5-2018

# Approximate Set Union via Approximate Randomization

Pengfei Gu
*The University of Texas Rio Grande Valley*

# APPROXIMATE SET UNION VIA APPROXIMATE RANDOMIZATION

A Thesis

by

PENGFEI GU

Submitted to the Graduate College of
The University of Texas Rio Grande Valley
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2018

Major Subject: Computer Science

# APPROXIMATE SET UNION VIA APPROXIMATE RANDOMIZATION

A Thesis
by
PENGFEI GU

COMMITTEE MEMBERS

Dr. Bin Fu
Chair of Committee

Dr. Zhixiang Chen
Committee Member

Dr. Robert Schweller
Committee Member

May 2018

# ABSTRACT

Gu, Pengfei, Approximate Set Union via Approximate Randomization. Master of Science (MS), May, 2018, 70 pp., 2 figures, 26 references, 40 titles.

We develop an randomized approximation algorithm for the size of set union problem $|A_1 \cup A_2 \cup ... \cup A_m|$, which given a list of sets $A_1, ..., A_m$ with approximate set size $m_i$ for $A_i$ with $m_i \in ((1-\beta_L)|A_i|, (1+\beta_R)|A_i|)$, and biased random generators with $\text{Prob}(x = \text{RandomElement}(A_i)) \in \left[\frac{1-\alpha_L}{|A_i|}, \frac{1+\alpha_R}{|A_i|}\right]$ for each input set $A_i$ and element $x \in A_i$, where $i = 1, 2, ..., m$. The approximation ratio for $|A_1 \cup A_2 \cup ... \cup A_m|$ is in the range $[(1-\varepsilon)(1-\alpha_L)(1-\beta_L), (1+\varepsilon)(1+\alpha_R)(1+\beta_R)]$ for any $\varepsilon \in (0,1)$, where $\alpha_L, \alpha_R, \beta_L, \beta_R \in (0,1)$. The complexity of the algorithm is measured by both time complexity and round complexity. The algorithm is allowed to make multiple membership queries and get random elements from the input sets in one round. Our algorithm makes adaptive accesses to input sets with multiple rounds. Our algorithm gives an approximation scheme with $\text{O}(m \cdot (\log m)^{\text{O}(1)})$ running time and $\text{O}(\log m)$ rounds, where $m$ is the number of sets. We prove that our algorithm runs sublinear in time under certain condition that each element in $A_1 \cup A_2 \cup ... \cup A_m$ belongs to $m^a$ for any fixed $a > 0$. A $\mathcal{O}\left(r(r+l|\lambda|)^3 l^3 d^4\right)$ running time dynamic programming algorithm is proposed to deal with an interesting problem in number theory area that is to count the number of lattice points in a $d-$dimensional ball $B_d(r,p,d)$ of radius $r$ with center at $p \in D(\lambda,d,l)$, where $D(\lambda,d,l) = \{(x_1,\cdots,x_d) : (x_1,\cdots,x_d)$ with $x_k = i_k + j_k\lambda$ for an integer $j_k \in [-l,\ l]$, and another arbitrary integer $i_k$ for $k = 1,2,...,d\}$. We prove that it is #P-hard to count the number of lattice points in a set of balls, and we also show that there is no polynomial time algorithm to approximate the number of lattice points in the intersection of $n$-dimensional balls unless P=NP.

# DEDICATION

I dedicate this thesis to my academic advisor, Dr. Bin Fu, for his full support, expert guidance, and persistent encouragement. Without his incredible patience and help, this thesis would not have been possible.

# ACKNOWLEDGMENTS

First, I would like to thank all those who helped me when I am working on this thesis. I would like to express the deepest appreciation to my academic advisor, Dr. Bin Fu, who has been a constant source of knowledge and inspiration. Without his full support, patient guidance and instructive advice through my study and research, this thesis work could not have reached its present form. In addition, I express my appreciation to Dr. Zhixiang Chen and Dr. Robert Schweller for having served on my comittee. Then I highly thank department of Computer Science to provide me a position as GTA. Finally, I want to thanks my classmates, all the people in department of Computer Science, and my family.

## TABLE OF CONTENTS

# CHAPTER I

# INTRODUCTION

Computing the cardinality of set union is a basic algorithmic problem that has a simple and natural definition. It is related to the following problem: given a list of sets $A_1,...,A_m$ with set size $|A_i|$, and random generators RandomElement$(A_i)$ for each input set $A_i$, where $1 = 1,2,...,m$, compute $|A_1 \cup A_2 \cup ... \cup A_m|$. This problem is #P-hard if each set is 0,1-lattice points in a high dimensional cube [35]. Karp, Luby, and Madras [29] developed a $(1+\varepsilon)$ randomized approximation algorithm to improve the runnning time for approximating the number of distinct elements in the union $A_1 \cup \cdots \cup A_m$ to linear O$((1+\varepsilon)m/\varepsilon^2)$ time. Their algorithm is based on the input that provides the exact size of each set and a uniform random element generator of each set. Bringmann and Friedrich [8] applied Karp, Luby, and Madras' algorithm in deriving approximate algorithm for high dimensional geometric object with uniform random sampling. They also proved that it is #P-hard to compute the volume of the intersection of high dimensional boxes, and showed that there is no polynomial time $2^{d^{1-\varepsilon}}$-approximation unless NP=BPP. In the algorithms mentioned above, some of them were based on random sampling, and some of them provided exact set sizes when approximating the cardinalities of multisets of data and some of them dealt with two multiple sets. However, in realty, it is really hard to give an uniform sampling or exact set size especially when deal with high dimensional problems.

A similar problem has been studied in the streaming model: given a list of elements with multiplicity, count the number of distinct items in the list. This problem has a more general format to compute frequency moments $F_k = \sum_{i=1}^{m} n_i^k$, where $n_i$ denotes the number of occurrences of $i$ in the sequence. This problem has received a lot of attention in the field of streaming algorithms [2, 4, 5, 7, 14, 15, 18, 19, 20, 21, 25, 28].

**Motivation:** The existing approximate set union algorithm [29] needs each input set has a uniform random generator. In order to have approximate set union algorithm with broad application, it is essential to have algorithm with biased random generator for each input set, and see how approximation ratio depends on the bias. In this paper, we propose a randomized approximation algorithm to approximate the size of set union problem by extending the model used in [29]. In order to show why approximate randomization method is useful, we generalize the algorithm that was designed by Karp, Luby, and Madras [29] to an approximate randomization algorithm. A natural problem that counting of lattice points in d-dimensional ball is discussed to support the useful of approximate randomization algorithm. In our algorithm, each input set $A_i$ is a black box that can provide its size $|A_i|$, generate a random element RandomElement$(A_i)$ of $A_i$, and answer the membership query $(x \in A_i?)$ in $O(1)$ time. Our algorithm can handle input sets that can generate random elements with bias with $\text{Prob}(x = \text{RandomElement}(A_i)) \in \left[\frac{1-\alpha_L}{|A_i|}, \frac{1+\alpha_R}{|A_i|}\right]$ for each input set $A_i$ and approximate set size $m_i$ for $A_i$ with $m_i \in [(1-\beta_L)|A_i|, (1+\beta_R)|A_i|]$.

As the communication complexity is becoming important in distributed environment, data transmission among variant machines may be more time consuming than the computation inside a single machine. Our algorithm complexity is also measured by the number of rounds. The algorithm is allowed to make multiple membership queries and get random elements from the input sets in one round. Our algorithm makes adaptive accesses to input sets with multiple rounds. The round complexity is related a distributed computing complexity if input sets are stored in a distributed environment, and the number of rounds indicates the complexity of interactions between a central server, which runs the algorithm to approximate the size of set union, and clients, which save one set each.

Computation via bounded queries to another set has been well studied in the field of structural complexity theory. Polynomial time truth table reduction has a parallel way to access oracle with all queries to be provided in one round [9]. Polynomial time Turing reduction has a sequential way to access oracle by providing a query and receiving an answer in one round [12]. The constant-round truth table reduction (for example, see [16]) is between truth table reduction, and

Turing reduction. Our algorithm is similar to a bounded round truth table reduction to input sets to approximate the size set union. Karp, Luby, and Madras [29]'s algorithm runs like a Turing reduction which has the number of adaptive queries proportional to the time.

We design approximation scheme for the number of lattice points in a $d$-dimensional ball with its center at $D(\lambda,d,l)$, where $D(\lambda,d,l)$ to be the set points $p_d = (x_1,\cdots,x_d)$ with $x_i = i + j\lambda$ for an integer $j \in [-l,l]$, another arbitrary integer $i$, and an arbitrary real number $l$. It returns an approximation in the range $[(1-\beta)C(r,p,d),(1+\beta)C(r,p,d)]$ in a time $poly\left(d,\frac{1}{\beta},|l|,|\lambda|\right)$, where $C(r,p,d)$ is the number of lattice points in a $d$-dimensional ball with radius $r$ and center $p \in D(\lambda,d,l)$. We also show how to generate a random lattice point in a $d$-dimensional ball with its center at $D(\lambda,d,l)$. It generates each lattice point inside the ball with a probability in $\Big[\frac{1-\alpha}{C(r,p,d)}$,

$1+\frac{\alpha}{C(r,p,d)}$ in a time $poly\left(d,\frac{1}{\alpha},|l|,|\lambda|,\log r\right)$, where the $d$-dimensional ball has radius $r$ and center $p \in D(\lambda,d,l)$. Without the condition that a ball center is inside $D(\lambda,d,l)$, counting the number of lattice points in a ball may have time time complexity that depends on dimension number $d$ exponentially even the radius is as small as $d$. Counting the number of lattice points inside a four dimensional ball efficiently implies an efficient algorithm to factorize the product of two prime numbers ($n = pq$) as $C(\sqrt{n},(0,...,0),4) - C(\sqrt{n-1},(0,...,0),4) = 8(1+pA+q+n)$ (see [3, 27]). Therefore, a fast exact counting lattice points inside a four dimensional ball implies a fast algorithm to crack RSA public key system.

This gives a natural example to apply our approximation scheme to the number of lattice points in a list of balls. We prove that it is #P-hard to count the number of lattice points in a set of balls, and we also show that there is no polynomial time algorithm to approximate the number of lattice points in the intersection n-dimensional balls unless P=NP. We found that it is a elusive problem to develop an $poly\left(d,\frac{1}{\varepsilon}\right)$ time $(1+\varepsilon)$-approximation algorithm for the number of lattice points of $d$-dimensional ball with a small radius. We are able to handle the case with ball centers in $D(\lambda,d,l)$, which can approximate an arbitrary center by adjusting parameters $\lambda$ and $l$. This is our main technical contributions about lattice points in a high dimensional ball.

It is a classical problem in analytic number theory for counting the number of lattice points in d-dimensional ball, and has been studied in a series of articles [1, 6, 10, 11, 13, 22, 23, 26, 30, 31, 33, 34, 36, 37, 38, 39, 40] in the field of number theory. Researchers are interested in both upper bounds and lower bounds for the error term $E_d(r) = N_d(r) - \pi^{\frac{d}{2}}\Gamma(\frac{1}{2}d+1)^{-1}r^d$, where $N_d(r) = \#\{x \in \mathbb{Z}^d : |x| \leq r\}$ is the number of lattice points inside a sphere of radius $r$ centered at the origin and $\pi^{\frac{d}{2}}\Gamma(\frac{1}{2}d+1)^{-1}r^d$ (where $\Gamma(.)$ is Leonhard Gamma Function) is the volume of a $d-dimensional$ sphere of radius $r$. When $d = 2$, the problem is called “Gauss Circle Problem"; Gauss proved that $E_2(r) \leq r$. Gauss’s bound was improved in a papers [13, 22, 26]. Walfisz[39] showed that $E_d(r) = \Omega_{\pm}(r^{d-2})$ and $E_d(r) \leq r^{d-2}$, where $f(x) = \Omega_+(F(x))(f(x) = \Omega_-(F(x)))$ as $x \rightarrow \infty$ if there exist a sequence $\{x_n\} \rightarrow \infty$ and a positive number $C$, such that for all $n \geq 1$, $f(x_n) > C|F(x_n)|$ $(f(x_n) < -C|F(x_n)|)$. Most of the above results focus on the ball centered at the origin, and few papers worked on variable centers but also consider fixed dimensions and radii going to infinity[6, 10, 36, 40].

**Our Contributions:** We have the following contributions to approximate the size of set union. 1. It has constant number of rounds to access the input sets. This reduces an important complexity in a distributed environment where each set stays a different machine. It is in contrast to the existing algorithm that needs $\Omega(m)$ rounds in the worst case. 2. It handles the approximate input set sizes and biased random sources. The existing algorithms assume uniform random source from each set. Our approximation ratio depends on the approximation ratio for the input set sizes and bias of random generator of each input set. The approximate ratio for $|A_1 \cup A_2 \cup \cdots \cup A_m|$ is controlled in the range in $[(1-\varepsilon)(1-\alpha_L)(1-\beta_L),(1+\varepsilon)(1+\alpha_R)(1+\beta_R)]$ for any $\varepsilon \in (0,1)$, where $\alpha_L, \alpha_R, \beta_L, \beta_R \in (0,1)$. 3. It runs in sublinear time when each element belongs to at least $m^a$ sets for any fixed $a > 0$. We have not seen any sublinear results about this problem. 4. We show a tradeoff between the number of rounds, and the time complexity. It takes a $\log m$ rounds with time complexity $\mathrm{O}\left(m(\log m)^{O(1)}\right)$, and takes $\mathrm{O}\left(\frac{1}{\xi}\right)$ rounds, with a time complexity $\mathrm{O}\left(m^{1+\xi}\right)$. We still maintain the time complexity nearly linear time in the classical model. Our algorithm is based on a new approach that is different from that in [29]. 5. We identify two additional parameters $z_{min}$ and $z_{max}$ that affect both the complexity of rounds and time, where $z_{min}$ is the least number of sets that an element belongs to, and $z_{max}$ is the largest number of sets that an element belongs to.

Our algorithm developed in the randomized model only accesses a small number of elements from the input sets. The algorithm developed in the streaming model algorithm accesses all the elements from the input sets. Therefore, our algorithm is incomparable with the results in the streaming model [2, 4, 5, 7, 14, 15, 18, 19, 20, 21, 25, 28].

The rest of paper is organized as follows. In Chapter 2, we define the computational model and complexity. Chapter 3 presents some theorems that play an important role in accuracy analysis. In Chapter 4, we give a randomized approximation algorithm to approximate the size of set union problem; time complexity and round complexity also analysis in Chapter 4. Chapter 5 discusses a natural problem that counting of lattice points in high dimensional balls to support the useful of approximation randomized algorithm. A application of high dimensional balls in Maximal Coverage gives in Chapter 6. In Chapter 7, we generalize the algorithm that was designed by Karp, Luby, and Madras [29] to an approximate randomization algorithm. Chapter 8 summaries the conclusions.

# CHAPTER II

# MODEL OF RANDOMIZATION

In this section, we show our model of computation, and the definition of complexity. Assume that $A_1$ and $A_2$ are two sets. Their union $A_1 \cup A_2$ contains the elements in either $A_1$ or $A_2$. Define $A_2 - A_1$ to be the set of elements in $A_2$, but not in $A_1$. Define their intersection $A_1 \cap A_2$ to be the set of elements in both $A_1$ and $A_2$. For example, $A_1 = \{3, 5\}$ and $A_2 = \{1, 3, 7\}$, then $A_1 \cup A_2 = \{1, 3, 5, 7\}$, $A_2 - A_1 = \{1, 7\}$, and $A_1 \cap A_2 = \{3\}$. For a finite set $A$, define $|A|$ to be the number of elements in $A$. A real number $s$ is an $(1+\varepsilon)$-approximation for $|A|$ if $(1-\varepsilon)|A| \le s \le (1+\varepsilon)|A|$. For a real number $x$, let $\lceil x \rceil$ be the least integer $y \ge x$, $\lfloor x \rfloor$ be the largest integer $z \le x$ and $[x]$ be the integer part of $x$. For examples, $\lceil 3.3 \rceil = 4$, $\lfloor 3.2 \rfloor = 3$, $[3.2] = 3$ and $[3.8] = 3$. Let $\mathbb{N} = \{0, 1, 2, \cdots\}$ be the set of nonnegative integers, $\mathbb{R} = (-\infty, +\infty)$ be the set of all real numbers, $\mathbb{R}^+ = [0, +\infty)$ be the set of all nonnegative real numbers and $\mathbb{Z}$ be the set of all integer numbers.

## 2.1 Model of Randomization

**Definition 2.1.1.** *Let A be a set of elements.*

- *A $\alpha$-biased random generator for set A is a generator that each element in A is generated with probability in the range* $\left[\frac{1-\alpha}{|A|}, \frac{1+\alpha}{|A|}\right]$.
- *A $(\alpha_L, \alpha_R)$-biased random generator for set A is a generator that each element in A is generated with probability in the range* $\left[\frac{1-\alpha_L}{|A|}, \frac{1+\alpha_R}{|A|}\right]$.

**Definition 2.1.2.** *Let L be a list of sets $A_1, A_2, \cdots, A_m$ such that each supports the following operations:*

- *the size of $A_i$ has an approximation $m_i \in [(1-\beta_L)|A_i|, (1+\beta_R)|A_i|]$ for $i = 1, 2, \cdots, m$.*

*Both* $M = \sum_{i=1}^{m} m_i$ *and m are part of the input,*

- *function* RandomElement$(A_i)$ *returns a* $(\alpha_L, \alpha_R)$*-biased approximate random element* $x$ *from* $A_i$ *for* $i = 1, 2, \cdots, m$*,*

- *function query*$(x, A_i)$ *function returns* 1 *if* $x \in A_i$*, and* 0 *otherwise.*

**Definition 2.1.3.** *For a list L of sets* $A_1, A_2, \cdots, A_m$ *and real numbers* $\alpha_L, \alpha_R, \beta_L, \beta_R \in [0, 1)$*, it is called* $((\alpha_L, \alpha_R), (\beta_L, \beta_R))$*-list if each set* $A_i$ *is associated with a number* $s_i$ *with* $(1-\beta_L)|A_i| \leq m_i \leq (1+\beta_R)|A_i|$ *for* $i = 1, 2, \cdots, m$*, and the set* $A_i$ *has a* $(\alpha_L, \alpha_R)$*-biased random generator* RandomElement*(*$A_i$*).*

**Definition 2.1.4.** *The model of randomized computation for our algorithm is defined below:*

- *the input is a list L defined in Definition 2.1.2,*

- *it allows all operations defined in Definition 2.1.2.*

### 2.2 Round and Round Complexity

Our algorithm has several rounds to access input sets. We also measure the round complexity, which is the number of rounds.

Our algorithm is considered as a client-server interaction. The algorithm is controlled by the server side, and each set is a client. In *one round*, the server asks some questions to clients which are selected.

The *round complexity* is the total number of rounds used in the algorithm. At each round, the algorithm send multiple requests to random generators, and membership queries, and receives the answers from them.

The parameters $m, \varepsilon, \gamma$ may be used to determine the time complexity and round complexity, where $\varepsilon$ controls the accuracy of approximation, $\gamma$ controls the failure probability, and $m$ is the number of sets.

# CHAPTER III

# PRELIMINARIES

During the accuracy analysis, Hoeffiding Inequality [24] and Chernoff Bound (see [32]) play an important role. They show how the number of samples determines the accuracy of approximation.

