SKETCHES FOR AGGREGATE ESTIMATIONS OVER DATA STREAMS

By

FLORIN I. RUSU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2009

TABLE OF CONTENTS

page

LIST OF TABLES . . . 6

LIST OF FIGURES . . . 7

ABSTRACT . . . 9

CHAPTER

1 INTRODUCTION . . . 10

1.1 Contributions . . . 13
1.2 Outline . . . 15

2 PRELIMINARIES . . . 16

2.1 Problem Formulation . . . 16
2.2 Sketches . . . 17
2.3 Confidence Bounds . . . 18
2.3.1 Distribution-Independent Confidence Bounds . . . 19
2.3.2 Distribution-Dependent Confidence Bounds . . . 20
2.3.3 Mean Estimator . . . 21
2.3.4 Median Estimator . . . 22
2.3.5 Mean vs Median . . . 24
2.3.6 Median of Means Estimator . . . 25
2.3.7 Minimum Estimator . . . 27

3 PSEUDO-RANDOM SCHEMES FOR SKETCHES . . . 29

3.1 Generating Schemes . . . 30
3.1.1 Problem Definition . . . 30
3.1.2 Orthogonal Arrays . . . 31
3.1.3 Abstract Algebra . . . 33
3.1.4 Bose-Chaudhuri-Hocquenghem Scheme (BCH) . . . 34
3.1.5 Extended Hamming 3-Wise Scheme (EH3) . . . 36
3.1.6 Reed-Muller Scheme . . . 36
3.1.7 Polynomials over Primes Scheme . . . 37
3.1.8 Toeplitz Matrices Scheme . . . 38
3.1.9 Tabulation Based Schemes . . . 38
3.1.10 Performance Evaluation . . . 39
3.2 Size of Join using AGMS Sketches . . . 41
3.2.1 Variance for BCH5 . . . 43
3.2.2 Variance for BCH3 . . . 43
3.2.3 Variance for EH3 . . . 46
3.2.4 Empirical Evaluation . . . 50
3.3 Conclusions . . . 51

4 SKETCHING SAMPLED DATA STREAMS . . . 55

4.1 Sampling . . . 58
4.1.1 Generic Sampling . . . 58
4.1.2 Bernoulli Sampling . . . 59
4.1.3 Sampling with Replacement . . . 61
4.2 Sketches . . . 63
4.3 Sketches over Samples . . . 64
4.3.1 Generic Sampling . . . 65
4.3.2 Bernoulli Sampling . . . 71
4.3.3 Sampling with Replacement . . . 73
4.3.4 Discussion . . . 75
4.4 Experimental Evaluation . . . 76
4.5 Conclusions . . . 78

5 STATISTICAL ANALYSIS OF SKETCHES . . . 82

5.1 Sketches . . . 85
5.1.1 Basic AGMS Sketches . . . 86
5.1.2 Fast-AGMS Sketches . . . 88
5.1.3 Fast-Count Sketches . . . 89
5.1.4 Count-Min Sketches . . . 90
5.1.5 Comparison . . . 92
5.2 Statistical Analysis of Sketch Estimators . . . 92
5.2.1 Basic AGMS Sketches . . . 93
5.2.2 Fast-AGMS Sketches . . . 94
5.2.3 Count-Min Sketches . . . 99
5.2.4 Fast-Count Sketches . . . 102
5.3 Empirical Evaluation . . . 103
5.3.1 Testbed and Methodology . . . 104
5.3.2 Results . . . 105
5.3.3 Discussion . . . 108
5.4 Conclusions . . . 109

6 SKETCHES FOR INTERVAL DATA . . . 116

6.1 Sketch Applications . . . 118
6.1.1 Size of Spatial Joins . . . 119
6.1.2 Selectivity Estimation for Building Dynamic Histograms . . . 119
6.2 Problem Formulation . . . 120
6.3 Dyadic Mapping (DMAP) . . . 121
6.3.1 Dyadic Intervals . . . 121
6.3.2 Dyadic Mapping Method . . . 127
6.3.3 Algorithm DMAP COUNTS . . . 130
6.4 Fast Range-Summable Generating Schemes . . . 131
6.4.1 Scheme BCH3 . . . 132

6.4.2 Scheme EH3 . . . 140
6.4.3 Four-Wise Independent Schemes . . . 143
6.4.4 Scheme RM7 . . . 144
6.4.5 Approximate Four-Wise Independent Schemes . . . 145
6.4.6 Empirical Evaluation . . . 147
6.5 Fast Range-Summation Method . . . 148
6.6 Experimental Results . . . 153
6.7 Discussion . . . 157
6.8 Conclusions . . . 158

7 CONCLUSIONS . . . 167

REFERENCES . . . 169

BIOGRAPHICAL SKETCH . . . 173

LIST OF TABLES

Table page

3-1 Orthogonal array $\mathrm{OA}(8, 4, 2, 3)$. 52

3-2 Truth table for the function $x_1 \oplus x_2$. 52

3-3 Generation time and seed size. 52

5-1 Families of ± 1 random variables. 110

5-2 Families of hash functions. 110

5-3 Expected theoretical performance. 110

5-4 Expected statistical/empirical performance. 110

6-1 Sketching time per interval. 162

6-2 Sketching time per interval (ns). 162

LIST OF FIGURES

Figure page

2-1 AGMS Sketches. 28

3-1 EH3 error. 53

3-2 BCH5 error. 53

3-3 Scheme comparison (full). 54

3-4 Scheme comparison (detail). 54

4-1 Size of join variance. 79

4-2 Self-join size variance. 79

4-3 Size of join error. 80

4-4 Self-join size error. 80

4-5 Size of join sample size. 81

4-6 Self-join size sample size. 81

5-1 The distribution of AGMS sketches for self-join size. 111

5-2 The distribution of F-AGMS sketches for self-join size. 111

5-3 The distribution of CM sketches for self-join size. 111

5-4 The distribution of FC sketches for self-join size. 111

5-5 F-AGMS kurtosis. 112

5-6 F-AGMS efficiency. 112

5-7 Confidence bounds for F-AGMS sketches as a function of the skewness of the data. 113

5-8 Accuracy as a function of the Zipf coefficient for self-join size estimation. 113

5-9 Accuracy as a function of the correlation coefficient for size of join estimation. . 113

5-10 Relative performance for size of join estimation. 114

5-11 Accuracy as a function of the skewness of the data for size of join estimation. . . 114

5-12 Accuracy as a function of the available space budget. 114

5-13 Confidence bounds for CM sketches as a function of the skewness of the data. . 115

5-14 Update time as a function of the number of counters in a sketch that has only one row. . . . 115

6-1 The number of polynomial evaluations. . . . 162

6-2 The set of dyadic intervals over the domain $I = \{0, 1, \ldots, 15\}$. . . . 163

6-3 Dyadic mappings. . . . 163

6-4 Fast range-summation with domain partitioning. . . . 163

6-5 LANDO $\bowtie$ LANDC. . . . 164

6-6 LANDO $\bowtie$ SOIL. . . . 164

6-7 LANDC $\bowtie$ SOIL. . . . 165

6-8 Selectivity estimation. . . . 165

6-9 Accuracy of sketches for interval data. . . . 166

6-10 Update time per interval as a function of the number of partitions for the SOIL data set. . . . 166

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

SKETCHES FOR AGGREGATE ESTIMATIONS OVER DATA STREAMS

By

Florin I. Rusu

May 2009

Chair: Alin Dobra
Major: Computer Engineering

In this work, we present methods to speed-up the sketch computation. Sketches are randomized algorithms that use small amount of memory and that can be computed in one pass over the data. Frequency moments represent important distributional characteristics of the data that are required in any statistical modeling method. This work focuses on AGMS-sketches used for the estimation of the second frequency moment.

Fast-AGMS sketches use hash functions to speed-up the computation by reducing the number of basic estimators that need to be updated. We show that hashing also changes the distribution of the estimator, thus improving the accuracy by orders of magnitude. The second method to speed-up the sketch computation is related to the degree of randomization used to build the estimator. We show that by using 3-wise independent random variables instead of the proposed 4-wise, significant improvements are obtained both in computation time and memory usage while the accuracy of the estimator stays the same. The last speed-up method we discuss is combining sketches and sampling. Instead of sketching the entire data, the sketch is built only over a sample of the data. We show that the accuracy of the estimator is not drastically affected even when the sample contains a small amount of the original data.

When the three speed-up methods are put together, it is possible to sketch streams having input rates of millions of tuples per second in small memory while providing similar accuracy as the original AGMS sketches.

CHAPTER 1
INTRODUCTION

Over the last decade, the development of computers and information technology has been tremendous. The clocks of processors reached un-imaginable speed rates, the capacity of the storage devices has increased exponentially, and the amount of data transferred over the communication networks has scaled to un-thinkable sizes. Although we are able to generate, store, and transport rivers of data, we do not have the computation power to efficiently process them. We are flooded with data we cannot process and analyze exactly. Thus, new models of computation have been proposed.

The data stream model [6, 47] changes the perspective we look at the data and computation. In the data stream model [6, 47], the input data are not available for random access from disk or memory, but rather arrive as one or more continuous data streams. Data streams differ from the conventional relational model in several ways. First, the data elements in the stream arrive online in a completely arbitrary order. Second, data streams are potentially unbounded in size. And third, once an element from a data stream has been processed, it is discarded or archived, making it difficult to be subsequently referenced unless it is explicitly stored in memory, which is typically small relative to the size of the data stream. In other words, data stream processing implies only *one pass* over the data and using *small space*. Given these stringent requirements, approximation and randomization are key ingredients in processing rapid data streams, contrary to traditional applications which focus largely on providing precise answers.

Given the properties of the data stream computational model, there are multiple factors that need to be considered when designing algorithms. Since the size of the stream is potentially unbounded, methods to store or summarize the data are required. It is possible to store the entire data in a distributed fashion or to summarize only the most important characteristics of the stream. If the entire stream is stored, no data is lost and any subsequent processing is nothing else than another streaming algorithm

with potentially different sources. A solution to the original problem is only delayed. If a summary of the data is built, it should capture the most important features of the stream relevant for the desired computation. There are multiple alternatives that can be considered depending on the actual problem. In the most general case, a uniform sample with a fixed size can be extracted from the unbounded stream. This allows the computation of a large number of functions of the data stream at the expense of a costly overhead. For specific properties that are defined before-hand, better synopsis structures [31] can be used. In the case when the data is distributed over multiple sources—either originally or because they were entirely stored—it is infeasible to re-assemble the stream in a single location for further processing because of the communication overhead. A more efficient solution is to create a synopsis at each site and then transfer only the synopsis. Of course, the necessary condition for this to work is that the single synopsis created by composing the local synopses is a true summary of the entire data. The update rate of the stream is an important factor that also needs to be considered in this computational model. In order for any of the synopsis to be applicable, it should be able to keep-up with the rate of the stream. Unfortunately, this is not always the case in practice where extremely fast streams are available—consider for example the stream of data read from the hard-drives in a datawarehouse environment. In such scenarios, even the fastest synopses are not always good enough and new solutions might be required.

Sketches are randomized synopses that summarize large amounts of data in a much smaller space. While sampling maintains the individual identity of the elements contained in the samples, the identity of an element is entirely lost once sketched. Due to this summarization, specific sketches need to be built for different problems, which is not always the case for samples. At the same time, the maintenance overhead is usually much smaller for sketches, making them more amenable for the data stream computational model. There exist multiple categories of sketches proposed in the literature, each of them

solving a different class of problems. Flajolet-Martin (FM) sketches [26] are used for the estimation of different set cardinalities such as the number of distinct elements, the size of the union, intersection, and difference of two data sets. Alon-Gibbons-Matias-Szegedy (AGMS) sketches [3] were initially proposed for the computation of the second frequency moment of a data set and then extended to the estimation of the dot-product of two vectors. [20] further extended AGMS-sketches to the processing of complex aggregate functions over joins of multiple relations. Sketches based on stable distributions [37] were proposed for the computation of different norms of a data stream. All types of sketches share the same idea of combining randomness with the data in order to reduce the memory space. They differ in the type of randomness they use, the update procedure, and the way they define the estimator.

Aggregate estimation represents a long-standing topic in the database literature. The proposed estimation techniques evolved as the application scenarios changed. Query optimization was the starting point for statistics estimation. In order to select the optimal execution plan for a query, the optimizer utilizes pre-computed statistics. These were initially stored inside the database for each table under the form of number of distinct elements and distributional histograms. Based on these statistics, a cost was computed for plans containing any possible combination of relational operators. There exists a large body of work on histogram computation and maintenance and on the effectiveness of histograms for cardinality estimation (see [22, 31] for a complete list). Since the theoretical characterization of the estimations using histograms was hard to achieve, sampling methods were also investigated in this context [22, 31]. The main advantage sketches have over the other estimation techniques is that they can be maintained up-to-date while the database is updated without significant overhead. At the same time, sketches were characterized theoretically from the beginning. Approximate query answering represents the next level in the treatment of aggregate estimation by the database community (see the AQUA project white paper from Bell Labs). In a datawarehouse environment

containing terabytes of data, queries have un-acceptable running times. In order to decrease the response time, it is acceptable to provide approximate answers with clear error guarantees for specific classes of applications – analytical and exploratory. Thus, random summaries of the datawarehouse are computed and the queries are evaluated on them much faster. If these summaries are pre-computed and stored inside the database, there is no significant difference between query optimization and approximate query processing. The main difference is the visibility: while the query optimizer is not explicitly visible to the user, the approximate results provided by an approximate query processing system immediately affect a user. Another alternative to approximate query answering is online aggregation [35]. In this case the estimates are computed at run-time from the actual data. As more data are seen, the estimates get better and better, to the point where they are the true answer to the query when the entire data are processed. The focus is different in such a system. Instead of providing a single estimate based on pre-computed statistics, a continuous estimate is updated throughout the entire execution of a query. The main requirement for such a system to work is that the data are processed in a random fashion. This is necessary because the estimators are computed from small enough samples that need to fit into memory at each particular instance of time.

The focus of this work is AGMS-sketches for aggregate estimations. More specific, estimating the dot-product of two streams, and the second frequency moment of a stream as a particular case, is the central problem of this study. The goal is to completely understand the strengths and weaknesses of AGMS-sketches both from a theoretical and practical perspective, thus putting them on the map of the current approximation techniques. The ultimate aim is to make AGMS-sketches usable in practice.

1.1 Contributions

The focus of this research work is on sketch synopsis for aggregate queries over data streams. We aim at deeply understanding the theoretical foundations that lie at the basis of the sketching methods and further improve the existing techniques both from a

theoretical and practical perspective. Our final goal is to make sketch synopsis practical for the application in a data stream environment. To this end, a detailed theoretical characterization represents a necessary prerequisite.

Our contributions are fundamental for sketch synopsis, targeting the basic foundations of the method. The directions of our research and the corresponding theoretical and practical findings of our work can be summarized as follows:

- Pseudo-random numbers with specific properties lie at the ground of many randomization algorithms, including sketching techniques. We study the existing random number generation methods and identify the practical algorithms that apply to sketches. Our main finding is that the best accuracy performance is obtained for a 3-wise independent generating scheme, i.e., EH3, which is completely surprising because the general belief was that 4-wise independent random variables are required for sketch synopsis. We provide the theoretical results that support our findings and show that the 4-wise independence requirement is necessary only to simplify the theoretical analysis. From a practical point of view, 3-wise independent random variables require less memory and are faster to generate—they provide a factor of 2 improvement both in space and time compared to 4-wise independent random variables.

- Although using 3-wise independent random numbers represents a considerable improvement in the time performance of the sketch synopsis, this could not be sufficient for the high rate data streams that have to be processed in the current networking equipment. To alleviate this problem, we propose the combination of sampling and sketch synopsis to further increase the processing rate. We introduce the first sketch over samples estimator in the literature and provide the complete theoretical characterization. Our main finding with respect to the new estimator is that almost the same accuracy results can be obtained even when the sketch is computed over a 1% sample of the data rather than the entire data set. This implies a 100 factor improvement in speed or processing time.

- Multiple sketch synopsis using hashing have been proposed in order to further improve the processing time while maintaining the original accuracy requirements. Although equivalent from a theoretical analysis based on frequency moments, these methods had surprisingly different results in practice. This motivated us to realize a more detailed statistical analysis of the sketch estimators based on the probability distribution. Our surprising finding is that the hybrid sketch Fast-AGMS synopsis outperforms almost always the other sketches although the original theoretical results showed that they were equivalent. We provide both the statistical support and significant experimental evidence for these results.

- Sketches were proved to be a viable solution for handling the interval data streams that appear in spatial applications. Dyadic mapping (DMAP), the state-of-the-art solution for sketching intervals, uses a set of transformations to avoid the sketching of each point in the input interval at the cost of a significant degradation in accuracy. The natural solution for sketching intervals is to use efficient algorithms for summing-up the values in the interval, solution called fast range-summation. We realize a detailed study of the random number generating schemes in order to identify the fast range-summable schemes. Our main finding is that while none of the 4-wise independent schemes is fast range-summable, two of the 3-wise independent schemes, i.e., BCH3 and EH3, have efficient fast range-summable algorithms. The gain in accuracy over DMAP when using the EH3 fast range-summable scheme is significant, as much as a factor of 8. We show that fast range-summation cannot be applied in conjunction with hash-based sketches. To alleviate this and the accuracy performance of DMAP, we develop hybrid algorithms with significantly improved results for both solutions.

The contributions and findings are further detailed in the chapters corresponding to each of the general problems introduced in this section. In order to ease the reading, the experimental results are presented separately for each problem.

1.2 Outline

An introduction to sketches and methods to analyze randomization algorithms is given in Preliminaries (Chapter 2). Pseudo-random number generating schemes and the analysis of the original AGMS sketching synopsis are presented in Chapter 3. Chapter 4 contains the details of the combined sketch over samples estimator. The statistical analysis of the sketch synopsis and estimators is provided in Chapter 5. The extension to sketches for interval data streams is detailed in Chapter 6. Chapter 7 contains the conclusions.

CHAPTER 2
PRELIMINARIES

The material in this section serves as a basis for the rest of the work. The material is further detailed accordingly to the demands of each subsequent section. The formal definition of the problem we deal with in this work—estimating the size of join over data streams using sketches—is provided in Section 2.1. Then we take a close look at the existing methods to derive confidence bounds for randomized algorithms—sketches are ultimately a randomized algorithm—in Section 2.3. The basic AGMS sketching method we considerably refine and improve throughout the work is introduced in Section 2.2.

2.1 Problem Formulation

Let $S = (e_1, w_1), (e_2, w_2), \ldots, (e_s, w_s)$ be a data stream, where the keys e_i are members of the set $I = \{0, 1, \ldots, N-1\}$ and w_i represent frequencies. The *frequency vector* $\bar{f} = [f_0, f_1, \ldots, f_{N-1}]$ over the stream S consists of the elements f_i defined as $f_i = \sum_{j:e_j=i} w_j$. The key idea behind the existing sketching techniques is to represent the domain-size frequency vector as a much smaller *sketch* vector $\bar{x}_f$ [14] that can be easily maintained as the updates are streaming by and that can provide good approximations for a wide spectrum of queries.

Our focus is on sketching techniques that approximate the *size of join* of two data streams. The size of join is defined as the *inner-product* of the frequency vectors $\bar{f}$ and $\bar{g}$, $\bar{f} \odot \bar{g} = \sum_{i=0}^{N-1} f_i g_i$. As shown in [51], this operator is generic since other classes of queries can be reduced to the size of join computation. For example, a range query over the interval $[\alpha, \beta]$, i.e., $\sum_{i=\alpha}^{\beta} f_i$, can be expressed as the size of join between the data stream S and a virtual stream consisting of a tuple $(i, 1)$ for each $\alpha \leq i \leq \beta$. Notice that point queries are range queries over size zero intervals, i.e., $\alpha = \beta$. Also, the second frequency moment or the self-join size of S is nothing else than the inner-product $\bar{f} \odot \bar{f}$.

2.2 Sketches

AGMS sketches are randomized schemes that were initially introduced for approximating the second frequency moment of a relation in small space [3]. Afterwards, they were applied to the general size of join problem [4].

The size of join of two relations F and G, each with a single attribute A, consists in computing the quantity $|F \bowtie_A G| = \sum_{i \in I} f_i g_i$, where I is the domain of A and f_i and g_i, respectively, are the frequencies of the tuples in the two relations. The exact solution to this problem is to maintain the frequency vectors $\bar{f}$ and $\bar{g}$ and then to compute the size of join. Such a solution would not work if the amount of memory or the communication bandwidth is smaller than $Min(|I|, \textit{cardinality of relation})$.

The approximate solution based on sketches is defined as follows [4]:

1. Start with a family of 4-wise independent ± 1 random variables ξ corresponding to the elements in the domain I.

2. Define the sketches $X_F = \sum_{i \in I} f_i \xi_i = \sum_{t \in R} \xi_{t.A}$ and $X_G = \sum_{i \in I} g_i \xi_i = \sum_{t \in S} \xi_{t.A}$. The sketch summarizes the frequency vector of the relation as a single value by randomly projecting each tuple over the values $+1$ and -1. Notice that the tuples with the same value are always projected either on $+1$ or -1.

3. Let the random variable X be $X = X_F X_G$. X has the properties that it is an unbiased estimator for the size of join $|F \bowtie_A G|$ and that it has small variance. An estimator with relative error at most ϵ with probability at least $1 - \delta$ can be obtained by taking the median over averages of multiple independent copies of the random variable X: the median is computed over $s_2 = 2 \log \frac{1}{\delta}$ such averages, each average containing $s_1 = \frac{8}{\epsilon^2} \frac{\text{Var}[X]}{E[X]^2}$ instances of X (Figure 2-1).

Sketches are perfectly suited for both data-streaming and distributed computation since they can be updated on pieces. For example, if the tuples in one relation are streamed one by one, the atomic sketch can be computed by simply adding the value ξ_i corresponding to the value i of the current item. For distributed computation, each party can compute the sketch of the data it owns. Then, by only exchanging the sketch with the other parties and adding them up, the sketch of the entire dataset can be computed.

The AGMS sketches described above have at their core ± 1 random variables. The values $+1$ and -1 can be obtained by simply generating random bits and then interpreting them as -1 and $+1$ values. The necessary and sufficient requirement for X to be an unbiased estimator for the size of join of F and G is that the ξ family of random variables to be 2-wise independent. The stronger 4-wise independence property is required in order to make the variance as small as possible, thus reducing the number of copies of X that need to be averaged in order to achieve the desired accuracy. But, as we show in Section 3.1, generating 4-wise independent random variables is more demanding both in the amount of required memory, as well as in time efficiency.

2.3 Confidence Bounds

The abstract problem we tackle throughout this work is the following. Given $X_1, \ldots, X_n$ independent instances of a generic random variable X, define an estimator for the expected value $E[X]$ and provide confidence bounds for the estimate. While $E[X]$ is the convergence value of the estimator (hopefully the true value of the estimated quantity), confidence bounds provide information about the interval where the expected value lies with high probability or, equivalently, the probability that a particular instance of X deviates by a given amount from the expectation $E[X]$.

In this section we provide an overview of the methods to derive confidence bounds for generic random variables in general, and sketches, in particular. There exist two types of confidence bounds: distribution-independent and distribution-dependent. Distribution-independent confidence bounds are derived from general notions in measure theory and are mainly used in theoretical computer science. Distribution-dependent confidence bounds assume certain distributions for the estimator of the expectation $E[X]$ and are largely used in statistics. While distribution-independent bounds are based on general inequalities, a detailed problem-specific analysis is required for distribution-dependent bounds. In the context of sketch estimators we show that

distribution-independent bounds, although easier to obtain, are unacceptably loose in some situations, thus making it necessary to derive tighter distribution-dependent bounds.

2.3.1 Distribution-Independent Confidence Bounds

As already specified, distribution-independent confidence bounds are derived from general inequalities on tail probabilities in measure theory. No assumption on the probability distribution of the estimator is made. We introduce the inequalities used for characterizing sketching techniques in what follows.

Theorem 1 (Markov Inequality [46]). *Let X be any random variable with expected value $E[X]$ and f any positive real function. Then for all $t \in \mathbb{R}^+$:*

$$P[f(X) \geq t] \leq \frac{E[f(X)]}{t} \tag{2–1}$$

Or, equivalently,

$$P[f(X) \geq tE[f(X)]] \leq \frac{1}{t} \tag{2–2}$$

Markov inequality states that the probability that a random variable deviates by a factor larger than t from its expected value is smaller than $\frac{1}{t}$. This is the tightest bound that can be obtained when only the expectation $E[X]$ is known. Tighter confidence bounds can be derived using Chebyshev inequality and its extension to higher moments. These bounds require more information on the distribution of the estimator for $E[X]$ which is not always available.

Theorem 2 (Chebyshev Inequality [46]). *Let X be a random variable with expectation $E[X]$ and variance $Var[X]$. Then, for any real function f and any $t \in \mathbb{R}^+$:*

$$P\left[|f(X) - E[f(X)]| \geq t\sqrt{Var[f(X)]}\right] \leq \frac{1}{t^2} \tag{2–3}$$

Chernoff bounds are exponential tail bounds applicable to sums of independent Poisson trials. They are largely used for the analysis and design of randomized algorithms (including sketches) because of the tight bounds (logarithmic) they provide.

Theorem 3 (Chernoff Bound [46]). *Let $X_1, \ldots, X_n$ be independent Poisson trials (random variables taking only the values 0 or 1) such that, for $1 \leq i \leq n$, $P[X_i = 1] = p_i$, where $0 < p_i < 1$. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = E[X] = \sum_{i=1}^{n} p_i$, and any $\delta > 0$,*

$$P[X > (1+\delta)\mu] < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu} \tag{2–4}$$

Corollary 1 (Chernoff Upper Tail [46]). *For the same setup as in Theorem 3, it holds:*

$$P[X > (1+\delta)\mu] < e^{\frac{-\mu\delta^2}{3}} \tag{2–5}$$

Corollary 2 (Chernoff Lower Tail [46]). *For the same setup as in Theorem 3, it holds:*

$$P[X < (1-\delta)\mu] < \left[\frac{e^{\delta}}{(1-\delta)^{(1-\delta)}}\right]^{\mu} < e^{\frac{-\mu\delta^2}{2}} \tag{2–6}$$

Notice that the above inequalities require the computation of a certain number of frequency moments of the random variable X. The larger the number of moments computed, the tighter the confidence bounds. Unfortunately, the computation of higher moments demands larger degrees of independence between the instances of X and it cannot always be carried out exactly. Moreover, the improvement gained by computing higher moments is usually a constant factor which is asymptotically insignificant. Thus, distribution-independent confidence bounds are mostly expressed in terms of the first two frequency moments (expectation and variance) using Markov and Chebyshev inequalities.

2.3.2 Distribution-Dependent Confidence Bounds

In order to compute distribution-dependent confidence bounds, a parametric distribution is assumed for the estimator of $E[X]$. Then confidence bounds are derived from the cumulative distribution function (cdf) of the assumed distribution. The parameters of the considered distribution are generally computed from the frequency moments of X, i.e., a number of moments equal with the number of parameters have to be computed. Since a large number of distributions have only two parameters, e.g., Normal, Gamma, Beta, etc., only the expectation and the variance of X have to be determined.

Notice that although both types of confidence bounds require the computation of the frequency moments of X, the actual bounds are extracted in different ways. The question that immediately arises is which confidence bounds should be used. Typically, distribution-dependent bounds are tighter, but there exist assumptions that need to be satisfied in order for them to hold. Specifically, the distribution of the estimator for $E[X]$ has to be similar in shape with the assumed parametric distribution. In the following we provide a short overview of the results from statistics on distribution-dependent confidence bounds that are used throughout this work.

2.3.3 Mean Estimator

Usually the *mean* $\bar{X}$ of $X_1, \ldots, X_n$ is considered as the proper estimator for $E[X]$. It is known from statistics [54] that when the distribution of X is normal, the mean $\bar{X}$ is the *uniformly minimum variance unbiased estimator* (UMVUE), the *minimum risk invariant estimator* (MRIE), and the *maximum likelihood estimator* (MLE) for $E[X]$. This is strong evidence that $\bar{X}$ should be used as the estimator of $E[X]$ when the distribution of X is normal or almost normal. *Central Limit Theorem* (CLT) extends the characterization of the mean estimator to arbitrary distributions of X.

Theorem 4 (Mean CLT [54])**.** *Let $X_1, \ldots, X_n$ be independent samples drawn from the distribution of the random variable X and $\bar{X}$ be the average of the n samples. Then, as long as $Var[X] < \infty$:*

$$\bar{X} \rightarrow_d N\left(E[X], \frac{Var[X]}{n}\right) \tag{2–7}$$

Essentially, CLT states that the distribution of the mean is asymptotically a normal distribution centered on the expected value and having variance $\frac{\mathrm{Var}[X]}{n}$ irrespective of the distribution of X. Confidence bounds for $\bar{X}$ can be immediately derived from the cdf of the normal distribution:

Theorem 5 (Mean Bounds [53])**.** *For the same setup as in Theorem 4, the asymptotic confidence bounds for* $\bar{X}$ *are:*

$$P\left[|\bar{X} - E[X]| > z_{\alpha/2}\sqrt{\frac{Var[X]}{n}}\right] < \alpha \tag{2–8}$$

where z_β *is the* β *quantile (*$\beta \in [0,1]$*) of the normal* $N(0,1)$ *distribution, i.e., the point for which the probability of* $N(0,1)$ *to be smaller than the point is* β*.*

Since fast series algorithms for the computation of z_β are widely available[1] , the computation of confidence bounds for $\bar{X}$ is straightforward. Usually, the CLT approximation of the distribution of the mean and the confidence bounds produced with it are correct starting with hundreds of samples being averaged. If the number of samples is smaller, confidence bounds can be determined based on the Student t-distribution [53]. The only difference is that the β quantile $t_{n-1,\beta}$ of the Student t-distribution with $n-1$ degrees of freedom has to be used instead of the β quantile z_β of the normal distribution in Theorem 5.

Notice that in order to characterize the mean estimator, only the variance of X has to be determined. When $\mathrm{Var}[X]$ is not known – this is the case for sketches since estimating the variance is at least as hard as estimating the expected value – the variance can be estimated from the samples in the form of sample variance. This is the common practice in statistics and also in database literature (approximate query processing with sampling).

2.3.4 Median Estimator

Although the mean is the preferable estimator in most circumstances, there exist distributions for which the mean cannot be used as an estimator of $E[X]$. For Cauchy distributions (which have infinite variance) the mean can be shown to have the same distribution as a single random sample. In such cases the *median* $\tilde{X}$ of the samples is

[1] The GNU Scientific Library (GSL) implements pdf, cdf, inverse cdf, and other functions for the most popular distributions, including Normal and Student t.

the only viable estimator of the expected value. The necessary condition for the median to be an estimator of the expected value is that the distribution of the estimator to be symmetric, case in which the mean and the median coincide. We start the investigation of the median estimator by introducing its corresponding central limit theorem and then show how to derive confidence bounds.

Theorem 6 (Median CLT [54]). *Let $X_1, \ldots, X_n$ be independent samples drawn from the distribution of the random variable X and $\tilde{X}$ be the median of the n samples. Then, as long as the probability density function f of the distribution of X has the property $f(\theta) > 0$:*

$$\tilde{X} \rightarrow_d N\left(\theta, \frac{1}{4n \cdot f(\theta)^2}\right) \tag{2–9}$$

where θ is the true median of the distribution.

Median CLT states that the distribution of the median is asymptotically a normal distribution centered on the true median and having the variance equal to $\frac{1}{4n \cdot f(\theta)^2}$. In order to compute the variance of this normal distribution and derive confidence bounds from it, the probability density function (pdf) of X has to be determined or at least estimated at the true median θ. Since $f(\theta)$ cannot be computed exactly in general, multiple estimators for the variance are proposed in the statistics literature [50]. We use the variance estimator proposed in [48] for deriving confidence bounds:

Theorem 7 (Median Bounds [48]). *For the same setup as in Theorem 6, the confidence bounds for $\tilde{X}$ are given by:*

$$P\left[|\tilde{X} - \theta| \leq t_{n-1,1-\alpha/2} \mathtt{SE}(\tilde{X})\right] \geq 1 - \alpha \tag{2–10}$$

where $t_{n-1,\beta}$ is the β quantile of the Student t-distribution with $n-1$ degrees of freedom and $\mathtt{SE}(\tilde{X})$ is the estimate for the standard deviation of $\tilde{X}$ given by:

$$\mathtt{SE}(\tilde{X}) = \frac{X_{(U_n)} - X_{(L_n+1)}}{2}, \text{ where } L_n = \left[\frac{n}{2}\right] - \left[\sqrt{\frac{n}{4}}\right], U_n = n - L_n \tag{2–11}$$

Notice that while the distribution corresponding to the mean estimator is centered on the expected value $E[X]$, the distribution of the median is centered on the true median, thus the requirement on the symmetry of the distribution for the median to be an estimator of $E[X]$.

2.3.5 Mean vs Median

For the cases when the distribution is symmetric, thus the expected value and the median coincide, or when the difference between the median and the expected value is insignificant, the decision with respect to which of the mean or the median to be used as an estimate for the expected value is reduced to establishing which of the two has smaller variance. Since for both estimators the variance decreases by a factor of n, the question is further reduced to comparing the variance $\mathrm{Var}[X]$ and the quantity $\frac{1}{4f(\theta)^2}$. The relation between these two quantities is established in statistics through the notion of *asymptotic relative efficiency*:

Definition 1 ([54]). *The relative efficiency of the median estimator $\tilde{X}$ with respect to the mean estimator $\bar{X}$, shortly the* efficiency *of the distribution of X with the probability density function f, is defined as:*

$$e(f) = 4f(\theta)^2 \, Var[X] \tag{2–12}$$

The efficiency of a distribution for which $E[X] = \theta$ indicates which of the mean or the median is a better estimator for $E[X]$. More precisely, $e(f)$ indicates the reduction in mean squared error if the median is used instead of the mean. When $e(f) > 1$, median is a better estimator, while for $e(f) < 1$ the mean provides better estimates.

An important case to consider is when X has normal distribution. In this situation the efficiency is independent of the parameters of the distribution and it is equal to $\frac{2}{\pi} \approx 0.64$ (derived from the above definition and the pdf of the normal distribution). This immediately implies that when the estimator is defined as the average of the samples, i.e., by Mean CLT the distribution of the estimator is asymptotically normal, the mean

estimator is more efficient than the median estimator. We exploit this result for analyzing sketches in Section 5.2. In terms of mean squared error, the mean estimator has error 0.64 times smaller, while in terms of root mean squared error or relative error, the mean estimator has error 0.8 times smaller.

As already pointed out, when the efficiency is supra-unitary, i.e., $e(f) > 1$, medians should be preferred to means for estimating the expected value, if the distribution is symmetric or almost symmetric. An interesting question is what property of the distribution – hopefully involving only moments since they are easier to compute for discrete distributions – indicates supra-unitary efficiency. According to the statistics literature [7], *kurtosis* is the perfect indicator of supra-unitary efficiency.

Definition 2 ([7]). *The kurtosis k of the distribution of the random variable X is defined as:*

$$k = \frac{E\left[(X - E\left[X\right])^4\right]}{Var\left[X\right]^2} \tag{2–13}$$

For normal distributions, the kurtosis is equal to 3 irrespective of the parameters. Even though there does not exist a distribution-independent relationship between the kurtosis and the efficiency, empirical studies [49] show that whenever $k \leq 6$ the mean is a better estimator of $E\left[X\right]$, while for $k > 6$ the median is the better estimator.

2.3.6 Median of Means Estimator

Instead of using only the mean or the median as an estimator for the expected value, we can also consider combined estimators. One possible combination that is used in conjunction with sketching techniques (see Section 4.2) is to group the samples into groups of equal size, compute the mean of each group, and then the median of the means, thus obtaining the overall estimator for the expected value. To characterize this estimator using distribution-independent bounds, a combination of the Chebyshev and Chernoff bounds (Theorem 2 and Theorem 3) can be used:

Theorem 8 ([3]). *The median Y of $2\ln(\frac{1}{\alpha})$ means, each averaging $\frac{16}{\epsilon^2}$ independent samples of the random variable X, has the property:*

$$P\left[|Y - E[X]| \leq \epsilon\sqrt{Var[X]}\right] \geq 1 - \alpha \tag{2–14}$$

We provide an example that compares the bounds for the median of means estimator. While distribution-independent bounds are computed using the results in Theorem 8, distribution-dependent bounds are computed through a combination of Mean CLT, Median CLT, and efficiency.

Example 1. *Suppose that we want to compute 95% confidence bounds for the median of means estimator. Then the number of means for which we compute the median should be $2\ln\frac{1}{0.05} = 2\ln 20 \approx 9$ according to Theorem 8. If the number of samples is n, then each mean is the average of $\frac{n}{9}$ samples, thus $\epsilon = \sqrt{\frac{144}{n}} = 12 \cdot \sqrt{\frac{1}{n}}$. The width of the confidence bound in terms of $\sqrt{\frac{Var[X]}{n}}$ is thus 12.*

By applying Mean CLT, the distribution of each mean is asymptotically normal with variance $\frac{Var[X]}{n/9}$. In practice, confidence bounds can be easily derived by applying the results in Theorem 7. We cannot do that in this example because the values of the 9 means are unknown. Instead we assume that the distribution of the means is asymptotically normal and, by Median CLT and the definition of efficiency, the median of the 9 means has variance $\frac{1}{9e(N)} \cdot \frac{Var[X]}{n/9}$, with $e(N) = \frac{2}{\pi}$ the efficiency of the normal distribution. The variance of Y is thus $\frac{\pi}{2} \cdot \frac{Var[X]}{n} \approx 1.57 \cdot \frac{Var[X]}{n}$. With this, the width of the CLT-based confidence bound for Y with respect to $\sqrt{\frac{Var[X]}{n}}$ is $\sqrt{1.57} \cdot 1.96 = 2.45$ (1.96 is the 95% quantile), which is $\frac{12}{2.45} \approx 4.89$ times smaller than the distribution-independent confidence bound.

This result confirms that distribution-dependent confidence bounds are tighter and is consistent with other results that compare the two types of bounds [53]. Confidence bounds of different widths can be computed in a similar manner, the only difference being the

values that are plugged into the formulas. For example, the ratio for the 99% *confidence bound is* 4.64 *and* 4.34 *for the* 99.9% *confidence interval.*

An important point in the above derivation of the CLT confidence bounds for Y is the fact that the confidence interval is wider by $\sqrt{\frac{\pi}{2}} \approx 1.25$ if medians are used, compared to the situation when the estimator is only the mean (with no medians). This implies that the median of means estimator is always inferior to the mean estimator irrespective of the distribution of X. A simple explanation for this is that the asymptotic regime of Mean CLT starts to take effect (the distribution becomes normal) since means are computed first and the mean estimator is more efficient than the median estimator. Thus, from a practical and statistical point of view based on the efficiency of the distribution, the estimator should be either the mean ($e < 1$), or the median ($e > 1$), but never the median of means.

2.3.7 Minimum Estimator

Although the minimum of the samples $X_1, \ldots, X_n$ is not an estimator for the expectation $E[X]$, a discussion on the behavior of the minimum estimator is included because of its relation to Count-Min sketches. It is known from statistics [12] that the minimum of a set of samples has an asymptotic distribution called the *generalized extreme value distribution* (GEV) independent of the distribution of X. The parameters of the GEV distribution can be computed from the frequency moments of X, thus confidence bounds for the minimum estimator can be derived from the cdf of GEV. Although this is a straightforward method to characterize the behavior of the minimum estimator, it is not applicable to Count-Min sketches (Section 5.2).

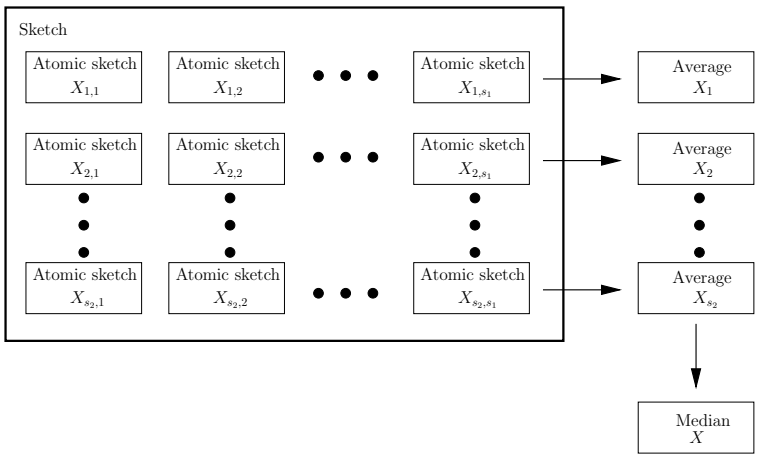


Figure 2-1. AGMS Sketches.

CHAPTER 3
PSEUDO-RANDOM SCHEMES FOR SKETCHES

The exact computation of aggregate queries, like the size of join of two relations, usually requires large amounts of memory, constrained in data-streaming, or communication, constrained in distributed computation, and large processing times. In this situation, approximation techniques with provable guarantees, like sketches, are one possible solution. Due to their linearity, AGMS sketches [3] have all these properties and they have been successfully used for the estimation of aggregates, like the size of join, over data-streams [4, 20] and in distributed environments, like sensor networks [41]. The performance of sketches depends crucially on the ability to generate particular pseudo-random numbers. In this chapter we investigate both theoretically and empirically the problem of generating k-wise independent pseudo-random numbers and, in particular, that of generating 3 and 4-wise independent pseudo-random numbers. Our specific contributions are:

- We provide a thorough comparison of the various generating schemes with the goal of identifying the efficient ones. To this end, we explain how the schemes can be implemented on modern processors and we use such implementations to empirically evaluate them.
- We provide thorough theoretical analysis for the behavior of the BCH3 and EH3 schemes as replacements for the 4-wise independent schemes in size of join estimations using AGMS sketches. We show that while BCH3 is a poor replacement for non-skewed distributions, EH3 is in general as good as or significantly better than any 4-wise independent scheme. The fact that EH3 gets within a constant factor of the error of the 4-wise independent schemes for the problem of computing the L^1-difference of two streaming vectors was theoretically proved in [25]. Here we generalize this result to size of join estimations using AGMS sketches and we show that EH3 can always replace the 4-wise independent schemes without sacrificing accuracy. Our empirical evaluation confirms these theoretical predictions.

In the rest of the chapter, we discuss the known generating schemes for random variables with limited independence in Section 3.1. In Section 3.2 we provide theoretical proof that the Extended Hamming (EH3) generating scheme works as well as the 4-wise

independent schemes. We provide empirical evidence for this fact and a thorough comparison between EH3 and the 4-wise schemes in Section 3.2.4.

