the distance from every interval in $T^1_{i,k}$ to I_i is smaller than or equal to d, and the distance from every interval in $T^2_{i,k}$ to I_i is greater than d. If $T^2_{i,k} \neq \emptyset$, we define $L(I_i, I_k)$ as the longest distance between intervals in $T^2_{i,k}$ and I_k. Let $L(I_i, I_k) = 0$ if $T^2_{i,k} = \emptyset$.

To be able to efficiently compute $L(I_i, I_k)$ for each pair $I_i, I_k (i < k)$ of intervals in $\mathcal{F}$ with $T^1_{i,k} \neq \emptyset$, let $Q(I_i) = \{I_j \mid I_j \in \mathcal{F}, I_j$ is contained in $RSUC(I_i, d)$ and $I_j \cap RSUC(I_i, d-1) = \emptyset\}$.

Lemma 2.5 *If $Q(I_i) \neq \emptyset$, then $d(I_{q(I_i)}, I_k) = L(I_i, I_k)$, where $q(I_i) = min\{j \mid I_j \in Q(I_i)\}$.*

Proof. We first show that $d(I_i, I_{q(I_i)}) > d$. Let I_m be the direct predecessor of $I_{q(I_i)}$ on the shortest I_i-$I_{q(I_i)}$ path. It follows immediately that $b_m > a_{q(I_i)} > b(RSUC(I_i, d-1))$, since $I_{q(I_i)} \in Q(I_i)$ and $I_m I_{q(I_i)} \in E(G)$. By Lemma 2.4, we have $d(I_i, I_m) \geq d$. Hence,

$d(I_i, I_{q(I_i)}) > d$, i.e., $I_{q(I_i)} \in T^2_{j,k}$. On the other hand, $q(I_i) = \min\{j \mid I_j \in Q(I_i)\}$ implies that $I_j \notin T^2_{j,k}$ for any j, $i < j < q(I_i)$. So, $d(I_{q(I_i)}, I_k) = L(I_i, I_k)$, the result follows. ■

Now we can compute $q(I_i)$ by using the $RSUC$ function as follows:

Procedure $q(I_i)$

(a) Let $M_{p_i} = \{p_h : p_h < p_i\}$ for each p_i in L.

(b) For every $I_i = [a_i, b_i]$ with $b(RSUC(I_i, d)) < b_n$, let $M_{RSUC(I_i,d))} = M_{b(RSUC(I_i,d))} - M_{a(RSUC(I_i,d))}$, and compute the $M_{RSUC(I_i,d))}$.

(c) Find the interval I_j whose left and right endpoints belong to $M_{RSUC(I_i,d))}$ such that $a_j > b(RSUC(I_i, d-1))$ and b_j is as small as possible. Let $q(I_i) = j$.

It is easy to see that the computation of $q(I_i)$ takes $O(n)$ time.

Lemma 2.6 *Let $RSUC(I_i, d) = I_j$. If $Q(I_i) = \emptyset$, then $d(I_{j+1}, I_k) = L(I_i, I_k)$.*

Proof. First, by Lemma 2.4, we have $d(I_i, I_{j+1}) > d$, so $I_{j+1} \in T^2_{i,k}$. On the other hand, $Q(I_i) = \emptyset$ implies that $I_{j'} \notin T^2_{i,k}$ for any $j' < j+1$. Thus, $d(I_{j+1}, I_k) = L(I_i, I_k)$, as required. ■

3 The algorithm and analysis

In this section we describe an algorithm designed to solve the p-center on interval graphs. Let $I_{max(a)}$ be the interval of $\mathcal{F}$ with the maximum left endpoint.

Algorithm 3.1 *Find an optimal p-center set of $G(\mathcal{F})$ given a set $\mathcal{F}$ of sorted intervals.*

Input. A set $\mathcal{F}$ of sorted intervals.

Output. A minimum-value p-center set of $G(\mathcal{F})$.

Method.

1. Calculate the dominating path $P(I_1)$ and $C(I_1)$ (as defined in Section 2), let $V(P(I_1)) = \{I_1 = I_{i_1}, I_{i_2}, \ldots, I_{i_{l(I_1)}} = I_n\}$, and $C(I_1) = \{I_{k_1}, I_{k_2}, \ldots, I_{k_p}\}$.

2. Calculate $c = \lceil \frac{l(I_1)}{p} \rceil, d = \lceil \frac{c-1}{2} \rceil$.

3. $j := 1, I_{w_1} := I_{k_1}, S := \{I_{w_1}\}$

4. $j := j + 1$.