**Theorem 3.0.1** ([24])**.** *Let $X_1, \ldots, X_m$ be $m$ independent random variables in $[0,1]$ and $X = \sum_{i=1}^{m} X_i$.*

*i. If $X_i$ takes 1 with probability at most $p$ for $i = 1, \ldots, m$, then for any $\varepsilon > 0$, $\Pr(X > pm + \varepsilon m) < e^{-\frac{\varepsilon^2 m}{2}}$.*

*ii. If $X_i$ takes 1 with probability at least $p$ for $i = 1, \ldots, m$, then for any $\varepsilon > 0$, $\Pr(X < pm - \varepsilon m) < e^{-\frac{\varepsilon^2 m}{2}}$.*

**Theorem 3.0.2.** *Let $X_1, \ldots, X_m$ be $m$ independent random 0-1 variables, where $X_i$ takes 1 with probability at least $p$ for $i = 1, \ldots, m$. Let $X = \sum_{i=1}^{m} X_i$, and $\mu = E[X]$. Then for any $\delta > 0$, $\Pr(X < (1-\delta)pm) < e^{-\frac{1}{2}\delta^2 pm}$.*

**Theorem 3.0.3.** *Let $X_1, \ldots, X_m$ be $m$ independent random 0-1 variables, where $X_i$ takes 1 with probability at most $p$ for $i = 1, \ldots, m$. Let $X = \sum_{i=1}^{m} X_i$. Then for any $\delta > 0$, $\Pr(X > (1+\delta)pm) < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{pm}$.*

Define $g_1(\delta) = e^{-\frac{1}{2}\delta^2}$ and $g_2(\delta) = \frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}$. Define $g(\delta) = \max(g_1(\delta), g_2(\delta))$. We note that $g_1(\delta)$ and $g_2(\delta)$ are always strictly less than 1 for all $\delta > 0$. It is trivial for $g_1(\delta)$. For $g_2(\delta)$, this can be verified by checking that the function $f(x) = (1+x)\ln(1+x) - x$ is increasing and $f(0) = 0$. This is because $f'(x) = \ln(1+x)$ which is strictly greater than 0 for all $x > 0$.

We give a bound for $\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}$. Let $u(x) = \frac{e^{x}}{(1+x)^{(1+x)}}$. We consider the case $x \in [0,1]$. We have

$$\ln u(x) \quad = \quad x-(1+x)\ln(1+x) \leq x-(1+x)(x-\frac{x^2}{2}) = x-(x+\frac{x^2}{2}-\frac{x^3}{3}) \leq -\frac{x^2}{6}.$$

Therefore,

$$u(x) \leq e^{-\frac{x^2}{6}} \tag{3.1}$$

for all $x \in [0,1]$. We let

$$g^*(x) = e^{-\frac{x^2}{6}} \tag{3.2}$$

We have $g(x) \leq g^*(x)$ for all $x \in [0,1]$.

A well known fact, called union bound, in probability theory is the inequality

$$\Pr(E_1 \cup E_2 \ldots \cup E_m) \leq \Pr(E_1) + \Pr(E_2) + \ldots + \Pr(E_m),$$

where $E_1, E_2, \ldots, E_m$ are $m$ events that may not be independent. In the analysis of our randomized algorithm, there are multiple events such that the failure from any of them may fail the entire algorithm. We often characterize the failure probability of each of those events, and use the above inequality to show that the whole algorithm has a small chance to fail after showing that each of them has a small chance to fail.

# CHAPTER IV

## ALGORITHM BASED ON ADAPTIVE RANDOM SAMPLINGS

In this section, we develop a randomized algorithm for the size of set union when the approximate set sizes and biased random generators are given for the input sets. We give some definitions before the presentation of the algorithm. The algorithm developed in this section has an adaptive way to access the random generators from the input sets. All the random elements from input sets are generated in the beginning of the algorithm, and the number of random samples is known in the beginning of the algorithm. The results in this section show a tradeoff between the time complexity and the round complexity.

**Definition 4.0.1.** *Let $L = A_1, A_2, \cdots, A_m$ be a list of finite sets.*

1. *For an element $x$, define $T(x,L) = |\{i : 1 \le i \le m \text{ and } x \in A_i\}|$.*
2. *For an element $x$, and a subset of indices with multiplicity $H$ of $\{1,2,\cdots,m\}$, define $S(x,H) = |\{i : i \in H \text{ and } x \in A_i\}|$.*
3. *Define $minThickness(L) = \min\{T(x,L) : x \in A_1 \cup A_2 \cup \cdots \cup A_m\}$.*
4. *Define $maxThickness(L) = \max\{T(x,L) : x \in A_1 \cup A_2 \cup \cdots \cup A_m\}$.*
5. *Let $W$ be a subset with multiplicity of $A_1 \cup \cdots \cup A_m$, define $F(W,h,s) = \frac{s}{h} \sum\limits_{x \in W} \frac{1}{T(x,L)}$, and $F'(W) = \sum\limits_{x \in W} \frac{1}{T(x,L)} = \frac{h}{s} F(U,h,s)$.*
6. *For a $\delta \in (0,1)$, partition $A_1 \cup A_2 \cup \cdots \cup A_m$ into $A'_1, \cdots, A'_k$ such that $A'_i = \{x : x \in A_1 \cup A_2 \cup \cdots \cup A_m \text{ and } T(x,L) \in [(1+\delta)^{i-1}, (1+\delta)^i)\}$. Define $v(\delta, z_1, z_2, L) = k$, which is the number of sets in the partition under the condition that $z_1 \le T(x,L) \le z_2$.*

### 4.1 Overview of Algorithm

We give an overview of the algorithm. For an list $L$ of input sets $A_1, \cdots, A_m$, each set $A_i$ has an approximate size $m_i$ and a random generator. It is easy to see that $|A_1 \cup A_2 \cup \cdots \cup A_m| = \sum_{i=1}^{m} \sum_{x \in A_i} \frac{1}{T(x,L)}$. The first phase of the algorithm generates a set $R_1$ of sufficient random samples from the list of input sets with $\frac{m_1+\cdots+m_m}{|R_1|} \cdot \sum_{x \in R_1} \frac{1}{T(x,L)}$ is close to $\sum_{i=1}^{m} \sum_{x \in A_i} \frac{1}{T(x,L)}$. We will use the variable *sum* with initial value zero to approximate it. Each stage $i$ removes the set $V_i$ of elements from $R_i$ that each element $x \in V_i$ satisfies $T(x,L) \in \left[\frac{T_i}{4f_5(m)}, T_i\right]$, and all elements $x \in R_i$ with $T(x,L) \in \left[\frac{T_i}{f_5(m)}, T_i\right]$ are in $V_i$, where $T_i = \max\{T(x,L) : x \in R_i\}$ and $f_5(m)$ is a function at least 8, which will determine the number of rounds, and the trade off between the running time and the number of rounds. In phase $i$, we choose a set $H_i$ of $u_i$ (to be large enough) of indices from $1, \cdots, m$, and use $\frac{S(x,H_i)m}{u_i}$ to approximate $T(x,L)$. It is accurate enough if $u_i$ is large enough. The elements left in $R_i - V_i$ will have smaller $T(x,L)$. The set $R_{i+1}$ will be built for the next stage $i+1$. When $R_i - V_i$ is shrinked to $R_{i+1}$ by random sampling in $R_i - V_i$, each element in $R_{i+1}$ will have its weight to be scaled by a factor $\frac{|R_i - V_i|}{h_{i+1}}$. When an element $x$ is put into $V_i$, it is removed from $R_i$, and an approximate value of $\frac{1}{T(x,L)}$ multiplied by its weight is added to *sum*. Finally, we will prove that $sum \cdot (m_1 + \cdots + m_m)$ is close to $\sum_{i=1}^{m} \sum_{x \in A_i} \frac{1}{T(x,L)}$, which is equal to $|A_1 \cup A_2 \cup \cdots \cup A_m|$.

### 4.2 Algorithm Description

Before giving the algorithm, we define an operation that selects a set of random elements from a list $L$ of sets $A_1, \cdots, A_m$. We always assume $m \geq 2$ throughout the paper.

**Definition 4.2.1.** *Let $L$ be a list of $m$ sets $A_1, A_2, \cdots, A_m$ with $m_i \in [(1-\beta_L)|A_i|, (1+\beta_R)|A_i|]$ and $(\alpha_L, \alpha_R)$-biased random generator* RandomElement*($A_i$) for $i = 1, 2, \cdots, m$, and $M = m_1 + m_2 + \cdots + m_m$. A random choose of $L$ is to get an element $x$ via the following two steps:*

- *With probability $\frac{m_i}{M}$, select a set $A_i$ among $A_1, \cdots, A_m$.*
- *Get an element $x$ via* RandomElement$(A_i)$.

We give some definitions about the parameters and functions that affect our algorithm below. We assume that $\epsilon \in (0,1)$ is used to control the accuracy of approximation, and $\gamma \in (0,1)$ is used to control the failure probability. Both parameters are from the input. In the following algorithm, the parameters $z_{min}$ and $z_{max}$ with $1 \le z_{\min} \le minThickness(L) \le maxThickness(L) \le z_{\max} \le m$ can help speed up the computation. The algorithm is still correct if we use default case with $z_{min} = 1$ and $z_{max} = m$.

- The following parameters are used to controlled the accuracy of approximation at different stages of algorithm:

$$\epsilon_0 = \frac{\epsilon}{9}, \epsilon_1 = \frac{\epsilon_0}{6(\log m)}, \epsilon_2 = \frac{\epsilon_1}{4}, \epsilon_3 = \frac{\epsilon_0}{3}, \tag{4.1}$$

$$\delta = \frac{\epsilon_2}{2}. \tag{4.2}$$

- The following parameters are used to control the failure probability at several stages of the algorithm:

$$\gamma_1 = \frac{\gamma}{3}, \gamma_2 = \frac{\gamma}{6\log m}. \tag{4.3}$$

- Function $f_1(.)$ is used to control the number of rounds of the algorithm. Its growth rate is mainly determined by the parameter $c_1$ that will be determined later.

$$f_1(m) = 8m^{c_1} \text{ with } c_1 \ge 0,. \tag{4.4}$$

$$f_2(m) = \frac{2v(\delta, z_{\min}, z_{\max}, L)}{\epsilon_3} + \frac{2\log \frac{m}{z_{\min}}}{\epsilon_3 \log(1+\delta)}, \tag{4.5}$$

$$f_3(m) = \frac{f_1(m)}{2} \cdot \frac{\log \frac{\gamma_2}{2}}{\log \frac{1}{g^*(1)}}, \tag{4.6}$$

$$f_4(m) = \frac{f_2(m) \log \frac{m^2}{\epsilon_1}}{\log \frac{1}{g^*(\epsilon_1)}} + \frac{f_3(m)}{\epsilon_2 f_1(m)} \tag{4.7}$$

- Function $f_5(.)$ determine the number of random samples from the input sets in the beginning of the algorithm.

$$f_5(m) \quad = \quad \frac{m f_4(m)}{z_{\min}}. \tag{4.8}$$

- The following parameter is also used to control failure probability in a stage of the algorithm.

$$\gamma_3 \quad = \quad \frac{\gamma_2}{2 f_5(m)}, \text{ and} \tag{4.9}$$

- Function $f_6(.)$ affects the number of random indices in the range $\{1, 2, \cdots, m\}$. Those random indices will be used to choose input sets to detect the approximate $T(x, L)$ for those random samples $x$.

$$f_6(m) \quad = \quad \frac{f_3(m) f_1(m) v(\delta, z_{\min}, z_{\max}, L)}{\varepsilon_2}. \tag{4.10}$$

**Algorithm 1** ApproximateUnion($L, z_{\min}, z_{\max}, M, \gamma, \varepsilon$)

**Input** : $L$ is a list of $m$ sets $A_1, A_2, \cdots, A_m$ with $m \geq 2$, $m_i \in [(1-\beta_L)|A_i|, (1+\beta_R)|A_i|]$ and $(\alpha_L, \alpha_R)$-biased random generator RandomElement($A_i$) for $i = 1, 2, \cdots, m$, integers $z_{min}$ and $z_{max}$ with $1 \leq z_{\min} \leq minThickness(L) \leq maxThickness(L) \leq z_{\max} \leq m$, parameter $\gamma \in (0,1)$ to control the failure probability, parameter $\varepsilon \in (0,1)$ to control the accuracy of approximation, and $M = m_1 + m_2 + \cdots + m_m$ as the sum of sizes of input sets.

**Output** : $sum \cdot M$.

```
    Let $h_1 = f_5(m)$;
    Let $i = 1$;
    Let currentThickness$_1 = z_{max}$ ;
    Let $s_1 = \frac{m}{currentThickness_1}$;
    Let $s'_1 = 1$;
    Let $sum = 0$;
    Obtain a set $R_1$ of $h_1$ random chooses of $L$ (see Definition 4.2.1);
    Stage $i$
        Let $u_i = s_i \cdot f_6(m)$;
        Select $u_i$ random indices $H_i = \{k_1, \cdots, k_{u_i}\}$ from $\{1, 2, \cdots, m\}$.
        Compute $S(x, H_i)$ for each $x \in R_i$.
        Let $V_i$ be the subset of $R_i$ with elements $x$ satisfying $S(x, H_i) \geq \frac{currentThickness_i}{2 f_1(m) \cdot m} \cdot u_i$;
        Let $sum = sum + s'_i \sum_{x \in V_i} \frac{u_i}{S(x, H_i) m}$;
        Let $currentThickness_{i+1} = \frac{currentThickness_i}{f_1(m)}$;
        Let $s_{i+1} = \frac{m}{currentThickness_{i+1}}$;
        Let $h_{i+1} = \frac{h_1}{s_{i+1}}$;
        If $(|R_i| - |V_i| < h_{i+1})$
        Then
        {
            Let $R_{i+1} = R_i - V_i$;
            Let $a_i = 1$;
        }
```

```
Else
{
    Let $R_{i+1}$ be a set of random $h_{i+1}$ samples from $R_i - V_i$;
    Let $a_i = \frac{|R_i|-|V_i|}{h_{i+1}}$;
}
Let $s'_{i+1} = s'_i \cdot a_i$;
Let $i = i+1$;
If $(currentThickness_i < z_{min})$
    Return $sum \cdot M$ and terminate the algorithm;
Else
    Enter the next Stage $i$.
```

We let $M = m_1 + m_2 + \cdots + m_m$ and $z_{\min}$ be part of the input of the algorithm. It makes the algorithm be possible to run in a sublinear time when $z_{\min} \geq m^a$ for a fixed $a > 0$. Otherwise, the algorithm has to spend $\Omega(m)$ time to compute $M$.

### 4.3 Proof of Algorithm Performance

The accuracy and complexity of algorithm ApproximateUnion(.) will be proven in the following lemmas. Lemma 4.3.1 gives some basic properties of the algorithm. Lemma 4.3.3 shows that $R_1$ has random samples are used so that $F(R_1, h_1, 1)\left(\sum\limits_{i=1}^{m} m_i\right)$ is an accurate approximation for $\sum\limits_{i=1}^{m}\sum\limits_{x\in A_i}\frac{1}{T(x,L)}$.

**Lemma 4.3.1.** *The algorithm ApproximateUnion(.) has the following properties:*

1. $g^*(\varepsilon_1)^{\frac{f_4(m)}{f_2(m)}} \leq \frac{\varepsilon_1}{m^2}$.
2. $v(\delta, z_{\min}, z_{\max}, L) = \mathrm{O}\left(\frac{\log\frac{z_{\max}}{z_{\min}}}{\delta}\right)$.
3. $\frac{2v(\delta,z_{\min},z_{\max},L)}{f_2(m)} \leq \varepsilon_3$ *and* $f_2(m) = \mathrm{O}\left(\frac{\log\frac{m}{z_{\min}}}{\delta\varepsilon_3}\right)$,
4. $R_i$ *contains at most* $h_i$ *items, and*

5. $f_4(m) = \mathrm{O}\left(\frac{1}{\varepsilon^4}\left(\log\frac{m}{z_{\min}}\right)\cdot\log\frac{m}{\varepsilon}\cdot(\log m)^3 + \frac{\log m \log\frac{1}{\gamma}}{\varepsilon}\right)$.

*Proof.* The statements are easily proven according to the setting in the algorithm.

Statement 1: It follows from equations (3.2) and (4.7).

Statement 2: By Definition 4.0.1, we need $v(\delta, z_{\min}, z_{\max}, L)$ with $z_{min}(1+\delta)^{v(\delta, z_{\min}, z_{\max}, L)} \geq z_{max}$. Thus, we have $v(\delta, z_{\min}, z_{\max}, L) \leq 2\left(\frac{\log\frac{z_{\max}}{z_{\min}}}{\log(1+\delta)}\right) = \mathrm{O}\left(\frac{\log\frac{z_{\max}}{z_{\min}}}{\delta}\right)$ since $\log(1+\delta) = \Theta(\delta)$.

Statement 3: It is easy to see that $\log(1+\delta) = \Theta(\delta)$ and $1 \leq z_{max} \leq m$. It follows from equation (4.5), and Statement 2.

Statement 4: It follows from lines 19 to 5 in the algorithm.

Statement 5: We also have $\frac{f_3(m)}{\varepsilon_2 f_1(m)} = \frac{1}{2\varepsilon_2}\cdot\frac{\log\frac{\gamma_2}{2}}{\log\frac{1}{g^*(1)}}$. By equation (4.7), Statement 3 and equations (4.1), we have

$$
\begin{aligned}
f_4(m) &= \frac{f_2(m)\log\frac{m^2}{\varepsilon_1}}{\log\frac{1}{g^*(\varepsilon_1)}} + \frac{f_3(m)}{\varepsilon_2 f_1(m)} && (4.11)\\
&\leq \left(6f_2(m)\cdot\log\frac{m^2}{\varepsilon_1}\cdot\left(\frac{\log m}{\varepsilon_0}\right)^2 + \frac{f_3(m)}{\varepsilon_2 f_1(m)}\right) && (4.12)\\
&\leq \left(6f_2(m)\cdot\log\frac{m^2}{\varepsilon_1^2}\cdot\left(\frac{\log m}{\varepsilon_0}\right)^2 + \frac{1}{2\varepsilon_2}\cdot\frac{\log\frac{\gamma_2}{2}}{\log\frac{1}{g^*(1)}}\right) && (4.13)\\
&= \left(\frac{6}{\delta\varepsilon_3\varepsilon_0^2}(\log\frac{m}{z_{\min}})\cdot\log\frac{m}{\varepsilon_1}\cdot(\log m)^2 + \frac{12\log m}{\varepsilon_0}\cdot\frac{\log\frac{\gamma_2}{2}}{\log\frac{1}{g^*(1)}}\right) && (4.14)\\
&= \mathrm{O}\left(\frac{1}{\varepsilon^4}\left(\log\frac{m}{z_{\min}}\right)\cdot\log\frac{m}{\varepsilon}\cdot(\log m)^3 + \frac{\log m\log\frac{1}{\gamma}}{\varepsilon}\right). && (4.15)
\end{aligned}
$$

□

Lemma 4.3.2 gives an upper bound for the number of rounds for the algorithm. It shows how round complexity depends on $z_{max}, z_{min}$ and $f_1(.)$.

**Lemma 4.3.2.** *The number of phases of the algorithm is* $\mathrm{O}\left(\frac{\log\frac{z_{max}}{z_{min}}}{\log f_1(m)}\right)$.

*Proof.* By line 3 of the algorithm, we have $\text{currentThickness}_1 = z_{max}$. Variable $\text{currentThickness}_i$ is reduced by a factor $f_1(m)$ each phase as $\text{currentThickness}_{i+1} = \frac{\text{currentThickness}_i}{f_1(m)}$ by line 14 of the algorithm. By the termination condition of line 8 of the algorithm, if $y$ is the number of phases of the algorithm, we have $y \le y'$, where $y'$ is any integer with $\frac{z_{max}}{f_1(m)^{y'}} < z_{min}$. Thus, $y = \mathrm{O}\left(\frac{\log \frac{z_{max}}{z_{min}}}{\log f_1(m)}\right)$. □

Lemma 4.3.3 shows the random samples, which are saved in $R_1$ in the beginning of the algorithm, will be enough to approximate the size of set union via $F(R_1, h_1, 1)M$. In the next a few rounds, algorithm will approximate $F(R_1, h_1, 1)$.

**Lemma 4.3.3.** *With probability at least* $1-\gamma_1$*,* $F(R_1, h_1, 1)M \in [(1-\varepsilon_0)(1-\alpha_L)(1-\beta_L)|A_1 \cup \cdots \cup A_m|, (1+\varepsilon_0)(1+\alpha_R)(1+\beta_R)(1+\delta)|A_1 \cup \cdots \cup A_m|]$.