3.1 Generating Schemes

Based on the published literature [3], the AGMS sketches described in Section 2.2 need ± 1 4-wise independent random variables that can be generated in small space in order to produce reliable estimations. We address the question of whether the 4-wise independence is actually required later in the chapter, but the 2-wise independence is a minimum requirement, since otherwise the estimator is not even unbiased. In this section we review a number of generating schemes for random variables with limited degree of independence and we discuss how they can be implemented on modern processors. In the subsequent sections we refer back to these generating schemes.

3.1.1 Problem Definition

Let I be a domain—without loss of generality we assume that $I = \{0, 1, \ldots, N-1\}$—and ξ be a family of two-valued random variables with a member for each $i \in I$, denoted by ξ_i. We assume that these random variables take the values ± 1 with the same probability. For different values, a mapping has to be defined. Usually, the domain I is large, for example $N = 2^{32}$, so a large amount of space is required to store even one instance of the family ξ. In order to be space efficient, each ξ_i is generated on demand according to a generating function:

$$\xi_i(S) = (-1)^{f(S,i)}, \quad i \in I \tag{3–1}$$

S is a random seed that can be stored in small space and the function f can be efficiently computed from the index i and S. The space used by S and the function f depend upon the desired degree of independence k. To allow simple generation, the seed S is uniformly and randomly chosen from its domain. With S generated this way, the k-wise independence property required for ξ translates into requiring that function f, for each k different indices i_1, i_2,...,i_k, generates any of the 2^k possible outcomes the same number

of times, as S covers its domain. [9] give a formal definition of uniform k-wise independent random variables, also known as k-universal hashing functions:

Definition 3. *A family ξ of ± 1 random variables defined over the sample space I is* ***k-wise uniform independent*** *if for any k different instances of the family $\xi_{i_1}, \xi_{i_2}, \ldots, \xi_{i_k}$, and any k ± 1 values $v_1, v_2, \ldots, v_k$, it holds:*

$$Pr[\xi_{i_1} = v_1 \wedge \xi_{i_2} = v_2 \wedge \cdots \wedge \xi_{i_k} = v_k] = \frac{1}{2^k} \tag{3–2}$$

3.1.2 Orthogonal Arrays

The problem of generating two-valued k-wise independent random variables is equivalent to the construction of *orthogonal arrays*. Precisely, ± 1 k-wise independent random variables over the domain $I = \{0, 1, \ldots, N-1\}$ form an orthogonal array with $|S|$ runs, N variables, two levels and strength k, denoted OA$(|S|, N, 2, k)$, where $|S|$ is the size of the seed space. Since there exist clear relations defining the size of an orthogonal array, first we overview their main properties. Then, we extend these relations for characterizing the size of the seed space. Our presentation follows the approach in [36].

Definition 4. *An $r \times c$ matrix A with entries from a set M is called an* ***orthogonal array*** *$OA(r, c, m, t)$ with m levels, strength t and index λ – for some t in the range $0 \leq t \leq c$ – if every $r \times t$ sub-matrix of A contains each t-tuple based on M exactly λ times as a row.*

M is the set containing the elements $\{0, 1, \ldots, m-1\}$, where m is the number of symbols or the number of levels. The number of rows r is known as the size of the array or the number of runs. c, the number of columns, represents the number of constraints, factors, or variables. A less formal way of defining an orthogonal array is to say that it is an array with the property that in any t columns you see equally often all possible t-tuples. When each tuple appears exactly once, i.e., $\lambda = 1$, we say that the array has index unity.

Example 2. *Table 3-1 represents an orthogonal array $OA(8, 4, 2, 3)$ that has index unity.*

Given the values of the parameters, the size of an orthogonal array is expressed in *Rao's Inequalities*. A particular form of these inequalities, for $m = 2$, was introduced in the context of k-wise independent random variables by [2]. We present the inequalities here and later we show how they are used for characterizing different construction schemes.

Theorem 9 ([36]). *The parameters of an orthogonal array $OA(r, c, s, t)$ satisfy the following inequalities:*

$$\begin{aligned} r &\geq \sum_{i=0}^{u} \binom{c}{i} (m-1)^i, \text{ if } t = 2u, \\ r &\geq \sum_{i=0}^{u} \binom{c}{i} (m-1)^i + \binom{c-1}{u} (m-1)^{u+1}, \text{ if } t = 2u+1, \end{aligned} \tag{3–3}$$

for $u \geq 0$, $r \equiv 0 \mod s^t$.

Since the problem of generating two-valued random variables that are k-wise independent is identical to constructing an orthogonal array $\mathrm{OA}(|S|, N, 2, k)$. In our particular case, we are interested in finding optimal constructions, that is, constructions that minimize the number of runs given fixed values for the number of variables, i.e., $N = 2^n$, and the number of levels, since such constructions reduce the size of the seed, thus the space to generate and store it. Also, we need general constructions that exist for any value of the strength, i.e., the degree of independence.

The constructions based on BCH and Reed-Muller error-correcting codes are widely used. As stated in [36], the BCH arrays are the densest known for $N = 2^n$ variables, when n is odd. When n is even, there exist arrays with fewer runs based on Kerdock codes and, respectively, Delsarte-Goethals codes. However, these constructions exist only for even values of n and their advantage over the corresponding BCH arrays is minimal—one or two bits less for representing the number of runs—while overhead is incurred for computations in $\mathbb{Z}_4$, the ring of integers modulo 4. Both BCH and

Reed-Muller constructions touch the bounds given by Rao's Inequalities only for small values of k, e.g., 2, 3.

3.1.3 Abstract Algebra

As mentioned previously, ± 1 random variables with limited independence can be obtained by generating $\{0, 1\}$ random variables and mapping them to ± 1. Since 0 and 1 are the only elements of the Galois Field with order 2, denoted by $GF(2)$, abstract algebra [19] is the ideal framework in which to talk about generating families of random variables with limited independence. The field $GF(2)$ has two operations: addition (boolean XOR) and multiplication (boolean AND). Abstract algebra provides two ways to extend the base field $GF(2)$—vector spaces and extension fields—both of them useful for generating limited independence random variables.

$GF(2)^k$ *Vector Spaces* are spaces obtained by bundling together k dimensions, each with a $GF(2)$ domain. The only operation we are interested in here is the dot-product between two vectors $\bar{v}$ and $\bar{u}$, defined as: $\bar{v} \cdot \bar{u} = \oplus_{j=0}^{k-1} v_j \odot u_j$. For $GF(2)^k$ vector spaces this corresponds to AND-ing the arguments and then XOR-ing all the bits in the result.

$GF(p)$ *Prime Fields* are fields over the domain $\{0, 1, \ldots, p-1\}$ with both the multiplication and the addition defined as the arithmetic multiplication and addition modulo the prime p.

$GF(2^k)$ *Extension Fields* are fields defined over the domain $\{0, 1, \ldots, 2^k - 1\}$ that have two operations: addition, $+$, with zero element 0, and multiplication, $\cdot$, with unity element 1. Both addition and multiplication have to be associative and commutative. Also, multiplication is distributive over addition. All the elements, except 0, must have an inverse with respect to the multiplication operation. The usual representation of the extension fields $GF(2^k)$ is as polynomials of degree $k-1$ with the most significant bit as the coefficient for X^{k-1}, and the least significant as the constant term. The addition of two elements is simply the addition, term by term, of the corresponding polynomials. The multiplication is the polynomial multiplication modulo an irreducible polynomial of degree

k that defines the extension field. With this representation, the addition is simple (just XOR the bit representations), but the multiplication is more intricate since it requires both polynomial multiplication and division.

As we will see, a large number of schemes use dot-products in vector spaces. Dot-products can be implemented by simply AND-ing the two vectors and XOR-ing the resulting bits. While AND-ing entire words (integers) on modern architectures is extremely fast, XOR-ing the bits of a word (which has to be eventually performed) is problematic since no high-level programming language supports such an operation (this operation is actually the parity-bit computation). To speed-up this operation, which is critical, we implemented it in *Assembly* for Pentium processors to take advantage of the supported 8-bit parity computation.

3.1.4 Bose-Chaudhuri-Hocquenghem Scheme (BCH)

For all the schemes, we assume the domain to be $I = \{0, \ldots, 2^n - 1\}$, for a generic n (the description of the schemes depends on n). We also make the convention that $[a, b]$ is equivalent with the vector obtained by concatenating the vectors a and b. The size of a, b, and $[a, b]$ is clear from the context.

The BCH scheme was first introduced in [2] and it is based on BCH error-correcting codes. This scheme can generate $(2k + 1)$-wise independent random variables by using uniformly random seeds S that are $kn + 1$ bits in size. S can be represented as a vector $S = [s_0, S_0, \ldots, S_{k-1}]$, where s_0 is one bit and each S_i, $0 \leq i < k$, is a vector of size n. If s_0 is dropped from the seed, $2k$-wise independent random variables are obtained. The generating function $f(S, i)$ is defined as:

$$f(S, i) = S \cdot [1, i, \ldots, i^{2k-1}] \tag{3–4}$$

where i^{2k-1} is computed in the extension field $GF(2^n)$. This scheme comes close to the theoretical bounds [36] on how dense the seed space can be – it is the scheme with the

densest seed requirement amongst all the known schemes. The proof that this scheme produces $(2k+1)$-wise independent families can be found in [2].

Implementing i^{2k-1} over finite fields is problematic on modern processors if speed is paramount, but in the special case when only 3-wise independence is required this problem is avoided (see the speed comparison at the end of the section). The 3-wise independent version of this scheme is:

$$f(S,i) = S \cdot [1,i] \tag{3–5}$$

Example 3. *We give an example that shows how the BCH schemes work. Consider that* $n = 16$ *and the seeds have the values:* $s_0 = 1$, $S_0 = 7,469 = (0001110100101101)_2$, *and* $S_1 = 346 = (0000000101011010)_2$. *We generate the* 3 *and* 5*-wise random variables corresponding to* $i = 2,500 = (0000100111000100)_2$. *Then,* $i^3 = 15,625,000,000 = (1001010001000000)_2$, *where the exponentiation is computed arithmetically, rather than over the extension field* $GF(2^{16})$, *and only the least significant* 16 *bits are kept.*

$$S_0 \cdot i = \bigoplus_{j=0}^{15} \begin{pmatrix} 0001110100101101 \\ 0000100111000100 \end{pmatrix} = \bigoplus_{j=0}^{15} 0000100100000100 = 1 \tag{3–6}$$

$$S_1 \cdot i^3 = \bigoplus_{j=0}^{15} \begin{pmatrix} 0000000101011010 \\ 1001010001000000 \end{pmatrix} = \bigoplus_{j=0}^{15} 0000000001000000 = 1 \tag{3–7}$$

Computing $S_0 \cdot i$, *first the AND of the two binary values is calculated. Second, the sequential XOR of the resulting bits gives the final result. The same is equivalent for* $S_1 \cdot i^3$. *When implementing these schemes, the sequential XOR can be computed only at the end, after all the intermediate results are XOR-ed. The result for the* 3*-wise case is:*

$$s_0 \oplus S_0 \cdot i = 1 \oplus 1 = 0 \tag{3–8}$$

while the result for the 5*-wise case is:*

$$s_0 \oplus S_0 \cdot i \oplus S_1 \cdot i^3 = 1 \oplus 1 \oplus 1 = 1 \tag{3–9}$$

3.1.5 Extended Hamming 3-Wise Scheme (EH3)

The Extended Hamming 3-wise scheme is a modification of BCH3 and it was introduced in [25]. It requires seeds S of size $n+1$ and its generating function is defined as:

$$f(S,i) = S \cdot [1,i] \oplus h(i) \tag{3–10}$$

where $h(i)$ is a nonlinear function of the bits of i. A possible form for h is:

$$h(i) = i_0 \vee i_1 \oplus \cdots \oplus i_{n-2} \vee i_{n-1} \tag{3–11}$$

Function h does not change the amount of independence, thus, from the traditional AGMS sketches theory, it is not as good as a 4-wise independent scheme. As we will show in Section 3.2, the use of EH3 actually results in size of join AGMS sketches estimations as precise as a 4-wise independent scheme gives[1] .

From the point of view of a fast implementation, only a small modification has to be added to the implementation of BCH3 – the computation of function $h(i)$. As the experimental results show, there is virtually no running time difference between these schemes if a careful implementation is deployed on modern processors.

3.1.6 Reed-Muller Scheme

The BCH schemes require computations over extension fields for degrees of independence greater than 3. Since the AGMS sketches need 4-wise independent random variables, alternative schemes that require only simple computations might be desirable. The Reed-Muller scheme [36] generalizes the BCH codes in a different way in order to obtain higher degrees of independence. Seeds of size $1 + \binom{n}{1} + \cdots + \binom{n}{t}$ are required to obtain a degree of independence of $k = 2^{t+1} - 1$, $t > 0$. We introduce here only the 7-wise

[1] The first theoretical result on the benefits of EH3 is provided in [25]. Our results are significantly more general.

independent version of the scheme that requires $1 + n + \frac{n(n-1)}{2}$ seed bits:

$$f(S, i) = S \cdot [1, i, i^{(2)}] \tag{3–12}$$

where

$$i^{(2)} = [i_0 \odot i_1, i_0 \odot i_2, \ldots, i_{n-2} \odot i_{n-1}] \tag{3–13}$$

Example 4. *We exemplify the Reed-Muller scheme (RM7), i.e., $t = 2$, for $n = 4$. $s_0 = 0$, $S_0 = 11 = (1011)_2$, and $S_1 = 45 = (101101)_2$. S_1 contains $\binom{4}{2} = 6$ bits. The index variable is $i = 6 = (0110)_2$. $i^{(2)} = [0 \odot 1, 0 \odot 1, 0 \odot 0, 1 \odot 1, 1 \odot 0, 1 \odot 0] = (000100)_2$.*

$$S_0 \cdot i = \bigoplus_{j=0}^{3} \begin{pmatrix} 1011 & \odot \\ 0110 & \end{pmatrix} = \bigoplus_{j=0}^{3} 0010 = 1 \tag{3–14}$$

$$S_1 \cdot i^{(2)} = \bigoplus_{j=0}^{5} \begin{pmatrix} 101101 & \odot \\ 000100 & \end{pmatrix} = \bigoplus_{j=0}^{5} 000100 = 1 \tag{3–15}$$

The final result is:

$$s_0 \oplus S_0 \cdot i \oplus S_1 \cdot i^{(2)} = 0 \oplus 1 \oplus 1 = 0 \tag{3–16}$$

An important fact about the Reed-Muller scheme is that the seeds required are significantly large. For example, for a domain of size 2^{32}, RM7 needs seeds of $1 + 32 + \frac{32 \cdot 31}{2} = 529$ bits. For comparison, the BCH5 seeds are only $1 + 2 \cdot 32 = 65$ bits in size.

3.1.7 Polynomials over Primes Scheme

The generating schemes in the previous sections are derived from error-correcting codes. A different method to generate k-wise independent random variables uses polynomials over a prime number field.

The generating function is defined as in [59]:

$$f(S, i) = \sum_{j=0}^{k-1} a_j i^j \mod p \tag{3–17}$$

for some prime $p > i$ with each a_j picked randomly from $\mathbb{Z}_p$. The seed S consists of the k coefficients a_j, $0 \leq j < k$, each represented on $\lceil \log p \rceil$ bits. This scheme generates k-wise independent random variables that have p values. To obtain the ± 1 random variables, we can take the least significant bit in the binary representation and map it to ± 1[2] .

The drawback of the polynomials scheme is that it requires multiplications over the field of the prime p, task that can be computationally intensive because it actually involves divisions and modulo operations. [10] introduce a method to increase the speed of these computations. The idea is to use Mersenne primes of the form $p = 2^l - 1$ which facilitate the computation of the remainder after a division with simple bitwise operations such as shifts, AND, OR, etc.

3.1.8 Toeplitz Matrices Scheme

The 2-universal Toeplitz family of 2^m-valued hash functions [8, 34] is defined over $m \times n$ Toeplitz matrices, i.e., matrices whose diagonals are homogeneous (all the entries in each diagonal contain the same value). Thus, a Toeplitz matrix is completely specified by the values in its first row and its first column. Each function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ in the Toeplitz family is specified by a seed $S = [U, v]$, where U is a random $m \times n$ Toeplitz matrix over $GF(2)$, and $v \in \{0,1\}^m$ is a random vector. Given a vector $i \in \{0,1\}^n$, $f(S, i) = Ui + v$, where the operations are over $GF(2)$. Notice that each f in the family is specified by $n + 2m - 1$ bits, the seed.

For $m = 1$, i.e., two-valued 2-wise independent random variables, the Toeplitz scheme is identical with BCH3, thus we do not need to analyze this scheme independently.

3.1.9 Tabulation Based Schemes

For degrees of independence higher than three, both the BCH and the polynomials schemes require exponentiation over the extension field $GF(2^n)$, computations that are

[2] This mapping introduces a small bias since the two values are not uniformly distributed. The bias is in the order of $\frac{1}{p}$, thus negligible.

time consuming. In this case, it could be faster to pre-compute and store (tabulate) the random variables in memory, instead of generating them on the fly. When the random variable corresponding to an index i is needed, it is simply extracted from a look-up table. For large domains multiple tables are maintained, each corresponding to a different sub-domain. The random variable is obtained as a simple function of the tabulated values.

[10] show that the function $\bar{f}$ mapping $i_0 i_1 \dots i_{n-1}$ to $f_0[i_0] \oplus f_1[i_1] \oplus \dots \oplus f_{n-1}[i_{n-1}]$ is 3-universal if each of the independent tabulated functions f_j is 3-universal. This implies that in order to obtain a 3-universal hash function for a large sequence, it is enough to divide the sequence into sub-sequences and to tabulate 3-universal functions for each such sub-sequence. The value of the function is computed by XOR-ing the tabulated values corresponding to each sub-sequence. This result was subsequently extended to 4-universal hashing by [58]. They show that the function $\bar{f}[ab] = f_0[a] \oplus f_1[b] \oplus f_2[a+b]$ is 4-universal if f_0, f_1, and f_2 are independent 4-universal hash functions.

The tabulation method reduces the processing time, but increases the required memory. As long as the the pre-computed tables fit in the fast memory, we expect an improvement in processing time. Since AGMS sketches require multiple instances of the estimator in order to provide the desired error guarantees, the space requirement could become a problem.

3.1.10 Performance Evaluation

We implemented BCH3, BCH5, EH3, and RM7, and we used the implementation in [58] for the polynomials over primes and the tabulation schemes. BCH5 was implemented by performing the i^3 operation arithmetically not in an extension field. This does not change the degree of independence of BCH5 for domains not too large, but it significantly improves the performance. The polynomials scheme uses Mersenne primes in order to speed-up the implementation. For the tabulation scheme, both 8 and 16 bits sub-sequences were tested, but only the best result is reported.

We ran our experiments on a two-processor *Xeon 2.80 GHz* machine[3] , each with a *512 KB* cache. The system has *1 GB* available memory and uses a *Fedora Core 3* operating system. As a comparison for our results, the time to read a word from a memory location that is not cached takes about 250 ns on this machine. We used the special assembly implementation of the dot-product for all the methods. Experiments consisted in generating $10,000$ variables i and $10,000$ seeds and then computing all possible combinations of random variables, i.e., $100,000,000$. Each experiment was run 100 times and the average of the results is reported. The relative error or error ($relative\ error = \frac{|true\ value - experimental\ value|}{true\ value}$) of a run was in general under 1%, with a maximum of 1.2%. Since the results are so stable, we do not report the actual error for individual experiments. The generation time, in nanoseconds per random variable, is reported in Table 3-3. When compared to the memory random access time, all the schemes, except RM7, are much faster. Indeed, EH3 is at least as fast as BCH3 (we believe it is actually faster since the extra operations maintain a better flow through the processor pipelines) and both are significantly faster than the polynomials scheme. The tabulation scheme is the fastest both for 2-wise and 4-wise independent random variables since the tables fit in the cache. Notice that EH3 and BCH5 give close results to the tabulation scheme although they generate the value of the random variables from scratch.

Table 3-3 also contains the size of the seed for the given schemes. n represents the number of bits the domain I can be represented on. For the polynomials over primes scheme, n is the smallest power of 2 for which $2^n \geq p$. As noted, the BCH schemes have the densest seeds, while the Reed-Muller scheme needs the largest seed. For the same degree of independence, the polynomials over primes scheme requires a seed double in size compared to BCH. The tabulation scheme does not store seeds. Instead it stores the

[3] While the machine has two processors, only one of them was used in our experiments.

values of the random variables for non-overlapping sub-domains. Thus, it has the most stringent memory requirement of all the schemes.

When deciding what scheme to use, the tradeoffs involved should be analyzed. While the tabulation scheme is the fastest, it also requires the largest amount of memory. For AGMS sketches estimations this could become a drawback. The BCH schemes have the least restrictive memory requirement while being quite fast. We will see that for fast range-summation (efficient sketching of intervals) a *real* generating scheme gives the best results. With a tabulation scheme all the points inside an interval have to be independently sketched, a solution that we try to improve.

3.2 Size of Join using AGMS Sketches

Using the current understanding of the AGMS sketches applied to the size of join problem, the 4-wise independence of the random variables is required in order to keep the variance small (we show why latter in the section). What we show in this section is that the Extended Hamming EH3 scheme [25] not only can be used as a reasonable replacement of the 4-wise independent schemes (this is the theoretical result proved in [25] for the L^1-difference of two streaming functions), but it is usually as good, in absolute big-$\mathcal{O}$ notation terms, and in certain situations significantly surpasses the 4-wise independent schemes for AGMS sketches solutions to the size of join problem. We provide here both the theoretical support for this statement and the empirical evidence.

We proceed by taking a close look at the estimation of the size of join of two relations using AGMS sketches. As already presented in Section 2.2, the size of join $|F \bowtie_A G|$ of two relations F and G with a common attribute A is defined as $|F \bowtie_A G| = \sum_{i \in I} f_i g_i$. To estimate this quantity, we use a ± 1 family of random variables $\{\xi_i | i \in I\}$ that are at least 2-wise independent (but not necessarily more). We define the sketches, one for each relation, as $X_F = \sum_{i \in I} f_i \xi_i$ and $X_G = \sum_{i \in I} g_i \xi_i$. The random variable $X = X_F X_G$ estimates, on expectation, the size of join $|F \bowtie_A G|$. It is easy to show that, due to the 2-wise independence, $E[X] = \sum_{i \in I} f_i g_i$, which is exactly the size of join. Since all the

generating schemes published in the literature (see Section 3.1 for a review of the most important ones) are 2-wise independent, they all produce estimates that are correct on average. The main difference between the schemes is the variance of X, $\mathrm{Var}[X]$. The smaller the variance, the better the estimation; in particular, the error of the estimate is proportional with $\frac{\sqrt{\mathrm{Var}[X]}}{E[X]}$.

We analyze now the variance of X. Following the technique in [3], we first compute $E[X^2]$ since $\mathrm{Var}[X] = E[X^2] - E[X]^2$. We have:

$$E\left[X^2\right] = E\left[\left(\sum_{i\in I} f_i\xi_i \sum_{i\in I} g_i\xi_i\right)^2\right] = \sum_{i\in I}\sum_{j\in I}\sum_{k\in I}\sum_{l\in I} f_i f_j g_k g_l E\left[\xi_i\xi_j\xi_k\xi_l\right] \tag{3–18}$$

The expression $E[\xi_i\xi_j\xi_k\xi_l]$ is equal to 1 if groups of two variables are the same (i.e., $i = j \wedge k = l$ or $i = k \wedge j = l$ or $i = l \wedge j = k$) since $\xi_i^2 = 1$ irrespective of the actual value of ξ_i ($1^2 = 1$ and $(-1)^2 = 1$). $E[\xi_i\xi_j\xi_k\xi_l]$ is equal to 0 if two variables are identical, say $i = j$, and two are different, since then $E[\xi_i\xi_j\xi_k\xi_l] = E[1 \cdot \xi_k\xi_l] = 0$, using the 2-wise independence property. The same is true when three of the variables are equal, say $i = j = k$, but the fourth one is not, since $E[\xi_i\xi_j\xi_k\xi_l] = E[\xi_i^3\xi_l] = 0$ (we use the fact that $\xi_i^3 = \xi_i$). The above observations are not dependent on what generating scheme is used, as long as it is at least 2-wise independent. The contribution of the 1 values to $E[X^2]$ adds up to:

$$\sum_{i\in I}\sum_{j\in I} f_i^2 g_j^2 + 2 \cdot \left(\sum_{i\in I} f_i g_i\right)^2 - 2 \cdot \sum_{i\in I} f_i^2 g_i^2 \tag{3–19}$$

In the expression of the variance $\mathrm{Var}[X]$, the middle term has the coefficient 1, instead of 2, since $E[X]^2$ is subtracted. The difference between the various generating schemes consists in what other terms have to be added to the above formula. The additional terms correspond only to the values of i, j, k, l that are all different (otherwise the contribution is 1 or 0 irrespective to the scheme, as explained before). We explore what are the additional terms for three of the schemes, namely BCH5, BCH3, and EH3.

3.2.1 Variance for BCH5

The BCH5 scheme is 4-wise independent, which means that for $i \neq j \neq k \neq l$, we have:

$$E\left[\xi_i \xi_j \xi_k \xi_l\right] = E\left[\xi_i\right] \cdot E\left[\xi_j\right] \cdot E\left[\xi_k\right] \cdot E\left[\xi_l\right] = 0 \tag{3–20}$$

since all the ξs are independent and $E\left[\xi_i\right] = 0$ for all generators. This means that no other terms are added to the above variance. The following formula results:

$$\mathrm{Var}\left[X\right]_{\mathrm{BCH5}} = \sum_{i \in I} \sum_{j \in I} f_i^2 g_j^2 + \left(\sum_{i \in I} f_i g_i \right)^2 - 2 \cdot \sum_{i \in I} f_i^2 g_i^2 \tag{3–21}$$

3.2.2 Variance for BCH3

BCH3 is only 3-wise independent, thus clearly it is not the case that, for all $i \neq j \neq k \neq l$, $E\left[\xi_i \xi_j \xi_k \xi_l\right] = 0$. To characterize the value of the expectation for different variables, the following results are extensively applied.

Proposition 1. *Consider the function* $XOR(x_1, x_2, \ldots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$ *defined over* n *binary variables* $x_1, x_2, \ldots, x_n$. *Then, the function XOR takes the values* 0 *and* 1 *equally often, each value* 2^{n-1} *times.*

Proof. We prove this proposition by induction on n.

Base case: $n = 2$. The truth table for the XOR function with two variables is given in Table 3-2. Function XOR takes the values 0 and 1 the same number of times, $2^{2-1} = 2$.

Inductive case: We suppose that the proposition is true for n and we have to show that it is also true for $n + 1$, that is, function $x_1 \oplus x_2 \oplus \cdots \oplus x_n \oplus x_{n+1}$ takes both values 0 and 1 2^n times each. For every combination of the first n variables, x_{n+1} takes one time the value 0 and one time the value 1. When $x_{n+1} = 0$, function XOR of $n + 1$ variables is identical with function XOR of n variables. This means that it takes the values 0 and 1 2^{n-1} times each. The effect of x_{n+1} being equal with 1 is to invert the result of the function XOR with n variables, that is, the combinations that gave result 0, give 1 now, and the combinations that gave 1 for n variables, give 0 for $n + 1$ variables. As we already

know that the number of combinations for both cases is equal to 2^{n-1} times each, by summing the results for $x_{n+1} = 0$ and $x_{n+1} = 1$, we obtain that function XOR with $n+1$ variables takes the values 0 and 1 2^n times each. □

Proposition 2. *Let the function F on n binary variables $x_1, x_2, \ldots, x_n$ be defined as:*

$$F(x_1, x_2, \ldots, x_n) = C \oplus S_1 \odot x_1 \oplus S_2 \odot x_2 \oplus \cdots \oplus S_n \odot x_n \tag{3–22}$$

where $C \in \{0,1\}$ is a constant and $S_k \in \{0,1\}$, for $1 \leq k \leq n$, are parameters. If there exists at least one S_k that is equal to 1, function F takes the values 0 and 1 equally often, each value 2^{n-1} times. Otherwise, function F evaluates to the constant C for all the combinations of $x_1, x_2, \ldots, x_n$.

Proof. Let $S_{k_1}, S_{k_2}, \ldots, S_{k_r}$, $0 \leq r < n$, be the parameters that are constrained to the value 0 and $S_{k_{r+1}}, \ldots, S_{k_n}$ be the parameters that equal 1. Then, we can rewrite function F as:

$$F(x_1, x_2, \ldots, x_n) = C \oplus x_{k_{r+1}} \oplus \cdots \oplus x_{k_n} \tag{3–23}$$

When $C = 0$ it has no effect on the value of the function F, while $C = 1$ inverts the result of the function. We know from Proposition 1 that the function XOR with $n - r$ variables takes the values 0 and 1 2^{n-r-1} times each. The 2^n combinations of n variables consist now of 2^r repetitions of the 2^{n-r} combinations of $n - r$ variables. It results that function F takes the value 0, as well as value 1, $2^{n-r-1} \cdot 2^r = 2^{n-1}$ times, for $C = 0$. The same result holds for $C = 1$ since $2^n - 2^{n-1} = 2^{n-1}$. For $r = n$, we have $F(x_1, x_2, \ldots, x_n) = C$. □

Lemma 1. *Assume the ξs are generated using the BCH3 scheme. Then, for $i \neq j \neq k \neq l$, $E[\xi_i \xi_j \xi_k \xi_l] = 0$ if $i \oplus j \oplus k \oplus l \neq 0$, and $E[\xi_i \xi_j \xi_k \xi_l] = 1$ if $i \oplus j \oplus k \oplus l = 0$, where $\oplus$ is the bitwise XOR.*

Proof. Let $S = [s_0, S_1]$ be the $(n+1)$-bits random seed with s_0 its first bit and S_1 the last n bits. Using the notations and the definition of BCH3 in Section 3.1, we have:

$$\xi_i = (-1)^{s_0 \oplus S_1 \cdot i} \tag{3–24}$$

With this, we obtain:

$$\begin{aligned} E\left[\xi_i\xi_j\xi_k\xi_l\right] &= E\left[(-1)^{s_0\oplus S_1\cdot i}\cdot(-1)^{s_0\oplus S_1\cdot j}\cdot(-1)^{s_0\oplus S_1\cdot k}\cdot(-1)^{s_0\oplus S_1\cdot l}\right] \\ &= E\left[(-1)^{S_1\cdot i\oplus S_1\cdot j\oplus S_1\cdot k\oplus S_1\cdot l}\right] \\ &= E\left[(-1)^{S_1\cdot(i\oplus j\oplus k\oplus l)}\right] \end{aligned} \tag{3–25}$$

Using the result in Proposition 2 with $C = 0$ and $S_1, \ldots, S_n$ set as the last n bits of the seed S, we know that the expression $S_1\cdot(i\oplus j\oplus k\oplus l)$ takes the values 0 and 1 equally often for random seeds S_1 when $i\oplus j\oplus k\oplus l \neq 0$ – this immediately implies that $E\left[\xi_i\xi_j\xi_k\xi_l\right] = 0$. When $i\oplus j\oplus k\oplus l = 0$, $S_1\cdot(i\oplus j\oplus k\oplus l) = 0$ irrespective of S_1, giving $E\left[\xi_i\xi_j\xi_k\xi_l\right] = 1$. □

This implies that the variance for BCH3 contains an additional term besides the ones that appear in the variance formula for BCH5. The extra term has the following form:

$$\Delta\mathrm{Var}\left[\mathrm{BCH3}\right] = \sum_{i\in I}\sum_{j\in I, j\neq i}\sum_{k\in I, k\neq i,j} f_i f_j g_k g_{i\oplus j\oplus k} \tag{3–26}$$

since $i\oplus j\oplus k\oplus l = 0$ implies $l = i\oplus j\oplus k$. The additional term in the BCH3 variance can be significantly large, implying a big increase over the variance of BCH5. However, as we will see in the experimental section, the influence of the extra-term almost vanishes for high-skew data and the variance of BCH3 becomes comparable with the variance of BCH5.

Example 5. *In order to give a clear image on the values $\Delta Var[BCH3]$ can take, we provide two extreme examples. First, consider that both relations F and G have uniform distributions with the value f, respectively g. This gives:*

$$\Delta Var[BCH3]_{uniform} = f^2g^2\sum_{i\in I}\sum_{j\in I, j\neq i}\sum_{k\in I, k\neq i,j} 1 \lessapprox f^2g^2|I|^3 \tag{3–27}$$

value that is an order of $|I|$ greater than the BCH5 variance, thus dominating it and increasing the BCH3 variance to possibly extreme high values. Second, consider that both F and G are very skewed. Actually, there exists only one positive frequency in F (f),

respectively G (g). The extra-variance becomes:

$$\Delta\, Var\left[BCH3\right]_{skewed} = fg \leq f^2 g^2 \tag{3–28}$$

which is smaller than the corresponding BCH5 variance, thus it can be ignored.

3.2.3 Variance for EH3

As BCH3, the Extended Hamming scheme (EH3) is also 3-wise independent, which might suggest that the variance of EH3 is similar to the variance of BCH3 (extra terms not in the variance of BCH5 have to appear, otherwise the scheme would be 4-wise independent). As we show next, even though only positive terms appeared in the variance for BCH3, in the EH3 variance negative terms appear as well. These negative terms, in certain circumstances, can compensate completely for the positive terms and give a variance that becomes **zero**.

Lemma 2. *Assume the ξs are generated using the EH3 scheme. Then, for $i \neq j \neq k \neq l$,*

$$E\left[\xi_i\xi_j\xi_k\xi_l\right] = \begin{cases} 0, & \text{if } i \oplus j \oplus k \oplus l \neq 0 \\ 1, & \text{if } i \oplus j \oplus k \oplus l = 0 \wedge \\ & \quad h(i) \oplus h(j) \oplus h(k) \oplus h(l) = 0 \\ -1, & \text{if } i \oplus j \oplus k \oplus l = 0 \wedge \\ & \quad h(i) \oplus h(j) \oplus h(k) \oplus h(l) = 1 \end{cases} \tag{3–29}$$

where $\oplus$ is the bitwise XOR.

Proof. We know that

$$\xi_i = (-1)^{s_0 \oplus S_1 \cdot i \oplus h(i)} \tag{3–30}$$

for random variables generated using the EH3 scheme. Replacing this form into the expression for $E\left[\xi_i\xi_j\xi_k\xi_l\right]$ and applying the same manipulations as in the proof of

Proposition 1, we get:

$$E\left[\xi_i\xi_j\xi_k\xi_l\right] = E\left[(-1)^{S_1\cdot(i\oplus j\oplus k\oplus l)\oplus(h(i)\oplus h(j)\oplus h(k)\oplus h(l))}\right] \tag{3–31}$$

If we use again Proposition 2 with $C = h(i) \oplus h(j) \oplus h(k) \oplus h(l)$ and $S_1, \ldots, S_n$ set as the last n bits of the seed S, we first observe that the expectation is 0 when $i \oplus j \oplus k \oplus l \neq 0$. When $i \oplus j \oplus k \oplus l = 0$, the expected value is always $(-1)^C$. For BCH3 the constant C always took the value 0, thus the expectation in that case was always 1. For EH3, the value of C depends on the function h – it is 1 when $h(i) \oplus h(j) \oplus h(k) \oplus h(l) = 1$. This implies the value -1 for $E\left[\xi_i\xi_j\xi_k\xi_l\right]$. □

Using the above result, we observe that $E\left[\xi_i\xi_j\xi_k\xi_l\right] = -1$ when $i \oplus j \oplus k \oplus l = 0$ and $h(i) \oplus h(j) \oplus h(k) \oplus h(l) = 1$. This means that, even though EH3 can have all the 1 terms BCH3 has, it can also have -1 terms, thus, it can potentially compensate for the 1 terms. Indeed, the following results show that this is exactly what happens under certain independence assumptions.

In order to predict the performance of EH3, we perform an average-case analysis of the scheme. Since AGMS sketches are randomized schemes, the average-case analysis is a more appropriate characterization than a worst-case analysis. [25] provide a worst-case analysis of EH3 for the particular case of computing the L^1-difference of two functions. For the size of join problem, the situation is more complicated because the frequencies have to be considered too and the parameter we are interested in is the self-join size of each relation.

To obtain an average analysis, consider first the theorem:

Theorem 10. *For i, j, k taking all the possible values over the domain $I = \{0, \ldots, 4^n - 1\}$, $n > 0$, the function $g(i, j, k) = h(i) \oplus h(j) \oplus h(k) \oplus h(i \oplus j \oplus k)$ takes the value 0 z_n times*

and the value 1 y_n *times, where* z_n *and* y_n *are given by the following recursive equations:*

$$z_1 = 40, \quad y_1 = 24$$

$$z_n = 40 \cdot z_{n-1} + 24 \cdot y_{n-1}$$

$$y_n = 24 \cdot z_{n-1} + 40 \cdot y_{n-1}$$

Proof. The base case, $n = 1$, can be easily verified by hand. The recursion is based on the observation that the groups of two bits from different h functions interact independently and give 40 zero values and 24 one values. When the results of two groups are XOR-ed, a zero is obtained if both groups are zero or both are one; a one is obtained if a group equals zero and the other group equals one. □

We have to characterize the behavior of function $g(i, j, k)$ for $i \neq j \neq k$ (when at least two of these variables are equal, we obtain the special case of the variance for BCH5 that we have already considered). The number of times at least two out of the three variables are equal is $eq_n = 3 \cdot (4^n)^2 - 2 \cdot 4^n$, which allows us to determine the desired quantities, i.e., the number of zeros is $z_n - eq_n$, while the number of ones is y_n. To determine the average behavior of EH3, we assume that neither the frequencies in F are correlated with the frequencies in G nor the frequencies in G are correlated amongst themselves. This allows us to model the quantity $l = i \oplus j \oplus k$ as a uniformly distributed random variable L over the domain I. Moreover, due to the same independence assumptions, function $g(i, j, k)$ can be modeled as a random variable G, independent of L, that has the same macro behavior as $g(i, j, k)$, i.e., it takes the values 0 and 1 the same number of times. With these random variables, we get:

$$E\left[g_L\right] = \frac{1}{|I|} \sum_{l \in I} g_l \tag{3–32}$$

and

$$E\left[(-1)^G\right] = 1 \cdot P[G = 0] + (-1) \cdot P[G = 1] = \frac{z_n - eq_n - y_n}{z_n - eq_n + y_n} \tag{3–33}$$

The expected value of the additional terms that appear in the variance of EH3 is given by:

$$\begin{aligned} E\left[\Delta\text{Var}\left[\text{EH3}\right]\right] &= E\left[\sum_{i\in I}\sum_{j\in I}\sum_{k\in I} f_i f_j g_k (-1)^G g_L\right] \\ &= \sum_{i\in I}\sum_{j\in I}\sum_{k\in I} f_i f_j g_k E\left[(-1)^G\right] E\left[g_L\right] \\ &= \frac{1}{|I|}\left(\sum_{i\in I} f_i\right)^2 \left(\sum_{i\in I} g_i\right)^2 \frac{z_n - eq_n - y_n}{z_n - eq_n + y_n} \end{aligned} \tag{3–34}$$

Overall, the variance of the EH3 generating scheme is:

$$\text{Var}\left[X\right]_{\text{EH3}} = \frac{1}{|I|}\left(\sum_{i\in I} f_i\right)^2 \left(\sum_{i\in I} g_i\right)^2 \frac{z_n - eq_n - y_n}{z_n - eq_n + y_n} + \text{Var}\left[X\right]_{\text{BCH5}} \tag{3–35}$$

Notice that the additional term over the variance for BCH5 is inversely proportional with the size of the domain I. Also, the last factor in the additional term takes small sub-unitary values. The combined effect of these two is to drastically decrease the influence of the extra-term on the EH3 variance, making it close to the BCH5 variance. Actually, there exist situations for which the EH3 variance is significantly smaller than the BCH5 variance. It can even become equal to **zero**. The following corollary states this surprising result:

Corollary 3. *If $f_i = f$ and $g_i = g$, $i \in I$, with f and g constants, and $|I| = 4^n$, then:*

$$Var\left[X\right]_{EH3} = 0 \tag{3–36}$$

The reason the variance of EH3 is **zero** when the distribution of both F and G is uniform is the fact that the -1 terms cancel out entirely the 1 terms. For less extreme cases, when the distribution of the two relations is close to a uniform distribution, EH3 significantly outperforms BCH5. This intuition is confirmed by the the following experimental results.

3.2.4 Empirical Evaluation

The purpose of the empirical evaluation is twofold. First, we want to validate the theoretical models for the variance of different generating schemes, especially the average behavior of EH3. Then, we want to compare the BCH3, EH3, and BCH5 schemes for estimations using AGMS sketches.

The main findings of our experimental study can be summarized as follows:

- **Validation of the theoretical model for the EH3 generating scheme.** Our study shows that the behavior of the EH3 generating scheme is well predicted by the theoretical model in Section 3.2.
- **BCH3 vs EH3 vs BCH5.** EH3 has approximately the same error, or, in the case of low-skew distributions, a significantly better error than the BCH5 scheme. This justifies our recommendation to use EH3 instead of BCH5. For high-skew data, BCH3 has approximately the same error as BCH5 and EH3, making it the perfect solution for sketching when the data is skewed.

We performed the experiments using the setup in Section 3.1. We give detailed descriptions of the datasets and the comparison methodology for each group of experiments.

Validation of the EH3 Model: To validate the average error formula in Equation 3–35, we generated Zipf distributed data with the Zipf coefficient ranging from 0 to 5 over domains of various sizes in order to estimate the self-join size. The prediction is performed using AGMS sketches with only one median, i.e., only averaging is used to decrease the error of the estimate. In Figure 3-1 we depict the comparison between the average error of the EH3 scheme and the theoretical prediction given by Equation 3–35 for a domain with the size $16,384$ and a relation containing $100,000$ tuples. Notice that, when the value of the Zipf coefficient is larger than 1, the prediction is accurate. When the Zipf coefficient is between 0 and 1, the error of EH3 is much smaller (it is zero for a uniform distribution). This is explained in Corollary 3, which states that the variance of EH3 is zero when the distribution of the data is uniform and the size of the domain is a power of 4. For completeness, we depict the comparison between the theoretical model and the experimental results for BCH5 scheme in Figure 3-2.

BCH3 vs EH3 vs BCH5: We performed the same experiments as in the previous section for BCH3, EH3, and BCH5, except that the number of medians was fixed to 10. The results of the experiments are depicted in Figure 3-3 (full-size picture) and Figure 3-4 (detailed picture focusing on EH3 and BCH5). Notice that the errors of BCH3, EH3, and BCH5 are virtually the same for Zipf coefficients greater than 1. While the EH3 error is significantly smaller for Zipf coefficients lower than 1, the BCH3 error can get extremely high values (100% relative error). These results confirm our theoretical characterizations both for BCH3 and EH3 schemes. When compared to the results in Figure 3-1, the errors are smaller by a factor of 3. This is due to the fact that 10 medians were used instead of only 1 and the medians have almost the same effect in reducing the error as the averages – the same observation was made in [18].

