TR-90-007

Linear Time Algorithms for Weighted Tailored 2–Partition Problem and Weighted 2–Center Problem Under ℓ_∞–Distance

TR-90-007

Linear Time Algorithms for Weighted Tailored 2–Partition Problem and Weighted 2–Center Problem Under ℓ_∞–Distance

M. T. Ko and Y. T. Ching

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

Corresponding Address:
M. T. Ko
Institute of Information Science
Academia Sinica
128, Yen–Chiu–Yuan Road, Sec. II
Taipei, Taiwan 11529, R. O. C.

Abstract

In this paper, weighted tailored 2–partition problem and weighted 2–center problem under ℓ_∞–distance are considered. An $O(2^{d-1} \cdot d \cdot n)$ algorithm to solve the weighted tailored 2–partition problem and an $(O(\binom{d}{2} \cdot n + d^2 \cdot \log d)$ time algorithm to solve the weighted 2–center problem in d–dimensional case are presented.

Section 1. Introduction

In the **weighted 2–center problem** and the **weighted tailored 2–partition problem** under ℓ_∞–distance, a **demand point set** P with weights is given, we are asked to find two **service points** satisfying certain criterions depending on the ℓ_∞–distance. In this paper, the given demand point set is denoted as $P = \{ p_1, p_2, ..., p_n \}$ where p_i is with weight w_i respectively. If p_i's are in d–dimensional space, p_i's coordinate are denoted as $(x_i^1, x_i^2, ..., x_i^d)$, $i = 1, ..., n$. Specially, in the 2–dimensional space, they are denoted as (x_i, y_i), $i = 1, 2, ..., n$, and in the 1–dimensional case, we use p_i as the coordinate of demand point p_i without ambiguity. The ℓ_∞–distance between two points $p = (x^1, x^2, ..., x^d)$ and $q = (y^1, y^2, ..., y^d)$, $\ell_\infty(p, q)$ is $\max \{ |x^i - y^i| \mid i = 1, 2, ..., d \}$. Let $C = \{ c_1, ..., c_m \}$ be a set of service points. Let the ℓ_∞–distance of a point p to the set C, denoted as $\ell_\infty(p, C)$, be $\min \{ \ell_\infty(p, c_i) \mid i = 1, 2, ..., m \}$. The **weighted ℓ_∞–distance** of a demand point p_i to a service point c is $w_i \cdot \ell_\infty(p_i, c)$. For a service point c and a number r, define

$$D(c, r) = \{ p_i \in P \mid w_i \cdot \ell_\infty(p_i, c) \le r \},$$

the set of demand points serviced by service point c within weighted ℓ_∞–distance r.

The **m–center problem** is to find the minimum r such that there exists m service points $c_1, c_2, ..., c_m$ with $\bigcup_{j=1}^{m} D(c_j, r) = P$. The minimum r, denoted by r^* throughout the paper, is called the **m–radius** of the demand point set. A set of m service points achieves the minimum r is called an **optimal solution**. Sometime we also call r as the optimal solution without ambiguity. The **tailored m–partition problem**, quite similar to the m–center problem, is for the given m positive real numbers $r_1, r_2, ..., r_m$, determine whether there exists service points $c_1, c_2, ..., c_m$ such that $\bigcup_{j=1}^{m} D(c_j, r_j) = P$. **In this paper, we consider the problem for m = 2.**

The m–center problem is one of the most important location problems. There are many literatures consider the m–center problem on planar demand point set under ℓ_p–distance. Especially, the problem under Euclidean distance (ℓ_2–distance) and rectilinear distance (ℓ_1–distance) accepts much attention. Since the rectilinear distance is equivalent to the ℓ_∞–distance in 2–dimensional space under a linear transformation, the rectilinear m–center problem is equivalent to the m–center problem under ℓ_∞–distance in the 2–dimensional case. It is proved that the m–center problems under 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 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]. The tailored 2–partition problem has an $O(n \cdot \log n)$ algorithm for the 2–dimensional case [Hershberger and Suri 1989].

In this paper, we will present an $O(\binom{d}{2} \cdot n + d^2 \log d)$ time algorithm for the d–dimensional weighted 2–center problem under ℓ_∞–distance. The algorithm improves the previous $O(n \log n)$ result in the 2–dimensional case. Also, an $O(2^d \cdot d \cdot n)$ algorithm for the d–dimensional weighted tailored 2–partition problem is given. The result improves the $O(n \log n)$ algorithm for the 2–dimensional case in [Hershberger and Suri 1989].

In the following, we will give two basic lemmas and an $O(2^d \cdot d \cdot n)$ algorithm for the weighted tailored 2–partition problem in section 2, an $O(n)$ algorithm for the 1–dimensional weighted 2–center problem in section 3, an $O(n)$ algorithm for

the 2–dimensional case in section 4, and then with the linear time algorithm for the 2–dimensional case, we design an $O(\binom{d}{2} \cdot n + d^2 \cdot \log d)$ algorithm for the d–dimensional case in section 5. In section 6, we give the concluding remarks.

Section 2. Basic Lemmas and the Algorithm for Tailored 2–Partition Problem

In this section, we first introduce special service points which used in our algorithms. Then, we prove two lemmas on the properties of the special service points and present an algorithm for the tailored 2–partition problem.

To introduce the special service points, for the demand point set P, we define the following functions of weighted distance r.

$$c_s^k(r) = \min \{ x_i^k + r/w_i \mid 1 \le i \le n \}, k = 1, 2, ..., d,$$

$$c_l^k(r) = \max \{ x_i^k - r/w_i \mid 1 \le i \le n \}, k = 1, 2, ..., d,$$

where the indices s and l are the abbreviation of 'small' and 'large' respectively. The demand point p_i that $x_i^k + r/w_i = c_s^k(r)$ (that $x_i^k - r/w_i = c_l^k(r)$ resp.) is called the **small (large resp.) extreme demand point** in the k–th coordinate with respect to r. The function $c_s^k(r)$ ($c_l^k(r)$ resp.) then means the border line to service the small (large resp.) demand point in the k–th coordinate within r.

By $c_s^k(r)$ and $c_l^k(r)$, k = 1, ..., d, we define **center point** $c_{\epsilon_1\epsilon_2...\epsilon_d}(r)$ by

$$c_{\epsilon_1\epsilon_2...\epsilon_d}(r) = (c_{\epsilon_1}^k(r), c_{\epsilon_2}^k(r), ..., c_{\epsilon_d}^k(r)), \text{ where } \epsilon_i \text{ is } s \text{ or } l.$$

An illustration of functions $c_s^k(r)$, $c_l^k(r)$ and center points in 2–dimensional case is shown in Figure 1.