*Proof.* Let $A = |A_1 \cup \cdots \cup A_m|$ and $U = |A_1| + |A_2| + \cdots + |A_m|$. For an arbitrary set $A_i$ in the list $L$, and an arbitrary element $x \in A_i$, with at least the following probability $x$ is selected via RandomElement($A_i$) at line 7 of Algorithm ApproximateUnion(.).

$$\begin{aligned} \frac{m_i}{m_1+m_2+\cdots+m_m}\frac{1-\alpha_L}{|A_i|} &\ge \frac{(1-\beta_L)|A_i|}{M}\frac{1-\alpha_L}{|A_i|} \\ &= \frac{(1-\beta_L)(1-\alpha_L)}{M}. \end{aligned}$$

Similarly, with at most the following probability $x$ is chosen via RandomElement($A_i$) at line 7 of Algorithm ApproximateUnion(.).

$$\begin{aligned} \frac{m_i}{m_1+m_2+\cdots+m_m}\frac{1+\alpha_R}{|A_i|} &\le \frac{(1+\beta_R)|A_i|}{M}\frac{1+\alpha_R}{|A_i|} \\ &= \frac{(1+\beta_R)(1+\alpha_R)}{M}. \end{aligned}$$

Define $\rho_1 = 1-(1-\beta_L)(1-\alpha_L)$ and $\rho_2 = (1+\beta_R)(1+\alpha_R)-1$. Each element $x$ in $A_1 \cup A_2 \cup \cdots \cup A_m$ is selected with probability in $\left[\frac{(1-\rho_1)T(x,L)}{M}, \frac{(1+\rho_2)T(x,L)}{M}\right]$.

Define $T_1 = \left\{A'_j : |A'_j| \le \frac{A}{f_2(m)}\right\}$, and $T_2 = \left\{A'_j : |A'_j| > \frac{A}{f_2(m)}\right\}$ (see 6 of Definition (4.0.1)). Let $t_j = \min\left\{T(x,L) : x \in A'_j\right\}$. We discuss two cases:

Case 1: $A'_j \in T_1$. When one element $x$ is chosen, the probability that $x \in A'_j$ is in the range

$\left[\frac{(1-\rho_1)t_j|A'_j|}{M}, \frac{(1+\rho_2)(1+\delta)t_j|A'_j|}{M}\right]$. Let $p_j = \frac{(1+\rho_2)(1+\delta)t_j\cdot\frac{A}{f_2(m)}}{M}$. Since $z_{\min} \le minThickness(L)$, we have $z_{\min} \le minThickness(L) \le t_j$. It is easy to see that $mA \ge U$. We have

$$\begin{aligned} p_j h_1 &= \frac{(1+\rho_2)(1+\delta)t_j\cdot\frac{A}{f_2(m)}}{M}\cdot\frac{mf_4(m)}{z_{\min}} && (4.16)\\ &\ge \frac{(1+\rho_2)(1+\delta)\cdot f_4(m)mA}{f_2(m)M} && (4.17)\\ &\ge \frac{(1+\rho_2)(1+\delta)\cdot f_4(m)U}{f_2(m)M} && (4.18)\\ &\ge \frac{(1+\rho_2)(1+\delta)\cdot f_4(m)}{f_2(m)(1+\beta_R)} && (4.19)\\ &= \frac{(1+\alpha_R)(1+\delta)\cdot f_4(m)}{f_2(m)}. && (4.20) \end{aligned}$$

Let $\omega_1(m) = \frac{(1+\alpha_R)(1+\delta)\cdot f_4(m)}{f_2(m)}$. Thus, $p_j h_1 \ge \omega_1(m)$.

Let $R_{i,j}$ be the elements of $R_i$ and also in $A'_j$. By Theorem 3.0.3, with probability at most $P_j = g^*(1)^{p_j\cdot h_1} \le g^*(1)^{\omega_1(m)} \le \frac{\gamma_1}{2}$ (by equation (4.5), equation (4.7) and inequality (4.20)), there are more than $2p_jh_1 = \frac{2(1+\rho_2)(1+\delta)t_j\cdot\frac{A}{f_2(m)}}{M}\cdot h_1$ elements to be chosen from $A'_j$ into $R_1$. Thus,

$$F'(R_{1,j}) \le \frac{2p_jh_1}{t_j} \le \frac{2(1+\rho_2)(1+\delta)\cdot A}{f_2(m)M}\cdot h_1. \qquad (4.21)$$

with probability at most $P_j$ to fail.

Case 2: $A'_j \in T_2$. When $h_1$ elements are selected to $R_1$, let $v_j$ be the number of elements selected in $A'_j$. When one element $x$ is chosen, the probability that $x \in A'_j$ is in the range

$$\left[\frac{(1-\rho_1)t_j|A'_j|}{M}, \frac{(1+\rho_2)(1+\delta)t_j|A'_j|}{M}\right].$$

Let $p_{j,1} = \frac{(1-\rho_1)t_j|A'_j|}{M}$ and $p_{j,2} = \frac{(1+\rho_2)(1+\delta)t_j|A'_j|}{M}$.

We have

$$\begin{aligned}
p_{j,1}h_1 &= \frac{(1-\rho_1)t_j|A'_j|}{M}\cdot h_1 && (4.22)\\
&> \frac{(1-\rho_1)t_j\frac{A}{f_2(m)}}{M}\cdot h_1 \geq \frac{(1-\rho_1)z_{min}A\cdot h_1}{f_2(m)M} = \frac{(1-\rho_1)z_{min}A\cdot mf_4(m)}{z_{min}f_2(m)M} && (4.23)\\
&\geq \frac{(1-\rho_1)f_4(m)}{f_2(m)(1+\beta_R)}. && (4.24)
\end{aligned}$$

We have

$$\begin{aligned}
p_{j,2}h_1 &= \frac{(1+\rho_2)(1+\delta)t_j|A'_j|}{M}\cdot h_1 && (4.25)\\
&> \frac{(1+\rho_2)(1+\delta)t_j\cdot\frac{A}{f_2(m)}}{M}\cdot h_1 && (4.26)\\
&\geq \frac{(1+\rho_2)(1+\delta)z_{min}\cdot A}{f_2(m)M}\cdot h_1 && (4.27)\\
&= \frac{(1+\rho_2)(1+\delta)z_{min}\cdot A}{f_2(m)M}\cdot\frac{mf_4(m)}{z_{min}} && (4.28)\\
&\geq \frac{(1+\rho_2)(1+\delta)\cdot f_4(m)}{f_2(m)(1+\beta_R)} && (4.29)\\
&\geq \frac{(1+\alpha_R)(1+\delta)\cdot f_4(m)}{f_2(m)}. && (4.30)
\end{aligned}$$

With probability at most $g^*(\varepsilon_3)^{p_{j,1}\cdot h_1} \leq \frac{\gamma_1}{4}$ (by equation (4.5), equation (4.7) and inequality (4.24)),

$$v_j < \frac{(1-\varepsilon_3)(1-\rho_1)t_j|A'_j|}{M}\cdot h_1 = (1-\varepsilon_3)(1-\rho_1)t_jh_1\cdot\frac{|A'_j|}{M}. \qquad (4.31)$$

With probability at most $g^*(\varepsilon_3)^{p_{j,2}\cdot h_1} \leq \frac{\gamma_1}{4}$ (by equation (4.5), equation (4.7) and inequality (4.30)),

$$v_j > \frac{(1+\varepsilon_3)(1+\rho_2)(1+\delta)t_j|A'_j|}{M}\cdot h_1 = (1+\varepsilon_3)(1+\rho_2)(1+\delta)t_jh_1\cdot\frac{|A'_j|}{M}. \qquad (4.32)$$

Therefore, with probability at least $1-\gamma_1/2$, we have

$$v_j \in \left[(1-\varepsilon_3)(1-\rho_1)t_j h_1 \cdot \frac{|A'_j|}{M}, (1+\varepsilon_3)(1+\rho_2)(1+\delta)t_j h_1 \cdot \frac{|A'_j|}{M}\right]. \tag{4.33}$$

Thus, we have that there are sufficient elements of $A'_j$ to be selected with high probability, which follows from Theorem 3.0.2 and Theorem 3.0.3.

In the rest of the proof, we assume that inequality (4.21) holds if the condition of Case 1 holds, and inequality (4.33) holds if the condition of Case 2 holds.

Now we consider

$$
\begin{aligned}
& F(R_1,h_1,1) = \frac{1}{h_1}\sum_{x\in R_1}\frac{1}{T(x,L)} && (4.34)\\
= \;& \frac{1}{h_1}\left(\sum_{R_{1,j}\ with\ A'_j\in T_1}\sum_{x\in R_{1,j}}\frac{1}{T(x,L)} + \sum_{R_{1,j}\ with\ A'_j\in T_2}\sum_{x\in R_{1,j}}\frac{1}{T(x,L)}\right) && (4.35)\\
\leq \;& \frac{1}{h_1}\frac{2(1+\rho_2)(1+\delta)v(\delta,z_{\min},z_{\max},L)\cdot A}{f_2(m)M}\cdot h_1 && (4.36)\\
+ \;& \frac{1}{h_1}\sum_{R_{1,j}\ with\ A'_j\in T_2}\sum_{x\in R_{1,j}}\frac{1}{T(x,L)} && (4.37)\\
= \;& \frac{1}{h_1}\left(\frac{2(1+\rho_2)(1+\delta)v(\delta,z_{\min},z_{\max},L)\cdot A}{f_2(m)M}\cdot h_1 + \sum_{R_{1,j}\ with\ A'_j\in T_2}\frac{v_j}{t_j}\right) && (4.38)\\
\leq \;& \frac{1}{h_1}\left(\frac{2(1+\rho_2)(1+\delta)v(\delta,z_{\min},z_{\max},L)\cdot A}{f_2(m)M}\cdot h_1\right) && (4.39)\\
+ \;& \frac{1}{h_1}\left(\sum_{R_{1,j}\ with\ A'_j\in T_2}(1+\varepsilon_3)(1+\rho_2)(1+\delta)h_1\cdot\frac{|A'_j|}{M}\right) && (4.40)\\
= \;& \frac{2(1+\rho_2)(1+\delta)v(\delta,z_{\min},z_{\max},L)\cdot A}{f_2(m)M} && (4.41)\\
+ \;& \sum_{R_{1,j}\ with\ A'_j\in T_2}(1+\varepsilon_3)(1+\rho_2)(1+\delta)\cdot\frac{|A'_j|}{M} && (4.42)\\
\leq \;& \left(\frac{2(1+\rho_2)(1+\delta)v(\delta,z_{\min},z_{\max},L)}{f_2(m)} + (1+\varepsilon_3)(1+\rho_2)(1+\delta)\right)\frac{A}{M} && (4.43)\\
= \;& \left(1+\varepsilon_3+\frac{2v(\delta,z_{\min},z_{\max},L)}{f_2(m)}\right)(1+\rho_2)(1+\delta)\frac{A}{M} && (4.44)\\
\leq \;& (1+2\varepsilon_3)(1+\rho_2)(1+\delta)\frac{A}{M} && (4.45)\\
\leq \;& (1+\varepsilon_0)(1+\rho_2)(1+\delta)\frac{A}{M}. && (4.46)
\end{aligned}
$$

The transition from (4.44) to (4.45) is by Statement 3 of Lemma 4.3.1. For the lower bound part, we have the following inequalities:

$$F(R_1,h_1,1) = \frac{1}{h_1}\sum_{x\in R_1}\frac{1}{T(x,L)} \tag{4.47}$$

$$\geq \frac{1}{h_1}\left(\sum_{R_{1,j}\ with\ A'_j\in T_2}\ \sum_{x\in R_{1,j}}\frac{1}{T(x,L)}\right) \tag{4.48}$$

$$= \frac{1}{h_1}\left(\sum_{R_{1,j}\ with\ A'_j\in T_2}\frac{v_j}{t_j}\right) \tag{4.49}$$

$$\geq \frac{1}{h_1}\left(\sum_{R_{1,j}\ with\ A'_j\in T_2}(1-\varepsilon_3)(1-\rho_1)h_1\cdot\frac{|A'_j|}{M}\right) \tag{4.50}$$

$$= \frac{(1-\varepsilon_3)(1-\rho_1)}{M}\left(\sum_{R_{1,j}\ with\ A'_j\in T_2}|A'_j|\right) \tag{4.51}$$

$$= \frac{(1-\varepsilon_3)(1-\rho_1)}{M}\sum_{R_{1,j}\ with\ A'_j\in T_1}|A'_j| + \sum_{R_{1,j}\ with\ A'_j\in T_2}|A'_j| \tag{4.52}$$

$$- \frac{(1-\varepsilon_3)(1-\rho_1)}{M}\sum_{R_{1,j}\ with\ A'_j\in T_1}|A'_j| \tag{4.53}$$

$$= \frac{(1-\varepsilon_3)(1-\rho_1)}{M}\left(A-\sum_{R_{1,j}\ with\ A'_j\in T_1}|A'_j|\right) \tag{4.54}$$

$$= \frac{(1-\varepsilon_3)(1-\rho_1)}{M}\left(A-\frac{v(\delta,z_{\min},z_{\max},L)A}{f_2(m)}\right) \tag{4.55}$$

$$= \left(1-\frac{v(\delta,z_{\min},z_{\max},L)}{f_2(m)}\right)(1-\varepsilon_3)(1-\rho_1)\frac{A}{M} \tag{4.56}$$

$$\geq (1-\varepsilon_3)(1-\varepsilon_3)(1-\rho_1)\frac{A}{M} \tag{4.57}$$

$$\geq (1-\varepsilon_0)(1-\rho_1)\frac{A}{M}. \tag{4.58}$$

The transition from (4.56) to (4.57) is by Statement 3 of Lemma 4.3.1. Therefore, $F(R_1,h_1,1)M \in [(1-\varepsilon_0)(1-\rho_1)A,(1+\varepsilon_0)(1+\rho_2)(1+\delta)A]$. □

Lemma 4.3.4 shows that at stage $i$, it can approximate $T(x,L)$ for all random samples with highest $T(x,L)$ in $R_i$. Those random elements with highest $T(x,L)$ will be removed in stage $i$ so that the algorithm will look for random elements with smaller $T(x,L)$ in the coming stages.

**Lemma 4.3.4.** *After the execution of Stage i, with probability at least $1-\gamma_2$, we have the following three statements:*

1. *every element $x \in R_i$ with $T(x,L) \geq \frac{\text{currentThickness}_i}{4f_1(m)}$ has $S(x,H_i) \in \left[(1-\varepsilon_1)\frac{T(x,L)}{m}u_i, (1+\varepsilon_1)\frac{T(x,L)}{m}u_i\right]$, and*

2. *every element $x \in V_i$ with $T(x,L) \geq \frac{\text{currentThickness}_i}{f_1(m)}$, it satisfies the condition in line 12 of the algorithm.*

3. *every element $x \in V_i$ with $T(x,L) < \frac{\text{currentThickness}_i}{4f_1(m)}$, it does not satisfy the condition in line 12 of the algorithm.*

*Proof.* It follows from Theorem 3.0.2 and Theorem 3.0.3. There are $u_i = s_i f_6(m)$ indices are selected among $\{1,2,\cdots,m\}$. Let $p = \frac{T(x,L)}{m}$.

Statment 1: We have $pu_i = \frac{T(x,L)}{m} \cdot s_i f_6(m) \geq \frac{\text{currentThickness}_i}{4f_1(m)} \cdot \frac{1}{m} \cdot s_i f_6(m) = \frac{\text{currentThickness}_i}{4f_1(m)} \cdot \frac{1}{m} \cdot \frac{m}{\text{currentThickness}_i} \cdot f_6(m) = \frac{f_6(m)}{4f_1(m)}$.

With probability at most $P_1 = g^*(\varepsilon_1)^{pu_i} \leq \frac{\gamma_3}{2}$ (by equations (4.10), and (4.6)), $S(x,H_i) < (1-\varepsilon_1)\frac{T(x,L)}{m}u_i$. With probability at most $P_2 = g^*(\varepsilon_1)^{pu_i} \leq \frac{\gamma_3}{2}$ (by equations (4.10) and (4.6)), $S(x,H_i) > (1+\varepsilon_1)\frac{T(x,L)}{m}u_i$.

There are at most $h_i$ elements in $R_i$ by Statement 4 of Lemma 4.3.1. Therefore, with probability at most $h_i(P_1+P_2) \leq h_1(P_1+P_2) \leq h_1 \cdot \gamma_3 = \frac{\gamma_2}{2}$,

$$S(x,H_i) \notin \left[(1-\varepsilon_1)\frac{T(x,L)}{m}u_i, (1+\varepsilon_1)\frac{T(x,L)}{m}u_i\right].$$

Statement 2: This statement of the lemma follows from Statement 1.

Statement 3: This part of the lemma follows from Theorem 3.0.2 and Theorem 3.0.3. For $x \in V_i$ with $T(x,L) < \frac{\text{currentThickness}_i}{4f_1(m)}$, let $p = \frac{\text{currentThickness}_i}{4f_1(m)}$. With probability at most $g^*(1)^{pu_i} \leq \frac{\gamma_2}{2}$ (by equations (4.10), and (4.6)), we have $S(x,H_i) \geq 2pu_i$. □

**Lemma 4.3.5.** *Let x and y be positive real numbers with $1 \leq y$. Then we have*

- $1 - xy < (1-x)^y$,

- *If $xy < 1$, then $(1+x)^y < 1+2xy$, and*

- *If $x_1, x_2 \in [0,1)$, then $1-x_1-x_2 \le (1-x_1)(1-x_2)$, and $(1+x_1)(1+x_2) \le 1+2x_1+x_2$.*

*Proof.* By Taylor formula, we have $(1-x)^y = 1-xy+\frac{y\cdot(y-1)}{2}\theta^2$ for some $\theta \in [0,x]$. Thus, we have $(1-x)^y \ge 1-yx$. Note that the function $(1+\frac{1}{z})^z$ is increasing, and $\lim_{z\to+\infty}(1+\frac{1}{z})^z = e$. We also have $(1+x)^y \le (1+x)^{\frac{1}{x}\cdot xy} \le e^{xy} \le 1+xy+(xy)^2 \le 1+2xy$.

It is trivial to verify Statement 4.3.5. $1-x_1-x_2 \le (1-x_1)(1-x_2)$. Clearly, $(1+x_1)(1+x_2) = 1+x_1+x_2+x_1x_2 \le 1+2x_1+x_2$. □

Lemma 4.3.6 shows that how to gradually approximate $F(R_1,h_1,1)M$ via several rounds. It shows that the left random samples stored in $R_{i+1}$ after stage $i$ is enough to approximate $F'(R_i - V_i)$.

**Lemma 4.3.6.** *Let $y$ be the number of stages. Let $V_i$ be the set of elements removed from $R_i$ in Stage $i$. Then we have the following facts:*

- *With probability at least $1-\gamma_2$, $a_iF'(R_{i+1}) \in [(1-\varepsilon_1)F'(R_i-V_i),(1+\varepsilon_1)F'(R_i-V_i)]$, and*

- *With probability at least $1-2y\gamma_2$, $\sum\limits_{i=1}^{y} s_i'F'(V_i) \in [(1-y\varepsilon_1)S,(1+2y\varepsilon_1)S]$, where $S = F(R_1,h_1,1)$.*

*Proof.* Let $h_i' = h_i - |V_i|$. If an local is too small, it does not affect the global sum much. In $R_{i+1}$, we deal with the elements $x$ of $T(x,L) < \frac{\text{currentThickness}_i}{f_1(m)}$. By Lemma 4.3.4, with probability at least $1-\gamma_2$, $R_i - V_i$ does not contain any $x$ with $T(x,L) \ge \frac{\text{currentThickness}_i}{f_1(m)}$.

Let $t_{i,j}$ be the number of elements of $A_j'$ in $R_i$ with multiplicity. Let $B_{i,j}$ be the set of elements in both $R_i$ and $A_j'$ with multiplicity.

Statement 1: We discuss two cases:

Case 1: $|R_i| - |V_i| < h_{i+1}$. This case is trivial since $R_{i+1} = R_i - V_i$ and $a_i = 1$ according to the algorithm.

In the following Case 2, we assume the condition of Case 1 is false. Thus, $h_i' \ge h_{i+1}$.

Case 2: $|R_i| - |V_i| \geq h_{i+1}$. We have

$$F'(R_i - V_i) \geq \frac{h'_i}{\frac{\text{currentThickness}_i}{f_1(m)}} \geq \frac{h_{i+1}}{\frac{\text{currentThickness}_i}{f_1(m)}} \tag{4.59}$$

$$= \frac{\frac{h_1}{s_{i+1}} \cdot f_1(m)}{\text{currentThickness}_i} = \frac{f_4(m)}{z_{min}}. \tag{4.60}$$

Two subcases are discussed below.

Subcase 2.1: $t_{i,j} \leq f_3(m)$, in this case, $B_{i,j}$ has a small impact for the global sum.

