University of Texas Rio Grande Valley

ScholarWorks @ UTRGV

Computer Science Faculty Publications and Presentations

College of Engineering and Computer Science

6-15-2018

Approximate Set Union via Approximate Randomization

Bin Fu
The University of Texas Rio Grande Valley

Pengfei Gu
The University of Texas Rio Grande Valley

Yuming Zhao

Follow this and additional works at: https://scholarworks.utrgv.edu/cs_fac

Part of the Computer Sciences Commons

Recommended Citation

Approximate Set Union via Approximate Randomization *

Bin Fu[1], Pengfei Gu[1], and Yuming Zhao[2]

[1]Department of Computer Science
University of Texas - Rio Grande Valley, Edinburg, TX 78539, USA

[2]School of Computer Science
Zhaoqing University, Zhaoqing, Guangdong 526061, P.R. China

Abstract

We develop a randomized approximation algorithm for the size of set union problem $|A_1 \cup A_2 \cup ... \cup A_m|$, which is 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 $[(1 - \epsilon)(1 - \alpha_L)(1 - \beta_L), (1 + \epsilon)(1 + \alpha_R)(1 + \beta_R)]$ for any $\epsilon \in (0, 1)$, 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. Our algorithm can handle input sets that can generate random elements with bias, and its approximation ratio depends on the bias. Our algorithm gives a flexible tradeoff with time complexity $\text{O}(m^{1+\xi})$ and round complexity $\text{O}\left(\frac{1}{\xi}\right)$ for any $\xi \in (0, 1)$. 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 $\text{O}\left(r(r + l|\lambda|)^3 l^3 d^4\right)$ 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.

1. 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 $i = 1, 2, ..., m$, 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 $(1 + \epsilon)$-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 $O((1 + \epsilon)m/\epsilon^2)$ time. Their algorithm is based on the input that provides the exact size of each set and an 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 $2^{d^{1-\epsilon}}$-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 Prob($x = $ 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 in $D(\lambda, d, l)$, 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 $\left[\frac{1-\alpha}{C(r,p,d)}, \frac{1+\alpha}{C(r,p,d)}\right]$ 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 an elusive problem to develop a poly $\left(d, \frac{1}{\epsilon}\right)$ time $(1+\epsilon)$-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 λ 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, 39, 38, 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 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 papers [13, 22, 26]. Walfisz [38] 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 $x \to \infty$ if there exist a sequence $\{x_n\} \to \infty$ 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 $[(1-\epsilon)(1-\alpha_L)(1-\beta_L), (1+\epsilon)(1+\alpha_R)(1+\beta_R)]$ for any $\epsilon \in (0, 1)$, 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 $\log m$ rounds with time complexity O $\left(m(\log m)^{O(1)}\right)$, and takes O $\left(\frac{1}{\xi}\right)$ rounds, with a time complexity 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].

Organization: The rest of paper is organized as follows. In Section 2, we define the computational model and complexity. Section 3 presents some theorems that play an important role in accuracy analysis. In Section 4, we give a randomized approximation algorithm to approximate the size of set union problem; time complexity and round complexity also analysis in Section 4. Section 5 discusses a natural problem that counting of lattice points in high dimensional balls to support the useful of approximation randomized algorithm. An application of high dimensional balls in Maximal Coverage gives in Section 6. In Section 7, we summarize with conclusions.

2. Computational Model and Complexity

In this section, we show our model of computation, and the definition of complexity.

2.1. Model of Randomization

Definition 1. Let A be a set of elements.

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

ii. A (α_L, α_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. Let L be a list of sets A_1, A_2, $\cdots$, A_m such that each supports the following operations:

i. 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.

ii. Function RandomElement(A_i) returns a (α_L, α_R)-biased approximate random element x from A_i for $i = 1, 2, \cdots, m$.

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

Definition 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 m_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 (α_L, α_R)-biased random generator RandomElement(A_i).

Definition 4. The model of randomized computation for our algorithm is defined below:

i. The input is a list L defined in Definition 2.

ii. It allows all operations defined in Definition 2.

2.2. Round and Round Complexity

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

Our algorithm is considered as a client-server interaction (see Fig. 1). 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.
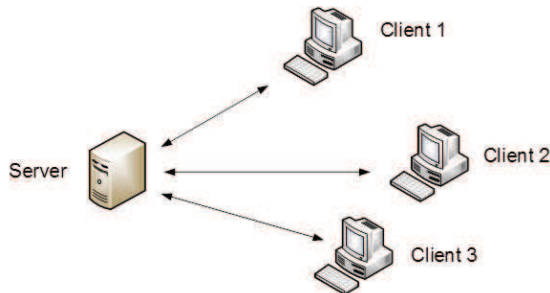

Figure 1: Client-server Interaction

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

3. 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 5 ([24]). *Let X_1, ..., 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 $\epsilon > 0$, $\Pr(X > pm + \epsilon m) < e^{-\frac{\epsilon^2 m}{2}}$.

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

Theorem 6. *Let X_1, ..., 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 7. *Let X_1, ..., 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 $x > 0$.

5

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

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

Therefore,

$$u(x) \le e^{-\frac{x^2}{6}} \tag{1}$$

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

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

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

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

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

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.

4. 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 8. Let $L = A_1, A_2, \cdots, A_m$ be a list of finite sets.

i. For an element x, define $T(x, L) = \left|\{i : 1 \le i \le m \text{ and } x \in A_i\}\right|$.

ii. For an element x, and a subset of indices with multiplicity H of $\{1, 2, \cdots, m\}$, define $S(x, H) = \left|\{i : i \in H \text{ and } x \in A_i\}\right|$.

iii. Define $\text{minThickness}(L) = \min\{T(x, L) : x \in A_1 \cup A_2 \cup \cdots \cup A_m\}$.

iv. Define $\text{maxThickness}(L) = \max\{T(x, L) : x \in A_1 \cup A_2 \cup \cdots \cup A_m\}$.

v. 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
$F'(W) = \sum\limits_{x \in W} \frac{1}{T(x,L)} = \frac{h}{s} F(W, h, s)$.

vi. 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 $A'_i = \{x : x \in A_1 \cup A_2 \cup \cdots \cup A_m \text{ and } T(x, L) \in [(1+\delta)^{i-1}, (1+\delta)^i)\}$ where $i = 1, 2, ..., k$. Define $v(\delta, z_1, z_2, L) = k$, which is the number of sets in the partition under the condition that $z_1 \le T(x, L) \le z_2$.

6

4.1. Overview of Algorithm

We give an overview of the algorithm. For a 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. The set R_1 has the property that $\frac{m_1 + \cdots + m_m}{|R_1|} \cdot \sum_{x \in R_1} \frac{1}{T(x,L)}$ 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 $T(x,L) \in \left[\frac{T_i}{4f_1(m)}, T_i \right]$, and all elements $x \in R_i$ with $T(x,L) \in \left[\frac{T_i}{f_1(m)}, T_i \right]$ are in V_i, where $T_i = \max\{T(x,L) : x \in R_i\}$ and $f_1(m)$ 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 $|A_1 \cup A_2 \cup \cdots \cup A_m|$.

Example 1. Let L be a list of 10 sets $A_1, A_2, \cdots, A_{10}$, where $A_i = B_i \cup C$ with $|C| = 1000$ and $|B_i| = 100$ for $i = 1, 2, \cdots, 10$. In the beginning of the algorithm, we generate a set R_1 of $h_1 = 220$ random samples from list L, where there are 200 random samples with higher thickness $T(x, L)$, namely, these 200 random samples locate in C and 20 random samples with lower thickness $T(x, L)$, say, these 20 random samples locate in B_i. At the first round, we only need select sets A_1, A_3, and A_6 to approximate the thickness $T(x, L)$ of the 200 random samples locating at C. Then at the second round, we have to select all the sets $A_1, A_2, \cdots, A_{10}$, to approximate the thickness $T(x, L)$ of the 20 random samples coming from B_i (See Fig. 2).
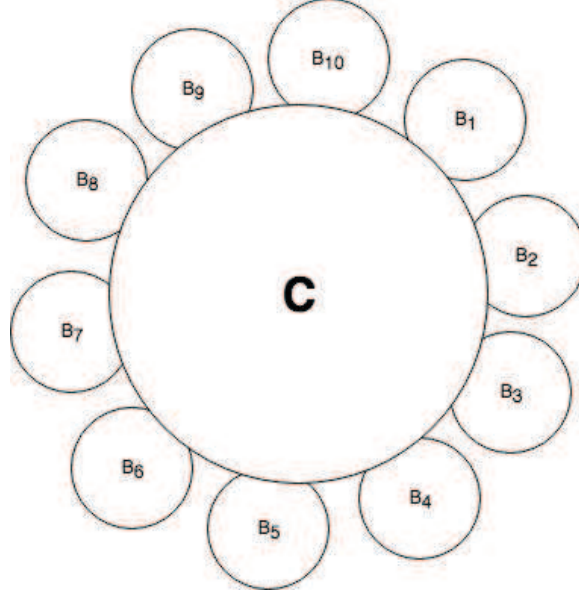
7

Figure 2: Set Union of Ten Sets

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 9. 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 (α_L, α_R)-biased random generator RandomElement(A_i) for $i = 1, 2, \cdots, m$, and $M = m_1 + m_2 + \cdots + m_m$. A *random choice* of L is to get an element x via the following two steps:

   i. With probability $\frac{m_i}{M}$, select a set A_i among $A_1, \cdots, A_m$.

   ii. Get an element x from set A_i 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 two integer parameters z_{min} and z_{max} with $1 \leq z_{\min} \leq \text{minThickness}(L) \leq \text{maxThickness}(L) \leq z_{\max} \leq m$ can help speed up the computation. The algorithm is still correct if we use default case with $z_{min} = 1$ and $z_{max} = m$.

   i. The following parameters are used to control the accuracy of approximation at different stages of algorithm:

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

$$\delta = \frac{\epsilon_2}{2}. \tag{4}$$

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

$$\gamma_1 = \frac{\gamma}{3}, \gamma_2 = \frac{\gamma}{6 \log m}. \tag{5}$$

iii. 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:

$$f_1(m) \quad = \quad 8m^{c_1} \text{ with } c_1 \geq 0, \tag{6}$$

iv. Function $f_2(.)$ is used to check the number of random samples in A'_j of Stage 1 in the algorithm. We will use different ways to control the accuracy of approximation between the case $|A_i| \leq \frac{|A_1 \cup A_2 \cup \cdots \cup A_m|}{f_2(m)}$ and the other case $|A_i| > \frac{|A_1 \cup A_2 \cup \cdots \cup A_m|}{f_2(m)}$. It is mainly used in the proof of Lemma 15 that shows it keeps the accuracy of approximation when algorithm goes from Stage i to Stage $i+1$.

$$f_2(m) \quad = \quad \frac{2v(\delta, z_{\min}, z_{\max}, L)}{\epsilon_3} + \frac{2 \log \frac{m}{z_{\min}}}{\epsilon_3 \log(1 + \delta)}. \tag{7}$$

v. Function $f_3(.)$ is used as a threshold to count the number $t_{i,j}$ of random samples in $R_i \cap A'_j$ of Stage i in the algorithm. We will use different ways to control the accuracy of approximation between the case $t_{i,j} \leq f_3(m)$ and the other case $t_{i,j} > f_3(m)$. It is mainly used in the proof of Lemma 12 that shows that the number of random samples at Stage 1 will provide enough accuracy of approximation.

$$f_3(m) \quad = \quad f_1(m) \cdot \frac{6 \ln \frac{2}{\gamma_2}}{\epsilon_2^2}. \tag{8}$$

vi. Function $f_4(.)$ is used to determine the growth rate of function Function $f_5(.)$, which is defined by equation (10).

$$f_4(m) \quad = \quad \frac{f_2(m) \log \frac{m^2}{\epsilon_1}}{6\epsilon_1^2} + \frac{f_3(m)}{\epsilon_2 f_1(m)}. \tag{9}$$

vii. Function $f_5(.)$ determines the number of random samples from the input sets in the beginning of the algorithm:

$$f_5(m) \quad = \quad \frac{m f_4(m)}{z_{\min}}. \tag{10}$$

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

$$\gamma_3 \quad = \quad \frac{\gamma_2}{2 f_5(m)}. \tag{11}$$

ix. 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:

$$f_6(m) \quad = \quad f_1(m) \left(\frac{24}{\epsilon_1^2} \cdot \ln \frac{2}{\gamma_3} \right). \tag{12}$$

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

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 (α_L, α_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 $\epsilon \in (0, 1)$ to control the accuracy of approximation, and $M = m_1 + m_2 + \cdots + m_m$ as the sum of sizes of input sets.

Output : $sum \cdot M$.

1:      Let $h_1 = f_5(m)$
2:      Let $i = 1$
3:      Let currentThickness$_1 = z_{max}$
4:      Let $s_1 = \frac{m}{currentThickness_1}$
5:      Let $s'_1 = 1$
6:      Let $sum = 0$
7:      Obtain a set R_1 of h_1 random choices of L (see Definition 9)
8:      Stage i
9:          Let $u_i = s_i \cdot f_6(m)$
10:         Select u_i random indices $H_i = \{k_1, \cdots, k_{u_i}\}$ from $\{1, 2, \cdots, m\}$
11:         Compute $S(x, H_i)$ for each $x \in R_i$
12:         Let V_i be the subset of R_i with elements x satisfying $S(x, H_i) \geq \frac{currentThickness_i}{2f_1(m) \cdot m} \cdot u_i$
13:         Let $sum = sum + s'_i \sum\limits_{x \in V_i} \frac{u_i}{S(x, H_i)m}$
14:         Let $currentThickness_{i+1} = \frac{currentThickness_i}{f_1(m)}$
15:         Let $s_{i+1} = \frac{m}{currentThickness_{i+1}}$
16:         Let $h_{i+1} = \frac{h_1}{s_{i+1}}$
17:         If $(|R_i| - |V_i| < h_{i+1})$
18:         Then
19:         {
20:             Let $R_{i+1} = R_i - V_i$
21:             Let $a_i = 1$
22:         }
23:         Else
24:         {
25:             Let R_{i+1} be a set of random h_{i+1} samples from $R_i - V_i$
26:             Let $a_i = \frac{|R_i| - |V_i|}{h_{i+1}}$
27:         }
28:         Let $s'_{i+1} = s'_i \cdot a_i$
29:         Let $i = i + 1$
30:         If $(currentThickness_i < z_{min})$
31:             Return $sum \cdot M$ and terminate the algorithm
32:         Else
33:             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 10 gives some basic properties of the algorithm. Lemma 12 shows that R_1 has random

samples are used so that $F(R_1, h_1, 1)\left(\sum_{i=1}^m m_i\right)$ is an accurate approximation for $\sum_{i=1}^m \sum_{x\in A_i} \frac{1}{T(x,L)}$.

Lemma 10. *The algorithm ApproximateUnion(.) has the following properties:*

    i. $g^*(\epsilon_1)^{\frac{f_4(m)}{f_2(m)}} \le \frac{\epsilon_1}{m^2}$.

    ii. $v(\delta, z_{\min}, z_{\max}, L) = O\left(\frac{\log \frac{z_{\max}}{z_{\min}}}{\delta}\right)$.

    iii. $\frac{2v(\delta, z_{\min}, z_{\max}, L)}{f_2(m)} \le \epsilon_3$ *and* $f_2(m) = O\left(\frac{\log \frac{m}{z_{\min}}}{\delta \epsilon_3}\right)$.

    iv. R_i *contains at most* h_i *items.*

    v. $g^*(\epsilon_2)^{\frac{f_3(m)}{f_1(m)}} \le \frac{\gamma_2}{2}$.

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

    vii. $f_6(m) = O\left(\frac{m^{c_1}}{\epsilon^2}\cdot(\log m)^2(\log \frac{2}{\gamma}+\log m + \log f_4(m))\right)$.

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

    Statement i: It follows from equations (2) and (9).

    Statement ii: By Definition 8, we need $v(\delta, z_{\min}, z_{\max}, L)$ with $z_{min}(1+\delta)^{v(\delta, z_{\min}, z_{\max}, L)} \ge z_{\max}$. Thus, we have $v(\delta, z_{\min}, z_{\max}, L) \le 2\left(\frac{\log \frac{z_{\max}}{z_{\min}}}{\log(1+\delta)}\right) = O\left(\frac{\log \frac{z_{\max}}{z_{\min}}}{\delta}\right)$ since $\log(1+\delta) = \Theta(\delta)$.

    Statement iii: It is easy to see that $\log(1+\delta) = \Theta(\delta)$ and $1 \le z_{max} \le m$. It follows from equation (7), and Statement ii.

    Statement iv: It follows from lines 19 to 27 in the algorithm.

    Statement v: It follows from equation (8).

    Statement vi: By equation (9), Statement iii and equation (3), we have

$$
\begin{aligned}
f_4(m) &= \frac{f_2(m)\log \frac{m^2}{\epsilon_1}}{6\epsilon_1^2} + \frac{f_3(m)}{\epsilon_2 f_1(m)}\\[2mm]
&\le \left(6f_2(m)\cdot \log \frac{m^2}{\epsilon_1}\cdot\left(\frac{\log m}{\epsilon_0}\right)^2 + \frac{f_3(m)}{\epsilon_2 f_1(m)}\right)\\[2mm]
&= O\left(f_2(m)\cdot \log \frac{m^2}{\epsilon_1^2}\cdot\left(\frac{\log m}{\epsilon_0}\right)^2 + \frac{1}{\epsilon_2}\cdot\frac{\ln \frac{2}{\gamma_2}}{\epsilon_2^2}\right)\\[2mm]
&= O\left(\frac{1}{\delta\epsilon_3\epsilon_0^2}(\log \frac{m}{z_{\min}})\cdot \log \frac{m}{\epsilon_1}\cdot(\log m)^2 + \frac{\log m}{\epsilon_0}\cdot\frac{\ln \frac{2}{\gamma_2}}{\epsilon_2^2}\right)\\[2mm]
&= O\left(\frac{1}{\delta\epsilon_3\epsilon_0^2}(\log \frac{m}{z_{\min}})\cdot \log \frac{m}{\epsilon_1}\cdot(\log m)^2 + \frac{\log m}{\epsilon_0}\cdot\frac{\ln \frac{2}{\gamma_2}}{\epsilon_2^2}\right)\\[2mm]
&= O\left(\frac{1}{\epsilon^4}\left(\log \frac{m}{z_{\min}}\right)\cdot \log \frac{m}{\epsilon}\cdot(\log m)^3 + \frac{(\log m)^3(\ln \frac{1}{\gamma} + \log m)}{\epsilon^3}\right).
\end{aligned}
$$

Statement vii: By equation (12), we have

$$f_6(m) = f_1(m)\left(\frac{24}{\epsilon_1^2}\cdot\ln\frac{2}{\gamma_3}\right) \tag{13}$$

$$= O\left(\frac{m^{c_1}}{\epsilon_1^2}\cdot\ln\frac{2}{\gamma_3}\right) \tag{14}$$

$$= O\left(\frac{m^{c_1}}{\epsilon^2}\cdot(\log m)^2(\log\frac{2}{\gamma_2}+\ln f_5(m))\right) \tag{15}$$

$$= O\left(\frac{m^{c_1}}{\epsilon^2}\cdot(\log m)^2(\log\frac{2}{\gamma_2}+\ln m+\ln f_4(m))\right) \tag{16}$$

$$= O\left(\frac{m^{c_1}}{\epsilon^2}\cdot(\log m)^2(\log\frac{2}{\gamma_2}+\log m+\log f_4(m))\right) \tag{17}$$

$$= O\left(\frac{m^{c_1}}{\epsilon^2}\cdot(\log m)^2(\log\frac{2}{\gamma}+\log m+\log f_4(m))\right). \tag{18}$$

∎

Lemma 11 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 11. *The number of rounds of the algorithm is* $O\left(\frac{\log\frac{z_{max}}{z_{min}}}{\log f_1(m)}\right)$.