[Figure 1]

A sequence of 's' and 'l' is called a **side index**. For simplicity of notation, we denote a side index by E. The center points $c_E(r)$ for arbitrary side index E and real number r are the special service points considered in our algorithms for both the 2–center problem and the tailored 2–partition problem. The **dual side index** of $E = \epsilon_1\epsilon_2...\epsilon_d$ is the side index $\bar{E} = \bar{\epsilon}_1\bar{\epsilon}_2...\bar{\epsilon}_d$, where

$$\bar{\epsilon}_i = \begin{cases} s \text{ if } \epsilon_i = l \\ l \text{ if } \epsilon_i = s \end{cases}.$$

The center points $c_E(r)$ and $c_{-E}(r)$ are the **dual center point** to each other. $C_E(r)$ $= \{c_E(r), c_{-E}(r)\}$ is a special solution considered for the weighted 2–center problem.

In the 2–dimensional case, we use $c_\epsilon^x(r)$ and $c_\epsilon^y(r)$ to denote $c_\epsilon^1(r)$ and $c_\epsilon^2(r)$ respectively.

The following two lemmas show the properties of center points in the two considered problem. Lemma 1 is for the 2–center problem. A version of lemma 1 in the 2–dimensional case was proved in [Ko and Lee 1988].

Lemma 1:

Let r^* be the 2–radius of P. Then a number $r \geq r^*$ if and only if there is a side index E such that $D(c_E(r), r) \cup D(c_{-E}(r), r) = P$.

Proof:

(if part) It is trivial by the definition of r^*.

(only if part)

Since $r \geq r^*$, there is two service points c_1 and c_2 such that $D(c_1, r) \cup D(c_2, r)$ $= P$. We will prove the lemma by proving a stronger statement:

If $D(c_1, r) \cup D(c_2, r) = P$ then there is a side index E such that the center points $c_E(r)$ and $c_{-E}(r)$ satisfying $D(c_E(r), r) \supseteq D(c_1, r)$ and $D(c_{-E}(r), r) \supseteq D(c_2, r)$.

In particular, $D(c_E(r), r) \cup D(c_{-E}(r), r) \supseteq D(c_1, r) \cup D(c_2, r) = P$.

Let us show the statement in the 1–dimensional case first. Consider the case that $c_1 \leq c_2$. In this case, the service point c_1 (c_2 resp.) services the small (large resp.) extreme demand point within r. By definition of $c_s(r)$ and $c_l(r)$, we have that $c_1 \leq c_s(r)$ and $c_2 \geq c_l(r)$. All the demand points on the the left side of $c_s(r)$ are in $D(c_s(r), r)$. For the demand point $p_i \in D(c_1, r)$ and on the right side of $c_s(r)$,

since $w_i \cdot (p_i - c_s(r)) \le w_i \cdot (p_i - c_1) \le r$, we have that $p_i \in D(c_s(r), r)$. Thus, we have $D(c_s(r), r) \supseteq D(c_1, r)$. By a similar argument, we have $D(c_f(r), r) \supseteq D(c_2, r)$. In the case that $c_1 \ge c_2$, we have $D(c_f(r), r) \supseteq D(c_1, r)$ and $D(c_s(r), r) \supseteq D(c_2, r)$.

Now, consider the d–dimensional case. Let $c_j = (c_j^1, c_j^2, ..., c_j^d)$, $j = 1, 2$, $P^k = \{ x_i^k \mid i = 1, 2, ..., n\}$, the projection of P in the k–th coordinate and $D^k(c_j^k, r) = \{ x_i^k \mid w_i \cdot |x_i^k - c_j^k| \le r \}$, $k = 1, 2, ..., d$ and $j = 1, 2$.

It is obvious that the projection of $D(c_j, r)$ in the k–th coordinate is contained in $D^k(c_j^k, r)$, for all $k = 1, 2, ..., d$ and $j = 1, 2$. Since that $D(c_1, r) \cup D(c_2, r) = P$, we have $D^k(c_1^k, r) \cup D^k(c_2^k, r) = P^k$, for all $k = 1, 2, ..., d$. By the statement for the 1–dimensional case already proved, there is a side symbol ϵ_k such that $D^k(c^k_{\epsilon_k}(r), r) \supseteq D^k(c_1^k(r), r)$ and $D^k(c_{-\epsilon_k}^k(r), r) \supseteq D^k(c_2^k(r), r)$ for all $k = 1, 2, ..., d$. Let side index $E = \epsilon_1 \epsilon_2 ... \epsilon_d$. We claim that $D(c_E(r), r) \supseteq D(c_1, r)$ and $D(c_{-E}(r), r) \supseteq D(c_2, r)$. Let p_i be any demand point in $D(c_1, r)$. Since $x_i^k \in D^k(c_1^k, r) \subseteq D^k(c^k_{\epsilon_k}(r), r)$, $w_i \cdot |x_i^k - c^k_{\epsilon_k}(r)| \le r$ for all $k = 1, 2, ...d$. Thus,

$$w_i \cdot \ell_\infty(p_i, c_E(r)) = \max \{ w_i \cdot |x_i^k - c^k_{\epsilon_k}(r)| \mid k = 1, 2, ..., d \} \le r.$$

We conclude that $D(c_E(r), r) \supseteq D(c_1, r)$. By a similar argument, we can conclude that $D(c_{-E}(r), r) \supseteq D(c_2, r)$. Thus, the proof is completed.

Q.E.D.

By Lemma 1, there is an optimal solution, referred as **special optimal solution,** consists of $c_E(r^*)$ and $c_{-E}(r^*)$ for some side index E. The special optimal solution is the optimal solution to be found by our algorithm.

Lemma 2 is for the weighted tailored 2–partition problem.

Lemma 2:

For numbers r_1 and r_2, where $r_1 \geq r_2$, if there exists service points c_1 and c_2 such that $D(c_1, r_1) \cup D(c_2, r_2) = P$, then there exists a side index E such that $D(c_E(r_1), r_1) \cup D(c_2, r_2) = P$.

Proof:

Since $D(c_1, r_1) \cup D(c_2, r_2) = P$ and $r_1 \geq r_2$, $D(c_1, r_1) \cup D(c_2, r_1) = P$. By the proved stronger statement in Lemma 1, there exists a side index E such that $D(c_E(r_1), r_1) \supseteq D(c_1, r_1)$. Thus, $D(c_E(r_1), r_1) \cup D(c_2, r_2) = P$.

