Fast Range-Summable Random Variables for Efficient Aggregate Estimation [*]

Florin Rusu
University of Florida
frusu@cise.ufl.edu

Alin Dobra
University of Florida
adobra@cise.ufl.edu

ABSTRACT

Exact computation for aggregate queries 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 the only viable solution. The performance of sketches crucially depends on the ability to efficiently generate particular pseudo-random numbers. In this paper 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 that are fast range-summable (i.e., they can be summed up in sub-linear time). Our specific contributions are: (a) we provide an empirical comparison of the various pseudo-random number generating schemes, (b) we study both theoretically and empirically the fast range-summation practicality for the 3 and 4-wise independent generating schemes and we provide efficient implementations for the 3-wise independent schemes, (c) we show convincing theoretical and empirical evidence that the extended Hamming scheme performs as well as any 4-wise independent scheme for estimating the size of join using AMS-sketches, even though it is only 3-wise independent. We use this generating scheme to produce estimators that significantly outperform the state-of-the-art solutions for two problems – *size of spatial joins* and *selectivity estimation*.

1. INTRODUCTION

Exact computation for aggregate queries 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 that can be maintained over data-streams or that can be used for estimations in distributed environments are the only viable solution. Due to their linearity, AMS-sketches [4] have all these properties and they have been successfully used for the estimation of aggregates like the size of join over data-streams [3, 8] and for computations in distributed environments like sensor networks [17]. AMS-sketches depend on the ability to efficiently generate large families of random variables with particular properties: limited degree of independence and, for applications in which the input is specified as a set of intervals, *fast range-summation* (i.e., the ability to sketch an interval in time sub-linear in the size of the interval). As we show in this paper, the last property is especially useful when AMS-sketches are applied for solving problems like *size of spatial joins* [7], L^1*-difference* of two vectors [10], and *selectivity estimation* for the dynamic construction of histograms [22] over data-streams or in a distributed environment.

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

- We provide an empirical comparison of the various generating schemes with the goal of identifying the efficient ones. To this, we explain how the schemes can be implemented on modern processors and we use such implementations to empirically evaluate the generating schemes. The main conclusion of this study is that the schemes based on BCH codes need the smallest seeds and have the most efficient implementations.

- 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 practical, even though the scheme based on Reed-Muller codes can be theoretically range-summed in sub-linear time [6]. Two of the 3-wise independent schemes, the 3-wise BCH scheme and the *extended Hamming scheme* [10], are practical. For both, we explain how they can be efficiently implemented and we conduct an empirical study to determine their performance.

- We show, both theoretically and empirically, that the *extended Hamming scheme* (EH3) [10] is as good, and sometimes much better, than any 4-wise independent generating scheme for estimations using AMS-sketches. The fact that EH3 gets within a constant factor of

[*]Material in this paper is based upon work supported by the National Science Foundation under Grant No. 0448264.

the error of the 4-wise independent schemes for the problem of computing the L^1-difference of two streaming vectors was theoretically proved by Feigenbaum et al. [10]. Here we show a significantly stronger result, namely that EH3 can always replace the 4-wise independent schemes for estimations using AMS-sketches without sacrificing accuracy. Moreover, EH3 has two clear advantages: it can be implemented more efficiently and, more important, it is practically fast range-summable. We show that the estimation of the size of spatial joins using EH3 has significantly smaller errors than the solution proposed in [7].

In the rest of the paper, we first give some introductory notions in Section 2, then we discuss the known generating schemes for random variables with limited independence in Section 3. In Section 4 we investigate which generating schemes are fast range-summable from a practical point of view. In Section 5 we provide theoretical proof that the extended Hamming generating scheme (EH3) works as well as the 4-wise independent schemes with the added benefit that it is fast range-summable. We provide empirical evidence of this fact in Section 6 together with a thorough comparison of the estimations using EH3 and the 4-wise schemes on two applications with previously proposed solutions. We conclude in Section 7.

2. PRELIMINARIES

In this section we give some preliminaries that are useful for understanding the rest of the paper.

2.1 Random Sketches

Sketches [3] are randomized schemes for approximating aggregates such as the size of join. They are particularly suited when the computation is memory restricted – in the case of data-streams – or communication restricted – in the case of distributed systems such as sensor networks.

To introduce a sketch-based solution, consider the problem of computing the size of the natural join of two relations, R and S, each with a single attribute A, $|R \bowtie_A S|$. If we let I to be the domain of the attribute A and r_i and s_i to be the frequency of the value $i \in I$ in R and S, respectively, the size of join problem is to estimate the quantity $|R \bowtie_A S| = \sum_{i \in I} r_i s_i$. The straightforward solution to this problem is to maintain the frequency vectors $\vec{r}$ and $\vec{s}$ 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 $|I|$. The solution based on sketches is defined as follows:

1. Start with a family of 4-wise independent ± 1 random variables $\xi_i, i \in I$ (i.e., any four random variables in the family are independent).
2. Define the sketches $X_R = \sum_{i \in I} r_i \xi_i = \sum_{t \in R} \xi_{t.A}$ and, similarly, $X_S = \sum_{i \in I} s_i \xi_i = \sum_{t \in S} \xi_{t.A}$.
3. Define the random variable $X = X_R X_S$. X has the properties that it is an unbiased estimator for the size of join $|R \bowtie_A S|$ 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 medians of averages of multiple independent copies of the random variable X; the number of medians is proportional to $\log \frac{1}{\delta}$, while the number of averaged random variables is proportional to $\frac{\mathrm{Var}(X)}{\epsilon^2 E[X]^2}$.

Sketches are perfectly suited for both data-streaming and distributed communication since they can be updated on pieces. For example, if the tuples in the relation R are streamed one by one, X_R can be computed by simply adding the value ξ_i, where i is the value of the current item. For distributed computation, each party can compute the sketch of the data it owns; by exchanging only the value of the sketch with the other parties and simply adding up the sketches, the sketch of the entire data can be obtained.