Proof: By line 3 of the algorithm, we have currentThickness$_1 = z_{max}$. Variable currentThickness$_i$ is reduced by a factor $f_1(m)$ each phase as currentThickness$_{i+1} = \frac{\text{currentThickness}_i}{f_1(m)}$ by line 14 of the algorithm. By the termination condition of line 30 of the algorithm, if y is the number of phases of the algorithm, we have $y \leq y'$, where y' is any integer with $\frac{z_{max}}{f_1(m)^{y'}} < z_{min}$.
Thus, $y = O\left(\frac{\log\frac{z_{max}}{z_{min}}}{\log f_1(m)}\right)$. ∎

Lemma 12 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 12. *With probability at least* $1-\gamma_1$, $F(R_1, h_1, 1)M \in [\frac{(1-\epsilon_0)(1-\alpha_L)(1-\beta_L)}{1+\delta}|A_1\cup\cdots\cup A_m|, (1+\epsilon_0)(1+\alpha_R)(1+\beta_R)(1+\delta)|A_1\cup\cdots\cup A_m|]$.

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(.),

$$\frac{m_i}{m_1+m_2+\cdots+m_m}\frac{1-\alpha_L}{|A_i|} \geq \frac{(1-\beta_L)|A_i|}{M}\frac{1-\alpha_L}{|A_i|}$$
$$= \frac{(1-\beta_L)(1-\alpha_L)}{M}.$$

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

$$\frac{m_i}{m_1+m_2+\cdots+m_m}\frac{1+\alpha_R}{|A_i|} \leq \frac{(1+\beta_R)|A_i|}{M}\frac{1+\alpha_R}{|A_i|}$$
$$= \frac{(1+\beta_R)(1+\alpha_R)}{M}.$$

12

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

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 vi of Definition 8). 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 $\left[\frac{(1-\rho_L)t_j|A'_j|}{M}, \frac{(1+\rho_R)(1+\delta)t_j|A'_j|}{M}\right]$. Let $p_j = \frac{(1+\rho_R)(1+\delta)t_j \cdot \frac{A}{f_2(m)}}{M}$. Since $z_{\min} \le \text{minThickness}(L)$, we have $z_{\min} \le \text{minThickness}(L) \le t_j$. It is easy to see that $mA \ge U$. We have

$$
\begin{aligned}
p_j h_1 &= \frac{(1+\rho_R)(1+\delta)t_j \cdot \frac{A}{f_2(m)}}{M} \cdot \frac{mf_4(m)}{z_{\min}} \\
&\ge \frac{(1+\rho_R)(1+\delta)\cdot f_4(m)mA}{f_2(m)M} \\
&\ge \frac{(1+\rho_R)(1+\delta)\cdot f_4(m)U}{f_2(m)M} \\
&\ge \frac{(1+\rho_R)(1+\delta)\cdot f_4(m)}{f_2(m)(1+\beta_R)} \\
&= \frac{(1+\alpha_R)(1+\delta)\cdot f_4(m)}{f_2(m)}.
\end{aligned}
\tag{19}
$$

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 $R_{1,j}$ be the elements of R_1 and also in A'_j. By Theorem 7, 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 (7), equation (9) and inequality (19)), there are more than $2p_j h_1 = \frac{2(1+\rho_R)(1+\delta)t_j \cdot \frac{A}{f_2(m)}}{M} \cdot h_1$ elements to be chosen from A'_j into R_1. Thus,

$$
F'(R_{1,j}) \le \frac{2p_j h_1}{t_j} = \frac{2(1+\rho_R)(1+\delta)\cdot A}{f_2(m)M} \cdot h_1,
\tag{20}
$$

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

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

We have

$$
\begin{aligned}
p_{j,1} h_1 &= \frac{(1-\rho_L)t_j|A'_j|}{M} \cdot h_1 \\
&> \frac{(1-\rho_L)t_j \frac{A}{f_2(m)}}{M} \cdot h_1 \ge \frac{(1-\rho_L)z_{min}A \cdot h_1}{f_2(m)M} = \frac{(1-\rho_L)z_{min}A \cdot mf_4(m)}{z_{min}f_2(m)M} \\
&\ge \frac{(1-\rho_L)f_4(m)}{f_2(m)(1+\beta_R)}.
\end{aligned}
\tag{21}
$$

13

We have

$$
\begin{aligned}
p_{j,2}h_1 &= \frac{(1+\rho_R)(1+\delta)t_j|A_j'|}{M} \cdot h_1 \\
&> \frac{(1+\rho_R)(1+\delta)t_j \cdot \frac{A}{f_2(m)}}{M} \cdot h_1 \\
&\geq \frac{(1+\rho_R)(1+\delta)z_{min} \cdot A}{f_2(m)M} \cdot h_1 \\
&= \frac{(1+\rho_R)(1+\delta)z_{min} \cdot A}{f_2(m)M} \cdot \frac{m f_4(m)}{z_{min}} \\
&\geq \frac{(1+\rho_R)(1+\delta) \cdot f_4(m)}{f_2(m)(1+\beta_R)} \\
&\geq \frac{(1+\alpha_R)(1+\delta) \cdot f_4(m)}{f_2(m)}. \quad\quad (22)
\end{aligned}
$$

With probability at most $g^*(\epsilon_3)^{p_{j,1} \cdot h_1} \leq \frac{\gamma_1}{4}$ (by equation (7), equation (9) and inequality (21)),

$$
v_j < \frac{(1-\epsilon_3)(1-\rho_L)t_j|A_j'|}{M} \cdot h_1 = (1-\epsilon_3)(1-\rho_L)t_jh_1 \cdot \frac{|A_j'|}{M}.
$$

With probability at most $g^*(\epsilon_3)^{p_{j,2} \cdot h_1} \leq \frac{\gamma_1}{4}$ (by equation (7), equation (9) and inequality (22)),

$$
v_j > \frac{(1+\epsilon_3)(1+\rho_R)(1+\delta)t_j|A_j'|}{M} \cdot h_1 = (1+\epsilon_3)(1+\rho_R)(1+\delta)t_jh_1 \cdot \frac{|A_j'|}{M}.
$$

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

$$
v_j \in \left[(1-\epsilon_3)(1-\rho_L)t_jh_1 \cdot \frac{|A_j'|}{M}, (1+\epsilon_3)(1+\rho_R)(1+\delta)t_jh_1 \cdot \frac{|A_j'|}{M} \right]. \quad\quad (23)
$$

Thus, we have that there are sufficient elements of A_j' to be selected with high probability, which follows from Theorem 6 and Theorem 7.

In the rest of the proof, we assume that inequality (20) holds if the condition of Case 1 holds, and inequality (23) holds if the condition of Case 2 holds.

Now we consider

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

$$= \frac{1}{h_1} \left(\sum_{R_{1,j} \ with \ A'_j \in T_1} \sum_{x \in R_{1,j}} \frac{1}{T(x, L)} + \sum_{R_{1,j} \ with \ A'_j \in T_2} \sum_{x \in R_{1,j}} \frac{1}{T(x, L)} \right)$$

$$\leq \frac{1}{h_1} \frac{2(1 + \rho_R)(1 + \delta)v(\delta, z_{\min}, z_{\max}, L) \cdot A}{f_2(m)M} \cdot h_1$$

$$+ \frac{1}{h_1} \sum_{R_{1,j} \ with \ A'_j \in T_2} \sum_{x \in R_{1,j}} \frac{1}{T(x, L)}$$

$$\leq \frac{1}{h_1} \left(\frac{2(1 + \rho_R)(1 + \delta)v(\delta, z_{\min}, z_{\max}, L) \cdot A}{f_2(m)M} \cdot h_1 + \sum_{R_{1,j} \ with \ A'_j \in T_2} \frac{v_j}{t_j} \right)$$

$$\leq \frac{1}{h_1} \left(\frac{2(1 + \rho_R)(1 + \delta)v(\delta, z_{\min}, z_{\max}, L) \cdot A}{f_2(m)M} \cdot h_1 \right)$$

$$+ \frac{1}{h_1} \left(\sum_{R_{1,j} \ with \ A'_j \in T_2} (1 + \epsilon_3)(1 + \rho_R)(1 + \delta)h_1 \cdot \frac{|A'_j|}{M} \right)$$

$$= \frac{2(1 + \rho_R)(1 + \delta)v(\delta, z_{\min}, z_{\max}, L) \cdot A}{f_2(m)M}$$

$$+ \sum_{R_{1,j} \ with \ A'_j \in T_2} (1 + \epsilon_3)(1 + \rho_R)(1 + \delta) \cdot \frac{|A'_j|}{M}$$

$$\leq \left(\frac{2(1 + \rho_R)(1 + \delta)v(\delta, z_{\min}, z_{\max}, L)}{f_2(m)} + (1 + \epsilon_3)(1 + \rho_R)(1 + \delta) \right) \frac{A}{M}$$

$$= \left(1 + \epsilon_3 + \frac{2v(\delta, z_{\min}, z_{\max}, L)}{f_2(m)} \right)(1 + \rho_R)(1 + \delta)\frac{A}{M} \tag{24}$$

$$\leq (1 + 2\epsilon_3)(1 + \rho_R)(1 + \delta)\frac{A}{M} \tag{25}$$

$$\leq (1 + \epsilon_0)(1 + \rho_R)(1 + \delta)\frac{A}{M}.$$

The transition from (24) to (25) is by Statement iii of Lemma 10. For the lower bound part, we have the following inequalities:

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

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

$$\geq \frac{1}{h_1} \left(\sum_{R_{1,j} \ with \ A'_j \in T_2} \frac{v_j}{(1+\delta)t_j} \right)$$

$$\geq \frac{1}{h_1} \left(\sum_{R_{1,j} \ with \ A'_j \in T_2} (1 - \epsilon_3)(1 - \rho_L)h_1 \cdot \frac{|A'_j|}{(1+\delta)M} \right)$$

$$= \frac{(1-\epsilon_3)(1-\rho_L)}{(1+\delta)M} \left(\sum_{R_{1,j} \ with \ A'_j \in T_2} |A'_j| \right)$$

$$= \frac{(1-\epsilon_3)(1-\rho_L)}{(1+\delta)M} \sum_{R_{1,j} \ with \ A'_j \in T_1} |A'_j| + \frac{(1-\epsilon_3)(1-\rho_L)}{(1+\delta)M} \sum_{R_{1,j} \ with \ A'_j \in T_2} |A'_j|$$

$$- \frac{(1-\epsilon_3)(1-\rho_L)}{(1+\delta)M} \sum_{R_{1,j} \ with \ A'_j \in T_1} |A'_j|$$

$$= \frac{(1-\epsilon_3)(1-\rho_L)}{(1+\delta)M} \left(A - \sum_{R_{1,j} \ with \ A'_j \in T_1} |A'_j| \right)$$

$$\geq \frac{(1-\epsilon_3)(1-\rho_L)}{(1+\delta)M} \left(A - \frac{v(\delta, z_{\min}, z_{\max}, L)A}{f_2(m)} \right)$$

$$= \left(1 - \frac{v(\delta, z_{\min}, z_{\max}, L)}{f_2(m)} \right) \frac{(1-\epsilon_3)(1-\rho_L)}{(1+\delta)} \frac{A}{M} \qquad (26)$$

$$\geq (1-\epsilon_3)\frac{(1-\epsilon_3)(1-\rho_L)}{(1+\delta)} \frac{A}{M} \qquad (27)$$

$$\geq \frac{(1-\epsilon_0)(1-\rho_L)}{(1+\delta)} \frac{A}{M}.$$

The transition from (26) to (27) is by Statement 3 of Lemma 10. Therefore, $F(R_1, h_1, 1)M \in \left[\frac{(1-\epsilon_0)(1-\rho_L)}{(1+\delta)}A, \ (1+\epsilon_0)(1+\rho_R)(1+\delta)A \right]$. $\blacksquare$

Lemma 13 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 13. *After the execution of Stage i, with probability at least $1 - \gamma_2$, we have the following three statements:*

    i. *Every element $x \in R_i$ with $T(x,L) \geq \frac{\text{currentThickness}_i}{4f_1(m)}$ has $S(x, H_i) \in$*
    $\left[(1-\epsilon_1)\frac{T(x,L)}{m}u_i, (1+\epsilon_1)\frac{T(x,L)}{m}u_i \right]$.

    ii. *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.*

16

iii. 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 6 and Theorem 7. 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 i: 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 $P_1 = g^*(\epsilon_1)^{pu_i} \leq \frac{\gamma_3}{2}$ (by equations (8), and (12)), $S(x, H_i) < (1 - \epsilon_1)\frac{T(x,L)}{m}u_i$. With probability at most $P_2 = g^*(\epsilon_1)^{pu_i} \leq \frac{\gamma_3}{2}$ (by equations (8) and (12)), $S(x, H_i) > (1 + \epsilon_1)\frac{T(x,L)}{m}u_i$.

There are at most h_i elements in R_i by Statement iv of Lemma 10. Therefore, with probability at most $h_i(P_1 + P_2) \leq h_1(P_1 + P_2) \leq h_1 \cdot \gamma_3 = \frac{\gamma_2}{2}$,

$$S(x, H_i) \notin \left[(1 - \epsilon_1)\frac{T(x,L)}{m}u_i, (1 + \epsilon_1)\frac{T(x,L)}{m}u_i \right].$$

Statement ii: This statement of the lemma follows from Statement i.

Statement iii: This part of the lemma follows from Theorem 6 and Theorem 7. 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 $g^*(1)^{pu_i} \leq \frac{\gamma_3}{2}$ (by equations (8), and (12)), we have $S(x, H_i) \geq 2pu_i$. There are at most h_i elements in R_i by Statement iv of Lemma 10. Therefore, with probability at most $h_1 \cdot \gamma_3 \leq \frac{\gamma_2}{2}$, there exists one $x \in V_i$ with $T(x,L) < \frac{\text{currentThickness}_i}{4f_1(m)}$ to satisfy $S(x, H_i) \geq 2pu_i$. ∎

Lemma 14. *Let x and y be positive real numbers with $1 \leq y$. Then we have:*

i. $1 - xy < (1 - x)^y$.

ii. If $xy < 1$, then $(1 + x)^y < 1 + 2xy$.

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

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 $(1 - x)^y \geq 1 - yx$. 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 $(1 + x)^y \leq (1 + x)^{\frac{1}{x} \cdot xy} \leq e^{xy} \leq 1 + xy + (xy)^2 \leq 1 + 2xy$.

It is trivial to verify Statement iii. $1 - x_1 - x_2 \leq (1 - x_1)(1 - x_2)$. Clearly, $(1 + x_1)(1 + x_2) = 1 + x_1 + x_2 + x_1 x_2 \leq 1 + 2x_1 + x_2$. ∎

Lemma 15 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 15. *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:*

i. With probability at least $1 - \gamma_2$, $a_i F'(R_{i+1}) \in [(1 - \epsilon_1)F'(R_i - V_i), (1 + \epsilon_1)F'(R_i - V_i)]$, and

ii. With probability at least $1 - 2y\gamma_2$, $\sum_{i=1}^{y} s_i' F'(V_i) \in [(1 - y\epsilon_1)S, (1 + 2y\epsilon_1)S]$, 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 13, with probability at least $1 - \gamma_2$, $R_i - V_i$ does not contain any x with $T(x,L) \geq \frac{\text{currentThickness}_i}{f_1(m)}$.

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 i: 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 (line (19) to line (22)).

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

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

$$
\begin{aligned}
F'(R_i - V_i) &\geq \frac{h'_i}{\frac{\text{currentThickness}_i}{f_1(m)}} \geq \frac{h_{i+1}}{\frac{\text{currentThickness}_i}{f_1(m)}} \\
&= \frac{\frac{h_1}{s_{i+1}} \cdot f_1(m)}{\text{currentThickness}_i} = \frac{f_4(m)}{z_{min}}.
\end{aligned}
\tag{28}
$$

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 6 and Theorem 7, 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 (8)),

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

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,

$$
\begin{aligned}
a_i F'(B_{i+1,j}) &\leq f_1(m) \cdot \frac{2f_3(m)}{z_{min}f_1(m)^2} = \frac{2f_3(m)}{z_{min}f_1(m)} \\
&= \frac{2f_3(m)}{f_4(m)f_1(m)} \cdot \frac{f_4(m)}{z_{min}} \tag{29} \\
&\leq \frac{2f_3(m)}{f_1(m)f_4(m)} \cdot F'(R_i - V_i) \tag{30} \\
&\leq \frac{2\epsilon_2 \cdot F'(R_i - V_i)}{v(\delta, z_{min}, z_{max}, L)}. \tag{31}
\end{aligned}
$$

The transition from (29) to (30) is by inequality (28). The transition from (30) to (31) is by inequality (9).

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

$$
qh_{i+1} = \frac{t_{i,j}}{h'_i} \cdot h_{i+1} = t_{i,j} \cdot \frac{h_{i+1}}{h'_i} \geq t_{i,j} \cdot \frac{h_{i+1}}{h_i} \geq \frac{f_3(m)}{f_1(m)}.
\tag{32}
$$

With probability at most $g^*(\epsilon_2)^{qh_{i+1}} \leq \frac{\gamma_2}{2}$ (by inequality (32) and Statement v of Lemma 10), we have that $|B_{i+1,j}| < (1-\epsilon_2)qh_{i+1}$. With probability at most $g^*(\epsilon_2)^{qh_{i+1}} \leq \frac{\gamma_2}{2}$ (by inequality (32) and Statement v of Lemma 10), we have that $|B_{i+1,j}| > (1+\epsilon_2)qh_{i+1}$. They follow from Theorem 6 and Theorem 7.

We assume $|B_{i+1,j}| \in [(1 - \epsilon_2)qh_{i+1}, (1 + \epsilon_2)qh_{i+1}]$. Thus, $a_i F'(B_{i+1,j}) \in [(1-\epsilon_2)t_{i,j}, (1+\epsilon_2)t_{i,j}]$. So, $a_i F'(B_{i+1,j}) \in \left[\frac{(1-\epsilon_2)F'(R_{i,j})}{1+\delta}, (1 + \epsilon_2)F'(R_{i,j})(1 + \delta) \right]$.

We have

18

$$a_i F'(R_{i+1}) \;=\; a_i \left(\sum_j F'(B_{i+1,j}) \right) \tag{33}$$

$$\leq \;(1+\epsilon_2)(1+\delta)F'(R_i - V_i) + \frac{2\epsilon_2 F'(R_i - V_i)}{v(\delta, z_{\min}, z_{\max}, L)} \cdot v(\delta, z_{\min}, z_{\max}, L) \tag{34}$$

$$\leq \;((1+\epsilon_2)(1+\delta) + 2\epsilon_2)F'(R_i - V_i) \tag{35}$$

$$\leq \;(1+4\epsilon_2)F'(R_i - V_i) \tag{36}$$

$$\leq \;(1+\epsilon_1)F'(R_i - V_i). \tag{37}$$

The transition from (33) to (34) is by inequality (31). The transition from (35) to (36) is based on equation (4). The transition from (36) to (37) is based on equations (3).

We have

$$a_i F'(R_{i+1}) \;=\; a_i \left(\sum_j F'(B_{i+1,j}) \right) \tag{38}$$

$$\geq \;\frac{(1-\epsilon_2)F'(R_i - V_i)}{(1+\delta)} - \frac{2\epsilon_2 F'(R_i - V_i)}{v(\delta, z_{\min}, z_{\max}, L)} \cdot v(\delta, z_{\min}, z_{\max}, L) \tag{39}$$

$$\geq \;\left(\frac{(1-\epsilon_2)}{(1+\delta)} - 2\epsilon_2 \right) F'(R_i - V_i) \tag{40}$$

$$\geq \;(1-4\epsilon_2)F'(R_i - V_i) \tag{41}$$

$$\geq \;(1-\epsilon_1)F'(R_i - V_i). \tag{42}$$

The transition from (38) to (39) is based on inequality (31). The transition from (41) to (42) is based on equations (3).

