國立交通大學

資訊科學與工程研究所

碩士論文

最大密度矩形之找尋問題

Density Finding on a Rectangle

研 究 生：羅偉力

指導教授：蔡錫鈞　教授

中華民國九十八年三月

最大密度矩形之找尋問題
Density Finding on a Rectangle

研 究 生：羅偉力　　Student：Wei-Li Luo

指導教授：蔡錫均　　Advisor：Shi-Chun Tsai

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

March 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年三月

Density Finding on a Rectangle

Wei-Li Luo

March 6, 2009

Abstract

We define the *density finding problem on a rectangle*(DFR for short) as follows. Given an m-by-n rectangle R, each unit block is attached with a value and a weight. A subrectangle S in R is an m'-by-n' rectangle where $1 \leq m' \leq m$ and $1 \leq n' \leq n$. The value(weight) of S is the sum of the value(weight) of each block in S. Let A and W be the value and weight of S respectively. The goal is to find a subrectangle S in R such that the density of S is closest to a specified real number δ, where the density of S is defined as the ratio of A and W, and $L \leq W \leq U$ for two specified positive numbers L and U.

When $m = 1$, Luo et al. [10] give a $O(n \log n)$ time solution. Moreover, if $\delta \to \infty$, Chung et al. [5] and Bernholt et al. [3] both give $O(n)$ time solutions in different ways. In this thesis, we will give a $O(m^2 n \log n)$ time solution for any δ and $O(m^2 n)$ time solution if $\delta \to \infty$ when $m < n$. Besides, we show that solving DFR takes $\Omega(mn \log n)$ when $m < n$.

Acknowledgements

I am grateful to my advisor, Dr. Shi-Chun Tsai, for his guidance and encouragement. I thank my family for their spiritual support.

Also, I am obliged to all members in CCIS lab. They supply their help during the studying and writing period.

Moreover, I thank my friends, Doss, Parallelism, Dogswang, Cooty, Duncan, Kong, Mardy, and all I meet in Hsin-Chu these years.

Specially, I deeply appreciate and respect Emerson, my friend and spiritual advisor. He gives me many ideas in studying and life.

To Dan and Mao.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the topics in VLSI chip design is thermal analysis. The intent of the design is to decrease the heat and cool down the hot chips by proper placement. The designers may set a temperature limit. Once a region is found that it is too hot, i.e., exceeds the specified temperature, some hot chips should be replaced. To handle this problem, we abstract the thermal sources as unit squares attached with positive values on a 2D plane. Thus, the problem reduces to finding a region with the largest density of thermal sources. We describe the problem below.

Definition 1 (Abstracted VLSI chip design problem)**.** *Let R be a rectangle on a 2D plane. The height and the width of R are m and n respectively, $m, n \in \mathbb{N}$. We divide R into $m \times n$ blocks. Each row has n blocks, and there are m rows totally. We call the block b_{ij} if it locates at the ith row and the jth column. For each block b_{ij}, we assign a value-weight pair (a_{ij}, w_{ij}), where $w_{ij} > 0$ for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$.*

By the above definition, we may consider a block b_{ij} with value $a_{ij} > 0$ and $w_{ij} = 1$ as a thermal source, and a block is empty if $a_{ij} = 0$ and $w_{ij} = 1$. Here w_{ij} represents the area on a 2D plane.

In this paper, we generalize the abstracted VLSI chip design problem. We define the problem formally.

Definition 2 (Rectangular density finding problem, DFR for short)**.** *Let (A, W) be a pair of $m \times n$ matrices. Each entry in value matrix A is a real number. Each entry in weight*

(0,1)	(0,1)	(0,1)	(1,1)	(0,1)	(0,1)
(4,1)	(0,1)	(0,1)	(0,1)	(0,1)	(0,1)
(0,1)	(0,1)	(2,1)	(0,1)	(0,1)	(0,1)
(0,1)	(0,1)	(0,1)	(0,1)	(0,1)	(1,1)

Figure 1.1: An example of abstracted VLSI chip design. Each block contains a (value,weight)-pair.

matrix W is positive. A sub-matrix A' of A is a matrix whose entries are the intersection of the rows $i, i+1, \dots, j$ and the columns $k, k+1, \dots, l$, $1 \le i \le j \le m$, $1 \le k \le l \le n$. We say the sub-matrix A' is defined by the 4-tuple indices (i, j, k, l). We define the sub-matrix W' of W as the same of A'. We name the sum of all entries in A' value, and the sum of all entries in W' weight. The ratio of the value and the weight is called density. Given a pair of matrices (A, W), a lower bound of weight L, an upper bound of weight U, and a real number δ, the goal is to find a 4-tuple indices (i, j, k, l), such that the weight of W' is between L and U, and the density is closest to δ.

The above is a generalization of *abstracted VLSI chip design problem.* It is a variation of *density finding problem.* We list some original problems below.

When $m = 1$, it is the case of [8].

Definition 3 (Segmental density finding problem)**.** *Given a sequence S of number pairs (a_k, w_k) with $w_k > 0$ for $k = 1, \dots, n$, two numbers L and U and a real number δ, we want to find a segment $S(i, j)$ such that the density of the segment is closest to δ over all (i, j)-pairs with $L \le w_i + \cdots + w_j \le U$, i.e., find*

$$\min_{i \le j} \left| \frac{a_i + a_{i+1} + \cdots + a_j}{w_i + w_{i+1} + \cdots + w_j} - \delta \right|, \text{ where } L \le w_i + \cdots + w_j \le U.$$

If $\delta \to \infty$, it reduces to the maximum-density segment problem [5].

Figure 1.2 is an example of *segmental density finding problem.*

In [11], the input is a set of points on a 2D plane, and no two points lie on the same horizontal or vertical line.

(2,3)	(-1,2)	(5,4)	(3,1)	(1,2)	(1,3)	(4,5)

Figure 1.2: An example of segmental density finding problem.

Definition 4 (Maximum-density region problem)**.** *Let R be a rectangle containing n points. In R, every two points lie on the different horizontal and vertical lines. We attach a positive real number a_i to each point p_i. We say that a_i is the value of p_i. An axis-parallel region T is a rectangle contained in R and its boundaries are parallel to $R's$. Let $A(T) = \sum_{p_i \in T} a_i$ and the area of T be $W(T)$. We define the density of T as the ratio of $A(T)$ and $W(T)$. Given R with n points, the goal is to find T such that T covers at least two points and the density of T achieves maximum.*

We show an instance of *maximum-density region problem* in figure 1.3. For all six points in R, the values are the same.

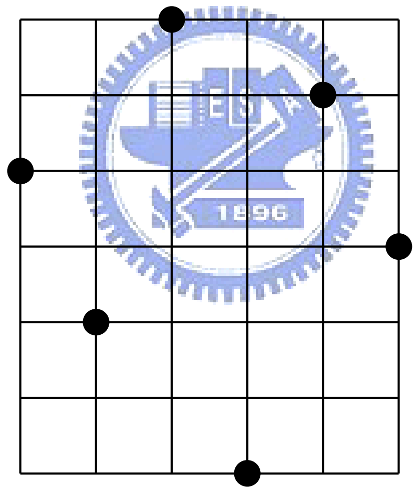

Figure 1.3: An example of maximum-density region problem.

