A Linear Time Algorithm for the Weighted Rectilinear 2–Center Problem

M. T. Ko and Y. T. Ching

Institute of Information Science
Academia Sinica, R. O. C.

Abstract

Given a set of demand points with positive weights respectively on the plane, the weighted rectilinear m–center problem is to find m service points such that the maximum among the weighted distances of demand points to their respective nearest service points is minimized. In this paper, we present a linear time algorithm to solve the weighted rectilinear 2–center problem.

1. Introduction

For two points $p = (p_x, p_y)$ and $q = (q_x, q_y)$ on the plane, the **rectilinear distance** between p and q is $\ell_\infty(p, q) = \max \{|p_x-q_x|, |p_y-q_y|\}$. The rectilinear distance between a point p and a set of points S is $\ell_\infty(p, S) = \min_{q \in S} \{ \ell_\infty(p,q) | q \in S\}$. Formally, given a set P of n demand points $p_1, p_2, \ldots, p_n$ with weights $w_1, w_2, \ldots, w_n$ respectively, the **weighted rectilinear m–center problem** on P is to find a service point set C of cardinality m, such that the cost function of C, $\max_{1 \le i \le n} w_i \cdot \ell_\infty(p_i, C)$ is minimal among all possible sets C. The **weighted distance of p_i to C** is $w_i \cdot \ell_\infty(p_i, C)$. The minimum of the cost function is called the **m–radius of P** and the corresponding service point set C is called an **optimal solution.**

The m–center problem is one of the most important location problems. It is proved that the m–center problems with Euclidean distance and rectilinear distance, where m is arbitrary, are NP–complete [Megiddo and Supowit 1984]. For the case that m is fixed, there are some results. For general m, an $O(n^{m-2} \log n)$ time algorithm for the unweighted rectilinear m–center problem is known [Ko, Lee and Chang 1987]. The weighted rectilinear 1–center problem can be solved in linear time [Megiddo 1983]. The unweighted rectilinear 2–center problem also has a linear time algorithm [Drezner 1987]. The weighted rectilinear 2–center and 3–center problems already have $O(n \log n)$ time algorithms, where n is the number of input points [Ko and Lee 1988].

In this paper, we will present a linear time algorithm for the weighted rectilinear 2–center problem which improves the previous $O(n \log n)$ result.

2. Overview and the Algorithm for the 1–Dimensional Case

The weighted 2–center problem on P is equivalent to finding a partition of P with partition sets S_1 and S_2 such that the maximum of 1–radii of S_1 and S_2 is minimized. The minimized partition is called an **optimal partition**. Let r^* be the 2–radius of P, and $\{c_1, c_2\}$ be an optimal solution. Then $S_1 = \{ p \in P \,|\, \ell_\infty(p, c_1) \le \ell_\infty(p, c_2)\}$ and $S_2 = P \setminus S_1$ form an optimal partition. The above partition is separated by the path of points (the point for the 1–dimensional case) with equal distance to c_1 and c_2. Since there exists an optimal partition separated by a path on the plane, in the following, we consider partitions separated by paths only. According to the relative position of c_1 and c_2, we will use S_{ul}, S_{ur}, S_{ll}, and S_{lr} to denote the partition sets. For example, c_1 is on the upper left side of c_2, then the partition set associate to c_1 is denoted by S_{ul} and that associate to c_2 is denoted by S_{lr}. In the 1–dimensional case, S_{left} and S_{right} are used. In fact, there can be many optimal partitions. Let us see the following example with all points on a line.

A demand point in the 1–dimensional case is called a weighted number. A weighted number p with x–coordinate x and weight w be denoted by (x,w). In the example, we have 10 weighted numbers: $p_1 = (1,6)$, $p_2 = (5,3)$, $p_3 = (10,2)$, $p_4 = (4,2)$, $p_5 = (8,6)$, $p_6 = (3,1)$, $p_7 = (6,9)$, $p_8 = (2,3)$, $p_9 = (7,7)$ and $p_{10} = (9,4)$, as shown in Figure 1.

