Finding The Maximum Density Axes Parallel Regions for Weighted Point Sets

Ananda Swarup Das * Prosenjit Gupta † Kannan Srinathan ‡ Kishore Kothapalli §

Abstract

In this work we study the problem of finding axes-parallel regions of maximum density for weighted point sets in $\mathbb{R}^2$ and $\mathbb{R}^3$. The 2-d variant is motivated by applications in thermal analysis of VLSI chips.

1 Introduction

We are given a set of n weighted points in $\mathbb{R}^2$ (respectively in $\mathbb{R}^3$). Our goal is to find the highest density axes parallel rectangle in $\mathbb{R}^2$ (in $\mathbb{R}^3$ we find the highest density axes parallel 3-dimensional box) where density of a region is defined as the sum of weights per unit area (respectively per unit volume). The problem for finding the highest density axes parallel rectangle for $\mathbb{R}^2$ was introduced by Majumder et al. in [1] and has applications in thermal analysis of VLSI chip.

Preliminaries and Problems: Let us first define the term density formally. The definition that we are providing here has been mentioned in [1].

Definition 1 *Let F be a rectangular floor containing a set S of n points such that no two points lie on the same horizontal or vertical line. Each point $p_i \in S$ is associated with a positive real weight w_i. The density of an axes parallel region R with area $A(R)$ is defined as $\frac{\sum_{p_i \in R} w_i}{A(R)}$.*

It should be mentioned that in case of unweighted points (or when points are of equal weight), the density is $\frac{|S \cap R|}{A(R)}$. Next, we introduce the main problem that we study in this work.

Problem 1.1 *Given a set S of n points in a plane such that each point has a positive weight associated with it, find the cluster of $k \geq 2$ points in S such that the minimum area axes parallel rectangle covering them attains the highest density.*

*International Institute of Information Technology , Hyderabad, India, anandaswarup@gmail.com

†Heritage Institute of Technology, Kolkata, India , prosenjit_gupta@acm.org

‡International Institute of Information Technology , Hyderabad, India, srinathan@iiit.ac.in

§International Institute of Information Technology , Hyderabad, India, kkishore@iiit.ac.in

Our Contribution:
In this work we first show that if the coordinates of the n points are integers in $[0, U] \times [0, U]$, and the points have distinct positive weights $w(p)$, then finding the highest density axes parallel rectangle can be done in $O(n \log n \log U)$ time using $O(n \log^2 n)$ storage. We then present another data structure which solves Problem 1.1 in $O(n \log n \log U)$ time using $O(n \frac{\log^2 n}{\log \log n})$ storage. We finally discuss how to find the highest density axes parallel 3-dimensional box provided the coordinates of the n weighted points are integers in $[0, U] \times [0, U] \times [0, U]$.

Special note: We assume that all the coordinates of the points are distinct. This particular assumption of distinctness is important for the correctness of the solution as because, if there are two points whose x coordinates (or y coordinates) are the same, the area of the axes parallel rectangle induced by them is zero and hence the corresponding density will be ∞.

2 Solution

In [1], Majumder et al. proved the following,

Lemma 1 *Let each point $p_i \in S$ be associated with a positive weight w_i and there exists a cluster $S' \subseteq S$ of $k \geq 2$ points such that no two of them lie in the same horizontal (vertical) line. Then there exists a pair $p_i, p_j \in S'$ such that the density of the smallest area axes parallel rectangle containing (p_i, p_j) is greater than the density of the axes parallel rectangle containing S'.*

In short, the Lemma 1 says the following: let the cluster S' contains $k > 2$ points and let the density D' of S' is the sum of the weights of the k points in S' divided by the smallest area of the axes parallel rectangle bounding all the points of S'. Also assume that S' is the cluster of highest density among all possible clusters for the points of S. Then as a contradiction, it can be shown that there exist two points $p_i, p_j \in S'$ such that the density of the smallest area axes parallel rectangle containing (p_i, p_j) is greater than the density of the axes parallel rectangle containing S'. It therefore means that the maximum density occurs for a cluster $C \subseteq S$ containing only two points. Though Majumder et al. proved the Lemma 1, they solved the unweighted

version of the problem in which case, the problem gets reduced to finding the smallest area axes parallel rectangle enclosing two points of S as diagonally opposite corners. No efficient solution is however known for the problem with a weighted point set. In this work, we show a simple "divide and conquer" technique for the problem assuming that the coordinates of the weighted points are integers in $[0, U] \times [0, U]$.