Trees are variation of sequences. When the root of the tree and each internal node have one child only, such a simplified tree can be regarded as a sequence. We state [9] below.

Definition 5 (Maximum-density path problem)**.** *Let $T = (V, E)$ be a rooted, undirected tree with node set V and edge set E. Two functions $\boldsymbol{a} : e \to \mathbb{R}$ and $\boldsymbol{w} : e \to \mathbb{N}$ represent the* *value and weight functions respectively for all $e \in E$. For a path $p = e_1 e_2 \dots e_k$, we denote $A(p) = \sum_{i=1}^{k} a(e_i)$ and $W(p) = \sum_{i=1}^{k} w(e_i)$. We say that $A(p)$ is the value of p and $W(p)$ is the weight of p. The density of a path p is defined as the ratio of $A(p)$ and $W(p)$. Given a tree with n nodes, two functions a and w, a lower bound of weight L and an upper bound of weight U, we would like to find a maximum-density path p with $L \le W(p) \le U$.*

See figure 1.4 as an example of *maximum-density path problem*. We assign a value and a weight for each edge. We denote the value and the weight as a pair.

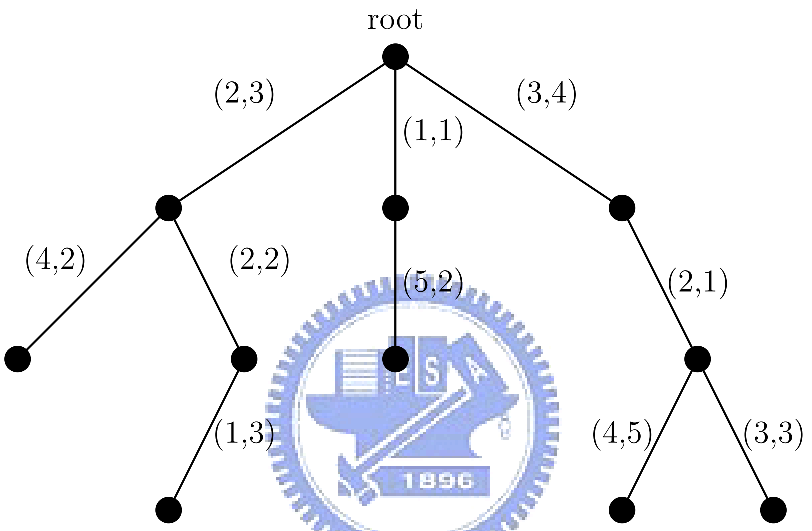

Figure 1.4: An example of maximum-density path problem.

For the segmental density finding problem, Lee et al. give a $O(n \log^2 m)$ time algorithm where $m = \min\{\lfloor \frac{U-L}{w_{min}} \rfloor, n\}$ and $w_{min} = \min_{r=1}^{n} w_r$ in [8]. Luo et al. give a $O(n \log n)$ time result by Minkowski sum in [10]. When δ is close to infinity, the maximum-density segment problem can be solved in $O(n)$ time in [5], [6], [8] and [3]. If the value is uniform in the maximum-density region problem, the problem takes $O(n \log^2 n)$ time by [11], [1], and [4]. In [9], Lin et al. give an algorithm that solves the maximum-density path problem in $O(nL)$ time when the weight of each edge is 1, and Lau et al. [7] present an algorithm that runs in $O(n \log^2 n)$ time for the generalized case.

In the past decade, a number of researches focus on the *density finding problems*. For the network design, we want to upgrade the network by replacing a path with high speed edges. The weight of an edge may represent the building cost and the value may represent the profit. We are also given a budget constraint which limits the weight of the path to be upgraded. Thus, the goal is to find a weight-constrained path, and we hope the profit of the path is as large as possible. [12] and [9] research the network design problem as the maximum-density path problem. Specially, when the weight of each edge is uniform, the time complexity reduces to $O(n \log n)$ in [12] and $O(nL)$ in [9]. In [7], Lau et al. extend the problem to finding a subtree with maximum density, which is *NP-hard*. They also give a $O(nL^2)$-time algorithm when the tree of integer edge weights is given.

In biology, GC content of the DNA sequences is considered. Given a DNA sequence, we want to find a consecutive segment with the highest GC-ratio. The GC-rich problem is formulated as the maximum-density segment problem, and [5] and [6] give optimal algorithms with time complexity $O(n)$. Furthermore, [8] gives a $O(n \log n)$-time algorithm that finds a segment whose density is closest to a given real number δ. When $\delta \to \infty$, it reduces to the maximum-density segment problem. In [3], Bernholt et al. research the *Minkowski sum* and give a $O(n)$-time algorithm that solves the maximum-density segment problem and other subsequence problems in computational biology. Moreover, Luo et al. [10] give a $O(n \log n)$-time algorithm that solves the density-finding problem mentioned above.

In this thesis, we present an algorithm solving the DFR(Rectangular density finding problem, Definition 2) and give a lower bound of time complexity. The remainder of the thesis is organized as follows. Chapter 2 shows the algorithms and proofs. In chapter 3 we discuss the lower bound of DFR. Chapter 4 gives more quasiconvex functions.

For convenience, we use the notations *rectangle* and *matrix* interchangeably. Given two matrices with order $m \times n$, value matrix A and weight matrix W, we use an $m \times n$ rectangle R to describe them. The height and the width of R is m and n respectively. Each unit square in R is called a *block*, and there are mn blocks in R. For each $a_{ij} \in A$ and $w_{ij} \in W$, there is a (a_{ij}, w_{ij})-pair in the block of the ith row and jth column. See the example in figure 1.5.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \qquad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix}$$

(a_{11}, w_{11})	(a_{12}, w_{12})	(a_{13}, w_{13})	(a_{14}, w_{14})
(a_{21}, w_{21})	(a_{22}, w_{22})	(a_{23}, w_{23})	(a_{24}, w_{24})
(a_{31}, w_{31})	(a_{32}, w_{32})	(a_{33}, w_{33})	(a_{34}, w_{34})

Rectangle R

Figure 1.5: The mapping between matrix and rectangle.

Chapter 2

Finding a δ-density Sub-rectangle

In this chapter, we give an algorithm for the special case, finding a constant-height rectangle first. Then, we give a method to solve the general case.

2.1 Constant Height

In the following paragraph, we assume that $m \leq n$ and the height of a sub-rectangle is a constant t, $1 \leq t \leq m$. In order to apply [10], we need all t-by-n rectangles so that we can find the δ-closest segment in $O(n \log n)$ time for each rectangle. We use an algorithm for preprocessing step first. Then, for every t-by-n rectangle, we apply the algorithm in [10] to accomplish our work. We state it below and use it as a subroutine.