It is easy to see that $S_1 = \{ p_3, p_5, p_7, p_9, p_{10}\}$, the set of points with x–coordinates ≥ 6 and S_2 the set of the rest points form an optimal partition. The 2–radius of this set, 108/13, is contributed by the 1–radius of S_1. The partition $S_1 \cup \{p_2, p_4, p_6\}$ and $S_2 \setminus \{ p_2, p_4, p_6\}$ is also an optimal partition, since the addition of p_2, p_4, p_6 to S_1 does not affect the 1–radius of S_1.

Basically, our approach is to find the separating path (separating point for the 1–dimensional case) of an optimal partition sets with a binary search. After a search is executed, a larger subsets of S_1 and S_2 are identified. After all, the separating path is found and the whole optimal partition sets S_1 and S_2 are identified. In a search, it computes 1–radii of the two partition sets separated by the current searched path to determine the next direction to search. To achieve a linear time algorithm, in each iteration, we prune a portion of demand points whose weighted distance to the center is shorter than that of some others in the identified subsets of S_1 and S_2, respectively.

The separating path for which we search in our algorithm is defined by a special optimal solution introduced in [Ko and Lee 1988]. To introduce the special optimal solution, some notation is needed. Consider four functions

$$x_{left}(r) = \min_{p \in P} \{p_x + r/w_p\},$$
$$x_{right}(r) = \min_{p \in P} \{p_x - r/w_p\},$$
$$y_{lower}(r) = \min_{p \in P} \{p_y + r/w_p\}, \text{ and}$$
$$y_{upper}(r) = \min_{p \in P} \{p_y - r/w_p\}$$

and four points defined by $x_{left}(r)$, $x_{right}(r)$, $y_{lower}(r)$, and $y_{upper}(r)$,

$$c_{ul}(r) = (x_{left}(r), y_{upper}(r)),$$
$$c_{ur}(r) = (x_{right}(r), y_{upper}(r)),$$
$$c_{ll}(r) = (x_{left}(r), y_{lower}(r)), \text{ and}$$
$$c_{lr}(r) = (x_{right}(r), y_{lower}(r)).$$

The four points c_{ul}, c_{ur}, c_{ll}, and c_{lr} are called the upper–left, upper–right, lower–left and lower–right center points respectively. The special solutions considered are $\{c_{ul}(r), c_{lr}(r)\}$ and $\{c_{ur}(r), c_{ll}(r)\}$ for some r. We call a demand point is covered by a service point c within r if $w_p \cdot \ell_\infty(p, c) \leq r$. To the x–direction and y–direction, we call c covers p within r horizontally (vertically resp.) if $w_p \cdot |p_x - c_x| \leq r$ ($w_p \cdot |p_y - c_y| \leq r$ resp.).

Lemma 1: [Ko and Lee 1988]
Let r^* be the 2–radius of P. A number $r \geq r^*$ if and only if one of the special solutions $\{c_{ul}(r), c_{lr}(r)\}$ and $\{c_{ll}(r), c_{ur}(r)\}$ can cover all points in P within r.

By the Lemma 1, $\{c_{ul}(r^*), c_{lr}(r^*)\}$ or

$\{c_{ll}(r^*), c_{ur}(r^*)\}$ is an optimal solution. The above optimal solution is the special optimal solution used in our algorithm.

For the 2–dimensional case, the prune process is applied to x–direction and y–direction respectively as two 1–dimensional cases. Thus, in the following, we review the prune process in the Megiddo's prune–and–search algorithm [Megiddo 1983] for the 1–dimensional weighted 1–center problem.

Let x_i, $i = 1, 2, \ldots n$, be the given n weighted numbers with weights w_i, $i = 1, 2, \ldots, n$, respectively. The prune process is as follows.

1. Compute x_m, the median of x_i, $i = 1, 2, \ldots n$.
2. Let x^* be the optimal center. Determine whether $x^* \geq x_m$, $x^* \leq x_m$.
3. Without loss of generality, we assume that $x^* \leq x_m$. For the case $x^* \geq x_m$, the process is similar. There are n/2 weighted numbers $\geq x_m$, denoted by $x'_1, \ldots, x'_{\lceil n/2 \rceil}$. Consider pairs (x'_{2i-1}, x'_{2i}), $i = 1, 2, \ldots, \lceil n/4 \rceil$.
4. Compute the solution $x_{2i-1,2i}$ of equation $w'_{2i-1} \cdot (x'_{2i-1} - x) = w'_{2i} \cdot (x'_{2i} - x)$ for $i = 1, 2, \ldots, \lceil n/4 \rceil$.
5. Compute x'_m, the median of x'_i's.
6. Determine whether $x^* \geq x'_m$ or $x^* \leq x'_m$.
7. Without loss of generality, we assume that $x^* \geq x'_m$. For the case $x^* \leq x'_m$, the process is similar.
8. For all the pairs with $x_{2i-1,2i} \geq x'_m$, prune away the weighted point with shorter weighted distance to the x^* in each pair.