Q.E.D.

By Lemma 2, for a weighted tailored 2–partition problem on P with numbers $r_1 \geq r_2$, if the answer is yes, we may have a solution with $c_E(r_1)$, for some side index E, as one of the two service points. Exploiting this lemma, we have the following algorithm to solve the weighted tailored 2–partition problem. The algorithm exhaustively check all the side indices. To each side index E, the algorithm removes from P the demand points serviced by $c_E(r_1)$ within r_1 and checks whether all the remaining demand points can be serviced within r_2 by a service point.

ALGORITHM TAILORED 2_PARTITION

INPUT: P, demand point set; $r_1 \geq r_2$, positive numbers

OUTPUT: YES or NO

FOR all E side index DO

    P' := P \ $D(c_E(r_1), r_1)$

    r' := the 1–radius of P'

    IF r' $\leq r_2$ THEN RETURN 'YES' and STOP

ENDFOR

RETURN 'NO'.

For each side index, it takes $O(d \cdot n)$ time to solve the weighted 1–center problem of P'. Thus, the complexity of the algorithm is $O(2^d \cdot d \cdot n)$.

Section 3. The Algorithm for the 1–Dimensional 2–Center Problem

In this section, we will first describe the idea of our algorithm for the weighted 2–center problem in the 1–dimensional and 2–dimensional cases. In this section, all points mentioned are in 1–dimensional or 2–dimensional space.

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 $S_1 = \{ p_i \mid \ell_\infty(p_i, c_1) \leq \ell_\infty(p_i, c_2) \}$ 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 ℓ_∞–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_{ls}, S_{ll}, S_{ss}, and S_{sl} to denote the partition sets. For example, $c_1^1 \leq c_2^1$ and $c_1^2 \geq c_2^2$ then the partition set associate to c_1 is denoted by S_{sl} and that associate to c_2 is denoted by S_{ls}. In the 1–dimensional case, S_l and S_s are used. In fact, there can be many optimal partitions. Let us see the following example with all demand 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 2.

[Figure 2]

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. In a identified subset of S_1 or S_2, the pruned demand points are those of shorter weighted ℓ_∞–distance to the corresponding service point in the optimal solution than that of some others in the same identified subset. The optimal solution we use is the special optimal solution consists of center points introduced in section 2.

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 p_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 $\{ p_i \mid i = 1, 2, \ldots n \}$
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 $p'_1, \ldots, p'_{\lceil n/2 \rceil}$. Consider pairs (p'_{2i-1}, p'_{2i}), $i = 1, 2, \ldots, \lceil n/4 \rceil$.

4. Compute the solution $x_{2i-1\ 2i}$ of equation

   $w'_{2i-1} \cdot (p'_{2i-1} - x) = w'_{2i} \cdot (p'_{2i} - x)$ for $i = 1, 2, \ldots, \lceil n/4 \rceil$.

5. Compute x'_m, the median of $\{ x_{2i-1\ 2i} \mid i = 1, 2, \ldots, \lceil n/4 \rceil \}$.

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 to the optimal center 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_s and S_l with respect to the special optimal solution $\{ c_s(r^*), c_l(r^*) \}$, where r^* is the 2–radius. For a subset of S_s (S_l resp.), thus, we should determine whether $x_m \geq c_s(r^*)$ ($x_m \leq c_l(r^*)$ resp.) or not. Without

loss of generality, we consider the case of a subset of S_s. It is symmetric for the case of S_l. The process is as follows:

1. $r := \max \{ w_i \cdot (x_m - p_i) \mid p_i \le x_m \}$
2. Consider special solution $\{ c_s(r), c_l(r) \}$.

   IF $\{c_s(r), c_l(r)\}$ services all weighted numbers within r

   THEN $c_s(r^*) \le x_m$

   ELSE $c_s(r^*) > x_m$.

In the step 1, it determine the r such that $x_m = c_s(r)$. In the step 2, the special solution $\{ c_s(r), c_l(r) \}$ for all the weighted numbers is considered. By Lemma 1, if they service all weighted numbers within r, then the 2–radius of P, $r^* \le r$. In other words, the left center $c_s(r^*)$ of the special optimal solution is on the left side of x_m. Otherwise, $c_s(r^*)$ 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: P, weighted number set

Output: the 2–radius of P

(* S_1 (S_2 resp.): the set of weighted numbers already determined to be serviced by $c_s(r^*)$ ($c_l(r^*)$ resp.) *)

(* R : the weighted numbers still not determined *)

1. $R := P$, $S_1 := \phi$, and $S_2 := \phi$.

2. CASE

   $|R| = 0$: Output r_1

   $|R| = 1$: $T_1 := S_1$, $T_2 := S_2 \cup R$,

   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 GOTO 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. GOTO 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_s and S_l 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 and T_2, and $O(|S_1| + |S_2|)$ time to prune 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 $O(F_{\log_2 n})$.

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

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 $F_{\log_2 n} \le O(n)$. Thus, the time complexity of our algorithm is $O(n)$.

Section 4. The Algorithm for the 2–Dimensional Weighted 2–Center Problem

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 3, made of two semi–lines and a line segment. The separating path is exactly the locus of points of equal ℓ_∞–distance to the two service points 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 the case that $c_1^1 \leq c_2^1$ and $c_1^2 \leq c_2^2$, the semi–lines are of slope –1 as shown in Figure 3c and 3d. In the case that $c_1^1 \leq c_2^1$ and $c_1^2 \geq c_2^2$, the semi–lines are of slope 1 as shown in Figure 3a and 3b. In the case that $|c_1^1 - c_2^1| > |c_1^2 - c_2^2|$, the line segment is vertical as shown in Figure 3b and 3d. For the other case, the line segment is horizontal as shown in Figure 3a and 3c. 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.

[Figure 3]

By Lemma 1, the special optimal solution can be two types. One consists of center points with side indices *ss* and *ll*, the other consists of center points with side indices *sl* and *ls*. 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 center points with side indices *sl* and *ls*.**

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

Lemma 3: [Ko and Lee 1988]

$D_{**}(r) \subset D_{**}(r')$ if $r \leq r'$.