Given the theoretical and the experimental results in this section, and the fact that EH3 can be implemented more efficiently than BCH5, we recommend the exclusive use of the EH3 random variables for size of join estimations using AGMS sketches.

3.3 Conclusions

In this chapter we conducted both a theoretical as well as an empirical study of the various schemes used for the generation of the random variables that appear in AGMS sketches estimations. We provide theoretical and empirical evidence that EH3 can replace the 4-wise independent schemes for the estimation of the size of join using AGMS sketches. The main recommendation of this chapter is to use the EH3 random variables for AGMS sketches estimations of the size of join since they can be generated more efficiently and use smaller seeds than any of the 4-wise independent schemes. At the same time, the error of the estimate is as good as or, in the case when the distribution has low skew, better than the error provided by a 4-wise independent scheme.

Table 3-1. Orthogonal array $\mathrm{OA}(8, 4, 2, 3)$.

0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 3-2. Truth table for the function $x_1 \oplus x_2$.

x_1	x_2	$x_1 \oplus x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Table 3-3. Generation time and seed size.

Scheme	Time (ns)	Seed size
BCH3	10.8	$n+1$
EH3	7.3	$n+1$
Polynomials2	31.4	$2n$
Tabulation2	5.1	$2^b \cdot \frac{n}{b}$
BCH5	12.7	$2n+1$
Polynomials4	106.4	$4n$
RM7	$3,301$	$1+n+\frac{n(n-1)}{2}$
Tabulation4	10.3	$2^b \cdot \left(\frac{n}{b}\right)^{\log_2 3}$

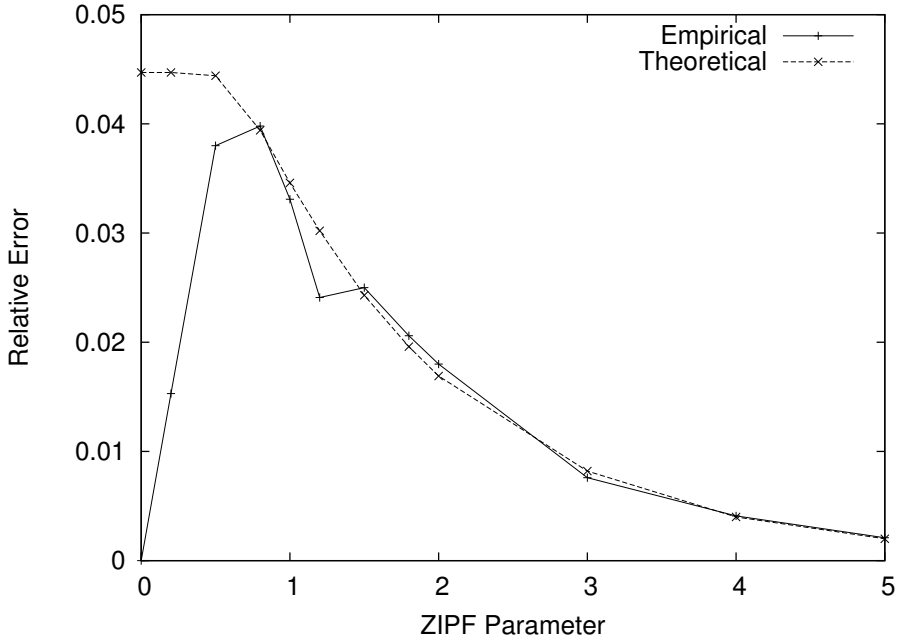


Figure 3-1. EH3 error.

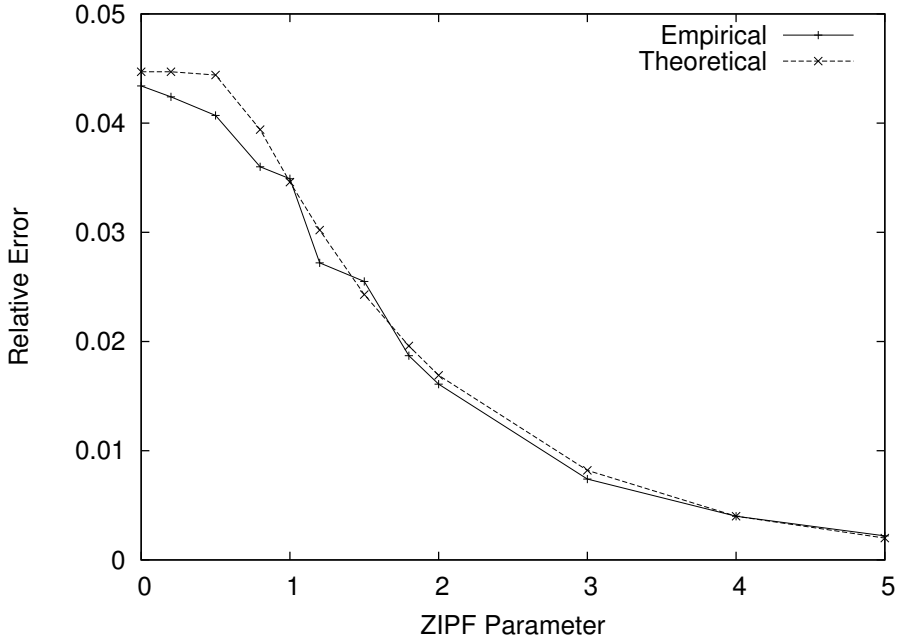


Figure 3-2. BCH5 error.

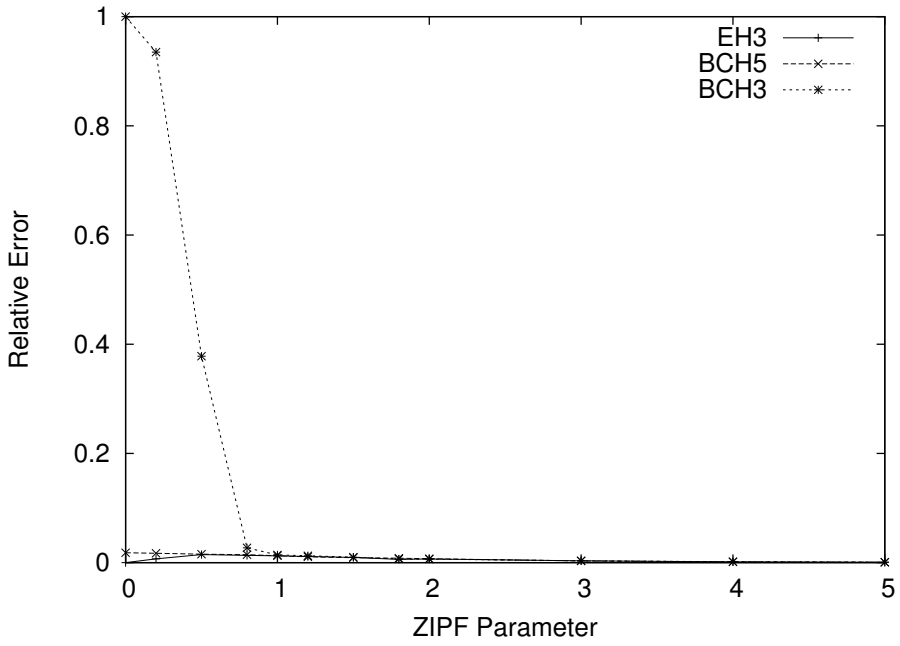


Figure 3-3. Scheme comparison (full).

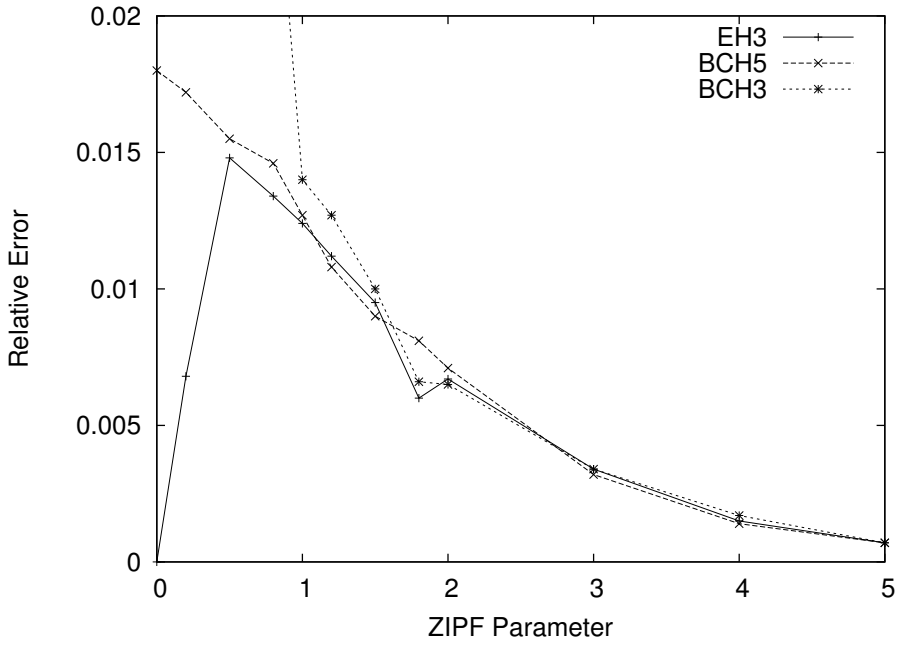


Figure 3-4. Scheme comparison (detail).

CHAPTER 4
SKETCHING SAMPLED DATA STREAMS

Sampling is used as a universal method to reduce the running time of computations – the computation is performed on a much smaller sample and then the result is only adjusted. Sketches are a popular approximation method for data streams and they proved to be useful for estimating frequency moments and aggregates over joins. A possibility to further improve the time performance of sketches is to compute the sketch over a sample of the stream rather than the entire data stream.

In this chapter we analyze the behavior of the sketch estimator when computed over a sample of the stream, not the entire data stream, for the size of join and the self-join size problems. Our analysis is developed for a generic sampling process. We instantiate the results of the analysis for two particular types of sampling—Bernoulli sampling which is used for load shedding and sampling with replacement which is used to generate i.i.d. samples from a distribution—and compare these particular results with the results of the basic sketch estimator. Our experimental results show that the accuracy of the sketch computed over a small sample of the data is in general close to the accuracy of the sketch estimator computed over the entire data even when the sample size is only 1% of the dataset size. This is equivalent to a speed-up factor of 100 when updating the sketch.

Data streaming received a lot of attention from the research community in the last decade. The requirement to process fast data streams motivates the need for approximation methods that make use of both small space and small time. AGMS sketches [3, 4] and their improved variant F-AGMS sketches [14, 52] proved to be a viable solution for estimating aggregates over joins. The main strengths of the sketching techniques are the simple and fast update procedure, the small memory requirement, and provable error guarantees. When the data streams that need to be processed are extremely fast, for example in the case of networking data or large datasets streamed over the Internet, it is desirable to further reduce the update time of sketches in order to

achieve the required processing rates. Sampling is a universal method for data reduction and, in principle, it can be used to reduce the amount of data that needs to be sketched. If samples are sketched instead of the original data, an immediate update time reduction results. This is similar to the existing *load shedding* techniques employed in data stream processing engines [56]. The main concern when samples rather than the original data are sketched is how to extend the error guarantees sketches provide to this new situation. The formulas resulting from such an analysis could be used to determine how aggressive the load shedding can be without a significant loss in the accuracy of the sketch over samples estimator. In other words, how much can the update time be improved without a significant accuracy degradation.

A seemingly unrelated, but, as shown in the chapter, technically related, problem is analyzing streams of samples from unknown distributions. Samples from unknown distributions—the so called *i.i.d. samples*—are the input to most of the online data-mining algorithms [21]. In this case the samples are not used as a data reduction technique, but rather they are the only information available about the unknown distribution. A fundamental problem in this context is how to characterize the unknown distribution using only the samples. This is one of the fundamental problems in statistics [54]. If the samples are streamed, as is the case in online data-mining, the aim is to characterize the unknown distribution by using small space only, thus making sketches a natural candidate for computations that involve aggregates. It is a simple matter to use sketches in order to estimate aggregates of the samples. If predictions about the unknown distribution need to be made, the problem is significantly more difficult. Interestingly, this problem is mathematically similar to the load shedding problem in which sampling is used to reduce the update time of sketching. Both problems require the analysis of sketching samples: Bernoulli sampling in the case of load shedding and sampling with replacement in the case of characterizing unknown distributions from samples.

In this chapter we analyze the sketch over samples estimator for a generic sampling process. Then we instantiate the results for Bernoulli sampling and sampling with replacement. Our technical contributions are:

- We analyze the sampling methods in the frequency domain. This is a building block necessary to combine the analysis of sketches and sampling.
- We provide a generic analysis of the sketch over samples estimator. The analysis consists in expressing the first two frequency moments of the estimator in terms of the moments of the sampling frequency random variables.
- We instantiate the generic analysis to derive formulas for sketching Bernoulli samples. This immediately indicates how random load shedding for sketching data streams behaves.
- We instantiate the generic analysis to derive formulas for sketching samples with replacement from a large population. The analysis readily generalizes to sketching i.i.d. samples from an unknown distribution. The ability to sketch i.i.d. data is important if sketches are to be used for data-mining applications.
- We present empirical evidence that the analysis is necessary since the error of the sketch over samples estimator is not simply the sum of the errors of the two individual estimators. The interaction, which is predicted by the analysis, plays a major role. The experiments also point out that in the majority of the cases a 1% sample results in minimal error degradation – the sketching of streams can thus be speed-up by a factor of 100.

There exists a large body of work on approximate query processing methods. The idea of combining two estimators to capitalize on the strengths of both is not new. F-AGMS sketches [14] are essentially a combination of random histograms and AGMS sketches. [30] presents a method to build incremental histograms from samples. To the best of our knowledge, sketching and sampling have not been combined in a principled fashion before. The main difficulty in characterizing sketches over samples is the fact that the sampling analysis [35, 39] is performed in the tuple domain while the sketch analysis [3] is performed in the frequency domain. This is the first obstacle we overcome in this chapter. The work on sketching probabilistic data streams [13, 38] is somehow similar to our work. The important difference is the fact that sampling is part of the estimate in our work while it represents only a way to interpret the probabilistic data in

the related work. The results in [13] do not characterize the sketch over sample estimator but approximate the probabilistic aggregates using sketches. The only overlap in terms of analysis seems to be the computation of the expected value of sketch over samples for the second frequency moment computation in [38].

In the rest of the chapter, Section 4.1 gives an overview on sampling while Section 4.2 introduces sketches. The formal analysis of the combined sketch over samples estimator is detailed in Section 4.3. The empirical evaluation of the combined estimator is presented in Section 4.4.

4.1 Sampling

Sampling as an approximation technique consists in obtaining samples F' and G' from relations F and G, respectively, computing the size of join aggregate over the samples, and applying a correction to ensure that the sampling estimator is unbiased. This method is generic and applies to all types of sampling. In order to simplify the theoretical developments, we keep the treatment of sampling as generic as possible.

In Section 2.1 we expressed the size of join aggregate as a function of f_i and g_i, the frequencies of value i of the join attribute in relations F and G, respectively. If we define f'_i and g'_i to be the frequencies of i in F' and G', respectively, the size of join of the sample relations is:

$$|F' \bowtie_A G'| = \sum_{i \in I} f'_i g'_i \tag{4–1}$$

f'_i and g'_i are random variables that depend on the type of sampling and the parameters of the sampling process. Interestingly, a large part of the characterization of sampling can be carried out without specifying the type of sampling. This is also true for sketches over samples in Section 4.3.

4.1.1 Generic Sampling

In general, $|F' \bowtie_A G'|$ is not an unbiased estimator for the size of join $|F \bowtie_A G|$. Fortunately, in the majority of the cases a constant correction that *scales* for the difference in size between the samples and the original relations can be made to obtain an unbiased

estimator. If we define the estimator for the size of join as $X = C \sum_{i \in I} f'_i g'_i$, where C is the scaling factor, we can determine the value of C such that X is unbiased. In order to derive error bounds for the estimator X, the moments $E[X]$ and $\mathrm{Var}[X]$ have to be computed. It turns out that expressions for $E[X]$ and $\mathrm{Var}[X]$ can be written for generic sampling, as we show below. There are two distinct cases that need separate treatment. The first case is when relations F and G are different and the samples are obtained independently from the two relations. The second case is when F and G are identical, thus only one sample is available. This situation arises in the case of self-join size.

Size of Join: When F' and G' are obtained independently, the random variables f'_i and g'_i are independent. We immediately have:

$$\begin{aligned} E[X] &= C \sum_{i \in I} E[f'_i] E[g'_i] \\ \mathrm{Var}[X] &= C^2 \left[\sum_{i \in I} \sum_{j \in I} E\left[f'_i f'_j\right] E\left[g'_i g'_j\right] - \left(\sum_{i \in I} E[f'_i] E[g'_i] \right)^2 \right] \end{aligned} \tag{4–2}$$

Self-Join Size: When F and G are identical and only the sample F' is available, the random variables f'_i and g'_i are also identical. Then, we obtain:

$$\begin{aligned} E[X] &= C \sum_{i \in I} E\left[f'^2_i\right] \\ \mathrm{Var}[X] &= C^2 \left[\sum_{i \in I} \sum_{j \in I} E\left[f'^2_i f'^2_j\right] - \left(\sum_{i \in I} E\left[f'^2_i\right] \right)^2 \right] \end{aligned} \tag{4–3}$$

A specific type of sampling would determine the value of the expectations $E[f'_i]$, $E[g'_i]$, $E\left[f'_i f'_j\right]$, $E\left[g'_i g'_j\right]$, $E[f'^2_i]$ and $E\left[f'^2_i f'^2_j\right]$. Deriving final formulas for $E[X]$ and $\mathrm{Var}[X]$ and determining the constant C becomes just a matter of plugging in these quantities for a specific type of sampling.

4.1.2 Bernoulli Sampling

Consider that the sampling process is Bernoulli sampling. Each tuple in F and G is selected independently in the sample F' and G' with probability p or q, $0 \leq p \leq 1$,

$0 \leq q \leq 1$, respectively. Then, f_i' and g_i' are independent binomial random variables, $f_i' = \text{Binomial}(f_i, p)$ and $g_i' = \text{Binomial}(g_i, q)$, respectively, with expected values:

$$\begin{aligned} E\left[f_i'\right] &= pf_i \\ E\left[g_i'\right] &= qg_i \end{aligned} \tag{4–4}$$

Size of Join: The expectation of the size of join estimator $E[X]$ is then:

$$E[X] = pq \sum_{i \in I} f_i g_i \tag{4–5}$$

which is different from the size of join $|F \bowtie_A G|$. Thus, in order for X to be an unbiased estimator, it has to be scaled with the product of the inverses of the two probabilities p and q, respectively, i.e., $C = \frac{1}{pq}$. For deriving the variance $\text{Var}[X] = E[X^2] - E^2[X]$, expectations of the form $E\left[f_i' f_j'\right]$ have to be computed. f_i' and f_j' are independent random variables whenever $i \neq j$ because the decision to include a tuple in the sample is independent for each tuple, thus $E\left[f_i' f_j'\right] = E\left[f_i'\right] E\left[f_j'\right]$. When $i = j$, $E\left[f_i'^2\right]$ is the second frequency moment of a binomial random variable. The variance $\text{Var}[X]$ is then:

$$\text{Var}[X] = \frac{1}{p^2 q^2} \left[\sum_{i \in I} E\left[f_i'^2\right] E\left[g_i'^2\right] - \sum_{i \in I} E^2\left[f_i'\right] E^2\left[g_i'\right] \right] \tag{4–6}$$

where $\sum_{i \in I} \sum_{j \in I} E\left[f_i' f_j'\right] E\left[g_i' g_j'\right] = \sum_{i \in I} E\left[f_i'^2\right] E\left[g_i'^2\right] + \left(\sum_{i \in I} E\left[f_i'\right] E\left[g_i'\right]\right)^2 - \sum_{i \in I} E^2\left[f_i'\right] E^2\left[g_i'\right]$ is used for simplifying the formula. The frequency moments of standard random variables, as the binomial is, can be found in any statistical resource, for example [54]. After we plug in the second frequency moment of the binomial random variables, i.e., $E\left[f_i'^2\right] = pf_i\left(1 - p + pf_i\right)$ and $E\left[g_i'^2\right] = qg_i\left(1 - q + qg_i\right)$, in Equation 4–6, we obtain the final formula for the variance:

$$\text{Var}[X] = \frac{1-p}{p} \sum_{i \in I} f_i g_i^2 + \frac{1-q}{q} \sum_{i \in I} f_i^2 g_i + \frac{(1-p)(1-q)}{pq} \sum_{i \in I} f_i g_i \tag{4–7}$$

Self-Join Size: The expectation of the self-join size estimator is given by:

$$E[X] = p^2 \sum_{i \in I} f_i^2 + p(1-p) \sum_{i \in I} f_i \tag{4–8}$$

Then, the self-join size unbiased estimator X has to be defined as $X = \frac{1}{p^2} \sum_{i \in I} f_i'^2 - \frac{1-p}{p} \sum_{i \in I} f_i$, with the scaling constant $C = \frac{1}{p^2}$. The effect of the extra term is minimal, the only modification being that a bias correction is needed when the estimate is actually computed. The variance is not affected since $\text{Var}[aX + b] = a^2 \text{Var}[X]$, for a, b constants. The expectations $E\left[f_i'^2 f_j'^2\right]$ follow the same rules as $E\left[f_i' f_j'\right]$ in the size of join variance. They are independent for $i \neq j$, while for $i = j$ they generate the fourth frequency moment. After simplifications, we obtain the following formula for the variance:

$$\text{Var}[X] = \frac{1}{p^4} \cdot \left[\sum_{i \in I} E\left[f_i'^4\right] - \sum_{i \in I} E^2\left[f_i'^2\right] \right] \tag{4–9}$$

4.1.3 Sampling with Replacement

A sample of fixed size can be generated by repeatedly choosing a random tuple from the base relation for the specified number of times. If the same tuple can appear in the sample multiple times, the process is sampling with replacement. In this case the random variables corresponding to the frequencies in the sample, f_i' and g_i', respectively, are the components of a multinomial random variable with parameters the size of the sample and the probability $\frac{f_i}{|F|}$ and $\frac{g_i}{|G|}$, respectively, where $|F|$ and $|G|$ are the size of F and G. Since each component of a multinomial random variable is a binomial random variable, the expectations in Equation 4–4 still hold but with different probabilities:

$$\begin{aligned} E[f_i'] &= \frac{|F'|}{|F|} f_i \\ E[g_i'] &= \frac{|G'|}{|G|} g_i \end{aligned} \tag{4–10}$$

Size of Join: The size of join estimator X is defined as follows:

$$X = \frac{|F|}{|F'|} \frac{|G|}{|G'|} \sum_{i \in I} f_i' g_i' \tag{4–11}$$

where $\frac{|F|}{|F'|}$ and $\frac{|G|}{|G'|}$ are scaling factors that make the estimator unbiased, i.e., $C = \frac{|F|}{|F'|}\frac{|G|}{|G'|}$. The expectations of the form $E\left[f'_i f'_j\right]$ needed for the derivation of the variance quantify the interaction between the binomial components of the multinomial random variable characterizing the sampling frequencies. They can be derived from the moment generating function corresponding to a multinomial distribution. With $E\left[f'_i f'_j\right] = \frac{|F'|(|F'|-1)}{|F|^2} f_i f_j$ and $E\left[f'^2_i\right] = \frac{|F'|(|F'|-1)}{|F|^2} f_i^2 + \frac{|F'|}{|F|} f_i$, and $E\left[g'_i g'_j\right] = \frac{|G'|(|G'|-1)}{|G|^2} g_i g_j$ and $E\left[g'^2_i\right] = \frac{|G'|(|G'|-1)}{|G|^2} g_i^2 + \frac{|G'|}{|G|} g_i$, respectively, the variance is:

$$\begin{aligned}\mathrm{Var}\left[X\right] = \frac{|F|}{|F'|}\frac{|G|}{|G'|}\sum_{i\in I} f_i g_i + |F|\frac{|G'|-1}{|G'|}\sum_{i\in I} f_i g_i^2 + \frac{|F'|-1}{|F'|}|G|\sum_{i\in I} f_i^2 g_i \\ + \frac{\left(|F'|-1\right)\left(|G'|-1\right) - |F'||G'|}{|F'||G'|}\left(\sum_{i\in I} f_i g_i\right)^2\end{aligned} \tag{4–12}$$

The more complicated formula of the variance in the case of sampling with replacement is due to the constraint imposed by the fixed size of the sample, thus the more complicated interaction between the random variables corresponding to the frequencies f'_i and g'_i.

Self-Join Size: Following a similar treatment as for Bernoulli sampling, the self-join size unbiased estimator X is defined as follows:

$$X = \frac{|F|^2}{|F'|\left(|F'|-1\right)}\sum_{i\in I} f'^2_i - \frac{|F|^2}{|F'|-1} \tag{4–13}$$

with the scaling constant $C = \frac{|F|^2}{|F'|(|F'|-1)}$. Notice that the estimator is defined for sample sizes larger than 1, but in order to compute the variance the size of the sample has to be larger than 3. The bias correction requires knowledge only about the size of the base relation and the size of the sample. The variance can be computed in a similar way as for Bernoulli sampling since the components of a multinomial random variable are binomial random variables, thus they have the same frequency moments. Expectations of the form $E\left[f'^2_i f'^2_j\right]$ appearing in the variance can be derived from the moment generating function

of the multinomial random variable, i.e.:

$$E\left[f_i'^2 f_j'^2\right] = \frac{|F'|\left(|F'|-1\right)}{|F|^2} f_i f_j \left[1 + \frac{|F'|-2}{|F|}\left(f_i + f_j\right) + \frac{\left(|F'|-2\right)\left(|F'|-3\right)}{|F|^2} f_i f_j\right] \quad (4\text{–}14)$$

4.2 Sketches

While sampling techniques select a random subset of tuples from the input relation, sketching techniques summarize all the tuples as a small number of random variables. This is accomplished by projecting the domain of the input relation on a significantly smaller domain using random functions. Multiple sketching techniques are proposed in the literature for estimating the size of join and the second frequency moment (see [52] for details). Although using different random functions, i.e., $\{+1, -1\}$ or hashing, the existing sketching techniques have similar analytical properties, i.e., the sketch estimators have the same variance. For this reason we focus on the basic AGMS sketches [3, 4] throughout the chapter.

The basic AGMS sketch of relation F consists of a single random variable S that summarizes all the tuples t from F. S is defined as:

$$S = \sum_{t \in F} \xi_{t.A} = \sum_{i \in I} f_i \xi_i \quad (4\text{–}15)$$

where ξ is a family of $\{+1, -1\}$ random variables that are 4–wise independent. Essentially, a random value of either $+1$ or -1 is associated to each point in the domain of attribute A. Then, the corresponding random value is added to the sketch S for each tuple t in the relation. We can define a sketch T for relation G in a similar way and using the same family ξ.

Size of Join: The sketch-based estimator X defined as:

$$X = S \cdot T = \sum_{i \in I} f_i \xi_i \cdot \sum_{j \in I} g_j \xi_j \quad (4\text{–}16)$$

is an unbiased estimator for the size of join $|F \bowtie_A G|$. The variance of the sketch estimator is given by:

$$\text{Var}[X] = \sum_{i \in I} f_i^2 \sum_{j \in I} g_j^2 + \left(\sum_{i \in I} f_i g_i \right)^2 - 2 \sum_{i \in I} f_i^2 g_i^2 \tag{4–17}$$

Self-Join Size: The unbiased estimator for the self-join size is defined as:

$$X = S^2 = \sum_{i \in I} \sum_{j \in I} f_i f_j \xi_i \xi_j \tag{4–18}$$

The variance of the sketch estimator is given by:

$$\text{Var}[X] = 2 \left[\left(\sum_{i \in I} f_i^2 \right)^2 - \sum_{i \in I} f_i^4 \right] \tag{4–19}$$

A common technique to reduce the variance of an estimator is to generate multiple independent instances of the basic estimator and then to build a more complex estimator as the average of the basic estimators. While the expected value of the complex estimator is equal with the expectation of one basic estimator, the variance is reduced by a factor of n since $\text{Var}\left[\frac{1}{n} \cdot \sum_{k=1}^{n} X_k\right] = \frac{1}{n^2} \sum_{k=1}^{n} \text{Var}[X_k] = \frac{\text{Var}[X_k]}{n}$, where n is the number of basic estimators being averaged. This technique can be applied to reduce the variance of the sketch estimator if different families ξ are used for the basic estimators (see [3, 4] for details).

4.3 Sketches over Samples

Given the ability of sampling to make predictions about an entire dataset from a randomly selected subset and that sketches require the summarizing of the entire dataset in order to determine any of its properties, an interesting question that immediately arises is how to combine these two randomized techniques. Although the intuitive answer to this question seems to be simple – the sketch is computed over a sample of the data instead of the entire dataset – as we show in Section 4.4, the behavior of the combined estimator is not the simple composition of the individual behaviors of the ingredients. A

careful analytical characterization of the estimator needs to be carried out. Furthermore, the sampling process can be either explicit and executed as an individual step before sketching is done or implicit, situation in which the input dataset is assumed to be a sample from a large population. In the first case, a significant speed-up in updating the sketch structure can be obtained since only a random subset of the data is actually sketched. This process is essentially a *load shedding* technique for sketching extremely fast data streams that cannot be otherwise sketched. It can be implemented as an explicit Bernoulli sampling that randomly filters the tuples that update the sketch structure. In the second case, the data is assumed to be a sample from a large population and the goal is to determine properties of the population based on the sample. The sample itself is assumed to be large enough so it cannot be stored explicitly, thus sketching is required. If the population is infinite, the entire process can be seen as *sketching i.i.d. samples* from an unknown distribution. For practical reasons we view the data as a sample with replacement from a finite population and we carry out the analysis in this context. The analysis straightforwardly extends to i.i.d. samples if all estimators are normalized by the size of the population and the limit, when the population size goes to infinity, is taken. In such a circumstance, the frequencies in the original unknown population become densities of the unknown population, but everything else remains the same.

In this section, we provide a generic framework for sketching sampled data streams in order to estimate the size of join and the self-join size. Then we compute the first two frequency moments of the combined estimator for two particular types of sampling – Bernoulli sampling and sampling with replacement. This provides sufficient information to allow the derivation of confidence bounds.

4.3.1 Generic Sampling

Consider F' to be a generic sample obtained from relation F. Sketching the sample F' is similar to sketching the entire relation F and consists in summarizing the sampled

tuples t' as follows:

$$S = \sum_{t' \in F'} \xi_{t'.A} = \sum_{i \in I} f'_i \xi_i \tag{4–20}$$

where ξ is a family of $\{+1, -1\}$ random variables that are 4–wise independent. A sample G' from relation G can be sketched in a similar way using the same family ξ:

$$T = \sum_{t' \in G'} \xi_{t'.A} = \sum_{i \in I} g'_i \xi_i \tag{4–21}$$

Size of Join: We define the estimator X for the size of join $|F \bowtie_A G|$ based on the sketches computed over the samples as follows:

$$X = C \cdot ST = C \cdot \sum_{i \in I} f'_i \xi_i \cdot \sum_{j \in I} g'_j \xi_j \tag{4–22}$$

Notice that the estimator is similar to the sketch estimator computed over the entire dataset in Equation 4–16 multiplied with a constant scaling factor that compensates for the difference in size. The constant C is the part that comes from sampling.

Self-Join Size: The self-join size or second frequency moment of a relation is the particular case of size of join between two instances of the same relation. One way of analyzing the sketches over samples estimator for the self-join size problem is to build two independent samples and two independent sketches from the same base relation and then to apply the results corresponding to size of join. Although sound from an analytical point of view, this solution is inefficient in practice. In the following we consider a practical solution that requires the construction of only one sample and one sketch from the base relation. A new estimator for the self-join size has to be defined instead, but the analysis is closely related to the analysis of the size of join estimator. With S defined in Equation 4–20, we define the self-join size estimator X as follows:

$$X = S^2 = C \cdot \left(\sum_{i \in I} f'_i \xi_i \right)^2 = C \cdot \sum_{i \in I} f'_i \xi_i \cdot \sum_{j \in I} f'_j \xi_j \tag{4–23}$$

where C is the same scaling factor compensating for the difference in size. Notice that the difference between the size of join estimator and the self-join size estimator is only at the sampling level since the same family of ξ random variables is used for sketching in both cases. For this reason we carry out the analysis for the two estimators in parallel and make the distinction only when necessary.

In order to derive confidence bounds for the estimator X, the first two moments, expected value and variance, have to be computed. Intuitively, the scaling factor C should compensate for the difference in size and make the estimator unbiased. Since the two processes, sampling and sketching, are independent and sequential, the interaction between them is minimal and the sum of the two variances should be a good estimator for the variance of the combined estimator. In the following, we derive the exact formulas for the expectation and the variance in the generic case. The independence of the families of random variables corresponding to sampling and sketching, f'_i, g'_i and ξ, respectively, plays an important role in simplifying the computation. This independence is due to the independence of the two random processes. Then we consider Bernoulli sampling and sampling with replacement and show that the intuition is correct in the case of expectation. For the variance, the intuition proves to be wrong since the interaction between sketching and sampling is more complex and it can be characterized only through a detailed analysis. The empirical results in Section 4.4 confirm our findings.

The expectation $E[X]$ can be derived as follows:

$$\begin{aligned} E[X] &= C \cdot E\left[\sum_{i\in I}\sum_{j\in I} f'_i\xi_i \cdot g'_j\xi_j\right] \\ &= C \cdot \sum_{i\in I}\sum_{j\in I} E\left[f'_i g'_j\right] E[\xi_i\xi_j] \\ &= C \cdot \sum_{i\in I} E[f'_i g'_i] \end{aligned} \tag{4–24}$$

since $E[\xi_i\xi_j] = E[\xi_i] \cdot E[\xi_j] = 0$ whenever $i \neq j$ due to the 4–wise independence of the family ξ, and $E[\xi_i^2] = 1$.

Size of Join: For size of join, one more simplification step is possible due to the independence of the sampling processes corresponding to the two relations:

$$E[X] = C \cdot \sum_{i \in I} E[f'_i] E[g'_i] \tag{4–25}$$

Self-Join Size: For self-join size, the samples are identical, thus:

$$E[X] = C \cdot \sum_{i \in I} E\left[f'^2_i\right] \tag{4–26}$$

In essence, the expectation $E[X]$ is completely determined by the properties of the sampling process, i.e., the frequency moments of the random variables corresponding to the sampling frequencies. The interaction between sketching and sampling is minimal.

In order to compute the variance $\mathrm{Var}[X]$, the expectation $E[X^2]$ has to be evaluated first since $\mathrm{Var}[X] = E[X^2] - E^2[X]$. Although $E[X^2]$ contains 8 random variables, the computation is again significantly simplified by the independence of the sampling and sketching processes. $E[X^2]$ can be derived as follows:

$$\begin{aligned} E\left[X^2\right] &= C^2 \cdot E\left[\sum_{i \in I} \sum_{j \in I} \sum_{i' \in I} \sum_{j' \in I} f'_i \xi_i \cdot g'_j \xi_j \cdot f'_{i'} \xi_{i'} \cdot g'_{j'} \xi_{j'}\right] \\ &= C^2 \cdot \sum_{i \in I} \sum_{j \in I} \sum_{i' \in I} \sum_{j' \in I} E\left[f'_i g'_j f'_{i'} g'_{j'}\right] \cdot E\left[\xi_i \xi_j \xi_{i'} \xi_{j'}\right] \end{aligned} \tag{4–27}$$

Size of Join: For size of join, more simplifications are possible due to the independence of the sampling processes corresponding to the two relations and the 4–wise independence of the ξ family (see [51] for details). With $E[X^2]$ given by:

$$\begin{aligned} &E\left[X^2\right] = \\ &C^2 \cdot \left[\sum_{i \in I} \sum_{j \in I} E\left[f'^2_i\right] E\left[g'^2_j\right] + 2 \cdot \sum_{i \in I} \sum_{j \in I} E\left[f'_i f'_j\right] E\left[g'_i g'_j\right] - 2 \cdot \sum_{i \in I} E\left[f'^2_i\right] E\left[g'^2_i\right]\right] \end{aligned} \tag{4–28}$$

we obtain the formula for the variance:

$$\begin{aligned}\mathrm{Var}\left[X\right] = C^2 \cdot \Bigg[& \sum_{i\in I} E\left[f_i'^2\right] \sum_{j\in I} E\left[g_j'^2\right] + 2 \cdot \sum_{i\in I} \sum_{j\in I} E\left[f_i' f_j'\right] E\left[g_i' g_j'\right] \\ & -2 \cdot \sum_{i\in I} E\left[f_i'^2\right] E\left[g_i'^2\right] - \left(\sum_{i\in I} E\left[f_i'\right] E\left[g_i'\right]\right)^2 \Bigg]\end{aligned} \tag{4–29}$$

Notice that the variance of sketching over generic sampling is an expression depending only on the properties of the sampling process. More precisely, in order to evaluate the variance, only expectations of the form $E\left[f_i'\right]$ and $E\left[f_i' f_j'\right]$ have to be computed, where f_i' and f_j' are random variables corresponding to the frequencies in the sample.

Self-Join Size: For self-join size, the samples are identical, thus $E\left[X^2\right]$ is given by:

$$E\left[X^2\right] = C^2 \cdot \left[3 \cdot \sum_{i\in I} \sum_{j\in I} E\left[f_i'^2 f_j'^2\right] - 2 \cdot \sum_{i\in I} E\left[f_i'^4\right]\right] \tag{4–30}$$

The variance of the self-join size estimator is then:

$$\mathrm{Var}\left[X\right] = C^2 \left[3 \cdot \sum_{i\in I} \sum_{j\in I} E\left[f_i'^2 f_j'^2\right] - 2 \cdot \sum_{i\in I} E\left[f_i'^4\right] - \left(\sum_{i\in I} E\left[f_i'^2\right]\right)^2\right] \tag{4–31}$$

The averaging technique applied to reduce the variance of basic sketches in Section 4.2 cannot be used straightforwardly in the case of sketches computed over samples. This is the case since, although the basic sketch estimators are built independently using different ξ families of random variables, they are computed over the same sample and this introduces correlations between any two estimators. The variance of the average estimator is in this case:

$$\mathrm{Var}\left[\frac{1}{n} \cdot \sum_{k=1}^{n} X_k\right] = \frac{1}{n}\left[\mathrm{Var}\left[X_k\right] + (n-1) \cdot \mathrm{Cov}_{k\neq l}\left[X_k, X_l\right]\right] \tag{4–32}$$

where n is the number of basic estimators being averaged and $\mathrm{Cov}\left[X_k, X_l\right] = E\left[X_k X_l\right] - E\left[X_k\right] E\left[X_l\right]$ is the covariance between any two basic estimators. Thus, in order to derive the variance of the average estimator, the expectation $E\left[X_k X_l\right]$ has to be evaluated first. This derivation is similar to the derivation of $E\left[X^2\right]$ with the difference that the

estimators X_k and X_l are built over the same sample using different sketch families of random variables:

$$\begin{aligned} E\left[X_k X_l\right] &= C^2 \cdot E\left[\sum_{i\in I}\sum_{j\in I}\sum_{i'\in I}\sum_{j'\in I} f'_i\xi_i^{(k)} \cdot g'_j\xi_j^{(k)} \cdot f'_{i'}\xi_{i'}^{(l)} \cdot g'_{j'}\xi_{j'}^{(l)}\right] \\ &= C^2 \cdot \sum_{i\in I}\sum_{j\in I}\sum_{i'\in I}\sum_{j'\in I} E\left[f'_i g'_j f'_{i'} g'_{j'}\right] E\left[\xi_i^{(k)}\xi_j^{(k)}\right] E\left[\xi_{i'}^{(l)}\xi_{j'}^{(l)}\right] \end{aligned} \tag{4–33}$$

Size of Join: With $E\left[X_k X_l\right] = C^2 \cdot \sum_{i\in I}\sum_{j\in I} E\left[f'_i f'_j\right] E\left[g'_i g'_j\right]$, we can write a formula for the variance of the average estimator as follows:

$$\begin{aligned} \mathrm{Var}\left[\frac{1}{n}\cdot\sum_{k=1}^{n} X_k\right] = C^2 \cdot \Bigg[&\sum_{i\in I}\sum_{j\in I} E\left[f'_i f'_j\right] E\left[g'_i g'_j\right] - \left(\sum_{i\in I} E\left[f'_i\right] E\left[g'_i\right]\right)^2 \\ &+ \frac{1}{n}\left(\sum_{i\in I} E\left[f_i'^2\right]\sum_{j\in I} E\left[g_j'^2\right] + \sum_{i\in I}\sum_{j\in I} E\left[f'_i f'_j\right] E\left[g'_i g'_j\right] - 2\cdot\sum_{i\in I} E\left[f_i'^2\right] E\left[g_i'^2\right]\right)\Bigg] \end{aligned} \tag{4–34}$$

which is the sum of the variance of the generic sampling estimator in Equation 4–2 and a term that seems to be similar to the variance of the sketch estimator in Equation 4–17. Notice that the complexity of the formula has not increased when compared to the variance of the basic estimator (Equation 4–29), only expectations of the form $E\left[f'_i\right]$ and $E\left[f'_i f'_j\right]$ having to be computed. At the same time, the improvement is less significant than a factor of n obtained in the case of independent estimators.

Self-Join Size: In the case of the self-join size estimator, $E\left[X_k X_l\right] = C^2 \cdot \sum_{i\in I}\sum_{j\in I} E\left[f_i'^2 f_j'^2\right]$, and then, the variance of the average estimator is given by:

$$\begin{aligned} \mathrm{Var}\left[\frac{1}{n}\cdot\sum_{k=1}^{n} X_k\right] = C^2 \cdot \Bigg[&\sum_{i\in I}\sum_{j\in I} E\left[f_i'^2 f_j'^2\right] - \left(\sum_{i\in I} E\left[f_i'^2\right]\right)^2 \\ &+ \frac{2}{n}\left(\sum_{i\in I}\sum_{j\in I} E\left[f_i'^2 f_j'^2\right] - \sum_{i\in I} E\left[f_i'^4\right]\right)\Bigg] \end{aligned} \tag{4–35}$$

Again, the variance of the sketching over samples estimator is the sum of the generic sampling estimator in Equation 4–3 and a term that is related to the variance of the

sketch estimator. Notice that the formulas for the self-join size estimator depend only on the second or the fourth frequency moment of the random variables associated with the sampling frequencies.