It is obvious that in a prune process, it takes O(n) time and prunes $\lfloor n/8 \rfloor$ points away, where n is number of the input demand points.

In the step 8 of the above prune process, since the weighted number of longer weighted distance of each pair is still remained, the 1–radius of the set of weighted number after the prune process is the same as that of the original set.

In the step 2 and step 6 in the above prune process, the side of x^* which x_m and x'_m lie in should be determined. In the 1–center problem, it is to compare the longest weighted distance of x_m (or x'_m resp.) to the weighted numbers on its right side and that on its left side. In the 2–center problem, we will apply the prune process to subsets of partition sets S_{left} and S_{right} with respect to the special optimal solution $\{ x_{left}(r^*), x_{right}(r^*) \}$, where r^* is the 2–radius. For a subset of S_{left} (S_{right} resp.), thus, we should determine whether $x_m \geq x_{left}(r^*)$ ($x_m \leq x_{right}(r^*)$ resp.) or not. Without loss of generality, we consider the case of a subset of S_{left}. It is symmetric for the case of S_{right}. The process is as follows:

```
1. r := max {$w_p \cdot (x_m - p_x) \mid p_x \leq x_m$}
2. Consider special solution { $x_{left}(r)$, $x_{right}(r)$}.
   If {$x_{left}(r)$, $x_{right}(r)$} covers all
      weighted numbers within r
    then $x_{left}(r^*) \leq x_m$
    else $x_{left}(r^*) > x_m$.
```

In the step 1, it determine the r such that $x_m = x_{left}(r)$. In the step 2, the special solution $\{ x_{left}(r), x_{right}(r) \}$ for all the weighted numbers is considered. By Lemma 1, if they cover all weighted numbers within r, then the 2–radius of P, $r^* \leq r$. In other words, the left center

$x_{left}(r^*)$ of the special optimal solution is on the left side of x_m. Otherwise, $x_{left}(r^*)$ is on the right side of x_m.

Now, we present the algorithm for the 1–dimensional weighted 2–center problem.

Algorithm 1–Dimensional Weighted 2–Center:

```
Input: weighted number set P
Output: the 2–radius of P
(* $S_1$ ($S_2$ resp.): the set of weighted numbers already determined to be covered by the left (right resp.) center *)
(* R : the weighted numbers still not determined *)
1. $R := P$, $S_1 := \phi$, and $S_2 := \phi$.
2. case
     $R = \phi$:
       then  Output $r_1$
     $|R| = 1$:
       then  $T_1 := S_1$, $T_2 := S_2 \cup R$,
             compute $r_1'$, $r_2'$ the 1–radius
             of $T_1$ and $T_2$ respectively,
             Output min $\{\max\{r_1, r_2\}, \max\{r_1', r_2'\}\}$
       else  Go to step 3.
3. Compute $x_m$, the median of weighted numbers in R.
4. $R_1 := \{x_i \in R \mid x_i \leq x_m\}$
   $R_2 := \{x_i \in R \mid x_i > x_m\}$.
   $T_1 := S_1 \cup R_1$ and $T_2 := S_2 \cup R_2$.
5. Compute $r_1$, the 1–radius of $T_1$ and $r_2$, the 1–radius of $T_2$.
6. If  $r_1 = r_2$  then output $r_1$.
   If  $r_1 < r_2$  then $S_1:=T_1$, $R:=R \setminus R_1$.
   If  $r_1 > r_2$  then $S_2:=T_2$, $R:=R \setminus R_2$.
7. Apply prune process to $S_1$ and $S_2$.
8. Go to step 2.
```

From step 3 to step 6, the algorithm conduct a binary search to find the separating point. According to the order of 1–radii of T_1 and T_2, we identify larger subsets S_1 and S_2 of S_{left} and S_{right} respectively. In step 7, we apply the prune process to S_1 and S_2 individually.