5. If $j = p + 1$ and $d(I_{max(a)}, I_{w_p}) \leq d$, output S.

6. If $j = p + 1$ and $d(I_{max(a)}, I_{w_p}) > d$, $S := C(I_1)$, output S.

7. If $j < p + 1$, $I_{w_j} := I_{k_j}$.

8. If $L(I_{w_{j-1}}, I_{w_j}) \leq d$, $I_{w_j} := I_{w_j+1}$.

   8.1. If $L(I_{w_{j-1}}, I_{w_j}) > d$ or $I_{w_j} = I_n$, then $S := S \cup \{I_{w_j-1}\}$, $I_{w_j} := I_{w_j-1}$, go to step 4.

   8.2. If $L(I_{w_{j-1}}, I_{w_j}) \leq d$, $I_{w_j} := I_{w_j+1}$, go to step 8.1.

9. If $L(I_{w_{j-1}}, I_{w_j}) > d$, $I_{w_j} := LSUC(I_{w_j}, L(I_{w_{j-1}}, I_{w_j}) - d)$, go to step 8.

For the sake of clarity we illustrate Algorithm 3.1 with an example. We consider a 2-center problem on the interval graph given in Figure 1. We first calculate the dominating path $P(I_1)$ and $C(I_1)$ (as defined in Section 2), $V(P(I_1)) = \{I_1, I_4, I_9, I_{10}\}$, $C(I_1) = \{I_4, I_{10}\}$, $c = \lceil \frac{l(I_1)}{p} \rceil = 4/2 = 2$, and $d = \lceil \frac{c-1}{2} \rceil = 1$. For $j = 1, I_{w_j} := I_4$; For $j = 2, I_{w_j} := I_{10}$, and $L(I_{w_1}, I_{w_2}) = 2 > 1$. In Step 9, $I_{w_2} := LSUC(I_{w_2}, 1) = LSUC(I_{10}, 1) = I_9$. The algorithm ends in Step 5. We obtain an optimal solution $S = \{I_4, I_9\}$.

We next verify the validity of Algorithm 3.1.

Theorem 3.1 *Algorithm 3.1 produces an optimal solution for the p-center problem on interval graphs.*

Proof. Let $S = \{I_{w_1}, I_{w_2}, \ldots, I_{w_p}\}$ be the vertex set produced by Algorithm 3.1. By Theorem 2.1, it is clear that $OPT(\mathcal{F}) = d + 1$ or $OPT(\mathcal{F}) = d$. If $OPT(\mathcal{F}) = d$, we will prove that the algorithm ends in Step 5, and S is an optimal solution for the problem. Let $M = \{I_{l_1}, I_{l_2}, \ldots, I_{l_p}\}$ be the optimal solution for the problem that contains as many of the vertices of S as possible. Let $I_{l_{j+1}}$ be the first vertex of M that is different from S. If $j = 0$, i.e., $I_{l_1} \neq I_{w_1} = I_{i_{d+1}}$. Since $d(I_1, I_{w_1}) = d(I_1, I_{i_{d+1}}) = d$ and $d(I_1, I_{l_1}) \leq d$, it follows from Lemma 2.4 that $l_1 < w_1$. Hence, for any $1 < k < w_1$, we have $d(I_k, I_{w_1}) \leq d(I_1, I_{w_1}) = d$, and for any $k > w_1$, we have $d(I_k, I_{w_1}) \leq d(I_k, I_{l_1})$. Thus, $M' = (M - \{I_{l_1}\}) \cup \{I_{w_1}\}$ is an optimal solution for the problem, and $|M \cap S| < |M' \cap S|$, which contradicts the choice of M. If $j > 0$, then

$l_1 = w_1, l_2 = w_2, \ldots, l_j = w_j$. Since M is an optimal solution, $L(I_{w_j}, I_{l_{j+1}}) = L(I_{l_j}, I_{l_{j+1}}) \leq d$. From Algorithm 3.1, it is valid that $w_{j+1} \geq l_{j+1}$, and $L(I_{w_j}, I_{w_{j+1}}) \leq d$. Then, for any $1 \leq k < w_j = l_j, d(I_k, I_{w_j}) = d(I_k, I_{l_j}) \leq d(I_k, I_{l_{j+1}})$; for any $w_j < k < w_{j+1}, d(I_{w_j}, I_k) \leq d$ or $d(I_k, I_{w_{j+1}}) \leq d$; for any $w_{j+1} < j \leq n$, $d(I_k, I_{w_{j+1}}) \leq d(I_k, I_{l_{j+1}})$. So, $M' = (M - \{I_{l_{j+1}}\}) \cup \{I_{w_{j+1}}\}$ is an optimal solution, and $|M \cap S| < |M' \cap S|$, contradicting the choice of M again. Hence $S = M$. Since M is an optimal solution, $d(I_{max(a)}, I_{w_p}) \leq d$, and so the algorithm ends in Step 5. If $OPT(\mathcal{F}) = d + 1$, then $C(I_1)$ is an optimal solution, and Algorithm 3.1 outputs $S = C(I_1)$, so the result follows. ■