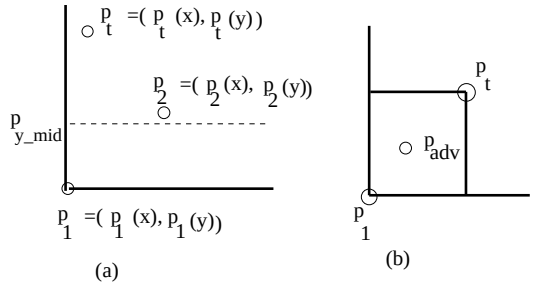

Figure 1: (a) The y coordinate of the end point of the dotted semi infinite horizontal segment is $p_{y\_mid} = \frac{p_1(y)+p_t(y)}{2}$. (b) The rectangle enclosing the points p_1, p_t as diagonally opposite corner points cannot be a candidate for highest density rectangle as it contains the point p_{adv} in it.

2.1 Our Algorithm

1. Consider a point $p_1 \in S$. Let $p_1 = (p_1(x), p_1(y))$. Consider the northeast quadrant $NE(p) = (P_1(x), \infty] \times (P_1(y), \infty]$.

2. Let the weight of p_1 be $w(p_1)$. Create a query box $q = [p_1(x), \infty) \times [p_1(y), \infty) \times [w(p_1), \infty)$.

3. Find the point with the smallest x coordinate in the query box q. Let this point be denoted by $p_t = (p_t(x), p_t(y))$.

4. Consider the axes parallel rectangle $R_{1,t}$ enclosing the points p_1, p_t as the diagonally opposite corners. Check if the rectangle $R_{1,t}$ contains any point $p_{adv} \in S$. See Figure 1 (b).

   (a) If $R_{1,t} \cap S = \emptyset$, then $R_{1,t}$ is a candidate for being the highest density rectangle.

   (b) Else, as per Lemma 1, $R_{1,t}$ cannot be the highest density rectangle.

5. Repeat the above steps but this time with the query box $q = [p_1(x), \infty) \times [p_1(y), \lfloor \frac{p_1(y)+p_t(y)}{2} \rfloor] \times [w(p_1), \infty)$. The reason for this step is explained in Lemma 2.

6. Stop the algorithm if $\lfloor \frac{p_1(y)+p_t(y)}{2} \rfloor = p_1(y)$.

7. Follow a similar procedure for the southeast quadrant $SE(p_1)$, southwest quadrant $SW(p_1)$, and northwest quadrant $NW(p_1)$ for the point p_1.

8. Repeat the steps (1) to (7) for all the points in S.

Consider the query box $q = [p_1(x), \infty) \times [p_1(y), \infty) \times [w(p_1), \infty)$ mentioned in the step (2) of the algorithm and let $S_{NE(p_1)}$ be the set of all the points in S lying in the northeast quadrant of p_1 such that these points have their respective weight greater than the weight of p_1. Then we have the following lemma.

Lemma 2 *Let $p_t, p_2 \in S_{NE(p_1)}$ be two points such that (a) $p_t = (p_t(x), p_t(y))$ has the smallest x coordinate among all points in $S_{NE(p_1)}$ and (b) the axes parallel rectangle $R_{1,2}$ enclosing p_1 and p_2 as diagonally opposite corners has the highest density in $S_{NE(p_1)}$. Then the y coordinate of p_2 must be less than $\frac{p_1(y)+p_t(y)}{2}$.*