By Lemma 3, a demand point is serviced by $c_{**}(r)$ within r, then it is serviced by $c_{**}(r')$ within r' provided that $r' \geq r$. Consider Lemma 3 in 1–dimensional case, it means that a demand point serviced by $c_{**}(r)$ within r in y–direction (in x–direction resp.) is serviced by $c_{**}(r')$ within r' in y–direction (in x–direction resp.).

For a set of demand points on the plane, S, denote S_x, called **x–version of S**, the projection of S on the x–coordinate and S_y, called **y–version of S**, the projection of S on the y–coordinate.

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 $C_{xy} = \{c_{sl}(r_{xy}), c_{ls}(r_{xy})\}$, we partition the demand point set P into four subsets as follows.

$$D'_{sl} = \{ p_i \mid p_i \in D_{sl}(r_{xy}) \text{ and } \ell_\infty(p_i, c_{sl}(r_{xy})) \leq \ell_\infty(p_i, c_{ls}(r_{xy})) \},$$

$$D'_{ls} = \{ p_i \mid p_i \in D_{ls}(r_{xy}) \text{ and } \ell_\infty(p_i, c_{ls}(r_{xy})) \leq \ell_\infty(p_i, c_{sl}(r_{xy})) \},$$

$$R_{ll} = \{ p_i \mid w_i \cdot \ell_\infty(p_i, C_{xy}) > r_{xy}, w_i \cdot |x_i - c^x_s(r_{xy})| > r_{xy} \},$$

$$R_{ss} = \{ p_i \mid w_i \cdot \ell_\infty(p_i, C_{xy}) > r_{xy}, w_i \cdot |x_i - c^x_l(r_{xy})| > r_{xy} \},$$

[Figure 4]

The Figure 4 illustrates the partition conceptually. The union of R_{ll} and R_{ss} is the set not serviced by $c_{sl}(r_{xy})$ and $c_{ls}(r_{xy})$ within r_{xy}. The points in R_{ll} (R_{ss} resp.) are

serviced by $c_{ls}(r_{xy})$ in x–direction (in y–direction resp.), and serviced by $c_{sl}(r_{xy})$ in y–direction (in x–direction resp.) within r_{xy}, but not serviced by $c_{ls}(r_{xy})$ in y–direction (in x–direction resp.), and not serviced by $c_{sl}(r_{xy})$ in x–direction (in y–direction resp.) within r_{xy}.

Lemma 4: The 2–radius of P, $r^* \geq r_{xy}$.

Proof:

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

$$\begin{aligned} r_x &\leq \max\{\min\{w_i \cdot |x_i - c_s^x(r^*)|, w_i \cdot |x_i - c_l^x(r^*)|\} \mid 1 \leq i \leq n\} \\ &\leq \max\{\min\{w_i \cdot \ell_\infty(p_i, c_{sl}(r^*)), w_i \cdot \ell_\infty(p_i, c_{ls}(r^*))\} \\ &= r^*. \end{aligned}$$

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

By Lemma 4, if $D'_{sl} \cup D'_{ls} = P$ then the 2–radius of P is exactly r_{xy}. Thus, in the following, we consider the case that $D'_{sl} \cup D'_{ls} \neq P$ only.

Lemma 5: If $D'_{sl} \cup D'_{ls} \neq P$, then $c_s^x(r^*) \leq c_l^x(r^*)$ and $c_l^y(r^*) \geq c_s^y(r^*)$.

Proof:

If $c_s^x(r^*) > c_l^x(r^*)$, there is an r such that $r_{xy} \leq r < r^*$ and $c_s^x(r) = c_l^x(r)$. Thus, $c_{sl}(r)$ ($c_{ls}(r)$ resp.) can service all the demand points within r in x–direction by itself. Since $r \geq r_{xy}$, $\{c_{sl}(r), c_{ls}(r)\}$ services all demand points within r. It contradicts r^* is the 2–radius of P. The proof of $c_l^y(r^*) \geq c_s^y(r^*)$ is similar.

Q.E.D.

By Lemma 5, 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 special solutions consisting of center points of side indices *sl* and *ls*.

Now, given a line L of slope 1, R_{ll} 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 $T_{1,L} = R_{ss} \cup D'_{sl} \cup R_{L,upper}$, $T_{2,L} = R_{ss} \cup D'_{ls} \cup R_{L,right}$ and the weighted 1–center problems on $(T_{1,L})_x$ and $(T_{2,L})_y$. There is a line L_{ll} 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_{ss}, there is a line L of slope 1 partitions R_{ss} into $R_{L,left}$ and $R_{L,lower}$ such that the maximum of the 1–radii of the x–version of $R_{ll} \cup D'_{ls} \cup R_{L,lower}$ and the y–version of $R_{ll} \cup D'_{sl} \cup R_{L,left}$ is minimized. The minimized maximum for the R_{ss} is denoted as r_2.

We claim the following lemma for the optimal solution of P among special solutions with side indices *sl* and *ls*. Denote the above optimal solution as r".

Lemma 6:

The maximum of r_1 and r_2 is the optimal solution among the type of special solutions with side indices *sl* and *ls*.

Proof:

Let r' denote the maximum of r_1 and r_2. 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 serviced by $C' = \{c_{sl}(r'), c_{ls}(r')\}$ within r'. By Lemma 3, all demand points in $D'_{sl} \cup D'_{ls}$ are serviced by C' within r', since $r' \geq r_{xy}$. Let p_i be any point not in $D'_{sl} \cup D'_{ls}$. We consider the case that $p_i \in R_{ll}$ and p_i is on the upper side of the separating line L_{ll}. Since $p_i \in$

R_{ll} and $r' \geq r_{xy}$, $c_{sl}(r')$ services p_i within r' in y–direction. Since p_i is on the upper side of line L_{ll} and $r' \geq r_1$, $c_{sl}(r')$ services p_i within r' in x–direction. Thus, $w_i \cdot \ell_\infty(p_i, C') \leq r'$. By the same argument, we also can prove that $w_i \cdot \ell_\infty(p_i, C') \leq r'$ for the demand point p_i 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_{ll} with separating line L^* which is the locus of points $p = (x, y)$ satisfying $x - c_s^x(r'') = y - c_s^y(r'')$. 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_{ll}, $r'' \geq r_1 = r'$.