Let $p = \frac{f_3(m)}{h'_i}$. By Theorem 3.0.2 and Theorem 3.0.3, with probability at least $1 - g^*(1)^{ph_{i+1}} = 1 - g^*(1)^{\frac{f_3(m)}{f_1(m)}} \geq 1 - \frac{\gamma_2}{2}$ (by equation (4.6)),

$$|B_{i+1,j}| \leq 2ph_{i+1} = 2 \cdot \frac{f_3(m)}{h'_i} \cdot \frac{h_i}{f_1(m)} = \frac{2f_3(m)}{f_1(m)} \cdot \frac{h_i}{h'_i} \leq \frac{2f_3(m)}{f_1(m)} \cdot \frac{h_i}{h_{i+1}} \leq \frac{2f_3(m)}{f_1(m)^2}. \tag{4.61}$$

We assume $|B_{i+1,j}| \leq \frac{2f_3(m)}{f_1(m)^2}$. We have $F'(B_{i+1,j}) \leq \frac{|B_{i+1,j}|}{z_{min}} \leq \frac{2f_3(m)}{z_{min}f_1(m)^2}$. Clearly, $a_i \leq f_1(m)$. Thus,

$$a_iF'(B_{i+1,j}) \leq f_1(m) \cdot \frac{2f_3(m)}{z_{min}f_1(m)^2} = \frac{2f_3(m)}{z_{min}f_1(m)} \tag{4.62}$$

$$= \frac{2f_3(m)}{f_4(m)f_1(m)} \cdot \frac{f_4(m)}{z_{min}} \tag{4.63}$$

$$\leq \frac{2f_3(m)}{f_1(m)f_4(m)} \cdot F'(R_i - V_i) \tag{4.64}$$

$$\leq 2\varepsilon_2 \cdot F'(R_i - V_i). \tag{4.65}$$

The transition from (4.63) to (4.64) is by inequality (4.60). The transition from (4.64) to (4.65) is by inequality (4.7).

Subcase 2.2: $t_{i,j} > f_3(m)$ in $R_i$, in this case, $B'_j$ does not lose much accuracy. From $R_i$ to $R_{i+1}$, $h_{i+1} = \frac{h_i}{f_1(m)}$ elements are selected.

Let $q = \frac{t_{i,j}}{h'_i}$. We have

$$qh_{i+1} = \frac{t_{i,j}}{h'_i} \cdot h_{i+1} = t_{i,j} \cdot \frac{h_{i+1}}{h'_i} \geq t_{i,j} \cdot \frac{h_{i+1}}{h_i} \geq \frac{f_3(m)}{f_1(m)}. \tag{4.66}$$

With probability at most $g^*(\varepsilon_2)^{qh_{i+1}} \leq \frac{\gamma_2}{2}$ (by inequality (4.66)), we have that $|B_{i+1,j}| < (1-\varepsilon_2)qh_{i+1}$. With probability at most $g^*(\varepsilon_2)^{qh_{i+1}} \leq \frac{\gamma_2}{2}$, we have that $|B_{i+1,j}| > (1+\varepsilon_2)qh_{i+1}$. They follow from Theorem 3.0.2 and Theorem 3.0.3.

We assume $|B_{i+1,j}| \in [(1-\varepsilon_2)qh_{i+1}, (1+\varepsilon_2)qh_{i+1}]$. Thus, $a_iF'(B_{i+1,j}) \in [(1-\varepsilon_2)t_{i,j}, (1+\varepsilon_2)t_{i,j}]$. So, $a_iF'(B_{i+1,j}) \in \left[\frac{(1-\varepsilon_2)F'(R_{i,j})}{1+\delta}, (1+\varepsilon_2)F'(R_{i,j})(1+\delta)\right]$.

We have

$$\begin{aligned}
a_iF'(R_{i+1}) &= a_i\left(\sum_j F'(R_{i+1,j})\right) && (4.67)\\
&\leq (1+\varepsilon_2)(1+\delta)F'(R_i - V_i) && (4.68)\\
&+ \frac{2f_3(m)}{f_1(m)f_4(m)} \cdot v(\delta, z_{\min}, z_{\max}, L)F'(R_i - V_i) && (4.69)\\
&= ((1+\varepsilon_2)(1+\delta) + 2\varepsilon_2)\,F'(R_i - V_i) && (4.70)\\
&\leq ((1+\varepsilon_2)(1+\delta) + 2\varepsilon_2)F'(R_i - V_i) && (4.71)\\
&\leq (1+4\varepsilon_2)F'(R_i - V_i) && (4.72)\\
&\leq (1+\varepsilon_1)F'(R_i - V_i). && (4.73)
\end{aligned}$$

The transition from (4.70) to (4.71) is based on equation (4.10). The transition from (4.71) to (4.72) is based on equation (4.2). The transition from (4.72) to (4.73) is based on equations (4.1).

We have

$$
\begin{aligned}
a_i F'(R_{i+1}) &= a_i \left( \sum_j F'(R_{i+1,j}) \right) && (4.74)\\
&\geq \frac{(1-\varepsilon_2)F'(R_i - V_i)}{(1+\delta)} - \frac{2 f_3(m)}{f_1(m) f_4(m)} \cdot v(\delta, z_{\min}, z_{\max}, L) F'(R_i - V_i) && (4.75)\\
&\geq \left( \frac{(1-\varepsilon_2)}{(1+\delta)} - 2\varepsilon_2 \right) F'(R_i - V_i) && (4.76)\\
&\geq \left( \frac{(1-\varepsilon_2)}{(1+\delta)} - 2\varepsilon_2 \right) F'(R_i - V_i) && (4.77)\\
&\geq (1-4\varepsilon_2) F'(R_i - V_i) && (4.78)\\
&\geq (1-\varepsilon_1) F'(R_i - V_i). && (4.79)
\end{aligned}
$$

The transition from (4.76) to (4.77) is based on the equation (4.10). The transition from (4.77) to (4.78) is based on equation (4.2). The transition from (4.78) to (4.79) is based on equations (4.1).

Statement 2: In the rest of the proof, we assume that if $|R_i| - |V_i| \geq h_{i+1}$, then $F'(R_{i+1}) = F'(R_i - V_i)$, and if $|R_i| - |V_i| < h_{i+1}$, then $f_1(m)F'(R_{i+1}) \in [(1-\varepsilon_1)F'(R_i - V_i), (1+\varepsilon_1)F'(R_i - V_i)]$.

In order to prove Statement 4.3.6, we give an inductive proof that

$$s'_{k+1} F'(R_{k+1}) + \sum_{i=1}^{k} s'_i F'(V_i) \in [(1-\varepsilon_1)^k S, (1+\varepsilon_1)^k S].$$

It is trivial for $k = 0$. Assume that $s'_k F'(R_k) + \sum_{i=1}^{k-1} s'_i F'(V_i) \in [(1-\varepsilon_1)^{k-1} S, (1+\varepsilon_1)^{k-1} S]$.

Since $F'(R_k) = F'(R_k - V_k) + F'(V_k)$, we have

$$a_k F'(R_{k+1}) + F'(V_k) \in [(1-\varepsilon_1)F'(R_k), (1+\varepsilon_1)F'(R_k)].$$

.

Thus, we have

$$\begin{aligned}
s'_{k+1}F'(R_{k+1})+\sum_{i=1}^{k}s'_iF'(V_i) &= s'_{k+1}F'(R_{k+1})+s'_kF'(V_k)+\sum_{i=1}^{k-1}s_iF'(V_i)\\
&= s'_k(a_kF'(R_{k+1})+F'(V_k))+\sum_{i=1}^{k-1}s_iF'(V_i)\\
&\leq (1+\varepsilon_1)s'_kF'(R_k)+\sum_{i=1}^{k-1}s_iF'(V_i)\\
&\leq (1+\varepsilon_1)\left(s'_kF'(R_k)+\sum_{i=1}^{k-1}s_iF'(V_i)\right)\\
&\leq (1+\varepsilon_1)^kS.
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
s'_{k+1}F'(R_{k+1})+\sum_{i=1}^{k}s'_iF'(V_i) &= s'_{k+1}F'(R_{k+1})+s_kF'(V_k)+\sum_{i=1}^{k-1}s'_iF'(V_i)\\
&= s'_k\left(a_kF'(R_{k+1})+F'(V_k)\right)+\sum_{i=1}^{k-1}s'_iF'(V_i)\\
&\geq (1-\varepsilon_1)s'_kF'(R_k)+\sum_{i=1}^{k-1}s'_iF'(V_i)\\
&\geq (1-\varepsilon_1)\left(s'_kF'(R_k)+\sum_{i=1}^{k-1}s'_iF'(V_i)\right)\\
&\geq (1-\varepsilon_1)^kS.
\end{aligned}$$

Thus, we have $s'_{k+1}F'(R_{k+1})+\sum_{i=1}^{k}s'_iF'(V_i)\in[(1-\varepsilon_1)^kS,(1+\varepsilon_1)^kS]$.

Therefore, with probability at least $1-y\gamma_2-y\gamma_2$, $\sum_{i=1}^{y}s'_iF'(V_i)\in[(1-\varepsilon_1)^yS,(1+\varepsilon_1)^yS]\subseteq[(1-\varepsilon_1y)S,(1+2\varepsilon_1y)S]$ by Lemma 4.3.5.

□

Lemma 4.3.7 gives the time complexity of the algorithm. The running times depends on several parameters.

**Lemma 4.3.7.** *The algorithm ApproximateUnion(.) runs in* $\mathrm{O}\left(\frac{mf_4(m)f_6(m)}{z_{\min}}\cdot\left(\frac{\log\frac{z_{\max}}{z_{\min}}}{\log f_1(m)}\right)\right)$ *time.*

*Proof.* Let $y$ be the total number of stages. By Lemma 4.3.2, we have $y = \mathrm{O}\left(\frac{\log \frac{z_{max}}{z_{min}}}{\log f_1(m)}\right)$.

The time of each stage is $t_i = h_i \cdot u_i = h_1 f_6(m) = \frac{m}{z_{\min}} f_4(m) f_6(m)$, which is mainly from line 12 of the algorithm. Therefore, the total time is $\sum_{i=1}^{y} t_i \leq \frac{m}{z_{\min}} \cdot f_4(m) f_6(m) y$.

□

We have Theorem 4.3.8 to show the performance of the algorithm. The algorithm is sublinear if $minThickness(L) \geq m^a$ for a fixed $a > 0$, and has a $z_{\min}$ with $minThickness(L) \geq z_{\min} \geq m^b$ for a positive fixed $b$ ($b$ may not be equal to $a$) to be part of input to the algorithm.

**Theorem 4.3.8.** *The algorithm ApproximateUnion(.) takes* $\mathrm{O}\left(\frac{m f_4(m) f_6(m)}{z_{\min}} \cdot \left(\frac{\log \frac{z_{\max}}{z_{\min}}}{\log f_1(m)}\right)\right)$ *time and* $\mathrm{O}\left(\frac{\log \frac{z_{\max}}{z_{\min}}}{\log f_1(m)}\right)$ *rounds such that with probability at least* $1-\gamma$*, it gives a*

$$sum \cdot M \in [(1-\varepsilon)(1-\alpha_L)(1-\beta_L) \cdot |A_1 \cup \cdots \cup A_m|, (1+\varepsilon)(1+\alpha_R)(1+\beta_R) \cdot |A_1 \cup \cdots \cup A_m|],$$

*where* $z_{\min}$ *and* $z_{\max}$ *are parameters with* $1 \leq z_{\min} \leq minThickness(L) \leq maxThickness(L)$ $\leq z_{\max} \leq m$*, where functions* $f_4(.)$ *and* $f_1(.)$ *are defined in equations (4.4), and (4.7), respectively.*

*Proof.* Let $y$ be the number of stages. By Lemma 4.3.3, with probability at least $1-\gamma_1$,

$$\begin{aligned} F(R_1, h_1, 1)\left(\sum_{i=1}^{m} m_i\right) \in [ \quad & (1-\varepsilon_0)(1-\alpha_L)(1-\beta_L)|A_1 \cup \cdots \cup A_m|, \\ & (1+\varepsilon_0)(1+\alpha_R)(1+\beta_R)(1+\delta)|A_1 \cup \cdots \cup A_m|]. \end{aligned}$$

By Lemma 4.3.4, with probability at least $1 - y\gamma_2$, $sum \in [(1-\varepsilon_1)S', (1+\varepsilon_1)S']$, where $S' = \sum_{i=1}^{y} s_i' F'(V_i)$.

By Lemma 4.3.6, with probability at least $1 - 2y\gamma_2$,

$$\begin{aligned} sum \cdot M \in [ \quad & (1-y\varepsilon_1)(1-\varepsilon_0)(1-\varepsilon_1)(1-\alpha_L)(1-\beta_L) \cdot |A_1 \cup \cdots \cup A_m|, \\ & (1+2y\varepsilon_1)(1+\varepsilon_0)(1+\varepsilon_1)(1+\alpha_R)(1+\beta_R)(1+\delta)|A_1 \cup \cdots \cup A_m|]. \end{aligned}$$

Now assume

$$\begin{aligned} sum \cdot M \in [ \quad & (1-y\varepsilon_1)(1-\varepsilon_0)(1-\varepsilon_1)(1-\alpha_L)(1-\beta_L)\cdot|A_1\cup\cdots\cup A_m|, \\ & (1+2y\varepsilon_1)(1+\varepsilon_0)(1+\varepsilon_1)(1+\alpha_R)(1+\beta_R)(1+\delta)|A_1\cup\cdots\cup A_m|]. \end{aligned}$$

By Statement 4.3.5 of Lemma 4.3.5, we have $1-\varepsilon \le 1-y\varepsilon_1-\varepsilon_0-\varepsilon_1 \le (1-y\varepsilon_1)(1-\varepsilon_0)(1-\varepsilon_1)$, and $(1+2y\varepsilon_1)(1+\varepsilon_0)(1+\varepsilon_1)(1+\delta) \le (1+2y\varepsilon_1)(1+2\varepsilon_0+\varepsilon_1)(1+\delta) \le (1+2y\varepsilon_1)(1+4\varepsilon_0+2\varepsilon_1+\delta) \le (1+8\varepsilon_0+4\varepsilon_1+2\delta+2y\varepsilon_1) \le (1+8\varepsilon_0+4\varepsilon_1+\varepsilon_2+2y\varepsilon_1) \le (1+8\varepsilon_0+\frac{\varepsilon_0}{3}+\frac{\varepsilon_0}{3}+\frac{\varepsilon_0}{3}) \le 1+9\varepsilon_0 \le 1+\varepsilon$. Therefore,

$$sum \cdot M \in [(1-\varepsilon)(1-\alpha_L)(1-\beta_L)\cdot|A_1\cup\cdots\cup A_m|,(1+\varepsilon)(1+\alpha_R)(1+\beta_R)\cdot|A_1\cup\cdots\cup A_m|].$$

The algorithm may fail at the case after selecting $R_1$, or one of the stages. By the union bound, the failure probability is at most $\gamma_1+2\gamma_2\cdot\log m \le \gamma$. We have that with probability at least $1-\gamma$ to output the sum that satisfies the accuracy described in the theorem. The running time and the number of rounds of the algorithm follow from Lemma 4.3.7 and Lemma 4.3.2, respectively.

□

Since $1 \le z_{\min} \le minThickness(L) \le maxThickness(L) \le z_{\max} \le m$, we have the following Corollary 4.3.8.1. Its running time is almost linear in the classical model.

**Corollary 4.3.8.1.** *There is a* $O(poly(\frac{1}{\varepsilon},\log\frac{1}{\gamma})\cdot m\cdot(\log m)^{O(1)})$ *time and* $O(\log m)$ *rounds algorithm for* $|A_1\cup A_2\cup\cdots A_m|$ *such that with probability at least* $1-\gamma$*, it gives*

$$sum \cdot M \in [(1-\varepsilon)(1-\alpha_L)(1-\beta_L)\cdot|A_1\cup\cdots\cup A_m|,(1+\varepsilon)(1+\alpha_R)(1+\beta_R)\cdot|A_1\cup\cdots\cup A_m|].$$

*Proof.* We let $f_1(m)=8$ with $c_1=0$ in equation (4.4). Let $z_{min}=1$ and $z_{max}=m$. It follows from Theorem 4.3.8 and Statement 5 of Lemma 4.3.1. □

**Corollary 4.3.8.2.** *For each* $\xi > 0$*, there is a* $O(poly(\frac{1}{\varepsilon},\log\frac{1}{\gamma})\cdot m^{1+\xi})$ *time and* $O(\frac{1}{\xi})$ *rounds* *algorithm for* $|A_1 \cup A_2 \cup \cdots A_m|$ *such that with probability at least* $1-\gamma$*, it gives a sum* $\cdot M \in [(1-\varepsilon)(1-\alpha_L)(1-\beta_L) \cdot |A_1 \cup \cdots \cup A_m|, (1+\varepsilon)(1+\alpha_R)(1+\beta_R) \cdot |A_1 \cup \cdots \cup A_m|]$.

*Proof.* We let $f_1(m) = 8m^{\xi/2}$ with $c_1 = \frac{\xi}{2}$ in equation (4.4). Let $z_{min} = 1$ and $z_{max} = m$. It follows from Theorem 4.3.8 and Statement 5 of Lemma 4.3.1. □

An interesting open problem is to find an $\mathrm{O}(m)$ time and $\mathrm{O}(\log m)$ rounds approximation scheme for $|A_1 \cup A_2 \cup \cdots A_m|$ with a similar accuracy performance as Corollary 4.3.8.1. We were not able to adapt the method from Karp, Luby, and Madras [29] to solve this problem.

# CHAPTER V

# APPROXIMATE RANDOM SAMPLING FOR LATTICE POINTS IN HIGH DIMENSIONAL BALL

In this section, we propose algorithms to approximate the numebr of lattice points in high dimensional ball, and we also develop algorithms to generate a random lattice point inside a high dimensional ball.

Before present the algorithms, some definitions are given below.

**Definition 5.0.1.** *Let integer $d > 0$ be a dimensional number, $\mathbb{R}^d$ be the $d-$dimensional Euclidean Space*

- *For two points $p,\ q \in \mathbb{R}^d$, define $||p-q||$ to be Euclidean Distance.*
- *A point $p \in \mathbb{R}^d$ is a lattice point if $p = (y_1, ..., y_d)$ with $y_i \in \mathbb{Z}$ for $i = 1,2,...,d$.*
- *Let $p \in \mathbb{R}^d$, and $r > 0$. Define $B_d(r, p, d)$ be a $d-$dimensional ball of radius $r$ with center at $p$.*
- *For $q = (\mu_1, \mu_2, ..., \mu_d) \in \mathbb{R}^d$ with $\mu_i$ be real number for $i = 1,2,...,d$. Define $B_d(r, q, k) = \{(z_1, z_2, ..., z_d) \in \mathbb{R}^d : z_1 = \mu_1, ..., z_{d-k} = \mu_{d-k}$ and $\sum\limits_{i=1}^{d} (\mu_i - z_i)^2 \leq r^2\}$.*
- *Let $p \in \mathbb{R}^d$, and $r > 0$. Define $C(r, p, d)$ be the number of lattice points in the $d-$dimensional ball of radius $r$ with the center at $p$.*
- *Let $\lambda$, $l$ be real numbers. Define $D(\lambda, d, l) = \{(x_1, \cdots, x_d) : (x_1, \cdots, x_d)$ with $x_k = i_k + j_k\lambda$ for an integer $j_k \in [-l,\ l]$, and another arbitrary integer $i_k$ for $k = 1,2,...,d.\}$*

- *Let $\lambda$, $l$ be real numbers. Define $D^{*}(\lambda,d,l)=\{(x_1,\cdots,x_d):(x_1,\cdots,x_d)$ with $x_k=j_k\lambda$ for an integer $j_k\in[-l,\ l]$ with $k=1,2,...,d.\}$*

- *Let $\lambda=a^{-m}$, where $a$ and $m$ are integer and $a\geq 2$. Define $D^{**}(\lambda,d)=\{(x_1,\cdots,x_d):$ $(x_1,\cdots,x_d)$ with $x_k=i_k+j_k\lambda$ for an integer $j_k\in[-\lambda^{-1}+1,\lambda^{-1}-1]$, and another arbitrary integer $i_k$ for $k=1,2,...,d.\}$*

## 5.1 Randomized Algorithm for Approximating Lattice Points for High Dimensional Ball

In this section, we develop algorithms to approximate the number of lattice points in a $d$-dimensional ball $B_d(r,p,d)$. Two subsubsections are discussed below.