Theorem 1 ([10]). *Segmental density finding problem with input size $O(n)$ can be solved in optimal $O(n \log n)$ time. We call this algorithm DenFind.*

Given a matrix P with order $m \times n$, we construct a matrix Q such that each element Q_{ij} represents the sum of $P_{1j}, P_{2j}, \ldots, P_{ij}$. The preprocessing algorithm PA does it. The algorithm is listed in figure 2.1. The preprocessing step will be used in the next algorithm.

Lemma 1. *Algorithm PA takes $O(mn)$ time to compute $Q_{ij} = \sum_{k=1}^{i} P_{kj}$, $1 \leq i \leq m, 1 \leq j \leq n$.*

Proof. Line 1 to 3 cost $O(n)$ time, and line 4 to 8 use $O(mn)$ time. The correctness of Q_{ij} follows by the recursive definition in line 4 to 8. □

Preprocessing Algorithm (PA)

Input: An $m \times n$ matrix P.

Output: An $m \times n$ matrix Q that $Q_{ij} = \sum_{k=1}^{i} P_{kj},\ 1 \le i \le m, 1 \le j \le n$.

```
1: for $j = 1$ to $n$ do
2:     $Q_{0j} := 0$
3: end for
4: for $i = 1$ to $m$ do
5:     for $j = 1$ to $n$ do
6:         $Q_{ij} \leftarrow Q_{i-1,j} + P_{ij}$
7:     end for
8: end for
```

Figure 2.1: Algorithm for Preprocessing

Figure 2.2 is an example of matrices P and Q.

3	-1	-2	5	-2
-4	7	0	-3	6
2	4	-1	-4	3

(a) matrix P

3	-1	-2	5	-2
-1	6	-2	2	4
1	10	-3	-2	7

(b) matrix $Q = PA(P)$

Figure 2.2: Matrices P and $Q = PA(P)$.

Now we state an algorithm for the constant-height case. Given two matrices A and W as input, we calculate $A' = PA(A)$ and $W' = PA(W)$ first. Next, we calculate $A'_{i+t,j} - A'_{ij} = A_{i+1,j} + A_{i+2,j} + \cdots + A_{i+t,j}$ and $W'_{i+t,j} - W'_{ij} = W_{i+1,j} + W_{i+2,j} + \cdots + W_{i+t,j}$ for all j. They can be regarded as a sequence of (value,weight)-pairs. Then, we apply the algorithm *DenFind* to find a δ-density-close segment in the sequence. We iterate the above actions for all feasible i and obtain a set of candidates. Finally, we choose one of them whose density is closest to δ. We show the algorithm in figure 2.3.

Constant-Height Algorithm(CHA)

Input: Two $m \times n$ matrices, value matrix A and weight matrix W, a lower bound L, an upper bound U, a constant height t, a real number δ.

Output: A t-height sub-rectangle whose weight is between L and U and its density is closest to δ.

```
 1: Let $A'$ be the output of PA($A$) and $W'$ of PA($W$).
 2: for $i = 1$ to $m - t + 1$ do
 3:     for $j = 1$ to $n$ do
 4:         $a_j \leftarrow A'_{i+t-1,j} - A'_{i-1,j}$
 5:         $w_j \leftarrow W'_{i+t-1,j} - W'_{i-1,j}$
 6:     end for
 7:     Let sequence $S = \{(a_1, w_1), \ldots, (a_n, w_n)\}$.
 8:     Apply DenFind algorithm on sequence $S$.
 9:     Output the result.
10: end for
11: Among all possible results, choose the one whose density is closest to $\delta$.
```

Figure 2.3: Algorithm for Constant Height

We show an example of the results after executing line 3 to 6 of algorithm CHA. Matrices A and W represent the value matrix and the weight matrix respectively. Furthermore, $A' = PA(A)$ and $W' = PA(W)$. The constant height t is 2, and the pairs in each block are (value,weight)-pairs.

Now we prove the correctness of CHA. The algorithm calculates all t-by-n sequences, and use the algorithm *DenFind* stated in theorem 1. In other words, we examine all possible solutions from the input. Therefore, we ensure that the rectangle whose density is closest to δ would be found after the algorithm stops.

3	-1	-2	5	-2
-4	7	0	-3	6
2	4	-1	-4	3

(a) matrix A

2	1	2	1	1
1	3	4	1	3
1	2	1	2	1

(b) matrix W

3	-1	-2	5	-2
-1	6	-2	2	4
1	10	-3	-2	7

(c) matrix A'

2	1	2	1	1
3	4	6	2	4
4	6	7	4	5

(d) matrix W'

(-1,3)	(6,4)	(-2,6)	(2,2)	(4,4)

(e) sequence of $i = 1$

(-2,2)	(11,5)	(-1,5)	(-7,3)	(9,4)

(f) sequence of $i = 2$

Figure 2.4: The execution of line 3 to 6.

Lemma 2. *Algorithm CHA correctly constructs every t-height rectangle.*

Proof. We show that line 3 to 6 construct a t-height rectangle. From algorithm PA, $Q_{ij} = \sum_{1\le k\le i} P_{kj}$ by recursion. Hence,

$$a_j = A'_{i+t-1,j} - A'_{i-1,j} = (\sum_{1\le k\le i+t-1} A_{kj}) - (\sum_{1\le k\le i-1} A_{kj}) = A_{ij} + \cdots + A_{i+t-1,j},$$

$$w_j = W'_{i+t-1,j} - W'_{i-1,j} = (\sum_{1\le k\le i+t-1} W_{kj}) - (\sum_{1\le k\le i-1} W_{kj}) = W_{ij} + \cdots + W_{i+t-1,j}$$

for all i. Thus, the sequence $\{(a_1, w_1), \ldots, (a_n, w_n)\}$ represents a t-by-n rectangle. $\square$

Then, we consider the time complexity of CHA. We prove that CHA takes at most $O(mn \log n)$ time during the execution.

Lemma 3. *Algorithm CHA takes $O(mn \log n)$ time.*

Proof. Line 1 uses $O(mn)$ time by lemma 1. Line 3 to 6 cost $O(n)$ time. From Theorem 1, line 7 to 9 take $O(n \log n)$ time. The first **for** loop runs at most $(m-t+1)$ times. Therefore, it takes $O(mn \log n)$ time. □

In the last of this section, we give a theorem of DFR with constant-height constraint.

Theorem 2. *DFR with constant height can be solved in $O(mn \log n)$ time.*

Proof. Algorithm CHA solves DFR with any constant height t. The correctness follows by lemma 2. The computing time follows by lemma 3. □

2.2 General Case

Now we discuss the general case of DFR. We use the idea of algorithm PA and algorithm CHA, and modify the latter to reach our goal.

The algorithm GA works as follows. We transpose the matrices A and W if the number of rows are greater than the number of columns. This step saves execution time. Next, for all possible height t, we call the algorithm CHA as a subroutine. Finally, among all candidates, we choose one whose density is closest to δ. The algorithm GA is shown in figure 2.5.

Now we prove the correctness of GA. Like CHA, the algorithm GA calculates all n-width sequences whose height varies from 1 to m. It ensures that every possible sub-rectangle is under consideration. The sub-rectangle whose density is closest to δ will be output when the algorithm *DenFind* executes the corresponding sequence.