Let p_i be any point in T_{1,L^*}. If p_i is not in R_{ll}, it is obvious that $w_i \cdot |x_i - c_s^x(r'')| \leq r''$. Consider that $p_i \in R_{ll}$. If $x_i - c_s^x(r'') \leq 0$, then $x_i - c_s^x(r'') \leq r''$. Thus, we should consider the case that $x_i - c_s^x(r'') > 0$. Since p_i is above L^*, $x_i - c_s^x(r'') \leq y_i - c_s^y(r'')$. Let $C^* = \{c_{sl}(r''), c_{ls}(r'')\}$. In case that $w_i \cdot \ell_\infty(p_i, C^*) = w_i \cdot \ell_\infty(p_i, c_{sl}(r''))$, $w_i \cdot |x_i - c_s^x(r'')| \leq w_i \cdot \ell_\infty(p_i, c_{sl}(r'')) \leq r''$.

For the case that $w_i \cdot \ell_\infty(p_i, C^*) = w_i \cdot \ell_\infty(p_i, c_{ls}(r''))$,

$$
\begin{aligned}
& w_i \cdot |x_i - c_s^x(r'')| \\
= \; & w_i \cdot (x_i - c_s^x(r'')) \\
\leq \; & w_i \cdot (y_i - c_s^y(r'')) \\
= \; & w_i \cdot |y_i - c_s^y(r'')| \\
\leq \; & w_i \cdot \ell_\infty(p_i, c_{ls}(r'')) \\
\leq \; & r''.
\end{aligned}
$$

Thus, we conclude the 1–radius of $(T_{1,L^*})_x \leq r''$. Similarly, for any point p_i in T_{2,L^*}, we also conclude $w_i \cdot |y_i - c_s^y(r'')| \leq r''$. 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 6, the problem now is to search the optimal separating lines for R_{ll} and R_{ss} efficiently. We will show the algorithm for R_{ll} only. It is similar for R_{ss}.

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 $(x_i - y_i)$'s where $p_i \in R_{ll}$ 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 $T_{1,L_{ll}}$ and in $T_{2,L_{ll}}$, 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.

Section 5. The Algorithm for the d–Dimensional Weighted 2–Center Problem

In this section, we will present an algorithm for the d–dimensional weighted 2–center problem, which is based upon the algorithm for the 2–dimensional case.

Let us first consider $P^{ij} = \{ (x_k^i, x_k^j) \mid k = 1, 2, ..., n \}$, the i–j version of P, which is the projection of P to the i–th and j–th coordinates. Let r_+^{ij} (r_-^{ij} resp.) be the optimal solution among the special solutions consists of center points with side indices *ss* and *ll* (*sl* and *ls* resp.) for P^{ij}. For a type of special solutions with side index $E = \epsilon_1\epsilon_2...\epsilon_d$, it induces a type of special solution with side index $E_{ij} = \epsilon_i\epsilon_j$ on the i–j version of P for all $\binom{d}{2}$ versions. Let r_E^{ij} be the optimal solution of P^{ij} among the special solutions with side index E_{ij}. If $\epsilon_i = \epsilon_j$, r_E^{ij} is r_+^{ij}; otherwise r_E^{ij} is r_-^{ij}.

Lemma 7:

The optimal solution among special solutions with side index E is the maximum of r_E^{ij}, $1 \le i < j \le d$.

Proof:

Let r' be the optimal solution and r'' be the maximum of r_E^{ij}, $1 \le i < j \le d$. We should prove that r' = r''.

First, let us show r' ≥ r''. Let $c_{E_{ij}}(r') = (c_{\epsilon_i}^i(r'), c_{\epsilon_j}^j(r'))$ and $c_{-E_{ij}}(r') = (c_{-\epsilon_i}^i(r'), c_{-\epsilon_j}^j(r'))$. It suffices to prove that $r' \ge r_E^{ij}$, for all $1 \le i \le j \le n$. In other words, it suffices to prove that $D(c_{E_{ij}}(r'), r') \cup D(c_{E_{ij}}(r'), r') = P^{ij}$ for all $1 \le i < j \le d$. Let $\ell_\infty^{ij}(p_k, c) = \max \{|x_k^i - c^i|, |x_k^j - c^j|\}$, where $c = (c^1, ..., c^d)$. It is obvious that $\ell_\infty^{ij}(p_k, c) \le \ell_\infty(p_k, c)$. Thus,

$$\begin{aligned} & \max \{ \min \{ w_k \cdot \ell_\infty^{ij}(p_k, c_E(r')), w_k \cdot \ell_\infty^{ij}(p_k, c_{\sim E}(r')) \} \mid k = 1, 2, \ldots n \} \\ \leq\ & \max \{ \min \{ w_k \cdot \ell_\infty(p_k, c_E(r')), w_k \cdot \ell_\infty(p_k, c_{\sim E}(r')) \} \mid k = 1, 2, \ldots n \} \\ =\ & r' \end{aligned}$$

That means $D(c_{E_{ij}}(r'), r') \cup D(c_{E_{ij}}(r'), r') = P^{ij}$, for all $1 \leq i \leq j \leq d$.

Now, let us show that $r' \leq r''$. It suffices to show that $D(c_E(r''), r'') \cup D(c_{\sim E}(r''), r'') = P$. For any $p_i \in P$, let $\ell_\infty(p_i, c_E(r'')) = |x_i^k - c_{\epsilon_k}^k(r'')|$ and $\ell_\infty(p_i, c_{\sim E}(r'')) = |x_i^t - c_{\sim \epsilon_t}^t(r'')|$. If $k \neq t$,

$$\begin{aligned} & w_i \cdot \ell_\infty(p_i, C_E(r'')) \\ =\ & w_i \cdot \min \{ |x_i^k - c_{\epsilon_k}^k(r'')|, |x_i^t - c_{\sim \epsilon_t}^t(r'')| \} \\ \leq\ & r_E^{it} \\ \leq\ & r''. \end{aligned}$$

In the case that $k = t$, we also conclude that $w_i \cdot \ell_\infty(p_i, C_E(r'')) \leq r''$. Thus, $D(c_E(r''), r'') \cup D(c_{\sim E}(r''), r'') = P$. Q.E.D.

By Lemma 7, a straightforward algorithm is for each of 2^{d-1} types of special solutions, find the optimal solution, and then take the minimum among 2^{d-1} optimal solutions as the 2–radius of P. But it takes $O(\binom{d}{2} \cdot n)$ time to compute r_+^{ij} and r_-^{ij}, for all $1 \leq i < j \leq n$ and takes $O(2^{d-1} \cdot d^2)$ time to compute the maximum of r_E^{ij}, $1 \leq i < j \leq d$ for all side indices E. Totally, the straightforward algorithm takes $O(\binom{d}{2} \cdot n + 2^{d-1} \cdot d^2)$ time. In the following, we design an algorithm which takes $O(\binom{d}{2} \cdot n + d^2 \cdot \log d)$ time which reduces the factor $2^{d-1} \cdot d^2$ into $d^2 \cdot \log d$.