Proof: See Figure 1 (a). Since $p_2(x) > p_t(x) > p_1(x)$, $p_2(x) - p_t(x) < p_2(x) - p_1(x)$. By Lemma 1, $p_2(y) < p_t(y)$ or else the rectangle $R_{1,2}$ will also contain the point p_t and hence cannot have highest density. If $p_2(y) \in [\frac{p_1(y)+p_t(y)}{2}, p_t(y)]$, then $p_t(y) - p_2(y) \leq p_2(y) - p_1(y)$. Therefore the area of the rectangle $R_{t,2}$, the one enclosing p_t, p_2 as the diagonally opposite points will have its area $A(R_{t,2}) < A(R_{1,2})$. Since $w(p_t) > w(p_1)$, $\frac{w(p_t)+w(p_2)}{A(R_{t,2})} > \frac{w(P_1)+w(P_2)}{A(R_{1,2})}$, a contradiction to the fact that $R_{1,2}$ has the highest density.□

We therefore conclude the section by stating the following lemma.

Lemma 3 *When the coordinates of the points are integers and in the range of $[0, U] \times [0, U]$, the maximum number of candidate rectangles we generate for any point p_1 is $O(\log U)$. The total number of candidate rectangles thus generated is $O(n \log U)$.*

3 The Choice of Data Structures

As evident from our algorithm in Section 2.1, we need two particular data structures namely (a) a 2-d range aggregate data structure D such that given a query rectangle q we can efficiently decide if $q \cap D = \emptyset$ or not, and (b) a 3-d range successor data structure for efficient execution of our algorithm. We skip the discussion on 2-d range queries as they are very well studied [8] and focus on 3-d range successor problem. Formally the problem can be defined as follows.

Problem 3.1 *Given a set S of n points in $\mathbb{R}^3$ preprocess them into a data structure such that given an axes parallel d-box q for $d = 3$, one can efficiently report the point with the smallest x coordinate in $q \cap S$.*

The above problem has been studied in [3, 7]. The data structure presented in [7] takes expected $O(n \log n \log\log n)$ preprocessing time, occupies $O(n \log^2 n)$ space and can be queried in $O(\log n \log\log n)$ time. The data structure presented in [3] can be built in $O(n^{1+\epsilon})$ time, occupies $O(n^{1+\epsilon})$ space and can be queried in $O(1)$ time. For any data structure for the range successor problem let $P(n)$ be the preprocessing time and let $Q(n)$ be the query time. Since we may need to answer $O(n \log U)$ range successor queries in the worst case for solving Problem 1.1, the total time required to answer range successor queries is $R(n) = P(n) + O(n \log U)Q(n)$. Hence we propose a data structure RSQ to solve the 3-dimensional range successor problem so that the $R(n)$ value is smaller than that obtained by using the solutions from [3] and [7]. RSQ is a variant of range aggregate tree with fractional cascading [8] and is also a variant of [4].

3.1 The Preprocessing Algorithm

1. Let $x_1, x_2, \ldots, x_n$ be a sorted list of points on the real line, being the x-projections of the points in S. Consider the elementary intervals created by the partitioning of the real line induced by these points. Construct a balanced binary tree T_x, associating the above elementary intervals with its leaves.

2. To each internal node μ, assign an interval $int(\mu)$ which is union of the elementary intervals of the points associated with the leaf nodes of the subtree rooted at μ.

3. At each internal node $\mu \in T_x$ maintain an array A_μ which stores the y coordinates of the points present in the leaf nodes of the subtree rooted at μ.

4. Also maintain a range minima data structure RM_{A_μ} (see [6]) such that given two indices i, j of A_μ, we can return the maximum weight among the points whose y coordinates are stored between $A_\mu[i]$ to $A_\mu[j]$.

5. Let w and v be the two children of μ. Since $A_\mu = A_w \cup A_v$, each index i of A_μ has two pointers one pointing to the smallest value in A_w which is greater than equal to $A_\mu[i]$ and the other pointing to the smallest value in A_v greater than equal to $A_\mu[i]$. Similarly each index i of A_μ has two pointers one pointing to the largest value in A_w which is smaller than equal to $A_\mu[i]$ and the other pointing to the largest value in A_v smaller than equal to $A_\mu[i]$.