Lemma 4. *The rectangle whose density is closest to δ will be output by algorithm GA.*

Proof. The first and second **for** loop ensure that we check every sequence with any height. The remainders then follows by lemma 2. □

General Algorithm(GA)

Input: Two $m \times n$ matrices, value matrix A and weight matrix W, a lower bound L, an upper bound U and a real number δ.

Output: A feasible sub-rectangle that its density is closest to δ.

```
1: if $m > n$ then
2:     $A \leftarrow A^T$
3:     $W \leftarrow W^T$
4: end if
5: for $t = 1$ to $m$ do
6:     Call CHA($A$,$W$,$L$,$U$,$t$,$\delta$).
7: end for
8: Among all possible results, choose the one whose density is closest to $\delta$.
```

Figure 2.5: Algorithm for General Case

Then, we discuss the time consumed in GA. We prove that algorithm GA uses at most $O(m^2 n \log n)$ time when $m \leq n$ and $O(mn^2 \log m)$ when $m > n$.

Lemma 5. *Let $f = \min\{m,n\}$ and $g = \max\{m,n\}$. The algorithm GA takes $O(f^2 g \log g)$ time.*

Proof. The algorithm executes line 1 to 4 if m is greater than n. It ensures that the number of rows in input matrices is always smaller than the number of columns. In line 5, the **for** loop executes f times, and the subroutine CHA takes $O(fg \log g)$ time when called. Therefore, it takes $O(f^2 g \log g)$ time. □

Therefore, we obtain the following theorem.

Theorem 3. *Let $f = \min\{m,n\}$ and $g = \max\{m,n\}$. Then, DFR can be solved in $O(f^2 g \log g)$ time.*

Proof. The correctness and the computing time are immediately followed by lemma 4 and lemma 5. Hence the result. □

To compare the case of one-dimension sequences, we give the following corollary. Note that the number of input is n.

Corollary 1. *Assume that the rank of the input of DFR, value matrix and weight matrix, is $\sqrt{n} \times \sqrt{n}$. Then it costs $O(n^{1.5} \log n)$ time to find the boundary-parallel sub-rectangle whose density is closest to δ.*

We state the special case that $\delta \to \infty$ below. In order to improve the efficiency, we need another faster algorithm that solves the *maximum-density segment problem*. The following theorem describe it.

Theorem 4 ([5], [6], [8], [3])**.** *The maximum-density segment problem can be solved in optimal $O(n)$ time. We call such a algorithm MaxDenFind.*

With the use of theorem 4, we can enhance the efficiency by modifying the algorithms mentioned in figure 2.3 and figure 2.5 above.

Theorem 5. *When $\delta \to \infty$, DFR with constant height can be solved in $O(mn)$ time.*

Proof. We modify the CHA in figure 2.3. In line 8, we replace the *DenFind* algorithm by *MaxDenFind* algorithm. The remainder follows by theorem 2. □

Theorem 6. *When $\delta \to \infty$, DFR can be efficiently solved in $\min\{O(m^2n), O(n^2m)\}$ time.*

Proof. We modify the CHA in figure 2.3. In line 8, we replace the *DenFind* algorithm by *MaxDenFind* algorithm. The remainder follows by theorem 3. □

Like corollary 1, we list the special case that the input is a $\sqrt{n} \times \sqrt{n}$ rectangle, and the number of input is n.

Corollary 2. *Assume that the rank of the input of DFR, value matrix and weight matrix, is $\sqrt{n} \times \sqrt{n}$. Then it costs $O(n^{1.5})$ time to find the boundary-parallel sub-rectangle with maximal density.*

Chapter 3

Lower Bound of the Rectangular Density Finding Problem

We give a lower bound for the DFR by reduction in this chapter. We use the *segmental density finding problem* (SDF for short), which has a lower bound $\Omega(n \log n)$ time.

Theorem 7 ([6], [2]). *Solving segmental density finding problem needs $\Omega(n \log n)$ time in the algebraic decision tree model of computation even when $L = 1$, $U = n$, $\delta = 0$, and the weights are uniform, i.e., $w_i = 1$ for all i.*

Proof. Let $P = \{b_1, b_2, \ldots, b_{n+1}\}$ be the instance of element uniqueness problem. We construct an instance of SDF as follows. Let $S = \{Z_i = (b_i - b_{i+1}, 1) \mid i = 1, \ldots, n\}$, $L = 1$, $U = n$, $\delta = 0$. If $b_i = b_j$ for some $i < j$ in P, then the value and the weight of the segment $(Z_i, Z_{i+1}, \ldots, Z_{j-1})$ are 0 and $j - i$ respectively. That is, the density is 0. On the other hand, if the density of the segment $(Z_i, Z_{i+1}, \ldots, Z_{j-1})$ is 0, then the value of the segment must be 0, i.e., $b_i - b_j = 0$. Hence, $b_i = b_j$. Since solving element uniqueness problem needs $\Omega(n \log n)$ time, and the reduction takes $O(n)$ time, it is obvious that solving SDF needs $\Omega(n \log n)$ time. $\square$

In the following section, we reduce SDF to DFR, and discuss the time complexity.

3.1 Reduction Model

Assume that the input of SDF is a sequence $S = \{Z_1 = (a_1, 1), Z_2 = (a_2, 1), \dots, Z_n = (a_n, 1)\}$, two bounds $L \geq 1$ and $U \leq n$ and a specified density δ. We divide S into some groups,

$G_1 = \{Z_1, Z_2, \dots, Z_U\}$,

$G_2 = \{Z_{U+1}, Z_{U+2}, \dots, Z_{2U}\}$,

$\dots$,

$G_{\lfloor \frac{n}{U} \rfloor} = \{Z_{(\lfloor \frac{n}{U} \rfloor - 1)U+1},\ Z_{(\lfloor \frac{n}{U} \rfloor - 1)U+2}, \dots,\ Z_{\lfloor \frac{n}{U} \rfloor U}\}$,

$G_{\lfloor \frac{n}{U} \rfloor + 1} = \{Z_{\lfloor \frac{n}{U} \rfloor U+1}, Z_{\lfloor \frac{n}{U} \rfloor U+2}, \dots, Z_n\}$.

The number of groups is $\lceil \frac{n}{U} \rceil$. Each group has U elements except for the last group. If n divides U, then the last group is $G_{\frac{n}{U}}$ and $|G_{\frac{n}{U}}| = U$. Else, the last group is $G_{\lfloor \frac{n}{U} \rfloor + 1}$ and $|G_{\lfloor \frac{n}{U} \rfloor + 1}| = n - \lfloor \frac{n}{U} \rfloor$. For the latter case, we insert some insignificant pairs x to $G_{\lfloor \frac{n}{U} \rfloor + 1}$ such that the order of $G_{\lfloor \frac{n}{U} \rfloor + 1}$ is U. For simplicity, we assume that n is a multiple of U and $n = tU$ for some integer t.