In each iteration, it takes $O(|R|)$ time to find the median of R, $O(|T_1| + |T_2|)$ time to compute 1–radii of T_1 an d T_2, and $O(|S_1| + |S_2|)$ time away $(|S_1| + |S_2|)/8$ points. Since the factor $O(|T_1| + |T_2|)$ dominates all the others, we calculate this factor in each iteration for the total time complexity of this algorithm. Let $f_k = |T_1| + |T_2|$ in the k–th iteration, that is, the number of weighted numbers remained in the k–th iteration. It is obvious that $f_1 = n$. Since at the k–th iteration, the belonging partition sets of $f_{k-1} - n/2^k$ points are already determined, $(f_{k-1} - n/2^k)/8$ are pruned. Thus, we have the following recursive formula.

$$\begin{aligned} f_k &= f_{k-1} - (f_{k-1} - n/2^k)/8 \\ &= n/2^{k+3} + 7 \cdot f_{k-1}/8. \end{aligned}$$

Let $F_s = \sum_{k=1}^{s} f_k$. Since there are $\log_2$ n iterations, the time complexity of our algorithm is dominated by $O(F_{\log_2 n})$.

$$\begin{aligned} F_s &= \sum_{k=1}^{s} f_k \\ &= (7/8) \sum_{k=1}^{s} f_{k-1} + \sum_{k=1}^{s} n/2^{k+3} \\ &\leq (7/8) \cdot F_{s-1} + n/8 \\ &\leq \dots \\ &\leq (7/8)^{s-1} F_1 + (\sum_{i=0}^{s-2} (7/8)^i) \cdot n/8 \\ &\leq (7/8)^{s-1} \cdot F_1 + n \end{aligned}$$

By the above inequality, and $(7/8)^{\log_2 n-1} \cdot n = (8/7) \cdot n^{\log_2 7-2} < 8n/7$, we have that $F_{\log_2 n} \leq O(n)$. Thus, the time complexity of our algorithm is O(n).

3. An O(n) Algorithm for the 2–dimensional Case

The basic scheme of our algorithm for the 2–dimensional case is the same as that for the 1–dimensional case. But, instead of a separating point, a separating path of an optimal partition is to be searched in the 2–dimensional case. Using the special optimal solution, the separating path is, as shown in Figure 2, made of two semi–lines and a line segment. The separating path is exactly the locus of points of equal rectilinear distance to the two centers in the optimal solution. According to the relative position of the two centers in the optimal solution, the two semi–lines are both of slope 1 or –1, and the segment is parallel to x–axis or y–axis. In fact, we will search the two semi–lines individually and find an optimal partition different from that defined by a special optimal solution on some demand points which will not affect the final solution.

By Lemma 1, the optimal solution may consist of a lower–left center point and an upper–right center point or a lower–right center point and an upper–left center point. We may find the optimal solution among each type of special solutions and take the minimal of the optimal solutions of these two types as the optimal solution. Thus, in the following, without loss of generality, we present the algorithm to find the optimal solution among special solutions consisting of an upper–left center point and a lower–right center point.

Before presenting our idea, a lemma is needed. Let $D_{**}(r)$ be the set of demand points covered by $c_{**}(r)$ within r where $_{**}$ is ul, ur, ll or lr.

Lemma 2: [Ko and Lee 1988]
$D_{**}(r) \subset D_{**}(r')$ if $r \leq r'$.

By Lemma 2, a demand point is covered by $c_{**}(r)$ within r, then it is covered by $c_{**}(r')$ within r' provided that $r' \geq r$. Consider Lemma 2 in 1–dimensional case, it means that a demand point covered by $c_{**}(r)$ within r vertically (horizontally resp.) is covered by $c_{**}(r')$ within r' vertically (horizontally resp.).

For a set of demand points on the plane, S, denote S_x, called **x–version of S**, the set of weighted numbers, $\{(p_x, w_p) \mid p \in S\}$ and S_y, called **y–version of S**, the set of weighted numbers $\{(p_y, w_p) \mid p \in S\}$.