6. Now, at the node μ, construct a height balanced binary search tree $T_{\mu,y}$ on the points of A_μ.

7. At each node $\phi \in T_{\mu,y}$, store a sorted array W_ϕ which stores the weights of the points whose y coordinates are stored in the leaf nodes of subtree rooted at ϕ.

8. Maintain a range minima data structure RM'_ϕ such that given two indices i, j of W_ϕ, we can return the minimum x coordinate among the points stored between $W_\phi[i]$ to $W_\phi[j]$.

Lemma 4 *The data structure RSQ for range successor queries can be built in $O(n \log^2 n)$ time and occupies $O(n \log^2 n)$ space.*

3.2 The Query Algorithm

Let our query be $q = [x_1, \infty) \times [y_1, y_2] \times [wt, \infty)$. We wish to find out the the point $p \in S$ with the smallest x coordinate and fitting inside q.

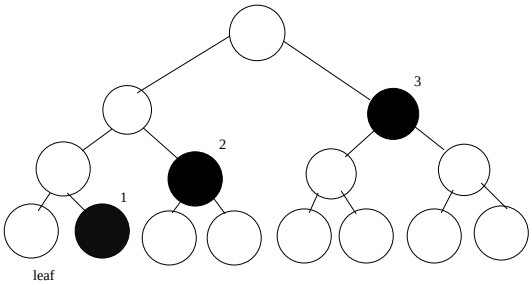

Figure 2: The nodes marked black are the ones to which the interval $[x_1, \infty)$ is allocated

1. Find the leaf node $leaf \in T_x$ such that it contains the value x_1. Trace the path π from the node $leaf$ to the root node of the tree T_x.

2. For any node v which is a right sibling of any node u on the path π, its interval $int(v) \subset [x_1, \infty)$. Allocate the semi-infinite interval $[x_1, \infty)$ to the nodes v. See Figure 2. The nodes marked black are the ones to which the interval $[x_1, \infty)$ is allocated. In general, the interval $[x_1, \infty)$ will be allocated to $O(\log n)$ canonical nodes of T_x. Since $[x_1, \infty)$ is a semi-infinite interval, it will be allocated to at most one node at each level of the tree. For $l \in O(\log n)$, let us number these nodes as $v_1, \ldots, v_l$, starting from the leaf level of the tree T_x. See Figure 2.

3. Search the array A_{root} to find the indices i and j such that $y_1 \leq A_{root}[i] < A_{root}[j] \leq y_2$. Then find the smallest value greater than y_1 and the largest value smaller than y_2 in the all the arrays starting from $A_1, \ldots, A_l$. This can be done by chasing the pointers starting from A_{root}.

4. As v_1 is a leaf node, $|A_{v_1}| = 1$. Check if the y coordinate stored in the node marked 1 is in between y_1 and y_2. If so, check if the weight of the point is greater than wt. If so, we have our desired point in the node 1.

5. Else we move to the node marked 2. Let i', j' be the indices such that $y_1 \leq A_2[i'] < y_2 \leq A_2[j']$.

6. Using Range Minima data structure RM_{A_2}, find the maximum weight w' among the points stored in between $A_2[i']$ and $A_2[j']$.

7. Let $w' > wt$. It means we have our desired point at the node 2. We move to the node 2.

   - Consider the data structure $T_{\mu,y}$ for $\mu = 2$, created in steps (6) to (8) of the preprocessing step. By repeating steps similar to (4) to (7) of the query algorithm on the tree $T_{\mu,y}$, on required auxiliary arrays W_ϕ and on required range minima data structures RM'_ϕ, we can find out the point with the smallest x coordinate present in $S \cap q$.
   - Return the point discovered in the previous step

8. On the other hand, if $w' < wt$, we move to the next node that we have marked as node 3.

9. For any query q, the above steps continue until

   - we discover the point with the smallest x coordinate in q or
   - we have visited all the l nodes and have failed to find the point $p \in S$ with the smallest x coordinate and fitting in q .