To evaluate the complexity of the algorithm, we first note that Step 1 can be implemented to run in $O(n)$ time by Lemma 2.1. The computation of $q(I_i)$ for each I_i can be performed in $O(n)$ time by Procedure $q(I_1)$. By Lemmas 2.1, 2.2, 2.5, and 2.6, after an $O(n)$ time processing step, $L(I_{w_j}, I_{w_{j+1}})$ can be computed in constant time. Therefore, the marginal effort needed to compute Step 4 to Step 10 is $O(n)$. The total computational effort to solve the problem amounts to $O(n)$. We have thus established the following result.

Theorem 3.2 *Algorithm 3.1 solves the p-center problem on interval graphs with unit lengths in $O(n)$ time if the endpoints of the intervals are sorted.*

4 Concluding remarks

In this paper we presented an $O(n)$ time algorithm to solve the p-center problem on interval graphs with unit lengths under the assumption that the endpoints of the intervals are sorted. The p-center problem on interval graphs with general edge lengths is NP-hard. It remains an interesting question whether we can develop an approximation algorithm for the p-center problem on interval graphs with general edge lengths.

Acknowledgements

The authors would like to thank the referees for their valuable comments and suggestions on an earlier version of this paper.

References

[1] J. Bar-Ilan and D. Peleg, Approximation algorithms for selecting network centers, Proceedings of Workshop on Algorithms and Data Structures '91 1991: 343-354.

[2] S. Bespamyatnikh, B. Bhattacharya, M. Keil, D. Kirkpatrick, M. Segal, Efficient algorithms for centers and medians in interval and circular-arc graphs, Networks 2002; 39: 144-152.

[3] K.S. Booth and G.S. Leuker, Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms, Journal of Computer and System Sciences 1976; 13: 335-379.

[4] M.L. Brandeau, S.S. Chiu, An overview of representative problems in location research, Management Science 1989; 35: 645-674.

[5] D. Chen, D.T. Lee, R. Sridhar, and C. Sekharam, Solving the all-pair shortest path query on interval and circular-arc graphs, Networks 1998; 31: 249-258.

[6] G. Frederickson, Parametric search and locating supply centers in trees, Proceedings of Workshop on Algorithms and Data Structures '91 1991: 299-319.

[7] M.R. Garey and D. Johnson, Computer and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.

[8] S.L. Hakimi, Optimal locations of switching centers and the absolute centers and medians of a graph, Operations Research 1964; 12: 450-459.

[9] M.M. Halldorsson, G. Kortsarz, and H. Shachnai, Sum coloring interval and k-claw free graphs with application to scheduling dependent jobs, Algorithmica 2003; 37: 187-209.

[10] D. Hochbaum and D.B. Shmoys, A unified approach to approximation algorithms for bottle neck problems, Journal of the ACM 1986; 33: 533-550.

[11] V.N. Hsu, T.J. Lowe, and A. Tamir, Structured p-facility location problems on the line solvable in polynomial time, Operations Research Letter 1997; 21: 159-164.

[12] S. Irani and V. Leung, Scheduling with conflicts on bipartite interval graphs, Journal of Scheduling 2003; 6: 287-307.

[13] O. Kariv and S.L. Hakimi, An algorithmic approach to network location problems I: The p-centers, SIAM Journal of Applied Mathematics 1979; 37: 514-538.

[14] C.P. Kruskal, L. Rudolph, and M. Snir, The power of parallel prefix, IEEE Transactions on Computers 1985; C-34: 965-968.

[15] M. Labbe, D. Peeters, and J.F. Thisse, "Location on networks," Handbooks in Operations Research and Management Science, M. Ball, T. Magnanti, R.L. Francis (Editors), Elsevier, Amsterdam, 1995; 8.

[16] R.E. Ladner, M.J. Fischer, Parallel prefix computation, Journal of the ACM 1980; 27: 831-838.

[17] S. Olariu, A simple linear-time algorithm for computing the center of an interval graph, Interational Journal of Computer Mathematics 1990; 24: 121-128.

[18] A. Tamir, Improved complexity bounds for center location problems on networks by using dynamic data structures, SIAM Journal of Discrete Mathematics 1988; 1: 377-396.

[19] B.C. Tansel, R.L. Francis, T.J. Lowe, Location on networks: A survey–Part I: The p-center and p-median problems, Management Science 1983; 29: 482-497.