Suppose that all the r_+^{ij} and r_-^{ij} for all $1 \leq i < j \leq n$ are already computed. Let us consider the relationship between side indices of P and the side indices of P^{ij} for $1 \leq i < j \leq d$. The relationship can be represented by an edge labeled complete graph G of d vertices $v_1, v_2, \ldots, v_d$. For a side index $E = \epsilon_1 \epsilon_2 \ldots \epsilon_d$, we label the edge

(v_i, v_j) by $+1$ if $\epsilon_i = \epsilon_j$ and -1 if $\epsilon_i \neq \epsilon_j$ for all $1 \leq i < j \leq d$. The dual side index $\bar{E}$ also induces the same edge labeling. It is easy to verify that the product of edge labels in any 3–cycle (cycle of length 3) is $+1$ in such an edge labeling. In fact, the product of the edge–labels in any cycle is also $+1$. Let $\Pi(S)$ denote the product of labels on edges in edge set S and $S_1 \oplus S_2$ denote symmetric difference of edge sets S_1 and S_2. Then $\Pi(S_1 \oplus S_2) = \Pi(S_1) \cdot \Pi(S_2)$. Consider a cycle $C = v_{i_0} v_{i_1} \ldots v_{i_k}$, where $v_{i_0} = v_{i_k}$. Consider 3–cycles $C_{j\ j+1} = v_{i_0} v_{i_j} v_{i_{j+1}} v_{i_0}$, $j = 1, \ldots, k-2$. It is easy to verify that $\overset{k-2}{\underset{j=1}{\oplus}} C_{j\ j+1} = C$. Since $C_{j\ j+1}$ is 3–cycle, $\Pi(C_{j\ j+1}) = +1$ for all $j = 1, \ldots, k-2$. Thus, $\Pi(C) = \Pi(C_{12}) \cdot \Pi(C_{23}) \cdot \ldots \cdot \Pi(C_{k-1\ k}) = +1$. An edge labelling satisfying the above product condition is called coherent.

Lemma 8:

A coherent edge labeling is uniquely determined by the labels on any spanning tree.

Proof:

Let T be any spanning tree with labels on the tree edges. Let e be any edge not in T. Let C_e denote the unique cycle in $T + e$. To make the labeling coherent, the label on e, f_e is forced to be $\Pi(C_e \setminus \{e\})$. Thus, the labeling is unique. Now, we prove that such a labeling is coherent. Consider any 3–cycle C with edge set e_1, e_2 and e_3. If only one of them is not in T, say e_1, then $C = C_{e_1}$. Thus, $\Pi(C) = +1$. If two of them are not in T, say e_1 and e_2, then $C_{e_1} \oplus C_{e_2} = C$. Thus, $\Pi(C) = \Pi(C_{e_1} \oplus C_{e_2}) = \Pi(C_{e_1}) \cdot \Pi(C_{e_2}) = 1$. If all the them are not in T, then $C_{e_1} \oplus C_{e_2} \oplus C_{e_3} = C$. Thus, $\Pi(C) = \Pi(C_{e_1}) \cdot \Pi(C_{e_2}) \cdot \Pi(C_{e_3}) = +1$. Thus, the labeling is coherent.

Q.E.D.

By Lemma 8, any coherent labeling is induced from a side index. Let f_{ij} be the label on edge (v_i, v_j) in the given coherent labeling. Consider the side index $E = \epsilon_1 \epsilon_2 \ldots \epsilon_d$, where $\epsilon_1 = s$ and $\epsilon_i = s$ if $f_{1i} = +1$ and $\epsilon_i = l$ if $f_{1i} = -1$. The side index

will induce a coherent labeling which is the same as the given one on the spanning tree made of edges (v_1, v_i), $i = 2, 3, ..., d$. Thus, by the above observation, they are the same.

Now, consider the complete graph G with edges costed. An edge (v_i, v_j) has two costs; one is r_+^{ij} for the label $+1$ and the other is r_-^{ij} for the label -1. Given a coherent labeling with label s_{ij} on edge (v_i, v_j), $1 \le i < j \le d$, define the cost of the labeling to be $\max \{ r_{s_{ij}}^{ij} \mid 1 \le i < j \le d \}$. In this formulation, by the correspondence between side indices and coherent labelings, the problem to find a side index E such that $\max \{ r_E^{ij} \mid 1 \le i < j \le d \}$ is minimized is equivalent to finding a minimal cost coherent labeling of G.

To find the minimal cost coherent labeling, our algorithm is to label the edges iteratively. By Lemma 8, we need to label the edges of a spanning tree only. Our process is as following. Our first algorithm is as following.

Algorithm Minimal Cost Coherent Labeling:

INPUT: a complete graph G of d vertices, with edge costs r_+^{ij} and r_-^{ij}, $1 \le i < j \le d$

OUTPUT: r^* :the cost of a minimal cost coherent labeling

1. Sort edge costs r_+^{ij} and r_-^{ij}, $1 \le i < j \le d$ into
   a nonincreasing sequence R[1], ..., R[d(d−1)].
2. FOR k = 1 to d(d−1) DO
   2.1 Let $R[k] = r_s^{ij}$.
   2.2 If (v_i, v_j) is not labeled and does not make cycle with labeled edges
   THEN Label (v_i, v_j) with −s.
3. ENDFOR
4. Scan the labeled edges to determine the corresponding side index E
5. $r^* = \max \{ r_E^{ij} \mid 1 \le i < j \le d \}$

Theorem 9:

Algorithm Minimal Cost Coherent Labeling correctly gives a minimal cost coherent labeling.

Proof:

Let $S = \{ s_{ij} \mid 1 \le i < j \le d \}$ be the resulting coherent labeling by executing Algorithm Minimal Cost Coherent Labeling and r be the cost of S. Let r_s^{ij} be the first one in the execution such that (1) edge (v_i, v_j) is already labeled or (2) edge makes a cycle C with the edge already labeled and $\Pi(C \setminus (v_i, v_j)) = s$. If condition (1) happens, it means (v_i, v_j) is labeled as s. If condition (2) happens, it means that in the resulting coherent labeling the edge (v_i, v_j) is labeled as s also. Thus, $r \ge r_s^{ij}$. Since r_s^{ij} is the first one the condition (1) or (2) happens, for all the costs larger than r_s^{ij}, their corresponding edge labels will not occur in the resulting coherent labeling. Thus, $r \le r_s^{ij}$. Therefore, we conclude that $r = r_s^{ij}$.