Statement ii: 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 $a_i F'(R_{i+1}) \in [(1-\epsilon_1)F'(R_i - V_i), (1+\epsilon_1)F'(R_i - V_i)]$.

In order to prove Statement ii, we give an inductive proof that $s'_{k+1}F'(R_{k+1}) + \sum_{i=1}^{k} s'_i F'(V_i) \in [(1-\epsilon_1)^k S, (1+\epsilon_1)^k S]$. It is trivial for $k = 0$. Assume that $s'_k F'(R_k) + \sum_{i=1}^{k-1} s'_i F'(V_i) \in [(1-\epsilon_1)^{k-1}S, (1+\epsilon_1)^{k-1}S]$.

Since $F'(R_k) = F'(R_k - V_k) + F'(V_k)$, we have $a_k F'(R_{k+1}) + F'(V_k) \in [(1-\epsilon_1)F'(R_k), (1+\epsilon_1)F'(R_k)]$.

Thus, we have

$$s'_{k+1}F'(R_{k+1}) + \sum_{i=1}^{k} s'_i F'(V_i) \;=\; s'_{k+1}F'(R_{k+1}) + s'_k F'(V_k) + \sum_{i=1}^{k-1} s_i F'(V_i)$$

$$=\; s'_k(a_k F'(R_{k+1}) + F'(V_k)) + \sum_{i=1}^{k-1} s_i F'(V_i)$$

$$\leq\; (1+\epsilon_1)s'_k F'(R_k) + \sum_{i=1}^{k-1} s_i F'(V_i)$$

$$\leq\; (1+\epsilon_1)\left(s'_k F'(R_k) + \sum_{i=1}^{k-1} s_i F'(V_i) \right)$$

$$\leq\; (1+\epsilon_1)^k S.$$

Similarly, we have

$$
\begin{aligned}
s'_{k+1}F'(R_{k+1}) + \sum_{i=1}^{k} s'_i F'(V_i) &= s'_{k+1}F'(R_{k+1}) + s_k F'(V_k) + \sum_{i=1}^{k-1} s'_i F'(V_i) \\
&= s'_k \left(a_k F'(R_{k+1}) + F'(V_k) \right) + \sum_{i=1}^{k-1} s'_i F'(V_i) \\
&\geq (1-\epsilon_1) s'_k F'(R_k) + \sum_{i=1}^{k-1} s'_i F'(V_i) \\
&\geq (1-\epsilon_1) \left(s'_k F'(R_k) + \sum_{i=1}^{k-1} s'_i F'(V_i) \right) \\
&\geq (1-\epsilon_1)^k S.
\end{aligned}
$$

Thus, we have $s'_{k+1}F'(R_{k+1}) + \sum_{i=1}^{k} s'_i F'(V_i) \in [(1-\epsilon_1)^k S, (1+\epsilon_1)^k S]$.

Therefore, with probability at least $1 - y\gamma_2 - y\gamma_2$, $\sum_{i=1}^{y} s'_i F'(V_i) \in [(1-\epsilon_1)^y S, (1+\epsilon_1)^y S] \subseteq [(1-\epsilon_1 y)S, (1+2\epsilon_1 y)S]$ by Lemma 14.

∎

Lemma 16 gives the time complexity of the algorithm. The running time depends on several parameters.

Lemma 16. *The algorithm ApproximateUnion(.) runs in* $O\left(\frac{m f_4(m) f_6(m)}{z_{\min}} \cdot \left(\frac{\log \frac{z_{\max}}{z_{\min}}}{\log f_1(m)} \right) \right)$ *time.*

Proof: Let y be the total number of stages. By Lemma 11, we have $y = O\left(\frac{\log \frac{z_{max}}{z_{min}}}{\log f_1(m)} \right)$.

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 17 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 17. *The algorithm ApproximateUnion(.) takes* $O\left(\frac{m f_4(m) f_6(m)}{z_{\min}} \cdot \left(\frac{\log \frac{z_{\max}}{z_{\min}}}{\log f_1(m)} \right) \right)$ *time and*

$O\left(\frac{\log \frac{z_{\max}}{z_{\min}}}{\log f_1(m)} \right)$ *rounds such that with probability at least* $1 - \gamma$, *it gives a*

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

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* $f_1(.)$, $f_4(.)$, *and* $f_6(.)$ *are defined in equations (9), (6), and (12), respectively.*

Proof: Let y be the number of stages. By Lemma 13, with probability at least $1 - y\gamma_2$,

$$
sum \in \left[(1-\epsilon_1) \sum_{i=1}^{y} s'_i F'(V_i), (1+\epsilon_1) \sum_{i=1}^{y} s'_i F'(V_i) \right].
$$

By Lemma 15, with probability at least $1 - 2y\gamma_2$,

$$
\sum_{i=1}^{y} s'_i F'(V_i) \in [(1-y\epsilon_1) F(R_1, h_1, 1), (1+2y\epsilon_1) F(R_1, h_1, 1)].
$$

20

By Lemma 12, with probability at least $1 - \gamma_1$,

$$F(R_1, h_1, 1)\left(\sum_{i=1}^{m} m_i\right) \in [\quad \frac{(1-\epsilon_0)(1-\alpha_L)(1-\beta_L)}{1+\delta}|A_1 \cup \cdots \cup A_m|,$$
$$(1+\epsilon_0)(1+\alpha_R)(1+\beta_R)(1+\delta)|A_1 \cup \cdots \cup A_m|].$$

Therefore, with probability at least $1 - \gamma_1 - 2y\gamma_2$,

$$sum \cdot M \in [\quad \frac{(1-y\epsilon_1)(1-\epsilon_0)(1-\epsilon_1)(1-\alpha_L)(1-\beta_L)}{1+\delta} \cdot |A_1 \cup \cdots \cup A_m|,$$
$$(1+2y\epsilon_1)(1+\epsilon_0)(1+\epsilon_1)(1+\alpha_R)(1+\beta_R)(1+\delta)|A_1 \cup \cdots \cup A_m|].$$

Now assume

$$sum \cdot M \in [\quad \frac{(1-y\epsilon_1)(1-\epsilon_0)(1-\epsilon_1)(1-\alpha_L)(1-\beta_L)}{1+\delta} \cdot |A_1 \cup \cdots \cup A_m|,$$
$$(1+2y\epsilon_1)(1+\epsilon_0)(1+\epsilon_1)(1+\alpha_R)(1+\beta_R)(1+\delta)|A_1 \cup \cdots \cup A_m|].$$

By Statement iii of Lemma 14, we have

$$1 - \epsilon \le 1 - y\epsilon_1 - \epsilon_0 - \frac{9}{8}\epsilon_1 \le (1-y\epsilon_1)(1-\epsilon_0)\left(1-\frac{9}{8}\epsilon_1\right) \le \frac{(1-y\epsilon_1)(1-\epsilon_0)(1-\epsilon_1)}{1+\delta},$$

and $(1+2y\epsilon_1)(1+\epsilon_0)(1+\epsilon_1)(1+\delta) \le (1+2y\epsilon_1)(1+2\epsilon_0+\epsilon_1)(1+\delta) \le (1+2y\epsilon_1)(1+4\epsilon_0+2\epsilon_1+\delta) \le (1+8\epsilon_0+4\epsilon_1+2\delta+2y\epsilon_1) \le (1+8\epsilon_0+4\epsilon_1+\epsilon_2+2y\epsilon_1) \le (1+8\epsilon_0+\frac{\epsilon_0}{3}+\frac{\epsilon_0}{3}+\frac{\epsilon_0}{3}) \le 1+9\epsilon_0 \le 1+\epsilon$. Therefore,

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

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 16 and Lemma 11, respectively. ∎

Since $1 \le z_{\min} \le \text{minThickness}(L) \le \text{maxThickness}(L) \le z_{\max} \le m$, we have the following Corollary 18. Its running time is almost linear in the classical model.

Corollary 18. *There is a* $O(\text{poly}(\frac{1}{\epsilon}, \log\frac{1}{\gamma}) \cdot m \cdot (\log m)^{O(1)})$ *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 a* $sum \cdot M \in [(1-\epsilon)(1-\alpha_L)(1-\beta_L) \cdot |A_1 \cup \cdots \cup A_m|, (1+\epsilon)(1+\alpha_R)(1+\beta_R) \cdot |A_1 \cup \cdots \cup A_m|]$.

Proof: We let $f_1(m) = 8$ with $c_1 = 0$ in equation (6). Let $z_{min} = 1$ and $z_{max} = m$. It follows from Theorem 17 and Statements vi and vii of Lemma 10 as we have the inequality (43):

$$\left(\frac{mf_4(m)f_6(m)}{z_{\min}} \cdot \left(\frac{\log\frac{z_{\max}}{z_{\min}}}{\log f_1(m)}\right)\right) = O(\text{poly}(\frac{1}{\epsilon}, \log\frac{1}{\gamma}) \cdot m \cdot (\log m)^{O(1)}). \tag{43}$$

∎

Corollary 19. *For each* $\xi > 0$, *there is a* $O(\text{poly}(\frac{1}{\epsilon}, \log\frac{1}{\gamma}) \cdot m^{1+\xi})$ *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* $sum \cdot M \in [(1-\epsilon)(1-\alpha_L)(1-\beta_L) \cdot |A_1 \cup \cdots \cup A_m|, (1+\epsilon)(1+\alpha_R)(1+\beta_R) \cdot |A_1 \cup \cdots \cup A_m|]$.

Proof: We let $f_1(m) = 8m^{\xi/2}$ with $c_1 = \frac{\xi}{2}$ in equation (6). Let $z_{min} = 1$ and $z_{max} = m$. It follows from Theorem 17 and Statements vi and vii of Lemma 10 as we have the inequality (44):

$$\left(\frac{mf_4(m)f_6(m)}{z_{\min}} \cdot \left(\frac{\log \frac{z_{\max}}{z_{\min}}}{\log f_1(m)} \right) \right) = O(\text{poly}(\frac{1}{\epsilon}, \log \frac{1}{\gamma}) \cdot m^{1+\xi}). \tag{44}$$

∎

An interesting open problem is to find an $O(m)$ time and $O(\log m)$ rounds approximation scheme for $|A_1 \cup A_2 \cup \cdots A_m|$ with a similar accuracy performance as Corollary 18. We were not able to adapt the method from Karp, Luby, and Madras [29] to solve this problem.