4.3.2 Bernoulli Sampling

We instantiate the formulas derived for generic sampling in Section 4.3.1 with the moments of the binomial random variables corresponding to the sampling frequencies. The fact that two binomial random variables corresponding to different elements of the domain are independent further simplify the formulas.

Size of Join: Following the analysis for Bernoulli sampling in Section 4.1.2, we identify the unbiased estimator for the size of join to be:

$$X = \frac{1}{pq} \sum_{i \in I} f'_i \xi_i \cdot \sum_{j \in I} g'_j \xi_j \tag{4–36}$$

with the constant $C = \frac{1}{pq}$. The generic variance in Equation 4–29 takes the following form in the case of Bernoulli sampling:

$$\begin{aligned} &\mathrm{Var}\left[X\right] = \\ &\frac{1}{p^2 q^2} \left[\sum_{i \in I} E\left[f_i'^2\right] \sum_{j \in I} E\left[g_j'^2\right] + \left(\sum_{i \in I} E\left[f'_i\right] E\left[g'_i\right] \right)^2 - 2 \cdot \sum_{i \in I} E^2\left[f'_i\right] E^2\left[g'_i\right] \right] \end{aligned} \tag{4–37}$$

since the random variables f'_i and g'_i are independent, and more, the pairs f'_i, f'_j are independent for $i \neq j$, thus the equality $\sum_{i \in I} \sum_{j \in I} E\left[f'_i f'_j\right] E\left[g'_i g'_j\right] = \sum_{i \in I} E\left[f_i'^2\right] E\left[g_i'^2\right] + \left(\sum_{i \in I} E\left[f'_i\right] E\left[g'_i\right]\right)^2 - \sum_{i \in I} E^2\left[f'_i\right] E^2\left[g'_i\right]$ holds. The generic variance of the average estimator in Equation 4–34 can be instantiated in a similar way:

$$\begin{aligned} \mathrm{Var}\left[\frac{1}{n} \cdot \sum_{k=1}^{n} X_k\right] = \frac{1}{p^2 q^2} \Bigg[&\sum_{i \in I} E\left[f_i'^2\right] E\left[g_i'^2\right] - \sum_{i \in I} E^2\left[f'_i\right] E^2\left[g'_i\right] \\ &+ \frac{1}{n} \left(\sum_{i \in I} E\left[f_i'^2\right] \sum_{j \in I} E\left[g_j'^2\right] + \left(\sum_{i \in I} E\left[f'_i\right] E\left[g'_i\right] \right)^2 - \sum_{i \in I} E\left[f_i'^2\right] E\left[g_i'^2\right] - \sum_{i \in I} E^2\left[f'_i\right] E^2\left[g'_i\right] \right) \Bigg] \end{aligned} \tag{4–38}$$

After we plug in the second frequency moment of the binomial random variables, we obtain:

$$\begin{aligned}\text{Var}\left[\frac{1}{n}\cdot\sum_{k=1}^{n}X_k\right] &= \frac{1}{n}\left[\sum_{i\in I}f_i^2\sum_{j\in I}g_j^2+\left(\sum_{i\in I}f_ig_i\right)^2-2\sum_{i\in I}f_i^2g_i^2\right]\\ &+\frac{n-1}{n}\left[\frac{1-p}{p}\sum_{i\in I}f_ig_i^2+\frac{1-q}{q}\sum_{i\in I}f_i^2g_i+\frac{(1-p)(1-q)}{pq}\sum_{i\in I}f_ig_i\right]\\ &+\frac{1}{n}\left[\frac{1-p}{p}\sum_{i\in I}f_i\sum_{j\in I}g_j^2+\frac{1-q}{q}\sum_{i\in I}f_i^2\sum_{j\in I}g_j+\frac{(1-p)(1-q)}{pq}\sum_{i\in I}f_i\sum_{j\in I}g_j\right]\end{aligned} \tag{4–39}$$

which is exactly the sum of the average sketch estimator individual variance (Equation 4–17), the Bernoulli sampling estimator individual variance (Equation 4–7), and an interaction term. The dominant term seems to be the variance of the sketch estimator due to the product of the sums of the squares of the frequencies. The interaction term also contains products of sums which can become quite large (and comparable) with the sketch variance in some particular scenarios. In the empirical evaluation section (Section 4.4) we provide a study that shows what is the relative significance of each of the three terms as a function of the data parameters.

Self-Join Size: The unbiased estimator for Bernoulli sampling is defined similarly to Equation 4–8:

$$X=\frac{1}{p^2}\left(\sum_{i\in I}f_i'\xi_i\right)^2-\frac{1-p}{p}\sum_{i\in I}f_i \tag{4–40}$$

with the scaling factor $C=\frac{1}{p^2}$ and an extra term for bias correction. The extra term does not affect the sketching over samples process, the only modification being that a bias correction is needed when the estimate is actually computed. Precisely, the bias correction only requires knowledge about the size of the base relation. The generic variance in Equation 4–31 is not significantly affected by the modification of the basic estimator since $\text{Var}\left[aX+b\right]=a^2\text{Var}\left[X\right]$ holds for constants a and b. Thus, the variance $\text{Var}\left[X\right]$ is:

$$\text{Var}\left[X\right]=\frac{1}{p^4}\left[\sum_{i\in I}E\left[f_i'^4\right]+2\left(\sum_{i\in I}E\left[f_i'^2\right]\right)^2-3\sum_{i\in I}E^2\left[f_i'^2\right]\right] \tag{4–41}$$

We used the equality $\sum_{i\in I}\sum_{j\in I} E\left[f_i'^2 f_j'^2\right] = \sum_{i\in I} E\left[f_i'^4\right] + \left(\sum_{i\in I} E\left[f_i'^2\right]\right)^2 - \sum_{i\in I} E^2\left[f_i'^2\right]$ for simplifications. The variance of the average estimator can be derived in a similar manner:

$$\begin{aligned}&\mathrm{Var}\left[\frac{1}{n}\cdot\sum_{k=1}^{n} X_k\right] = \\ &\frac{1}{p^4}\left[\sum_{i\in I} E\left[f_i'^4\right] - \sum_{i\in I} E^2\left[f_i'^2\right] + \frac{2}{n}\left(\left(\sum_{i\in I} E\left[f_i'^2\right]\right)^2 - \sum_{i\in I} E^2\left[f_i'^2\right]\right)\right]\end{aligned} \tag{4–42}$$

In order to obtain the exact formula, the second and the fourth frequency moments of the binomial random variable f_i' have to be plugged in. After simplification, we obtain the final formula for the average estimator variance:

$$\begin{aligned}\mathrm{Var}\left[\frac{1}{n}\cdot\sum_{k=1}^{n} X_k\right] &= \frac{2}{n}\left[\left(\sum_{i\in I} f_i^2\right)^2 - \sum_{i\in I} f_i^4\right] + \frac{4(1-p)}{p}\sum_{i\in I} f_i^3 \\ &+ \frac{2(1-p)(3-5p)}{p^2}\sum_{i\in I} f_i^2 + \frac{(1-p)(6p^2-6p+1)}{p^3}\sum_{i\in I} f_i \\ &+ \frac{2}{n}\left[\frac{(1-p)^2}{p^2}\sum_{i\in I}\sum_{j\in I, j\neq i} f_i f_j + \frac{2(1-p)}{p}\sum_{i\in I}\sum_{j\in I, j\neq i} f_i^2 f_j\right]\end{aligned} \tag{4–43}$$

which is, as expected, the sum of the sketch average estimator variance, the variance of the sampling estimator, and an interaction term.

4.3.3 Sampling with Replacement

In a similar way to Bernoulli sampling, we instantiate the formulas derived for generic sampling in Section 4.3.1 with the moments of the multinomial random variables corresponding to the sampling frequencies. Although the components of the multinomial random variable are binomial random variables, the formulas for sampling with replacement are more complicated due to the interaction between two different binomial random variables.

Size of Join: Following the analysis for sampling with replacement in Section 4.1.3, we identify the unbiased estimator for the size of join to be:

$$X = \frac{|F|}{|F'|}\frac{|G|}{|G'|} \cdot \sum_{i \in I} f'_i \xi_i \cdot \sum_{j \in I} g'_j \xi_j \tag{4–44}$$

with $C = \frac{|F|}{|F'|}\frac{|G|}{|G'|}$. The generic variance in Equation 4–29 takes the following form in the case of sampling with replacement:

$$\text{Var}\left[X\right] = \frac{|F|^2}{|F'|^2}\frac{|G|^2}{|G'|^2} \cdot \left[\sum_{i \in I} E\left[f'^2_i\right] \sum_{j \in I} E\left[g'^2_j\right] - \left(\sum_{i \in I} E\left[f'_i\right] E\left[g'_i\right]\right)^2 + 2 \cdot \sum_{i \in I} \sum_{j \in I, j \neq i} E\left[f'_i f'_j\right] E\left[g'_i g'_j\right]\right] \tag{4–45}$$

since the random variables f'_i and g'_i are independent and $\sum_{i \in I} \sum_{j \in I} E\left[f'_i f'_j\right] E\left[g'_i g'_j\right] = \sum_{i \in I} E\left[f'^2_i\right] E\left[g'^2_i\right] + \sum_{i \in I} \sum_{j \in I, j \neq i} E\left[f'_i f'_j\right] E\left[g'_i g'_j\right]$ holds. The generic variance of the average estimator in Equation 4–34 can be derived in a similar way. After simplification, we obtain:

$$\begin{aligned}
\text{Var}\left[\frac{1}{n} \cdot \sum_{k=1}^{n} X_k\right] &= \frac{\left(|F'|-1\right)\left(|G'|-1\right) - |F'||G'|}{|F'||G'|}\left(\sum_{i \in I} f_i g_i\right)^2 \\
&+ \frac{|F|}{|F'|}\frac{|G|}{|G'|}\sum_{i \in I} f_i g_i + |F|\frac{|G'|-1}{|G'|}\sum_{i \in I} f_i g_i^2 + \frac{|F'|-1}{|F'|}|G|\sum_{i \in I} f_i^2 g_i \\
&+ \frac{1}{n}\frac{\left(|F'|-1\right)\left(|G'|-1\right)}{|F'||G'|}\left[\sum_{i \in I} f_i^2 \sum_{j \in I} g_j^2 + \left(\sum_{i \in I} f_i g_i\right)^2 - 2\sum_{i \in I} f_i^2 g_i^2\right] \\
&+ \frac{1}{n}\left[\frac{|F|}{|F'|}\frac{|G|}{|G'|}\sum_{i \in I}\sum_{j \in I, j \neq i} f_i g_j + |F|\frac{|G'|-1}{|G'|}\sum_{i \in I}\sum_{j \in I, j \neq i} f_i g_j^2 + \frac{|F'|-1}{|F'|}|G|\sum_{i \in I}\sum_{j \in I, j \neq i} f_i^2 g_j\right]
\end{aligned} \tag{4–46}$$

which is exactly the sum of the sampling with replacement estimator individual variance (Equation 4–12), the average sketch estimator individual variance (Equation 4–17), and an interaction term.

Self-Join Size: For sampling with replacement, the unbiased estimator for the self-join size is defined as follows:

$$X = \frac{|F|^2}{|F'|\left(|F'|-1\right)}\left(\sum_{i\in I} f'_i\xi_i\right)^2 - \frac{|F|^2}{|F'|-1} \tag{4–47}$$

with $C = \frac{|F|^2}{|F'|(|F'|-1)}$. The variance can be computed in a similar way as for Bernoulli sampling since the components of a multinomial random variable are binomial random variables, thus they have the same frequency moments. Expectations of the form $E\left[f_i'^2 f_j'^2\right]$ have the same form as in Equation 4–14. We do not provide the formulas for the variance, but essentially the variance of the average estimator can still be written as the sum of the sketch average estimator variance, the variance of the sampling estimator, and an interaction term.

4.3.4 Discussion

The estimators for sketches over samples are defined almost as the basic sketching estimators. The only difference is a constant scaling factor that needs to be added to compensate for the difference in size. In the case of self-join size, an extra constant term is required to compensate for the bias. The variance of the combined estimator can be written as the sum of the sketch estimator, the sampling estimator, and an interaction term. This immediately gives an intuitive interpretation to the formulas we derived. Although the sketch variance seems to be the dominant term, the exact significance of each of the terms is dependent on the actual distribution of the data (see the experimental results in Section 4.4). When multiple sketch estimators are averaged in order to decrease the variance, the covariance also has to be considered since the sketch estimators are computed over the same sample, thus they are not completely independent. Our results show that the variance of the combined estimator does not decrease by a factor equal to the number of averages anymore, just the sketch term does.

4.4 Experimental Evaluation

We pursue three main goals in the experimental evaluation of the sketching over samples estimators. First, we want to determine what is the relative contribution of each of the three terms that appear in the variance of the average estimator. More precisely, we want to determine if the interaction term represents a significant amount from the variance. Second, we want to determine the behavior of the error of the sketch over samples estimator when compared with the error of the sketch estimator. And third, we want to identify what is the behavior of the estimation error as a function of the sample size.

In order to accomplish these goals, we designed a series of experiments over synthetic datasets. This allows a better control of the important parameters that affect the results. The datasets used in our experiments contain either 10 or 100 million tuples generated from a Zipfian distribution with the coefficient ranging between 0 (uniform) and 5 (skewed). The domain of the possible values is 1 million. In the case of size of join, the tuples in the two relations are generated completely independent. We used F-AGMS sketches [14] in all of the experiments due to their superior performance both in accuracy and update time (see [52] for details on sketching techniques). The number of buckets is either $5,000$ or $10,000$. This is equivalent to averaging $5,000$ or $10,000$ basic estimators. In order to be statistically significant, all the results presented in this section are the average of at least 100 independent experiments.

Figure 4-1 and 4-2 depict the relative contribution of each of the three terms appearing in the variance of the average estimator over Bernoulli samples (Equation 4–43 and 4–39). The relative contribution is represented as a function of the data skew for different sampling probabilities. A common trend both for size of join and self-join size is that the interaction term is highly significant for low skew data. This completely justifies the analysis we develop throughout the chapter since an analysis assuming that the variance of the composed estimator is the sum of the variance of the basic estimators

would be incorrect. As expected, the impact of the variance of the sampling estimator is more significant as the size of the sample is smaller. For self-join size (Figure 4-2), the variance is dominated by the term corresponding to the sampling estimator, while for size of join (Figure 4-1) the variance of the sketch estimator quantifies for almost the entire variance irrespective of the sampling probability. This is entirely supported by the existing theoretical results which show that sketches are optimal for estimating the second frequency moment while sampling is optimal for the estimation of size of join [4].

The experimental relative error, i.e., $\frac{|\text{estimation}-\text{true result}|}{\text{true result}}$, of the sketch over Bernoulli samples estimator is depicted in Figure 4-3 and 4-4 as a function of the data skew for different sampling probabilities. Probability $p = 1.0$ corresponds to sketching the entire dataset. These experimental results show that, with some exceptions, the sampling rate does not significantly affect the accuracy of the sketch estimator. For Zipf coefficients larger than 1, in the case of self-join size, and smaller than 3, in the case of size of join, the error of the sketch estimator is almost the same both when the entire dataset is sketched or when only one tuple out of a thousand is sketched. The impact of the sampling rate is significant only for low skew data in the case of self-join size. This is to be expected from the theoretical analysis since the interaction term dominates the variance. What cannot be explained from the theoretical analysis is the effect of the sampling rate for skewed data in the case of size of join. As shown in [52], the experimental behavior of F-AGMS sketches is in some cases orders of magnitude better than the theoretical predictions, thus although the theoretical variance is dominated by the variance of the sketch estimator, the empirical absolute value is small when compared to the variance of the sampling estimator. In the light of [52], the empirical results for high sampling rates are much better than the theoretical predictions, increasing the significance of the sampling rate for skewed data.

In Figure 4-5 and 4-6 we depict the experimental relative error as a function of the sample size for sampling with replacement. Since the actual size of the sample is different for different Zipf coefficients, we represent on the x axis the size of the sample as a fraction

from the population size, with 1 corresponding to a sample with replacement of size equal to the population size. As expected, the error is decreasing as the sample size becomes larger, but it stabilizes after a certain sample size (a 0.1 fraction of the population size for the included figures). This is due entirely to the error of the sketch estimator which exists even when the entire population is available.

In conclusion, the experimental evaluation section supports the need for the detailed analysis of the sketch over samples estimators in Section 4.3. Our experiments show that a significant speed-up (a factor of 10 in general and a factor of up to 1000 in some cases) can be obtained by sketching only a small sample of the data instead of the entire data. At the same time, when the data is a sample (with replacement) from a large population, properties of the entire population can be accurately inferred by sketching only a small sample (a fraction of 0.1 or less from the population size).

4.5 Conclusions

In this chapter we provide the moment analysis of the sketch over samples estimators for two types of sampling: Bernoulli and sampling with replacement. Sketching Bernoulli samples is essentially a load shedding technique for sketching data streams which results, as our theory and experiments suggest, in significant update time reduction – by as much as a factor of 100 – with minimal accuracy degradation. Sketching samples with replacement from an unknown distribution allows efficient characterization of the unknown distribution which has many applications to online data-mining.

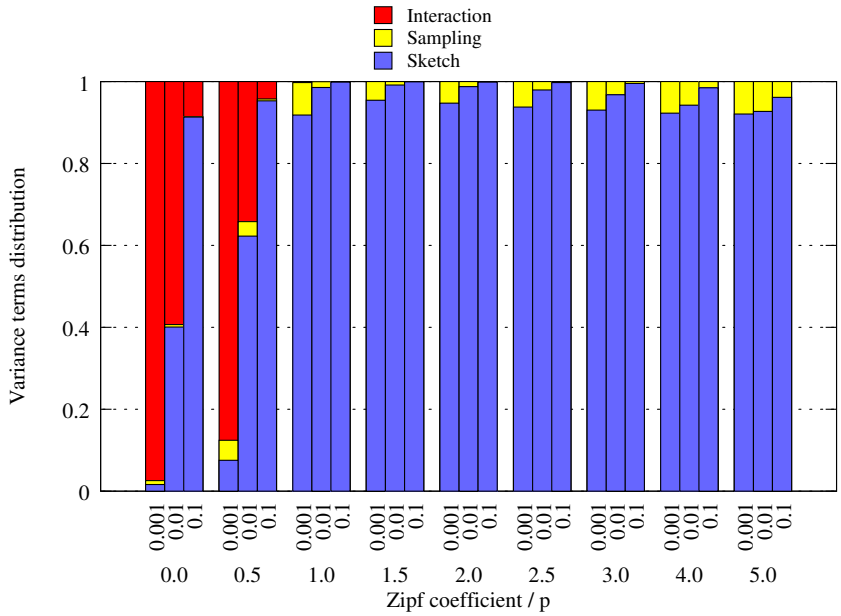


Figure 4-1. Size of join variance.

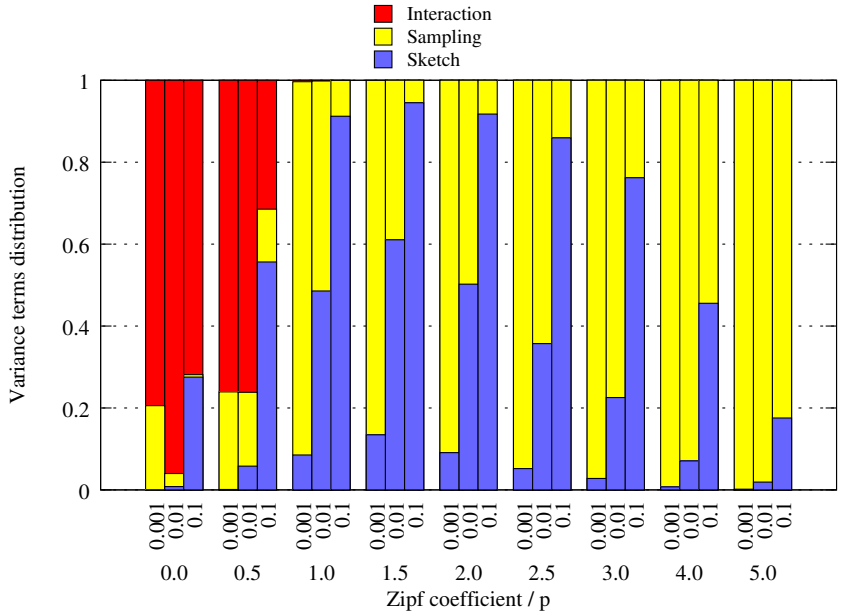


Figure 4-2. Self-join size variance.

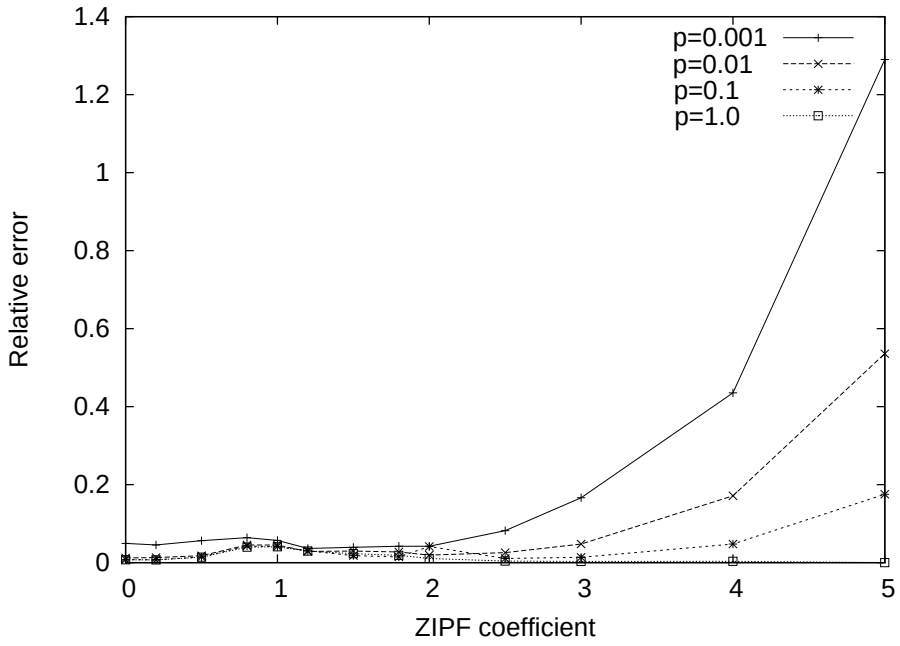


Figure 4-3. Size of join error.

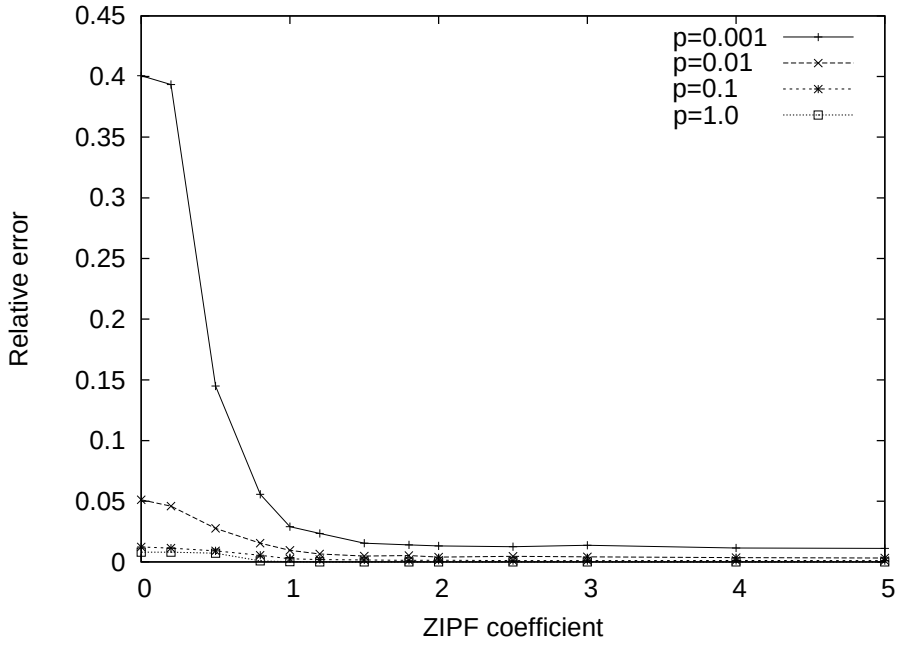


Figure 4-4. Self-join size error.

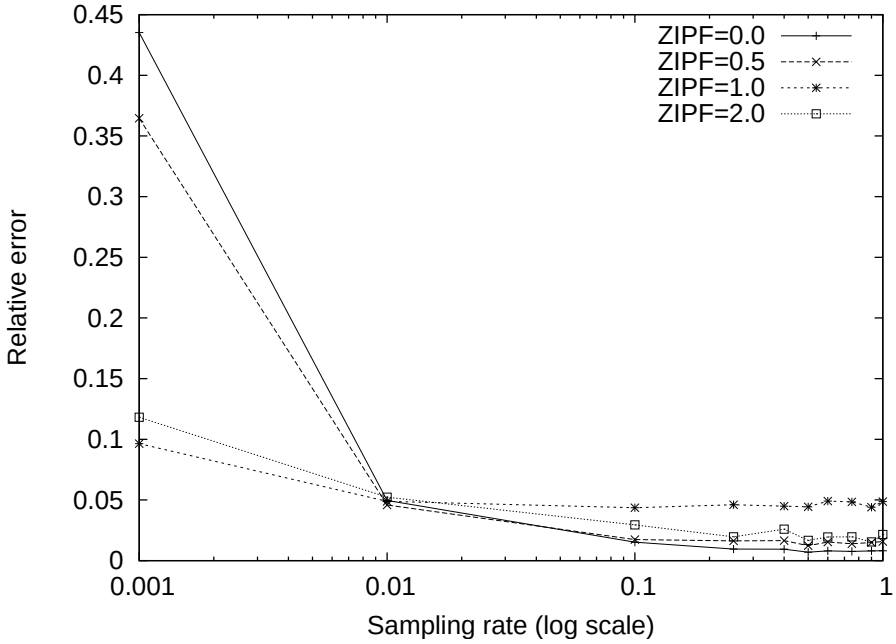


Figure 4-5. Size of join sample size.

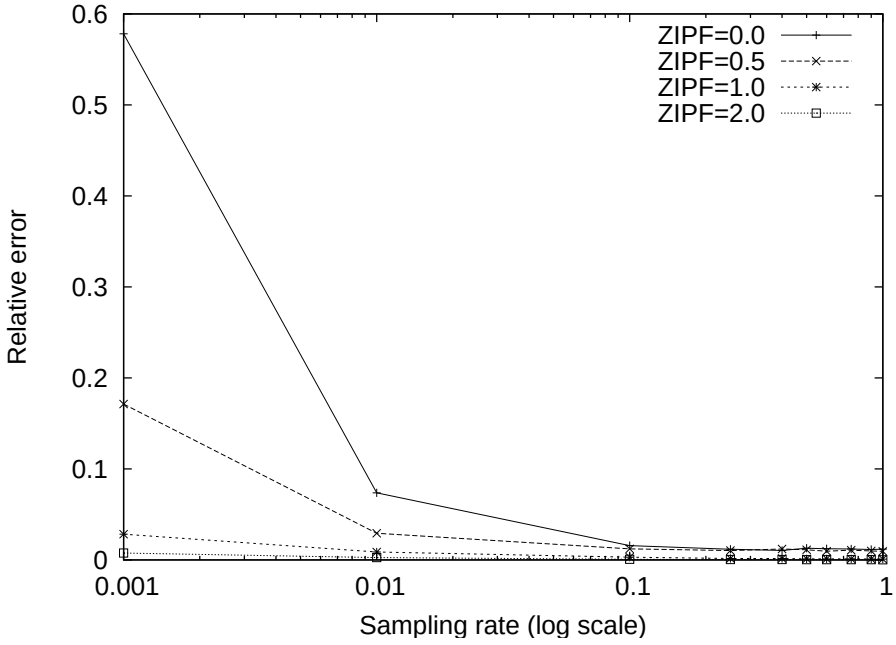


Figure 4-6. Self-join size sample size.

CHAPTER 5
STATISTICAL ANALYSIS OF SKETCHES

Sketching techniques provide approximate answers to aggregate queries both for data-streaming and distributed computation. Small space summaries that have linearity properties are required for both types of applications. The prevalent method for analyzing sketches uses moment analysis and distribution independent bounds based on moments. This method produces clean, easy to interpret, theoretical bounds that are especially useful for deriving asymptotic results. However, the theoretical bounds obscure fine details of the behavior of various sketches and they are mostly not indicative of which type of sketches should be used in practice. Moreover, no significant empirical comparison between various sketching techniques has been published, which makes the choice even harder.

In this chapter, we take a close look at the sketching techniques proposed in the literature from a statistical point of view with the goal of determining properties that indicate the actual behavior and producing tighter confidence bounds. Interestingly, the statistical analysis reveals that two of the techniques, Fast-AGMS and Count-Min, provide results that are in some cases orders of magnitude better than the corresponding theoretical predictions. We conduct an extensive empirical study that compares the different sketching techniques in order to corroborate the statistical analysis with the conclusions we draw from it. The study indicates the expected performance of various sketches, which is crucial if the techniques are to be used by practitioners. The overall conclusion of the study is that Fast-AGMS sketches are, for the full spectrum of problems, either the best, or close to the best, sketching technique.

Through research in the last decade, sketching techniques evolved as the premier approximation technique for aggregate queries over data streams. All sketching techniques share one common feature: they are based on randomized algorithms that combine random seeds with data to produce random variables that have distributions connected to the true value of the aggregate being estimated. By measuring certain characteristics

of the distribution, correct estimates of the aggregate are obtained. The interesting thing about all sketching techniques that have been proposed is that the combination of randomization and data is a linear operation with the result that, as observed in [14, 41], sketching techniques can be used to perform distributed computation of aggregates without the need to send the actual data values. The tight connection with both data-streaming and distributed computation makes sketching techniques important from both the theoretical and practical point of view.

Sketches can be used either as the actual approximation technique, in which case they require a single pass over the data or, in order to improve the performance, as the basic technique in multi-pass techniques such as *skimmed sketches* [28] and *red-sketches* [29]. For either application, it is important to understand as well as possible its approximation behavior depending on the characteristics of the problem and to be able to predict as accurately as possible the estimation error. As opposed to most approximation techniques—one of the few exceptions are sampling techniques [35]—theoretical approximation guarantees in the form of confidence bounds were provided for all types of sketches from the beginning [3]. All the theoretical guarantees that we know of are expressed as memory and update time requirements in terms of big-$\mathcal{O}$ notation, and are parameterized by ϵ, the target relative error, δ, the target confidence (the relative error is at most ϵ with probability at least $1 - \delta$), and the characteristics of the data – usually the first and the second frequency moments. While these types of theoretical results are useful in theoretical computer science, the fear is that they might hide details that are relevant in practice. In particular, it might be hard to compare methods, or some methods can look equally good according to the theoretical characterization, but differ substantially in practice. An even more significant concern, which we show to be perfectly justified, is that some of the theoretical bounds are too conservative.

In this chapter, we set out to perform a detailed study of the statistical and empirical behavior of the four basic sketching techniques that have been proposed in the research literature for computing size of join and related problems: AGMS [3, 4], Fast-AGMS [14], Count-Min [15], and Fast-Count [58] sketches. The initial goal of the study was to complement the theoretical results and to make sketching techniques accessible and useful for the practitioners. While accomplishing these tasks, the study also shows that, in general, the theoretical bounds are conservative by at least a constant factor of 3. For Fast-AGMS and Count-Min sketches, the study shows that the theoretical prediction is off by many orders of magnitude if the data is skewed. As part of our study we provide practical confidence intervals for all sketches except Count-Min. We use statistical techniques to provide confidence bounds at the same time the estimate is produced without any prior knowledge about the distribution[1] . Notice that prior knowledge is required in order to use the theoretical confidence bounds provided in the literature and might not actually be available in practice. As far as we know, there does not exist any detailed statistical study of sketching techniques and only limited empirical studies to asses their accuracy. The insight we get from the statistical analysis and the extensive empirical study we perform allows us to clearly show that, from a practical point of view, Fast-AGMS sketches are the best basic sketching technique. The behavior of these sketches is truly exceptional and much better than previously believed – the exceptional behavior is masked by the result in [14], but revealed by our detailed statistical analysis. The timing results for the three hash-based sketching techniques (Fast-AGMS, Fast-Count, and Count-Min) reveal that sketches are practical, easily able to keep up with streams of million tuples/second.

The rest of the chapter is organized as follows. In Section 5.1 we give an overview of the four basic sketching techniques proposed in the literature. Section 5.2 contains our

[1] This is the common practice for sampling estimators [35].

statistical analysis of the four sketching techniques with insights on their behavior. Section 5.3 contains the details and results of our extensive empirical study that corroborates the statistical analysis.

5.1 Sketches

Sketches are small-space summaries of data suited for massive, rapid-rate data streams processed either in a centralized or distributed environment. Queries are not answered precisely anymore, but rather approximately, by considering only the synopsis (sketch) of the data. All sketching techniques generate multiple random instances of an elementary sketch estimator, instances that are effectively random samples of the elementary estimator. While the random instances of the elementary sketch estimator are samples of the estimator, these samples should not be confused with the samples from the underlying data used by the sampling techniques [35]. The samples of the elementary sketch estimator are grouped and combined in different ways in order to obtain the overall estimate. Typically, in order to produce an elementary sketch estimator—the process is duplicated for each sample of the elementary sketch estimator—multiple counters corresponding to *random variables* with required properties are maintained. The collection of counters for all the samples of the elementary estimator is called the *sketch*. The existing sketching techniques differ in how the random variables are organized, thus the update procedure, the way the elementary sketch estimator is computed, and how the answer to a given query is computed by combining the elementary sketch estimators.
In this section we provide an overview of the existing sketching techniques used for approximating the size of join of two data streams (see Section 2.1). For each technique we specify the elementary sketch estimator, denoted by X possibly with a subscript indicating the type of sketch, and the way the elementary sketches are combined to obtain the final estimate Z.

5.1.1 Basic AGMS Sketches

The i^{th} entry of the size n AGMS (or, *tug-of-war*) [3, 4] sketch vector is defined as the random variable $x_f[i] = \sum_{j=0}^{N-1} f_j \cdot \xi_i(j)$, where $\{\xi_i(j) : j \in I\}$ is a family of uniformly distributed ± 1 4-wise independent random variables, with different families being independent. The advantage of using ± 1 random variables comes from the fact that they can be efficiently generated in small space [51]. When a new data stream item (e, w) arrives, all the counters in the sketch vector are updated as $x_f[i] = x_f[i] + w \cdot \xi_i(e)$, $1 \leq i \leq n$. The time to process an update is thus proportional with the size of the sketch vector.

It can be shown that $X[i] = x_f[i] \cdot x_g[i]$ is an unbiased estimator of the inner-product of the frequency vectors $\bar{f}$ and $\bar{g}$, i.e., $E[X[i]] = \bar{f} \odot \bar{g}$. The variance of the estimator is:

$$\text{Var}[X[i]] = \left(\sum_{j \in I} f_j^2\right)\left(\sum_{k \in I} g_k^2\right) + \left(\sum_{j \in I} f_j g_j\right)^2 - 2 \cdot \sum_{j \in I} f_j^2 g_j^2 \tag{5–1}$$

By averaging n independent estimators, $Y = \frac{1}{n}\sum_{i=1}^{n} X[i]$, the variance can be reduced by a factor of n, i.e., $\text{Var}[Y] = \frac{\text{Var}[X[i]]}{n}$, thus improving the estimation error. In order to make the estimation more stable, the original solution [3] returned as the result the median of m Y estimators, i.e., $Z = Median_{1 \leq k \leq m} Y[k]$. We provide an example to illustrate how AGMS sketches work.

Example 6. *Consider two data streams F and G given as pairs (key, frequency):*

$$\begin{aligned} F &= \{(1,5),(4,-2),(1,2),(2,3),(3,1),(1,-3),(3,2),(5,2),(4,3)\} \\ G &= \{(2,1),(4,3),(3,2),(1,3),(3,-2),(1,2),(5,-1),(1,2),(4,-1)\} \end{aligned} \tag{5–2}$$

We want to estimate the size of join $|F \bowtie G|$ of the two streams using sketches consisting of 3 counters. A family of 4-wise independent ± 1 random variables corresponds to each counter. Let the mappings from the key domain ($\{1,2,3,4,5\}$ in this case) to ± 1 to be given as in Table 5-1.

As the data is streaming by, all the counters in the corresponding sketch vector are updated. For example, the pair $(1, 5)$ *in* F *updates the counters in* x_f *as follows:* $x_f[1] = 5$, $x_f[2] = 5$, *and* $x_f[3] = -5$ *(the counters are initialized to* 0*), while after the pair* $(4, -2)$ *is processed the counters have the following values:* $x_f[1] = 7$, $x_f[2] = 3$, *and* $x_f[3] = -7$. *After all the elements in the two streams passed by, the two sketch vectors are:* $x_f = [7, 1, -5]$ *and* $x_g = [7, 7, -3]$. *The estimator* X *for the size of join* $|F \bowtie G|$ *consists of the values* $X = [49, 7, 15]$ *having the mean* $Y = 23.66$. *The correct result is* 31. *Multiple instances of* Y *can be obtained if other groups of* 3 *counters and their associated families of* ± 1 *random variables are added to the sketch. In this case the median of the instances of* Y *is returned as the final result.*

Notice the tradeoffs involved by the AGMS sketch structure. In order to decrease the error of the estimator (proportional with the variance), the size n of the sketch vector has to be increased. Since the space and the update-time are linear functions of n, an increase of the sketch size implies a corresponding increase of these two quantities.

The following theorem relates the accuracy of the estimator with the size of the sketch, i.e., $n = \mathcal{O}(\frac{1}{\epsilon^2})$ and $m = \mathcal{O}(\log \frac{1}{\delta})$.

Theorem 11 ([4])**.** *Let* $\bar{x}_f$ *and* $\bar{x}_g$ *denote two parallel sketches comprising* $\mathcal{O}\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$ *counters each, where* ϵ *and* $1 - \delta$ *represent the desired bounds on error and probabilistic confidence, respectively. Then, with probability at least* $1 - \delta$, $Z \in (\bar{f} \odot \bar{g} \pm \epsilon ||\bar{f}||_2 ||\bar{g}||_2)$. *The processing time required to maintain each sketch is* $\mathcal{O}\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$ *per update.*

$||\bar{f}||_2 = \sqrt{\bar{f} \odot \bar{f}} = \sqrt{\sum_{i \in I} f_i^2}$ is the L_2 norm of $\bar{f}$ and $||\bar{g}||_2 = \sqrt{\bar{g} \odot \bar{g}} = \sqrt{\sum_{i \in I} g_i^2}$ is the L_2 norm of $\bar{g}$, respectively. From the perspective of the abstract problem in Section 2.3, $X[i]$ represent the primitive instances of the generic random variable X. Median of means Z is the estimator for the expected value $E[X]$. The distribution-independent confidence bounds in Theorem 11 are derived from Theorem 8.

5.1.2 Fast-AGMS Sketches

As we have already mentioned, the main drawback of AGMS sketches is that any update on the stream affects all the entries in the sketch vector. Fast-AGMS sketches [14], as a refinement of Count sketches proposed in [11] for detecting the most frequent items in a data stream, combine the power of ± 1 random variables and hashing to create a scheme with a significantly reduced update time while preserving the error bounds of AGMS sketches. The sketch vector $\bar{x}_f$ consists of n counters, $x_f[i]$. Two independent random processes are associated with the sketch vector: a family of ± 1 4-wise independent random variables ξ and a 2-universal hash function $h : I \rightarrow \{1, \ldots, n\}$. The role of the hash function is to scatter the keys in the data stream to different counters in the sketch vector, thus reducing the interaction between the keys. Meanwhile, the unique family ξ preserves the dependencies across the counters. When a new data stream item (e, w) arrives, only the counter $x_f[h(e)]$ is updated with the value of the function ξ corresponding to the key e, i.e., $x_f[h(e)] = x_f[h(e)] + w \cdot \xi(e)$.

Given two parallel sketch vectors $\bar{x}_f$ and $\bar{x}_g$ using the same hash function h and ξ family, the inner-product $\bar{f} \odot \bar{g}$ is estimated by $Y = \sum_{i=1}^{n} x_f[i] \cdot x_g[i]$. The final estimator Z is computed as the median of m independent basic estimators Y, i.e., $Z = Median_{1 \leq k \leq m} Y[k]$. In the light of Section 2.3, Y corresponds to the basic instances while the median is the estimator for the expected value. We provide a simple example to illustrate the Fast-AGMS sketch data structure.

Example 7. *Consider the same data streams from Example 6. The sketch vector consists of* 9 *counters grouped into* 3 *rows of* 3 *counters each. The same families of* ± 1 *random variables (Example 6) are used, but a family corresponds to a row of counters instead of only one counter. An additional family of* 2*-universal hash functions corresponding to the rows of the sketch maps the elements in the key domain to only one counter in each row. The hash functions are specified in Table 5-2.*

For each stream element only one counter from each row is updated. For example, after the pair $(2, 3)$ *in* F *is processed, the sketch vector* x_f *looks like:* $x_f[1] = [10, 2, 0]$, $x_f[2] = [5, 0, -3]$, *and* $x_f[3] = [0, 3, -9]$ *(the counters are initialized to* 0*). After processing all the elements in the two streams, the two sketch vectors are:* $x_f = [[7, -1, 1], [5, -1, -3], [-2, 3, -6]]$ *and* $x_g = [[8, -2, 1], [9, -1, -1], [1, 1, -5]]$. *The estimator for the size of join* $|F \bowtie G|$ *consists of the median of* $Y = [59, 49, 31]$ *which is* 49, *while the correct result is still* 31.

The following theorem relates the number of sketch vectors m and their size n with the error bound ϵ and the probabilistic confidence δ, respectively.

Theorem 12 ([14]). *Let* n *be defined as* $n = \mathcal{O}(\frac{1}{\epsilon^2})$ *and* m *as* $m = \mathcal{O}(\log \frac{1}{\delta})$. *Then, with probability at least* $1 - \delta$, $Z \in (\bar{f} \odot \bar{g} \pm \epsilon ||\bar{f}||_2 ||\bar{g}||_2)$. *Sketch updates are performed in* $\mathcal{O}(\log \frac{1}{\delta})$ *time.*

The above theorem states that Fast-AGMS sketches provide the same guarantees as basic AGMS sketches, while requiring only $\mathcal{O}(\log \frac{1}{\delta})$ time to process the updates and using only one ξ family per sketch vector (and one additional hash function h). Moreover, notice that only the sketch vector size is dependent on the error ϵ.