### 5.1.1 Counting Lattice Points of High Dimensional Ball with Small Radius

In this section, we develop a dynamic programming algorithm to count the number of lattice points in $d-$dimensional ball $B_d(r,p,d)$. Some definitions and lemmas that is used to prove the performance of algorithm are given before present the algorithm.

**Definition 5.1.1.** *Let $p$ be a point in $\mathbb{R}^d$, and $p\in D(\lambda,d,L)$. Define $E(r',p,h,k)$ be the set of $k-$dimensional balls $B_d(r',q,k)$ of radii $r'$ with center at $q=(y_1,y_2,...,y_h,x_{h+1},...,x_d)$ where $h=d-k$ is the number of initial integers of the center $q$ and $y_t\in\mathbb{Z}$ for $t=1,2,...,h$.*

Lemma 5.1.1 shows that for any two balls with same dimensional number, if their radii equal and the number of initial integers of their center also equal, then they have same number of lattice points.

**Lemma 5.1.1.** *For two $k-$dimensional balls $B_d(r,q,k)$ and $B_d(r,q',k)$, if $B_d(r,q,k)\in E(r,p,h,k)$ and $B_d(r,q',k)\in E(r,p,h,k)$, then $C(r,q,k)=C(r,q',k)$.*

*Proof.* In order to prove that $C(r,q,k)=C(r,q',k)$, we need to show that the set of lattice points inside ball $B_d(r,q,k)$ is one-one mapping to the set of lattice points inside ball $B_d(r,q',k)$, where $q=(y_1,y_2,...,y_h,x_{h+1},...,x_d)$ and $q'=(y'_1,y'_2,...,y'_h,x_{h+1},...,x_d)$ with $y'_t,y_t\in\mathbb{Z}$ for $t=1,2,...,h$.

Statement 1: $\forall\ q_1=(z_1,z_2,...,z_d)\in B_d(r,q,k)$, where $z_t\in\mathbb{Z}$ for $t=1,2,...,d$.

we have

$$(z_1-y_1)^2+\cdots+(z_h-y_h)^2+(z_{h+1}-x_{h+1})^2+\cdots+(z_d-x_d)^2\leq r^2$$

then

$$(z_1+y'_1-y_1-y'_1)^2+\cdots+(z_h+y'_h-y_h-y'_h)^2+(z_{h+1}-x_{h+1})^2+\cdots+(z_d-x_d)^2\leq r^2.$$

Therefore, there exists a lattice point $(z_1+y'_1-y_1,...,z_h+y'_h-y_h,z_{h+1},...,z_d)\in B_d(r,q',k)$ correspoding to $q_1$.

Statement 2: $\forall\ q'_1=(z'_1,z'_2,...,z'_d)\in B_d(r,q',k)$, where $z'_t\in\mathbb{Z}$ for $t=1,2,...,d$.

we have

$$(z'_1-y'_1)^2+\cdots+(z'_h-y'_h)^2+(z'_{h+1}-x_{h+1})^2+\cdots+(z'_d-x_d)^2\leq r^2$$

and

$$(z'_1-y'_1+y_1-y_1)^2+\cdots+(z'_h-y'_h+y_h-y_h)^2+(z'_{h+1}-x_{h+1})^2+\cdots+(z'_d-x_d)^2\leq r^2.$$

Therefore, there exists a lattice point $(z'_1-y'_1+y_1,...,z'_h-y'_h+y_h,z'_{h+1},...,z'_d)\in B_d(r,q,k)$ correspoding to $q'_1$.

Based on above two statements, there exists a one-one mapping between the set of lattice points inside ball $B_d(r,q,k)$ and the set of lattice points inside ball $B_d(r,q',k)$.

Therefore, $C(r,q,k)=C(r,q',k)$. □

Lemma 5.1.2 shows that we can move ball $B_d(r,q,k)$ by an integer units in every dimension without changing the number of lattice points in the ball.

**Lemma 5.1.2.** *Let $\lambda$ be a real number. For two $k-$dimensional balls $B_d(r,q_1,k)$ and $B_d(r,q_2,k)$, where $q_1=(y_1,y_2,...,y_{d-k},x_{d-k+1},...,x_d)$, $q_2=(y'_1,y'_2,...,y'_{d-k},x'_{d-k+1},...,x'_d)$ with $y_t,y'_t\in\mathbb{Z}$,*

*$t = 1,2,...,d-k$, and $x_{t'} = i_{t'} + j_{t'}\lambda$, $i_{t'}$ is an integer and $j_{t'} \in [-l,l]$ for $t' = d-k+1,...,d,$. If $x'_{t'} = j_{t'}\lambda$, then we have $C(r,q_1,k) = C(r,q_2,k)$.*

*Proof.* Since $B_d(r,q_1,k) \in E(r,p,h,k)$ and $B_d(r,q_2,k) \in E(r,p,h,k)$ with $h = d-k$, we have

$$C(r,q_1,k) = C(r,q_2,k)$$

via Lemma 5.1.1. □

We define $R(r,p,d)$ contains the set of radii $r'$ for the balls generated by the intersection of $B_d(r,p,d)$ wiht hyper-plane $x_1 = y_1$, ..., $x_k = y_k$, ..., $x_d = y_d$.

**Definition 5.1.2.** *For a $d-$dimensional ball $B_d(r,p,d)$ of radius r with center at $p = (x_1,x_2,...,x_d)$*

- *Define $R(r,p,d) = \{r' : r'^2 = r^2 - \sum_{i=1}^{k}(y_i - x_i)^2$ with $y_i \in \mathbb{Z}$ and $\sum_{i=1}^{k}(y_i - x_i)^2 \leq r^2$ $for\ some$ integer k$\in$ [1, d]$\}$.*

Lemma 5.1.3 shows that we can reduce the cardinality of $R(r,p,d)$ from exponentional to polynomial when setting the element of the ball's center has same type (i.e. $p \in D(\lambda,d,l)$.)

**Lemma 5.1.3.** *Let $B_d(r,p,d)$ be a $d-$dimensional ball of radius r with center at p, where $p \in D^*(\lambda,d,l)$. Then $|R(r,p,d)| \leq 4(r+l|\lambda|)^3 l^3 d^3$ and $R(r,p,d)$ can be generated in $\mathrm{O}\left((r+l|\lambda|)^3 l^3 d^3\right)$ time.*

*Proof.* Since $r'^2 = r^2 - \sum_{i=1}^{k}(y_i - x_i)^2$ for $0 \leq k \leq d$, we have $r'$ as:

$$\begin{aligned}
r'^2 &= r^2 - (y_1 - j_1\lambda)^2 - \cdots - (y_d - j_d\lambda)^2 \\
&= r^2 - [y_1^2 - 2y_1 j_1\lambda + j_1^2\lambda^2] - \cdots - [y_d^2 - 2y_d j_d\lambda + j_d^2\lambda^2] \\
&= r^2 - \{y_1^2 + y_2^2 + \cdots + y_d^2\} \\
&+ \{2y_1 j_1 + 2y_2 j_2 + \cdots + 2y_d j_d\}\lambda \\
&- \{j_1^2 + j_2^2 + j_3^2 + \cdots + j_d^2\}\lambda^2.
\end{aligned}$$

Let $R' = \{r' | r'^2 = r^2 - (x + y\lambda + z\lambda^2)\ with\ x,\ y,\ and\ z\ is\ nonnegative\ integer\}$, it is easy to see that $r' \in R'$ then $R \subseteq R'$.

Let

$$\begin{cases} X = \{x' | x' = y_1^2 + y_2^2 + ... + y_d^2\ with\ y_i \in [r - l|\lambda|, r + l|\lambda|],\ 0 \leq i \leq d\}; \\ Y = \{y' | y' = 2y_1 j_1 + 2y_2 j_2 + ... + 2y_d j_d\ with\ y_i j_i \in [I(r - l|\lambda|), I(r + l|\lambda|)],\ 0 \leq i \leq d\}; \\ Z = \{z' | z' = j_1^2 + j_2^2 + j_3^2 + ... + j_d^2\ with\ j_i \in [-l, l],\ 0 \leq i \leq d,\}, \end{cases}$$

then we have:

$$\begin{cases} |Z| \leq dl^2; \\ |Y| \leq 4d(r + l|\lambda|)l; \\ |X| \leq d(r + l|\lambda|)^2. \end{cases} \tag{5.1}$$

For each $r' \in R$, we have $r'^2 = r^2 - (x + y\lambda + z\lambda^2)$ with $x \in X$, $y \in Y$, and $z \in Z$. Therefore, $|R| \leq dl^2 \cdot 4d(r + l|\lambda|)l \cdot d(r + l|\lambda|)^2 = 4(r + l|\lambda|)^3 l^3 d^3$ via inequality (5.1). Then $R(r, p, d)$ can be generated in $\mathcal{O}\left((r + l|\lambda|)^3 l^3 d^3\right)$ time.

□

Lemma 5.1.4 is a spacial case of Lemma 5.1.3. It shows that there at most $(r^2 + 1)a^{2m}$ cases of the radii when the elements of the center are the type like fractions in base $a$. For example, $p = (3.891, 5.436, ..., 5.743) \in \mathbb{R}^d$.

**Lemma 5.1.4.** *Let $\lambda = a^{-m}$ where $a$ is a interger with $a \geq 2$. Let $B_d(r, p, d)$ be a $d-$dimensional ball of radius $r$ with center at $p \in D^{**}(\lambda, d)$. Then $|R(r, p, d)| \leq (r^2 + 1)a^{2m}$ and $R(r, p, d)$ can be generated in $\mathcal{O}\left((r^2 + 1)a^{2m}\right)$ time.*

*Proof.* We have

$$\begin{aligned}
r'^2 \quad &= r^2 - (y_1 - j_1\lambda)^2 - \cdots - (y_d - j_d\lambda)^2 \\
&= r^2 - [y_1^2 - 2y_1 j_1\lambda + j_1^2\lambda^2] - \cdots - [y_d^2 - 2y_d j_d\lambda + j_d^2\lambda^2] \\
&= r^2 - \{y_1^2 + y_2^2 + \cdots + y_d^2\} \\
&+ \{2y_1 j_1 + 2y_2 j_2 + \cdots + 2y_d j_d\}\lambda \\
&- \{j_1^2 + j_2^2 + j_3^2 + \cdots + j_d^2\}\lambda^2.
\end{aligned}$$

via Lemma 5.1.3.

For each $r'^2$, it can be transformed into $r'^2 = r^2 - (x + y\lambda + z\lambda^2)$ with $x, y$ and $z$ are integers,

and

$$\begin{cases} |z| \le a^m; \\ |y| \le a^m; \\ |x| \le (r^2+1). \end{cases} \tag{5.2}$$

Therefore, $|R| \le (r^2+1)a^{2m}$ via inequality (5.2). Then $R(r,p,d)$ can be generated in $\mathscr{O}\left((r^2+1)a^{2m}\right)$ time.

□

**Definition 5.1.3.** *For a $d-$dimensional ball $B_d(r,p,d)$ of radius $r$ with center at $p = (x_1, x_2, ..., x_d)$*

- *Define $p[k] = (0, ..., 0, x_{k+1}, ..., x_d)$ for some integer $k \in [1,d]$.*
- *Define $Z(r,x,t)$ with $Z(r,x,t)^2 = r^2 - (t-x)^2$ if $|t-x| \le r$, where $t$ is a integer and $x \in \mathbb{R}$.*

We give a dynamic programming algorithm to count the number of lattice points in a $d-$dimensional ball $B_d(r,p,d)$.

**Algorithm 2** CountLatticePoint($r$, $p$, $d$)

**Input** : $p = (x_1, x_2, ..., x_d)$ where $x_k = i_k + j_k\lambda$ for an integer $j_k \in [-l, l]$, and another arbitrary integer $i_k$ for $k = 1, 2, ..., d$. $r$ is radius and $d$ is dimensional numbers.

**Output** : The number of lattice points of the $d-$dimensional ball $B_d(r, p, d)$.

```
1:    Let $r_0 = r$;
2:    For $k = d - 1$ to 0
3:        for each $r_k \in R(r, p, d)$
4:            let $C(r_k, p[k], d - k) = \sum_{t \in \mathbb{Z} \text{ and } t \in [-r_k + x_{k+1},\ r_k + x_{k+1}]} C(z(r_k, x_{k+1}, t), p[k+1], d - (k+1))$;
5:            save $C(r_k, p[k], d - k)$ to the look up table;
6:    Return $C(r_0, p[0], d)$.
```

We note that if $d-(k+1) = 0$ then $C(z(r_k, x_{k+1}, t), p[k+1], d-(k+1)) = 1$, otherwise $z(r_k, x_{k+1}, t)$ is in $R(r, p, d)$ (i.e. $C(z(r_k, x_{k+1}, t), p[k+1], d-(k+1))$ is avaiable in the table).

**Theorem 5.1.5.** *Assume $\lambda$ be a real number and $p \in D(\lambda, d, l)$. Then there is a $\mathcal{O}(r(r+l|\lambda|)^3 l^3 d^4)$ time algorithm to count $C(r, p, d)$.*

*Proof.* Line 2 has $d$ iterations, Line 3 takes $4(r+l|\lambda|)^3 l^3 d^3$ to compute $r_k$ via Lemma 5.1.3, and Line 4 has at most $2\lfloor r \rfloor + 1$ items to add up.

Therefore, the algorithm CountLatticePoints(.) takes $\mathcal{O}(r(r+l|\lambda|)^3 l^3 d^4)$ running time.

□

**Remark:** When $\lambda = \frac{1}{\pi}$, this is a specail case of Theorem 5.1.5, and the running time of the algorithm is $\mathcal{O}(r(r+l|\lambda|)^3 l^3 d^4)$. The algorithm can count the lattice points of high dimensional ball if the element of the center of the ball has same type like $i + j\lambda$ even though $\lambda$ is a irrational number.

Theorem 5.1.6 shows that the algorithm can count the number of lattice points of high dimensional ball if the element of the center of the ball has same type like fractions in base $a$.

**Theorem 5.1.6.** *Assume $\lambda = a^{-m}$ and $p \in D^{**}(\lambda, d)$, where m and a are integers with $a \geq 2$. Then there is a $\mathscr{O}(r^3 a^{2m} d)$ time algorithm to count $C(r, p, d)$.*

*Proof.* Line 2 has $d$ iterations, Line 3 takes $(r^2+1)a^{2m}$ to compute $r_k$ via Lemma 5.1.4, and Line 4 has at most $2\lfloor r \rfloor + 1$ items to add up.

Therefore, the algorithm CountLatticePoints(.) takes $\mathscr{O}(rd(r^2+1)a^{2m})$ running time.

□

**Corollary 5.1.6.1.** *Assume $\lambda = 10^{-m}$ and $p \in D^{**}(\lambda, d)$, where m is a integer. Then there is a $\mathscr{O}(r^3 10^{2m} d)$ time algorithm to count $C(r, p, d)$.*

### 5.1.2 Approximating Lattice Points in High Dimensional Ball with Large Radius

In this section, we present an $(1+\beta)$-approximation algorithm to approximate the number of lattice points in a $d-$dimensional ball $B_d(r, p, d)$ of large radius with an arbitrary center $p$, where $\beta$ is used to control the accuracy of approximation.

Some definitions are presented before prove theorems.

**Definition 5.1.4.** *For each lattice point $q = (y_1, y_2, ..., y_d) \in \mathbb{R}^d$ with $y_i \in \mathbb{Z}$ for $i = 1, 2, ..., d$*

- *define $Cube(q)$ to be the $d-$dimensional unit cube with center at $\left(y_1 + \frac{1}{2}, ..., y_d + \frac{1}{2}\right)$,*
- *define $I(B_d(r, p, d)) = \{q \mid Cube(q) \subset B_d(r, p, d)\}$,*
- *defien $E(B_d(r, p, d)) = \{q \mid Cube(q) \notin I(B_d(r, p, d))$ and $Cube(q) \cap B_d(r, p, d) \neq \emptyset\}$.*

Theorem 5.1.7 gives an $(1+\beta)-$approximation with running time $\mathscr{O}(d)$ algorithm to approximate the number of lattice point $C(r, p, d)$ with $p$ is an arbitrary center and $r > \frac{2d^{\frac{3}{2}}}{\beta}$.

**Theorem 5.1.7.** *For an arbitrary $\beta \in (0, 1)$, there is a $(1+\beta)-$approximation algorithm to compute $C(r, p, d)$ of $d-$dimensional ball $B_d(r, p, d)$ with running time $\mathscr{O}(d)$ for an arbitrary center p when $r > \frac{2d^{\frac{3}{2}}}{\beta}$.*

*Proof.* Let $|I(B_d(r,p,d))|$ is the number of lattice points $q \in I(B_d(r,p,d))$, $|E(B_d(r,p,d))|$ be the number of lattice points $q \in E(B_d(r,p,d))$, and $V_d(r)$ be the volume of a $d-$dimensional ball with radius $r$.

Now consider two $d-$dimensional balls $B_d(r-\sqrt{d},p,d)$ and $B_d(r+\sqrt{d},p,d)$ that have the same center as ball $B_d(r,p,d)$. Since every lattice point $q$ corresponds to a $Cube(q)$ via Definition 5.1.4, and the volume of the ball equals the sum of $Cube(q)$ that is contained by the ball, then we have:

$$\begin{cases} V_d(r-\sqrt{d}) \leq |I(B_d(r,p,d))| \leq V_d(r) \\ 0 \leq |E(B_d(r,p,d))| \leq V_d(r+\sqrt{d}) - V_d(r). \end{cases}$$

Therefore,

$$V_d(r-\sqrt{d}) \leq C(r,p,d) = |I(B_d(r,p,d))| + |E(B_d(r,p,d))| \leq V_d(r+\sqrt{d}).$$

Then the bias is $\frac{|I(B_d(r,p,d))|+|E(B_d(r,p,d))|}{V_d(r)}$ when using $V_d(r)$ to approximate $C(r,p,d)$.

The volume formula for a $d-dimensional$ ball of raduis $r$ is

$$V_d(r) = f(d) \cdot r^d$$

where $f(d) = \pi^{\frac{d}{2}} \Gamma\left(\frac{1}{2}d+1\right)^{-1}$ and $\Gamma(.)$ is Leonhard Euler's gamma function. Then

$$\begin{aligned} \frac{|I(B_d(r,p,d))|+|E(B_d(r,p,d))|}{V_d(r)} &\leq \frac{V_d(r+\sqrt{d})}{V_d(r)} \\ &= \frac{f(d)\cdot(r+\sqrt{d})^d}{f(d)\cdot r^d} \\ &= \left(1+\frac{\sqrt{d}}{r}\right)^d \\ &\leq e^{\frac{d^{\frac{3}{2}}}{r}} \\ &\leq 1+\frac{2d^{\frac{3}{2}}}{r}. \end{aligned}$$

Similarly, we have:

$$\begin{aligned}\frac{|I(B_d(r,p,d))|+|E(B_d(r,p,d))|}{V_d(r)} &\geq \frac{V_d(r-\sqrt{d})}{V_d(r)}\\ &= \frac{f(d)\cdot(r-\sqrt{d})^d}{f(d)\cdot r^d}\\ &= \left(1-\frac{\sqrt{d}}{r}\right)^d\\ &\geq 1-\frac{d^{\frac{3}{2}}}{r}\\ &\geq 1-\frac{2d^{\frac{3}{2}}}{r}\end{aligned}$$

From above two inequalities, we have:

$$\left(1-\frac{2d^{\frac{3}{2}}}{r}\right)\cdot V_d(r) \leq C(r,p,d) \leq \left(1+\frac{2d^{\frac{3}{2}}}{r}\right)\cdot V_d(r),$$

then we have:

$$\frac{1}{1+\frac{2d^{\frac{3}{2}}}{r}}\cdot C(r,p,d) \leq V_d(r) \leq \frac{1}{1-\frac{2d^{\frac{3}{2}}}{r}}\cdot C(r,p,d).$$

Simplify the above inequality, we have

$$\left(1-\frac{2d^{\frac{3}{2}}}{r-2d^{\frac{3}{2}}}\right)C(r,p,d) \leq V_d(r) \leq \left(1+\frac{2d^{\frac{3}{2}}}{r-2d^{\frac{3}{2}}}\right)C(r,p,d).$$

Thus, we have

$$(1-\beta)C(r,p,d) \leq V_d(r) \leq (1+\beta)C(r,p,d) \tag{5.3}$$

with $\beta > \frac{2d^{\frac{3}{2}}}{r-2d^{\frac{3}{2}}}$.