5. Approximate Random Sampling for Lattice Points in High Dimensional Ball

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

Before present the algorithms, some definitions are given below.

Definition 20. Let integer $d > 0$ be a dimensional number, $\mathbb{R}^d$ be the $d-$dimensional Euclidean Space.

i. For two points $p, \ q \in \mathbb{R}^d$, define $||p - q||$ to be Euclidean Distance.

ii. A point $p \in \mathbb{R}^d$ is a lattice point if $p = (y_1, ..., y_d)$ with $y_i \in \mathbb{Z}$ for $i = 1, 2, ..., d$.

iii. 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.

iv. Let $q = (\mu_1, \mu_2, ..., \mu_d) \in \mathbb{R}^d$. Define $B_d(r, q, k) = \{(z_1, z_2, ..., z_d) \in \mathbb{R}^d : z_1 = \mu_1, ..., z_{d-k} = \mu_{d-k} \text{ and } \sum_{i=1}^{d} (\mu_i - z_i)^2 \le r^2\}$.

v. 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.

vi. Let λ, l be real numbers. Define $D(\lambda, d, l) = \{(x_1, \cdots, x_d) : (x_1, \cdots, x_d) \text{ with } x_k = i_k + j_k\lambda \text{ for an integer } j_k \in [-l, \ l], \text{ and another arbitrary integer } i_k \text{ for } k = 1, 2, ..., d\}$.

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

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

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 21. 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 22 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 22. 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 there is a bijection bewtten the set of of lattice points inside ball $B_d(r, q, k)$ and 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 bijection 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 23 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 23. Let λ 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 22. ∎

   We define $R(r, p, d)$ be a set of radii r' for the balls that 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 24. For a $d-$dimensional ball $B_d(r, p, d)$ of radius r with center at $p = (x_1, x_2, ..., x_d)$.

   i. 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 25 shows that we can reduce the cardinality of $R(r, p, d)$ from exponential to polynomial when setting the element of the ball's center has same type (i.e. $p \in D(\lambda, d, l)$.)

Lemma 25. 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 $O\left((r + l|\lambda|)^3 l^3 d^3\right)$ 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:

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

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

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

then we have:

$$
\begin{cases}
|Z| \leq dl^2 \\
|Y| \leq 4d(r + l|\lambda|)l \\
|X| \leq d(r + l|\lambda|)^2.
\end{cases}
\tag{45}
$$

   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 (45). Then $R(r, p, d)$ can be generated in $O\left((r + l|\lambda|)^3 l^3 d^3\right)$ time.

   ■

   Lemma 26 is a spacial case of Lemma 25. 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 26. 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 $O\left((r^2 + 1)a^{2m}\right)$ time.

Proof: We have

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

via Lemma 25.

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

$$\begin{cases} |z| \leq a^m \\ |y| \leq a^m \\ |x| \leq (r^2 + 1). \end{cases} \tag{46}$$

Therefore, $|R| \leq (r^2 + 1)a^{2m}$ via inequality (46). Then $R(r, p, d)$ can be generated in $O\left((r^2 + 1)a^{2m}\right)$ time.

∎

Definition 27. For a $d-$dimensional ball $B_d(r, p, d)$ of radius r with center at $p = (x_1, x_2, ..., x_d)$.

    i. Define $p[k] = (0, ..., 0, x_{k+1}, ..., x_d)$ for some integer $k \in [1, d]$.

    ii. Define $Z(r, x, t)$ with $Z(r, x, t)^2 = r^2 - (t - x)^2$ if $|t - x| \leq r$, 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 $C(r_k, p[k], d - k) = \displaystyle\sum_{t \in \mathbb{Z} \ and \ t \in [-r_k + x_{k+1}, \ r_k + x_{k+1}]} C(z(r_k, x_{k+1}, t), p[k + 1], d -$
    $(k + 1))$
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 28. *Assume λ be a real number and $p \in D(\lambda, d, l)$, then there is a $O(r(r + l|\lambda|)^3 l^3 d^4)$ 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 25, and Line 4 has at most $2\lfloor r \rfloor + 1$ items to add up.

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

∎

Remark: When $\lambda = \frac{1}{\pi}$, this is a specail case of Theorem 28, and the running time of the algorithm is $O(r(r + l|\lambda|)^3 l^3 d^4)$. 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 λ is a irrational number.

Theorem 29 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 29. *Assume $\lambda = a^{-m}$ and $p \in D^{**}(\lambda, d)$, where m and a are integers with $a \geq 2$, then there is a $O(r^3 a^{2m} d)$ 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 26, and Line 4 has at most $2\lfloor r \rfloor + 1$ items to add up.

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

Corollary 30. *Assume* $\lambda = 10^{-m}$ *and* $p \in D^{**}(\lambda, d)$, *where* m *is a integer, then there is a* $O(r^3 10^{2m} d)$ *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 β is used to control the accuracy of approximation.

Some definitions are presented before prove theorems.

Definition 31. For each lattice point $q = (y_1, y_2, ..., y_d) \in \mathbb{R}^d$ with $y_i \in \mathbb{Z}$ for $i = 1, 2, ..., d$.

    i. Define $Cube(q)$ to be the d−dimensional unit cube with center at $\left(y_1 + \frac{1}{2}, ..., y_d + \frac{1}{2}\right)$.

    ii. Define $I(B_d(r, p, d)) = \{q \mid Cube(q) \subset B_d(r, p, d)\}$.

    iii. Define $E(B_d(r, p, d)) = \{q \mid Cube(q) \notin I(B_d(r, p, d)) \text{ and } Cube(q) \cap B_d(r, p, d) \neq \emptyset\}$.

Theorem 32 gives an $(1+\beta)$−approximation with running time $O(d)$ 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 32. *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* $O(d)$ *for an arbitrary center* p *when* $r > \frac{2d^{\frac{3}{2}}}{\beta}$.

Proof: Let $|I(B_d(r, p, d))|$ be 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 31, 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 Euler's gamma function. Then

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

Similarly, we have

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

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

$$
(1 - \beta) C(r, p, d) \leq V_d(r) \leq (1 + \beta) C(r, p, d) \tag{47}
$$

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

It takes $O(d)$ to compute $V_d(r) = f(d) \cdot r^d$, since it takes $O(d)$ to compute $f(d)$ where $f(d) = \pi^{\frac{d}{2}} \Gamma \left(\frac{1}{2} d + 1\right)^{-1}$. Therefore, the algorithm takes $O(d)$ running time to approximate $C(r, p, d)$ becasue of Equation (47). ∎

Theorem 33. *There is an $(1+\beta)$-approximation algorithm with running time $\mathrm{O}(d)$ to approximate $C(r,p,d)$ 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 $\mathrm{O}\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)$ 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 28.

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 32. ∎

Corollary 34. *There is a dynamic programming algorithm to count $C(r,p,d)$ of $B_d(r,p,d)$ with running time $\mathrm{O}\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)$ 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 subsections 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 $q = (y_1,y_2,...,y_{d-t},y_{d-t+1},x_{d-t+2},...,x_d)$.

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 35. *For an arbitrary $\beta \in (0, 1)$, assume λ be a real number and $p \in D(\lambda, d, l)$, then there is a* $O\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta} + l|\lambda|\right)^3\right)$ *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 $O\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta} + l|\lambda|\right)^3\right)$ to compute $C(r, p, d)$ via Theorem 33, then algorithm Small-BallRandomLatticePoint(.) takes $O\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta} + l|\lambda|\right)^3\right) + O(d)$ running time. Thus, the algorithm takes $O\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta} + l|\lambda|\right)^3\right)$ 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 α 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 36. 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

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

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 37 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 37. *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 $(1-\beta)C(r,q,k) \le P(r,q,k) \le (1+\beta)C(r,q,k)$.*

Proof: Two cases are considered.

Case 1: If $r \le \frac{2d^{\frac{3}{2}}}{\beta}$, we have $P(r,q,k) = C(r,q,k)$ via Definition 36.

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

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

via Theorem 32, where $V_k(r)$ be the volume of $k-$dimensional ball $B_d(r,q,k)$ with radius r.

Therefore, we have

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

because $P(r,q,k) = V_k(r)$ via Definition 36.

By combining the above two cases, we conclude that:

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

■

Lemma 38 shows that for two $k-$dimensional balls, if their radius are almost equal, then the number of their lattice points also are almost equal.

