Space efficiency in Synopsis construction algorithms

Sudipto Guha *

Department of Computer and Information Sciences
University of Pennsylvania, Philadelphia PA 19104

Abstract

Histograms and Wavelet synopses have been found to be useful in query optimization, approximate query answering and mining. Over the last few years several good synopsis algorithms have been proposed. These have mostly focused on the running time of the synopsis constructions, optimum or approximate, vis-a-vis their quality. However the space complexity of synopsis construction algorithms has not been investigated as thoroughly. Many of the optimum synopsis construction algorithms (as well as few of the approximate ones) are expensive in space. In this paper, we propose a general technique that reduces space complexity. We show that the notion of "working space" proposed in these contexts is redundant. We believe that our algorithm also generalizes to a broader range of dynamic programs beyond synopsis construction. Our modifications can be easily adapted to existing algorithms. We demonstrate the performance benefits through experiments on real-life and synthetic data.

1 Introduction

Wavelet and Histogram representations are important data analysis tools and have been used in image analysis and signal processing for a long time. Most applications of these techniques consider representing the input in terms of the broader characteristics of the data, referred to as a synopsis or signature. These synopses or signatures, typically constructed to minimize some desired error criterion, are used subsequently in a variety of ways. A few of the highlights include applications in OLAP/DSS systems by Haas *et. al.* [18], in approximate query answering by Amsaleg *et. al.* [2] and Acharya *et. al.* [1], and more recently in mining time series by Chakraborty *et. al.* [5].

Histograms were one of the earliest synopses used in the context of database query optimization [29, 25]. Since the introduction of serial histograms by Ioannidis [19] this area has been a focus of a significant body of research, e.g., [20, 28, 21, 11, 16] among many others. Matias, Vitter and Wang [24] gave one of the first proposals for using Wavelet based synopsis and over the last few years this topic has also received significant attention from different groups of researchers [4, 12, 9, 8, 26]. Histograms and Wavelets are not the only synopses structures – quantiles and samples have been used widely as well. We will not be able to cover the entire literature and point the reader to the survey of Gibbons and Matias [10]. In this paper we will focus on histograms and wavelets mostly, but the ideas have broad applicability.

Most of the existing histogram and wavelet synopsis construction algorithms employ a dynamic programming (DP) approach and several of them are expensive in space. It is sometimes opined that space used by these algorithms is not a problem with modern computers – consider $O(n^2B)$ space. For $B = 50$ and $n = 1024$ and using 4 bytes to represent a number, storing n^2B numbers amounts to 200 Megabytes; at $n = 8192$ we are at 12 Gigabytes. Such an algorithm will have difficulty as n increases, but a large n is exactly the situation where synopses are important. Issues of handling super-linear space have been important in synopsis construction algorithms. Jagadish *et. al.* [21], along with the first polytime V-Optimal histogram construction algorithm, introduced the notion of *working space* in the context of synopsis construction problems. They observed that the optimum algorithm had a local structure and suggested storing only the local information in main memory and storing the rest of the data structures in disk. They also suggested an alternate approach which blows up the running time by a factor of B, but maintains linear space. The working space approach

*Supported in part by an Alfred P. Sloan Research Fellowship and by an NSF Award CCF-0430376. Email:sudipto@cis.upenn.edu

Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

has since been used widely and we shall shortly see three examples in context of synopsis construction problems. However the idea of working space raises more questions – what is the best partitioning of data, the organization on disk, the choice between cache aware or oblivious algorithms, etc. The discussion leads to the area of external memory algorithms; see [3, 30] for surveys.

In this paper, we show that for a number of synopsis construction problems the local structure/information is the only information we need to store. We can recompute the rest of the information without a significant increase in running time. Therefore, in a partial sense, we show that the notion of "working space" is redundant for a variety of these problems. We believe our ideas will find use in a broader class of problems as well.

The space issue assumes more importance in emerging applications such as a sliding window data stream. In a fast data rate situation the total space is best allocated in the main memory. And if we had larger space available then we could store larger size synopses or consider larger size windows, and increase the accuracy of the overall application. This brings us to the central questions we study in this paper.

Question *We consider the following:*

1. *To construct a $O(B)$ size synopsis for $O(n)$ numbers can we design algorithms that use $O(n)$ space? Note that for offline algorithms a space complexity of $O(n)$ is optimal.*
2. *Are there general techniques that apply uniformly to a range of synopsis structures?*
3. *Can we demonstrate that the working space is the total space for many relevant applications?*
4. *Do such techniques allow "more", i.e., improve the running time using cleaner approaches ?*
5. *Are the techniques easily implementable ? Do we actually benefit from these new algorithms?*

We answer "yes" to the questions above. We take a fresh look at the approaches and identify the key ideas that lead us to space efficiency. We focus on three active areas.