It takes $\mathscr{O}(d)$ to compute $V_d(r) = f(d)\cdot r^d$, since it takes $\mathscr{O}(d)$ to compute $f(d)$ where $f(d) = \pi^{\frac{d}{2}}\Gamma\left(\frac{1}{2}d+1\right)^{-1}$. Thus, the algorithm takes $\mathscr{O}(d)$ running time to approximate $C(r,p,d)$. becasue of Equation (5.3). □

**Theorem 5.1.8.** *There is an $(1+\beta)$-approximation algorithm with running time $\mathscr{O}(d)$ to approx-*

*imate $C(r,p,d)$ of $B_d(r,p,d)$ with an arbitrry center $p$ when $r > \frac{2d^{\frac{3}{2}}}{\beta}$; and there is an dynamic programming algorithm with running time $\mathscr{O}\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)$ to count $C(r,p,d)$ with center $p \in D(\lambda,d,l)$ when $r \leq \frac{2d^{\frac{3}{2}}}{\beta}$.*

*Proof.* We discuss two cases based the radius of the $d$-dimensional ball.

Case 1: When counting the number of lattice points of a $d$-dimensional ball with center $p \in D(\lambda,d,l)$ for $r \leq \frac{2d^{\frac{3}{2}}}{\beta}$, apply Theorem 5.1.5,

Case 2: When approximating the number of lattice points of a $d$-dimensional ball with an arbitrary center $p$ for $r > \frac{2d^{\frac{3}{2}}}{\beta}$, apply Theorem 5.1.7. □

**Corollary 5.1.8.1.** *There is a dynamic programming algorithm to count $C(r,p,d)$ of $B_d(r,p,d)$ with running time $\mathscr{O}\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)$ for $p \in D(\lambda,d,l)$ when $r \leq \frac{2d^{\frac{3}{2}}}{\beta}$.*

## 5.2 A Randomized Algorithm for Generating Random Lattice Point of High Dimensional Ball

In this section, we propose algorithms to generate a random lattice point inside a high dimensional ball. Two subsubsections are discussed below.

### 5.2.1 Generating a Random Lattice Point inside High Dimensional Ball with Small Radius

In this section, we develop a recursive algorithm to generate a random lattice point inside a $d-$dimensional ball $B_d(r,p,d)$ of small radius with center $p \in D(\lambda,d,l)$.

The purpose of the algorithm RecursiveSmallBallRandomLatticePoint$(r,p,t,d)$ is to recursively generate a random lattice point in the ball $B_d(r,p,t)$.

**Algorithm 3** RecursiveSmallBallRandomLatticePoint(r, p, t, d)

**Input** : $p = (y_1, y_2, ..., y_{d-t}, x_{d-t+1}, ..., x_d)$ where $x_k = i_k + j_k\lambda$ with arbitrary integer $i_k$, integer $j_k \in [-l, l]$, and $y_i \in \mathbb{Z}, i = 1, 2, ..., d-t$, $t$ is a dimension number with $0 \leq t \leq d$.

**Output** : Generate a random lattice point inside $t-$dimensional ball.

```
1:  Save $C(r_k, p[k], d - k)$ into look up table C-Table by using Algorithm $CountLatticePoint(r, p, d)$ for $k = 0, 1, ..., d-1$.
2:  If $t = 0$
3:      Return lattice point $(y_1, y_2, ..., y_d)$.
4:  Else
5:      Return RecursiveSmallBallRandomLatticePoint$(r', q, t-1, d)$ with probability $\frac{C(r',q,t-1)}{C(r,p,t)}$, where $q = (y_1, y_2, ..., y_{d-t}, y_{d-t+1}, x_{d-t+2}, ..., x_d)$ with $y_{d-t+1} \in [x_{d-t+1} - r, x_{d-t+1} + r]$ satisfying $||p-q||^2 \leq r^2$, and $r'^2 = r^2 - ||p-q||^2$;
```

We note that $C(.,.,.)$ is available at C-Table in O(1) step and the implementation of line 5 of the algorithm is formally defined below: Partition $I = [1,\ C(r, p, t)] \cap \mathbb{Z}$ into $I_1, \cdots, I_w$, where $I_i$ is uniquely corresponds to an integer $y_{d-t+1} \in [x_{d-t+1} - r, x_{d-t+1} + r]$ satisfying

$q = (y_1, y_2, ..., y_{d-t}, y_{d-t+1}, x_{d-t+2}, ..., x_d)$, $||p-q||^2 \leq r^2$, and $|I_i| = C(r', q, t-1)$. Generate a random number $z \in I$. If $z \in I_i$ ($I_i$ is mapped to $y_{d-t+1}$), then it returns RecursiveSmallBallRandomLatticePoint $(r', q, t-1, d)$ with $q = (y_1, ..., y_{d-t+1}, x_{d-t+2}, ..., x_d)$.

The algorithm RandomSmallBallLatticePoint$(r, p, d)$ is to generate a random lattice point in the ball $B_d(r, p, d)$. It calls the function RecursiveSmallBallRandomLatticePoint(.).

**Algorithm 4** RandomSmallBallLatticePoint(r, p, d)

**Input** : $p = (x_1, x_2, ..., x_d)$ where $x_k = i_k + j_k\lambda$ with arbitrary integer $i_k$, integer $j_k \in [-l, l]$ for $k = 1, 2, \cdots, d$.

**Output** : Generate a random lattice point inside $d-$dimensional ball.

```
1:  Return RecursiveSmallBallRandomLatticePoint$(r, p, d, d)$
```

**Theorem 5.2.1.** *For an arbitrary $\beta \in (0, 1)$. Assume $\lambda$ be a real number and $p \in D(\lambda, d, l)$.*

*Then there is a* $\mathcal{O}\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)$ *time algorithm to generate a lattice point inside a* $d-$*dimensional ball* $B_d(r,p,d)$*.*

*Proof.* By algorithm RandomSmallBallLatticePoint(.), we can generate a random lattice point inside $d-$dimensional ball $B_d(r,p,d)$ with probability $\frac{C(r',q,d-1)}{C(r,p,d)}\cdot\frac{C(r'',q',d-2)}{C(r',q,d-1)}\cdot\ldots\cdot\frac{1}{C(r^{(d-1)},q^{(d-1)},0)}=\frac{1}{C(r,p,d)}$.

It takes $\mathcal{O}\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)$ to compute $C(r,p,d)$ via Theorem 5.1.8, then algorithm SmallBallRandomLatticePoint(.) takes $\mathcal{O}\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)+\mathcal{O}(d)$ running time. Thus, the algorithm takes $\mathcal{O}\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)$ running time. □

### 5.2.2 Generating a Random Lattice Point of High Dimensional Ball with Large Radius

In this section, we develop an $(1+\alpha)-$approximation algorithm to generate a random lattice point inside a $d-$dimensional ball $B_d(r,p,d)$ of large radius $r$ with arbitrary center $p$, where $\alpha$ is used to control the accuracy of approximation.

We first propose an approximation algorithm RecursiveBigBallRandomLatticePoint(.) to generate a random lattice point inside a $d-$dimensional ball $B_d(r,p,d)$ of radius $r$ with lattice point center $p$, then we apply algorithm RecursiveBigBallRandomLatticePoint(.) to design algorithm BigBallRandomLatticePoint(.) to generate an approximate random lattice point in a $d-$dimensional ball $B_d(r',p,d)$ of radius $r'$ with arbitrary center $p$.

Before present the algorithms, we give some definition and lemmas that is used to analysis algorithm RecursiveBigBallRandomLatticePoint(.).

**Definition 5.2.1.** *For an arbitrary* $\beta\in(0,1)$*. Let* $B_d(r,q,k)$ *be* $k-$*dimensional ball of radius* $r$ *with arbitrary center* $q$*. Define* $P(r,q,k)$ *as*

$$P(r,q,k)=\begin{cases} C(r,q,k) & r\le\frac{2d^{\frac{3}{2}}}{\beta}\\ V_k(r) & otherwise,\end{cases}$$

*where $C(r,q,k)$ is the number of lattice point of $k-$dimensional ball $B_d(r,q,k)$ and $V_k(r)$ is the volume of ball $B_d(r,q,k)$.*

Lemma 5.2.2 shows that we can use $P(r,q,k)$ to approximate $C(r,q,k)$ for $k-$dimensional ball $B_d(r,q,k)$ no matter how much the radius $r$ it is.

**Lemma 5.2.2.** *For an arbitrary $\beta \in (0,1)$. Let $B_d(r,q,k)$ be $k-$dimensional ball of radius $r$ with arbitrary center $q$, then $(1-\beta)C(r,q,k) \leq P(r,q,k) \leq (1+\beta)C(r,q,k)$.*

*Proof.* Two cases are considered.

Case 1: If $r \leq \frac{2d^{\frac{3}{2}}}{\beta}$, we have $P(r,q,k) = C(r,q,k)$ via Definition 5.2.1.

Case 2: If $r > \frac{2d^{\frac{3}{2}}}{\beta}$, we have:

$$(1-\beta)\cdot C(r,q,k) \leq V_k(r) \leq (1+\beta)\cdot C(r,q,k)$$

via Theorem 5.1.7, where $V_k(r)$ be the volume of $k-$dimensional ball $B_d(r,q,k)$ with radius $r$.

Therefore, we have

$$(1-\beta)\cdot C(r,q,k) \leq P(r,q,k) \leq (1+\beta)\cdot C(r,q,k),$$

because $P(r,q,k) = V_k(r)$ via Definition 5.2.1.

Combined the above two cases, we conclude that:

$$(1-\beta)C(r,q,k) \leq P(r,q,k) \leq (1+\beta)C(r,q,k).$$

□

Lemma 5.2.3 shows that for two $k-$dimensional balls, if their radius almost equal, then the number of their lattice points also almost equal, we can use $\varepsilon_5$ to control the radii, and use $\beta, \varepsilon_5$ to control the number of lattice points.

**Lemma 5.2.3.** *For an arbitrary $\beta \in (0,1)$ and a real number $\delta$. Let $B_d(r',q,k)$ be a $k-$dimensional ball of radius $r'$ with lattice center at $q$ and $B_d(r'',q,k)$ be a $k-$dimensional ball of radius $r'' > \frac{2d^{\frac{3}{2}}}{\beta}$ with lattice center at $q$, where $q = (y_1, y_2, ..., y_d)$ with $y_t \in \mathbb{Z}$ and $t = 1,2,...,d$. If $r'' \leq r' \leq (1+\delta)\,r''$, then $C(r'',q,k) \leq C(r',q,k) \leq \frac{1+\beta}{1-\beta}\,(1+\delta)^k C(r'',q,k)$.*

*Proof.* Let $V_d(r)$ be the volume of $d-$dimensional ball of radius $r$. Since the volume formula for a $d-dimensional$ ball of raduis $r$ is

$$V_d(r) = f(d) \cdot r^d$$

where $f(d) = \pi^{\frac{d}{2}}\Gamma\left(\frac{1}{2}d+1\right)^{-1}$ and $\Gamma(.)$ is Euler's gamma function. Then, we have the following as:

$$V_k(r'') \leq V_k(r') \leq V_k(r'') \cdot (1+\delta)^k.$$

Since $r'' > \frac{2d^{\frac{3}{2}}}{\beta}$, $r' \geq r'' > 2\frac{d^{\frac{3}{2}}}{\beta}$, then we have

$$\begin{cases} \frac{1}{1+\beta}V_k(r') \leq C(r',q,k) \leq \frac{1}{1-\beta}V_k(r') \\ \frac{1}{1+\beta}V_k(r'') \leq C(r'',q,k) \leq \frac{1}{1-\beta}V_k(r'') \end{cases} \tag{5.4}$$

via Theorem 5.1.7,

Plugging inequality (5.4) to above inequality, then we have:

$$\begin{aligned} C(r',q,k) \quad & \leq \frac{1}{1-\beta}V_k(r') \\ & \leq \frac{1}{1-\beta}V_k(r'') \cdot (1+\delta)^k \\ & = \frac{(1+\beta)}{(1-\beta)}\frac{1}{(1+\beta)}V_k(r'') \cdot (1+\delta)^k \\ & \leq \frac{(1+\beta)}{(1-\beta)} \cdot (1+\delta)^k C(r'',q,k) \end{aligned}$$

and we also have:

$$C(r',q,k) \geq C(r'',q,k).$$

Therefore:

$$C(r'',q,k) \le C(r',q,k) \le \frac{1+\beta}{1-\beta}(1+\delta)^k C(r'',q,k).$$

□

**Definition 5.2.2.** *For an integer interval $[a,b]$, $c \in \mathbb{Z}$, $r > 0$, and $\delta \in (0,1)$, an $(r,c,1+\delta)$-partition for $[a,b]$ is to divide $[a,b]$ into $[a_1,b_1],[a_2,b_2],\cdots,[a_w,b_w]$ that satisfies the following conditions:*

- $a_1 = a, a_{i+1} = b_i + 1$ *for* $i = 1,\cdots,w-1$.
- *for any* $x,y \in \{a_i,b_i\}$, $r^2-(x-c)^2 \le (1+\delta)^2(r^2-(y-c)^2)$ *and* $r^2-(y-c)^2 \le (1+\delta)^2(r^2-(x-c)^2)$.
- *for any* $x \in \{a_i,b_i\}$ *and* $y \in \{a_{i+1},b_{i+1}\}$, $r^2-(x-c)^2 > (1+\delta)^2(r^2-y^2)$ *or* $r^2-(y-c)^2 > (1+\delta)^2(r^2-x^2)$.

The purpose of the algorithm RecursiveBigBallRandomLatticePoint(.) is to recursivly generate a random lattice point inside the $d-$dimensional ball $B_d(r,p,d)$ of radius $r$ with lattice point center $p$.

**Algorithm 5** RecursiveBigBallRandomLatticePoint(r, p, t, d)

**Input** : $p = (z_1, z_2, ..., z_{d-t}, y_{d-t+1}, ..., y_d)$ where $z_i \in \mathbb{Z}$ with $1 \le i \le d-t$, and $y_i \in \mathbb{Z}$ with $d-t+1 \le i \le d$, $\alpha \in (0,1)$ is a parameter to control the bias, $r$ is radius, and $t$ is dimensional number.

**Output** : $Z = \{z_1, ..., z_d\}$.

```
1:  If $t = 0$
2:      Return $(z_1, z_2, ..., z_d)$.
3:  Let $I_1 = [a_1, b_1], \cdots, I_w = [a_w, b_w]$ be the union of intervals via $\left(r, y_{d-t+1}, 1 + \frac{\varepsilon_4}{g(d)}\right)$-partitions for $[\lceil y_{d-t+1} - r \rceil, y_{d-t+1}] \cap \mathbb{Z}$ and $[y_{d-t+1} + 1, \lfloor y_{d-t+1} + r \rfloor] \cap \mathbb{Z}$, where $\varepsilon_4 \in (0,1)$ and $g(d)$ is a function of $d$.
4:  Let $M = \sum_{i=1}^{w} (b_i - a_i + 1) P(r_i, p_i, t-1)$, where $p_i = (z_1, z_2, ..., z_{d-t}, b_i, y_{d-t+2}, ..., y_d)$, and $r_i^2 = r^2 - (b_i - y_{d-t+1})^2$;
5:  Return RecursiveBigBallRandomLatticePoint$(r_i', p_i', t-1, d)$ with probability $\frac{P(r_i, p_i, t-1)}{M}$, where $z_{d-t+1} = b_i$, $p_i = (z_1, z_2, ..., z_{d-t}, z_{d-t+1}, y_{d-t+2}, ..., y_d)$, and $r_i^2 = r^2 - (z_{d-t+1} - y_{d-t+1})^2$, $p_i' = (z_1, z_2, ..., z_{d-t}, z_{d-t+1}', y_{d-t+2}, ..., y_d)$, and $r_i'^2 = r^2 - (z_{d-t+1}' - y_{d-t+1})^2$ and a random integer $z_{d-t+1}' \in [a_i, b_i]$;
```

We note that the implementation of $\left(r, y_{d-t+1}, 1 + \frac{\varepsilon_4}{g(d)}\right)$-partitions in line 3 is as the following pictures:

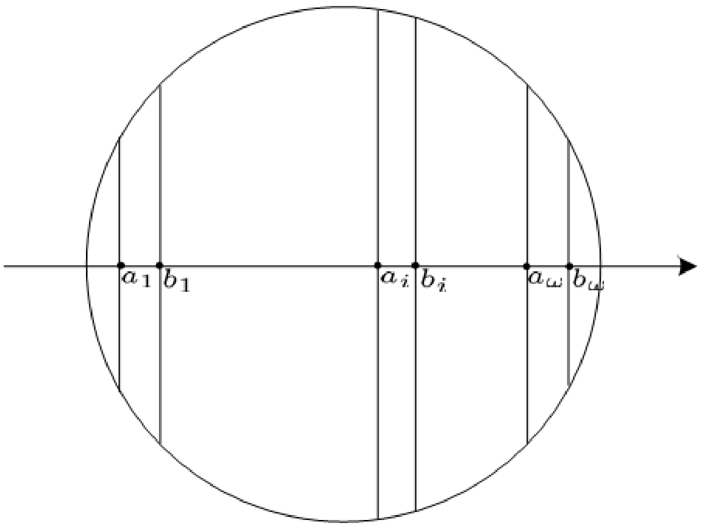

Figure 5.1: Example of $\left(r, y_{d-t+1}, 1+\frac{\varepsilon_4}{g(d)}\right)$-Partitions in 2D

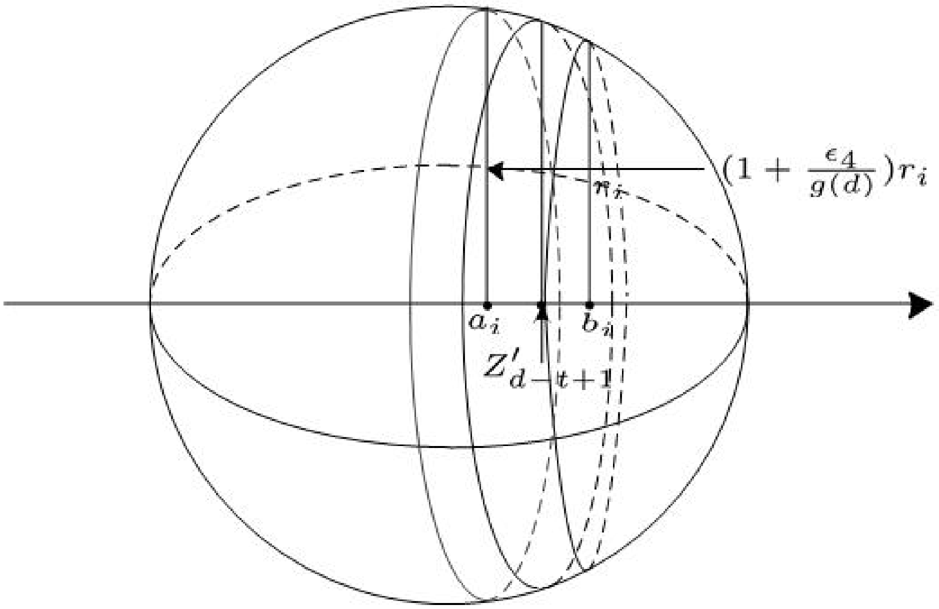

Figure 5.2: Example of $\left(r, y_{d-t+1}, 1+\frac{\varepsilon_4}{g(d)}\right)$-Partitions in 3D

We have the following algorithm that can generate an approximate random lattice point in a large ball with an arbitrary center, which may not be a lattice point.

**Definition 5.2.3.** *Let integer $d > 0$ be a dimensional number, $\mathbb{R}^d$ be the $d-$dimensional Euclidean Space*

- *A point $q = (x_1', x_2', ..., x_d') \in \mathbb{R}^d$ is the nearest lattice point of $p = (x_1, ..., x_d) \in \mathbb{R}^d$ if it satisfies $x_i' = \begin{cases} \lfloor x_i \rfloor & x_i - [x_i] \le \frac{1}{2} \\ \lceil x_i \rceil & x_i - [x_i] > \frac{1}{2}, \end{cases}$ for $x_i \ge 0$ or $x_i' = \begin{cases} \lceil x_i \rceil & |x_i| - [|x_i|] \le \frac{1}{2} \\ \lfloor x_i \rfloor & |x_i| - [|x_i|] < \frac{1}{2}, \end{cases}$ for $x_i < 0$, where $i = 1, 2, ..., d$.*

**Algorithm 6** BigBallRandomLatticePoint(r, p, d)

**Input** : $p = (x_1, \cdots, ..., x_d)$ where $x_i \in \mathbb{R}$ with $1 \le i \le d$, $\alpha \in (0,1)$ is a parameter to control the bias, $r$ is radius, and $k$ is dimensional number.