Lemma 5 *The data structure RSQ supports 3-dimensional range successor queries in $O(\log n)$ time.*

Proof: Finding the leaf node $leaf \in T_x$ needs $O(\log n)$ time. Next, searching the indices i, j such that $y_1 \leq A_{root}[i] < A_{root}[j] \leq y_2$, needs another $O(\log n)$ time. Once the indices i, j of the root node are found, finding the respective indices in all the arrays of the nodes marked black in Figure 2 needs $O(\log n)$ time. Next, we check if the point in the node marked 1 fits our requirement. If so, our job is done in $O(\log n)$ time. Else, we move to the node marked 2 and find the maximum weight among the points which are stored between $A_2[i']$ to $A_2[j']$ where $y_1 \leq A_2[i'] < A_2[j'] \leq y_2$. This needs $O(1)$ time using the range minima data structure. If we find the maximum weight to be greater than wt, we restrict our searching only at node 2. Next, we repeat similar searches at the tree $T_{2,y}$. This needs another $O(\log n)$ time. Hence the result. □

From Lemma 4 and Lemma 5, we conclude the following theorem.

Theorem 6 *A set S of n points in $\mathbb{R}^3$ can be preprocessed in time $O(n \log^2 n)$ into a data structure of size $O(n \log^2 n)$ so that given a query axes-parallel rectangle q, the range successor query can be answered in $O(\log n)$ time.*

By using Lemma 3 and Theorem 6, we can conclude the following:

Lemma 7 *If the points are in the range $[0, U] \times [0, U]$, $O(\log U)$ candidate rectangles for the point P_1 are generated in $O(\log U \log n)$ time. Hence the total time needed to find the highest density axes parallel rectangle is $O(n \log U \log n + n \log^2 n)$.*

The above lemma will also hold when the coordinates of the points are integers in $\mathbb{R} \times [0, U]$.

3.3 A Reduced Spaced Data Structure

Next we present RSL, a reduced space data structure for the 3-dimensional range successor problem.
Steps of Preprocessing :

1. Let us change the degree of the internal nodes of the tree T_x to $O(\sqrt{\log n})$ instead of two. The height of the tree is therefore $O(\frac{\log n}{\log \log n})$.

2. In each internal node $\mu \in T_x$, we create the auxiliary array A_μ. The array A_μ stores the sorted y coordinates of the points whose x coordinates are associated in the leaf nodes.

3. Any element of A_μ will now have with 2 pointers pointing to two elements in each of the auxiliary arrays belonging to the $\sqrt{\log n}$ children of μ. One pointer will point to the smallest value in the array A_w greater than the element and the other pointer will point to the largest element in the array A_w smaller than the element. Here w is a child of μ. Hence any element in A_μ will have $2\sqrt{\log n}$ pointers. The construction of A_μ is discussed later.

4. Repeat the steps (3), (6), (7), (8) of the preprocessing algorithm in sub section 3.1.

3.3.1 Construction of the array A_μ

Building the array A_μ is easy. Let $A_{w_1}, \ldots, A_{w_{\sqrt{\log n}}}$ are the sorted arrays present at the $\sqrt{\log n}$ children of the node μ. From each array A_{w_i} $\forall i = 1, \ldots, \sqrt{\log n}$, take its smallest element and construct a min-heap. The height of the heap will be $O(\log \log n)$. Now the root node of the heap will contain the smallest element of the heap. We will store the element of the root to the first available index of A_μ and this element will have a pointers to the elements currently present in the heap at

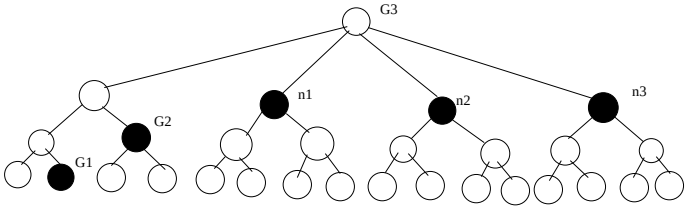

Figure 3: The nodes marked black are the ones to which the interval $[x_1, \infty)$ is allocated

their respective arrays. It will also have a pointer to the next value of the array A_{w_i} from which it originated. From the array A_{w_i} take the next element and insert it into the heap. This method has been used in [5] for reporting the top k weights in a query rectangle.