$S = \{Z_1, Z_2, \dots, Z_{12}\}$	
$U = 3$	$U = 5$
$G_1 = \{Z_1, Z_2, Z_3\}$	$G_1 = \{Z_1, Z_2, Z_3, Z_4, Z_5\}$
$G_2 = \{Z_4, Z_5, Z_6\}$	$G_2 = \{Z_6, Z_7, Z_8, Z_9, Z_{10}\}$
$G_3 = \{Z_7, Z_8, Z_9\}$	$G_3 = \{Z_{11}, Z_{12}, x, x, x\}$
$G_4 = \{Z_{10}, Z_{11}, Z_{12}\}$	x is insignificant

Figure 3.1: Example of dividing SDF: $|S| = 12$ when $U = 3$ and $U = 5$.

After dividing S into $t = \frac{n}{U}$ groups, we construct a rectangle for DFR according to the following conditions. There will be $2t - 3$ rows and $2U$ columns in the rectangle.

◆ For odd rows $2k - 1$, $k = 1, \dots, t - 1$, consider the groups $G_k = \{Z_{(k-1)U+1}, \dots, Z_{kU}\}$ and $G_{k+1} = \{Z_{kU+1}, \dots, Z_{(k+1)U}\}$. The block $b_{2k-1,j}$ is $Z_{(k-1)U+j}$ for $j = 1, \dots, 2U$.

◆ For even rows $2k$, $k = 1, \dots, t - 2$, the (value,weight)-pair of the block $b_{2k,j}$ is (ϵ, U), the difference of ϵ and δU is large.

Figure 3.2 shows the construction when $|S| = 12$ and $U = 3$.

$(a_1, 1)$	$(a_2, 1)$	$(a_3, 1)$	$(a_4, 1)$	$(a_5, 1)$	$(a_6, 1)$
(ϵ, U)	(ϵ, U)	(ϵ, U)	(ϵ, U)	(ϵ, U)	(ϵ, U)
$(a_4, 1)$	$(a_5, 1)$	$(a_6, 1)$	$(a_7, 1)$	$(a_8, 1)$	$(a_9, 1)$
(ϵ, U)	(ϵ, U)	(ϵ, U)	(ϵ, U)	(ϵ, U)	(ϵ, U)
$(a_7, 1)$	$(a_8, 1)$	$(a_9, 1)$	$(a_{10}, 1)$	$(a_{11}, 1)$	$(a_{12}, 1)$

Figure 3.2: The reduction from SDF to DFR.

We prove the correctness of the reduction. The reduction works because of the weight constraint U. The construction ensures the mapping of SDF and DFR.

Lemma 6. *If there exists a segment with density δ in SDF, then we can find a rectangle whose density is δ in DFR.*

Proof. Assume that the density of the segment $S(i, j)$ is exactly δ, and the weight of $S(i, j) = j - i + 1$ is between L and U. The element Z_i and $Z_j$belong to the $\lceil \frac{i}{U} \rceil$th and $\lceil \frac{j}{U} \rceil$th group respectively. Since $j - i + 1 \leq U$, we have

$$\lceil \frac{j}{U} \rceil - \lceil \frac{i}{U} \rceil \leq (\frac{j}{U} + 1) - \frac{i}{U} = \frac{j-i}{U} + 1 = \frac{j-i+1}{U} + \frac{U-1}{U} \leq 1 + \frac{U-1}{U} < 2,$$

which means that either Z_i and Z_j are in the same group or they belong to two consequtive groups. Consider the row $2\lceil \frac{i}{U} \rceil - 1$ in DFR. According to the reduction rule, the block $b_{2\lceil \frac{i}{U} \rceil - 1, i - (\lceil \frac{i}{U} \rceil - 1)U}$ is $Z_{(\lceil \frac{i}{U} \rceil - 1)U + i - (\lceil \frac{i}{U} \rceil - 1)U}$, which is Z_i exactly. On the other hand, consider the block $b_{2\lceil \frac{i}{U} \rceil - 1, j - (\lceil \frac{i}{U} \rceil - 1)U}$ in the same row. Since $j - (\lceil \frac{i}{U} \rceil - 1)U \leq 2U$, according to the reduction rule, the block is equal to Z_j. Hence, we find a rectangle in DFR whose density is exactly δ. □

Lemma 7. *We can find a segment with density δ in SDF if there exists a rectangle whose density is δ in DFR.*

Proof. We argue that the δ-density rectangle in DFR is of size 1-by-x, $x \leq 2U$ and is located in the odd rows first. By the construction rule, the value and the weight of each block in the even rows are ϵ and U, and $|\epsilon - \delta U|$ is large enough. That is, any block combined with the (ϵ, U)-block will contravene the weight constraint since it exceeds U.

The density of (ϵ, U)-block is $\frac{\epsilon}{U} \neq \delta$, which is far from δ. Thus, the δ-density rectangle is located in the odd rows.

Suppose that the density of a rectangle with the blocks $b_{2k-1,i}, b_{2k-1,i+1}, \ldots, b_{2k-1,j}$ is δ, $1 \leq k \leq t-1, 1 \leq i \leq j \leq 2U, L \leq j-i+1 \leq U$. It implies that the rectangle $R_\delta = \{Z_{(k-1)U+i}, Z_{(k-1)U+i+1}, \ldots, Z_{(k-1)U+j}\}$ is of density δ and its weight satisfies the constraint. Then in SDF, the R_δ is a consecutive sequence $\{(a_{(k-1)U+i}, 1), (a_{(k-1)U+i+1}, 1), \ldots, (a_{(k-1)U+j}, 1)\}$ with density δ exactly, and its weight is $j-i+1$, which is between L and U.

$\square$

From above, we know the correctness of the reduction. We analyze the time complexity of the reduction now.

Assume that the input size of SDF is n, i.e., n pairs of (value,weight)-pair. Then, the size of DFR is $O(n)$ because of the construction rules. From Theorem 7, we know that solving SDF needs $\Omega(n \log n)$ time even the weights are uniform. Therefore, solving DFR with height $O(\frac{n}{U})$ and width $O(U)$ also needs $\Omega(n \log n)$ time. That is, solving DFR with height $O(\frac{mn}{U})$ and width $O(U)$ takes $\Omega(mn \log mn)$ time. We state it as a theorem.

Theorem 8. *Solving DFR with input size $O(mn)$, where height is $O(\frac{mn}{U})$ and width is $O(U)$ needs $\Omega(mn \log mn)$ time.*

Similar to Theorem 3, we give a simplified conclusion of the lower bound.

Theorem 9. *Let $f = \min\{m, n\}$ and $g = \max\{m, n\}$. Then, solving DFR needs $\Omega(fg \log g)$ time.*

Chapter 4

More Quasiconvex Functions

Let P, Q be the sets of points on a 2D plane. Define the Minkowski sum of two sets P and Q be $P \bigoplus Q = \{p + q | p \in P, q \in Q\}$. Let K be a set of the constraints $k_i : a_i x + b_i y \geq c_i$, $a_i, b_i, c_i \in \mathbb{R}, i = 1, \ldots, r$. The constrained Minkowski sum of two sets P, Q and a constraint set K is $(P \bigoplus Q)_K = \{(x, y) \in P \bigoplus Q \mid a_i x + b_i y \geq c_i, i = 1, \ldots, r\}$. A function $f : D \to \mathbb{R}$ is called *quasiconvex* if for all points $s_1, s_2 \in D$ and all λ, $0 \leq \lambda \leq 1$, we have

$$f(\lambda s_1 + (1 - \lambda) s_2) \leq \max\{f(s_1), f(s_2)\}.$$

Figure 4.1 shows a quasiconvex function.