**Output** : Generate a random lattice point inside $d-$dimensional ball.

```
Let $q$ be the nearest lattice point of $p$ in $\mathbb{R}^d$;
Repeat
    Let $s$ =RecursiveBigBallRandomLatticePoint($r + \sqrt{d}, q, d$);
Until $s \in B_d(r, p, d)$;
Return $s$;
```

**Theorem 5.2.4.** *For an arbitrary $\alpha \in (0,1)$, there is an algorithm with runing time $\mathscr{O}\left(\frac{d^3 \log r}{\alpha}\right)$ and $(1+\alpha)-$bias for a $d-$dimensional ball $B_d(r, q, d)$ to generate a random lattice point with radius $r > \frac{2d^3}{\alpha}$ that centered at $q = (y_1, y_2, ..., y_d)$ with $y_t \in \mathbb{Z}$, $t = 1, 2, ..., d$.*

*Proof.* In line 5 of algorithm RecursiveBigBallRandomLatticePoint(.), define

$$r_i'^2 = \begin{cases} r^2 - (y_{d-t+1} - a_i)^2 & if \ a_i \le y_{d-t+1} \\ r^2 - (y_{d-t+1} - b_i)^2 & otherwise, \end{cases},$$

$$p_i' = \begin{cases} (z_1, z_2, ..., z_{d-t}, a_i, y_{d-t+2}, ..., y_d) & if \ a_i \le y_{d-t+1} \\ (z_1, z_2, ..., z_{d-t}, b_i, y_{d-t+2}, ..., y_d) & otherwise, \end{cases}$$

and

$$r_i^2 = \begin{cases} r^2 - (y_{d-t+1} - b_i)^2 & if\ b_i \leq y_{d-t+1} \\ r^2 - (y_{d-t+1} - a_i)^2 & otherwise, \end{cases}$$

$$p_i = \begin{cases} (z_1, z_2, ..., z_{d-t}, b_i, y_{d-t+2}, ..., y_d) & if\ b_i \leq y_{d-t+1} \\ (z_1, z_2, ..., z_{d-t}, a_i, y_{d-t+2}, ..., y_d) & otherwise, \end{cases}$$

let $v(i) = (b_i - a_i + 1)$, and $r'_i = \frac{r_i}{1+\frac{\varepsilon_4}{g(d)}}$.

Then we have:

$$\sum_i C(r'_i, p'_i, t-1)v(i) \leq C(r_i, p_i, t) \leq \sum_i C(r_i, p_i, t-1)v(i).$$

Since $r'_i = \frac{r_i}{1+\frac{\varepsilon_4}{g(d)}}$, then

$$\frac{1-\beta}{1+\beta}\left(1+\frac{\varepsilon_4}{g(d)}\right)^{-(t-1)} \sum_i C(r_i, p_i, t-1)v(i) \leq C(r_i, p_i, t)$$

and

$$C(r_i, p_i, t) \leq \sum_i C(r_i, p_i, t-1)v(i)$$

via Lemma 5.2.3, where $\delta = 1 + \frac{\varepsilon_4}{g(d)}$.

Via Lemma 5.2.2 we have:

$$\left(\frac{1-\beta}{1+\beta}\right)^2 \left(1+\frac{\varepsilon_4}{g(d)}\right)^{-(t-1)} \sum_i P(r_i, p_i, t-1)v(i) \leq P(r_i, p_i, t)$$

and

$$P(r_i, p_i, t) \leq \frac{1+\beta}{1-\beta} \sum_i P(r_i, p_i, t-1)v(i).$$

Thus, we have:

$$\left(\frac{1-\beta}{1+\beta}\right)^2 \left(1+\frac{\varepsilon_4}{g(d)}\right)^{-(t-1)} \leq \frac{P(r_i, p_i, t)}{\sum_i P(r_i, p_i, t-1)v(i)} \leq \frac{1+\beta}{1-\beta}.$$

From above inequality, we have:

$$\left(\frac{1-\beta}{1+\beta}\right)^2\left(1+\frac{\varepsilon_4}{g(d)}\right)^{-(t-1)}\frac{1}{P(r_i,p_i,t)}\le\frac{1}{\sum_i P(r_i,p_i,t-1)v(i)}\le\frac{1+\beta}{1-\beta}\frac{1}{P(r_i,p_i,t)}.$$

Via Lemma 5.2.2 we have:

$$\frac{(1-\beta)^2}{(1+\beta)^3}\left(1+\frac{\varepsilon_4}{g(d)}\right)^{-(t-1)}\frac{1}{C(r_i,p_i,t)}\le\frac{1}{\sum_i P(r_i,p_i,t-1)v(i)}\le\frac{1+\beta}{(1-\beta)^2}\frac{1}{C(r_i,p_i,t)}.$$

Let $g(d)=d^2$, $\varepsilon_4=\frac{\alpha}{4}$ and $\beta=\frac{\alpha}{\alpha+16d+16}$. Since Algorithm RecursiveBigBallRandomLatticePoint(.) has $d$ iteration, we can generate a random lattice point with bias of probability as:

$$\begin{aligned}
&\frac{P(r_i,p_i,d-1)}{\sum_i P(r_i,p_i,d-2)v(i)}\cdot\frac{P(r_i,p_i,d-2)}{\sum_i P(r_i,p_i,d-1)v(i)}\cdots\frac{P(r_i,p_i,0)}{\sum_i P(r_i,p_i,0)v(i)}\\
\le\;&\frac{1+\beta}{(1-\beta)^2}\frac{1}{C(r,p,d)}\cdot\left(\frac{1+\beta}{1-\beta}\right)^{d-1}\cdot P(r_i,p_i,0)\\
\le\;&\frac{1}{1-\beta}\frac{1}{C(r,p,d)}\cdot\left(\frac{1+\beta}{1-\beta}\right)^{d}\cdot(1+\beta)C(r_i,p_i,0)\\
=\;&\left(\frac{1+\beta}{1-\beta}\right)^{d+1}\frac{1}{C(r,p,d)}\\
=\;&\left(1+\frac{2\beta}{1-\beta}\right)^{d+1}\frac{1}{C(r,p,d)}\\
\le\;&e^{\frac{2\beta}{1-\beta}(d+1)}\frac{1}{C(r,p,d)}\\
\le\;&\left(1+\frac{4\beta}{1-\beta}(d+1)\right)\frac{1}{C(r,p,d)}\\
\le\;&(1+\alpha)\frac{1}{C(r,p,d)}
\end{aligned}$$

and

$$
\begin{aligned}
& \frac{P(r_i,p_i,d-1)}{\sum\limits_i P(r_i,p_i,d-2)v(i)} \cdot \frac{P(r_i,p_i,d-2)}{\sum\limits_i P(r_i,p_i,d-1)v(i)} \cdots \frac{P(r_i,p_i,0)}{\sum\limits_i P(r_i,p_i,0)v(i)} \\
\geq\ & \frac{(1-\beta)^2}{(1+\beta)^3}\left(1+\frac{\varepsilon_4}{g(d)}\right)^{-(d-1)} \frac{1}{C(r,p,d)} \left(\frac{1-\beta}{1+\beta}\right)^{2(d-1)} \left(1+\frac{\varepsilon_4}{g(d)}\right)^{\frac{-(d-1)(d-2)}{2}} (1-\beta)C(r_i,p_i,0) \\
=\ & \left(\frac{1-\beta}{1+\beta}\right)^{2d+1} \left(1+\frac{\varepsilon_4}{g(d)}\right)^{\frac{-(d-1)d}{2}} \frac{1}{C(r,p,d)} \\
\geq\ & \left(1-\frac{2\beta}{1+\beta}\right)^{2d+1} \left(1+\frac{\varepsilon_4}{g(d)}\right)^{\frac{-d^2}{2}} \frac{1}{C(r,p,d)} \\
\geq\ & \left(1-\frac{2\beta}{1+\beta}\right)^{2d} \left(1+\frac{\varepsilon_4}{g(d)}\right)^{-d^2} \frac{1}{C(r,p,d)} \\
\geq\ & \left(1-\frac{4\beta d}{1+\beta}\right) \left(1-\frac{\varepsilon_4 d^2}{g(d)}\right) \frac{1}{C(r,p,d)} \\
\geq\ & \left(1-\frac{4\beta d}{1+\beta}\right) (1-\varepsilon_4) \frac{1}{C(r,p,d)} \\
\geq\ & \left(1-\frac{4\beta d}{1+\beta}-\varepsilon_4\right) \frac{1}{C(r,p,d)} \\
\geq\ & \left(1-\frac{4\beta d}{1+\beta}\right) (1-\varepsilon_4) \frac{1}{C(r,p,d)} \\
\geq\ & (1-\alpha) \frac{1}{C(r,p,d)}
\end{aligned}
$$

Therefore, we can generate a random lattice point with probability between:

$$\left[(1-\alpha)\frac{1}{C(r,p,d)},\ (1+\alpha)\frac{1}{C(r,p,d)}\right].$$

The algorithm RecursiveBigBallRandomLatticePoint(.) forms a $\left(r, y_{d-t+1}, 1+\frac{\varepsilon_4}{g(d)}\right)$-partition $I_1,\cdots,I_w$ In line 3 for $[[\lceil y_{d-t+1}-r\rceil, \lfloor y_{d-t+1}+r\rfloor]\cap\mathbb{Z}$ and $[y_{d-t+1}+1, \lfloor y_{d-t+1}+r\rfloor]\cap\mathbb{Z}$. Then, there are at most $w$ number of $a_i$, where $w$ such that $\frac{r}{\left(1+\frac{\varepsilon_4}{g(d)}\right)^w} \leq 1$. Solving $w$, we have $w \geq \frac{g(d)\log r}{\varepsilon_4}$. And there are $d$ iterations in algorithm RecursiveBigBallRandomLatticePoint(.).

Thus, the running time of the algorithm is $\mathscr{O}\left(\frac{g(d)\log r}{\varepsilon_4}\cdot d\right) = \mathscr{O}(\frac{d^3\log r}{\varepsilon_4}) = \mathscr{O}\left(\frac{d^3\log r}{\alpha}\right)$. □

**Remark** : We note that there are at most one $(t-1)-$dimensional ball of radius $r < \frac{2d^3}{\alpha}$

with center at a lattice point, where $t = 1, 2, ..., d$. For this case, we can apply Theorem 5.2.1 with $\beta = 0$.

**Theorem 5.2.5.** *For arbitrary $\alpha \in (0,1)$, and $\alpha' \in (0,1)$, there is an $(1+\alpha')-$bias algorithm with runing time $\mathscr{O}\left(\frac{d^3\log(r+\sqrt{d})}{\alpha}\right)$ for a $d-$dimensional ball $B_d(r,q,d)$ to generate a random lattice point of radius $r > \frac{2d^{\frac{3}{2}}}{\alpha}$ with an arbitrary center.*

*Proof.* Consider another ball $B_d(r',q,d)$ of radius $r'$ with lattice center $q = (y_1, y_2, ..., y_d)$ that contains ball $B_d(r,p,d)$ , where $r' = r + \sqrt{d}$. Let $V_d(r)$ be the volume of a $d-$dimensional ball with radius $r$, then probability that a lattice point in $B_d(r',q,d)$ belongs to $B_d(r,p,d)$ is at least $(1-\alpha)\frac{C(r,p,d)}{C(r',p,d)}$.

Via Theorem 5.1.7, we have

$$\begin{cases} \frac{1}{1+\beta}V_d(r) \le C(r,p,d) \le \frac{1}{1-\beta}V_d(r) \\ \frac{1}{1+\beta}V_d(r+\sqrt{d}) \le C(r',q,d) \le \frac{1}{1-\beta}V_d(r+\sqrt{d}), \end{cases}$$

then we have

$$\frac{1-\beta}{1+\beta}\frac{V_d(r)}{V_d(r+\sqrt{d})} \le \frac{C(r,p,d)}{C(r',p,d)} \le \frac{1+\beta}{1-\beta}\frac{V_d(r)}{V_d(r+\sqrt{d})}.$$

The formula for a $d-dimensional$ ball of raduis $r$ is

$$V_d(r) = f(d)\cdot r^d$$

where $f(d) = \pi^{\frac{d}{2}}\Gamma\left(\frac{1}{2}d+1\right)^{-1}$ and $\Gamma(.)$ is Euler's gamma function. Let $\beta = \frac{\alpha}{8+\alpha}$ and $\alpha > \frac{2d^{\frac{3}{2}}}{r+\sqrt{d}}$,

$$\begin{aligned}
(1-\alpha)\frac{C(r,p,d)}{C(r',p,d)} \quad & \geq (1-\alpha)\frac{1-\beta}{1+\beta}\frac{f(d)\cdot r^d}{f(d)\cdot\left(r+\sqrt{d}\right)^d} \\
& = (1-\alpha)\frac{1-\beta}{1+\beta}\left(1-\frac{\sqrt{d}}{r+\sqrt{d}}\right)^d \\
& \geq (1-\alpha)\frac{1-\beta}{1+\beta}\left(1-\frac{d^{\frac{3}{2}}}{r+\sqrt{d}}\right) \\
& \geq (1-\alpha)\left(1-\frac{2\beta}{1-\beta}\right)\left(1-\frac{d^{\frac{3}{2}}}{r+\sqrt{d}}\right) \\
& \geq \left(1-\alpha-\frac{2\beta}{1-\beta}-\frac{d^{\frac{3}{2}}}{r+\sqrt{d}}\right)
\end{aligned}$$

Therefore, the probability a lattice point in $B_d(r',q,d)$ belongs to $B_d(r,p,d)$ fails is at most $\left(\alpha+\frac{2\beta}{1-\beta}+\frac{d^{\frac{3}{2}}}{r+\sqrt{d}}\right)$, where $\left(\alpha+\frac{2\beta}{1-\beta}+\frac{d^{\frac{3}{2}}}{r+\sqrt{d}}\right) < 1$, which means the algorithm BigBallRandomLatticePoint(.) fails with small possibility.

The probability to generate a random lattic point in ball $B_d(r',q,d)$ is in range of

$$\left[(1-\alpha)\frac{1}{C(r',q,d)},(1+\alpha)\frac{1}{C(r',q,d)}\right]$$

via Theorem 5.2.4. Then the bias that to generate a random lattic point in ball $B_d(r,p,d)$ is $\frac{\Pr(p_i)}{\sum_i \Pr(p_i)}$, where $\Pr(p_i) \in \left[(1-\alpha)\frac{1}{C(r',q,d)},(1+\alpha)\frac{1}{C(r',q,d)}\right]$.

Then, we have

$$\begin{aligned}
\frac{\Pr(p_i)}{\sum_i \Pr(p_i)} &\leq \frac{(1+\alpha)\frac{1}{C(r',q,d)}}{(1-\alpha)\frac{1}{C(r',q,d)}C(r,p,d)} \\
&= \frac{1+\alpha}{1-\alpha}\frac{1}{C(r,p,d)} \\
&= \left(1+\frac{2\alpha}{1-\alpha}\right)\frac{1}{C(r,p,d)},
\end{aligned}$$

and

$$\begin{aligned}
\frac{\Pr(p_i)}{\sum_i \Pr(p_i)} &\geq \frac{(1-\alpha)\frac{1}{C(r',q,d)}}{(1+\alpha)\frac{1}{C(r',q,d)}C(r,p,d)} \\
&= \frac{1-\alpha}{1+\alpha}\frac{1}{C(r,p,d)} \\
&= \left(1-\frac{2\alpha}{1+\alpha}\right)\frac{1}{C(r,p,d)} \\
&\geq \left(1-\frac{2\alpha}{1-\alpha}\right)\frac{1}{C(r,p,d)}.
\end{aligned}$$

Therefore, the probability to generate a random lattice point in $B_d(r,p,d)$ is range of

$$\left[(1-\alpha')\frac{1}{C(r,p,d)},(1+\alpha')\frac{1}{C(r,p,d)}\right],$$

where $\alpha' = \frac{2\alpha}{1-\alpha}$.

It takes $\mathscr{O}\left(\frac{d^3\log(r+\sqrt{d})}{\alpha}\right)$ running time to generate a random lattice point inside of $d-$ dimensional ball $B_d(r+\sqrt{d},p,d)$ with a lattice point center via Theorem 5.2.4. Thus, the algorithm BigBallRandomLatticePoint(.) takes $\mathscr{O}\left(\frac{d^3\log(r+\sqrt{d})}{\alpha}\right)$ running time to generate a random lattice. □

**Theorem 5.2.6.** *For an arbitrary $\alpha \in (0,1)$, there is an algorithm with runing time $\mathscr{O}\left(\frac{d^3\log(r+\sqrt{d})}{\alpha}\right)$ and $(1+\alpha)-$bias for a $d-$dimensional ball $B_d(r,q,d)$ to generate a random lattice pointo f ra-*

*dius $r > \frac{2d^{\frac{3}{2}}}{\alpha}$ with a arbitrary center; and there is a $\mathscr{O}\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)$ time algorithm to generate a lattice point inside a $d-$dimensional ball $B_d(r,p,d)$ of radius $r \leq \frac{2d^{\frac{3}{2}}}{\alpha}$ with center $p \in D(\lambda,d,l)$.*

*Proof.* We discuss two cases based the radius of the $d$-dimensional ball.

Case 1: When generate a random lattice point inside a $d$-dimensional ball of radius $r > \frac{2d^{\frac{3}{2}}}{\alpha}$ with center arbitrary center $p$, apply Theorem 5.2.5.

Case 2: When generate a random lattice point inside a $d$-dimensional ball of radius $r \leq \frac{2d^{\frac{3}{2}}}{\alpha}$ with center $p \in D(\lambda,d,l)$, apply Theorem 5.2.1. □

### 5.3 Count Lattice Point in the Union of High Dimensional Balls

In this section, we apply the algorithm developed in Section IV to count the total number of lattice point in the union of high dimensional balls.

**Theorem 5.3.1.** *There is a $\mathrm{O}\left(poly\left(\frac{1}{\varepsilon},\log\frac{1}{\gamma}\right)\cdot m\cdot(\log m)^{\mathrm{O}(1)}\right)$ time and $\mathrm{O}(\log m)$ rounds algorithm for the number of lattice points in $B_1\cup B_2\cup\cdots\cup B_m$ such that with probability at least $1-\gamma$, it gives a sum $\cdot M \in [(1-\varepsilon)(1-\alpha_L)(1-\beta_L)\cdot|B_1\cup\cdots\cup B_m|,(1+\varepsilon)(1+\alpha_R)(1+\beta_R)\cdot|B_1\cup\cdots\cup B_m|]$, where each ball $B_i$ satisfy that either its radius $r > \frac{2d^{\frac{3}{2}}}{\beta}$ or its center $p \in D(\lambda,d,l)$ and $|B_1\cup\cdots\cup B_m|$ is the total number of lattice point of union of $m$ high dimensional balls.*

*Proof.* Apply Theorem 5.1.8 and Theorem 5.2.6, we have $m_i$ for each ball $B_i$ with

$$m_i \in \left((1-\beta_L)C_i(r_i,p_i,t),(1+\beta_R)C_i(r_i,p_i,t)\right),$$

and biased random generators with

$$\mathrm{Prob}(x=\mathrm{RandomElement}(B_i)) \in \left[\frac{1-\alpha_L}{C_i(r_i,p_i,t)},\frac{1+\alpha_R}{C_i(r_i,p_i,t)}\right]$$

for each input ball $B_i$, where $C_i(r_i,p_i,t)$ is the number of lattice point of $t-$dimensional ball $B_i$ of radius $r_i$ for $i=1,2,...,m$. Then apply Theorem 4.3.8. □

### 5.3.1 Hardness to Count Lattice Points in a Set of Balls

In this section, we show that it is #P-hard to count the number of lattice points in a set of balls.