Lemma 8 *The data structure RSL for range successor queries can be built in $O(n\frac{\log^2 n}{\log\log n})$ time and occupies $O(n\frac{\log^2 n}{\log\log n})$ space.*

3.4 The Query Procedure

Let our query $q = [x_1, \infty) \times [y_1, y_2] \times [wt, \infty)$ and we would like to find the point with the smallest x coordinate in it. Let us denote the tree T_x as the primary tree. This tree is a variant of the range tree used by [2] and [3]. As mentioned in [2], an interval $[a, b]$ can be represented as a union of node ranges of some nodes $v_1, \ldots, v_k$ that can be grouped into $\frac{\log n}{\log\log n}$ groups $G_1, \ldots, G_h$. Each group G_i contains a set of children $v_{l_i}, v_{l_{i+1}}, \ldots, v_{l_r}$ for some node v_l. There are at most two groups in each level. Hence there are $O(\frac{\log n}{\log\log n})$ groups. Consider the interval $[x_1, \infty)$. We can also write this interval as $[x_1, x_{max}]$ where x_{max} is the maximum x coordinate among all the points of the set S. See Figure 3. Let $[x_1, x_{max}]$ is equal to the union of the intervals of the black nodes. These black nodes are divided into three groups and are assigned to the nodes marked G_1, G_2 and G_3.

Let p_1 be the point with the smallest x coordinate in q. Let us also suppose that p_1 is not present in the nodes marked by the groups G_1 and G_2. Now suppose we are at the node marked G_3. We need to decide, among the three children of G_3 marked as n_1, n_2, n_3, which node should we visit ? It can be noticed that if there is even a single point from the node marked n_1 fitting inside the query q, the point p_1 has to be present in n_1. Therefore, we need a way to decide if it is profitable to visit the node n_1 (or in fact any node). Remember that we have an array A_{G_3} at the node marked G_3 such that A_{G_3} sorts all the points that are stored at the leaf nodes of the subtree rooted at G_3. The array A_{G_3} is sorted according to the y coordinates of the points that are present in the leaf nodes of the tree rooted at node marked G_3. Moreover each index i of the array A_{G_3} has a pointer to the smallest value greater than $A_{G_3}[i]$ and the largest value smaller than $A_{G_3}[i]$ in the array A_{n_1}. So if we find the smallest value greater than y and the largest value smaller than y_1 for the array A_{G_3}, then we can easily do the same for A_{n_1}. Once we discover the two indices of the array A_{n_1}, using range minima query we can find the largest weight w' for the points whose y coordinates are stored in the array A_{n_1} in between those two indices in $O(1)$ time. If $w' > wt$, we focus our searching only at the node n_1. We move to the node n_1 and follow steps similar to that of the query algorithm in subsection 3.2 to find out the point with the smallest x coordinate present in $S \cap q$ and return it as an answer. On the other hand, if $wt < w'$, we move to the next node (n_2 as per our Figure 3).

Lemma 9 *The data structure RSL supports 3-dimensional range successor queries in $O(\log n)$ time.*

Theorem 10 *A set S of n points in $\mathbb{R}^3$ can be preprocessed in time $O(n\frac{\log^2 n}{\log\log n})$ into a data structure of size $O(n\frac{\log^2 n}{\log\log n})$ so that given a query axes-parallel rectangle q, the range successor query can be answered in $O(\log n)$ time.*

Using the above theorem, we conclude the following

Theorem 11 *Given a set of n weighted points whose coordinates are integers in $[0, U] \times [0, U]$, the highest density axes parallel rectangle can be found in time $O(n\frac{\log^2 n}{\log\log n} + n \log U \log n)$ using space of $O(n\frac{\log^2 n}{\log\log n})$.*

4 On Finding the Highest Density Axes Parallel 3-dimensional box

Before we start the section, we define our density as follows:

Definition 2 *Let S be a set of n points in $\mathbb{R}^3$. The density of the axes parallel d-box R (for $d = 3$) with volume $V(R)$ and covering the points in R, is defined as $\frac{\sum_{p_i \in R} w_i}{V(R)}$.*

In this section, we wish to solve the following problem

Problem 4.1 *We are given a set of n weighted points such that their coordinates are integers in the range of $[0, U] \times [0, U] \times [0, U]$. We wish to find the cluster of $k \geq 2$ points such that the minimum area axes parallel d-box for $d = 3$ covering them attains the highest density.*

Before we try to solve the Problem 4.1, we refer to the following fact proved by Majumder et al. in [1].