To solve the weighted rectilinear 2–center problem on P, the first step of our algorithm is to solve the weighted 2–center problem on P_x and P_y. Let r_x and r_y be the 2–radii of P_x and P_y respectively and r_{xy} be the maximum of r_x and r_y. With the value r_{xy} and the special service point set $C_{xy} = \{c_{ul}(r_{xy}), c_{lr}(r_{xy})\}$, we partition the demand point set P into four subsets as follows.

$$D'_{ul} = \{ p \mid p \in D_{ul}(r_{xy}) \text{ and } \ell_\infty(p, c_{ul}(r_{xy})) \leq \ell_\infty(p, c_{lr}(r_{xy})) \},$$

$$D'_{lr} = \{ p \mid p \in D_{lr}(r_{xy}) \text{ and } \ell_\infty(p, c_{lr}(r_{xy})) \leq \ell_\infty(p, c_{ul}(r_{xy})) \},$$

$$R_{upper-right} = \{ p \mid w_p \cdot \ell_\infty(p, C_{xy}) > r_{xy}, |p_x - x_{left}(r_{xy})| > r_{xy} \},$$

$$R_{lower-left} = \{ p \mid w_p \cdot \ell_\infty(p, C_{xy}) > r_{xy}, |p_x - x_{right}(r_{xy})| > r_{xy} \},$$

The Figure 3 illustrates the partition conceptually. The union of $R_{upper-right}$ and $R_{lower-left}$ is the set not covered by $c_{ul}(r_{xy})$ and $c_{lr}(r_{xy})$ within r_{xy}. The points in $R_{upper-right}$ ($R_{lower-left}$ resp.) are covered by $c_{lr}(r_{xy})$ horizontally (vertically resp.), and covered by $c_{ul}(r_{xy})$ vertically (horizontally resp.) within r_{xy}, but not covered by $c_{lr}(r_{xy})$ vertically (horizontally resp.), and not covered by $c_{ul}(r_{xy})$ horizontally (vertically resp.) within r_{xy}.

Observation 1: The 2–radius of P, $r^* \geq r_{xy}$.

Without loss of generality, assume that $r_{xy} = r_x$. Since r_x is the 2–radius of P_x,

$$\begin{aligned} r_x &\leq \max_{p \in P} \min \{w_p |p_x - x_{left}(r^*)|, \\ &\qquad w_p |p_x - x_{right}(r^*)|\} \\ &\leq \max_{p \in P} \min \{w_p \cdot \ell_\infty(p, c_{ul}(r^*)), \\ &\qquad w_p \cdot \ell_\infty(p, c_{lr}(r^*))\} \\ &= r^*. \end{aligned}$$

Thus, $r^* \geq r_x = r_{xy}$.

By the above observation, if $D'_{ul} \cup D'_{lr} = P$ then the 2–radius of P is exactly r_{xy}. Thus, in the following, we consider the case that $D'_{ul} \cup D'_{lr} \neq P$ only.

Observation 2: If $D'_{ul} \cup D'_{lr} \neq P$, then $c_{ul}(r^*)_x \leq c_{lr}(r^*)_x$ and $c_{ul}(r^*)_y \geq c_{lr}(r^*)_y$.

If $c_{ul}(r^*)_x > c_{lr}(r^*)_x$, there is an r such that $r_{xy} \leq r < r^*$ and $c_{ul}(r)_x = c_{lr}(r)_x$. Thus, $c_{ul}(r)$ ($c_{lr}(r)$ resp.) can cover all the demand points within r horizontally by itself. Since $r \geq r_{xy}$, $\{c_{ul}(r), c_{lr}(r)\}$ covers all demand points within r. It contradicts r^* is the 2–radius of P. The proof of $c_{ul}(r^*)_y \geq c_{lr}(r^*)_y$ is similar.

By this observation, the separating path of the optimal partition is made of two semi lines of slope 1 and a line segment parallel to x–axis or y–axis, as shown in Figure 2a or 2b. Thus, only separating lines of slope 1 are considered in case of finding an optimal solution among solutions consisting of an upper–left center point and a lower–right center point.