The motivation for studying constrainted Minkowski sum is that many subsequence problems from computational biology can be solved by maximizing a quasiconvex function over the points of a constrained Minkowski sum. In [3], the authors indicate some quasiconvex functions such as $f(x, y) = \frac{y}{x}$, $\frac{y}{\sqrt{x}}$, $\frac{|y|}{x}$ and y, for $x > 0$. Moreover, $ax + by$ and $\frac{by}{ax}$ are also quasiconvex functions introduced in [10]. In this chapter, we give more quasiconvex functions, $\frac{y^n}{x^n}$, for all $n \in \mathbb{N}$.

4.1 Objective Function $f(x, y) = \frac{y^n}{x^n}$

First, we show that the objective function $f(x, y) = \frac{y^n}{x^n}$ $(x > 0)$ is quasiconvex. Mathematical induction is used for the proofs. We prove the first statement of $n = 1$ and $n = 2$ by the definition of quasiconvex.

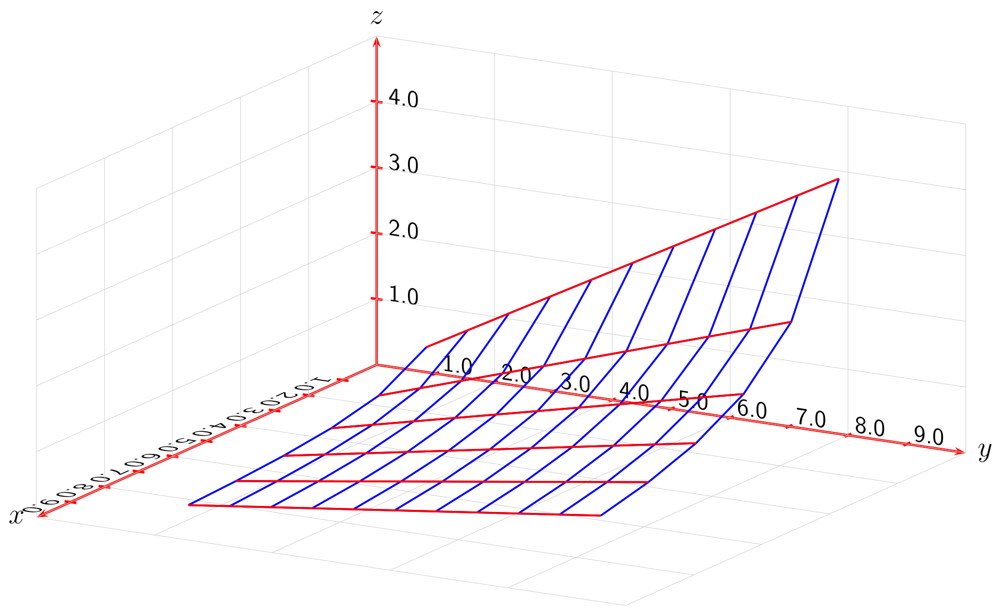

Figure 4.1: Quasiconvex function $f(x,y) = \dfrac{y}{x}$.

Lemma 8. *$f(x,y) = \frac{y}{x}$ is quasiconvex.*

Proof. Let $s_1 = (x_1, y_1)$, $s_2 = (x_2, y_2)$, $s = \lambda s_1 + (1-\lambda)s_2$. Without loss of generality, we assume that $f(s_1) \le f(s_2)$, i.e., $\dfrac{y_1}{x_1} \le \dfrac{y_2}{x_2}$ or $y_1x_2 \le y_2x_1$.

Then,

$$f(s) = f(\lambda s_1 + (1-\lambda)s_2) = f((\lambda x_1 + (1-\lambda)x_2, \lambda y_1 + (1-\lambda)y_2)$$

$$= \frac{\lambda y_1 + (1-\lambda)y_2}{\lambda x_1 + (1-\lambda)x_2}$$

$$= \frac{\lambda y_1x_2 + (1-\lambda)y_2x_2}{x_2(\lambda x_1 + (1-\lambda)x_2)}$$

$$\le \frac{\lambda y_2x_1 + (1-\lambda)y_2x_2}{x_2(\lambda x_1 + (1-\lambda)x_2)}$$

$$= \frac{y_2((\lambda x_1 + (1-\lambda)x_2))}{x_2(\lambda x_1 + (1-\lambda)x_2)}$$

$$= \frac{y_2}{x_2}$$

$$= \max\{\frac{y_1}{x_1}, \frac{y_2}{x_2}\} = \max\{f(s_1), f(s_2)\},\ Q.E.D.$$ □

Lemma 9. *$f(x,y) = \frac{y^2}{x^2}$ is quasiconvex.*

Proof. Let $s_1 = (x_1, y_1)$, $s_2 = (x_2, y_2)$, $s = \lambda s_1 + (1-\lambda)s_2$. Without loss of generality, we assume that $f(s_1) \le f(s_2)$, i.e., $\frac{y_1^2}{x_1^2} \le \frac{y_2^2}{x_2^2}$.
Then,

$$\begin{aligned}
&f(s) = f(\lambda s_1 + (1-\lambda)s_2) \\
&= \frac{(\lambda y_1 + (1-\lambda)y_2)^2}{(\lambda x_1 + (1-\lambda)x_2)^2} \\
&= \frac{x_2^2(\lambda^2 y_1^2 + 2\lambda(1-\lambda)y_1y_2 + (1-\lambda)^2 y_2^2)}{x_2^2(\lambda x_1 + (1-\lambda)x_2)^2} \\
&\le \frac{\lambda^2 y_2^2 x_1^2 + 2\lambda(1-\lambda)y_1y_2x_2^2 + (1-\lambda)^2 y_2^2 x_2^2}{x_2^2(\lambda x_1 + (1-\lambda)x_2)^2} \\
&\le \frac{\lambda^2 y_2^2 x_1^2 + 2\lambda(1-\lambda)y_2^2x_1x_2 + (1-\lambda)^2 y_2^2 x_2^2}{x_2^2(\lambda x_1 + (1-\lambda)x_2)^2} \\
&= \frac{y_2^2}{x_2^2} \times \frac{(\lambda x_1 + (1-\lambda)x_2)^2}{(\lambda x_1 + (1-\lambda)x_2)^2} \\
&= \frac{y_2^2}{x_2^2} \\
&= \max\{\frac{y_1^2}{x_1^2}, \frac{y_2^2}{x_2^2}\} = \max\{f(s_1), f(s_2)\},\ Q.E.D.
\end{aligned}$$

□

Then we prove the induction steps. We discuss two cases when n is even and n is odd respectively. The proofs follow the definition of the quasiconvex functions mentioned above.