Fact 1 *Let* $Q = \frac{\sum_{i=1}^{n} a_i}{\sum_{i=1}^{n} b_i}$ *for* $a_i, b_i > 0$, *then* $Min_{i=1}^{n} \frac{a_i}{b_i} \leq Q \leq Max_{i=1}^{n} \frac{a_i}{b_i}$

Now, assuming that all the coordinates of the points are distinct, we propose the following lemma

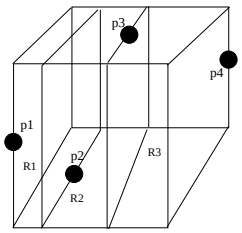

Figure 4: The case of axes parallel 3-dimensional box for $d = 3$

Lemma 12 *The highest density axes parallel* 3*-dimensional box will consist of two points.*

Proof: The proof is similar to the proof of Lemma 1 provided in [1]. See Figure 4. Let our highest density axes parallel 3- dimensional box contains k points $p_1, p_2, \ldots, p_k$. We partition the box into $k-1$ smaller pieces as shown in the Figure 4. The density of our box is $\frac{wt(p_1)+wt(p_2)+\ldots+wt(p_k)}{V(R)} \leq \frac{wt(p_1)+wt(p_2)}{V(R_1)} + \frac{wt(p_2)+wt(p_3)}{V(R_2)} + \ldots + \frac{wt(p_{k-1})+wt(p_k)}{V(R_{k-1})}$, where $V(R_i)$ denotes the volume of the rectangular axis parallel d-box R_i. Using Fact 1, we can prove the lemma. □

The highest density axes parallel 3-dimensional box will contain two points of the set S as diagonally opposite corners. Using the Lemma 12 and the ideas of Section 2.1 and data structure similar to 3.1, we have the following theorem.

Theorem 13 *Given a set of* n *weighted points whose coordinates are integers in range of* $[0, U] \times [0, U] \times [0, U]$, *the highest density axes parallel* 3*-dimensional box can be found in* $O(n \log U \log^3 n)$ *times using* $O(n \log^3 n)$ *space.*

References

[1] S. Majumder, B. B. Bhattacharya, *On the density and discrepancy of a 2D point set with applications to thermal analysis of VLSI chips.* Information Processing Letters **107** (2008), pp. 177–182.

[2] Y. Nekrich, *Orthogonal Range Searching in Linear and Almost-Linear Space.* Computational Geometry: Theory and Applications **42(4)** (2009), pp. 342–351.

[3] C. C. Yu, W. K. Hon, B. F. Wang, *Improved Data Structures for Orthogonal Range Successor Queries.* Computational Geometry: Theory and Applications **44** (2011), pp. 148– 159.

[4] S. Saxena, *Dominance made simple.* Information Processing Letters **109** (2009), pp. 419–421.

[5] S. Rahul, P. Gupta, R. Janardan K. S. Rajan, *Efficient top-k queries for orthogonal ranges*, In Proc. International Workshop on Algorithms and Computation, Springer Verlag Lecture Notes in Computer Science No. 6552, pp. 110–121.

[6] H. Yuan, M. Atallah, *Data Structures for Range Minimum Queries in Multidimensional Arrays.* In Proceedings of SODA 2010, pp. 150–160.

[7] H. P. Lenhof, M. H. M. Smid, *Using Persistence for adding range restrictions to searching problems.* RAIRO Theoretical Informatics and Applications. **28(1)** (1994), pp. 25–49.

[8] M. de. Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer, Verlag, 2000.