Example 1: Histograms. The V-Optimal histogram was introduced by Ioannidis and Poosala in [20]. Given n numbers, and a parameter B, the goal in this problem is to find the best piecewise constant representation of the data with at most B pieces, such that the sum of squares error between the data and the representation is minimized. In [28] this was shown to be a good measure for estimating point queries. Jagadish *et. al.* [21] gave an $O(n^2B)$ time $O(nB)$ space algorithm to find the optimum B bucket histogram synopsis. Their result was based on a DP approach which extends easily to a wide variety of error measures, workloads, etc. We note that this paper is also not concerned with any particular error measure and our ideas extend to the same space of measures as in [21]. As mentioned earlier, the notion of working space was shown to be effective in this context by the authors of [21], and they also proposed an $O(n^2B^2)$ algorithm which uses $O(n)$ space. Their algorithms implied that a "penalty" of $O(B)$ has to be paid in running time or space. We provide an algorithm which uses $O(n)$ space and $O(n^2B)$ time and improve the state-of-the-art, which shows that the working space notion is redundant for these problems.

Unfortunately due to space restrictions we cannot discuss generalizations to other error measures, workloads, range query histograms and various approximations; they can be found in [13].

Example 2: Wavelets Optimization problems Haar Wavelets are highly popular synopsis techniques, and in case of Euclidean error (or its square), the largest (normalized) coefficients of the data produce the best synopsis. This is not true for other measures such as maximum error and the problem remained open. Recently Garofalakis and Kumar [8] proposed an algorithm that uses $O(n^2B \log B)$ time and $O(n^2B)$ total space using a working space of $O(nB)$ to find the best B coefficients of the data which give the smallest maximum error. Since [8] several researchers[26, 23] observed improvements but the total space complexity remained above $O(nB)$.

The same question arises – do we need super-linear space? We show that we can find the best B coefficient that minimize the maximum error (also maximum relative error) in $O(n^2)$ time and $O(n)$ space, independent of B. Therefore, again we show that the "working space" is redundant in this context. For other error measures, e.g., workloads, weighted ℓ_p norms, the running time is $O(n^2 \log B)$. Our result also shows that the *error (not the coefficients)* of choosing the best B coefficients can be solved *simultaneously* for all $0 \leq B \leq n$ in time $O(n^2)$. Thus we can choose the "right" B where the marginal benefit drops by looking at the entire "spectrum". This solves the "dual problem" i.e., given an error bound, find the minimum B such that the best B term synopsis will be within the error bound, also in $O(n^2)$ time and $O(n)$ space. Note, storing all the coefficients in the answer for all $0 \leq B \leq n$ seem to require $\Omega(n^2)$ space – in $O(n)$ space we can compute the coefficients (as well as the error) for any single B or only the error for all B.

The technique we describe here, in the context of wavelet optimization algorithms, applies only to offline algorithms since various quantities are recomputed. In [14] we provide provably guaranteed

streaming approximations for these problems. We also discuss generalizations where the synopsis need not be restricted to coefficients of the data. See also the work of Karras and Mamoulis [22] in this proceeding.

Example 3: Extended Wavelets. Extended wavelets were proposed by Deligiannakis and Rossopoulos [7] for wavelet representations in presence of multiple measures. They proposed a knapsack type computation for the optimum solution in $O(nMB)$ space and working space $O(nM + MB)$ for n items with M measures and size of synopsis B. In a recent work, [17], we were able to improve the space complexity of the optimum solution to $O(B^2 + BM)$. Our algorithm showed that there were a few, $O(B)$, "critical" elements in the data and the optimum algorithm only needed to consider them. But the central problem remained the same on those "critical" elements. The state-of-the-art is a fundamental $O(nB)$ behaviour of the optimum algorithm (plugging $n = B$). Using the techniques in this paper, we improve the total space complexity to $O(BM)$, which was the "working space" (for $n = B$) thereby showing that the working space notion is redundant in this case as well.

Approximation Algorithms: So far, we have discussed optimum algorithms only. The space issue is also important in approximation algorithms – the discussion on Extended Wavelets provides some intuition. Most approximation algorithms reduce a search space by not considering all the elements. For Extended Wavelets the reduced set was also optimal. In general for approximation algorithms, the reduced search space achieves near optimality, e.g., in approximate histograms [16] the search space to extend the DP table by one was reduced from $O(n)$ to $O(B\epsilon^{-1}\log n)$. In [15, 27] it was shown that finding the best solution which is restricted to $O(B\epsilon^{-2}\log^2 n)$ well chosen boundary elements, gives us a near optimum solution for the entire dataset. But the central issue of space did not disappear - it is no surprise that the space complexity of the algorithms are $O(B^2\epsilon^{-1}\log n)$ and $O(B^2\epsilon^{-2}\log^2 n)$ (unless we pay the factor B in the running time) and we are back at the same situation where we were with the optimum algorithms. The central idea of the paper does carry over to approximate histograms. However they require significantly more technical details. In the interest of space we limit ourselves to optimum algorithms for the rest of this paper. The interested reader is referred to [13].

1.1 Our contributions and overview:

As mentioned earlier, our main contribution is a broad technique that improves a significant number of synopsis construction problems. More specifically

- We present the general framework and the key ideas in Section 3.
- We present **three** examples where our ideas improve the algorithms for optimum synopsis construction. We focus on:
  1. Histograms, discussed in Section 4. We give the first $O(n^2B)$ time $O(n)$ space optimum algorithm which applies to a large number of error measures.
  2. Wavelet reconstruction, discussed in Section 5. We give the first $O(n)$ space algorithm for a wide variety or error measures including all known measures. The running time is $O(n^2)$ for the maximum error (relative, absolute, or weighted/workload) measures and $O(n^2 \log B)$ for the others which involve sum.
  3. Extended Wavelets, discussed in Section 6. We reduce the space requirement of the algorithm to $O(BM)$ from $O(MB + B^2)$.
- We demonstrate that the approach is not merely theoretical – there are no large constants hiding in $O()$ notations. Using synthetic and real life data sets which were used in previous papers we show that the space efficient algorithms give us the benefit they promise. In case of Wavelets, our algorithm was superior to alternatives.