Lemma 10. *If* $f(x,y) = \frac{y^{2k}}{x^{2k}}$ *is quasiconvex, then so is* $g(x,y) = \frac{y^{2k+2}}{x^{2k+2}}, k \in \mathbb{N}$.

Proof. Let $s_1 = (x_1, y_1)$, $s_2 = (x_2, y_2)$, $s = \lambda s_1 + (1-\lambda)s_2$. Without loss of generality, we assume that $f(s_1) \le f(s_2)$, i.e., $\frac{y_1^{2k}}{x_1^{2k}} \le \frac{y_2^{2k}}{x_2^{2k}}$. Then, $\left[\left(\frac{y_1}{x_1}\right)^2\right]^k \le \left[\left(\frac{y_2}{x_2}\right)^2\right]^k$ implies $\left[\left(\frac{y_1}{x_1}\right)^2\right]^{k+1} \le \left[\left(\frac{y_2}{x_2}\right)^2\right]^{k+1}$ since $\left(\frac{y_1}{x_1}\right)^2 \ge 0$ and $\left(\frac{y_2}{x_2}\right)^2 \ge 0$.

Consider $g(s)$,

$$\begin{aligned}
&g(s) = g(\lambda s_1 + (1-\lambda)s_2) \\
&= \frac{(\lambda y_1 + (1-\lambda)y_2)^{2k}}{(\lambda x_1 + (1-\lambda)x_2)^{2k}} \times \frac{(\lambda y_1 + (1-\lambda)y_2)^2}{(\lambda x_1 + (1-\lambda)x_2)^2} \\
&\le \frac{y_2^{2k}}{x_2^{2k}} \times \frac{y_2^2}{x_2^2} \text{ (by lemma 9)} \\
&= \frac{y_2^{2k+2}}{x_2^{2k+2}} = \max\{g(s_1), g(s_2)\},\ Q.E.D.
\end{aligned}$$

□

Lemma 11. *If* $f(x,y) = \dfrac{y^{2k-1}}{x^{2k-1}}$ *is quasiconvex, then so is* $g(x,y) = \dfrac{y^{2k+1}}{x^{2k+1}}, k \in \mathbb{N}$.

Proof. Let $s_1 = (x_1, y_1),\ s_2 = (x_2, y_2),\ s = \lambda s_1 + (1-\lambda)s_2$. Without loss of generality, we assume that $f(s_1) \le f(s_2)$, i.e., $\dfrac{y_1^{2k-1}}{x_1^{2k-1}} \le \dfrac{y_2^{2k-1}}{x_2^{2k-1}}$.

Consider $g(s) = g(\lambda s_1 + (1-\lambda)s_2) = \dfrac{(\lambda y_1 + (1-\lambda)y_2)^{2k-1}}{(\lambda x_1 + (1-\lambda)x_2)^{2k-1}} \times \dfrac{(\lambda y_1 + (1-\lambda)y_2)^2}{(\lambda x_1 + (1-\lambda)x_2)^2}$. Assume that $a = \dfrac{\lambda y_1 + (1-\lambda)y_2}{\lambda x_1 + (1-\lambda)x_2}$, $b = \dfrac{y_2}{x_2}$, $a^{2k-1} \le b^{2k-1}$. There are three cases of a and b, (1)$a < 0, b < 0$, (2)$a < 0, b \ge 0$ and (3)$a \ge 0, b \ge 0$. We discuss these cases below.

Case (1), $a < 0, b < 0$ implies $a^2 \ge b^2 > 0$. Thus, $a^{2k+1} = a^{2k-1} \times a^2 \le b^{2k-1} \times b^2 = b^{2k+1}$.

Case (2), $a < 0, b \ge 0$ implies $a^{2k+1} < 0 \le b^{2k+1}$.

Case (3), $a \ge 0, b \ge 0$ implies $0 \le a^2 \le b^2$. Thus, $a^{2k+1} = a^{2k-1} \times a^2 \le b^{2k-1} \times b^2 = b^{2k+1}$.

Therefore,

$$g(s) = \frac{(\lambda y_1 + (1-\lambda)y_2)^{2k+1}}{(\lambda x_1 + (1-\lambda)x_2)^{2k+1}} = a^{2k+1} \le b^{2k+1} = \frac{y_2^{2k+1}}{x_2^{2k+1}} = \max\{g(s_1), g(s_2)\},\ Q.E.D. \qquad \square$$

Therefore, we obtain the following theorem for $f(x,y) = \dfrac{y^n}{x^n}$.

Proposition 1. *The objective function* $f(x,y) = \dfrac{y^n}{x^n}$ *is quasiconvex.*

Proof. By mathematical induction, lemma 8, lemma 9, lemma 10 and lemma 11, we obtain the result. $\square$

Chapter 5

Efficiency Comparison over two algorithms of Maximum-density Finding Problem

In chapter 2, we mention that the maximum-density segment problem can be solved in linear time when the input size is $O(n)$. [5] and [3] both give a $O(n)$ time algorithm. In the former, the algorithm executes according to density comparison between segments. In the latter, it reduces to geometric problem and use Minkowski sum to solve it. We compare the time efficiency in this chapter.

For convenience, we call the algorithm proposed by [5] A1, the algorithm proposed by [3] A2. Besides, an algorithm called A3, which uses two for loops and takes $O(n^2)$ time to solve the maximum-density finding problem, would be compared with A1 and A2. We show the algorithm below.

We take advantage of the function clock() for the time measurement. For each algorithm, we execute it 100 times for each input size N ($N = 5000, 10000, \ldots, 50000$), average the consuming time and plot them on the same diagram. Figure 5.2 shows the result.

According the data, the average executing time of A2 is about 8 times to A1. Though both take linear time to solve the maximum-density finding problem in time complexity, A2 seems to consume more time than A1. The reason that leads to this result is probably the computation of the convex hull.

Pseudocode of A3

Input: A sequence of pairs $\{(a_1, w_1),(a_2, w_2),\ldots,(a_n, w_n)\}$, two positive numbers L and U.