5.1.3 Fast-Count Sketches

Fast-Count sketches, introduced in [58], provide the error guarantees and the update time of Fast-AGMS sketches, while requiring only one underlying random process – hashing. The tradeoffs involved are the size of the sketch vector (or, equivalently, the error) and the degree of independence of the hash function. The sketch vector consists of the same n counters as for AGMS sketches. The difference is that there exists only a 4-universal hash function associated with the sketch vector. When a new data item (e, w) arrives, w is directly added to a single counter, i.e., $x_f[h(e)] = x_f[h(e)] + w$, where $h : I \rightarrow \{1, \ldots, n\}$ is the 4-universal hash function.

The size of join estimator is defined as (this is a generalization of the second frequency moment estimator in [58]):

$$Y = \frac{1}{n-1}\left[n \cdot \sum_{i=1}^{n} x_f[i] \cdot x_g[i] - \left(\sum_{i=1}^{n} x_f[i]\right)\left(\sum_{i=1}^{n} x_g[i]\right)\right] \tag{5–3}$$

The complicated form of Y is due to the bias of the natural estimator $Y\prime = \sum_{i=1}^{n} x_f[i] \cdot x_g[i]$. Y is obtained by a simple correction of the bias of $Y\prime$. It can be proved that Y is an unbiased estimator of the inner-product $\bar{f} \odot \bar{g}$. Its variance is almost identical to the variance of the Y estimator for AGMS (Fast-AGMS) sketches in (5–1). The only difference is the multiplicative factor, $\frac{1}{n-1}$ for Fast-Count sketches, compared to $\frac{1}{n}$ for AGMS sketches. Hence, given desirable error guarantees, Fast-Count sketches require one additional entry in the sketch vector. For large values of n, e.g., $n > 100$, the difference in variance between AGMS (Fast-AGMS) and Fast-Count sketches can be ignored and the results in Theorem 12 apply. Notice that in practice multiple instances of Y are computed and the final estimator for the expected value of the size of join is the mean (average) of these instances. We provide an example that shows how Fast-Count sketches work.

Example 8. *Consider the same data streams from Example 6. The sketch vector consists of* 9 *counters grouped into* 3 *rows of* 3 *counters each. The same hash functions as in Example 7 are used (suppose that they are* 4*-universal). For each stream element only one counter from each row is updated. For example, after the pair* $(2, 3)$ *in* F *is processed, the sketch vector* x_f *looks like:* $x_f[1] = [10, -2, 0]$*,* $x_f[2] = [5, 0, 3]$*, and* $x_f[3] = [0, 3, 5]$ *(the counters are initialized to* 0*). After processing all the elements in the two streams, the two sketch vectors are:* $x_f = [[7, 1, 5], [5, 5, 3], [2, 3, 8]]$ *and* $x_g = [[8, 2, -1], [9, -1, 1], [-1, 1, 9]]$*. The estimator for the size of join* $|F \bowtie G|$ *consists of the vector* $Y = [21, 6, 51]$*. The average* 26 *of the elements in* Y *is returned as the final estimate.*

5.1.4 Count-Min Sketches

Count-Min sketches [15] have almost the same structure as Fast-Count sketches. The only difference is that the hash function is drawn randomly from a family of 2-universal

hash functions instead of 4-universal. The update procedure is identical to Fast-Count sketches, only the counter $x_f[h(e)]$ being updated as $x_f[h(e)] = x_f[h(e)] + w$ when the item (e, w) arrives. The size of join estimator is defined in a natural way as $Y = \sum_{i=1}^{n} x_f[i] \cdot x_g[i]$ (notice that Y is actually equivalent with the above $Y\prime$ estimator). It can be shown that Y is an overestimate of the inner-product $\bar{f} \odot \bar{g}$. In order to minimize the over-estimated quantity, the minimum over m independent Y estimators is computed, i.e., $Z = Min_{1 \leq k \leq m} Y[k]$. Notice the different methods applied to correct the bias of the size of join estimator $Y\prime$. While Fast-Count sketches define an unbiased estimator Y based on $Y\prime$, Count-Min sketches select the minimum over multiple such overestimates. The following example illustrates the behavior of Count-Min sketches.

Example 9. *For the same setup as in Example 8, exactly the same sketch vectors are obtained after updating the two streams. Only the final estimator is different. It is the minimum of* $Y = [53, 43, 73]$*, that is* 43*.*

The relationship between the size of the sketch and the accuracy of the estimator Z is expressed by the following theorem:

Theorem 13 ([15])**.** $Z \leq \bar{f} \odot \bar{g} + \epsilon ||\bar{f}||_1 ||\bar{g}||_1$ *with probability* $1 - \delta$*, where the size of the sketch vector is* $n = \mathcal{O}(\frac{1}{\epsilon})$ *and the minimum is taken over* $m = \mathcal{O}(\log \frac{1}{\delta})$ *sketch vectors. Updates are performed in time* $\mathcal{O}(\log \frac{1}{\delta})$*.*

$||\bar{f}||_1 = \sum_{i \in I} f_i$ and $||\bar{g}||_1 = \sum_{i \in I} g_i$ represent the L_1 norms of the vectors $\bar{f}$ and $\bar{g}$, respectively. Notice the dependence on the L_1 norm, compared to the dependence on the L_2 norm for AGMS sketches. The L_2 norm is always smaller than the L_1 norm. In the extreme case of uniform frequency distributions, L_2 is quadratically smaller than L_1. This implies increased errors for Count-Min sketches as compared to AGMS sketches, or, equivalently, more space in order to guarantee the same error bounds (even though the sketch vector size is only $\mathcal{O}(\frac{1}{\epsilon})$).

5.1.5 Comparison

Given the above sketching techniques, we qualitatively compare their expected performance based on the existing theoretical results. The techniques are compared relatively to the result obtained by the use of AGMS sketches for the self-join size problem, known to be asymptotically optimal [3]. The size of join results are considered relatively to the product of the L_2 (L_1 for Count-Min) norms of the data streams. Notice that large results correspond to the particular self-join size problem. Low skew corresponds to frequency vectors for which the ratio $\frac{L_1}{L_2}$ is close to $\sqrt{N}$ (uniform distribution), while for high skew the ratio $\frac{L_1}{L_2}$ is close to 1.

Table 5-3 summarizes the distribution-independent confidence bounds predicted by the theory. Since the bounds for AGMS, Fast-AGMS, and Fast-Count sketches are identical, they have the same behavior from a theoretical perspective. For small size of join results, the performance of these three methods worsens. Count-Min sketches have a distinct behavior due to their dependency on the L_1 norm. Their performance is highly influenced not only by the size of the result, but also by the skewness of the data. For low skew data, the performance is significantly worse than the performance of AGMS sketches. Since $L_1 \geq L_2$, the theoretical performance for Count-Min sketches is always worse than the performance of AGMS (Fast-AGMS, Fast-Count) sketches.

5.2 Statistical Analysis of Sketch Estimators

The goals pursued in refining the sketching techniques were to leverage the randomness and to decrease the update time while maintaining the same error guarantees as for the original AGMS sketches. As we have previously seen, all kinds of tradeoffs are involved. The main drawback of the existing theoretical results is that they characterize only the asymptotic behavior, but do not provide enough details about the behavior of the sketching techniques in practice (they ignore important details about the estimator because they are derived from distribution-independent confidence bounds). From a purely practical point of view, we are interested in sketching techniques that are reasonably easy

to implement, are fast (i.e., small update time for the synopsis data-structure), have good accuracy and can estimate as precisely as possible their error through confidence intervals. Although the same goals are pursued from the theoretical point of view, in theory we insist on deriving simple formulas for the error expressed in terms of asymptotic big-$\mathcal{O}$ notation. This is perfectly reflected by the theoretical results we presented in the previous section. The problem with theoretical results is the fact that, since we always insist on expressible formulas, we might ignore details that matter at least in some cases – the theoretical results are always conservative, but they might be too conservative sometimes. In this section, we explore the sketching techniques from a statistical perspective by asking the following questions that reflect the difference between the pragmatic and the theoretical points of view:

- All sketching techniques combine multiple independent instances of elementary sketches using the estimators from Section 2.3 (Mean, Median, Minimum) in order to define a more accurate estimator for the expected value. We ask the question which of the estimators is more accurate for each of the four sketching techniques?
- How *tight* are the theoretical distribution-independent confidence bounds? And is it possible to derive tighter distribution-dependent confidence bounds that work in practice based on the estimator chosen in the previous question? We are not interested in tight bounds only for some situations, but in confidence bounds that are realistic for all situations. The golden standard we are aiming for is confidence bounds similar to the ones for sampling techniques [35].

We use a large-scale statistical analysis based on experiments in order to answer the above questions. The plots in this section have statistical significance and are not highly sensitive at the experimental setup (Section 4.4).

5.2.1 Basic AGMS Sketches

We explore which estimator—mean, median, or minimum—to use for AGMS sketches instead of the median of means estimator proposed in the original paper [3] and if that would be advisable. In order to accomplish this task, we plotted the distribution of the basic sketch for a large spectrum of problems. Figure 5-1 is a generic example for the form of the distribution. It is clear from this figure that both the minimum and the median

are poor choices. The median is a poor choice because the distribution of the elementary AGMS sketch is not symmetric and there exists a variable gap between the mean of the distribution and the median, gap that is not easily to compute and, thus, to compensate for. In order to verify that the mean is the optimal estimator (as the theory predicts), we plot its distribution for the same input data (Figure 5-1). As expected, the distribution is normal and its expected value is the true result. As explained in Section 2.3.6, the mean is always preferable to the median of means as an estimator for the expected value of a random variable given as samples. This is the case because once averaging over the sample space the distribution of the estimator starts to become normal (Mean CLT) and it is known that the mean is more efficient than the median for normal distributions (Section 2.3.5).

Although the median of means estimator has no statistical significance, it allows the derivation of exponentially decreasing distribution-independent confidence intervals based on Chernoff bound (Theorem 3). To derive tighter distribution-dependent confidence bounds based only on the mean estimator, we can use Theorem 5. The value of the variance is either the exact one (if it can be determined) or, more realistically, an estimate computed from the samples. The distribution-independent confidence bounds in Theorem 8 are wider by a factor of approximately 4 than the CLT bounds, as derived in Example 1. This discrepancy between the distribution-independent bounds and the effective error was observed experimentally in [18, 51], but it was not explained. In conclusion, the mean estimator seems the right choice from a practical perspective considering its advantages over the median of means estimator of which it is anyway a part.

5.2.2 Fast-AGMS Sketches

Comparing Theorem 11 and 12 that characterize AGMS and Fast-AGMS (F-AGMS) sketches, respectively, we observe that the predicted accuracy is identical, but Fast-AGMS have significantly lower update time. This immediately indicates that F-AGMS should

be preferred to AGMS. In the previous section, we saw a discrepancy of a factor of approximately 4 between the distribution-independent bounds and the CLT-based bounds for AGMS sketches and the possibility of a significant improvement if the median of means estimator is replaced by means only. In this section, we investigate the statistical properties of F-AGMS sketches in order to identify the most adequate estimator and to possibly derive tighter distribution-dependent confidence bounds.

We start the investigation on the statistical properties of Fast-AGMS sketches with the following result on the first two frequency moments (expected value and variance) of the basic estimator:

Proposition 3 ([14])**.** *Let X be the Fast-AGMS estimator obtained with a family of 4-universal hash functions $h : I \rightarrow B$ and a 4-wise independent family ξ of ± 1 random variables. Then,*

$$E_{h,\xi}[X] = E[X_{AGMS}]$$
$$E_h[Var_\xi[X]] = \frac{1}{B} Var[X_{AGMS}]$$

The first two moments of the elementary Fast-AGMS sketch coincide with the first two moments of the average of B elementary AGMS sketches (in order to have the same space usage). This is a somewhat unexpected result since it suggests that hashing plays the same role as averaging when it comes to reducing the variance and that the transformation on the distribution of elementary F-AGMS sketches is the same, i.e., the distribution becomes normal and the variance is reduced by a factor equal to the number of buckets. The following result on the fourth frequency moment of F-AGMS represents the first discrepancy between the distributions of Fast-AGMS and AGMS sketches:

Proposition 4. *With the same setup as in Proposition 3, we have:*

$$Var_h[Var_\xi[X]] = \frac{B-1}{B^2}\left[3\left(\sum_i f_i^2 g_i^2\right)^2 + 4\sum_i f_i^3 g_i \sum_j f_j g_j^3 + \right. \\ \left. + \sum_i f_i^4 \sum_j g_j^4 - 8\sum_i f_i^4 g_i^4\right] \tag{5–4}$$

$Var_h[Var_\xi[X]]$ is a lower bound on the fourth moment of the estimator for which we cannot derive a simple closed-form formula because the ξ family is only 4-wise independent and 8-wise independence is required to remove the dependency of the formula on the actual generating scheme. Also, if the hash function h is only 2-universal, instead of 4-universal, more terms are added to the expression, thus making the fourth moment even larger. We use kurtosis (the ratio between the fourth frequency moment and the square of the variance, see Section 2.3.5) to characterize the distribution of the Fast-AGMS basic estimator. From Figure 5-5 that depicts the experimental kurtosis and its lower bound in Proposition 4, we observe that when the Zipf coefficient is larger than 1 the kurtosis grows significantly, to the point that it gets around 1000 for a Zipf coefficient equal to 5. Given these values of the kurtosis, we expect that the distribution of the F-AGMS estimator to be (close to) normal for Zipf coefficients smaller than 1 (kurtosis is equal to 3 for normal distributions, see Section 2.3.5) and then to suffer a drastic change as the Zipf coefficient increases. Large kurtosis is an indicator of distributions that are more concentrated than the normal distribution, but also that have heavier tails [7]. Indeed, Figure 5-2 confirms experimentally these observations for Zipf coefficients equal to 0.2 and 1.5, respectively. The interaction between hashing and the frequent items is an intuitive explanation for the transformation suffered by the F-AGMS distribution as a function of the Zipf coefficient. For low skew data (uniform distribution) there does not exist a significant difference between the way the frequencies are spread into the buckets by the hash function. Although there exists some variation due to the randomness of the hash function, the distribution of the estimator is normally centered on the true value. The

situation is completely different for skewed data which consists of some extremely high frequencies and some other small frequencies. The impact of the hash function is dominant in this case. Whenever the high frequencies are distributed in different buckets (this happens with high probability) the estimation is extremely accurate. When at least two high frequencies are hashed into the same bucket (with small probability) the estimator is orders of magnitude away from the true result. This behavior explains perfectly the shape of the distribution for skewed data: the majority of the mass of the distribution is concentrated on the true result while some small mass is situated far away in the heavy tails. Notice that although large values of kurtosis capture this behavior, an extremely large number of experiments is required to observe the behavior in practice. For example, in Figure 5-5 the experimental kurtosis lies under the lower bound in some cases because the colliding events did not appear even after 10 million experiments.

Given the different shapes of the distribution, no single estimator (mean or median, since minimum is clearly not a valid estimator for the expected value) is always optimal for Fast-AGMS sketches. While mean is optimal for low skew data since the distribution is normal (see Section 2.3.5), median is clearly preferable for skewed data because of the large values of kurtosis. The symmetry condition required for the median to be an estimator for the expected value is satisfied because of the symmetric ± 1 random variables. In the following, we consider median as the estimator for Fast-AGMS sketches even though its error is larger by a factor of 1.25 for low skew data compared to the error of the mean. In order to quantify exactly what is the gain of the median over the mean, we use the concept of efficiency (see Section 2.3.5). Unfortunately, we cannot derive an analytical formula for efficiency because it depends on the value of the probability density function at the true median, which we actually try to determine. The alternative is to estimate empirically the efficiency as a function of kurtosis which, as we have already seen, is a good indicator for the distribution of the F-AGMS estimator and can be computed analytically. Figure 5-6 depicts the experimental efficiency as a function of kurtosis for

sketches with various number of buckets. As expected, efficiency increases as the kurtosis increases, i.e., as the data becomes more skewed, and gets to some extreme values in the order of 10^{10}. While efficiency is independent of the number of buckets in the sketch, the value of kurtosis is limited, with larger values corresponding to a sketch with more buckets. This implies that efficiency is not a simple function of the kurtosis and other parameters of the sketch and the data have also to be considered. Consequently, although we could not quantify exactly what is the gain of using the median instead of the mean, the extremely large values of efficiency clearly indicate that median is the right estimator for F-AGMS sketches.

The distribution-independent confidence bounds given by Theorem 12 are likely to be far too conservative because they are derived from the first two frequency moments using Chebyshev and Chernoff inequalities. These bounds are identical to the bounds for AGMS sketches since the two have the same expected value and variance. The significant discrepancy in the fourth moment and the shape of the distribution (Figure 5-1 and 5-2 depict the distributions for the same data) between F-AGMS and AGMS is not reflected by the distribution-independent confidence bounds. Figure 5-7(a) confirms the huge gap (as much as 10 orders of magnitude) that exists between the distribution-independent bounds and the experimental error. Practical distribution-dependent confidence bounds can be derived from the median bounds in Theorem 7. A comparison between distribution-independent confidence bounds, distribution-dependent confidence bounds and the experimental error (95%) is depicted in Figure 5-7(b). Two important facts can be drawn from these results: first, the distribution-independent bounds are too large for large Zipf coefficients and, second, the median bounds are always accurate. Figure 5-7(a) also reveals that the ratio between the actual error and the prediction is not strongly dependent on the correlation between the data for the same Zipf coefficient. This implies that in order to characterize the behavior of F-AGMS sketches for the size of join problem only the Zipf coefficient of the distribution of the two streams has to be considered.

5.2.3 Count-Min Sketches

Based on Theorem 13, we expect Count-Min (CM) sketches to over-estimate the true value by a factor proportional with the product of the sizes of the two streams and inversely proportional with the number of buckets of the sketch structure. This is the only sketch that has error dependencies on the first frequency moment, not the second frequency moment, and the amount of memory (number of hashing buckets), not the squared root of the amount of memory. While the dependency on the first frequency moment is worse than the dependency on the squared root of the second frequency moment since the first is always larger or equal than the second, the dependency on the amount of memory is favorable to Count-Min sketches. According to the theoretic distribution-independent confidence bounds, we expect Count-Min sketches to have weak performance for relations with low skew, but comparable performance to AGMS sketches (not much better though) for skewed relations. In this section, we take a closer look at the distribution of the basic CM estimator and discuss the methods to derive confidence bounds for Count-Min sketches.

We start the study of the distribution of the elementary CM estimator with the following result that characterizes the frequency moments of the estimator:

Proposition 5. *If X_{CM} is the elementary Count-Min estimator then:*

$$E[X_{CM}] = \sum_{i \in I} f_i g_i + \frac{1}{B}\left(\sum_{i \in I} f_i \sum_{j \in I} g_j - \sum_{i \in I} f_i g_i\right) \tag{5–5}$$

$$Var[X_{CM}] \geq \frac{1}{B} Var[X_{AGMS}] \tag{5–6}$$

Equation 5–5 is proved in [15]. The inequality in Equation 5–6 becomes equality if the hash functions used are 4-universal. For 2-universal hashes, the variance increases depending on the particular generating scheme and no simple formula can be derived. Most of the proof of Equation 5–6 for 4-universal hashes is embedded in the computation of the variance for Fast-Count sketches (see Section 5.2.4), but the exact formula does

not appear in previous work[2] . The expected value of X_{CM} is always an over-estimate for the true result – this is the reason why the minimum estimator is chosen. Interestingly, the variance of the estimator coincides with the variance of averages of B AGMS sketches and the variance of Fast-AGMS sketches. In order to characterize the distribution of CM sketches we conducted an extensive statistical study based on experiments. As for Fast-AGMS sketches, the distribution of the X_{CM} estimator is highly dependent on the skewness of the data and the randomness of hashing. The fundamental difference is that the distribution is not symmetric anymore because ± 1 random variables are not used. The generic shape of the distribution has the majority of the mass concentrated to the left extremity while the right tail is extremely long. The intuition behind this shape lies in the way hashing spreads the data into buckets: with high probability the data is evenly distributed into the buckets (this situation corresponds to the left peak) while with some extremely low probability a large number of items collide into the same bucket (this situation corresponds to the right tail). Although the shape is generic, the position of the left peak (the minimum of the distribution) depends heavily on the actual data. For low skew data the peak is far away from the true value. As the data becomes more skewed the peak starts to translate to the left, to the point it gets to the true value. The movement towards the true value while increasing the Zipf parameter is due to the importance high frequencies start to gain. For low skew data (uniform distribution) the position of the peak is given by the average number of frequencies that are hashed into the same bucket. For skewed data dominated by some high frequencies the peak is situated at the point corresponding to the high frequencies being hashed into different buckets. Since high frequencies dominate the result, the estimate is in this case closer to the true

[2] Since proving the formula is a simple matter of rewriting the equations in [58], we do not provide the proof here.

value. Figure 5-3 depicts the distribution of X_{CM} for Zipf coefficients equal to 1.0 and 2.0, respectively.

The distribution-independent confidence bounds for CM sketches in [15] are derived from the Markov inequality. Essentially, the error bounds are expressed in terms of the expected value of the over-estimated quantity $\frac{1}{B}\left(\sum_{i\in I} f_i \sum_{j\in I} g_j - \sum_{i\in I} f_i g_i\right)$ in $E[X_{CM}]$. Neither the variance nor the bias are considered in deriving these bounds. To verify the accuracy of the confidence bounds, we plot in Figure 5-13 the ratio between the experimental error obtained for data sets with different Zipf and correlation coefficients (see Section 4.4) and the corresponding predicted error. The main observation from these results is that the ratio between the actual error and the prediction decreases as the Zipf coefficient increases, to the point where the gap is many orders of magnitude. In what follows we provide an intuitive explanation for this behavior.

For low skew data the error is almost entirely due to the bias, correctly estimated by the expected value, thus the perfect correspondence between the actual error and the prediction. This observation is inferred from Figure 5-3(a) which plots the distribution of the elementary sketch estimator for Zipf coefficient equal with 1.0. In this situation, the standard deviation of the elementary estimator is much smaller than the bias. If multiple instances of the elementary sketch are obtained, they will all be relatively close to the expected value (no more than a number of standard deviations to the left), thus their minimum will be close to the expected value. The fact that the standard deviation is small when compared to the bias for low skew data can be predicted using Proposition 5 based on the fact that L_2 norm is much smaller than L_1 norm for low skew data.

For high skew, the standard deviation becomes significantly larger than the bias as it can be seen in Figure 5-3(b). In this situation, even though the bias is still significant, with high probability some of the samples of the elementary sketch will be close to the true value, thus the minimum of multiple elementary sketches will have significantly smaller error. Notice how the shape of the distribution changes when the Zipf coefficient

increases: it is normal-like for low skew, but it has no left tail for high skew. The distribution is forced to take this shape when the standard deviation is larger than the bias since CM sketch estimators cannot take values smaller than the true value. Referring back to the moments of the CM elementary estimator in Proposition 5, for large skew the standard deviation is comparable to the expected value, but the bias is much smaller since most of the result is given by the large frequencies whose contribution is accurately captured by the estimator.

While the above discussion gives a good intuition why the theory gives reasonable error predictions for low skew data and makes large errors for high skew data, unfortunately it does not lead to better bounds for skewed data. In order to provide tight confidence bounds, the distribution of the minimum of multiple elementary sketch estimators has to be characterized. While CLT-based results exist for the minimum estimator (see Section 2.3.7), they provide means to characterize the variance, but not the bias of the minimum estimator. Determining the bias of the minimum is crucial for correct predictions of the error for large skew, but it seems a difficult task since it depends on the precise distribution of the data not only some characteristics like the first few moments. It is worth mentioning that tighter bounds for CM sketches can be obtained if the Zipf coefficient of the data is determined by other means [16]. Notice the particular problems of deriving confidence bounds for CM sketches: high errors are correctly predicted while small errors are incorrectly over-estimated. Consequently, Count-Min sketches are difficult to use in practice because their behavior cannot be predicted accurately.

5.2.4 Fast-Count Sketches

Fast-Count (FC) elementary estimator is essentially the bias-corrected version of the Count-Min elementary estimator. The bias correction is a translation by bias and a scaling by the factor $\frac{B}{B-1}$. This can be observed in Figure 5-4 that depicts the distribution of Fast-Count sketches. Everything stated for CM sketch distribution still holds for the distribution of FC sketches, with the major difference that Fast-Count sketches are

unbiased, while Count-Min sketches are biased. Given the unbiased estimator and the asymmetric shape of the distribution, mean is the only viable estimator for the expected value, which is also the true value in this case.

The distribution-independent confidence bounds for FC sketches, derived in a similar manner using Chebyshev and Chernoff bounds, are identical to those for AGMS and Fast-AGMS sketches because the first two moments of the distributions are equal. Tighter distribution-dependent confidence bounds are derived using Mean CLT for AGMS and Median CLT for Fast-AGMS sketches, respectively. Although the mean estimator is also used for FC sketches, the asymptotic regime of Mean CLT does not apply in this case because the number of samples averaged is only in the order of tens. The alternative is to use the Student t-distribution for modeling the behavior of the mean (see Section 2.3.3), but the improvement over the distribution-independent bounds is not so remarkable. In conclusion, both distribution-independent and distribution-dependent bounds can be used for FC sketches without a significant advantage for any of them.

5.3 Empirical Evaluation

The main purpose of the experimental evaluation is to validate and complement the statistical results we obtained in Section 5.2 for the four sketching techniques. The specific goals are: (1) establish the relative accuracy performance of the four sketching techniques for various problems, and (2) determine the actual update performance. Our main tool in establishing the accuracy of sketches is to measure their error on synthetic data sets for which we control both the skew, via the Zipf coefficient, and the correlation. This allows us to efficiently cover a large spectrum of problems and to draw insightful observations about the performance of sketches. We then validate the findings on real-life data sets and other synthetic data generators.

The main findings of the study are:

- AGMS and Fast-Count (FC) sketches have virtually identical accuracy throughout the spectrum of problems if only averages are used for AGMS. FC sketches are preferable since they have significantly smaller update time.

- The performance of Count-Min sketches is strongly dependent on the skew of the data. For small skew, the error is orders of magnitude larger than the error of the other types of sketches. For large skew, CM sketches have the best performance – much better than AGMS and FC.

- Fast-AGMS (F-AGMS) sketches have error at most 25% larger than AGMS sketches for small skew, but the error is orders of magnitude (as much as 6 orders of magnitude for large skew) smaller for moderate and large skew. Their error for large skew is slightly larger than the error of CM sketches.

- All sketches, except CM for small skew, are practical in evaluating self-join size queries. This is to be expected since AGMS sketches are asymptotically optimal [3] for this problem. For size of join problems, F-AGMS sketches remain practical well beyond AGMS and FC sketches. CM sketches have good accuracy as long as the data is skewed.

- F-AGMS, FC, and CM sketches (all of them are based on random hashing) have fast and comparable update performance that ranges between $50 - 400$ ns depending on the size of the sketch.

5.3.1 Testbed and Methodology

Sketch Implementation: We implemented a generic framework that incorporates the sketching techniques mentioned throughout the chapter. Algorithms for generating random variables with limited degree of independence [45, 51] are at the core of the framework. Since the sketching techniques have a similar structure, they are designed as a hierarchy parameterized on the type of random variables they employ. Applications have only to instantiate the sketching structures with the corresponding size and random variables, and to call the update and the estimation procedures.

Data Sets: We used two synthetic data generators and one real-life data set in our experiments. The data sets cover an extensive range of possible inputs, thus allowing us to infer general results on the behavior of the compared sketching techniques.

Census data set [17]: This real-life data set was extracted from the Current Population Survey (CPS) data repository, which is a monthly survey of about $50,000$ households. Each month's data contains around $135,000$ tuples with 361 attributes. We ran experiments for estimating the size of join on the *weekly wage (PTERNWA)*

numerical attribute with domain size 288, 416 for the surveys corresponding to the months of September 2002 (15, 563 records) and September 2006 (14, 931 records)[3] .

Estan's et al. [24] synthetic data generator: Two tables with approximately 1 million tuples each with a Zipf distribution for the frequencies of the values are randomly generated. The values are from a domain with 5 million values, and for each of the values its corresponding frequency is chosen independently at random from the distribution of the frequencies. We used in our experiments the *memory-peaked* (Zipf=0.8) and the *memory-unpeaked* (Zipf=0.35) data sets.

Synthetic data generator: We implemented our synthetic data generator for frequency vectors. It takes into account parameters such as the domain size, the number of tuples, the frequency distribution, and the correlation (decor = 1 − correlation) coefficient. Out of the large variety of data sets that we conducted experiments on, we focus in this experimental evaluation on frequency vectors over a $2^{14} = 16,384$ size domain that contain 1 million tuples and having Zipf distributions (the Zipf coefficient ranges between 0 and 5). The degree of correlation between two frequency vectors varies from full correlation to complete independence.

Answer-Quality Metrics: Each experiment is performed 100 times and the average relative error, i.e., $\frac{|\text{actual}-\text{estimate}|}{\text{actual}}$, over the number of experiments is reported. In the case of direct comparison between two methods, the ratio between their average relative errors is reported. Although we performed the experiments for different sketch sizes, the results are reported only for a sketch structure consisting of 21 vectors with 1024 counters each ($n = 1024$, $m = 21$), since the same trend was observed for the other sketch sizes.

5.3.2 Results

Self-Join Size Estimation: The behavior of the sketching techniques for estimating the self-join size as a function of the Zipf coefficient of the frequency distribution is

[3] After eliminating the records with missing values.

depicted in Figure 5-8 both on a normal (a) as well as logarithmic (b) scale. As expected, the errors of AGMS and FC sketches are similar (the difference for (close to) uniform distributions is due to the EH3 [51] random number generator). While F-AGMS has almost the same behavior as FC (AGMS) for small Zipf coefficients, the F-AGMS error is drastically decreasing for Zipf coefficients larger than 0.8. These are due to the effect the median estimator has on the distribution of the predicted results: for small Zipf coefficients the distribution is normal, thus the performance of the median estimator is approximately 25% worse, while for large Zipf coefficients the distribution is focused around the true result (Section 5.2). CM sketches have extremely poor performance for distributions (close to) uniform. This can be explained theoretically by the dependency on the L_1 norm, much larger than the L_2 norm in this regime. Intuitively, uniform distributions have multiple non-zero frequencies that are hashed into the same bucket, thus highly over-estimating the predicted result. The situation changes dramatically at high skew when it is highly probable that each non-zero frequency is hashed to a different bucket, making the estimation almost perfect. Based on these results, we can conclude that F-AGMS is the best (or close to the best) sketch estimator for computing the second frequency moment, irrespective of the skew.

Join Size Estimation: In order to determine the performance of the sketching techniques for estimating the size of join, we conducted experiments based on the Zipf coefficient and the correlation between the two frequency vectors. A correlation coefficient of 0 corresponds to two identical frequency vectors (self-join size). For a correlation coefficient of 1, the frequencies in the two vectors are completely shuffled. The results for different Zipf coefficients are depicted in Figure 5-9 as a function of the correlation. It can be clearly seen how the relation between the sketch estimators is changing as a function of the skew (behavior identical to the self-join size). Moreover, it seems that the degree of correlation is affecting similarly all the estimators (the error increases as the degree of correlation is increasing), but it does not affect the relative order given by the

Zipf coefficient. The same findings are reinforced in Figure 5-10 which depicts the relative performance, i.e., the ratio of the average relative errors, between pairs of estimators for computing the size of join. Figure 5-11 plots the accuracy for estimating the size of join of two streams with different skew coefficients. While the error of F-AGMS and FC increases with the skewness of the free stream, the error of CM stays almost constant, having a minimum where the two streams have equal Zipf coefficients. At the same time, it seems that the value of the error is determined by the smallest skew parameter. Consequently, we conclude that, as in the case of self-join size, the Zipf coefficient is the only parameter that influences the relative behavior of the sketching techniques for estimating the size of join of two frequency vectors.

Memory Budget: The accuracy of the sketching methods (AGMS is excluded since its behavior is identical to FC, but its update time is much larger) as a function of the space available is represented in Figure 5-12 for one of Estan's synthetic data sets (a) and for the census real-life data set (b). The error of CM sketches is orders of magnitude worse than the error of the other two methods for the entire range of available memory (due to the low skew). The accuracy of F-AGMS is comparable with that of FC for low skew data, while for skewed data F-AGMS is clearly superior. Notice that the relative performance of the techniques is not dependent on the memory budget.

Update Time: The goal of the timing experiment is to clarify if there exist significant differences in update time between the hash sketches since the random variables they use are different. As shown in Figure 5-14, all the schemes have comparable update time performance, CM sketches being the fastest, while FC sketches are the slowest. Notice that the relative gap between the schemes shrinks when the number of counters is increasing since more references are made to the main memory. As long as the sketch vector fits into the cache, the update rate is extremely high (around 10 million updates

can be executed per second on the test machine[4]), making hash sketches a viable solution for high-speed data stream processing.

5.3.3 Discussion

As we have seen, the statistical and empirical study in this chapter paints a different picture than suggested by the theory (see Table 5-3). Table 5-4 summarizes these results qualitatively and indicates that on skewed data, F-AGMS and CM sketches have much better accuracy than expected.

The statistical analysis in Section 5.2 revealed that the theoretical results for Fast-AGMS (F-AGMS) and Count-Min (CM) sketches do not capture the significantly better accuracy with respect to AGMS and Fast-Count (FC) sketches for skewed data. The reason there exists such a large gap between the theory and the actual behavior is the fact that the median, for F-AGMS, and the minimum, for CM, have a fundamentally different behavior than the mean on skewed data. This behavior defies statistical intuition since most distributions that are encountered in practice have relatively small kurtosis, usually below 20. The distributions of approximation techniques that use hashing on skewed data can have kurtosis in the 1000 range, as we have seen for F-AGMS sketches. For these distributions, the median, as an estimator for the expected value, can have error 10^6 smaller than the mean.

An interesting property of all sketching techniques is that the relationship between their accuracy does not change significantly when the degree of correlation changes, as indicated by Figure 5-10. The relationship is strongly influenced by the skew though, which suggests that the nature of the individual relations, but not the interaction between them, dictates how well sketching techniques behave.

[4] The results in Figure 5-14 are for a Xeon 2.8 GHz processor with 512 KB of cache. The main memory is 4 GB.

The relationship between sketches in Figure 5-10 also indicates that F-AGMS sketches essentially work as well as AGMS and FC for small skew and just slightly worse than CM for large skew. It seems that F-AGMS sketches combine in an ideal way the benefits of AGMS sketches and hashes and give good performance throughout the spectrum of problems without the need to determine the skew of the data. While CM sketches have better performance for large skew, their use seems riskier since their performance outside this regime is poor and their accuracy cannot be predicted precisely for large skew. It seems that, unless extremely precise information about the data is available, F-AGMS sketches are the safe choice.

5.4 Conclusions

In this chapter we studied the four basic sketching techniques proposed in the literature, AGMS, Fast-AGMS, Fast-Count, and Count-Min, from both a statistical and empirical point of view. Our study complements and refines the theoretical results known about these sketches. The analysis reveals that Fast-AGMS and Count-Min sketches have much better performance than the theoretical prediction for skewed data, by a factor as much as 10^6 to 10^8 for large skew. Overall, the analysis indicates strongly that Fast-AGMS sketches should be the preferred sketching technique since it has consistently good performance throughout the spectrum of problems. The success of the statistical analysis we performed indicates that, especially for estimators that use minimum or median, such analysis gives insights that are easily missed by classical theoretical analysis. Given the good performance, the small update time, and the fact that they have tight error guarantees, Fast-AGMS sketches are appealing as a practical basic approximation technique that is well suited for data stream processing.

Table 5-1. Families of ±1 random variables.

Counter	Key domain				
	1	2	3	4	5
1	+1	+1	+1	−1	−1
2	+1	−1	−1	+1	+1
3	−1	+1	−1	+1	−1

Table 5-2. Families of hash functions.

Row	Key domain				
	1	2	3	4	5
1	1	1	3	2	3
2	1	3	2	1	2
3	3	2	3	3	1

Table 5-3. Expected theoretical performance.

Sketch	Size of Join		
	Large		Small
	Low Skew	High Skew	
AGMS	0	0	−
Fast-AGMS	0	0	−
Fast-Count	0	0	−
Count-Min	−	0	−

Table 5-4. Expected statistical/empirical performance.

Sketch	Size of Join		
	Large		Small
	Low Skew	High Skew	
AGMS	0	0	−
Fast-AGMS	0	+	+
Fast-Count	0	0	−
Count-Min	−	+	+

(a) Basic estimator (b) Average estimator

Figure 5-1. The distribution of AGMS sketches for self-join size.

(a) Zipf=0.2 (b) Zipf=1.5

Figure 5-2. The distribution of F-AGMS sketches for self-join size.

(a) Zipf=1.0 (b) Zipf=2.0

Figure 5-3. The distribution of CM sketches for self-join size.

(a) Basic estimator (b) Average estimator

Figure 5-4. The distribution of FC sketches for self-join size.

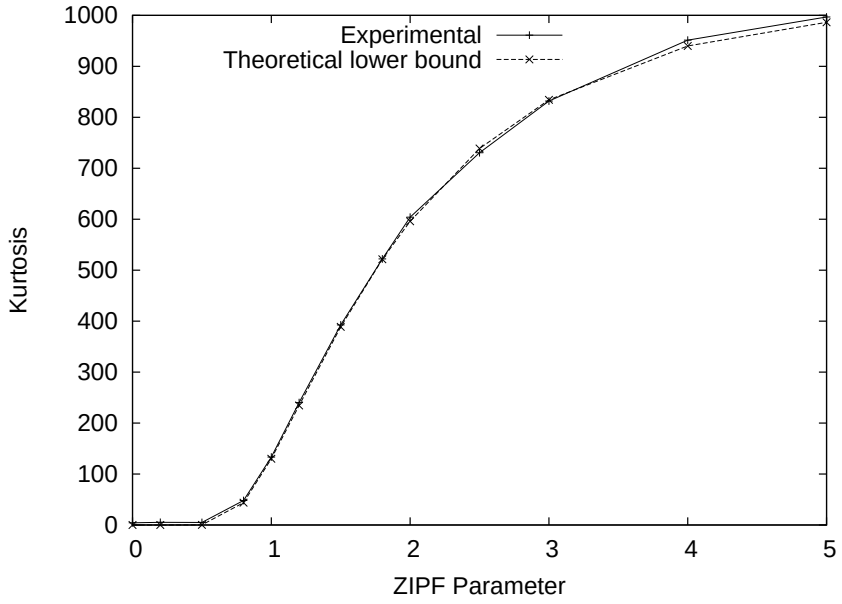


Figure 5-5. F-AGMS kurtosis.

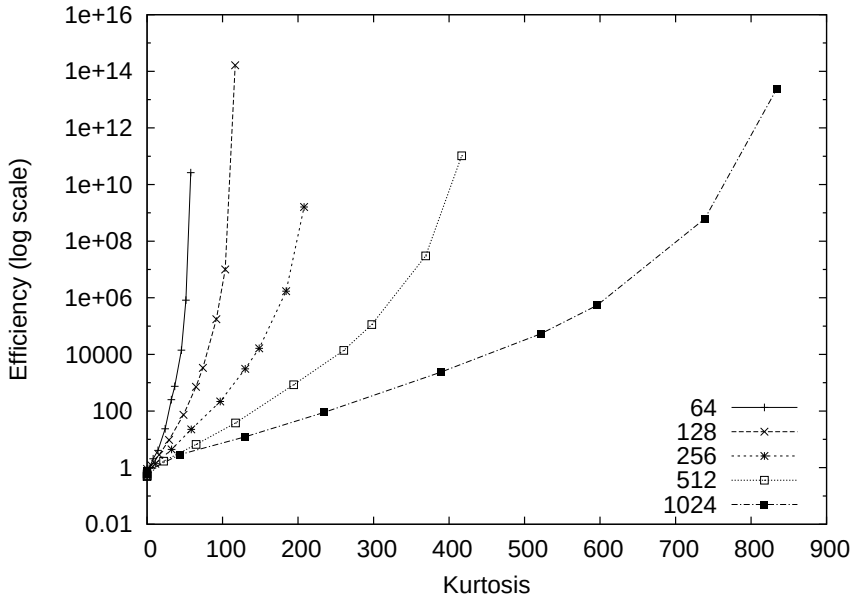
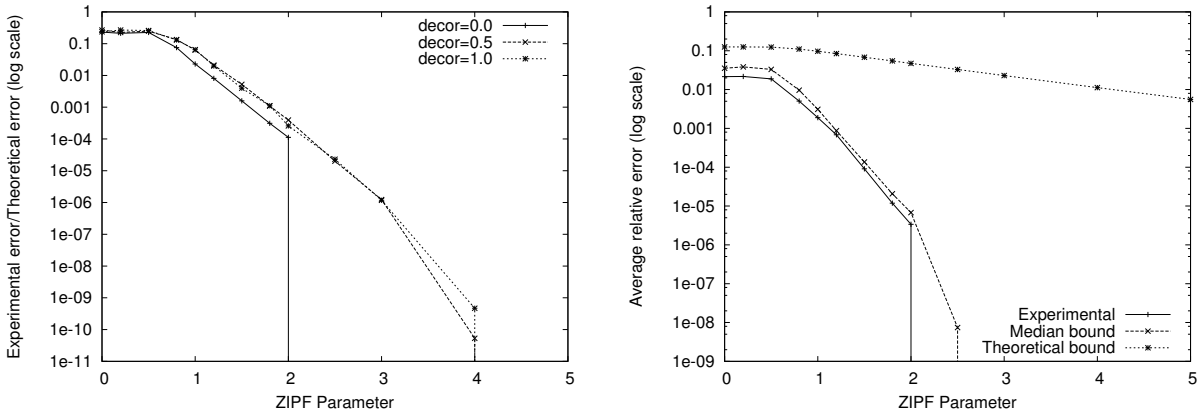
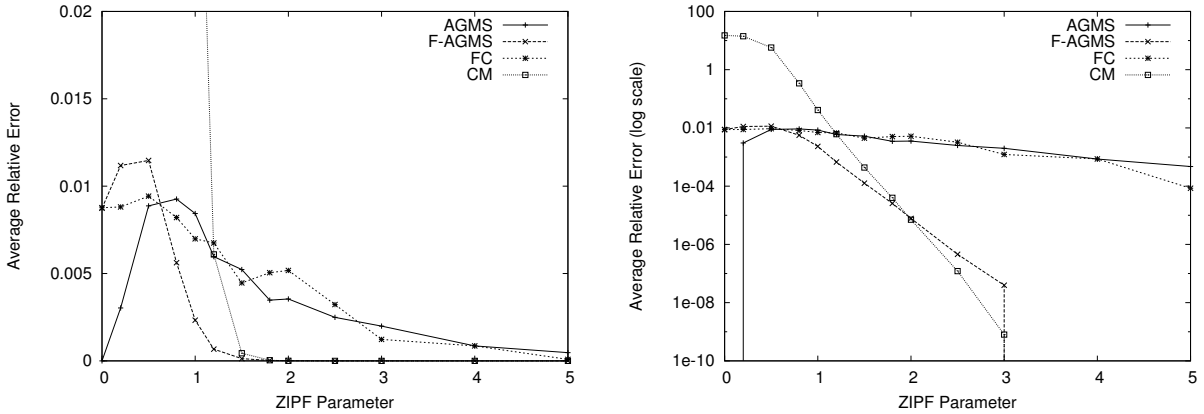


Figure 5-6. F-AGMS efficiency.