Note that we do not discuss quality of synopses across techniques or error measures, which are of independent interest. Our goal in this paper is to develop space efficient algorithms which apply to a multitude of measures and synopsis techniques.

2 Definitions and Preliminaries

The V-Optimal histogram construction problem is:

Problem 1 *Given a set of n numbers $X = x_1, \ldots, x_n$ the problem seeks to construct a B piecewise constant representation (function) H such that $\|X - H\|_2^2 = \sum_i (x_i - H(i))^2$ is minimized. Each "piece" is a bucket and defines a subrange $[p, q]$ of the range $[1, n]$.*

Many other error measures exist, notably workloads or weighted ℓ_p norms, maximum error $\|X - H\|_\infty = \max_i |x_i - H(i)|$ etc. Several researchers have proposed the relative error metric which uses some function of $\frac{x_i - H(i)}{\max\{|x_i|, c\}}$ to compute the error, e.g., maximum relative error $\max_i \left|\frac{x_i - H(i)}{\max\{|x_i|, c\}}\right|$. The definitions also extend to range query histograms which we discuss in [13].

Wavelets are multi-resolution representations of the data. There is a huge literature on this topic, including excellent textbooks [6]. Most of the database literature

focus on Haar wavelets, due to their simplicity and the existence of fast algorithms for transforming the data to a wavelet representation. The inverse transformation is even simpler, it is simply the addition of at most $\log n$ numbers. Haar Wavelets represent the data X (of length n, assumed power of 2) by the set of orthogonal vectors, ψ_i, defined below:

$$\psi_1(j) = 1 \quad \text{for all } j$$
$$\psi_{2^s+t}(j) = \begin{cases} 1 & \text{if } (t-1)\frac{n}{2^s}+1 \le j \le \frac{tn}{2^s} - \frac{n}{2^{s+1}} \\ -1 & \text{if } \frac{nt}{2^s} - \frac{n}{2^{s+1}} + 1 \le j \le \frac{tn}{2^s} \end{cases}$$
$$(1 \le t \le 2^s, 1 \le s \le \log n)$$

For $n = 4$ the (non-normalized) $\{\psi_i\}$ are shown below

$$\{1,1,1,1\}, \{1,1,-1,-1\}, \{1,-1,0,0\}, \{0,0,1,-1\}$$

and they extend naturally to larger powers of 2. The inverse transform of Z is defined as $\mathcal{W}^{-1}(Z) = \sum_i Z_i\psi_i$; it is convenient that $\mathcal{W}^{-1}(Z)_i$ can be computed in $O(\log n)$ time without generating the full inverse. To compute $\mathcal{W}(X)$, we compute the average $\frac{x_{2i+1}+x_{2i+2}}{2}$ and the difference $\frac{x_{2i+1}-x_{2i+2}}{2}$ for each pair of consecutive elements as i ranges over $0, 2, 4, 6, \ldots$. The difference coefficients form the last $n/2$ entries of $\mathcal{W}(X)$. The process is repeated on the $n/2$ average coefficients – *their* difference coefficients yield the $n/4+1, \ldots, n/2$'th coefficients of $\mathcal{W}(X)$. The process stops when we compute the overall average, which is the first element of $\mathcal{W}(X)$. The **support** of a vector ψ_i is the set $\{j|\psi_i(j) \neq 0\}$. These support sets are nested and are of cardinality which are powers of 2 (dyadic). Therefore we naturally have a complete binary "coefficient tree". The x_j correspond to the leaves, and the coefficients correspond to the non-leaf nodes of the tree. Assigning a value c_i to a coefficient corresponds to assigning $+c_i$ to all leaves j that are **left descendants** (descendants of the left child) and $-c_j$ to all right descendants. The wavelet synopsis construction problem is defined below:

Problem 2 (Minimize Error) *Given a set of n numbers $X = x_1, \ldots, x_n$ the problem seeks to choose at most B terms from the wavelet representation $\mathcal{W}(X)$ of X, say denoted by Z, s.t. a suitable function of $X - \mathcal{W}^{-1}(Z)$ denoting the error is minimized. For example, $\|X - \mathcal{W}^{-1}(Z)\|_\infty$ is the minimum maximum absolute error problem.*

The above definition extends to a broader class *, e.g., weighted-ℓ_p/workloads. As mentioned earlier, we can also solve the dual problem.

Problem 3 (Minimizing synopsis size) *Given a target error ϵ find the wavelet reconstruction with smallest number of coefficients that is within the error bound for a suitable error function.*

The definition of Extended wavelets is in Section 6.

*Note that $\|X - \mathcal{W}^{-1}(Z)\|_2^2$ follows from Parseval's equality.

3 The general technique

The fundamental basis of our approach is the following: we want to use a divide and conquer algorithm. However simple divide and conquer (think merge-sort) does not work in these scenarios. There are two main problems in synopsis construction scenarios.