Lemma 38. *For an arbitrary $\beta \in (0,1)$ and a real number δ, 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 $r'' \le r' \le (1+\delta)r''$, then $C(r'',q,k) \le C(r',q,k) \le \frac{1+\beta}{1-\beta}(1+\delta)^k C(r'',q,k)$.*

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:

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

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

$$\begin{cases} \frac{1}{1+\beta}V_k(r') \le C(r',q,k) \le \frac{1}{1-\beta}V_k(r') \\ \frac{1}{1+\beta}V_k(r'') \le C(r'',q,k) \le \frac{1}{1-\beta}V_k(r'') \end{cases} \tag{48}$$

via Theorem 32,

Plugging inequality (48) to above inequality, then we have

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

and we also have

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

Therefore,

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

∎

Definition 39. 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:

i. $a_1 = a, a_{i+1} = b_i + 1$ for $i = 1, \cdots, w - 1$.

ii. For any $x, y \in \{a_i, b_i\}$, $r^2 - (x - c)^2 \leq (1 + \delta)^2 (r^2 - (y - c)^2)$ and $r^2 - (y - c)^2 \leq (1 + \delta)^2 (r^2 - (x - c)^2)$.

iii. 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 $1 \leq i \leq d - t$, and $y_i \in \mathbb{Z}$ with $d - t + 1 \leq i \leq d$, $\alpha \in (0, 1)$ 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 $\left(r, y_{d-t+1}, 1 + \frac{\epsilon_4}{g(d)}\right)$-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 $\epsilon_4 \in (0, 1)$ 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 $\left(r, y_{d-t+1}, 1 + \frac{\epsilon_4}{g(d)}\right)$-partitions in line 3 is as the following pictures:
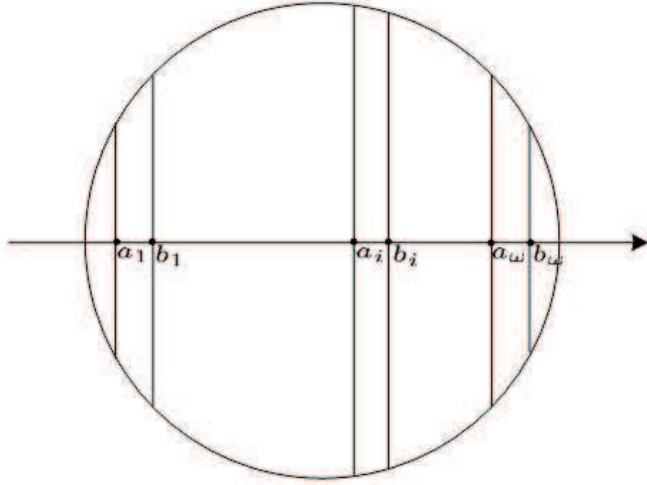
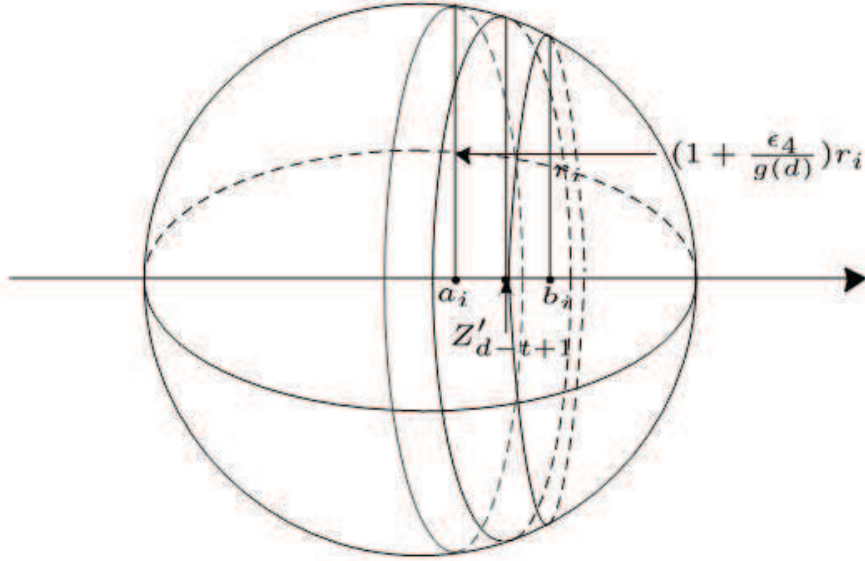Figure 3: Example of $\left(r, y_{d-t+1}, 1+\frac{\epsilon_4}{g(d)}\right)$-Partitions in 2D

Figure 4: Example of $\left(r, y_{d-t+1}, 1+\frac{\epsilon_4}{g(d)}\right)$-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 40. Let integer $d > 0$ be a dimensional number, $\mathbb{R}^d$ be the $d-$dimensional Euclidean Space.

    i. 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

$$x_i' = \begin{cases} \lfloor x_i \rfloor & x_i - \lceil x_i \rceil \leq \frac{1}{2} \\ \lceil x_i \rceil & x_i - \lceil x_i \rceil > \frac{1}{2}, \end{cases} \quad \text{for } x_i \geq 0 \text{ or } x_i' = \begin{cases} \lceil x_i \rceil & |x_i| - [|x_i|] \leq \frac{1}{2} \\ \lfloor x_i \rfloor & |x_i| - [|x_i|] < \frac{1}{2}, \end{cases} \quad \text{for } x_i < 0,$$

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 $1 \leq i \leq d$, $\alpha \in (0, 1)$ 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.

1:      Let q be the nearest lattice point of p in $\mathbb{R}^d$
2:      Repeat
3:          Let $s =$ RecursiveBigBallRandomLatticePoint$(r + \sqrt{d}, q, d)$
4:      Until $s \in B_d(r, p, d)$
5:      Return s

Theorem 41. *For an arbitrary $\alpha \in (0, 1)$, there is an algorithm with runing time* $\mathrm{O}\left(\frac{d^3 \log r}{\alpha}\right)$ *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
$$r_i'^2 = \begin{cases} r^2 - (y_{d-t+1} - a_i)^2 & if \ a_i \leq y_{d-t+1} \\ r^2 - (y_{d-t+1} - b_i)^2 & otherwise, \end{cases},$$
$$p_i' = \begin{cases} (z_1, z_2, ..., z_{d-t}, a_i, y_{d-t+2}, ..., y_d) & if \ a_i \leq y_{d-t+1} \\ (z_1, z_2, ..., z_{d-t}, b_i, y_{d-t+2}, ..., y_d) & otherwise, \end{cases}$$
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 $r_i' = \frac{r_i}{1 + \frac{\epsilon_4}{g(d)}}$, 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 $r_i' = \frac{r_i}{1 + \frac{\epsilon_4}{g(d)}}$, then

$$\frac{1 - \beta}{1 + \beta}\left(1 + \frac{\epsilon_4}{g(d)}\right)^{-(t-1)} \sum_i C(r_i, p_i, t - 1)v(i) \leq C(r_i, p_i, t)$$

and

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

via Lemma 38, where $\delta = 1 + \frac{\epsilon_4}{g(d)}$.

Via Lemma 37 we have

$$\left(\frac{1 - \beta}{1 + \beta}\right)^2 \left(1 + \frac{\epsilon_4}{g(d)}\right)^{-(t-1)} \sum_i P(r_i, p_i, t - 1)v(i) \leq P(r_i, p_i, t)$$

and

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

33

Thus, we have

$$\left(\frac{1-\beta}{1+\beta}\right)^2 \left(1+\frac{\epsilon_4}{g(d)}\right)^{-(t-1)} \leq \frac{P(r_i,p_i,t)}{\sum\limits_i P(r_i,p_i,t-1)v(i)} \leq \frac{1+\beta}{1-\beta}.$$

From above inequality, we have

$$\left(\frac{1-\beta}{1+\beta}\right)^2 \left(1+\frac{\epsilon_4}{g(d)}\right)^{-(t-1)} \frac{1}{P(r_i,p_i,t)} \leq \frac{1}{\sum\limits_i P(r_i,p_i,t-1)v(i)} \leq \frac{1+\beta}{1-\beta}\frac{1}{P(r_i,p_i,t)}.$$

Via Lemma 37 we have

$$\frac{(1-\beta)^2}{(1+\beta)^3}\left(1+\frac{\epsilon_4}{g(d)}\right)^{-(t-1)}\frac{1}{C(r_i,p_i,t)} \leq \frac{1}{\sum\limits_i P(r_i,p_i,t-1)v(i)} \leq \frac{1+\beta}{(1-\beta)^2}\frac{1}{C(r_i,p_i,t)}.$$

Let $g(d) = d^2$, $\epsilon_4 = \frac{\alpha}{4}$ and $\beta = \frac{\alpha}{\alpha+16d+16}$. Since Algorithm RecursiveBigBallRandomLattice-Point(.) has d iteration, we can generate a random lattice point with bias of probability as:

$$\frac{P(r_i,p_i,d-1)}{\sum\limits_i P(r_i,p_i,d-2)v(i)} \cdot \frac{P(r_i,p_i,d-2)}{\sum\limits_i P(r_i,p_i,d-1)v(i)} \cdots \frac{P(r_i,p_i,0)}{\sum\limits_i P(r_i,p_i,0)v(i)}$$

$$\leq \quad \frac{1+\beta}{(1-\beta)^2}\frac{1}{C(r,p,d)} \cdot \left(\frac{1+\beta}{1-\beta}\right)^{d-1} \cdot P(r_i,p_i,0)$$

$$\leq \quad \frac{1}{1-\beta}\frac{1}{C(r,p,d)} \cdot \left(\frac{1+\beta}{1-\beta}\right)^{d} \cdot (1+\beta)C(r_i,p_i,0)$$

$$= \quad \left(\frac{1+\beta}{1-\beta}\right)^{d+1}\frac{1}{C(r,p,d)}$$

$$= \quad \left(1+\frac{2\beta}{1-\beta}\right)^{d+1}\frac{1}{C(r,p,d)}$$

$$\leq \quad e^{\frac{2\beta}{1-\beta}(d+1)}\frac{1}{C(r,p,d)}$$

$$\leq \quad \left(1+\frac{4\beta}{1-\beta}(d+1)\right)\frac{1}{C(r,p,d)}$$

$$\leq \quad (1+\alpha)\frac{1}{C(r,p,d)}$$

34

and

$$\frac{P(r_i, p_i, d-1)}{\sum_i P(r_i, p_i, d-2)v(i)} \cdot \frac{P(r_i, p_i, d-2)}{\sum_i P(r_i, p_i, d-1)v(i)} \cdots \frac{P(r_i, p_i, 0)}{\sum_i P(r_i, p_i, 0)v(i)}$$

$$\geq \frac{(1-\beta)^2}{(1+\beta)^3}\left(1+\frac{\epsilon_4}{g(d)}\right)^{-(d-1)}\frac{1}{C(r,p,d)}\left(\frac{1-\beta}{1+\beta}\right)^{2(d-1)}\left(1+\frac{\epsilon_4}{g(d)}\right)^{\frac{-(d-1)(d-2)}{2}}(1-\beta)C(r_i, p_i, 0)$$

$$= \left(\frac{1-\beta}{1+\beta}\right)^{2d+1}\left(1+\frac{\epsilon_4}{g(d)}\right)^{\frac{-(d-1)d}{2}}\frac{1}{C(r,p,d)}$$