The type of sketch described here uses ± 1 random variables. ± 1 values can be obtained by simply generating random bits with required properties and then interpreting the $\{0, 1\}$ values as ± 1. The minimum requirement for this family of random variables is to be 2-wise independent, which ensures that X is an unbiased estimator for the size of join. The stronger 4-wise independence property is usually required in order to make the variance as small as possible, which reduces the number of copies of X that need to be averaged in order to achieve a given precision.

The above sketches using ± 1 random variables with limited independence can be extended so that results of large classes of queries can be approximated. For example, Dobra et al. [8] show how to extend the sketches to compute complex aggregates over general equi-joins, Das et al. [7] show how to approximate the size of spatial joins, Ganguly et al. [11] show how to compute aggregates over expressions involving set operators. What all these schemes have in common is the dependency on ± 1 random variables and the fact that a certain amount of independence is required in order to keep the variance small. For applications such as spatial joins an extra property is required for the random variables: fast range-summation, i.e., the ability to compute $\sum_{i \in [\alpha, \beta]} \xi_i$ for a range $[\alpha, \beta]$ in time sub-linear in the size of the range.

2.2 Abstract Algebra

As mentioned in the previous section, ± 1 random variables with limited independence can be obtained by generating $\{0, 1\}$ random variables and mapping them to ± 1. Since $\{0, 1\}$ are the only elements of the *Galois Field* with order 2, denoted by $GF(2)$, abstract algebra is the ideal framework in which to talk about the generation of 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 field $GF(2)$: *vector spaces* and *extension fields.* Both these extensions are useful for generating limited independence ± 1 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 $\mathbf{v}$ and $\mathbf{u}$, defined as: $\mathbf{v} \cdot \mathbf{u} = \oplus_{j=0}^{k-1} v_j \odot u_j$. For $GF(2)^k$ vector spaces this corresponds to AND-ing the arguments and then XOR-ing all the resulting bits.

$GF(p)$ *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.

2.3 Dyadic Intervals

As mentioned in Section 2.1, there exist applications requiring the ±1 limited independence random variables to be range-summable. One strategy to design such random variables is first to achieve range-summation for special intervals and then to extend it to general intervals. *Dyadic intervals* [13] are particularly useful for this purpose. Given a domain I of size $|I| = 2^n$, its dyadic intervals are all the intervals of the form $[q2^j, (q+1)2^j)$ with $0 \le j \le n$ and $0 \le q \le 2^{n-j}$. A graphical representation of the dyadic intervals defined over the domain $I = \{0, \ldots, 15\}$ is depicted in Figure 1.

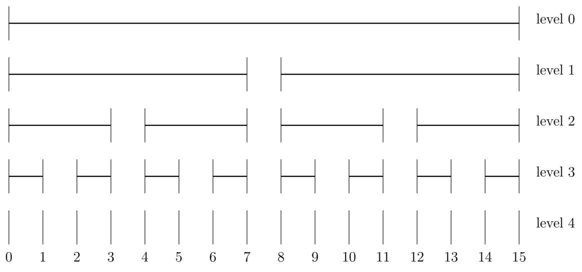

Figure 1: The set of dyadic intervals over the domain $I = \{0, 1, \ldots, 15\}$

Any interval $[\alpha, \beta]$ on I has a unique minimal decomposition – minimal dyadic cover – into at most $2n-1$ dyadic intervals. If a fast range-summable algorithm exists for dyadic intervals, then it can be extended to arbitrary intervals by simply summing up the contribution of each dyadic interval; the running time increases by at most a logarithmic factor. The minimal dyadic cover of $[\alpha, \beta]$ can be determined straightforwardly from the binary representation on n bits of the end-points of the interval. This allows an efficient implementation of the algorithms that use dyadic interval decomposition on modern processors.

3. GENERATING SCHEMES

Based on the published literature [4], AMS-sketches described in Section 2 need ±1 4-wise independent random variables that can be generated in small space in order to produce correct estimations. We address the problem of whether the 4-wise independence is actually required later in the paper, 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 that are either 2-wise[1] or 4-wise independent and we discuss how they can be implemented on modern processors. In the subsequent sections we refer back to these generating schemes.

In order to be space efficient, all the schemes have the following form:

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

where S is a random seed from some space, to be specified later, and function f can be efficiently computed from i and S (polynomial in the representation). Since a ξ_i is required for every different value in a large domain, e.g., $I = \{0, \ldots, 2^{32}-1\}$, generating methods that depend on small seeds are crucial for the success of sketching methods. Fortunately, there are several methods to generate ξ families from small seeds. An important characteristic of a generating scheme is the degree of independence. Since all the schemes are required to produce the values ±1 with the same probability, the k-wise independence requirement of a generating scheme can be expressed by the following definition [6]:

Definition 1 (uniform k-wise independent family). *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, $v_1, v_2, \ldots, v_k$, we have:*

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

In order to show that a scheme defined by a particular function $f(S,i)$ is uniform k-wise independent, it is enough to show this property holds for the bits generated by $f(S,i)$, which simplifies the exposition. This is the case since $(-1)^x$ maps 0 into 1 and 1 into -1.

Throughout this section we make extensive use of the notation in Section 2.2, in particular the dot product over vector spaces. Before we introduce the known generating schemes, we identify two common points of the majority of the schemes. First, 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 performed eventually) 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. The second point we make is that all the schemes need uniform random seeds from spaces of the form $\{0, \ldots, 2^n-1\}$. These seeds can be generated by simply choosing n uniformly independent bits and combining them to form the binary representation of the seed.

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

[1]Any $2k$-wise independent scheme can be transformed into a $(2k+1)$-wise scheme by simply XOR-ing the result with an extra random seed bit [14].

equivalent with the vector obtained by concatenating the vectors a and b. The size of a, b, and $[a, b]$ will be clear from the context.