- Partition. For example, we may take the following approach: we find the best b-bucket histogram for the range $[1, \ldots, \frac{n}{2}]$ and the range $[\frac{n}{2}+1, \ldots, n]$ for all $b \le B$ and try to merge them to find the best B bucket histogram for $[1, \ldots n]$. The idea fails because the optimum solution may not conform to a bucket ending at $\frac{n}{2}$! Apriori it is not clear where any of the bucket boundaries will be.

- Interaction. The above problem of bucket boundary disappears in the context of Wavelets, since the boundaries are aligned since the set $\{j|\psi_i(j) \neq 0\}$ has cardinality a power of 2 (dyadic). But a different problem arises: the wavelets are hierarchical and overlap. Thus in our simple strategy, there will be interaction between the two subproblems through their ancestors.

The problem is that we do not know how to divide the problem into manageable subproblems, i.e., eliminate interaction and partition problems simultaneously. We will use the following strategy:

We will use a dynamic program to find the interface – the paradigm can be viewed as Dynamic Programming meeting (being used for) Divide and Conquer.

For histograms the interface would be the boundary bucket which contains the partition; for wavelets this would be the interaction with the sibling. It may appear that we have achieved circularity, we have avoided a DP based solution to use another DP ! But the central observation is:

If the interface is small, i.e., can be specified succinctly, the new DP can be smaller.

The reader has probably guessed the game-plan by now, we will write a much smaller DP to find the interface and then recurse in each part. The parts can reuse the same space, since we will only be solving at most one subproblem at a time ! The above is easier said than done. We need to answer the following:

- *Does such an interface exist ?* Given a problem at hand, this will be the first step.

- *Can the interface be described succinctly ?* This affects the simplicity and the implementability of any algorithm as well as space complexity.

- *How to compute it ?* This would often involve solving for part of the original problem, e.g., if someone told us that the minimum error was 10, can we use the fact ? The answer is, no, we cannot use the fact that the error is 10. But if one could "prove" that 10 was the minimum – we can use that proof to find a good division/interface. The fact that we also know the error is an accident. We shall shortly see examples, but thinking ahead, a DP is a recursive "proof".

The central part of the paper will be the following general theorem, the proof of the theorem is almost self evident, the difficulty is in applying it.

Theorem 3.1 *Suppose there exists a partitioning that decomposes the problem into two subproblems of size no more than half the original, and we take $g(n, B)$ time to find the partitioning. Then the overall synopsis construction problem can be solved in time $f(n, B)$, where $f(n, B)$ is at most $g(n, B) + 2f(\frac{n}{2}, B)$, using a divide and conquer approach. In fact the running time can be further tightened to*

$$f(n, B) \leq g(n, B) + \max_{B_1 \leq B} \{f(\frac{n}{2}, B_1) + f(\frac{n}{2}, B - B_1)\}$$

The proof follows if we assume $f(n, B)$ is monotone non-decreasing in n, B.

4 Example I: Histograms

Given a set of n numbers $X = x_1, \ldots, x_n$ the V-Optimal histogram construction problem searches for a piecewise constant representation H with at most B pieces which minimizes $\|X - H\|_2^2$. In [21] a $O(n^2B)$ time algorithm was given to find the optimum histogram using $O(nB)$ space. They also observed that the space could be reduced to $O(n)$ at the expense of increasing the running time to $O(n^2B^2)$.

Several different optimization criteria have been proposed for histogram construction, e.g., ℓ_1, ℓ_∞, weighted variant/workloads, relative error, χ^2 to name a few, as well as extension to piecewise polynomials. However they are solved by a similar DP and the V-Opt histograms provide an excellent foil to discuss all of the measures at the same time.

4.1 V-OPT Algorithm of [21]

Jagadish *et. al.* [21] show that if a "bucket" or range $[j + 1, \ldots, i]$ is approximated by one value, as is the case in histograms, the best value v which minimizes the error $e(j + 1, i) = \sum_{r=j+1}^{i} (x_r - v)^2$ is the mean $\sum_{r=j+1}^{i} x_r$. Based on this they gave a natural DP algorithm which is given in Figure 1.
To compute $e(j + 1, i)$ in $O(1)$ time we maintain two arrays $\text{SUM}[i] = \sum_{r=1}^{i} x_i$ and $\text{SQ}[i] = \sum_{r=1}^{i} x_i^2$.

$$e(j+1, i) = \text{SQ}[i] - \text{SQ}[j] - (\text{SUM}[i] - \text{SUM}[j])^2/(i - j)$$

```
Algorithm VOPT
1.  Let E[i, b] = min. error b bucket histogram for [1, ..., i].
2.  Initially E[i, 1] = e(1, i) for all 1 ≤ i ≤ n
3.  For b = 2 to B do
4.      For i = 2 to n do {
5.          E[i, b] = ∞
6.          For j = i − 1 downto 1 do {
7.              E[i, b] = min{E[i, b], E[j, b − 1] + e(j + 1, i)}
                /* perform book-keeping if minimum is changed */
8.              (Optional) If (E[i, b] < e(j + 1, i)) break;
9.          } }
```

Figure 1: The VOPT algorithm

The strength of the above algorithm is its generality; it operates with virtually any reasonable definition of error in Line (7) as long as we can compute $e(j + 1, i)$ efficiently. The $O(n^2B^2)$ algorithm uses the array $E[*, b - 1]$ to construct $E[*, b]$ but discards $E[*, b - 1]$ immediately. It performs no bookkeeping since it cannot store $O(nB)$ items. So when the algorithm computes $E[*, B]$, it uses the minimizing j, i.e., $argmin_{1 \leq j \leq n}\ E[j, B - 1] + e(j + 1, n)$, to recursively find the buckets for $[1, \ldots, j]$. The optional statement in Line (8) significantly improves performance since it avoids useless searching.