**Theorem 5.3.2.** *It is #P-hard to count the number of lattice points in a set of d-dimensional balls even the centers are of the format $(x_1, \cdots, x_d) \in \mathbb{R}^d$ that has each $x_i$ to be either $1$ or $\frac{\sqrt{h}}{2}$ for some integer $h \le d$.*

*Proof.* We derive a polynomial time reduction from DNF problem to it. For each set of lattice points in a $h$-dimensional cube $\{0,1\}^h$, we design a ball with radius $r = \frac{\sqrt{h}}{2}$ and center at $C = (\frac{\sqrt{h}}{2}, \cdots, \frac{\sqrt{h}}{2})$. It is easy to see that this ball only covers the lattice points in $\{0,1\}^h$. Every $0,1$-lattice point in $0,1$ has distance to the center $C$ equal to $r$. For every lattice point $P \in R^h$ that is not in $\{0,1\}^h$ has distance $d$ with $d^2 \ge r^2 + (1+\frac{1}{2})^2 - (\frac{1}{2})^2 = r^2 + 2$. □

**Definition 5.3.1.** *For a center $c = (c_1, \cdots, c_d)$ and an even number $k > 0$ and a real $r > 0$, a d-dimensional k-degree ball $B_k(c,r)$ is $\{(x_1, \cdots, x_d) : (x_1, \cdots, x_d) \in \mathbb{R}^d$ and $\sum_{i=1}^{d} (x_i - c_i)^k \le r\}$.*

**Theorem 5.3.3.** *Let k be an even number at least $2$. Then we have*

1. *There is no polynomial time algorithm to approximate the number of lattice points in the intersection n-dimensional k-degree balls unless P=NP, and.*

2. *It is #P-hard to count the number of lattice points in the intersection n-dimensional k-degree balls.*

*Proof.* We derive a polynomial time reduction from 3SAT problem to it. For each clause $C = (x_i^* \vee x_j^* \vee x_k^*)$, we can get a ball to contain all lattice points in the 0-1-cube to satisfy $C$, each $x_i^*$ is a literal to be either $x_i$ or its negation $\bar{x}_i$.

Without loss of generality, let $C = (x_1 \vee x_2 \vee x_3)$. Let $\delta = 0.30$. Let center

$D_C = (d_1, d_2, d_3, d_4, d_5, \cdots, d_n) = (1-\delta, 1-\delta, 1-\delta, \frac{1}{2}, \frac{1}{2}, \cdots, \frac{1}{2})$, which has value $1-\delta$ in the first three positions, and $\frac{1}{2}$ in the rest. For $0,1$ assignment $(a_1, a_2, \cdots, a_n)$ of $n$ variables, if it satisfies $C$ if and only if $\sum\limits_{i=1}^{n}(a_i-d_i)^k \le 2(1-\delta)^k+\delta^k+(n-3)\cdot(\frac{1}{2})^k$. Therefore, we can select radius $r_C$ that satisfies $r_C^k = 2(1-\delta)^k+\delta^k+(n-3)\cdot(\frac{1}{2})^k$. We have the following inequalities:

$$\begin{cases} (2-\delta)^2 > (1+\delta)^k > 2(1-\delta)^k+\delta^k \\ (1+\frac{1}{2})^k > 2(1-\delta)^k+\delta^k+(\frac{1}{2})^k. \end{cases} \tag{5.5}$$

This is because we have the following equalities:

$$\begin{cases} (1+\delta)^2 = 1.69, \\ 2(1-\delta)^2+\delta^2 = 2\times 0.49+0.09 = 1.07, \\ 2(1-\delta)^2+\delta^2+(\frac{1}{2})^2 = 1.07+0.25 = 1.32, \\ (1+\frac{1}{2})^2 = 2.25. \end{cases} \tag{5.6}$$

If $Y = (y_1, y_2, \cdots, y_n)$ is not a $0,1$-lattice point, we discuss two cases:

- Case 1. $y_i \notin \{0,1\}$ for some $i$ with $1 \le i \le 3$.

  In this case we know that $\text{dist}(Y, D_C)^2 > r_C^2$ by inequality (5.6).

- Case 2. $y_i \notin \{0,1\}$ for some $i$ with $3 < i \le n$.

  In this case we know that $\text{dist}(Y, D_C)^2 > r_C^2$ by inequality (5.6).

If $Y = (y_1, y_2, \cdots, y_n)$ is a $0,1$-lattice point, we discuss two cases:

- Case 1. $Y$ satisfies $C$.

  In this case we know that $\text{dist}(Y, D_C)^2 \le r_C^2$.

- Case 2. $Y$ does not satisfy $C$.

  In this case we know that $\text{dist}(Y, D_C)^2 > r_C^2$ by inequality $(1-\delta)^2 > \delta^2$.

The ball $B_C$ with center at $D_C$ and radius $r_C$ contains exactly those 0,1-lattice points that satisfy clause $C$. This proves the first part of the theorem.

If there were any factor *c*-approximation to the intersection of balls, it would be able to test if the intersection is empty. This would bring a polynomial time solution to 3SAT.

It is well known that #3SAT is #P-hard. Therefore, It is #P-hard to count the number of lattice points in the intersection *n*-dimensional balls. This proves the second part of the theorem.

□

Another interesting open problem is if there is any polynomial time algorithm to count the number of lattice points in a *n*-dimensional ball with arbitrary center. For a ball with an arbitrary center, we do not have the recursion as we used.

# CHAPTER VI

## APPROXIMATION FOR THE MAXIMAL COVERAGE WITH BALLS

We apply the technology developed in this paper to the maximal coverage problem when each set is a set of lattice points in a ball with center in $D(\lambda, d, l)$.

The classical maximum coverage is that given a list of sets $A_1, \cdots, A_m$ and an integer $k$, find $k$ sets from $A_1, A_2, \cdots, A_m$ to maximize the size of the union of the selected sets in the computational model defined in Definition 2.1.2. For real number $a \in [0,1]$, an approximation algorithm is a $(1-a)$-approximation for the maximum coverage problem that has input of integer parameter $k$ and a list of sets $A_1, \cdots, A_m$ if it outputs a sublist of sets $A_{i_1}, A_{i_2}, \cdots, A_{i_k}$ such that $|A_{i_1} \cup A_{i_2} \cup \cdots \cup A_{i_k}| \geq (1-a)|A_{j_1} \cup A_{j_2} \cup \cdots \cup A_{j_k}|$, where $A_{j_1}, A_{j_2}, \cdots, A_{j_k}$ is a solution with maximum size of union.

**Theorem 6.0.1.** *[17] Let $\rho$ be a constant in $(0,1)$. For parameters $\xi, \gamma \in (0,1)$ and $\alpha_L, \alpha_R, \delta_L$, $\delta_R \in [0, 1-\rho]$, there is an algorithm to give a $\left(1-(1-\frac{\beta}{k})^k - \xi\right)$-approximation for the maximum cover problem, such that given a $((\alpha_l, \alpha_r), (\delta_L, \delta_R))$-list $L$ of finite sets $A_1, \cdots, A_m$ and an integer $k$, with probability at least $1-\gamma$, it returns an integer $z$ and a subset $H \subseteq \{1, 2, \cdots, m\}$ that satisfy*

1. *$|\cup_{j \in H} A_j| \geq \left(1-(1-\frac{\beta}{k})^k - \xi\right) C^*(L,k)$ and $|H| = k$,*
2. *$((1-\alpha_L)(1-\delta_L) - \xi)|\cup_{j \in H} A_j| \leq z \leq ((1+\alpha_R)(1+\delta_R) + \xi)|\cup_{j \in H} A_j|$, and*
3. *Its complexity is $(T(\xi, \gamma, k, m), R(\xi, \gamma, k, m), Q(\xi, \gamma, k, m))$ with*

$$T(\xi, \gamma, k, m) \quad = \quad \mathscr{O}\left(\frac{k^3}{\xi^2}\left(k \log\left(\frac{3m}{k}\right) + \log\frac{1}{\gamma}\right) m\right),$$

*where $\beta = \frac{(1-\alpha_L)(1-\delta_L)}{(1+\alpha_R)(1+\delta_R)}$ and $C^*(L,k)$ is the number of elements to be covered in an optimal solution.*

We need Lemma 6.0.2 to transform the approximation ratio given by Theorem 6.0.1 to constant $(1-\frac{1}{e})$ to match the classical ratio for the maximum coverage problem.

**Lemma 6.0.2.** *For each integer $k \geq 2$, and real $b \in [0,1]$, we have*

1. $(1-\frac{b}{k})^k \leq \frac{1}{e} - \frac{\eta}{e}(b+\frac{b}{2k}-1)$*, and*

2. *If $\xi \leq \frac{\eta}{e}(b+\frac{b}{2k}-1)$, then $1-(1-\frac{b}{k})^k - \xi > 1-\frac{1}{e}$, where $\eta = e^{-\frac{1}{4}}$.*

*Proof.* Let function $f(x) = 1-\eta x - e^{-x}$. We have $f(0) = 0$. Taking differentiation, we get $\frac{df(x)}{dx} = -\eta + e^{-x} > 0$ for all $x \in (0, \frac{1}{4})$.

Therefore, for all $x \in (0, \frac{1}{4})$,

$$e^{-x} \leq 1 - \eta x. \tag{6.1}$$

The following Taylor expansion can be found in standard calculus textbooks. For all $x \in (0,1)$,

$$\ln(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \cdots .$$

Therefore, we have

$$\begin{aligned}
(1-\frac{b}{k})^k &= e^{k\ln(1-\frac{b}{k})} = e^{k(-\frac{b}{k}-\frac{b^2}{2k^2}-\frac{b^3}{3k^3}-\cdots)} = e^{-b-\frac{b^2}{2k}-\frac{b^3}{3k^2}-\cdots} \\
&\leq e^{-b-\frac{b}{2k}} = e^{-1}\cdot e^{1-b-\frac{b}{2k}} \qquad (6.2) \\
&\leq e^{-1}\cdot(1-\eta\cdot(b+\frac{b}{2k}-1)) \leq \frac{1}{e} - \frac{\eta}{e}(b+\frac{b}{2k}-1). \qquad (6.3)
\end{aligned}$$

Note that the transition from (6.2) to (6.3) is based on inequality (6.1).

The part 2 follows from part 1. This is because $1-(1-\frac{b}{k})^k - \xi \geq 1-\frac{1}{e}+\frac{\eta}{e}(b+\frac{b}{2k}-1) - \xi \geq 1-\frac{1}{e}$. □

**Theorem 6.0.3.** *There is a poly*$(\lambda, d, l, k, m)$ *time* $(1 - \frac{1}{e})$*-approximation algorithm for maximal coverage problem when each set is the set of lattice points in a ball with center in* $D(\lambda, d, l)$*.*

*Sketch.* Let $\alpha = \alpha_L = \alpha_R = \delta_L = \delta_R = \frac{1}{ck}$ with $c = 100$, and $b = \beta = \frac{1-\alpha_L)(1-\delta_L)}{(1+\alpha_R)(1+\delta_R)}$. It is easy to see $(b + \frac{b}{2k} - 1) \geq \frac{1}{4k}$. Let $\xi = \frac{\eta}{e}(b + \frac{b}{2k} - 1) = \Theta(\frac{1}{k})$. It follows from Theorem 6.0.1, Lemma 6.0.2, Theorem 5.1.8 and Theorem 5.2.6. □

# CHAPTER VII

# SELF-ADJUSTING COVERAGE ALGORITHM FOR SET UNION PROBLEM UNDER MODEL OF RANDOMIZATION

In this section, we generalize the algorithm that was designed by Karp, Luby, and Madras [29] to approximate the union set $|A_1 \cup \cdots \cup A_m|$ to a randomized approximation algorithm under model of randomization.

The union of set problem is that given a list of sets $A_1,..,A_m$ to compute the $|A_1 \cup \cdots \cup A_m|$.

In [29], the Self-Adjusting Coverage Algorithm was designed under the below model:

- The size of $A_i$ (i.e. $|A_i|$) is part of input.
- It generates an random element $x \in |A_1 \cup \cdots \cup A_m|$ with probability $\frac{1}{|A_i|}$ for some integer $i \in [1,m]$.
- Given any $s \in |A_1 \cup \cdots \cup A_m|$, it is easy to decide whether or not $x \in A_i$ for some integer $i \in [1,m] \cap \mathbb{Z}$.

We generalize the the Self-Adjusting Coverage Algorithm to a randomized approximation algorithm under model of randomization that defined below:

- It provides an approximate set size $m_i$ of $A_i$ with $m_i \in [(1-\beta_L)|A_i|,(1+\beta_R)|A_i|]$ for $i = 1,2,...,d$.
- It generates an random element $x \in |A_1 \cup \cdots \cup A_m|$ with probability in range of in the range $\left[\frac{1-\alpha_L}{|A_i|}, \frac{1+\alpha_R}{|A_i|}\right]$ for some integer $i \in [1,m]$.

- Given any $s \in |A_1 \cup \cdots \cup A_m|$, it is easy to decide whether or not $x \in A_i$ for some integer $i \in [1, m] \cap \mathbb{Z}$.

The running time of the algorithm Self-Adjusting Coverage Algorithm under the model of randomization is $\mathrm{O}\left(\frac{1+\varepsilon}{\varepsilon^2} \log \frac{2}{\gamma} \frac{(1+\alpha)(1-\beta_R)}{1-\beta_L} m\right)$ with probability in range $[(1-\varepsilon)(1-\alpha)\frac{1-\beta_L}{1+\beta_R}, (1+\varepsilon)(1+\alpha)\frac{1+\beta_L}{1-\beta_R}]$, where $\varepsilon$ controls the accuracy of approximation, and $\gamma$ controls the failure probability.

We can not obtain $\mathscr{O}(\log m)$ rounds under model of randomization for round complexity.

An interesting open problem is to find an $\mathrm{O}(m)$ time and $\mathrm{O}(\log m)$ rounds approximation scheme for $|A_1 \cup A_2 \cup \cdots A_m|$.

# CHAPTER VIII

# CONCLUSIONS

We introduce an almost linear bounded rounds randomized approximation algorithm for the size of set union problem $|A_1 \cup A_2 \cup ... \cup A_m|$, which given a list of sets $A_1, ..., A_m$ with approximate set size and biased random generators. The definition of round is introduced. We prove that our algorithm runs sublinear in time under certain condition. A polynomial time approximation scheme is proposed to approximae the number of lattice points in the union of d-dimensional ball if each ball center satisfy $D(\lambda, d, l)$. We prove that it is #P-hard to count the number of lattice points in a set of balls, and we also show that there is no polynomial time algorithm to approximate the number of lattice points in the intersection of $n$-dimenisonal $k$-degree balls unless P=NP.

# REFERENCES

[1] Sukumar D. Adhikari and Y. F. S. Pétermann. Lattice points in ellipsoids. *Acta Arith.*, 59(4):329–338, 1991.

[2] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 20–29, 1996.

[3] George E. Andrews, Shalosh B. Ekhad, and Doron Zeilberger. A short proof of jacobi's formula for the number of representations of an integer as a sum of four squares. *The American Mathematical Monthly*, Vol. 100, No. 3:274–276, 1993.

[4] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *Randomization and Approximation Techniques, 6th International Workshop, RANDOM 2002, Cambridge, MA, USA, September 13-15, 2002, Proceedings*, pages 1–10, 2002.

[5] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.*, pages 623–632, 2002.

[6] József Beck. On a lattice point problem of l. moser I. *Combinatorica*, 8(1):21–47, 1988.

[7] Jaroslaw Blasiok. Optimal streaming and tracking distinct elements with high probability. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2432–2448, 2018.

[8] Karl Bringmann and Tobias Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom.*, 43(6-7):601–610, 2010.

[9] Samuel R. Buss and Louise Hay. On truth-table reducibility to SAT and the difference hierarchy over NP. In *Proceedings: Third Annual Structure in Complexity Theory Conference, Georgetown University, Washington, D. C., USA, June 14-17, 1988*, pages 224–233, 1988.

[10] K. Chandrasekharan and Raghavan Narasimhan. On lattice-points in a random sphere. *Bull. Amer. Math. Soc.*, 73(1):68–71, 1967.

[11] Jing-R. Chen. Improvement on the asymptotic formulas for the number of lattice points in a region of the three dimensions (ii). *Scientia Sinica*, 12(5).

[12] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158, 1971.

[13] K. Corr*á*adi and I. K*á*tai. A comment on k. s. gangadharan's paper entitled "two classical lattice point problems". *Magyar Tud. Akad. Mat. Fiz. Oszt. Kozl*, 17.

[14] Philippe Flajolet, Éric Fusy, Olivier Gandoue, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *2007 Conference on Analysis of Algorithms, AofA 07*, pages 127–146, 2007.

[15] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[16] Lance Fortnow and Nick Reingold. PP is closed under truth-table reductions. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 - July 3, 1991*, pages 13–15, 1991.

[17] Bin Fu. Partial sublinear time approximation and inapproximation for maximum coverage. arXiv:1604.01421, April 5, 2016.

[18] Sumit Ganguly, Minos N. Garofalakis, and Rajeev Rastogi. Tracking set-expression cardinalities over continuous update streams. *VLDB J.*, 13(4):354–369, 2004.

[19] Phillip B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, pages 541–550, 2001.

[20] Phillip B. Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *SPAA*, pages 281–291, 2001.

[21] Peter J. Haas, Jeffrey F. Naughton, S. Seshadri, and Lynne Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland.*, pages 311–322, 1995.

[22] James L. Hafner. New omega theorems for two classical lattice point problems. *Inventiones Mathematicae*, 63(2):181–186, 1981.

[23] D. R. Heath-Brown. Lattice points in sphere. *Numb. Theory Prog.*, 2:883–892, 1999.

[24] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[25] Zengfeng Huang, Wai Ming Tai, and Ke Yi. Tracking the frequency moments at all times. *CoRR*, abs/1412.1763, 2014.

[26] M. N. Huxley. Exponential sums and lattice points ii. *Proc. London Math. Soc.*, 66(2):279–301, 1993.

[27] C. Jacobi. *Gesammelte Werke, Berlin 1881-1891.* Reprinted by Chelsea, New York, 1969.

[28] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 41–52, 2010.

[29] Richard M. Karp, Michael Luby, and Neal Madras. Monte-carlo approximation algorithms for enumeration problems. *J. Algorithms*, 10(3):429–448, 1989.

[30] J. E. Mazo and A. M. Odlyzko. Lattice points in high-dimensional spheres. *Monatsh. Math.*, 110(1):47–61, 1990.

[31] Annika Meyer. On the number of lattice points in a small sphere and a recursive lattice decoding algorithm. *Des. Codes Cryptography*, 66(1-3):375–390, 2013.

[32] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 2000.

[33] G. Szeg*ö*. Beitr*ä*ge zur theorie der laguerreschen polynome ii: Zahlentheoretische anwendungen. *Math. Z.*, 25(1).

[34] Kai-M. Tsang. Counting lattice points in the sphere. *Bulletin of the London Mathematical Society*, 32(6).

[35] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[36] A. I. Vinogradov and M. M. Skriganov. The number of lattice points inside the sphere with variable cente, analytic number theory and the theory of functions, 2. *Zap. Nauen. Sem. Leningrad Otdel. Mat. Inst. Steklov (LOMI)*, 91:25–30, 1979.

[37] I. M. Vinogradov. On the number of integer points in a sphere. *Izu. Akad. Nauk SSSR Ser. Mat.*, 27(5):957–968, 1963.

[38] A. Walfisz. Weylsche exponentialsummen in der neueren zahlentheorie. *VEB Deutscher Verlag der Wissenschaften*, 1963.

[39] Arnold Walfisz. *Gitterpunkte in mehrdimensionalen Kugeln*. Instytut Matematyczny Polskiej Akademi Nauk(Warszawa), 1957.

[40] A. A. Yudin. On the number of integer points in the displaced circles. *Acta Arith*, 14(2):141–152, 1968.

# BIOGRAPHICAL SKETCH

Pengfei Gu was born on January 17, 1991 in China. He obtained his bachelor degree of applied mathematics at Tianjin University of Technology and Education in 2014. In August 2016, he has earned master degree of mathematics in University of Texas Rio Grande Valley. After graduation, he pursued for the second master degree of computer science in University of Texas Rio Grande Valley. His research interest is Approximation Algorithms. He received his second master degree in May 2018 at University of Texas Rio Grande Valley. His permanent mailing address is Yongxing County, Chenzhou City, Hunan Province, China, 423313. His personal email address is: pengfeigu01@gmail.com.