3.1 BCH Scheme

The BCH scheme was first introduced in [2] and it is based on BCH codes. This scheme can generate k-wise independent random variables by using uniformly random seeds S that are $kn + 1$ bits in size and has the function $f(S, i)$ defined as:

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

where 1 is a bit and i^{2k-1} is computed in the extension field $GF(2^n)$. This scheme comes close to the theoretical bound based on *Rao's inequalities* [14] on how small the seed space can be – it is the scheme with the smallest 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). Since the 3-wise independent version of this scheme, called BCH3 throughout the paper, is important in latter developments, we provide here its particular form:

$$f(S, i) = S \cdot [1, i] \tag{4}$$

The 4-wise independence required by the AMS-sketches can be obtained with $f(S, i) = S \cdot [1, i, i^3]$ and requires $(2n + 1)$-bit seeds, compared with $n + 1$ for BCH3.

3.1.1 *Extended Hamming 3-Wise Scheme (EH3)*

The extended Hamming 3-wise scheme is a modification of BCH3 and it was introduced in [10]. It requires seeds S of size $n + 1$ and its generating function is defined as:

$$f(S, i) = S \cdot [1, i] \oplus h(i) \tag{5}$$

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

$$h(i) = i_0 \vee i_1 \oplus \cdots \oplus i_{n-2} \vee i_{n-1} \tag{6}$$

Function h does not change the amount of independence, thus, from the traditional AMS-sketches theory, it is not as good as a 4-wise independent scheme, but, as proved theoretically by Feigenbaum et al. [10], it produces comparable errors. We discuss the importance of the function h in Section 4.

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.

3.2 Reed-Muller Scheme

The BCH schemes require computations over extension fields for degrees of independence greater than 3. Since the AMS-sketches need 4-wise independence, alternative schemes that require only simple computations might be desirable. The Reed-Muller scheme [14] 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 independent version of the scheme that requires $1 + n + \frac{n(n-1)}{2}$ seed bits:

$$f(S, i) = S \cdot [1, i, i^{(2)}] \tag{7}$$

where

$$i^{(2)} = [i_0 \odot i_1, i_0 \odot i_2, \ldots, i_{n-2} \odot i_{n-1}] \tag{8}$$

3.3 Polynomials over Primes Scheme

The generating schemes in the previous sections are derived from error-correcting codes. In this section, we present a different method of generating k-wise independent random variables that uses polynomials over a prime number field [15].

THEOREM 1 ([15]). *Let $p \geq N$ be a prime. Choose $a_0, a_1, \ldots, a_{k-1}$ uniformly and independently at random from $\mathbb{Z}_p$, and let $X_j = a_0 + a_1 j + a_2 j^2 + \cdots + a_{k-1} j^{k-1} \mod p$, for $0 \leq j < p$. Then $X_0, X_1, \ldots, X_{p-1}$ are uniform k-wise independent.*

This scheme generates k-wise independent random variables that have p values. In order to restrict the domain to two values, we consider the binary representation of the variables and take into consideration only one bit. The resulting two-valued random variables are not uniformly distributed – they are slightly biased. However, for large primes, e.g., $p = 2^{31} - 1$, the bias between the variables with value 0 and those with value 1 is negligible, i.e., $\frac{1}{2^{31}}$.

The seed of this scheme consists of the k coefficients a_j, $0 \leq j < k$, each represented on $\lceil \log p \rceil$ bits. Note that the size of the seed almost doubled compared to the seed needed for the corresponding BCH generating scheme.

3.4 Performance Evaluation

Scheme	Time (ns)	Seed size
BCH3	10.8	$n + 1$
EH3	7.3	$n + 1$
Massdal2	27.2	$2n$
BCH5[2]	12.7	$2n + 1$
Massdal4	101.2	$4n$
RM7	3,301	$1 + n + \frac{n(n-1)}{2}$

Table 1: Generation time and seed size

We implemented the 3-wise independent BCH (BCH3), the 5-wise independent BCH (BCH5), the extended Hamming (EH3), and the 7-wise independent Reed-Muller (RM7) schemes, and we used Massdal [20] for the implementation of the polynomials over primes scheme. We ran our experiments on a two-processor *Xeon 2.80 GHz* machine, each with a *512 KB* cache. The system had *1 GB* available memory and used 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 in either L1 or L2 cache takes about 250 ns on this machine. We used the special assembly implementation of the dot product for all the

[2]BCH5 was implemented by performing the i^3 operation arithmetically not in an extension field. This does not change the performance of BCH5 for domains not too large.

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 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 1. When compared to 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 Massdal. Out of the 4-wise or higher independence schemes, clearly BCH5 is the fastest, while RM7 is almost 300 times slower.

Table 1 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 seeds with the smallest size, 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.

4. FAST RANGE-SUMMABLE SCHEMES

When at least one of the input relations to the size of join problem in Section 2.1 is given as a union of intervals, the ± 1 family of random variables is required to be fast range-summable. 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. Formally, this property is defined as:

Definition 2 (bitwise range-summation [6]). *A generating scheme for two-valued k-wise independent random variables is called bitwise range-summable if there exists a polynomial-time function g such that*

$$g([\alpha,\beta],S) = \sum_{\alpha \leq i \leq \beta} f(S,i) \tag{9}$$

where α, β, and i are vectors over the domain $\{0,1\}^n$.

Computing the function g over general $[\alpha,\beta]$ intervals is usually not straightforward. The task is easier for dyadic intervals (see Section 2.3) due to their regularity. Fortunately, any scheme that is bitwise range-summable for dyadic intervals can be extended to general intervals $[\alpha,\beta]$ by simply determining its 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.