4.2 Applying Our paradigm

We will try to find the "interface" or the bucket that "contains" $\frac{n}{2}$.

Definition 4.1 Given n, B, define the *middle bucket* $MB(1, i, b)$ to be the bucket $[p, q]$ in the optimal b-bucket histogram for $[1, \ldots, i]$ that contains $\lfloor \frac{n}{2} \rfloor$. If that optimum solution uses b buckets to the left of p, we specify the middle bucket by the triple (p, q, b).

Lemma 4.2 *If $MB(1, n, B)$ is the triple (p, q, b) then the partitioning $[1, p-1]$ and $[q+1, n]$ satisfy the three criterion in the previous section.*

Proof: Follows immediately if we show an $O(n^2B)$ time $O(n)$ space algorithm to compute the triple. ∎

Assuming that such an algorithm exists (which we will see shortly), we can apply the Theorem 3.1 and the running time of the overall algorithm $f(n, B)$ can be bounded by

$$f(n, B) \leq g(n, B) + 2f(\frac{n}{2}, B)$$

Since $p \leq \frac{n}{2} \leq q$ the two subproblems on $[1, \ldots, p - 1]$ and $[q + 1, \ldots, n]$ are of size at most $\frac{n}{2}$. If the running time of the algorithm that finds the division is $g(n, B) = an^2B$ for some constant a, then $f(n, B) \leq 2cn^2B$ solves the above equation since

$$\begin{aligned} f(n, B) &\leq an^2B + 2f(\frac{n}{2}, B) \\ &\leq an^2B + 2a(\frac{n}{2})^2B + 4f(\frac{n}{4}, B) \end{aligned}$$

$$\leq an^2B + \frac{an^2B}{2} + 4f(\frac{n}{4}, B)$$
$$\leq an^2B + \frac{an^2B}{2} + \frac{an^2B}{4} + \cdots \leq 2an^2B$$

Therefore summing up, if we can find the **middle bucket** in time $O(n^2B)$ and space $O(n)$ then we can solve the V-Optimal histogram problem in time $O(n^2B)$ and space $O(n)$ using our framework (the subproblems can use the same space).

4.3 Finding Middle-bucket efficiently

Lemma 4.3 *Suppose the last bucket for the optimum b-bucket histogram for $[1, \ldots, i]$ is $[j+1, i]$. If $j < \frac{n}{2}$ then $MB(1, i, b+1) = (j, i, b)$ otherwise $MB(1, i, b+1) = MB(1, j, b)$.*

Proof: If $j \geq \frac{n}{2}$, the fact that any prefix of the buckets in an optimum histogram are optimum for their range (otherwise, the overall error would go down by choosing a different histogram) allow us to conclude that $MB(1, j, b)$ is also the bucket that contains $\frac{n}{2}$ for the best $b+1$ bucket histogram of $[1, i]$. In the other case, $MB(1, i, b+1) = (j, i, b)$ by construction. ∎

The above lemma gives us a **DP** for computing $MB(1, i, b)$. The pseudocode is given in Figure 2. The overall idea is to construct $M(1, n, B)$ from $MB(1, n, B-1)$. The array $M[i]$ keeps track of the $MB(1, n, b)$ for various b and is initialized for $b = 1$. Note that we are computing $E[i, b]$ in $newE$, and $MB(1, i, b)$ in $newM[i]$. We are using them to compute the analogous quantities for $b+1$. Observe that we know the minimum error in $E[n]$ at the end of the computation – but we do not know the buckets that give the error. We need the divide and conquer strategy to give us the buckets.

```
Algorithm Middle-Bucket(1,n,B)
1.  For i = 2 to n {
2.      M[i] ← (1, i, 0)
3.      E[i] ← e(1, i)
4.  }
5.  For b = 2 to B do {
6.      newE[i] = ∞
7.      For i = 2 to n do {
8.          For j = i − 1 downto 1 do {
9.              If (newE[i] > E[j] + e(j + 1, i)) {
10.                 newE[i] = E[j] + e(j + 1, i)
11.                 If (j ≥ n/2) then newM[i] ← M[j]
12.                 else newM[i] ← (j + 1, n, b − 1)
13.             }
14.             (Optional) If (newM[i] < e(j + 1, i)) break;
15.         }
16.         M ← newM; E ← newE;
17.     }
18. Now M[n] contains MB(1, n, B)
```

Figure 2: Algorithm to find $MB(1, n, B)$

Therefore we are ready to conclude the following

Theorem 4.4 *We can compute the V-Optimal histogram in time $O(n^2B)$ and space $O(n)$ by repeatedly finding the middle bucket and solving the two subproblems.*

```
Algorithm Wavelet Reconstruction
1.  We proceed bottom to top in the coefficient tree.
2.  At each node i (assuming the children are i_L and i_R),
    set E[i, b, S] as follows
min { min_{b'} max{E[i_L, b', S ∪ {i}], E[i_R, b − 1 − b', S ∪ {i}]}
    { min_{b'} max{E[i_L, b', S], E[i_R, b − b', S]}
```

Figure 3: The algorithm in [8].