Output: A segment whose weight is between L and U achieves the maximum density.

```
 1: MaxDensity $\leftarrow -\infty$
 2: for $i = 1$ to $n$ do
 3:     for $j = i$ to $n$ do
 4:         Calculate $A_{ij} = \Sigma_{k=i}^{j} a_k$ and $W_{ij} = \Sigma_{k=i}^{j} w_k$.
 5:         if $L \leq W_{ij} \leq U$ then
 6:             Calculate the density $D_{ij} = \frac{A_{ij}}{W_{ij}}$.
 7:         end if
 8:         if $D_{ij} >$ MaxDensity then
 9:             MaxDensity $\leftarrow D_{ij}$
10:         end if
11:     end for
12: end for
13: Output the index $i$ and $j$.
```

Figure 5.1: Pseudocode of A3

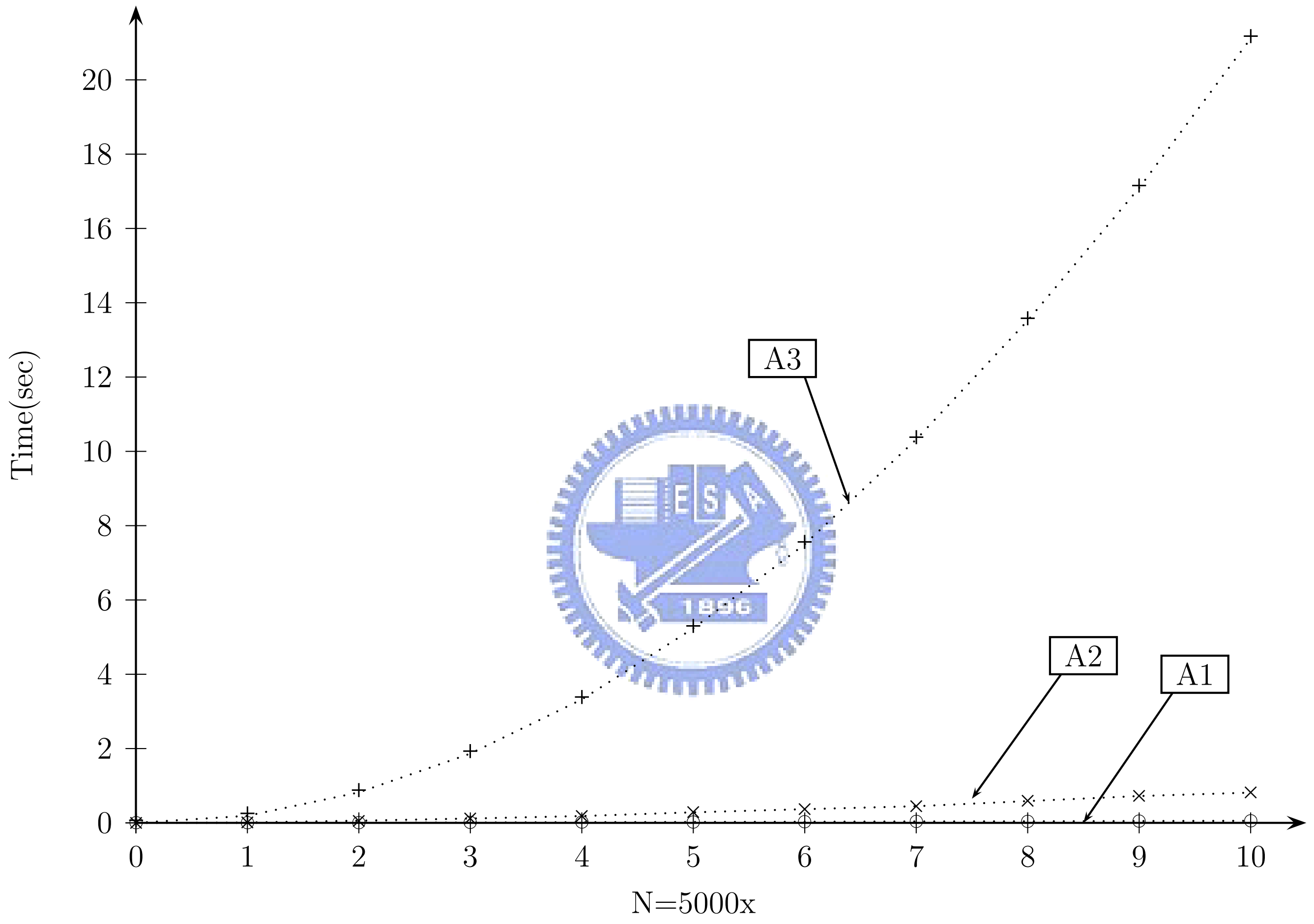

Figure 5.2: Comparison Diagram of A1(Chung [5]), A2(Bernholt [3]) and A3.

Chapter 6

Conclusion

We briefly state the conclusion in this paper.

In the rectangular density finding problem, we assume that the order of input matrices is m-by-n. Let $f = \min\{m, n\}$ and $g = \max\{m, n\}$. We present an algorithm solving rectangular density-finding problem in $O(f^2 g \log g)$ time. We also give a lower bound of $\Omega(fg \log g)$ time. If the specified density approaches infinity, we decrease the time to $O(f^2 g)$.

	Rectangular density finding problem
maximum density	$O(f^2 g)$
any density	$O(f^2 g \log g)$ and $\Omega(fg \log g)$

Table 6.1: Table of results.

For DFR, whether the upper bound can be decreased or the lower bound should be increased is still unknown. We leave it as an open problem.

Bibliography

[1] Alok Aggarwal and Maria Klawe. Applications of generalized matrix searching to geometric algorithms. *Discrete Applied Mathematics*, 27:3–23, 1990.

[2] Michael Ben-Or. Lower bounds for algebraic computation trees. *Annual ACM Symposium on Theory of Computing Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 80–86, 1983.

[3] Thorsten Bernholt, Friedrich Eisenbrand, and Thomas Hofmeister. A geometric framework for solving subsequence problems in computational biology efficiently. *In Proceedings of the 23rd ACM Symposium on Computational Geometry*, pages 310–318, 2007.

[4] B. Chazelle, R.L.Drysdale, and D.T.Lee. Computing the largest empty rectangle. *SIAM Journal of Computing*, 15(1):300–315, 1986.

[5] Kai-Min Chung and Huseh-I Lu. An optimal algorithm for the maximum-density segment problem. *SIAM J. COMPUT.*, 34(2):373–387, 2004.

[6] Michael H. Goldwasser, Ming-Yang Kao, and Hsueh-I Lu. Linear-time algorithms for computing maximum-density sequence segments with bioinformatics applications. *Journal of Computer and System Sciences*, 70:128–144, 2005.

[7] Hoong Chuin Lau, Trung Hieu Ngo, and Bao Nguyen Nguyen. Finding a length-constrained maximum-sum or maximum-density subtree and its application to logistics. *Discrete Optimization*, 3:385–391, 2006.

[8] D.T. Lee, Tien-Ching Lin, and Hsueh-I Lu. Fast algorithms for the density finding problem. *Algorithmica*, 2007.

[9] Rung-Ren Lin, Wen-Hhiung Kuo, and Kun-Mao Chao. Finding a length-constrained maximum-density path in a tree. *Journal of Combinatorial Optimization*, 9:147–156, 2005.

[10] Cheng-Wei Luo, Peng-An Chen, and Hsiao-Fei Liu. Constrained minkowski sum selection and finding. *Proceedings of the 25th Workshop on Combinatorial Mathematics and Computation Theory*, 2008.

[11] Subhashis Majumder and Bhargab B. Bhattacharya. On the density and discrepancy of a 2D point set with applications to thermal analysis of VLSI chips. *Information Processing Letters*, 107:177–182, 2008.

[12] Bang Ye Wu, Kun-Mao Chao, and Chuan Yi Tang. An efficient algorithm for the length-constrained heaviest path problem on a tree. *Information Processing Letters*, 69:63–67, 1999.