In this section, we study the bitwise range-summation property of the generating schemes presented in Section 3. For the 2-wise independent random variables, both the BCH3 scheme and its EH3 variant are fast range-summable. Previous work show that there exist other schemes that are fast range-summable for the 2-wise case. For example, the scheme based on the *Toeplitz* family of hash functions is shown to be fast range-summable in [5]. A related algorithm for p-valued 2-wise independent random variables generated using the polynomials over primes scheme is introduced in [1]. For the 4-wise case, the Reed-Muller generating scheme is the only scheme known to be fast range-summable [6, 12]. 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 [9], we obtain a simple method to determine if a generating scheme is fast range-summable. We apply this method to show that the 4-wise BCH and the polynomial schemes are not fast range-summable.

4.1 EH3 Scheme

Although Feigenbaum et al. [10] 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 range-summing BCH3 random variables [21] that is not included here due to lack of space.

The following theorem provides an analytical formula for computing the range-sum function g. Note that only one computation of the generating function f is required in order to determine the value of g over any dyadic interval. The proof can be found in the extended version of the paper [21].

Theorem 2. *Let $[q4^j,(q+1)4^j)$ be a dyadic interval with size at least 4, $j \geq 1$. The range-sum function $g([q4^j,(q+1)4^j),S) = \sum_{i=q4^j}^{(q+1)4^j} f(S,i)$ defined for the extended Hamming 3-wise scheme (EH3) is equal to:*

$$g([q4^j,(q+1)4^j),S) = (-1)^{\#ZERO} \cdot 2^j \cdot f(S,q4^j) \tag{10}$$

where f is the ± 1 generating function and $\#ZERO$ represents the number of two adjacent pair bits that OR to 0.

Based on the results in Theorem 2, Algorithm *H3Interval* computes function $g([\alpha,\beta],S) = \sum_{\alpha \leq i \leq \beta} f(S,i)$ 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 (10). Note 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 *H3Interval* computes function g in $\mathcal{O}(\log(\beta-\alpha))$ steps.

Algorithm 1 H3Interval($[\alpha,\beta]$, $S=[S_0,s_0]$)

```
1. Let $D = \{\delta_1,\ldots,\delta_m\}$ be the minimal dyadic cover of
   $[\alpha,\beta]$, where each $\delta$ has the form $[q4^j,(q+1)4^j)$
2. sum ← 0
3. **for** $\delta_k \in D$, $1 \leq k \leq m$ **do**
4.     sum ← sum + $(-1)^{\#\text{ZERO}} \cdot 2^j \cdot f(S,q4^j)$
5. **end for**
6. **return** sum
```

Example 1. *We show how Algorithm* H3Interval *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*

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

#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 the power greater than* 0.

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

4.2 Four-Wise Independent Schemes

In this section we investigate the bitwise range-summation property of the 4-wise independent generating schemes presented throughout this paper, 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 due to Ehrenfeucht and Karpinski [9] 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 $GF(2)$. 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. For our application, 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, whose proof is omitted due to scarcity of space, use this fact to show that the BCH5 and the polynomials over primes schemes are not fast range-summable.

Theorem 3. *The k-wise independent BCH schemes are not fast range-summable for* $k \geq 5$ *and* $n \geq 4$.

The BCH3 scheme is not covered by this theorem and, indeed, it is fast range-summable. In fact, by exploiting special properties of the BCH3 scheme, a fast range-summable algorithm can be implemented in $O(1)$ average time (with respect to random seeds) if arithmetic operations and dot-products are considered $O(1)$ operations. The reason for this is the fact that, when computing the sum over any interval $[\alpha, \beta]$, only the last bits of α and β that correspond to zero bits in the seed have to be processed before the result of the summation can be computed with a simple arithmetic formula. Since the number of contiguous zero bits at the end of a uniformly random seed can be shown to be approximately 1 on average, the entire computation can be finished in $O(1)$ time. A detailed presentation and proofs of these facts are deferred to the full version of the paper [21].

Theorem 4. *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* $[q2^l, (q+1)2^l)$.

Theorem 4 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 when $k \geq 2$.

4.3 RM7 Scheme

Together with the negative result about the hardness of counting the number of times an XOR of ANDs polynomial with terms containing more that three variables ANDed, Ehrenfeucht and Karpinski [9] provided an algorithm for such counting for formulae that contain only at most two variables ANDed 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 [6, 18]. We focus our discussion on the 2XOR-AND algorithm, but the same conclusions are applicable to the algorithm in [6, 18].

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 ANDed 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 old 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 7-wise Reed-Muller 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) 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 next section where we provide running time comparisons of the fast range-summable algorithms.

Since this fast range-summable algorithm is not practical and fast range-summable algorithms for BCH5 and polynomials over primes schemes do not exist, it does worth to investigate approximation algorithms for the 4-wise case. While such approximations are possible [16, 19], they are not more practical than the exact algorithm for RM7. A detailed theoretical and empirical evaluation of such approximation schemes can be found in the extended version of the paper [21].

4.4 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 3.4. The performance evaluation 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 2, 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 1 for the running times of individual sketches) – this happens since, 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 Reed-Muller 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 the previous section is that there is no practical fast range-summable algorithms for any of the known 4-wise generating schemes. Fortunately, as we show in the next section, the EH3 scheme can successfully replace the 4-wise independent generating schemes in applications using AMS-sketches.

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

Table 2: Sketching time per interval

5. STREAMS AND RANDOM SKETCHES

The AMS-sketches introduced in Section 2.1 are a versatile approximation method that can be applied to numerous estimation problems and that can accommodate multiple types of input. In this section we investigate applications that require fast range-summable random variables and explain how the generating schemes introduced earlier in the paper can be used for successful estimation. As mentioned in Section 2.1, the basic problem the AMS-sketches are solving is estimating the size of join of two relations when they are specified tuple-by-tuple over a data-stream. Here we consider a variation of this problem in which one of the relations is specified as a stream of intervals (this is equivalent to specifying every point inside the interval). First we introduce three practical applications for the size of join problem with interval-input data and then we explain how the fast range-summable generating schemes can be used to solve this problem. We also point out other solutions.