Implementation details: The above pseudocode is almost the actual code. We need a stack to keep track of the recursion. If we are solving to find a $b > 1$ bucket histogram for $[s, t]$, which is at the top of stack, we find the middle bucket (the 1 in the pseudocode is s and $t = n$, so $\frac{n}{2}$ is $\frac{t-s}{2}$ for this subproblem); if that is (p, q, b'), then we pop the top of stack and push a b'-bucket problem for $[s, p-1]$ and a $b - b' - 1$ bucket problem for $[q+1, t]$ to the stack. If $b = 1$, we have our bucket ! We simply pop and output it.

5 Example II: Wavelets

Recall that the Wavelet construction problem is that given a set of n numbers $X = x_1, \ldots, x_n$ the problem seeks to choose at most B terms from the wavelet representation $\mathcal{W}(X)$ of X, represented by Z s.t. a suitable error, e.g., $\|X - \mathcal{W}^{-1}(Z)\|_\infty$ is minimized.

5.1 Previous algorithm(s)

Recall that the interaction between two sibling intervals in the coefficient tree was through their ancestors. The idea is to enumerate all possible interactions. We follow the description of Garofalakis and Kumar [8]. The value of $\mathcal{W}^{-1}(Z)_j$ is fixed by the choices of all coefficients i such that j belongs to the support of i. Suppose S is a subset of the ancestors of a coefficient i. A natural DP emerges where $E[i, b, S]$ is defined to be *the minimum error such that exactly b coefficients that are descendants of i are chosen along with the coefficients of S*. The algorithm is given in Figure 3.

We can extend the algorithm to handle other errors by changing $\min_{b'} \max$ to $\min_{b'} \sum$, i.e., adding the contribution from the children. Clearly the number of entries in the array $E[]$ is Bn times 2^r where r is the maximum number of ancestors of any node. It is easy to see that $r = \log n + 1$ and thus the number of entries (space) is n^2B. For maximum relative error, we may perform binary search for the minimization and only need $\log B$ time (see [8]) and the overall time is $O(n^2B \log B)$. The running time increases to $O(n^2B^2)$ for other measures.

5.2 Dynamic Program to Divide and Conquer

As our game-plan suggests, we will write a DP to discover the "interface". In this case, the "interface"

```
Algorithm WaveOpt(i,v)
    /* Computes the best way of allocating 0 ≤ b ≤ B
    coefficients to the two subtrees at a node i. Returns
    an array of size B with the partition info, error etc.

1.  If (i == root) { /* root has no ancestor so v = 0 */
2.      A ← WaveOpt(child, W[root]);
3.      C ← WaveOpt(child, 0);
4.      We return the array D[b] = min{A[b − 1], C[b]}
5.      D[b] stores the error as well as if it was from A,C
6.  } else {
7.      If (i == leaf) Return D[b] = |x[i] − v| for all b
8.          /* change for other errors */
9.      else {
10.         A ← WaveOpt(i_L, v + w[i]);
11.         C ← WaveOpt(i_R, v − w[i]);
12.         P ← WaveOpt(i_L, v);
13.         Q ← WaveOpt(i_R, v);
14.         For b = 0 to B do {
15.             Let t1[b] be the best error if w[i] is chosen
16.             t1[b] = min_{0≤r≤b−1} max{A[r], C[b − 1 − r]}

17.             Likewise, t2[b] = min_{0≤r≤b} max{A[r], P[b − r]},
18.             for the case w[i] is not chosen
19.         }
20.         Return array D[b] = min{t1[b], t2[b]}. D[b] stores
21.         the error, the split r, and min was from t1 or t2.
22.     }
```

Figure 4: The algorithm WaveOpt

consists of two pieces of information (i) Is the parent chosen ? (ii) How are the coefficients divided between the two subtrees? Immediately, we come across a problem; (i) is recursive! The parent depends on its parent and so on. We definitely do not want to consider so many cases since we want "small interfaces". We observe that the dependence on the parent (and through it, on its parent) can be expressed in one number v as a sum or bias introduced by a combination of all of them. But we do not want to write down all possibilities, since there will be at least $O(n)$ of them for each node. The main idea we use is:

We can recurse within the DP to generate the set on the fly (without storing them).

We note that although we analyze the same cases as before, the sophistication of the recurse-within-DP is higher than the previous algorithm. This is a new algorithm in itself, and further, is needed to get to a much better result. The overall implementation is not complicated - the pseudocode is presented in Figure 4. The top-level call is $WaveOpt(root, 0)$. The next lemma follows from induction.

The top-level call is $WaveOpt(root, 0)$. But before we proceed further, we must assure ourselves that this recursion-within-DP does not go out of hand.

Lemma 5.1 *Each node with ℓ ancestors is called at most $2^\ell < 2n$ times. Therefore the running time of $WaveOpt(root, 0)$ is $O(n^2 B \log B)$.*

Now $WaveOpt(root, 0)$ returns the information of the split and if the root coefficient was chosen for the best solution. Based on that information, we could recursively compute the coefficients – *but we did not do so since we intended this algorithm to serve as a baseline representing all algorithms that computed the same table as [8].*

Implementation Details: The new algorithm simply needs a stack. At most one $WaveOpt$ call will be active between children of the same parent – the depth of the stack would be $O(\log n)$. Therefore for $WaveOpt(root, 0)$ we need $O(B \log n)$ space.