$$\geq \left(1-\frac{2\beta}{1+\beta}\right)^{2d+1}\left(1+\frac{\epsilon_4}{g(d)}\right)^{\frac{-d^2}{2}}\frac{1}{C(r,p,d)}$$

$$\geq \left(1-\frac{2\beta}{1+\beta}\right)^{2d}\left(1+\frac{\epsilon_4}{g(d)}\right)^{-d^2}\frac{1}{C(r,p,d)}$$

$$\geq \left(1-\frac{4\beta d}{1+\beta}\right)\left(1-\frac{\epsilon_4 d^2}{g(d)}\right)\frac{1}{C(r,p,d)}$$

$$\geq \left(1-\frac{4\beta d}{1+\beta}\right)(1-\epsilon_4)\frac{1}{C(r,p,d)}$$

$$\geq \left(1-\frac{4\beta d}{1+\beta}-\epsilon_4\right)\frac{1}{C(r,p,d)}$$

$$\geq \left(1-\frac{4\beta d}{1+\beta}\right)(1-\epsilon_4)\frac{1}{C(r,p,d)}$$

$$\geq (1-\alpha)\frac{1}{C(r,p,d)}.$$

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

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

In line 3 of algorithm RecursiveBigBallRandomLatticePoint(.), it forms a $\left(r, y_{d-t+1}, 1+\frac{\epsilon_4}{g(d)}\right)$-partition $I_1, \cdots, I_w$ 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 $\frac{r}{\left(1+\frac{\epsilon_4}{g(d)}\right)^w}\leq 1$. Solving w, we have $w\geq\frac{g(d)\log r}{\epsilon_4}$. And there are d iterations in algorithm RecursiveBigBallRandomLatticePoint(.).

Thus, the running time of the algorithm is $O\left(\frac{g(d)\log r}{\epsilon_4}\cdot d\right)=O(\frac{d^3\log r}{\epsilon_4})=O\left(\frac{d^3\log r}{\alpha}\right)$. ∎

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 35 with $\beta=0$.

Theorem 42. *For arbitrary $\alpha\in(0,1)$, and $\alpha'\in(0,1)$, there is an $(1+\alpha')-$bias algorithm with runing time $O\left(\frac{d^3\log(r+\sqrt{d})}{\alpha}\right)$ 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 32, we have

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

35

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

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 BigBallRandom-LatticePoint(.) 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 41. Then the bias 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

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

and

$$\frac{\Pr(p_i)}{\sum_i \Pr(p_i)} \geq \frac{(1-\alpha)\frac{1}{C(r',q,d)}}{(1+\alpha)\frac{1}{C(r',q,d)}C(r,p,d)}$$

$$= \frac{1-\alpha}{1+\alpha}\frac{1}{C(r,p,d)}$$

$$= \left(1 - \frac{2\alpha}{1+\alpha}\right)\frac{1}{C(r,p,d)}$$

$$\geq \left(1 - \frac{2\alpha}{1-\alpha}\right)\frac{1}{C(r,p,d)}.$$

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 $O\left(\frac{d^3 \log(r+\sqrt{d})}{\alpha}\right)$ running time to generate a random lattice point inside a d−dimensional ball $B_d(r+\sqrt{d},p,d)$ with a lattice point center via Theorem 41. Thus, the algorithm BigBallRandomLatticePoint(.) takes $O\left(\frac{d^3 \log(r+\sqrt{d})}{\alpha}\right)$ running time to generate a random lattice. ∎

Theorem 43. *For an arbitrary $\alpha \in (0,1)$, there is an algorithm with runing time $O\left(\frac{d^3 \log(r+\sqrt{d})}{\alpha}\right)$ and $(1+\alpha)$−bias for a d−dimensional ball $B_d(r,q,d)$ to generate a random lattice pointo f radius $r > \frac{2d^{\frac{3}{2}}}{\alpha}$ with a arbitrary center; and there is a $O\left(\frac{1}{\beta}d^{\frac{11}{2}}l^3\left(\frac{2d^{\frac{3}{2}}}{\beta}+l|\lambda|\right)^3\right)$ 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 42.

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 35. ∎

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

In this section, we apply the algorithm developed in Section 4 to count the total number of lattice point in the union of high dimensional balls.

Theorem 44. *There is a $O\left(\text{poly}\left(\frac{1}{\epsilon}, \log\frac{1}{\gamma}\right) \cdot m \cdot (\log m)^{O(1)}\right)$ time and $O(\log m)$ 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$\cdot M \in [(1-\epsilon)(1-\alpha_L)(1-\beta_L)\cdot|B_1\cup\cdots\cup B_m|, (1+\epsilon)(1+\alpha_R)(1+\beta_R)\cdot|B_1\cup\cdots\cup B_m|]$, 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 33 and Theorem 43, we have m_i for each ball B_i with

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

and biased random generators with

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

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 17. ∎

5.4. 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 45. *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 46. 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 $\{(x_1, \cdots, x_d) : (x_1, \cdots, x_d) \in \mathbb{R}^d \text{ and } \sum_{i=1}^{d}(x_i - c_i)^k \le r\}$.

Theorem 47. *Let k be an even number at least 2. Then we have:*

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

   ii. *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 $\sum_{i=1}^{n}(a_i - d_i)^k \le 2(1 - \delta)^k + \delta^k + (n - 3) \cdot (\frac{1}{2})^k$. 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:

$$\begin{cases} (2 - \delta)^2 > (1 + \delta)^k > 2(1 - \delta)^k + \delta^k \\ (1 + \frac{1}{2})^k > 2(1 - \delta)^k + \delta^k + (\frac{1}{2})^k. \end{cases} \tag{49}$$

This is because we have the following equalities:

$$\begin{cases} (1 + \delta)^2 = 1.69, \\ 2(1 - \delta)^2 + \delta^2 = 2 \times 0.49 + 0.09 = 1.07, \\ 2(1 - \delta)^2 + \delta^2 + (\frac{1}{2})^2 = 1.07 + 0.25 = 1.32, \\ (1 + \frac{1}{2})^2 = 2.25. \end{cases} \tag{50}$$

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

i. 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 (50).

ii. 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 (50).

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

i. Case 1. Y satisfies C.

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

ii. 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. ∎

6. 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. 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 $|A_{i_1} \cup A_{i_2} \cup \cdots \cup A_{i_k}| \ge (1-a)|A_{j_1} \cup A_{j_2} \cup \cdots \cup A_{j_k}|$, where $A_{j_1}, A_{j_2}, \cdots, A_{j_k}$ is a solution with maximum size of union.

Theorem 48. *[17] Let ρ 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*

i. $|\cup_{j \in H} A_j| \ge \left(1 - (1-\frac{\beta}{k})^k - \xi\right) C^*(L, k)$ *and* $|H| = k$,

ii. $((1-\alpha_L)(1-\delta_L) - \xi)|\cup_{j \in H} A_j| \le z \le ((1+\alpha_R)(1+\delta_R) + \xi)|\cup_{j \in H} A_j|$, *and*

iii. *Its complexity is* $(T(\xi, \gamma, k, m), R(\xi, \gamma, k, m), Q(\xi, \gamma, k, m))$ *with*

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

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 49 to transform the approximation ratio given by Theorem 48 to constant $(1-\frac{1}{e})$ to match the classical ratio for the maximum coverage problem.

Lemma 49. *For each integer $k \geq 2$, and real $b \in [0, 1]$, we have:*

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

   ii. 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 $x \in (0, \frac{1}{4})$,

$$e^{-x} \leq 1 - \eta x. \tag{51}$$

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

$$
\begin{aligned}
(1 - \frac{b}{k})^k &= e^{k \ln(1 - \frac{b}{k})} = e^{k(-\frac{b}{k} - \frac{b^2}{2k^2} - \frac{b^3}{3k^3} - \cdots)} = e^{-b - \frac{b^2}{2k} - \frac{b^3}{3k^2} - \cdots} \\
&\leq e^{-b - \frac{b}{2k}} = e^{-1} \cdot e^{1 - b - \frac{b}{2k}} \tag{52} \\
&\leq e^{-1} \cdot (1 - \eta \cdot (b + \frac{b}{2k} - 1)) \leq \frac{1}{e} - \frac{\eta}{e}(b + \frac{b}{2k} - 1). \tag{53}
\end{aligned}
$$

Note that the transition from (52) to (53) is based on inequality (51).

   The part ii follows from part i. 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 50. *There is a poly(λ, 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 $D(\lambda, d, l)$.*

Proof: [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 48, Lemma 49, Theorem 33 and Theorem 43. ∎

7. 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.

8. Acknowledgements

References

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

[2] N. Alon, Y. Matias, and M. 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] G. E. Andrews, S. B. Ekhad, and D. 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] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. 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] Z. Bar-Yossef, R. 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] J. Beck. On a lattice point problem of l. moser I. *Combinatorica*, 8(1):21–47, 1988.

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

[9] S. R. Buss and L. 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 R. Narasimhan. On lattice-points in a random sphere. *Bull. Amer. Math. Soc.*, 73(1):68–71, 1967.

[11] J.-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] S. 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. Corráadi and I. Kátai. A comment on k. s. gangadharan's paper entitled "two classical lattice point problems". *Magyar Tud. Akad. Mat. Fiz. Oszt. Kozl*, 17.

[14] P. Flajolet, É. Fusy, O. Gandoue, and F. 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] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[16] L. Fortnow and N. 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] B. Fu. Partial sublinear time approximation and inapproximation for maximum coverage. arXiv:1604.01421, April 5, 2016.

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

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

[21] P. J. Haas, J. F. Naughton, S. Seshadri, and L. 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] J. 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] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[25] Z. Huang, W. M. Tai, and K. 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] D. M. Kane, J. Nelson, and D. 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] R. M. Karp, M. Luby, and N. 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] A. 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] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 2000.

[33] G. Szegö. Beiträge zur theorie der laguerreschen polynome ii: Zahlentheoretische anwendungen. *Math. Z.*, 25(1).

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

[35] L. 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. *Gitterpunkte in mehrdimensionalen Kugeln*. Instytut Matematyczny Polskiej Akademi Nauk(Warszawa), 1957.

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

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