Now, given a line L of slope 1, $R_{upper-right}$ is partitioned into two subsets. One is the set of points above the line L, denoted as $R_{L,upper}$ and the other is that below the line L, denoted as $R_{L,right}$. To such a line L, we consider two sets $T_{1,L} = R_{lower-left} \cup D'_{ul} \cup R_{L,upper}$, $T_{2,L} = R_{lower-left} \cup D'_{lr} \cup R_{L,right}$ and the weighted 1–center problems on $(T_{1,L})_x$ and $(T_{2,L})_y$. There is a line $L_{upper-right}$ such that the maximum of the 1–radii of above two sets of weighted numbers is minimized. Denote the minimized maximum as r_1. Similarly, for the set $R_{lower-left}$, there is a line L of slope 1 partitions $R_{lower-left}$ into $R_{L,left}$ and $R_{L,lower}$ such that the maximum of the 1–radii of the x–version of $R_{upper-right} \cup D'_{lr} \cup R_{L,lower}$ and the y–version of $R_{upper-right} \cup D'_{ul} \cup R_{L,left}$ is minimized. The minimized maximum for the $R_{lower-left}$ is denoted as r_2.

We claim the following lemma for the 2–radius of P.

Lemma 3:
The maximum of r_1 and r_2 is the 2–radius of P.
Proof:

Let r' denote the maximum of r_1 and r_2, and r^* the 2–radius of P. We will first prove that $r' \geq r^*$ and then prove that $r' \leq r^*$.

To prove $r' \geq r^*$, we will show that all demand points are covered by $C' = \{c_{ul}(r'), c_{lr}(r')\}$ within r'. By Lemma 2, all points in $D'_{ul} \cup D'_{lr}$ are covered by C' within r', since $r' \geq r_{xy}$. Let p be any point not in $D'_{ul} \cup D'_{lr}$. We consider the case that $p \in R_{upper-right}$, and p is on the upper side of the separating line $L_{upper-right}$. Since $p \in R_{upper-right}$ and $r' \geq r_{xy}$, $c_{ul}(r')$ covers p within r' vertically. Since p is on the upper side of line $L_{upper-right}$ and $r' \geq r_1$, $c_{ul}(r')$ covers p within r' horizontally. Thus, $w_p \cdot \ell_\infty(p, C') \leq r'$. By the same argument, we also can prove that $w_p \cdot \ell_\infty(p, C') \leq r'$ for the demand point p in the other cases. Thus, we conclude that $r' \geq r^*$.

To prove that $r' \leq r^*$, without loss of generality, we assume $r' = r_1$. Consider the partition of $R_{upper-right}$ with separating line L* which is the locus of points p satisfying $|p_x - c_{ul}(r^*)_x| = |p_y - c_{lr}(r^*)_y|$. We will prove that the 1–radii of $(T_{1,L^*})_x$ and $(T_{2,L^*})_y$ is less than or equal to r^*. Since r_1 is the minimum among all partitions of $R_{upper-right}$, $r^* \geq r_1 = r'$.

Let p be any point in T_{1,L^*}. If p is not in $R_{upper-right}$, it is obvious that $w_p \cdot |p_x - c_{ul}(r^*)_x| \leq r^*$. Consider that $p \in R_{upper-right}$. Since p is above L*, $|p_x - c_{ul}(r^*)_x| \leq |p_y - c_{lr}(r^*)_y|$. Let $C^* = \{c_{ul}(r^*), c_{lr}(r^*)\}$. In case that $w_p \cdot \ell_\infty(p, C^*) = w_p \cdot \ell_\infty(p, c_{ul}(r^*))$, $w_p \cdot |p_x - c_{ul}(r^*)_x| \leq w_p \cdot \ell_\infty(p, c_{ul}(r^*)) \leq r^*$. For the case that $w_p \cdot \ell_\infty(p, C^*) = w_p \cdot \ell_\infty(p, c_{lr}(r^*))$,

$$\begin{aligned} & w_p \cdot |p_x - c_{ul}(r^*)_x| \\ \leq\ & w_p \cdot |p_y - c_{lr}(r^*)_y| \\ \leq\ & w_p \cdot \ell_\infty(p, c_{lr}(r^*)) \\ \leq\ & r^*. \end{aligned}$$

Thus, we conclude the 1–radius of $(T_{1,L^*})_x \leq r^*$. For any point p in T_{2,L^*}, we also conclude $w_p \cdot |p_y - c_{lr}(r^*)_y| \leq r^*$ similarly. Thus, the 1–radius of $(T_{2,L})_y \leq r^*$. Since r' is the minimum among that of partitions, we have $r^* \geq r'$. Q.E.D.