5.3 Further Optimization: SpaceWave

We make the following observation: if a node has t descendants including itself, then at most t (always be a power of 2) coefficients can be chosen in its subtree! We define the algorithm $Spacewave(i, v, t)$ which is almost the same as described in Figure 4 except it uses a *localB* which is $\min\{B, t\}$ at the lines $4, 14, 20$ and passes $t/2$ to its children on lines 10–13. Note that we compute a significantly smaller size table overall compared to [8].

Lemma 5.2 *If a node has ℓ ancestors and t descendants (including itself) then $2^\ell t = 2n$.*

Thus we return arrays of size $\min\{B, t\}$ from $SpaceWave(i, v, t)$ – since more than t coefficients cannot be chosen. A node with ℓ ancestor needs time

$$2^\ell \min\{B, t\} \log \min\{B, t\}$$

Number of nodes with $\ell > 1$ ancestors is at most 2^ℓ. Thus the total time taken over all the nodes is

$$\sum_{\ell=0}^{\log n+1} 2^\ell 2^\ell \min\{B, t\} \log \min\{B, t\}$$

The worst case is $B = n \geq t$. Using Lemma 5.2 and change of variables, $r = (1 + \log n) - \ell$, we get

$$\sum_{\ell=0}^{\log n+1} 2n2^\ell \log \frac{2n}{2^\ell} = \sum_{r=0}^{\log n+1} \frac{4n^2 r}{2^r} = O(n^2)$$

Observe the above holds for all B. For errors like workload/weighted-ℓ_p norms the the sum is over $2^{2\ell} \min^2\{B, t\}$ which evaluates to $O(n^2 \log B)$. The space required is likewise $\sum_{\ell=0}^{\log n+1} \min\{B, t\}$ which is $B(\log n - \log B)$ from the part where $\ell < \log n - \log B$. For the other part t forms a geometric series in decreasing powers of 2 as ℓ increases (From Lemma 5.2) and adds up to $O(B)$. Observe $B(\log n - \log B) = B \log(n/B) \leq n$ (for $B \leq n$)

Theorem 5.3 *We can solve the maximum error wavelet reconstruction in time $O(n^2)$ and space $O(n)$ using SpaceWave(). For other errors such as weighted-ℓ_p or workloads the running time is $O(n^2 \log B)$.*

Implementation Details: We know the "split" in the top level and allocation of coefficients to the halves as well as if the root coefficient is chosen. In fact is all numbers are non-negative the root must be chosen. We use an additional stack to store the subproblems – at most one subproblem is solved at a time. Further in the recursive calls in the subproblems, while we search for b coefficients we can set *localB* to be $\min\{b, t\}$.

The dual Problem: As mentioned in the introduction, the above algorithm allows us to construct the entire spectrum of error. Using the spectrum we can compute the minimum number B' of coefficients required for a target error ϵ. Using this minimum number B' found, we can now compute the B' coefficients that minimizes the error. The entire algorithm can be made to work in $O(n)$ space. We can summarize:

Theorem 5.4 *We can solve the dual wavelet reconstruction problem in time $O(n^2)$ and space $O(n)$ for maximum (relative or absolute) error metrics. For other metrics the running time is $O(n^2 \log n)$.*

However in terms of implementations the dual optimization can use significant amount of pruning – we relegate the discussion of pruning techniques to [13].

6 Example III: Extended Wavelets

Extended wavelets were introduced by Deligiannakis and Rossopoulos in [7]. They point out that for multi-measure data, there can be significant saving of space if we use a non-standard way of storing the information. There are several standard ways of extending 1-dimensional (Haar) wavelets to multiple dimensions, but irrespective of the number of dimensions, the format of the synopsis is a pair of numbers *(coefficient index,value)*. In multiple dimensions the size of the coefficient index is larger in bits, whereas in 1-dimensional transforms taken independently, we may not be taking correlations into account effectively. In Extended Wavelets we perform wavelet decomposition independently in each dimension. We then store tuples consisting of the coefficient index, a bitmap indicating the dimensions for which the coefficient in that dimension is chosen, and a list of values. Since the coefficient number and the bitmap is shared across the coefficients, we can store more coefficients than a simple union of unidimensional transforms or a full-fledged multi-dimensional representation.

Notice that there is no interaction between the benefits of storing a subset of coefficients for i and a subset for i'. The problem reduces naturally to a **Knapsack** problem where each item (coefficient i) can be present in increasing sizes $s_{i1}, s_{i2} \ldots, s_{iM}$ (which are integers) with increasing profits $p_{i1}, p_{i2} \ldots, p_{iM}$ (arbitrary) where M is the number of measures. This allows a dynamic program of $O(nMB)$ time and space, [7]. In [17] we reduce the interesting set of items from n to $O(B)$ – but the inherent problem remains the same.