Let T be the set of edges already labeled when r_s^{ij} is scanned. Let $F = \{ f_{ij} \mid 1 \le i < j \le d \}$ be any minimal cost coherent labeling. We claim that the cost of F, denoted as r'', is r. Consider the labels of F on edges in T. If they the same as that of S. Then in F the edge (v_i, v_j) is also labeled s. Thus, $r'' \ge r_s^{ij} = r$. Otherwise, there exist's an edge (v_a, v_b) in T such that $s_{ab} \ne f_{ab}$. Thus, $r_{f_{ab}}^{ab}$ is scanned before r_s^{ij}. That means $r_{f_{ab}}^{ab} \ge r_s^{ij}$. Hence, in this case, we also have $r'' \ge r_{f_{ab}}^{ab} \ge r_s^{ij} = r$. In another hand, by the minimality of r'', $r'' \le r$. Hence, $r'' = r$. That means S is also a minimal cost coherent labeling. Q.E.D.

The Algorithm Minimal Coherent Labeling is essentially the same as the Kruskal's algorithm for the minimal spanning tree problem. In the algorithm, the

step 1 take $O(d^2 \log d)$ time for sorting. In step 2, for each edge, it should check whether the edge makes a cycle with the labeled edges. The check process can be implemented by the set union find algorithm which takes $O(\log^* d)$ amortized time. Thus, step 2 takes $O(d^2 \log^* d)$ time. The step 4 takes $O(d)$ time. In total, the time complexity is $O(d^2 \log d)$.

Section 6. Concluding Remarks

We have presented an $O(\binom{d}{2} \cdot n + d^2 \log d)$ time algorithm for the ℓ_∞–distance weighted 2–center problem and an $O(2^d \cdot d \cdot n)$ algorithm for the weighted tailored 2–partition problem. In the 2–dimensional case, since the ℓ_∞–distance is equivalent to the rectilinear distance, we then solve both the problems under rectilinear distance in $O(n)$ time also. But, for the higher dimensional space, these two distances are different. That is still an open problem to solve the problems under rectilinear distance in higher dimensional spaces efficiently.

References:

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

[2] Hershberger, J. and Suri, S. (1989) " Finding Tailored Partitions," Proceedings of the Fifth Annual Symposium on Computational Geometry, pp. 255–265.

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

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

[5] 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.

[6] 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.