With Lemma 3, the problem now is to search the optimal separating lines for $R_{upper-right}$ and $R_{lower-left}$ efficiently. We will show the algorithm for $R_{upper-right}$ only. It is similar for $R_{lower-left}$.

The scheme of the algorithm is essentially the same as that for the 1–dimensional 2–center problem. We conduct a binary search for the position of separating line of the minimal partition. In each iteration, we compute m, the median of $(p_x - p_y)$'s where $p \in R_{upper-right}$ and is still not determined. Let L be the line satisfying $x - y = m$. Next, we compute the 1–radii of $(T_{1,L})_x$ and $(T_{2,L})_y$ to determine the next direction to search and make sure the belonging set of one half of undetermined points. To the points already known in $T_{1,L_{upper-right}}$ and in $T_{2,L_{upper-right}}$, we conduct the prune process to them according to their x–version and y–version respectively. It is obvious that the complexity is linear, the same as that for the 1–dimensional case.

4. Concluding Remarks

In this paper, we showed an optimal linear time algorithm for the weighted rectilinear 2–center problem on the plane.

The algorithm can be generalized to the higher dimensional case with linear–time complexity. There are 2^{d-1} types of special solutions. As the 2–dimensional case, we may find the optimal solution among each type of special solutions and the minimal one among that of all the types is the optimal solution. For a pair of coordinates x_i and x_j, projecting the demand points to the x_i–x_j plane, we have an, so called, i–j version of the demand point set. To a type of special solutions, we consider the corresponding special solutions on all C_2^d versions. Using the algorithm for the 2–dimensional case, we may find the optimal corresponding special solution for each version. The maximum of the optimal special solutions of these C_2^d versions is the optimal solution among the type of special solutions. Therefore, an $O(2^{d-1} \cdot C_2^d \cdot n)$ algorithm is obtained for the d–dimensional weighted rectilinear 2–center problem.

References:

[1] Drezner, Z. (1987) "On the Rectangular p–Center Problem", Naval Research Logistics, vol. 34, pp. 229–234.

[2] Ko, M. T. and Lee, R. C. T. (1988), "On Weighted Rectilinear 2–Center and 3–Center Problems," to appear in Information Sciences.

[3] Ko, M. T., Lee, R. C. T., Chang, J. S. (1987), "Rectilinear m–Center Problem," Proceedings of National Computer Symposium, Taipei, R.O.C..

[4] Megiddo, N. (1983) "Linear–time Algorithms for Linear Programming in $\mathbb{R}^3$ and Related Problems," SIAM Journal on Computing, vol. 12, no. 4, pp. 759–776.

[5] Megiddo, N. and Supowit, K. J. (1984) "On the Complexity of Some Common Geometric Location Problems," SIAM Journal on Computing, vol. 13, no. 1, pp. 182–196.

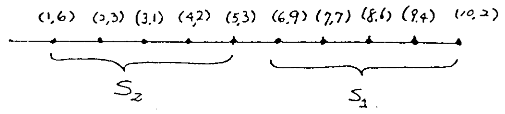

Figure 1.

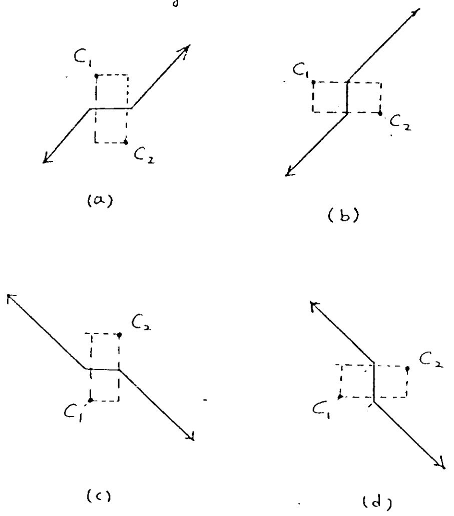

Figure 2

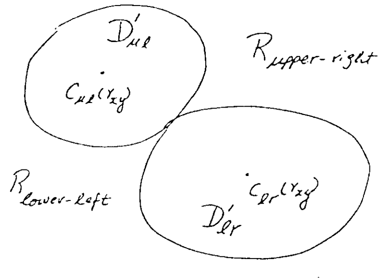

Figure 3