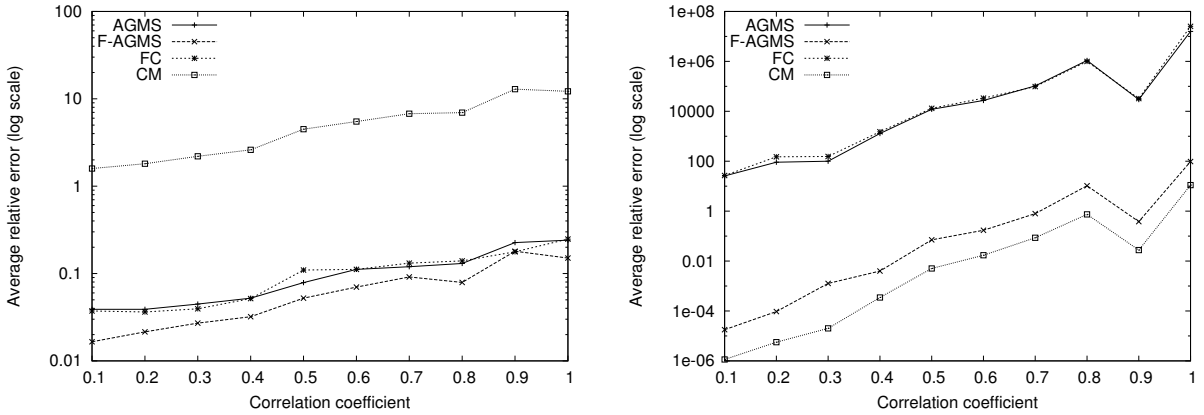
(a) Correlation (b) Confidence bounds

Figure 5-7. Confidence bounds for F-AGMS sketches as a function of the skewness of the data.



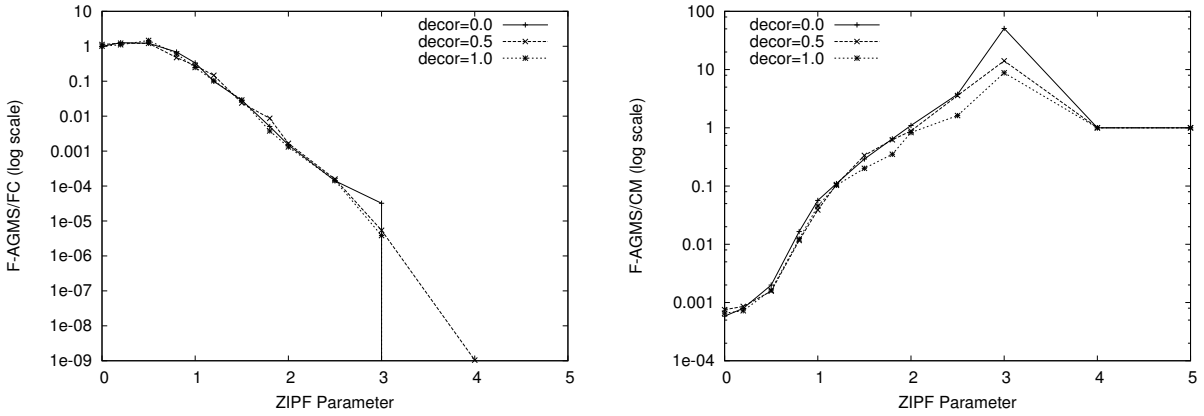
(a) Normal scale (b) Log scale

Figure 5-8. Accuracy as a function of the Zipf coefficient for self-join size estimation.



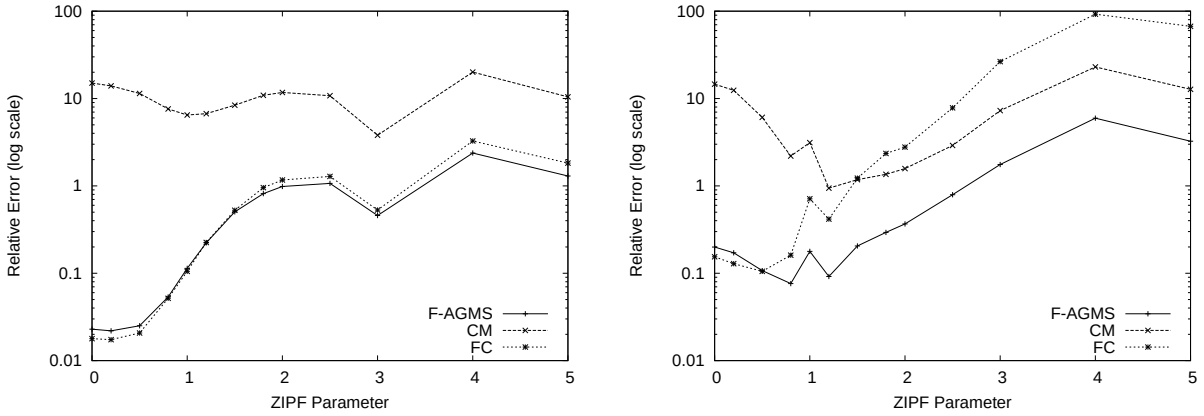
(a) Zipf = 0.8 (b) Zipf = 3.0

Figure 5-9. Accuracy as a function of the correlation coefficient for size of join estimation.



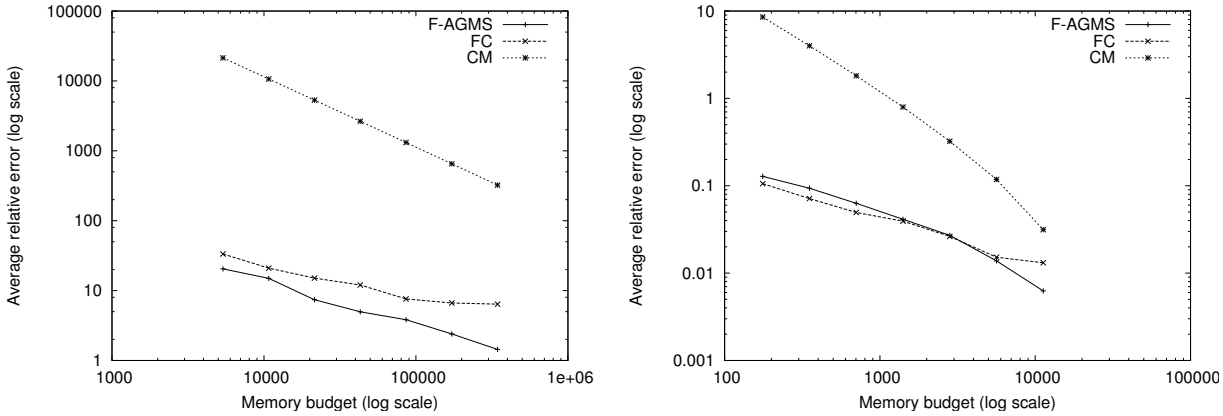
(a) F-AGMS vs AGMS (FC) (b) F-AGMS vs CM

Figure 5-10. Relative performance for size of join estimation.



(a) Zipf = 0.5 (b) Zipf = 1.0

Figure 5-11. Accuracy as a function of the skewness of the data for size of join estimation.



(a) Memory peaked (b) Census

Figure 5-12. Accuracy as a function of the available space budget.

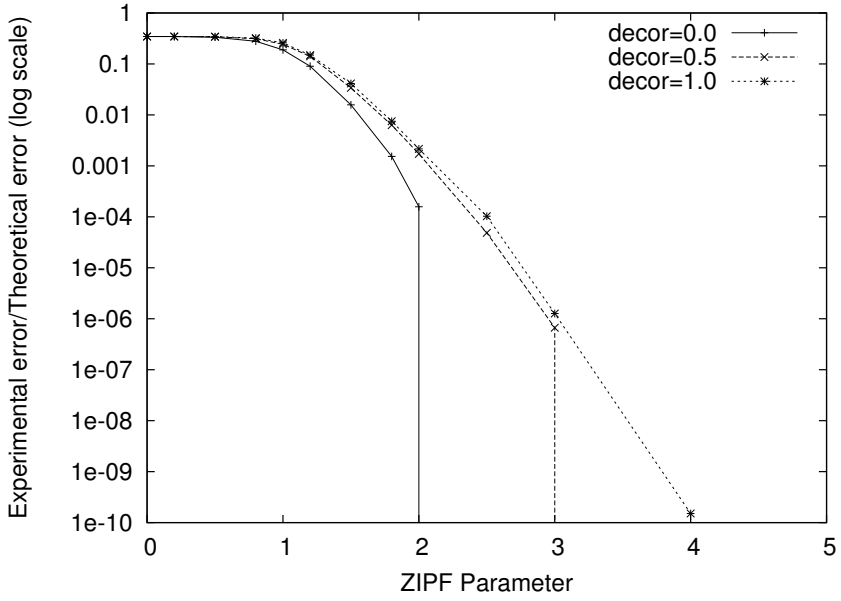


Figure 5-13. Confidence bounds for CM sketches as a function of the skewness of the data.

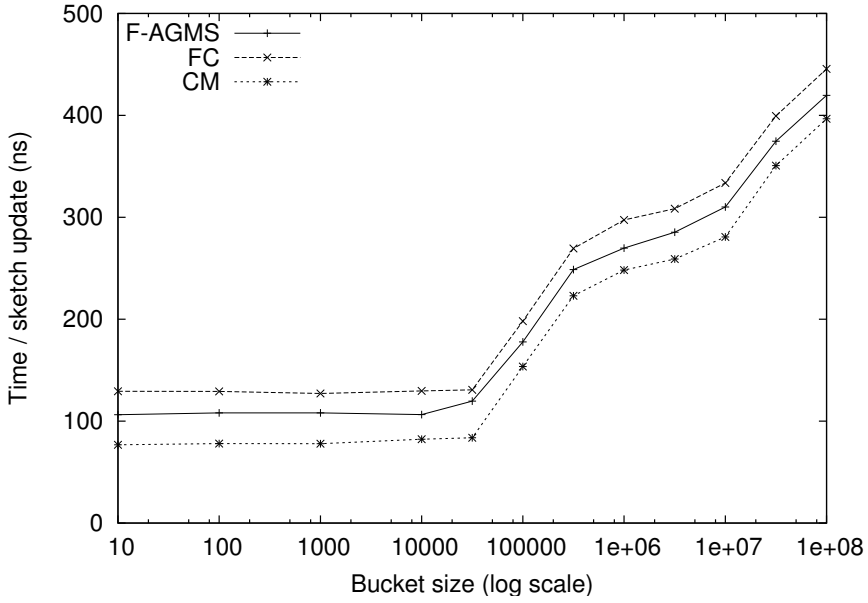


Figure 5-14. Update time as a function of the number of counters in a sketch that has only one row.

CHAPTER 6
SKETCHES FOR INTERVAL DATA

The problem we treat in this section is to estimate the size of join between a data stream given as points and a data stream given as intervals using sketches. Notice that this remains a size of join problem defined over the frequencies of individual points but, since one of the streams is specified by intervals rather than individual points, if basic sketches are used the update time is proportional to the size of the interval, which is undesirable. In order to apply sketching techniques for the derived problem, there exist two alternatives: either use random variables that are fast range-summable, or apply a domain transformation for reducing the size of the intervals. The two solutions, DMAP and fast range-summation, have update time sub-linear in the interval size. DMAP [18] consists in mapping both the intervals and the points into the space of dyadic intervals in order to reduce the size of the interval representation. Since both intervals and points map to a logarithmic number of dyadic intervals in the dyadic space, the update time becomes poly-logarithmic with respect to the input. Fast range-summation [51] uses properties of the pseudo-random variables in order to sketch intervals in sub-linear time, while points are sketched as before. While the update time of these methods is poly-logarithmic with respect to the size of the interval, since they are based on AGMS sketches, the update time is also proportional with the size of the sketch. In the light of the statistical and empirical evaluation of hash-based sketches, the update time due to sketching could be significantly improved without loss in accuracy. The question we ask and thoroughly explore in this section is whether hash-based sketches can be combined successfully with the two methods to sketch interval data. The insights gained from the statistical analysis and the empirical evaluation of the hash-based sketching techniques are applied in this section to provide variants of the two methods for sketching interval data that have significantly smaller update time and comparable accuracy.

In this chapter we investigate both theoretically and empirically the known methods for generating the random variables used by AGMS sketches with the goal of identifying the generating schemes that are practical, both for the traditional application of AGMS sketches, i.e., aggregate computation over streaming data, and for applications that involve interval input. More specifically, our contributions are:

- We provide a detailed study of the practicality of fast range-summation for the known generating schemes. We show that no 4-wise independent generating scheme is practically fast range-summable, even though the scheme based on Reed-Muller codes can be theoretically range-summed in sub-linear time [9].
- We provide a formal treatment of dyadic intervals that allows us to design efficient fast range-summable algorithms for two schemes: 3-wise independent BCH scheme (BCH3) [2] and Extended Hamming scheme (EH3) [25].
- We explain how two problems, size of spatial joins and selectivity estimation, can be reduced to the size of join problem, reduction that allows us to provide estimates of their results based on AGMS sketches. The resulting estimators significantly outperform the state-of-the-art solution based on dyadic mappings [18], sometimes by as much as a factor of 8 in relative error.

We apply the results obtained from the statistical study to design effective algorithms for sketching interval data. Sketches over interval data can be useful by themselves, but they are also a building block in solutions to more complex problems like the size of spatial joins [5, 18, 55]. DMAP and fast range-summation [51] are the existing solutions for sketching interval data. They are both inefficient when compared to hash-based sketches because of the use of AGMS sketches which have higher update time. In this chapter we study how DMAP and fast range-summation can use the more efficient hash-based sketches. In particular, we show that only DMAP can be extended to other types of sketches and, thus, a significant improvement in update time can be gained by a simple replacement of the underlying sketching technique. To improve the accuracy of DMAP, significantly inferior to that of the fast range-summation method, we propose a simple modification that keeps exact counts for some of the frequencies. We call this modification DMAP COUNTS. We also introduce a method to improve the update

performance of fast range-summation AGMS sketches based on a simple equi-width partitioning of the domain. The experimental results show that these derived methods keep the advantage of their base methods, while significantly improving their drawbacks, to the point where they are efficient both in accuracy and update time. The gain in update time can be as large as two orders of magnitude, thus making the improved methods practical. The empirical study suggests that DMAP is preferable when the update time is primordial and fast range-summation is desirable for better accuracy.

In the rest of the chapter, we first identify applications that can be reduced to sketching interval data in Section 6.1. Section 6.3 is a detailed study of the dyadic mapping method, with a particular emphasis on dyadic intervals. Section 6.4 treats fast range-summable generating schemes, while Section 6.5 deals with fast range-summation for sketching interval data. Section 6.6 contains an experimental study of the sketching methods for interval data.

6.1 Sketch Applications

AGMS sketches have been also used as building blocks in many applications. For example, [27] compute aggregates over expressions involving set operators, while [25] approximate the L^1-difference of two data-streams. A solution based on AGMS sketches for the wavelet decomposition of a data-stream is introduced in [32], while dynamic quantiles are approximated using *random subset sums* in [33]. In networking, sketches can be applied for change detection as in [42].

As already mentioned, sketching interval data is a fundamental problem that is used as a building block in more complex problems such as estimating the size of spatial joins and building dynamic histograms. For example, for the size of spatial joins problem in which two data streams of intervals are given, two sketches are built for each stream, one for the entire interval and one for the end-points. The size of the spatial join between the two interval data streams is subsequently estimated as the average of the product of the

interval sketch from one stream and the sketch for the end-points from the other stream (see [18] for complete details).

Although they were introduced for estimating the size of join of two point relations, AGMS sketches are a versatile approximation method that can accommodate multiple types of input. In this chapter we focus on a variation of the original size of join problem in which one of the relations is specified as intervals (this is equivalent to specifying every point inside the interval). To motivate the importance of the derived problem, we introduce two applications that can be expressed as the size of join of two relations, one consisting of points, and one containing intervals. We also identify a class of applications that can be reduced to size of join problems involving an interval relation and a point relation.

6.1.1 Size of Spatial Joins

Given two relations of line segments in the unidimensional space, the size of spatial join problem is to compute the number of segments from the two relations that overlap. The approach in [18], even though not explicitly stated, reduces the size of spatial join problem to two size of join problems, each involving a point relation and an interval relation: the size of join of the line segments from the first relation and the segment end-points from the second relation and, symmetrically, the size of join of the segment end-points from the first relation and the segments from the second relation.

In order to implement a solution based on AGMS sketches for the point-interval size of join problem, we have to deal first with efficiently sketching intervals. The solution proposed in [18] realizes a number of transformations for reducing the size of each interval, thus improving the sketching time. We present a detailed analysis of this generic method for sketching intervals in Section 6.3.

6.1.2 Selectivity Estimation for Building Dynamic Histograms

Any histogram construction algorithm has to evaluate the average frequency of the elements in a bucket, also known as the selectivity of the bucket. By introducing a virtual

relation having value 1 only for the elements in the bucket, and value 0 otherwise, the average frequency can be expressed as the size of join between the original relation and the virtual relation corresponding to the bucket, divided by the size of the bucket. Usually a bucket is an interval in the unidimensional space, thus the problem of sketching intervals has to be solved in order to implement an AGMS sketches solution for the selectivity estimation problem.

[57] introduce a solution based on sketches for building dynamic histograms. The streaming relation is summarized as a sketch and then the histogram is extracted from the sketch. The authors focused more on how to determine the buckets such that the resulting histogram to be optimal. They did not consider in detail the problem of sketching a bucket and took this step as a black-box in their algorithms.

6.2 Problem Formulation

The common point of the above applications is that they can be reduced to size of join problems between a point relation and an interval relation. In order to implement effective sketch-based solutions for this derived problem, the sketch of an interval has to be computed more efficiently than sketching each point in the interval.

The problem considered in this section is a derivation of the size of join problem defined in Section 2.1 in which one of the two data streams is given by (interval, frequency) pairs rather than (key, frequency) pairs. The frequency is attached to each element in the interval, not only to a single key. Formally, let $S = (e_1, w_1), \ldots, (e_s, w_s)$ and $T = ([l_1, r_1], v_1), \ldots, ([l_t, r_t], v_t)$ be two data streams, where the keys e_i and the intervals $[l_i, r_i]$, with $l_i \leq r_i$, are members of the set $I = \{0, 1, \ldots, N-1\}$, and w_i and v_i, respectively, represent frequencies. The computation using sketches of the *size of join* of the two data streams defined as the *inner-product* of their frequency vectors remains our focus. Notice that it is straightforward to reduce this problem to the basic size of join problem by observing that a pair $([l_i, r_i], v_i)$ in T can be represented as an equivalent set of pairs (e_j, v_i) in S, with e_j taking all the values between l_i and r_i, $l_i \leq e_j \leq r_i$. The

drawback of this solution is the time to process an interval which is linear in the size of the interval.

Summarizing, the problem we treat in this chapter is to **estimate the size of join between a point relation and an interval relation using AGMS sketches**. We start by revisiting the state-of-the-art solution based on dyadic mappings (DMAP). Then we introduce solutions that use fast range-summable random variables that are based on the known schemes for generating ±1 random variables (Section 3.1).

6.3 Dyadic Mapping (DMAP)

The dyadic mapping method [18], which we call here DMAP, uses dyadic intervals in order to reduce the size of an interval, thus making possible the efficient sketching of intervals. Since dyadic intervals are at the foundation of the method, we introduce them first. We will see that dyadic intervals are also important for fast range-summation (Section 6.4).

6.3.1 Dyadic Intervals

[33] introduce dyadic ranges for sketching intervals in the context of quantile computation. Although they are employed in different work [18, 25, 32], a formal treatment of dyadic intervals was not previously provided. Such a formal treatment is necessary to design efficient algorithms for methods that need dyadic interval decomposition. In this section we provide such a formal treatment.

For the rest of this section, consider that the domain I has size $|I| = N = 2^n$. If this is not the case, the domain can be extended to the first power of 2 that is greater than N. The dyadic intervals of the domain I are intervals that can be organized in a hierarchical structure – the dyadic intervals over the domain $\{0, \ldots, 15\}$ are depicted in Figure 6-2 – that has $n + 1$ layers. There exists only one interval at level 0 – the entire domain. On the subsequent levels, each interval is split in two equal-size intervals to produce smaller dyadic intervals. On the last level, all dyadic intervals have size one and they consist of all the individual points in the domain. Intuitively, such a construction produces intervals

that have size a power of 2 (more precisely all the intervals at level k have size 2^{n-k}) and that have boundaries aligned at multiples of powers of 2. This description is good enough to argue properties of dyadic intervals, but it does not suffice to formally prove the correctness of algorithms that use dyadic intervals. Insights from the strict theoretical treatment of dyadic intervals actually lead to very efficient implementations.

We start our presentation with a formal definition of dyadic intervals and some basic properties.

Definition 5. *A **dyadic interval** over the domain $I = \{0, 1, \ldots, N-1\}$, $|I| = N = 2^n$, is an interval of the form $[q2^j, (q+1)2^j)$, where $0 \leq j \leq n$ and $0 \leq q \leq 2^{n-j} - 1$.*

Proposition 6. *There are exactly 2^j dyadic intervals at level j, each containing 2^{n-j} points from I.*

Proof. Follows directly from definition. □

Proposition 7. *The dyadic intervals at level j, $0 \leq j \leq n$, form a partition of the domain I. That is, they are disjoint and their union is equal with the entire domain.*

Proof. If j is fixed, using the above definition, the dyadic intervals at level j are: $[0, 2^j), [2^j, 2 \cdot 2^j), \ldots, [(2^{n-j} - 1) \cdot 2^j, 2^{n-j} \cdot 2^j)$. Clearly, they form a partition of $[0, 2^n)$. □

Proposition 8. *Let δ_1 and δ_2 be two arbitrary distinct dyadic intervals. If $\delta_1 \cap \delta_2 \neq \emptyset$, then either $\delta_1 \subset \delta_2$ or $\delta_2 \subset \delta_1$.*

Proof. Let $\delta_1 = [q2^j, (q+1)2^j)$ and $\delta_2 = [r2^k, (r+1)2^k)$. We have two distinct cases, $j = k$, and $j \neq k$. We discuss each in turn.

Suppose $j = k$. Then either $q = r$, which is not possible since $\delta_1 \neq \delta_2$, or $q \neq r$. In the latter case, the two dyadic intervals, according to Proposition 7, are different elements of a partition of the domain, so they do not intersect, which is a contradiction.

Suppose, without loss of generality, that $j > k$. We have to show that $\delta_2 \subset \delta_1$. This is the only possibility since δ_1 has more elements than δ_2. We have two distinct cases: $r2^k \in \delta_1$ and $r2^k < q2^j$.

- If $r2^k \in \delta_1$ then either $r2^k \leq (q+1)2^j - 2^k$, in which case $\delta_2 \subset \delta_1$ since we can fit all 2^k elements of δ_2 from $r2^k$ to the end of δ_1, or $r2^k \in ((q+1)2^j - 2^k, (q+1)2^j)$. In this last case, we have $r2^k > (q+1)2^j - 2^k$ and $r2^k < (q+1)2^j$. Dividing the two inequalities by 2^k, which is a positive quantity, we get: $(q+1)2^{j-k} > r > (q+1)2^{j-k} - 1$. Since $j > k$, 2^{j-k} is an integer quantity. But both r and q are integers and this inequality is a contradiction since we cannot insert an integer between two consecutive integers.

- If $r2^k < q2^j$, since $\delta_1 \cap \delta_2 \neq \emptyset$, it must be the case that $(r+1)2^k > q2^j$. From these two inequalities, by dividing by 2^k, we have: $r < q2^{j-k} < r+1$. This is again a contradiction since $q2^{j-k}$ is an integer that has to be between two consecutive integers.

□

An interesting question is how to express an arbitrary interval as the union of dyadic intervals. Since the decomposition is not unique in general and it is desirable to have decompositions of small sizes, we look for minimal decompositions in terms of the number of elements. The concept of minimal dyadic cover of an interval and its properties are introduced next.

Definition 6. *The **minimal dyadic cover** of an interval $[\alpha, \beta]$, $D([\alpha, \beta])$, is the set of dyadic intervals $\delta_1, \delta_2, \ldots, \delta_m$ of smallest cardinality, for which $\delta_1 \cup \delta_2 \cup \cdots \cup \delta_m = [\alpha, \beta]$.*

Proposition 9. *The dyadic intervals in a minimal dyadic cover of an interval form a partition of that interval.*

Proof. Let $\delta_1 \cup \delta_2 \cup \cdots \cup \delta_m = [\alpha, \beta]$ be a minimal dyadic cover of the arbitrary interval $[\alpha, \beta]$. It is enough to show that no two dyadic intervals intersect. According to Proposition 8, if two dyadic intervals intersect, then one includes the other. The one that is included can be eliminated from the minimal dyadic cover of $[\alpha, \beta]$ without changing the coverage, which implies that the cover is not minimal, a contradiction. □

Definition 7. *$d([\alpha, \beta]) = q2^j$ is the **dyadic cut-point** of the interval $[\alpha, \beta]$ if j is the largest integer smaller than n ($|I| = 2^n$) for which there exists q such that $q2^j \in [\alpha, \beta]$.*

Proposition 10. *The dyadic cut-point of an interval is unique.*

Proof. Let $q2^j$ and $q'2^j$, $q < q'$, be two distinct dyadic cut-points of the same interval. Since, by definition, both are included in the interval, all integers $q''2^j$ with $q \leq q'' \leq q'$ are also included in the interval, so they are also dyadic cut-points. Since $q \neq q'$, at least one of the q'' has to be even, for example $q'' = 2r$, thus $q''2^j = r2^{j+1}$. This contradicts the fact that j is maximal and proves the uniqueness of $q2^j$. □

Using these properties, we formalize the relation between the dyadic cut-point of an interval and its minimal dyadic cover in the following lemma.

Lemma 3. *Let $[\alpha, \beta]$ be an interval, $d([\alpha, \beta]) = q2^j$ be its dyadic cut-point, and $D([\alpha, \beta])$ be its minimal dyadic cover. Then:*

1. *None of the dyadic intervals in $D([\alpha, \beta])$ contains the dyadic cut-point, except as a left end-point.*

2. *The minimal dyadic cover $D([\alpha, \beta])$ is the union of the minimal dyadic covers of $[\alpha, q2^j)$ and $[q2^j, \beta]$, i.e., $D([\alpha, \beta]) = D([\alpha, q2^j) \cup D([q2^j, \beta])$.*

3. *The minimal dyadic cover $D([q2^j, \beta])$ consists of a sequence of at most j dyadic intervals with strictly decreasing sizes.*

4. *The minimal dyadic cover $D([\alpha, q2^j)$ consists of a sequence of at most j dyadic intervals with strictly increasing sizes.*

5. *The minimal dyadic cover $D([\alpha, \beta])$ contains at most $2j$ dyadic intervals, from which at most two are from the same level.*

Proof.

1. Let δ be a dyadic interval in the minimal dyadic cover that contains the dyadic cut-point $d([\alpha, \beta]) = q2^j$, but not as its left end-point. Then, by Proposition 8, $[q2^j, (q+1)2^j) \subset \delta$. Equality is not possible because $q2^j$ would be at the start of δ. Then $[\frac{q}{2}2^{j+1}, (\frac{q}{2}+1)2^{j+1}) \subset \delta$ is the smallest dyadic interval that can contain the previous interval. This means that $\frac{q}{2}2^{j+1}$ is included in $[\alpha, \beta]$, which contradicts the fact that $q2^j$ is the dyadic cut-point.

2. Suppose the union of the minimal dyadic covers of $[\alpha, q2^j)$ and $[q2^j, \beta]$ is not the minimal dyadic cover of $[\alpha, \beta]$ and let $D'([\alpha, \beta])$ be this minimal cover. If $q2^j$ is not included in the core of any of the dyadic intervals in the minimal dyadic cover, then we can partition the minimal dyadic cover into two sets, one that covers $[\alpha, q2^j)$, and one that covers $[q2^j, \beta]$. These covers have to be minimal, otherwise the cover of $[\alpha, \beta]$ is not minimal. But, according to (1), $q2^j$ cannot be part of the core of

any dyadic interval in any dyadic cover of $[\alpha, \beta]$, so it has to be the case that the statement is correct.

3. We give a constructive proof of the statement. The claim is that the following algorithm finds the minimal dyadic cover of $[q2^j, \beta]$. Starting from the point $q2^j$, find the largest dyadic interval that fits into $[q2^j, \beta]$, and then iterate from the point immediately following this dyadic interval until all of $[q2^j, \beta]$ is covered. Since $[q2^j, \beta]$ is finite and each dyadic interval contains at least one point, the algorithm terminates in at most $\beta - q2^j + 1$ steps.
Let us show that the built cover is minimal and has size at most j. We consider an arbitrary interval $[q'2^{j'}, \beta]$ with $q'2^{j'}$ being its dyadic cut-point. Let $q''2^{j''}$ be the dyadic cut-point of the interval $(q'2^{j'}, \beta]$, i.e., the next dyadic cut-point since the current dyadic cut-point is excluded from the interval. We claim that the interval $[q'2^{j'}, q''2^{j''})$ is the largest dyadic interval that is part of the minimal dyadic cover of $[q'2^{j'}, \beta]$. First, $j'' < j'$ otherwise $q''2^{j''}$ would be the dyadic cut-point of $[q'2^{j'}, \beta]$. This means that $[q'2^{j'}, q''2^{j''})$ is a dyadic interval of size $2^{j''}$ since it is equivalent to $[2^{j'-j''}q' \cdot 2^{j''}, q''2^{j''})$ and it cannot be a union of dyadic intervals from level j'' because this would contradict the fact that $q''2^{j''}$ is a dyadic cut-point. In fact, this dyadic interval is the largest dyadic interval included in $[q'2^{j'}, \beta]$, otherwise we again contradict the fact that $q''2^{j''}$ is a dyadic cut-point. From this, it follows directly that the dyadic interval $[q'2^{j'}, q''2^{j''})$ is the unique largest dyadic interval in the minimal dyadic cover of $[q'2^{j'}, \beta]$. By induction, the sequence of dyadic intervals generated by the algorithm produces the minimal dyadic cover of $[q2^j, \beta]$. Since $j'' < j'$, the algorithm ends in at most j steps.

4. The proof is symmetric to 3.

5. Follows directly from 3 and 4.

□

Corollary 4. *The minimal dyadic cover of an interval $[\alpha, \beta]$ has cardinality at most $2(n-1)$, where $|I| = 2^n$. That is, $|D([\alpha, \beta)]| \leq 2(n-1)$.*

Proof. Follows directly from Lemma 3 by observing that $j = n - 1$ maximizes the cardinality of the minimal dyadic cover. □

The algorithm used in the constructive proof of Lemma 3 uses as a prerequisite the dyadic cut-point and starts determining the minimal dyadic cover to the left and to the right of this point. To avoid determining the dyadic cut-point, we can run the algorithm in reverse and find the dyadic intervals in the minimal dyadic cover starting from α and

going to the right, and starting from β and going to the left. Moreover, it is enough to inspect the binary representation of α and β to determine these dyadic intervals, as it is shown in Algorithm *MinimalDyadicCover* 6.8. For example, to determine the dyadic intervals covering $[\alpha, d([\alpha, \beta]))$, we inspect the bits of α starting from the least significant and whenever we detect a 1 bit, say in position j, we have to include the dyadic interval of size 2^j that covers all the integers that have the same binary representation as α on the most significant $n - j$ bits. For β the same pattern has to be followed, but the 0 bits signal the inclusion of a dyadic interval. At every step, a new bit in the description of α and β is analyzed and the algorithm stops when $\alpha = \beta + 1 = d([\alpha, \beta])$. Example 10 illustrates how the algorithm runs.

Example 10. *Consider the interval* $[100, 200]$*, with* $\alpha = 100 = (01100100)_2$ *and* $\beta = 200 = (11001000)_2$*. The algorithm inspects the bits in* α *and* β *starting from the least significant one. If a* 1 *bit is seen in* α*, useful processing has to be done. The same is equivalent for* 0 *bits in* β*. Bit* 0 *in* β *is* 0*, implying the addition to the minimal dyadic cover of the interval* $[200, 201)$ *and the new* β *becomes* $\beta^{(1)} = 199 = (11000111)_2$*. The first bit in* α *that is equal with* 1 *is the third one. It forces the addition of the interval* $[100, 104)$ *to the minimal dyadic cover and* $\alpha^{(1)} = 104 = (01101000)_2$*. The next steps of the algorithm add the intervals* $[104, 112), [192, 200), [112, 128)$ *and* $[128, 192)$ *to the minimal dyadic cover,*

$$D([100, 200]) = \{[100, 104), [104, 112), [112, 128), [128, 192), [192, 200), [200, 201)\} \quad (6\text{–}1)$$

At this point, $\alpha^{(3)} = 128 = (10000000)_2$ *and* $\beta^{(3)} = 127 = (01111111)_2$*, and the algorithm returns. Notice that* $\alpha^{(3)}$ *equals the dyadic cut-point of the interval,* $d([100, 200]) = 2^7 = 128$*.*

Algorithm *MinimalDyadicCover* 6.8 has running time $O(\log(\beta - \alpha))$ if the elementary operations on integers take $O(1)$ time (as it is the case for implementations on real computers). If the notion of dyadic cut-point is not developed and Lemma 3 is not proved,

the intuitive algorithm to find the minimal dyadic cover of an interval is: find the largest dyadic interval contained in $[\alpha, \beta]$; add this dyadic interval to the minimal dyadic cover and then recurse on the start and end left-over parts of $[\alpha, \beta]$. The largest dyadic interval contained in a given interval can be found in $O(\log(|I|)$ steps and this has to be done for each of $O(\log(\beta - \alpha))$ dyadic intervals in the decomposition of $[\alpha, \beta]$, resulting in an $O(\log(|I|) \log(\beta - \alpha))$ algorithm. Notice that there is an $O(\log(|I|)$ gap between the two algorithms. This is due to the fact that the algorithm we provide cleverly avoids the search for the maximal dyadic interval and it is able to determine what dyadic intervals are in the minimal cover by simply inspecting the bit representation of the end-points α and β.

The main conclusion of the above mathematical formalization is that any interval can be decomposed efficiently in at most a logarithmic number of dyadic intervals. This implies that instead of identifying an interval by all the points it contains, we can represent the interval by its minimal dyadic cover. Thus, a reduction from a linear-size representation to a logarithmic-size representation is obtained. As we will see, this is a significant improvement in the sketching context. Moreover, determining the minimal dyadic cover of an interval can be implemented very efficiently by considering only the binary representation of the interval end-points.

6.3.2 Dyadic Mapping Method

DMAP [18] employs dyadic intervals for efficiently sketching intervals in the context of the size of join between a point relation and an interval relation. As already mentioned, the dyadic representation of an interval is more compact than the point representation.

DMAP method uses dyadic intervals in order to reduce the representation of an interval, thus making possible the efficient sketching of intervals (see [18, 33, 51] for details). DMAP is based on a set of three transformations (Figure 6-3) to which the size of join operation is invariant. The original domain is mapped into the domain of all possible dyadic intervals that can be defined over it. An interval in the original domain is mapped

into its minimal dyadic cover in the new domain. By doing this, the representation of the interval reduces to at most a logarithmic number of points in the new domain, i.e., the number of sketch updates reduces from linear to logarithmic in the size of the interval. At the same time, a point in the original domain maps to the set of all dyadic intervals that contain the point in the new domain, thus increasing the number of sketch updates from one to logarithmic in the size of the original domain. DMAP allows the correct approximation of the size of join in the mapped domain with the added benefit that the sketch of each relation can be computed efficiently since both for an interval, as well as a point, at most $\log |I| = n$ dyadic intervals have to be sketched.

[18] prove that the size of join between a point relation and an interval relation is invariant to these transformations, that is, the size of join over the original domain is equal with the size of join over the mapped domain. The intuition behind the proof lies on the fact that for each point in an interval there exists exactly one dyadic interval in the minimal dyadic cover that contains that point.

The application of any of the sketching methods in the dyadic domain is straightforward. For a point, the sketch data structure is updated with all the dyadic intervals that contain the point (exactly $\log |I| = n$). For an interval, the sketch is updated with the dyadic intervals contained in the minimal dyadic cover (at most $2n - 2$, but still logarithmic in the size of the interval). Specifically, in the case of AGMS sketches all the counters are updated for each dyadic interval, while in the case of hash-based sketches (Fast-AGMS, Fast-Count, Count-Min) only one counter in each row is updated for each dyadic interval. Notice that the update procedure is identical to the procedure for point data streams since a dyadic interval is represented as a point in the dyadic domain. Once the sketches for the two data streams are updated, the estimation procedure corresponding to each type of sketch described in Section 4.2 is immediately applicable.

The experimental results in [51] showed that DMAP has significantly worse performance, as much as a factor of 8, than fast range-summable methods for AGMS

sketches. We provide an explanation for this behavior based on the statistical analysis in Section 5.2 and the empirical results in Section 4.4. At the same time, we provide evidence that the performance of DMAP for hash-based sketches (Fast-AGMS in particular) cannot be significantly better. To characterize statistically the performance of DMAP, we first look at the distribution of the two data streams in the dyadic domain. The distribution of the point data stream has a peak corresponding to the domain (the largest dyadic interval) due to the fact that this dyadic interval contains all the points, so its associated counters get updated for each streaming point. The dyadic intervals at the second level, of size half of the domain size, also have high frequencies due to the same reason. As the size of dyadic intervals decreases, their frequency decreases too, to the point it is exactly the true frequency for point dyadic intervals. Unfortunately, the high frequencies in the dyadic domain are outliers because their impact on the size of join result is minimal (for example, the domain dyadic interval appears in the size of join only if the interval data stream contains the entire domain as an interval). Practically, DMAP transforms the distribution of the point data stream into a skewed distribution dominated by outliers corresponding to large dyadic intervals. The effect of DMAP over the distribution of the interval data stream is far less dramatic, but more difficult to quantify. This is due to the fact that both the size of the interval and the position are important parameters. For example, two intervals of the same size, one which happens to be dyadic and one translated by only a position, can generate extremely different minimal dyadic covers and, thus, distributions in the dyadic domain. Even without any further assumptions on the distribution of the interval data stream, we expect the skewed distribution of the point stream to affect negatively the estimate, due to the outliers corresponding to large dyadic intervals. At the same time, we would expect not to have a significant difference between AGMS (Fast-Count) and Fast-AGMS sketches unless the distribution of the interval stream is also skewed towards large dyadic intervals. The reason for this lies in the fact that since the point stream over the dyadic domain is skewed and the behavior

of the sketch estimators for the size of join of two streams with different Zipf coefficients is governed by the smallest skew factor, the overall behavior is determined by the skew of the interval stream. Figure 5-11 shows a significant difference between FC and F-AGMS only when both streams are skewed.The experimental results in Section 6.6 verify these hypotheses.

An evident drawback of DMAP is that it cannot be extended easily to the case when both input data streams are given as intervals. If the sketches are simply updated with the dyadic intervals in the minimal dyadic cover, the size of join of the points in the dyadic domain is computed which is different from the size of join in the original domain because a point in the dyadic domain corresponds to a range of points in the original domain. Updating one of the sketches with the product of the size of the dyadic interval and the frequency instead of only the frequency seems to be an easy fix that would compensate for the reduction in the representation. This is not the case because a point can be part of different dyadic intervals with different sizes, situation that is not caught by moving in the dyadic domain.

6.3.3 Algorithm DMAP COUNTS

A possible improvement to the basic DMAP method is to keep exact counts for large dyadic intervals in both streams and to compute sketches only for the rest of the data. By doing this, the distribution of the point stream in the dyadic domain becomes closer to the original distribution since the effect of the outliers is neutralized. The contribution of the large dyadic intervals to the size of join is computed exactly through the counts, while the contribution of the rest of dyadic intervals is better approximated through the sketches. Although the evident resemblance between this technique and other types of complex sketches, e.g., count sketches [11], skimmed sketches [28], and red sketches [29], there is a subtle difference. While for all the other techniques the high frequencies have to be determined and represent an important fraction of the result, in this case they are known before and represent an outlier whose effect has to be minimized. In order to quantify

the error of this method and to determine the optimal number of exact counts, similar solutions to [11, 29] can be applied with the added complexity of dealing with interval distributions over a dyadic domain. The deeper insights such an analysis could reveal are hard to determine since even the exact behavior of DMAP is only loosely quantified in [18, 51]. The empirical results we provide in Section 6.6 show that the improvement is effective.

6.4 Fast Range-Summable Generating Schemes

In Section 6 we have seen that DMAP is the state-of-the-art solution for sketching intervals. DMAP uses dyadic mappings in order to reduce the size of an interval, thus decreasing the number of random variables that have to be generated. The fast range-summation property is the ability to compute the sum of random variables in an interval in time sub-linear in the size of the interval – the alternative is to generate and sum-up the values ξ_i for each i in the interval. This property is a characteristic of the generating scheme and it is formally defined in [9]:

Definition 8. *A generating scheme for two-valued k-wise independent random variables is called **fast range-summable** if there exists a polynomial-time function g such that*

$$g([\alpha, \beta], S) = \sum_{\alpha \leq i \leq \beta} \xi_i(S) = \sum_{\alpha \leq i \leq \beta} (-1)^{f(S,i)} \tag{6–2}$$

where α, β, and i are values in the domain I, with $\alpha \leq \beta$.

Computing the function g over general $[\alpha, \beta]$ intervals is usually not straightforward. The task is simpler for dyadic intervals (see Section 6.3.1) due to their regularity. Fortunately, any scheme that is fast range-summable for dyadic intervals can be extended to general intervals $[\alpha, \beta]$ by simply determining the minimal dyadic cover, computing the function g over each dyadic interval in the cover, and then summing-up these results. Since the decomposition of any $[\alpha, \beta]$ interval contains at most a logarithmic number of dyadic intervals, fast range-summable algorithms for dyadic intervals remain fast range-summable for general intervals. Notice how dyadic intervals are used in different

ways for fast range-summation and in DMAP. While DMAP represents an interval by its minimal dyadic cover and random variables are generated for each dyadic interval in the cover, dyadic intervals speed-up the summing of the random variables in fast range-summation methods.