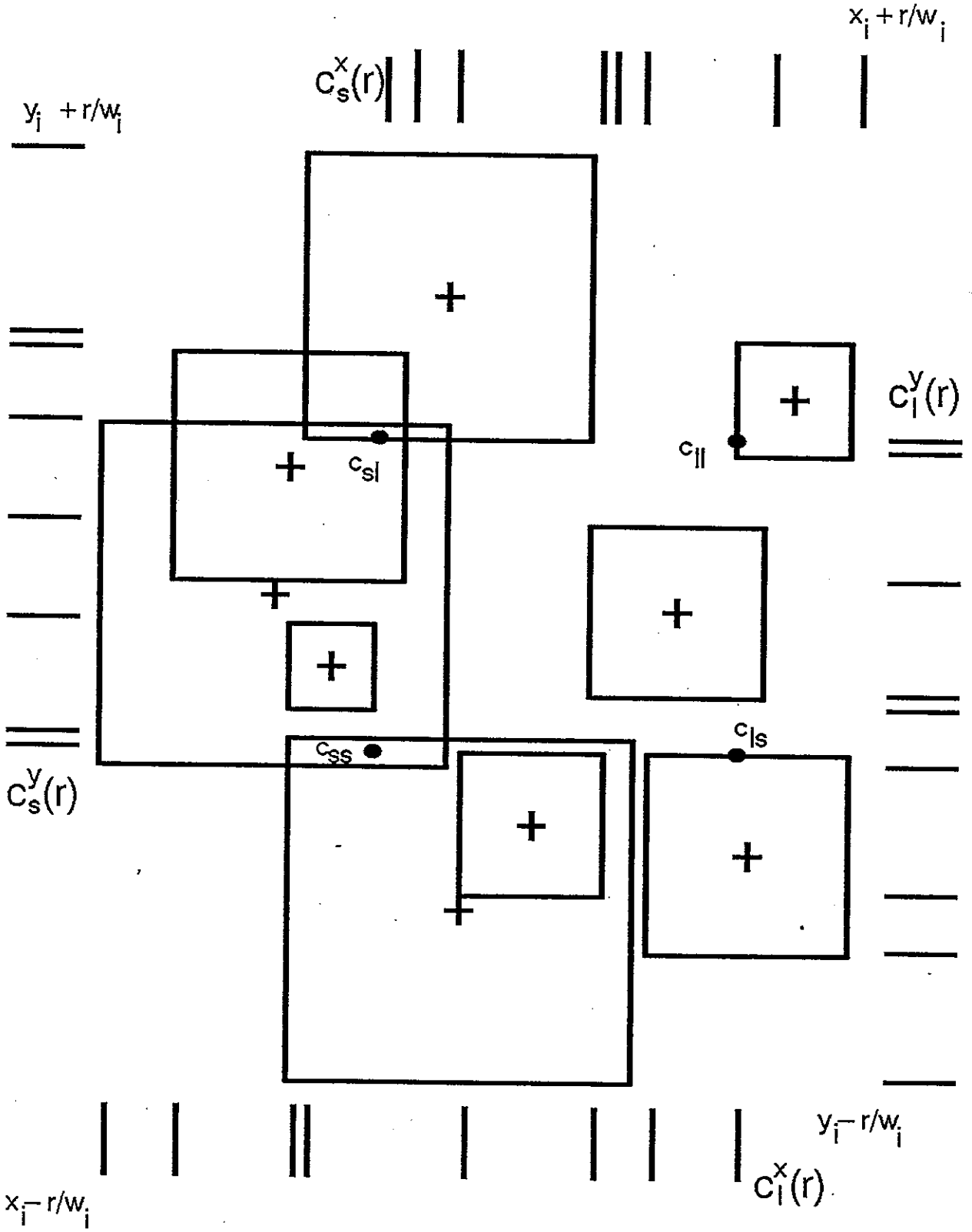

Figure 1

Illustration of center points

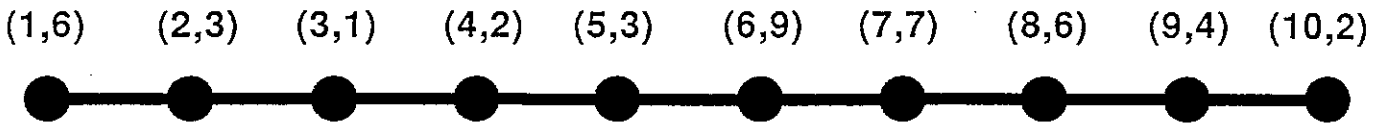

Figure 2

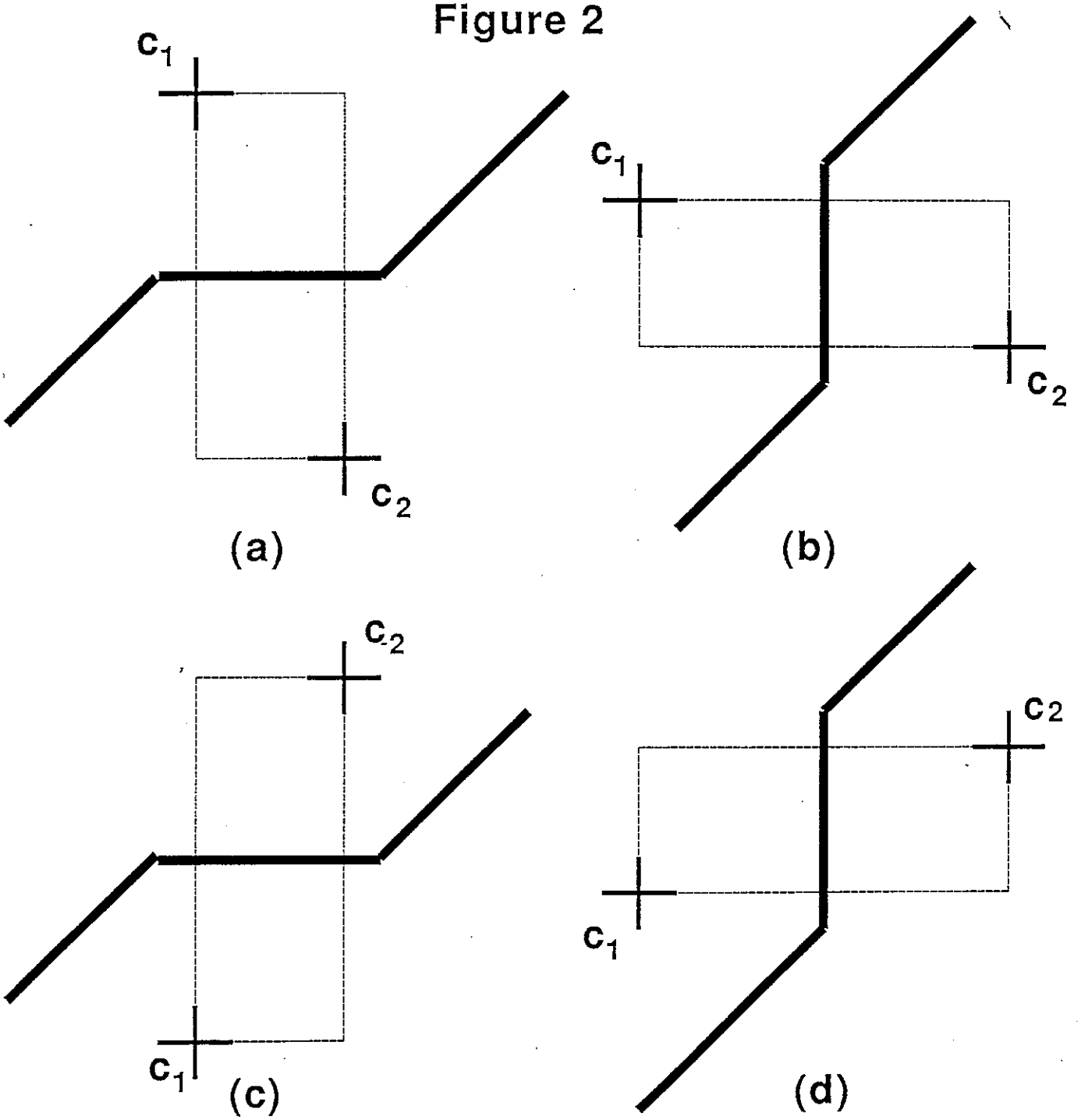

Figure 3

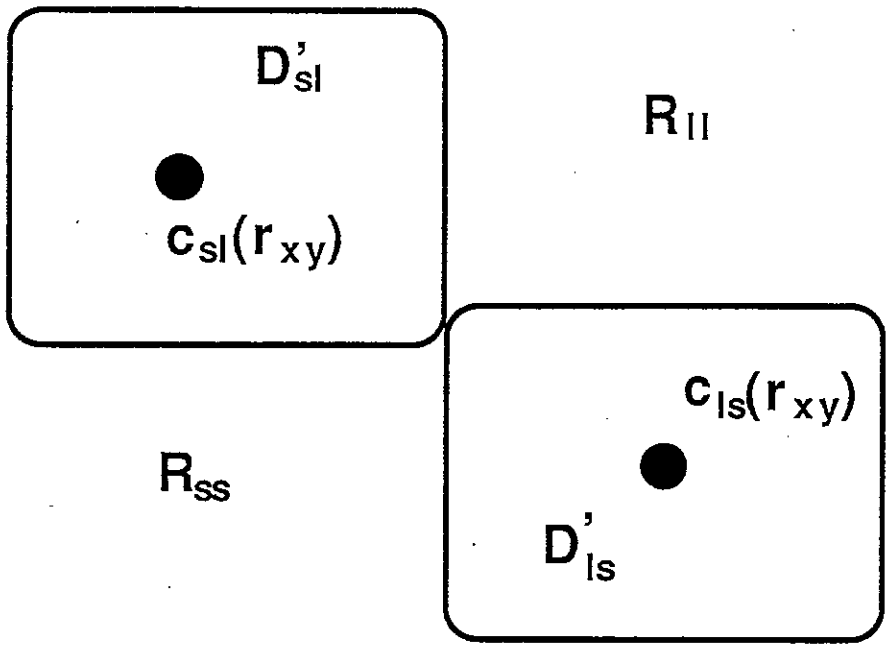

Figure 4