5.1 Applications of Interval-Input Sketches

We introduce three problems – spatial joins [7], L^1 estimation [10], and the construction of multidimensional histograms [22] – for which solutions using AMS-sketches have been proposed. Since all these problems can be reduced to the computation of the size of join where one of the relations is specified as a sequence of intervals, solutions based on fast range-summable random variables can be provided. For each of these applications we show the reduction to the size of join problem and how the fast range-summable AMS-sketches provide improved solutions.

APPLICATION 1 (SPATIAL SIZE OF JOIN [7]). *Given two sets of line segments in an unidimensional space, the spatial size of join problem is to compute the number of segments in the two sets that intersect. The approach in [7], even though not explicitly stated, expresses the solution as the average of two size of join estimators: 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 avoid the range-summation of the intervals, the proposed solution maps the initial domain into the domain of all possible dyadic intervals that can be defined over it. By doing this, the size of each initial 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, each segment end-point maps to a logarithmic (in the size of the initial domain) number of points in the new domain, increasing the number of sketch updates from one to a logarithmic factor. The problem can be generalized to multiple dimensions, see [7], by defining estimators over all possible combinations of full segments and end-points in each dimension. A more detailed discussion of this algorithm and our solution that uses fast range-summable random variables are presented in Section 5.2.*

APPLICATION 2 (L^1-DIFFERENCE [10]). *Given two vectors $\vec{a}$ and $\vec{b}$ with elements a_i and b_i, respectively, $i \in I$, where I is a large domain, compute in small space the L^1-difference of the two vectors defined as $\sum_i |a_i - b_i|$, when the vector elements are streamed in random order as tuples of the form (i, a_i) and (i, b_i). The reduction to the size of join problem is obtained by first fixing a maximum value, M, for the elements in the vectors $\vec{a}$ and $\vec{b}$. Two virtual relations can be introduced, R_a for $\vec{a}$ and R_b for $\vec{b}$, each containing tuples of the form (i, j) for every $i \in I$ and every $j \in \{0, \ldots, a_i\}$, $j \in \{0, \ldots, b_i\}$, respectively. It can be shown that $\sum_i |a_i - b_i|$ is the self-join size of the symmetric difference of R_a and R_b. To see why this is the case, notice that for every tuple in the symmetric difference the contribution to the self-join size is 1. There are exactly $|a_i - b_i|$ such contributions for each $i \in I$ since this is the size of the symmetric difference between the parts of R_a and R_b that have the first attribute value i. The relations R_a and R_b are specified by a number of ranges, one for each a_i and b_i, respectively. The solution to the self-join size problem has to accommodate the easy computation of the symmetric difference and has to have the input specified as intervals.*

APPLICATION 3 (DYNAMIC HISTOGRAMS [22]). *Any histogram construction algorithm, such as the one in [22], is based on evaluating the average frequency for any rectangular region (a potential bucket). The computation of the average frequency can be performed by computing the sum of the frequencies in the region and dividing by the size of the region. If two virtual relations are introduced, one that specifies the data for which the histogram is constructed and* *one that specifies the region by enumerating all the points in the region, it is easy to see that the sum of the frequencies in the region is exactly the size of the join of these two relations. The second relation is specified by a hyper-rectangle, thus solutions to the size of join problem that accommodate interval inputs are required.*

The common point of the above three applications is the fact that they can be reduced to efficiently computing the size of join of two relations that accommodate interval-input data for at least one of the arguments. Solutions using AMS-sketches based on random variables generated with fast range-summable schemes fit perfectly these requirements. For two of the applications, spatial joins and dynamic construction of histograms, an alternative solution based also on the use of AMS-sketches is possible [7]. We provide the description of this solution, that we call DMAP, in the next section.

5.2 Dyadic Mapping (DMAP)

In their work on spatial joins [7], Das et al. presented a solution based on AMS-sketches and dyadic intervals. The solution can be decomposed[2] in two parts: the reduction of the problem to the size of join problem when one of the input relations is specified as a stream of intervals and an approximation scheme for the size of join problem. The solution to the reduction problem was outlined in Application 1. In this section we focus on the solution to the size of join problem for interval-input relations. We call this solution *Dyadic Mapping* (DMAP) for reasons that will be clear shortly. We present here only the main idea of the method; details and proofs can be found in [7].

Let R and S be the two relations with a common attribute A for which the size of join is to be computed. Assume, without loss of generality, that the relation R is specified as a set of intervals, while the relation S as a set of points. In order to efficiently compute the size of join, DMAP generates two virtual relations, one for R and one for S, by mapping both the intervals in R and the points in S to the space of all dyadic intervals defined over the domain of the attribute A (for an introduction to dyadic intervals see Section 2.3). The tuples of the new relations consist of a single dyadic interval attribute over I, the original domain of the attribute A. Each interval $[\alpha, \beta]$ in R is decomposed into its minimal dyadic cover and a tuple for each member of the cover is introduced in the virtual relation R_d. For each point i in S, all the dyadic intervals containing the point (there are $\log |I|$ such intervals) are included in S_d, the virtual relation corresponding to S. The size of join of the two relations R and S is estimated through the size of join of R_d and S_d. It is easy to see that this gives the correct result since for each point $i \in [\alpha, \beta]$ there exists exactly one dyadic interval that contains i and that is included in the minimal dyadic cover of $[\alpha, \beta]$. The size of join of R_d and S_d can be estimated using the AMS-sketches and, thus, the size of join of the original relations R and S is obtained. The sketches of the relations R_d and S_d can be efficiently computed since both for an interval in R and a point in S at most $\log |I|$ dyadic intervals have to be sketched. Notice that, by mapping to the space of dyadic intervals, DMAP avoids the linear update time for the sketching of intervals.