In this section, we study the fast range-summation property of the generating schemes presented in Section 3.1. For 2-wise independent random variables, both the BCH3 scheme and its EH3 variant are fast range-summable. [8] show that the Toeplitz family of hash functions is fast range-summable. Since for two-valued random variables the Toeplitz scheme is equivalent with BCH3, the algorithm we introduce for BCH3 also applies for the Toeplitz scheme. For the 4-wise case, the Reed-Muller generating scheme is the only scheme known to be fast range-summable [9, 32].

As suggested in [43], reducing the range-summing problem to determining the number of boolean variables assignments that satisfy an XOR-AND logical expression, and using the results in [23], is a formal method to determine if a generating scheme is fast range-summable. We apply this method to show that neither BCH5 nor the polynomial schemes are fast range-summable.

6.4.1 Scheme BCH3

We prove that BCH3 is fast range-summable and we provide an average case constant time $O(1)$ algorithm for computing the sum of the ± 1 random variables in any $[\alpha, \beta]$ interval. The BCH3 generating function f has the following form:

$$f(S, i) = [s_0, S_0] \cdot [1, i] = s_0 \oplus \bigoplus_{k=0}^{n-1} S_{0,k} \odot i_k \tag{6–3}$$

where $\oplus$ denotes the bitwise XOR, and $\odot$ denotes the bitwise AND. Both S_0 and i are vectors over the space $GF(2)^n$. Given two vectors in $GF(2)^n$, α and β, with $\alpha \leq \beta$, when interpreted as binary numbers, the problem of fast range-summing function f over the

interval $[\alpha, \beta]$ is to compute the function g defined below:

$$\begin{aligned} g([\alpha,\beta],S) &= \sum_{\alpha\le i\le\beta} (-1)^{f(S,i)} \\ &= \sum_{\alpha\le i\le\beta} (-1)^{[s_0,S_0]\cdot[1,i]} = \sum_{\alpha\le i\le\beta} (-1)^{s_0\oplus\bigoplus_{k=0}^{n-1} S_{0,k}\odot i_k} \end{aligned} \tag{6–4}$$

Notice that the operations that involve the seed and the index variable are over $GF(2)$, but the summation over the interval $[\alpha, \beta]$ is in $\mathbb{Z}$. Initially, we consider that the interval $[\alpha, \beta]$ is dyadic. This means that it has the form $[q2^j, (q+1)2^j)$, with $0 \le j \le n$ and $0 \le q \le 2^{n-j} - 1$. Lemma 4 shows how to evaluate the function g for the BCH3 scheme over a dyadic interval. The results in Proposition 1 and 2 are used for the subsequent proofs.

Lemma 4. *Let $[q2^j, (q+1)2^j)$ be a dyadic interval with $1 \le j \le n$ and $0 \le q \le 2^{n-j} - 1$, and let function g be defined as:*

$$g([q2^j,(q+1)2^j),S) = \sum_{i=q2^j}^{(q+1)2^j-1} (-1)^{f(S,i)} \tag{6–5}$$

where $S = [S_0, s_0]$, with S_0 a vector in $GF(2)^n$ and $s_0 \in \{0,1\}$, and function f is defined as $f([S_0,s_0],i) = s_0 \oplus \bigoplus_{k=0}^{n-1}(S_{0,k} \odot i_k)$. Then, function g can take the following values:

$$g([q2^j,(q+1)2^j),[S_0,s_0]) = \begin{cases} 0 & , \text{ if } 2^j \text{ does not divide } S_0 \\ 2^j \cdot (-1)^{f([S_0,s_0],q2^j)} & , \text{ if } 2^j \text{ divides } S_0 \end{cases} \tag{6–6}$$

Proof. Function f can be rewritten as follows for the interval $[q2^j, (q+1)2^j)$:

$$\begin{aligned} f([S_0,s_0],i) &= s_0 \oplus \bigoplus_{k=j}^{n-1}(S_{0,k}\odot i_k) \oplus \bigoplus_{k=0}^{j-1}(S_{0,k}\odot i_k) \\ &= C \oplus \bigoplus_{k=0}^{j-1}(S_{0,k}\odot i_k) \end{aligned} \tag{6–7}$$

where $C \in \{0,1\}$ is constant for a given S_0. This was possible because the most significant $n-j$ bits of a $[q2^j, (q+1)2^j)$ dyadic interval are identical for all the values in the interval.

Function g becomes:

$$g([q2^j,(q+1)2^j),S) = \sum_{i=q2^j}^{(q+1)2^j-1} (-1)^{C\oplus\bigoplus_{k=0}^{j-1}(S_{0,k}\odot i_k)} \tag{6–8}$$

Applying Proposition 2 for the expression $C \oplus \bigoplus_{k=0}^{j-1}(S_{0,k} \odot i_k)$ (provided that $S_{0,k} \neq 0$, $0 \leq k < j$), we know that the values 0 and 1 are taken equally often, thus the sum in function g gives the result 0. When all the j least significant bits of S_0 are equal to 0, that is, 2^j divides S_0, function f equals the constant C and function g can be written as:

$$\begin{aligned} g([q2^j,(q+1)2^j),S) &= \sum_{i=q2^j}^{(q+1)2^j-1} (-1)^C \\ &= 2^j \cdot (-1)^C \end{aligned} \tag{6–9}$$

where $C = s_0 \oplus \bigoplus_{k=j}^{n-1}(S_{0,k} \odot i_k)$. In order to evaluate C and function g, it is enough to compute function f for any of the points in the dyadic interval, for example $q2^j$. □

Lemma 4 gives a method to compute the sum of the BCH3 random variables for any dyadic interval $[q2^j,(q+1)2^j)$ in a constant number of operations. Notice that, for the majority of cases, there is no need to generate even one random variable in the interval to find the result of the summation. Only when S_0 is a multiple of 2^j a computation of the function f is required in order to compute the sum. The value of the sum in this particular case is either -2^j or 2^j.

Thus far, we have seen that it is possible to fast range-sum the BCH3 random variables over a dyadic interval by generating at most one of the variables in the interval. The generation is required only when the least significant j bits in S_0 are all equal to 0. When this is not the case, i.e., there is at least one bit equal to 1, we know the result of the summation without computing anything. Verifying that the least significant j bits in the binary representation of a value are all equal to 0 can be done efficiently on any processor, e.g., AND-ing with $2^{j+1}-1$. We conclude that, in order to fast range-sum BCH3 random variables over a dyadic interval $[q2^j,(q+1)2^j)$, it is enough to analyze the

values of the least significant j bits in S_0 and, sometimes, to generate one of the variables in the interval. Both these tasks can be carried out efficiently.

The general fast range-summing problem is to compute the function g defined in (6–4) for any interval $[\alpha, \beta]$, $\alpha \leq \beta$. As we know from Section 6.3.1, any interval can be represented as a union of dyadic intervals. By decomposing the interval $[\alpha, \beta]$ into its minimal dyadic cover $D([\alpha, \beta]) = \{\delta_1, \delta_2, \ldots, \delta_m\}$, function g can be rewritten as:

$$\begin{aligned} g([\alpha, \beta], S) &= \sum_{\alpha \leq i \leq \beta} (-1)^{f(S,i)} = \sum_{i \in \cup_{k=1}^{m} \delta_k} (-1)^{f(S,i)} \\ &= \sum_{i \in \delta_1} (-1)^{f(S,i)} + \sum_{i \in \delta_2} (-1)^{f(S,i)} + \cdots + \sum_{i \in \delta_m} (-1)^{f(S,i)} \end{aligned} \tag{6–10}$$

where δ_k, $1 \leq k \leq m$, are dyadic intervals. Algorithm *MinimalDyadicCover* 6.8 computes the minimal dyadic cover of an interval $[\alpha, \beta]$, while Lemma 4 shows how to efficiently compute each of the sums in (6–10). The straightforward implementation of these two steps results into a fast range-summable algorithm for the BCH3 generating scheme since the number of intervals in the minimal dyadic cover is logarithmic in the size of the interval and the computation of the sum over each of these intervals needs constant time. We go one step further and introduce an algorithm that overlaps these two steps, decomposition and summation. Besides overlapping these two stages, the algorithm we propose has strong early termination conditions that make it a constant time algorithm for the average case (over the seed space), improving considerably over the existing logarithmic fast range-summable algorithms.

Instead of computing function g directly over the interval $[\alpha, \beta]$, we write it as a difference over the intervals $[0, \beta + 1)$ and $[0, \alpha)$, $g([\alpha, \beta], S) = g([0, \beta + 1), S) - g([0, \alpha), S)$. The advantage of this form comes from the fact that the minimal dyadic cover of a $[0, \gamma)$ interval, $\gamma \in GF(2)^n$, can be directly determined from the binary representation of γ. If $\gamma = \gamma_{n-1}\gamma_{n-2}\ldots\gamma_0$, $\gamma_k \in \{0, 1\}$ for $0 \leq k < n$, the minimal dyadic cover of the interval $[0, \gamma)$ contains a size 2^k interval for every γ_k equal with 1. Based on this observation and on Lemma 4, Algorithm *BCH3Interval* 6.8 computes function g over any $[0, \gamma)$ interval.

Lines $1-2$ initialize the variables *sum* and $\gamma\prime$. *sum* stores the result of function g. $\gamma\prime$ is just a substitute of γ that is modified inside the algorithm. When γ is odd, that is, $D([0,\gamma)$ contains a point interval, function f is computed separately for it (Lines $3-5$). The for-loop in the Lines $6-15$ is the core of the algorithm. It detects the dyadic intervals in $D([0,\gamma))$ and computes the sum of the random variables inside them according to Lemma 4.

There exist two exit points from the algorithm inside the for-loop. The first one (Lines $8-9$) is straightforward: the algorithm returns when the range-sum is computed over the entire interval. In this case $\gamma\prime$ equals 0 since it is decremented after the detection of each interval in the minimal dyadic cover. The second exit point in the for-loop (Lines $10-11$) is more interesting. We know from Lemma 4 that the range-sum is determined without any computation for dyadic intervals that have non-zero corresponding bits in the seed S_0. We iterate through the intervals in the dyadic cover in ascending order of their size and the existence of a single one bit in S_0 automatically determines the range-sum for all subsequent intervals in the cover. This enables us to compute the value of function g without generating other random variables (we know that the sum is 0 in this case).

Lines $12-15$ correspond to the case when the least significant k bits in S_0 are all equal to 0. As shown in Lemma 4, a random variable inside the dyadic interval has to be generated in this case. The subtraction in Line 14 imposes that the random variable corresponding to the first point in the dyadic interval is generated. Line 16 contains the last exit point. It can be attained only when the seed S_0 is equal to 0 and γ_{n-1} equals to 1. Notice that the operations that appear in the algorithm imply computations with 2 and its powers and that they can be efficiently implemented using bit operations (AND, XOR, SHIFT, etc.).

Algorithm *BCH3* 6.8 shows how to correctly invoke *BCH3Interval* 6.8 for $[\alpha,\beta]$ intervals. Notice that the first call to *BCH3Interval* 6.8 is with $\beta+1$. This is necessary for the correct computation of function g.

Example 11. *We show how Algorithm BCH3 6.8 works for the interval* $[100, 202]$ *and the seeds* $s_0 = 0$, $S_0 = 184 = (10111000)_2$. sum_1 *is computed over the interval* $[0, 203)$, *while* sum_2 *is calculated over the interval* $[0, 100)$. *Their difference is returned. For* $203 = (11001011)_2$, *function* f *is invoked first outside the for-loop, with* $\gamma\prime = 202 = (11001010)_2$, sum_1 *taking the value* $(-1)^0 = 1$. f *is then invoked for* $\gamma\prime = 200 = (11001000)_2$, *giving the result* $2 \cdot (-1)^0 = 2$. sum_1 *is updated to* 3. *The last time* f *is invoked for the interval* $[0, 203)$, $\gamma\prime = 192 = (11000000)_2$ *and the returned result is* $2^3 \cdot (-1)^1 = -8$, *the value of* sum_1 *until this point being* -5. *At the next iteration* $k = 4$ *and* $S_{0,3} = 1$ *and the routine BCH3Interval returns the value* -5 *for* sum_1. *For* $100 = (01100100)_2$ *function* f *is called only once, with* $\gamma\prime = 96 = (01100000)_2$, sum_2 *taking the value* $2^2 \cdot (-1)^1 = -4$. *At the fourth iteration of the for-loop,* -4 *is returned for* sum_2. *Finally, the range-sum of the BCH3 function for the interval* $[100, 202]$ *is* $-5 - (-4) = -1$.

Theorem 14. *Let* $[\alpha, \beta]$, $\alpha \leq \beta$, *be an interval, where* $\alpha, \beta \in GF(2)^n$, *and seed* $S = [S_0, s_0]$ *be a vector over* $GF(2)^{n+1}$, *i.e.,* $S_0 \in GF(2)^n$ *and* $s_0 \in \{0, 1\}$. *Then, Algorithm BCH3 6.8 computes the sum of the BCH3 random variables over the interval* $[\alpha, \beta]$.

Proof. Instead of range-summing over the interval $[\alpha, \beta]$, we apply Algorithm *BCH3Interval* 6.8 over the intervals $[0, \beta + 1)$ and $[0, \alpha)$ and return the difference of these results.

Algorithm *BCH3Interval* 6.8 has two termination points: exhausting the entire interval $[0, \gamma)$ and detecting the first least significant one bit in S_0. For every interval that is part of the minimal dyadic cover $D([0, \gamma))$, γ is decreased accordingly. When the entire minimal dyadic cover is determined, γ equals 0 and the algorithm returns. This is the ordinary termination, for which the algorithm executes the for-loop a logarithmic number of times in the size of the $[0, \gamma)$ interval. A reduction to a constant number of executions of the for-loop is provided by the second return point, the detection of the first one bit in S_0. We know from Lemma 4 that the range-sum over dyadic intervals that depend on non-zero seeds S_0 can be computed without generating any random variable

in the interval. Applying this result to the current $[0, \gamma)$ interval that is a union of dyadic intervals that depend on a non-zero seed S_0, the range-sum can be immediately returned.

Algorithm *BCH3Interval* 6.8 returns the range-sum over $[0, \gamma)$ intervals. The minimal dyadic cover $D([0, \gamma))$ is encoded in the binary representation of γ and consists of the intervals that have decreasing sizes as we go from 0 toward γ. That is, for $D([0, \gamma)) = \{\delta_1, \delta_2, \ldots, \delta_{\gamma_m}\}$, $|\delta_1| > |\delta_2| > \cdots > |\delta_{\gamma_m}|$ holds. The range-sum over each δ_k dyadic interval, $1 \leq k \leq \gamma_m$, is computed according to Lemma 4. Since the intervals are detected in the increasing order of their sizes, it is possible to determine the value of the range-sum without any other computation, when the fast termination condition is met: the partial sum is 0, as stated in Lemma 4. □

Theorem 15. *Algorithm BCH3Interval 6.8 performs on expectation* 2 *computations of the function f, one outside the loop and one inside the for-loop, given that the seeds $S = [S_0, s_0]$ are 2-wise independent and uniformly distributed over $GF(2)^{n+1}$. For* 100 *seeds S, the average number of function f computations is between* 1.723 *and* 2.277*, while for* 1,000 *seeds it is inside the interval* [1.912, 2.088]*. For* 10,000 *seeds it lies between* 1.972 *and* 2.028*. All these results have a* 0.95 *confidence probability.*

Proof. We consider that γ has the worst value for our algorithm, e.g., $\gamma = \overbrace{1 \ldots 11}^{n}$. This way, it always generates a computation of the function f. We prove that the fast termination condition makes the algorithm to execute the for-loop 1 time, on expectation, giving a total of 2 function f computations.

For this, we have to determine the average number of 0 consecutive bits that appear in the least significant positions of S_0, knowing that S_0 is uniformly distributed over the space $GF(2)^n$. Since the bits in S_0 are independent, the probability of having exactly k consecutive bits with value 0, $1 \leq k \leq n$, preceded by a 1, is $\frac{1}{2^{k+1}}$. We define the random variable X as the number of least significant consecutive 0 bits in S_0 over the uniform probability space $GF(2)^n$. The expected value of X, $E[X]$, gives us exactly what we are

looking for, that is, the average number of least significant consecutive 0 bits in S_0.

$$\begin{aligned} E[X] &= 1 \cdot \frac{1}{2^2} + 2 \cdot \frac{1}{2^3} + \cdots + n \cdot \frac{1}{2^{n+1}} \\ &= \frac{1}{2} \sum_{k=1}^{n} k \cdot \left(\frac{1}{2}\right)^k \lessapprox 1 \end{aligned} \tag{6–11}$$

$E[X]$ is a derived power series with ratio $\frac{1}{2}$ that converges to 1. For a complete characterization of the random variable X, we also compute its variance, $Var[X]$, which tells us what is the range around the expected value variable X takes values in.

$$Var[X] = E[X^2] - E^2[X] \tag{6–12}$$

$$\begin{aligned} E[X^2] &= 1^2 \cdot \frac{1}{2^2} + 2^2 \cdot \frac{1}{2^3} + \cdots + n^2 \cdot \frac{1}{2^{n+1}} \\ &= \frac{1}{2} \sum_{k=1}^{n} k^2 \cdot \left(\frac{1}{2}\right)^k \lessapprox 3 \end{aligned} \tag{6–13}$$

$$\begin{aligned} Var[X] &= E[X^2] - E^2[X] \\ &= 3 - 1^2 \lessapprox 2 \end{aligned} \tag{6–14}$$

Both the expectation and the variance of X have finite values, e.g., 1 and, respectively, 2. For a large enough number of seeds S, e.g., greater than 100, the central limit theorem [54] can be applied. Let the random variable Y be defined as:

$$Y = \frac{\sum_{k=1}^{N} X_k}{N} \tag{6–15}$$

where N is the number of seeds S, $N \geq 100$. Using the central limit theorem, we know that random variable Y can be approximated by a normal distribution with parameters $E[Y] = E[X]$ and $Var[Y] = \frac{Var[X]}{N}$. We determine the range of variable Y within a 0.95 confidence interval using the cumulative distribution function (cdf) of the normal distribution that approximates Y.

$$Y_l = 1 + \frac{2}{\sqrt{N}} \cdot \text{Erf}^{-1}(-0.95) \tag{6–16}$$

$$Y_r = 1 + \frac{2}{\sqrt{N}} \cdot \text{Erf}^{-1}(0.95) \tag{6–17}$$

Y_l and Y_r are the left and right interval end-points, while Erf^{-1} is the inverse of the error function [54]. Replacing N with 100, 1,000 and 10,000 in the above formulae, and adding 1 for the out of loop function f computation, we obtain the results stated in the theorem. □

In Theorem 15 we considered that γ takes the worst value for our algorithm, e.g., $\gamma = \overbrace{1 \ldots 11}^{n}$. In practice, this does not happen too often and the number of function f computations could be smaller. If we assume that the bits in γ take the values 0 and 1 with the same probability, the number of computations of f reduces to half, i.e., 1. Also, there exist cases when no computation of f is done, e.g., $\gamma_0 = 0$ and $S_{0,0} = 1$, i.e., the least significant bits in γ and S_0 are 0 and, respectively, 1.

Corollary 5. *Function f is computed on expectation 4 times when BCH3 random variables are range-summed over $[\alpha, \beta]$ intervals.*

Proof. Algorithm *BCH3Interval* 6.8 is called two times by Algorithm *BCH* 6.8 and each invocation computes function f 2 times on expectation. □

The result in Corollary 5 is for interval end-points α and β that are worst for our algorithm, e.g., both α and $\beta + 1$ end in 11. In the average case, with the bits taking the values 0 and 1 uniformly probable, the number of function f computations reduces to 2. This result is the best we could hope for, that is, computing the range-sum over an interval $[\alpha, \beta]$ by taking into consideration only its end-points, α and β.

6.4.2 Scheme EH3

Although [25] show that the random variables generated using the Extended Hamming scheme (EH3) are fast range-summable, the algorithm contained in the proof is abstract and not appropriate for implementation purposes. We propose a practical algorithm for the fast range-summation of the EH3 random variables. It is an extension of our constant-time algorithm for fast range-summing BCH3 random variables.

The following theorem provides an analytical formula for computing the range-sum function g. Notice that only one computation of the generating function f is required in order to determine the value of g over any dyadic interval.

Theorem 16. *Let $[q4^j, (q+1)4^j)$ be a dyadic interval*[1] *with size at least 4, $j \geq 1$. The range-sum function $g([q4^j, (q+1)4^j), S) = \sum_{i=q4^j}^{(q+1)4^j} (-1)^{f(S,i)}$ defined for EH3 scheme is equal to:*

$$g([q4^j, (q+1)4^j), S) = (-1)^{\#ZERO} \cdot 2^j \cdot (-1)^{f(S,q4^j)} \tag{6–18}$$

where #ZERO represents the number of two adjacent pair bits that OR to 0 (i.e., the number of groups of 00 bits).

Proof. The generating function f can be written for $i \in [q4^j, (q+1)4^j)$ as follows:

$$\begin{aligned} f(S,i) = f(S, q4^j) \oplus \\ S_{0,2j-1} \odot i_{2j-1} \oplus S_{0,2j-2} \odot i_{2j-2} \oplus i_{2j-1} \odot i_{2j-2} \oplus \cdots \oplus \\ S_{0,1} \odot i_1 \oplus S_{0,0} \odot i_0 \oplus i_1 \odot i_0 \end{aligned} \tag{6–19}$$

where the first part is fixed, while the last $2j$ bits of i take all the possible values. When the value of the changing expression is 0, $f(S,i) = f(S, q4^j)$, while when its value is 1, $f(S,i)$ is the negation of $f(S, q4^j)$. We know that any expression of the form $S_{0,j} \odot i_j \oplus S_{0,j-1} \odot i_{j-1} \oplus i_j \odot i_{j-1}$ has a $3:1$ distribution of the values, depending on the seed bits. Thus, the sum $\sum(S_{0,j} \odot i_j \oplus S_{0,j-1} \odot i_{j-1} \oplus i_j \odot i_{j-1})$ takes either the value 2 or -2 when i_j and i_{j-1} take all the possible values. When j blocks of this form are combined together, the resulting sum will be 2^j or -2^j. For one block, the sum is 2 only when $S_{0,j} \vee S_{0,j-1} = 0$, i.e., $S_{0,j} = S_{0,j-1} = 0$. For multiple blocks that are XOR-ed, the result is 1 only when an odd number of them takes the value 1. This implies that in order to obtain the result 2^j, $S_{0,j} = S_{0,j-1} = 0$ has to be valid for an odd number of blocks. If we denote by #ZERO the

[1] Although we call them dyadic intervals, these intervals are defined over powers of 4.

number of blocks for which the relation $S_{0,j} = S_{0,j-1} = 0$ holds, the range-sum function g can be written as:

$$g([q4^j, (q+1)4^j), S) = (-1)^{\#\text{ZERO}} \cdot 2^j \cdot (-1)^{f(S,q4^j)} \tag{6–20}$$

□

Based on the results in Theorem 16, Algorithm *EH3Interval* 6.8 computes function $g([\alpha, \beta], S) = \sum_{\alpha \le i \le \beta} (-1)^{f(S,i)}$ for any interval $[\alpha, \beta]$. First, the minimal dyadic cover of $[\alpha, \beta]$ is determined, then the sum over each dyadic interval is computed using (6–18). Notice that these two steps can be combined, the computation of g being performed while determining the minimal dyadic cover of $[\alpha, \beta]$. The minimal dyadic cover can be efficiently determined from the binary representation of α and β. Since any interval can be decomposed into a logarithmic number of dyadic intervals, algorithm *EH3Interval* 6.8 computes function g in $O(\log(\beta - \alpha))$ steps.

Example 12. *We show how Algorithm* EH3Interval *6.8 works for the interval* $[124, 197]$ *and the seed* $S = [s_0, S_0] = [0, 184 = (10111000)_2]$. *The minimal dyadic cover of* $[124, 197]$ *is*

$$D([124, 197]) = \{[124, 128), [128, 192), [192, 196), [196, 197), [197, 198)\} \tag{6–21}$$

#ZERO is equal with 1 *for the given* S_0, *the only pair OR-ing to* 0 *being the pair at the end. It affects the dyadic intervals with powers greater than* 0.

$$\begin{aligned} g([124, 197], S) &= g([124, 128), S) + g([128, 192), S) + g([192, 196), S) + \\ &\quad g([196, 197), S) + g([197, 198), S) \\ &= -2^1 \cdot (-1)^{f(S,124)} - 2^3 \cdot (-1)^{f(S,128)} - 2^1 \cdot (-1)^{f(S,192)} + \\ &\quad 2^0 \cdot (-1)^{f(S,196)} + 2^0 \cdot (-1)^{f(S,197)} \\ &= 2 + 8 + 2 + 1 - 1 = 12 \end{aligned} \tag{6–22}$$

While fast range-summing BCH3 random variables is constant-time on average, the EH3 algorithm is logarithmic in the size of the interval. Although both algorithms

compute partial sums over the dyadic intervals in the minimal dyadic cover, the BCH3 algorithm can return the final result after only a small number of function f invocations. This is not the case for EH3, which requires one invocation for each dyadic interval in the minimal dyadic cover.

6.4.3 Four-Wise Independent Schemes

In this section we investigate the fast range-summation property of the 4-wise independent generating schemes presented throughout this work, namely BCH and polynomials over primes. The discussion regarding the Reed-Muller scheme is deferred to the next section.

The main idea in showing that some of the schemes are not fast range-summable is to use the result in [23] on the problem of counting the number of times a polynomial over $GF(2)$, written as XOR of ANDs (sums of products with operations in $GF(2)$), takes each of the two values in $GF(2)$, as suggested by [43]. The result states that the problem is #P-complete if any of the terms of the polynomial written as an XOR of ANDs contains at least three variables. In our particular case, to show that a scheme is not fast range-summable, it is enough to prove that for some seed S the generating function $f(S, i)$, written as an XOR of ANDs polynomial in the bits of i, contains at least one term that involves three or more variables. The following results use this fact to show that the BCH5 and the polynomials over primes schemes are not fast range-summable.

Theorem 17. *BCH schemes are not fast range-summable for $k \geq 5$ and $n \geq 4$.*

Proof. We show that fast range-summing BCH5 random variables is equivalent to determining the number of assignments that satisfy a 3XOR-AND boolean formula, problem that is known to be #P-complete. Since for $k > 5$ any BCH scheme can be reduced to BCH5 for some particular values of the seed, i.e., $S_2 = S_3 = \cdots = S_{k/2} = \bar{0}$, if BCH5 is not fast range-summable implies that BCHk is not fast range-summable, for $k > 5$.

BCH5 necessitates the computation of i^3 over the extension field $GF(2^n)$. If we consider the bit representation of i, it can be shown that i^3 is a 3XOR-AND boolean formula for $n > 3$ (the idea is to use the bit equations of a multiplier). This reduction implies that BCH5 is not fast range-summable. □

Theorem 18. *Let $n = \lceil \log p \rceil$ be the number of bits the prime $p > 7$ can be represented on and $[q2^l, (q+1)2^l)$ be a dyadic interval with $l \geq 3$. Then, the function $f(S, i) = [(a_0 + a_1 i) \mod p] \mod 2$ is not fast range-summable over the interval $[q2^l, (q+1)2^l)$.*

Proof. Without loss of generality, we consider that $a_0 = 0$, $a_1 = p - 1$, and show that function g is a 3XOR-AND formula, implying that it is not fast range-summable. The expression $(p-1)\bar{i} \mod p$ takes the values $0, p-1, p-2, \ldots, p-7$ for $\bar{i} = 0, 1, \ldots, 7$. Out of these values, only $p-2, p-4,$ and $p-6$ are odd, function g taking the value 3. If we express the result of function g as an XOR-AND formula in terms of $i_0, i_1,$ and i_2, we obtain:

$$g(i_2 i_1 i_0, S) = i_1 \oplus i_2 \oplus i_1 \odot i_0 \oplus i_2 \odot i_0 \oplus i_2 \odot i_1 \oplus i_2 \odot i_1 \odot i_0 \tag{6–23}$$

which is a 3XOR-AND formula. We know that the number of assignments to i_2, i_1, i_0 that satisfy this formula cannot be determined in polynomial time. This implies that function f is not fast range-summable over dyadic intervals of size at least 8. □

Theorem 18 shows that for the polynomials over primes scheme with $k = 2$ there exist values for the coefficients a_0 and a_1 that make the scheme not fast range-summable for dyadic intervals with size greater or equal than $2^3 = 8$. Since the schemes for $k > 2$ can be reduced to $[(a_0 + a_1 i) \mod p] \mod 2$ by making $a_2 = \cdots = a_{k-1} = 0$, it results that the polynomials over primes scheme is not fast range-summable for any $k \geq 2$.

6.4.4 Scheme RM7

Together with the negative result on the hardness of counting the number of times an XOR of ANDs polynomial with terms containing more that three variables AND-ed, [23] provided an algorithm for such counting for formulae that contain only at most two

variables AND-ed in each term. This algorithm, that we refer to as 2XOR-AND, can be readily used to produce a fast range-summable algorithm for the 7-wise independent Reed-Muller (RM7) scheme. An algorithm for this scheme based on the same ideas was proposed in [9]. We focus our discussion on the 2XOR-AND algorithm, but the same conclusions are applicable to the algorithm in [9].

The observation at the core of the 2XOR-AND algorithm is the fact that polynomials with a special shape are fast range-summable. These are polynomials with at most two variables AND-ed in any term and with each variable participating in at most one such term. The other cases can be reduced to this case by introducing new variables that are linear combinations of the existing ones. To determine these linear combinations in the general case, systems of linear equations have to be constructed and solved, one for each variable. The overall algorithm is $O(n^3)$, with n the number of variables, if the summation is performed over a dyadic interval.

The 2XOR-AND algorithm can be used to fast range-sum random variables produced by the RM7 scheme since in the XOR of ANDs representation of this scheme as a polynomial of the bits of i (which is the representation used in Section 3.1) only terms with ANDs of at most two variables appear. Using the 2XOR-AND algorithm for each dyadic interval in the minimal dyadic cover of a given interval, the overall running time can be shown to be $O(n^4)$ where the size of I, the domain, is 2^n. While this algorithm is clearly fast range-summable using the definition, in practice it might still be too slow to be useful. Indeed this is the case, as it is shown in the empirical evaluation section where we provide a running time comparison of the fast range-summable algorithms.

6.4.5 Approximate Four-Wise Independent Schemes

Since the RM7 fast range-summable algorithm is not practical and fast range-summable algorithms for BCH5 and polynomials over primes schemes do not exist, it is worth investigating approximation algorithms for the 4-wise case. While such approximations are

possible [40, 44], we show that they are not more practical than the exact algorithm for RM7.

Let *xaf* be a multivariate polynomial in the variables $x_1, x_2, \ldots, x_n$ over $GF(2)$ and having the form:

$$xaf(x_1, x_2, \ldots, x_n) = t_1(x_1, x_2, \ldots, x_n) \oplus \cdots \oplus t_m(x_1, x_2, \ldots, x_n) \tag{6–24}$$

where for each $j = 1, 2, \ldots, m$, the term $t_j(x_1, x_2, \ldots, x_n)$ is the product of a subset of the variables $x_1, x_2, \ldots, x_n$. The approximate XOR-AND counting algorithm introduced in [40] needs $4m^2 \ln(\frac{2}{\delta})\frac{1}{\epsilon^2}$ random trials – particular value assignments to all of the variables – to provide a result with relative error at most ϵ with probability at least $1 - \delta$. Each of these trials necessitates the evaluation of the polynomial xaf at the particular assigned value. In order to have an efficient approximate algorithm, the number of trials it evaluates should be small. We show that for obtaining a good approximation of the range-sum problem, the number of trials is comparable with the size of the interval even for the RM7 scheme, thus making the algorithm impractical.

Figure 6-1 represents the number of evaluations of the polynomial xaf for the RM7 scheme. The ratio between the number of linear point-by-point evaluations and the number of evaluations invoked by the approximate algorithm in [40] is plotted for a number of variables that ranges between 1 and 32. The number of terms m in the RM7 polynomial is $\frac{n(n+1)}{2}$. In the average case, half of these terms are simplified by corresponding 0 seeds, giving a total of $\frac{n(n+1)}{4}$ terms. In order to obtain a satisfactory approximation of the XOR-AND counting result, we set the value of the factor $\ln(\frac{2}{\delta})\frac{1}{\epsilon^2}$ to 10^3. All these give the value $10^3 \cdot \frac{n^2(n+1)^2}{4}$ for the number of trials employed by the approximate XOR-AND counting algorithm. As the number of polynomial evaluations for the linear case is 2^n, the function plotted in Figure 6-1 is $\frac{2^n}{10^3 \cdot \frac{n^2(n+1)^2}{4}}$, where n takes values in the range $[1 \ldots 32]$.

The results in Figure 6-1 are not very encouraging. Only when the number of variables is greater than 25, the approximate algorithm needs less polynomial evaluations than the direct exhaustive method. But this implies intervals of extremely large size that are unlikely to appear in practice. If we also take into account the fact that the provided result is not exact and its error could propagate exponentially, we think that the approximate XOR-AND counting algorithm is not applicable to the range-summing problem.

6.4.6 Empirical Evaluation

We implemented the fast range-summable algorithms for the BCH3, EH3, and RM7 schemes and we empirically evaluated them with the same experimental setup as in Section 3.1.10. The evaluation procedure is based on 100 experiments that use a number of randomly generated intervals and an equal number of sketches chosen for each method such that the overall running time is in the order of minutes in order to obtain stable estimates of the running time per sketch. The results, depicted in Table 6-1, are the average of the 100 runs and have errors of at most 5%. Notice that the execution time of BCH3 for ranges is merely 7 times larger than the execution time for a single sketch (refer to Table 3-3 for the running times of individual sketches). This happens because, as we mentioned earlier, our algorithm for BCH3 is essentially $O(1)$. The Extended Hamming scheme EH3 has an encouraging running time of approximately 1.8 μs, thus about $550,000$ such computations can be performed per second on a modern processor. The RM7 fast range-summable algorithm is completely impractical since only about 40 computations can be performed per second. This is due to the fact that the algorithm is quite involved (a significant number of systems of linear equations have to be formed and solved). Even if special techniques are used to reduce the running time, at most a 32 factor reduction is possible and the scheme would be still impractical.

The net effect of these experimental results and of the theoretical discussions in this section is that there is no practical fast range-summable algorithm for any of the known 4-wise generating schemes, while the algorithm for BCH3 is extremely efficient.

6.5 Fast Range-Summation Method

While DMAP uses mappings in the dyadic domain in order to sketch intervals efficiently, fast range-summation methods are based on properties of the random variables that allow the sketching of an interval in a number of steps sub-linear in the size of the interval. Specifically, the sum of random variables over dyadic intervals is computed in a constant number of steps and, since there exists a logarithmic number of dyadic intervals in the minimal dyadic cover of any interval, the number of steps to sketch the entire interval is logarithmic in its size. [51] show that fast range-summation is a property of the generation scheme of the random variables and that there exist only two practical schemes applicable to AGMS sketches, EH3 and BCH3, respectively. Moreover, the performance of BCH3 is highly sensitive to the input data, so we consider only EH3 in this chapter. [51] use fast range-summation only in the context of AGMS sketches where the update of each key (interval) affects each counter in the sketch structure. More exactly, for all the elements in an interval, the same counter has to be updated (and all the counters overall). This is not the case anymore for hash-based sketches where different counters are updated for different keys (unless they are hashed into the same bucket). In this section we show that fast range-summation and random hashing are conflicting operations and, consequently, fast range-summation is not applicable to hash-based sketches (Fast-AGMS, Fast-Count, Count-Min). Fortunately, we show that fast range-summation for AGMS sketches can be applied in conjunction with deterministic partitions of the domain without loss in error, but with a significant improvement in the update time.

As mentioned in Section 4.2, the main drawback of AGMS sketches is the update time. For each stream element, each counter in the sketch structure has to be updated. Essentially, each counter is a randomized synopsis of the entire data. Fast range-summation

exploits exactly this additive property to sketch intervals efficiently since the update corresponding to each element in the interval has to be added to the same counter. Hash-based sketches partition randomly the domain I of the key attribute and associate a single counter in the sketch structure with each of these partitions. For each stream element, only the counter corresponding to its random partition is updated, thus the considerable gain in update time. In order to determine if fast range-summation can be extended to hash-based sketches, we focus on the interaction between random hashing, whose goal is to partition evenly the keys into buckets, and the efficient sketching of continuous intervals for which the maximum benefit is obtained when all the elements in the interval are placed into the same bucket. The following proposition relates the number of counters that have to be updated in a hash-based sketch to the size of the input interval:

Proposition 11. *Given a hash function $h : I \rightarrow B$ and an interval $[\alpha, \beta]$ of size l, the number of buckets touched by the function h when applied to the elements in $[\alpha, \beta]$ is on expectation $B\left[1 - \left(1 - \frac{1}{B}\right)^l\right]$.*

Proof. Let X_i be a 0/1 random variable corresponding to each of the B buckets of the hash function h, $0 \leq i < B$. X_i takes the value 1 when at least one element in the range $[\alpha, \beta]$ is hashed into the bucket i and the value 0 otherwise:

$$X_i = \begin{cases} 1 & , \quad \text{if } \exists j \in [\alpha, \beta] \text{ with } h(j) = i \\ 0 & , \quad \text{otherwise} \end{cases} \tag{6–25}$$

The expected value $E[X_i]$ can be computed as:

$$E[X_i] = P[X_i = 1] = 1 - P[X_i = 0] = 1 - \left(1 - \frac{1}{B}\right)^l \tag{6–26}$$

since the probability of an element to be hashed in the i^{th} bucket is $\frac{1}{B}$. The expected value of the number of buckets touched by h over $[\alpha, \beta]$ is then:

$$E[X] = E\left[\sum_{i=0}^{B-1} X_i\right] = \sum_{i=0}^{B-1} E[X_i] = B\left[1 - \left(1 - \frac{1}{B}\right)^l\right] \tag{6–27}$$

where X is defined as $X = \sum_{i=0}^{B-1} X_i$. □

In order to give some practical interpretation to the above proposition, we consider the size l to be proportional with the number of buckets B, i.e., $l = kB$, for $k > 0$. This allows us to rewrite Equation 6–27 as:

$$B\left[1 - \left(1 - \frac{1}{B}\right)^l\right] = B\left[1 - \left(1 - \frac{1}{B}\right)^{kB}\right] \approx B\left(1 - \frac{1}{e^k}\right) \tag{6–28}$$

where we used the approximation $\frac{1}{e} = \lim_{B\to\infty}\left(1 - \frac{1}{B}\right)^B$.

Corollary 6. *For a hash function with B buckets, 63.21% of the buckets are touched on expectation when h is applied to an interval of size B. The number of buckets increases to 86.46% when the size of the interval is twice the number of buckets B, and to 98.16% for $k = 4$.*

The above corollary states that for intervals of size at least four times the size of the hash almost all the buckets are touched on expectation. This eliminates completely the effect of hashing for sketching intervals since AGMS sketches already require the update of each counter in the sketch. The difference is that for AGMS sketches each counter is updated with the entire interval, while for hash-based sketches a counter is updated only with the elements in the interval assigned to the random partition corresponding to that counter. Although the number of updates per counter is smaller for hash-based sketches, determining how many (and which) elements in the given interval update the same counter without looking at the entire interval is a difficult problem. The only solution we are aware of is for 2-universal hash functions, so applicable only to Fast-AGMS and Count-Min sketches. It consists in applying the sub-linear algorithm proposed in [1] for

counting how many elements in the interval are hashed into a range of buckets either for each bucket or for ranges of increasing size. Notice that this actually is not even enough for Fast-AGMS sketches for which the interaction between hashing and EH3 (or BCH3) [51] has to be quantified. While fast range-summation takes advantage of properties of the generating scheme for the particular form of dyadic intervals, determining the contribution of the elements in the same random partition without considering each element separately has to be more difficult due to the lack of structure. Consequently, fast range-summation is directly applicable only to Count-Min sketches throughout the hash-based sketching techniques, with the requirement that each counter is updated when sketching an interval.

Fast Range-Summation with Domain Partitioning: The intermediate solution between AGMS sketches, which update all the counters, and hash-based sketches, which update only one counter for a given key, is sketch partitioning [20]. The domain I is partitioned in continuous blocks rather than random blocks. A number of counters from the sketch structure proportional to the size of the block is assigned to each block. When the update of a key has to be processed, only the counters in the block corresponding to that key are updated. This method can be easily extended to fast range-summing intervals without the need to update all the counters unless the size of the interval is close to the size of the domain. The intersection between the given interval and each partition is first determined and, for each non-empty intersection, the fast range-summation algorithm is applied only to the set of associated counters. Thus, a number of counters proportional with the number of non-empty intersections (and indirectly proportional to the size of the interval) has to be updated. In what follows we provide an example to illustrate how fast range-summation with sketch partitioning works.

Example 13. *Consider the domain* $I = \{0, \ldots, 15\}$ *to be split into* 4 *equi-width partitions as depicted in Figure 6-4. For simplicity, assume that the available AGMS sketch consists of* 8 *counters which are evenly distributed between the domain partitions,* 2 *for each*

partition. Instead of having a single estimator for the entire domain, a sketch estimator combining the counters in the partition is built for each partition. The final estimator is the sum of these individual estimators corresponding to each partition.

Figure 6-4 depicts the update procedure for the interval $[2, 7]$*. The non-empty intersections* $[2, 3]$ *and* $[4, 7]$ *correspond to partition* 0 *and* 1*, respectively. The fast range-summation algorithm is applied to each of these intervals only for the counters associated with the corresponding domain partition, not for all the counters in the sketch. In our example, fast range-summation is applied to interval* $[2, 3]$ *and the two counters associated to partition* 0*, and to interval* $[4, 7]$ *and the two counters associated to partition* 1*, respectively. Overall, only four counters are updated, instead of eight, for sketching the interval* $[2, 7]$*.*

The advantage of domain partitioning is the fact that the update time is smaller when compared to the basic fast range-summation method. This is the case because only a part of the sketch has to be updated if the interval is not too large with respect to the size of a partition. In particular, only the sketches corresponding to the partitions that intersect the interval need to be updated which means that the speedup is proportional to the ratio between the number of partitions and the average number of partitions an interval intersects. When points are sketched, only the counters corresponding to the partition the point belongs to need to be updated instead of all the counters in the sketch. In the above example only two counters have to be updated for each point, instead of eight.