[2]This decomposition was not explicitly made in [7].

To get an idea of how efficient DMAP is, we timed an implementation of this scheme using the same setup as in Section 4.4. The average execution time we obtained is $1,276$ ns, which is slightly faster than the time for EH3 ($1,799$ ns). The time to sketch a point is only 7.9 ns for EH3, but it is 416 ns (32×13 for $|I| = 2^{32}$) for DMAP. As we show in Section 6, the overall running times of EH3 and DMAP algorithms are comparable, with a slight advantage for DMAP.

5.3 Size of Join using Sketches

As we pointed out earlier, an alternative solution to the size of join problem when at least one of the input relations is given as a set of intervals is to use fast range-summable families of random variables to speed up the sketching of the intervals. Using the current understanding of the AMS-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 latter in the section why). Since, as it is shown in Section 4, there is no practical fast range-summable 4-wise independent generating scheme, it looks like DMAP is the only feasible alternative, given the state-of-the-art. What we show in this section is that the extended Hamming 3-wise scheme [10] not only can be used as a reasonable replacement of the 4-wise independent schemes (this is the theoretical result proved in [10] for Application 2), 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 AMS-sketches solutions to the size of join problem. We provide here the theoretical support for this statement, but defer the empirical evidence to Section 6.

We proceed by taking a close look at the estimation of the size of join of two relations using AMS-sketches. Let R and S be two relations with a common attribute A whose domain is I. For a given value $i \in I$, we denote by r_i and s_i the frequency of the value i in R and S, respectively. With this, the size of join $|R \bowtie_A S|$ is defined as $|R \bowtie_A S| = \sum_{i \in I} r_i s_i$. 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 $X_R = \sum_{i \in I} r_i \xi_i$ and $X_S = \sum_{i \in I} s_i \xi_i$. The random variable $X = X_R X_S$ estimates, on expectation, the size of join $|R \bowtie_A S|$. It is easy to show that, due to the 2-wise independence, $E[X] = \sum_{i \in I} r_i s_i$, which is exactly the size of join. Since all the generating schemes published in the literature (see Section 3 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 $\frac{\sqrt{\mathrm{Var}(X)}}{E[X]}$.

Let us now analyze the variance of X. Following the technique in [4], we first compute $E\left[X^2\right]$ since $\mathrm{Var}(X) = E\left[X^2\right] - E\left[X\right]^2$. We have:

$$\begin{aligned} E\left[X^2\right] &= E\left[\left(\sum_{i \in I} r_i \xi_i \sum_{i \in I} s_i \xi_i\right)^2\right] \\ &= \sum_{i \in I} \sum_{j \in I} \sum_{k \in I} \sum_{l \in I} r_i r_j s_k s_l E\left[\xi_i \xi_j \xi_k \xi_l\right] \end{aligned}$$

The expression $E\left[\xi_i \xi_j \xi_k \xi_l\right]$ 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 $E[\xi_i\xi_j\xi_k\xi_l] = E\left[\xi_i^3\xi_l\right] = 0$ (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 $E\left[X^2\right]$ adds up to:

$$E\left[X^2\right] = \sum_{i\in I}\sum_{j\in I} r_i^2 s_j^2 + 2\cdot\left(\sum_{i\in I} r_i s_i\right)^2 - 2\cdot\sum_{i\in I} r_i^2 s_i^2$$

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.

5.3.1 Variance for BCH5

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

$$E[\xi_i\xi_j\xi_k\xi_l] = E[\xi_i]\cdot E[\xi_j]\cdot E[\xi_k]\cdot E[\xi_l] = 0$$

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

$$\mathrm{Var}(X)_{\mathrm{BCH5}} = \sum_{i\in I}\sum_{j\in I} r_i^2 s_j^2 + \left(\sum_{i\in I} r_i s_i\right)^2 - 2\cdot\sum_{i\in I} r_i^2 s_i^2 \quad (11)$$

5.3.2 Variance for BCH3

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

PROPOSITION 1. *Let the function F on n binary variables $x_1, x_2, \ldots, x_n$ be defined as:*

$$F(x_1, x_2, \ldots, x_n) = C \oplus S_1 \odot x_1 \oplus S_2 \odot x_2 \oplus \cdots \oplus S_n \odot x_n$$

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 appearing 2^{n-1} times. Otherwise, function F evaluates to the constant C for all the possible combinations of $x_1, x_2, \ldots, x_n$.

The proof of this proposition is omitted due to scarcity of space, but it can be found in the extended version of the paper [21]. The result is intuitive though, since if any of the parameter bits S_k is nonzero, its corresponding variable x_k causes the function F to take the values 0 and 1 in exactly half of the cases, irrespective to the constant C.

Now we can prove the result for BCH3:

PROPOSITION 2. *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 bit-wise 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, we have:

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

With this, we obtain:

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

Using the result in Proposition 1 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 $E[\xi_i\xi_j\xi_k\xi_l] = 0$. 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 $E[\xi_i\xi_j\xi_k\xi_l] = 1$. □

This means 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:

$$\Delta\mathrm{Var}(\mathrm{BCH3}) = \sum_{i\in I}\sum_{j\in I, j\neq i}\sum_{k\in I, k\neq i,j} r_i r_j s_k s_{i\oplus j\oplus k}$$

where $i \oplus j \oplus k \oplus l = 0$ 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 for BCH5.

5.3.3 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 approaches to *zero*.

PROPOSITION 3. *Assume the ξs are generated using the EH3 scheme. Then, for $i \neq j \neq k \neq l$,*

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

where $\oplus$ is the bit-wise XOR.