Notice that, as shown in [20], any partitioning of the domain can be used and the number of counters associated to each partition can also be different from partition to partition. More precisely, any partitioning and any allocation scheme for the counters results in an unbiased estimator for the size of join. An important question though is what is the variance of the estimator, which is an indicator for the accuracy. In [20] a sophisticated method to partition and allocate the counters per partition was proposed in order to minimize the variance of the estimator. For gains to be obtained, regions of the

domain where high frequencies in one stream match small frequencies in the other have to be identified. Since in this particular situation we do not expect large frequencies for the interval stream, as explained in Section 6.3, we do not expect the sketch partitioning technique in [20] to be able to reduce the variance significantly. Moreover, using the fact that the variance of the estimator remains the same if the partitioning is random (see Proposition 3), as long as there does not exist significant correlation between the partitioning scheme and the input frequencies, we expect the variance of the estimator to remain the same. The expected distribution of the interval frequencies also suggests that a simple equi-width partitioning should behave reasonably well. Indeed, the experimental results in Section 6.6 show that this partitioning is effective in reducing the update time while the error of the estimate remains roughly the same.

6.6 Experimental Results

In this section we present the results of the empirical study designed to evaluate the performance of five of the algorithms for efficiently sketching intervals introduced previously. The five methods tested are: AGMS DMAP, F-AGMS DMAP (F-AGMS), F-AGMS DMAP with exact counts (F-AGMS COUNTS), fast range-summation AGMS (AGMS), and fast range-summation AGMS with sketch partitioning (AGMS P). Methods based on Count-Min sketches are excluded due to their high sensitivity to the input data, while for Fast-Count sketches the same behavior as for AGMS is expected (see Section 4.4), where applicable. The accuracy and the update time per interval are the two quantities measured in our study for the size of spatial join problem involving intervals (see [18]). But first, we want to compare EH3 with DMAP [18] (see Section 6.3) on the two applications introduced in Section 6.1, namely, size of spatial joins and selectivity estimation for histograms construction. The comparison with BCH3 is omitted since its error is significantly higher when compared to EH3 or DMAP. We do not perform comparisons with the fast range-summable version of the Reed-Muller scheme (RM7) since

its throughput is not higher than 40 sketch computations per second (EH3 is capable of performing more than 550,000 sketch computations per second as shown in Section 6.4).

Following the experimental setup in [18, 51], three real data sets are used in our experiments: LANDO, describing land cover ownership for the state of Wyoming and containing 33,860 objects; LANDC, describing land cover information such as vegetation types for the state of Wyoming and containing 14,731 objects; and SOIL, representing the Wyoming state soils at a $1:10^5$ scale and containing 29,662 objects. The use of synthetic generators for interval data is questionable because it is not clear what are acceptable distributions for the size of the intervals, as well as the position of the interval end-points. In a similar manner to Section 4.4, each experiment is performed 100 times and either the average relative error, i.e., $\frac{|\text{actual}-\text{estimate}|}{\text{actual}}$, or the median update time over the number of experiments is reported.

EH3 vs DMAP for Spatial Joins: We used the same experimental setup as in [18] to compare EH3 and DMAP for approximating the size of spatial joins. The average error for estimating the size of spatial joins for both EH3 and DMAP is depicted in Figure 6-5, 6-6, and 6-7. The sketch size varies between 4 and 40 K words of memory. Notice that in all the experiments EH3 significantly outperforms DMAP by as much as a factor of 8. This means that DMAP needs as much as 64 times more memory in order to achieve the same error guarantees.

Table 6-2 contains the timing results for sketching intervals using the two practical fast range-summable schemes (BCH3 and EH3) and the DMAP method. We present the average time for sketching an interval from each of the real datasets. As expected, BCH3 is the fastest scheme both for sketching only the interval, but also for sketching the interval as well as its end-points. The time to sketch an interval using DMAP is about half the time used by the EH3 fast range-summable algorithm. When sketching both the interval and its end-points, as is the case for the size of join between an interval relation and a point relation, the ratio reverses – DMAP uses twice as much time as EH3. This

happens because DMAP uses a logarithmic number of points (in the size of the domain) to represent each interval end-point, while EH3 has to generate only two random variables, one for each end-point.

The difference between the results reported in Table 6-2 and the previous results (Table 3-3 and 6-1) is due to the timing procedure. While for the previous results we measured only the generating/fast range-summing time, the results in Table 6-2 also include the overhead time (routine invocation, etc.).

EH3 vs DMAP for Selectivity Estimation: To compare EH3 and DMAP on the task of selectivity estimation, we used the synthetic data generator from [20]. It generates multi-dimensional data distributions consisting in regions, randomly placed in the two-dimensional space, with the number of points in each region Zipf distributed and the distribution within each region Zipf distributed as well. For the experiments we report here, we generated two-dimensional datasets with the domain for each dimension having the size 1024. A dataset consists of 10 regions of points. The distribution of the frequencies within each region has a variable Zipf coefficient, as shown in Figure 6-8. Notice that for small Zipf coefficients EH3 outperforms DMAP by a factor of 14. When the Zipf coefficient becomes larger, the gap between DMAP and EH3 shrinks considerably, but EH3 still outperforms DMAP by a large margin.

Accuracy: We pursue two goals in our accuracy experiments. First, we determine the dependence of the average relative error on the memory budget, i.e., the number of counters in the sketch structure. For this, we run experiments with different sketch configurations having either 4 or 8 rows in the structure and varying the number of counters in a row between 64 and 1024. Second, we want to establish the relation between the accuracy and the number of partitions for AGMS P. For this, we distribute the counters in the sketch into 4 to 64 groups corresponding to an equal number of partitions of the domain. Given the previous results in [51] for AGMS and AGMS DMAP, we expect the results for F-AGMS to be close to AGMS DMAP, with some improvement for

F-AGMS COUNTS which eliminates the effect of outliers to some extent. At the same time, we expect that partitioning does not significantly deteriorate the performance of AGMS P unless it is applied to the extreme where only one counter corresponds to each partition.

Figure 6-9 depicts the accuracy results for a specific parameter configuration. The same trend was observed for the other settings, with the normal behavior for the confined action of each parameter. As expected, the error decreases as the memory budget increases for all the methods (left plot). The behavior of DMAP methods is more sensitive to the available memory, without a significant difference between AGMS and F-AGMS sketches, but still slightly favorable to F-AGMS. What is significant is the effect of maintaining exact counts for F-AGMS DMAP. The error reduces drastically, to the point it is almost identical with the error of fast range-summation AGMS, the most accurate of the studied methods. This is due to limiting the effect of outliers that would otherwise significantly deteriorate the accuracy of the sketch. Notice the reduced levels of the error for fast range-summation methods even when only low memory is available. Our second goal was to detect the effect partitioning has on the accuracy of fast range-summation AGMS. From the right plot in Figure 6-9, we observe that partitioning has almost no influence on the accuracy of AGMS, the errors of the two methods being almost identical even when a significant number of partitions is used. Clearly, we expect the accuracy to drop after a certain level of partitioning, when the number of counters corresponding to a partition is small. The error of the other methods is plotted in (b) only for completeness. The fluctuations are due only to the randomness present in the methods since the experiments were repeated with the same parameters for each different configuration of AGMS P.

Update Time: Our objective is to measure the time to update the sketch structure for the presented sketching methods. For a sketch consisting of only one row of counters, we know that the time is linear for AGMS sketches since all the counters have to be updated. This is reflected in Figure 6-10 that depicts the update time per interval for

two sketch structures, one with 512 counters (left), and one with 4096 counters (right)[2] . Notice that Figure 6-10 actually plots the update time per interval as a function of the number of partitions, thus the constant curves for all the methods except AGMS P. As expected, the update time for AGMS P decreases as the number of partitions increases since the number of counters in a partition decreases. The decrease is substantial for a sketch with 512 counters, to the point where the update time is almost identical with the time for DMAP over F-AGMS sketches, the fastest method. Notice the significant gap of 2 to 4 orders of magnitude between the methods based on AGMS and those based on F-AGMS sketches, with the update time for F-AGMS being in the order of a few micro-seconds, while the update time for AGMS is in the order of milli-seconds.

6.7 Discussion

As it was already known [51], DMAP is inferior both in accuracy and update time to fast range-summation for AGMS sketches, facts re-proved by our experimental results. While DMAP can be used in conjunction with any type of sketching technique, fast range-summation is immediately applicable only to AGMS sketches. In order to improve the update time of AGMS, we propose AGMS P, a method that reduces the number of counters that need to be updated by partitioning the domain of the key and distributing the counters over the partitions. Even with a simple partitioning that splits the domain into buckets with the same size, as is done for equi-width histograms, the improvement we obtain is remarkable, the update time becoming comparable with that for F-AGMS sketches, while the error remains as good as the error of fast range-summation AGMS. The only improvement gained by using DMAP over F-AGMS sketches is in update time. With a simple implementation modification that keeps exact counts for large dyadic intervals (F-AGMS COUNTS), the error drops significantly and becomes comparable with

[2] We used the same machine as in Section 4.4.

the error of fast range-summation AGMS. The roots of this modified method lie in the statistical analysis presented in Section 5.2.

Overall, to obtain methods for sketching intervals that have both small error and efficient update time, the basic techniques (DMAP and fast range-summation) have to be modified. F-AGMS COUNTS is a modification of DMAP over F-AGMS sketches with extremely efficient update time and with error approaching the standard given by AGMS for large enough memory. AGMS P is a modification of fast range-summation AGMS that has excellent error and with update time close to that of F-AGMS sketches when the number of partitions is large enough. In conclusion, we recommend the use of F-AGMS COUNTS when the update time is the bottleneck and AGMS P when the available space is a problem.

6.8 Conclusions

Our primary focus in this chapter was the identification of the fast range-summable schemes that can sketch intervals in sub-linear time. We explain how the fast range-summable versions of two of the 3-wise independent schemes, BCH3 and EH3, can be implemented efficiently and we provide an empirical comparison with the only known 4-wise independent fast range-summable scheme (RM7) that reveals that only BCH3 and EH3 are practical. The EH3-based solutions significantly outperform the state-of-the-art DMAP algorithms for applications such as the size of spatial joins and the dynamic construction of histograms. BCH3 is the perfect solution for sketching highly-skewed interval data since we provide a constant-time algorithm for range-summing BCH3 random variables.

Fast range-summation remains the most accurate method to sketch interval data. Unfortunately, it is applicable only to AGMS sketches and, thus, it is not practical due to the high update time. The solution we propose in this chapter is based on the partitioning of the domain and of the counters in the sketch structure in order to reduce the number of counters that need to be updated. The improvement in update time is substantial, getting close to DMAP over Fast-AGMS sketches, the fastest method studied.

Moreover, by applying a simple modification inspired from the statistical analysis and the empirical study of the sketching techniques, the accuracy of DMAP over F-AGMS can be reduced significantly, to the point where it is almost equal with the accuracy of fast range-summation over AGMS for large enough space. Considering the overall results for sketching interval data, we recommend the use of the fast range-summation method with domain partitioning whenever the accuracy is critical and the use of DMAP COUNTS method over F-AGMS sketches in situations where the time to maintain the sketch is critical.

MinimalDyadicCover($[\alpha, \beta]$)

1 $j \leftarrow 0$
2 $D([\alpha, \beta]) \leftarrow \emptyset$
3 **while** $\alpha \leq \beta$
4 **do if** $\alpha_j = 1$
5 **then**
6 $D([\alpha, \beta]) \leftarrow D([\alpha, \beta]) \cup [\alpha, \alpha + 2^j)$
7 $\alpha \leftarrow \alpha + 2^j$
8 **if** $\beta_j = 0$
9 **then**
10 $D([\alpha, \beta]) \leftarrow D([\alpha, \beta]) \cup [\beta - 2^j + 1, \beta + 1)$
11 $\beta \leftarrow \beta - 2^j$
12 $j \leftarrow j + 1$
13 **return** $D([\alpha, \beta])$

BCH3Interval$([0,\gamma], S = [S_0, s_0])$

1 $\gamma\prime \leftarrow \gamma$

2 $sum \leftarrow 0$

3 **if** $\gamma\prime_0 = 1$

4 **then** $\gamma\prime \leftarrow \gamma\prime - 1$

5 $sum \leftarrow (-1)^{\mathrm{f}(S,\gamma\prime)}$

6 **for** $k \leftarrow 1$ **to** $n-1$

7 **do**

8 **if** $\gamma\prime = 0$

9 **then return** sum

10 **if** $S_{0,k\text{-}1} = 1$

11 **then return** sum

12 **else** $\triangleright$ $S_{0,k\text{-}1} = 0$

13 **if** $\gamma\prime_k = 1$

14 **then** $\gamma\prime \leftarrow \gamma\prime - 2^k$

15 $sum \leftarrow sum + 2^k \cdot (-1)^{\mathrm{f}(S,\gamma\prime)}$

16 **return** sum

BCH3$([\alpha,\beta], S = [S_0, s_0])$

1 $sum_1 \leftarrow$ BCH3Interval$([0, \beta+1), \mathrm{S})$

2 $sum_2 \leftarrow$ BCH3Interval$([0, \alpha), \mathrm{S})$

3 **return** $sum_1 - sum_2$

EH3Interval$([\alpha,\beta], S = [S_0, s_0])$

1 Let $D = \{\delta_1, \ldots, \delta_m\}$ be the minimal dyadic cover of $[\alpha,\beta]$

2 $sum \leftarrow 0$

3 **for** $\delta \in D$

4 **do** $sum \leftarrow sum + (-1)^{\#\mathrm{ZERO}} \cdot 2^j \cdot (-1)^{f(S,q4^j)}$

5 **return** sum

Table 6-1. Sketching time per interval.

Scheme	Time (ns)
BCH3	68.9
EH3	$1,798$
RM7	$26.4 \cdot 10^6$

Table 6-2. Sketching time per interval (ns).

Scheme	Interval			Interval + End-Points		
	LANDC	LANDO	SOIL	LANDC	LANDO	SOIL
BCH3	50	75	79	106	126	115
EH3	637	651	604	681	701	651
DMAP	409	298	309	1500	1401	1391

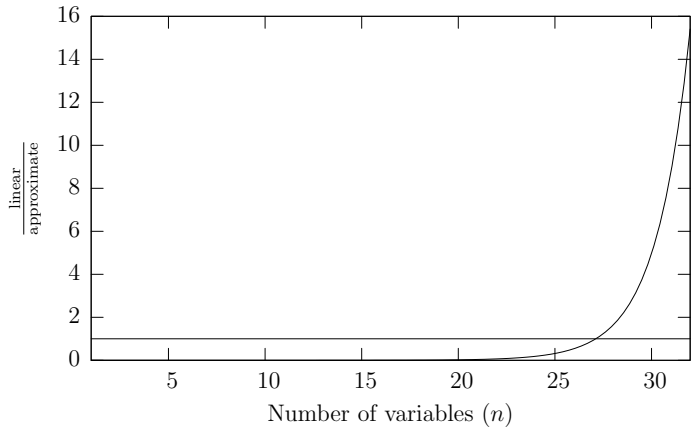


Figure 6-1. The number of polynomial evaluations.

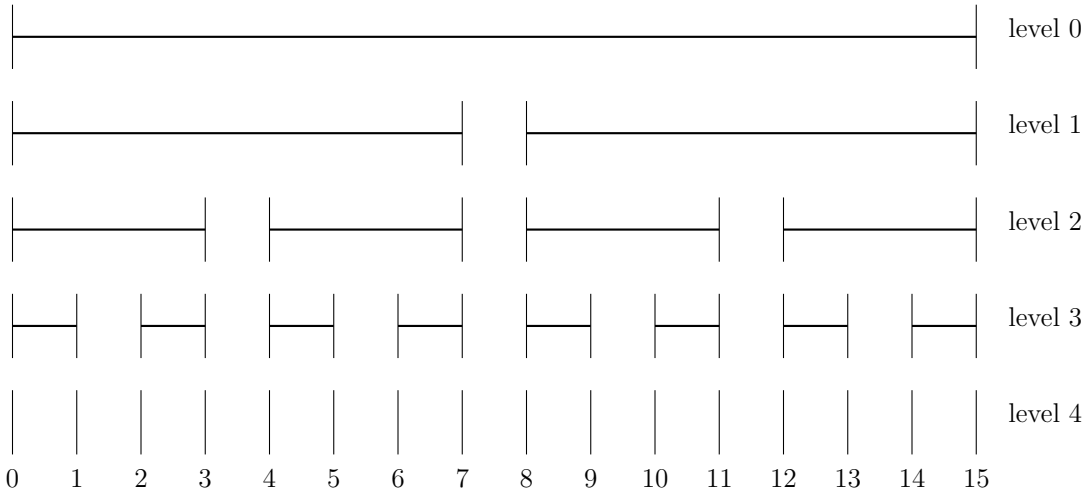
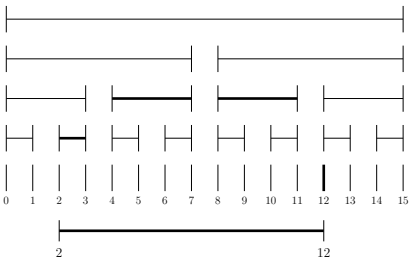
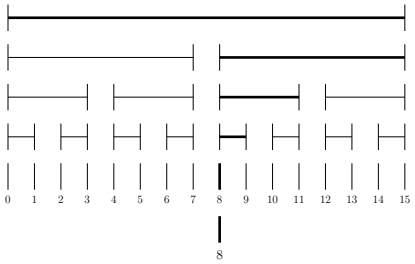


Figure 6-2. The set of dyadic intervals over the domain $I = \{0, 1, \ldots, 15\}$.



(a) Interval mapping



(b) Point mapping

Figure 6-3. Dyadic mappings.

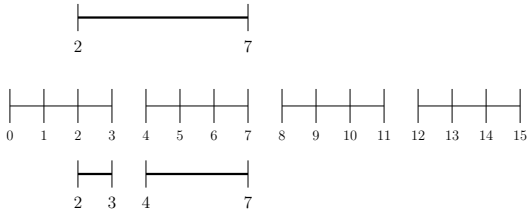


Figure 6-4. Fast range-summation with domain partitioning.

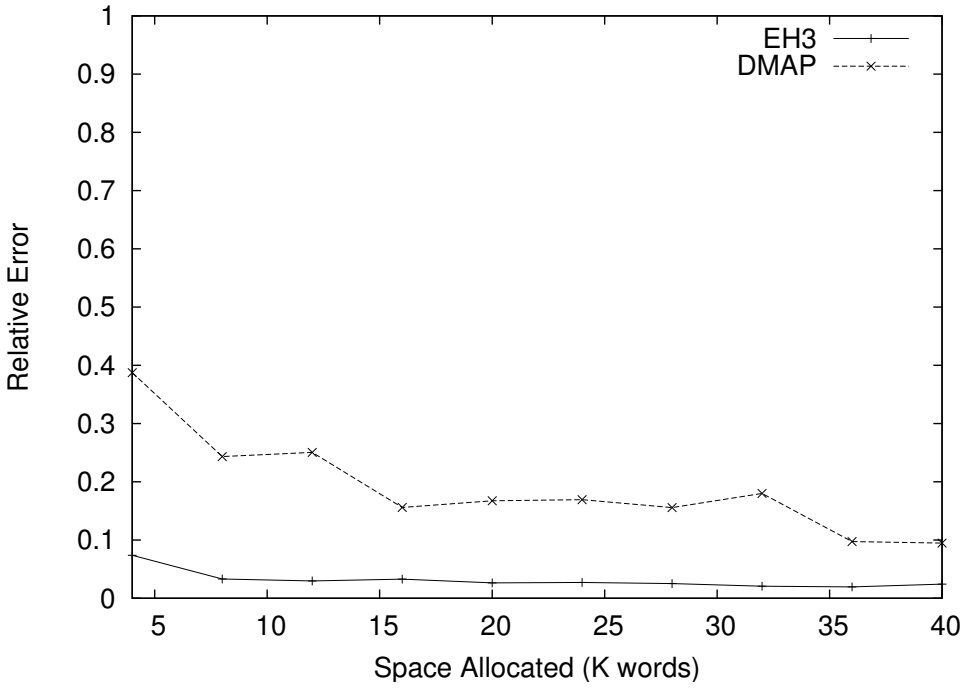


Figure 6-5. LANDO ⋈ LANDC.

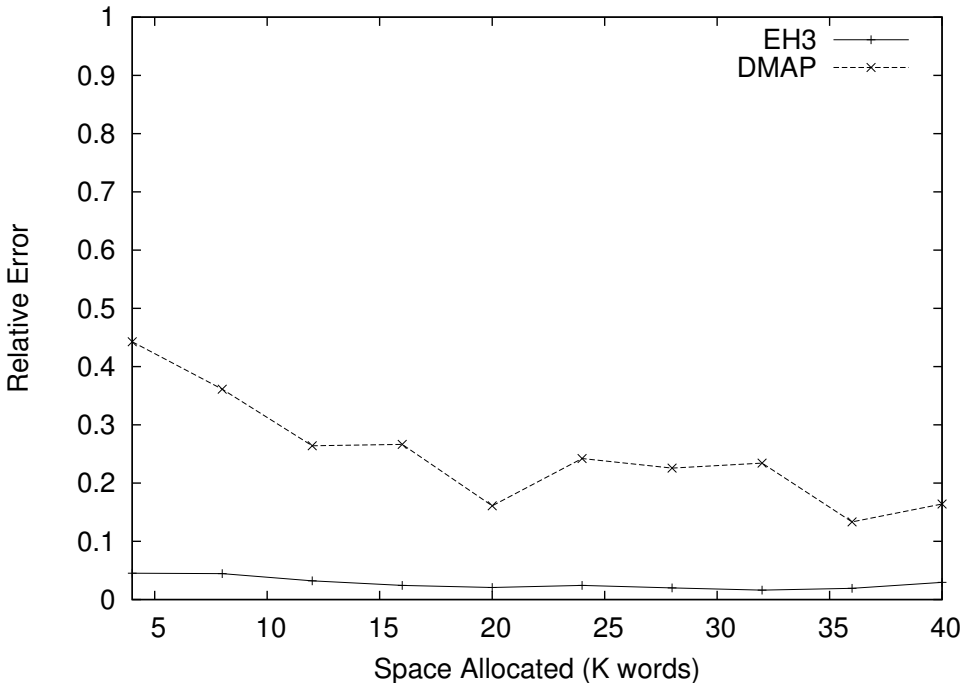


Figure 6-6. LANDO ⋈ SOIL.

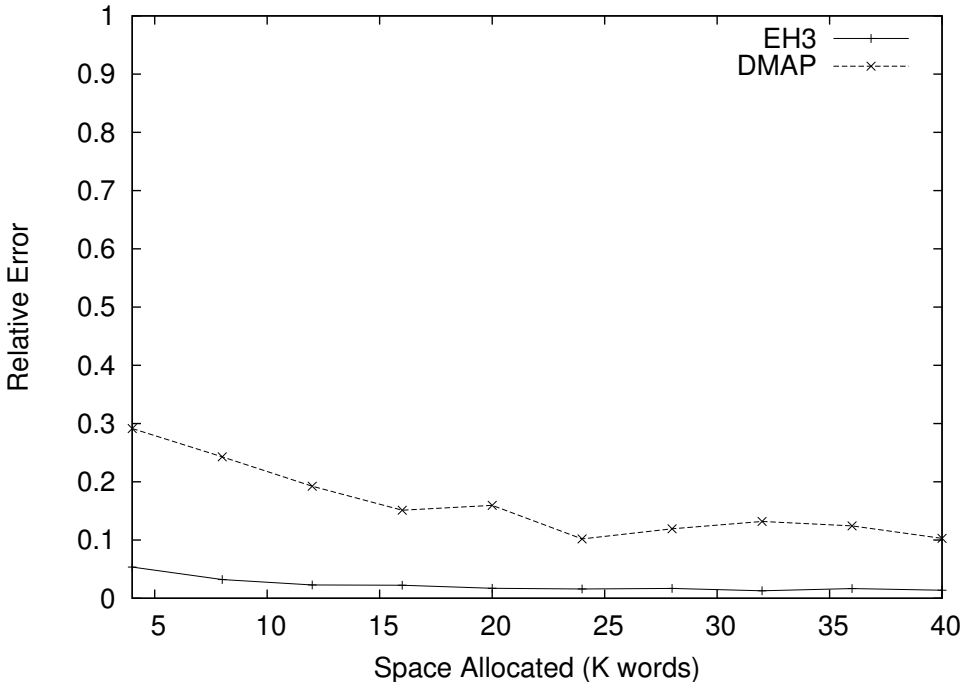


Figure 6-7. LANDC ⋈ SOIL.

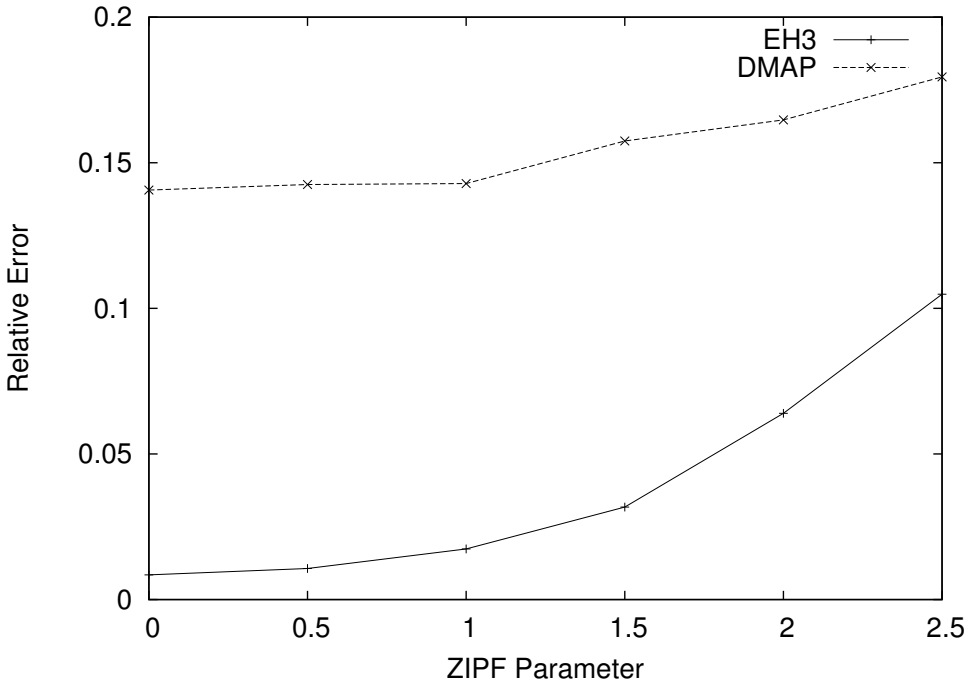
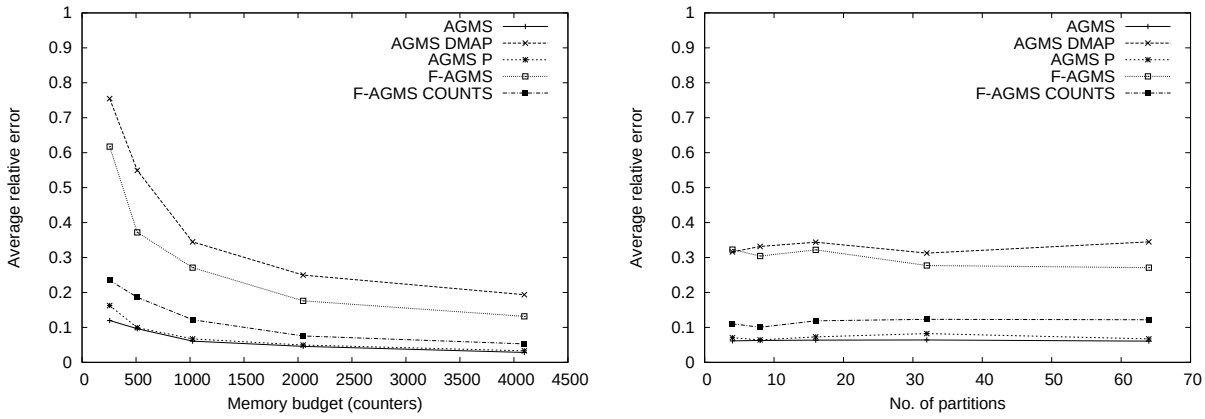
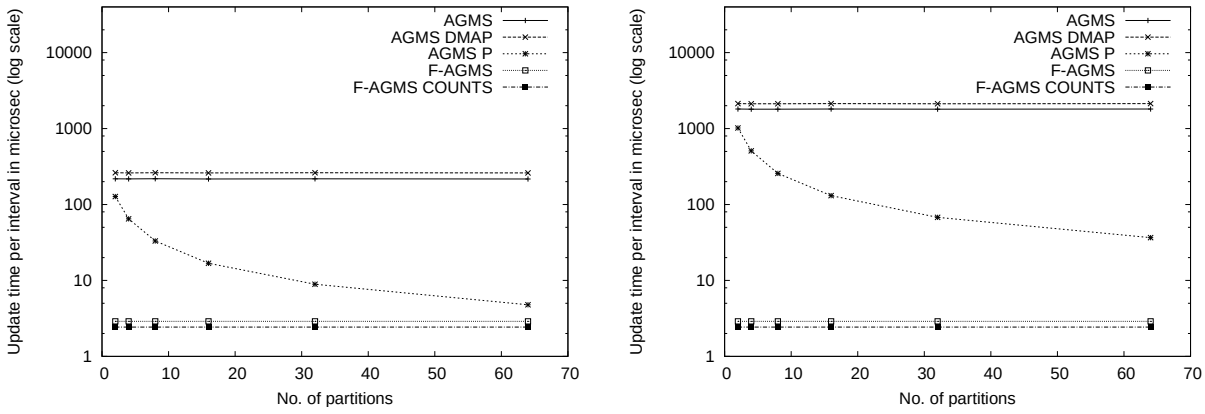


Figure 6-8. Selectivity estimation.



(a) LANDO ⋈ LANDC (b) AGMS P

Figure 6-9. Accuracy of sketches for interval data.



(a) SOIL with 512 counters (b) SOIL with 4096 counters

Figure 6-10. Update time per interval as a function of the number of partitions for the SOIL data set.

CHAPTER 7
CONCLUSIONS

The focus of this research work was on sketch synopsis for aggregate queries over data streams. We believe that our findings are essential for understanding the theoretical foundations that lie at the basis of the sketching methods. At the same time, we believe that the experimental results presented throughout this work support our final goal to make sketch synopsis practical for the application in a data stream environment. The theoretical and practical findings of our work can be summarized as follows:

- We conducted both a theoretical as well as an empirical study of the various schemes used for the generation of the random variables that appear in AGMS sketches estimations. We provided theoretical and empirical evidence that EH3 can replace the 4-wise independent schemes for the estimation of the size of join using AGMS sketches. Our main recommendation is to use the EH3 random variables for AGMS sketches estimations of the size of join since they can be generated more efficiently and use smaller seeds than any of the 4-wise independent schemes. At the same time, the error of the estimate is as good as or, in the case when the distribution has low skew, better than the error provided by a 4-wise independent scheme.

- We provided the moment analysis of the sketch over samples estimators for two types of sampling: Bernoulli and sampling with replacement. Sketching Bernoulli samples is essentially a load shedding technique for sketching data streams which results, as our theory and experiments suggest, in significant update time reduction – by as much as a factor of 100 – with minimal accuracy degradation. Sketching samples with replacement from an unknown distribution allows efficient characterization of the unknown distribution which has many applications to online data-mining.

- We studied the four basic sketching techniques proposed in the literature, AGMS, Fast-AGMS, Fast-Count, and Count-Min, from both a statistical and empirical point of view. Our study complements and refines the theoretical results known about these sketches. The analysis reveals that Fast-AGMS and Count-Min sketches have much better performance than the theoretical prediction for skewed data, by a factor as much as 10^6 to 10^8 for large skew. Overall, the analysis indicates strongly that Fast-AGMS sketches should be the preferred sketching technique since it has consistently good performance throughout the spectrum of problems. The success of the statistical analysis we performed indicates that, especially for estimators that use minimum or median, such analysis gives insights that are easily missed by classical theoretical analysis. Given the good performance, the small update time, and the fact that they have tight error guarantees, Fast-AGMS sketches are

appealing as a practical basic approximation technique that is well suited for data stream processing.

- We identified the fast range-summable schemes that can sketch intervals in sub-linear time. We explained how the fast range-summable versions of two of the 3-wise independent schemes, BCH3 and EH3, can be implemented efficiently and we provided an empirical comparison with the only known 4-wise independent fast range-summable scheme (RM7) that reveals that only BCH3 and EH3 are practical. The EH3-based solutions significantly outperform the state-of-the-art DMAP algorithms for applications such as the size of spatial joins and the dynamic construction of histograms. BCH3 is the perfect solution for sketching highly-skewed interval data since we provided a constant-time algorithm for range-summing BCH3 random variables. Fast range-summation remains the most accurate method to sketch interval data. Unfortunately, it is applicable only to AGMS sketches and, thus, it is not practical due to the high update time. The solution we proposed is based on the partitioning of the domain and of the counters in the sketch structure in order to reduce the number of counters that need to be updated. The improvement in update time is substantial, getting close to DMAP over Fast-AGMS sketches, the fastest method studied. Moreover, by applying a simple modification inspired from the statistical analysis and the empirical study of the sketching techniques, the accuracy of DMAP over F-AGMS can be reduced significantly, to the point where it is almost equal with the accuracy of fast range-summation over AGMS for large enough space. Considering the overall results for sketching interval data, we recommend the use of the fast range-summation method with domain partitioning whenever the accuracy is critical and the use of DMAP COUNTS method over F-AGMS sketches in situations where the time to maintain the sketch is critical.

REFERENCES

[1] P. Aduri and S. Tirthapura, "Range efficient computation of F_0 over massive data streams," *Proc. IEEE ICDE*, 2005.

[2] N. Alon, L. Babai, and A. Itai, "A fast and simple randomized parallel algorithm for the maximal independent set problem," *Journal of Algorithms*, vol. 7, no. 4, pp. 567–583, 1986.

[3] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," *Proc. ACM STOC*, 1996.

[4] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy, "Tracking join and self-join sizes in limited storage," *Journal of Computer and System Sciences*, vol. 64, no. 3, pp. 719–747, 2002.

[5] N. An, Z. Y. Yang, and A. Sivasubramaniam, "Selectivity estimation for spatial joins," *Proc. IEEE ICDE*, 2001.

[6] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," *Proc. ACM SIGMOD*, 2002.

[7] K. P. Balanda and H. L. MacGillivray, "Kurtosis: A critical review," *J. American Statistician*, vol. 42, no. 2, pp. 111–119, 1988.

[8] Z. Bar-Yosseff, R. Kumar, and D. Sivakumar, "Reductions in streaming algorithms, with an application to counting triangles in graphs," *Proc. ACM SODA*, 2002.

[9] A. R. Calderbank, A. Gilbert, K. Levchenko, S. Muthukrishnan, and M. Strauss, "Improved range-summable random variable construction algorithms," *Proc. ACM SODA*, 2005.

[10] L. Carter and M. N. Wegman, "Universal classes of hash functions," *Journal of Computer and System Sciences*, vol. 18, no. 2, pp. 143–154, 1979.

[11] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," *Proc. Int'l Conf. ICALP*, 2002.

[12] S. Coles, "An Introduction to Statistical Modeling of Extreme Values," *Springer-Verlag*, 2001.

[13] G. Cormode and M. Garofalakis, "Sketching probabilistic data streams," *Proc. ACM SIGMOD*, 2007.

[14] G. Cormode and M. Garofalakis, "Sketching streams through the net: distributed approximate query tracking," *Proc. 31st Int'l Conf. Very Large Data Bases (VLDB)*, 2005.

[15] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[16] G. Cormode and S. Muthukrishnan, "Summarizing and mining skewed data streams," *Proc. SIAM Data Mining*, 2005.

[17] CPS, http://www.census.gov/cps, accessed, Nov. 2006.

[18] A. Das, J. Gehrke, and M. Riedewald, "Approximation techniques for spatial data," *Proc. ACM SIGMOD*, 2004.

[19] W. E. Deskins, "Abstract Algebra," *Dover Publications*, 1996.

[20] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi, "Processing complex aggregate queries over data streams," *Proc. ACM SIGMOD*, 2002.

[21] P. Domingos and G. Hulten, "Mining high-speed data streams," *Proc. ACM SIGKDD*, 2000.

[22] W. Dumouchel, C. Faloutsos, P.J. Haas, J.M. Hellerstein, Y. Ioannidis, H.V. Jagadish, T. Johnson, R. Ng, V. Poosala, K.A. Ross, and K.C. Sevcik, "The New Jersey data reduction report," *IEEE Data Eng. Bulletin*, vol. 20, no. 1, pp. 3–45, 1997.

[23] A. Ehrenfeucht and M. Karpinski, "The computational complexity of (*xor-and*) counting problems," ICSI Technical Report TR-90-033, 1990.

[24] C. Estan and J. F. Naughton, "End-biased samples for join cardinality estimation," *Proc. IEEE ICDE*, 2006.

[25] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan, "An approximate L^1-difference algorithm for massive data streams," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 131–151, 2002.

[26] P. Flajolet and G.N. Martin, "Probabilistic counting algorithms for data base applications," *J. Comp. Syst. Sci.*, vol. 31, no. 1, pp. 182–209, 1985.

[27] S. Ganguly, M. Garofalakis, and R. Rastogi, "Processing set expressions over continuous update streams," *Proc. ACM SIGMOD*, 2003.

[28] S. Ganguly, M. Garofalakis, and R. Rastogi, "Processing data-stream join aggregates using skimmed sketches," *Proc. Int'l Conf. EDBT*, 2004.

[29] S. Ganguly, D. Kesh, and C. Saha, "Practical algorithms for tracking database join sizes," *Proc. Int'l Conf. FSTTCS*, 2005.

[30] P. Gibbons, Y. Matias, and V. Poosala, "Fast incremental maintenance of approximate histograms," *Proc. 23rd Int'l Conf. Very Large Data Bases (VLDB)*, 2001.

[31] P. B. Gibbons and Y. Matias, "Synopsis data structures for massive data sets," *DIMACS*, 1999.

[32] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss, "One-pass wavelet decompositions of data streams," *IEEE TKDE*, vol. 15, no. 3, pp. 541–554, 2003.

[33] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss, "Domain-driven data synopses for dynamic quantiles," *IEEE TKDE*, vol. 17, no. 7, pp. 927–938, 2005.

[34] O. Goldreich, "A sample of samplers - a computational perspective on sampling," *Electronic Colloquium on Computational Complexity*, vol. 4, no. 20, 1997.

[35] P. J. Haas and J. M. Hellerstein, "Ripple joins for online aggregation," *Proc. ACM SIGMOD*, 1999.

[36] A. S. Hedayat, N. J. A. Sloane, and J. Stufken, "Orthogonal Arrays: Theory and Applications," *Springer-Verlag*, 1999.

[37] P. Indyk, "Stable distributions, pseudorandom generators, embeddings, and data stream computation," *J. ACM*, vol. 53, no. 3, pp. 307–323, 2006.

[38] T. S. Jayram, A. McGregor, S. Muthukrishnan, and E. Vee, "Estimating statistical aggregates on probabilistic data streams," *Proc. ACM SIGMOD*, 2007.

[39] C. Jermaine, A. Dobra, S. Arumugam, S. Joshi, and A. Pol, "The sort-merge-shrink join," *ACM Transactions on Database Systems*, vol. 31, no. 4, pp. 1382–1416, 2006.

[40] M. Karpinski and M. Luby, "Approximating the number of zeroes of a GF[2] polynomial," *Journal of Algorithms*, vol. 14, no. 2, pp. 280–287, 1993.

[41] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," *Proc. IEEE FOCS*, 2003.

[42] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications," *Proc. ACM SIGCOMM*, 2003.

[43] K. Levchenko, http://www.cse.ucsd.edu/~klevchen/II-2005.pdf, accessed, Aug. 2005.

[44] M. Luby and A. Wigderson, "Pairwise independence and derandomization," EECS UC Berkeley Technical Report UCB/CSD-95-880, 1995.

[45] MassDAL. http://www.cs.rutgers.edu/~muthu/massdal.html, accessed, Jul. 2006.

[46] R. Motwani and P. Raghavan, "Randomized Algorithms," *Cambridge University Press*, 1995.

[47] S. Muthukrishnan, "Data streams: algorithms and applications," *Found. Trends Theor. Comput. Sci.*, vol. 1, no. 2, pp. 117–136, 2005.

[48] D. J. Olive, "A simple confidence interval for the median," Manuscript, http://www.math.siu.edu/olive/ppmedci.pdf, accessed, Nov. 2006.

[49] F. Pennecchi and L. Callegaro, "Between the mean and the median: the L_p estimator," *Metrologia*, vol. 43, no. 3, pp. 213–219, 2006.

[50] R. M. Price and D. G. Bonett, "Estimating the variance of the sample median," *J. Statistical Computation and Simulation*, vol. 68, no. 3, pp. 295–305, 2001.

[51] F. Rusu and A. Dobra, "Pseudo-random number generation for sketch-based estimations," *ACM Transactions on Database Systems*, vol. 32, no. 2, 2007.

[52] F. Rusu and A. Dobra, "Statistical analysis of sketch estimators," *Proc. ACM SIGMOD*, 2007.

[53] L. Sachs, "Applied Statistics – A Handbook of Techniques," *Springer-Verlag*, 1984.

[54] J. Shao, "Mathematical Statistics," *Springer-Verlag*, 1999.

[55] C. Sun, D. Agrawal, and A. El Abbadi, "Selectivity estimation for spatial joins with geometric selections," *Proc. Int'l Conf. EDBT*, 2002.

[56] N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker, "Load shedding in a data stream manager," *Proc. 29th Int'l Conf. Very Large Data Bases (VLDB)*, 2003.

[57] N. Thaper, S. Guha, P. Indyk, and N. Koudas, "Dynamic multidimensional histograms," *Proc. ACM SIGMOD*, 2002.

[58] M. Thorup and Y. Zhang, "Tabulation based 4-universal hashing with applications to second moment estimation," *Proc. ACM SODA*, 2004.

[59] M. Wegman and J. Carter, "New hash functions and their use in authentication and set equality," *Journal of Computer and System Sciences*, vol. 3, no. 22, pp. 265–279, 1981.

BIOGRAPHICAL SKETCH

Florin Rusu received his Ph.D. in Computer Science from the University of Florida. He worked under the supervision of Dr. Alin Dobra as a member of the Database Center.

Florin Rusu is originally from the city of Cluj-Napoca, in the historical region of Transylvania, Romania. Florin received his Bachelor of Engineering degree in 2004 from the Technical University of Cluj-Napoca, Faculty of Automation and Computer Science, under the supervision of Prof. Sergiu Nedevschi.

Florin has research interests in the area of approximate query processing for databases and data streaming, data warehouses, and database system design and implementation.