PROOF. We know that

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

for random variables generated using the EH3 scheme. Replacing this form into the expression for $E[\xi_i\xi_j\xi_k\xi_l]$ and applying the same manipulations as in the proof of Proposition 2, we get:

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

If we use again Proposition 1 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 $E\left[\xi_i\xi_j\xi_k\xi_l\right]$. $\square$

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 the worst case, an example can be built in which the -1 terms do not appear with nonzero coefficients, but the 1 terms do. In this case the performance of EH3 is equivalent to the performance of BCH3. These are pathological cases, though. An average analysis would me more useful to predict the performance of EH3. To obtain such an average analysis, consider first the proposition:

PROPOSITION 4. *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. $\square$

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 R are correlated with the frequencies in S nor the frequencies in S 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:

$$E\left[s_L\right] = \frac{1}{4^n}\sum_{l\in I} s_l$$

and

$$\begin{aligned} E\left[(-1)^G\right] &= 1\cdot P[G=0] + (-1)\cdot P[G=1] \\ &= \frac{z_n - eq_n - y_n}{z_n - eq_n + y_n} \end{aligned}$$

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

$$\begin{aligned} E\left[\Delta\mathrm{Var}\left(\mathrm{EH3}\right)\right] &= E\left[\sum_{i\in I}\sum_{j\in I}\sum_{k\in I} r_i r_j s_k (-1)^G s_L\right] \\ &= \sum_{i\in I}\sum_{j\in I}\sum_{k\in I} r_i r_j s_k E\left[(-1)^G\right] E\left[s_L\right] \\ &= \frac{1}{4^n}\left(\sum_{i\in I} r_i\right)^2\left(\sum_{i\in I} s_i\right)^2 \frac{z_n - eq_n - y_n}{z_n - eq_n + y_n} \end{aligned}$$

Overall, the variance of the EH3 generating scheme is:

$$\begin{aligned} \mathrm{Var}\left(X\right)_{\mathrm{EH3}} = &\frac{1}{4^n}\left(\sum_{i\in I} r_i\right)^2\left(\sum_{i\in I} s_i\right)^2 \frac{z_n - eq_n - y_n}{z_n - eq_n + y_n} \\ &+ \mathrm{Var}\left(X\right)_{\mathrm{BCH5}} \end{aligned} \tag{12}$$

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 proposition states this surprising result:

PROPOSITION 5. *If $r_i = r$ and $s_i = s$, $i \in I$, with r and s constants, and $|I| = 4^n$, then:*

$$Var\left(X\right)_{EH3} = 0$$

The reason the variance of EH3 is *zero* when the distribution of both R and S 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 experimental results in Section 6.

Given the theoretical results in this section, the experimental results in the following section, and the fact that EH3 is fast range-summable and can be implemented more efficiently than BCH5, we recommend the exclusive use of the EH3 random variables for estimations using AMS-sketches.

6. EMPIRICAL EVALUATION

The purpose of the empirical evaluation is threefold. First, we want to validate the theoretical models in Section 5, especially the average behavior of EH3. Second, we want to compare the EH3 and BCH5 generating schemes for estimations using AMS-sketches. Third, we want to compare EH3 with DMAP [7] (see Section 5.2) on two of the applications introduced in Section 5, namely, spatial joins and selectivity estimation for histograms construction. The reason we do not provide a comparison for the computation of the L^1-difference is the fact that DMAP cannot perform estimations when both relations are specified as sets of intervals. Furthermore, 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 4).

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 5.
- **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 always use EH3 instead of BCH5.
- **EH3 vs DMAP.** Our study shows that both for real and synthetic data applications, EH3 significantly outperforms DMAP, sometimes by as much as a factor of 8, with the same memory usage and smaller running time.

We performed the experiments using the setup in Section 3. We give detailed descriptions of both the datasets and the comparison methodology used for each group of experiments.

6.1 Validation of the EH3 Model

To validate the average error formula in Equation 12, 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 AMS-sketches with only one median, i.e., only averaging is used to decrease the error of the estimate. In Figure 2 we depict the comparison between the average error of the EH3 scheme and the theoretical prediction given by Equation 12 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 Proposition 5, 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.

6.2 EH3 vs BCH5

We performed the same experiments as in the previous section both for EH3 and BCH5, except that the number of medians was fixed to 10. The results of the experiments are depicted in Figure 3. Note that the error for EH3 and BCH5 is virtually the same for Zipf coefficients greater than 1, but for EH3 it is significantly smaller for Zipf coefficients lower than 1. When compared to the results in Figure 2, 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 [7].

6.3 EH3 vs DMAP for Spatial Joins

We used the same experimental setup as in [7] to compare EH3 and DMAP for approximating the size of spatial joins. Three datasets are used, 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 average error for estimating the size of spatial joins for both EH3 and DMAP is depicted in Figure 5, 6, and 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.

6.4 EH3 vs DMAP for Selectivity Estimation

To compare EH3 and DMAP on the task of selectivity estimation, we used the synthetic data generator from [8]. 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 4. 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.

7. CONCLUSIONS

In this paper 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 the AMS-sketches based estimations. Our primary focus 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 [10], can be implemented efficiently and we provide an empirical comparison with the only known 4-wise independent fast range-summable scheme (RM7 [6]) that reveals that only BCH3 and EH3 are practical. Moreover, we provide theoretical and empirical evidence that EH3 can replace the 4-wise independent schemes for the estimation of the size of join using AMS-sketches. The EH3-based solutions significantly outperform the state-of-the-art algorithms for applications such as the size of spatial joins and the dynamic construction of histograms.

The main recommendation of this paper is to always use the EH3 random variables for AMS-sketches estimations of the size of join since they can be generated more efficiently and use a smaller seeds than any of the 4-wise independent schemes. At the same time, the error of the estimate is as good or, in the case when the distribution has low skew, better than the error provided by a 4-wise independent scheme.

8. REFERENCES

[1] P. Aduri and S. Tirthapura. Range efficient

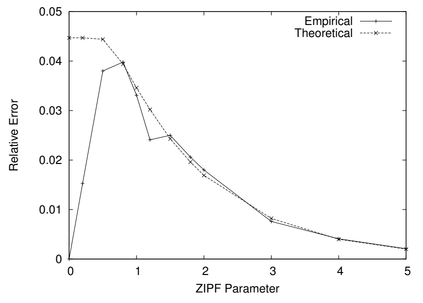

Figure 2: EH3 error

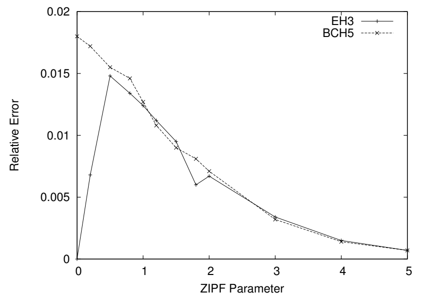

Figure 3: EH3 vs BCH5

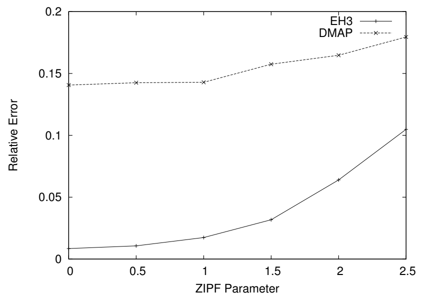

Figure 4: Selectivity estimation

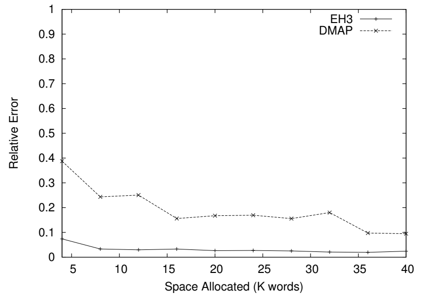

Figure 5: LANDO $\bowtie$ LANDC

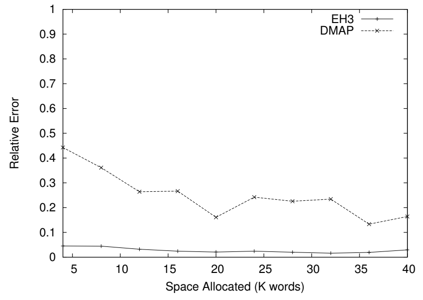

Figure 6: LANDO $\bowtie$ SOIL

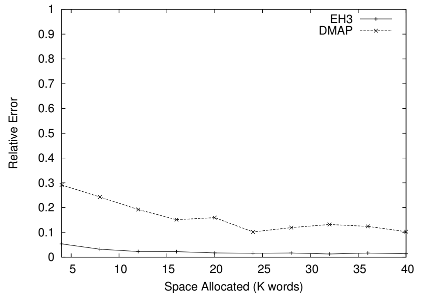

Figure 7: LANDC $\bowtie$ SOIL

computation of F_0 over massive data streams. In *ICDE 2005*, pages 32–43.

[2] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, 1986.

[3] N. Alon, P. B. Gibbons, Y. Matias, and M. Szegedy. Tracking join and self-join sizes in limited storage. *J. Comput. Syst. Sci.*, 64(3):719–747, 2002.

[4] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC 1996*, pages 20–29.

[5] Z. Bar-Yosseff, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA 2002*, pages 623–632.

[6] A. R. Calderbank, A. Gilbert, K. Levchenko, S. Muthukrishnan, and M. Strauss. Improved range-summable random variable construction algorithms. In *SODA 2005*, pages 840–849.

[7] A. Das, J. Gehrke, and M. Riedewald. Approximation techniques for spatial data. In *SIGMOD 2004*, pages 695–706.

[8] A. Dobra, M. Garofalakis, J. Gehrke, and R. Rastogi. Processing complex aggregate queries over data streams. In *SIGMOD 2002*, pages 61–72.

[9] A. Ehrenfeucht and M. Karpinski. The computational complexity of (*xor-and*) counting problems. Technical Report TR-90-033, ICSI, 1990.

[10] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate L^1-difference algorithm for massive data streams. *SIAM J. Comput.*, 32(1):131–151, 2002.

[11] S. Ganguly, M. Garofalakis, and R. Rastogi. Processing set expressions over continuous update streams. In *SIGMOD 2003*, pages 265–276.

[12] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. One-pass wavelet decompositions of data streams. *IEEE TKDE*, 15(3):541–554, 2003.

[13] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. Domain-driven data synopses for dynamic quantiles. *IEEE TKDE*, 17(7):927–938, 2005.

[14] A. S. Hedayat, N. J. A. Sloane, and J. Stufken. *Orthogonal Arrays: Theory and Applications.* Springer-Verlag, 1999.

[15] H. J. Karloff and Y. Mansour. On construction of k-wise independent random variables. In *STOC 1994*, pages 564–573.

[16] M. Karpinski and M. Luby. Approximating the number of solutions of a GF[2] polynomial. Technical Report CS-8544, U. of Bonn, 1990.

[17] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *FOCS 2003*, pages 482–491.

[18] K. Levchenko. `http://www.cse.ucsd.edu/~klevchen/II-2005.pdf`.

[19] M. Luby and A. Wigderson. Pairwise independence and derandomization. Technical Report UCB/CSD-95-880, EECS UC Berkeley, 1995.

[20] MassDAL. `http://www.cs.rutgers.edu/~muthu/massdal.html`.

[21] F. Rusu and A. Dobra. Fast range-summable random variables for efficient aggregate estimation. Manuscript, 2005.

[22] N. Thaper, S. Guha, P. Indyk, and N. Koudas. Dynamic multidimensional histograms. In *SIGMOD 2002*, pages 428–439.