Algorithm Ext-Opt
1. **For** $i = 1$ to n do
2. **For** $b = 0$ to B do {
3. $E[i, b] = \max\left\{ E[i-1, b], \max_{1 \le j \le M}\{E[i-1, b-s_j] + p_j\} \right\}$

Figure 5: The DP in [7, 17]

Due to shortage of space, we only indicate the broad picture. There are significant details which can be found in [13]. The central idea is to find in $O(n)$ time a partitioning as in histograms. But for this problem we need a *double-cut* which partitions both n and B and creates three pieces. In $O(nMB)$ time we can find the space allocation to all the pieces and get a $O(nMB)$ time $O(B+nM)$ space algorithm improving [7, 17].

7 Experimental Results

The main issue we investigated is the effect of the various additional data structures that were defined for the space efficient algorithms. We use both synthetic and real life data used in previous papers [9, 17, 16]. We did not write data to disk for the working space based algorithms – since organizing that data in the disk well is a nontrivial task and influences the running time of the working space based algorithms. For purposes of comparison we implemented baseline algorithms that compute the errors only.

We first describe the data sets and then in Section 7.2 we present the results on V-Optimal histograms. In Section 7.4 we present the result on maximum error wavelets. All experiments reported in this section were performed on Pentium-4 1.8 GHz machine with 1 GB of main memory, running RedHat Linux 9.0. All the methods were implemented using GCC compiler of Version 3.2.2.

See *http://www.cis.upenn.edu/∼sudipto/synopsis.html* for some of the executables and more discussion.

7.1 Data Sets

Synthetic Data Sets: The synthetic data sets were generated with Zipfian frequencies for various levels of skew that is controlled by the z parameter of the Zipfian. The z parameter values between 0.3 (low skew) and 1.5 (moderate skew), the distinct values between 256 ($= 2^8$) and 16384 ($= 2^{16}$), and the tuple count was set to 1,000,000. Note that the time and space complexities are not dependent on number of tuples and thus we did not vary this parameter. A permutation step was also applied on the produced Zipfian frequencies to decide the order of frequencies over the data domain. We show results with the `Normal` permutation as described in [9, 16].

Real Life Data Sets: We also experimented with real-life data sets. We used the *Cover Type* data set from the National Forest Service, which was downloaded from UC Irvine[†]. There are 581,012 tuples in the data set. Among 54 attributes, we report "hillshade3pm" (`CovType-HS3`) and "aspect" (`CovType-A`). Because these attributes have widely different distributions, they were used for performance study in [9]. `CovType-HS3` measures a hillshade index (from 0 to 255) at 3 pm on the summer solstice. Its histogram is bell-shaped and relatively smooth. `CovType-A` has uniformly spread distribution with a pipe-organ-style fluctuation and considerable peaks of noise. It had 360 distinct values and was padded with 0 to make the same dataset amenable for wavelets. To show scale up experiments on real life data we use the Dow-Jones Industrial Average (DJIA) data set available at StatLib[‡] that contains Dow-Jones Industrial Average (DJIA) closing values from 1900 to 1993. There were a few negative values (e.g. -9), which we removed. We focused on prefixes of the dataset of size upto 16384.

7.2 Experimental study: Histograms

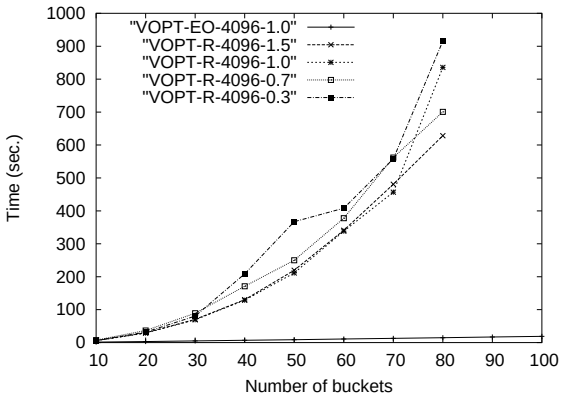
We compared the following:

- **VOPT-EO:** This is a baseline implementation which uses $O(n)$ space, and does not maintain the table (and therefore cannot compute the buckets). The running time of the algorithm is a good lower bound on all $O(n)$ space algorithms.
- **VOPT-R:** It represents the $O(n^2B^2)$ time and $O(n)$ space algorithm given in [21] which computes the error as well as the buckets. This algorithm uses the VOPT-EO algorithm recursively.
- **SpaceOpt:** It represents the space efficient algorithm given in Figure 2.

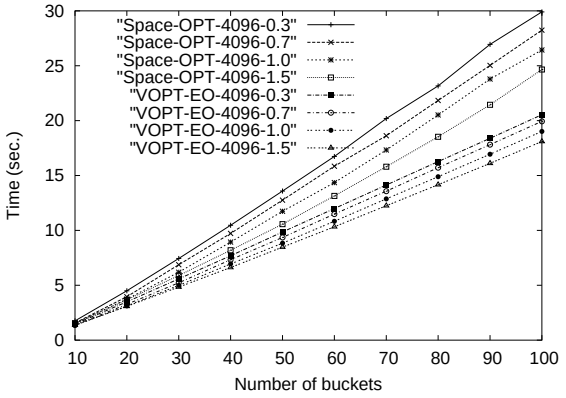
The algorithms compute the same quantity and we present the running times under various situations to see the increase in running time due to our recursive approach. We do not show the space comparison since the space usage of all the algorithms is almost exactly the size of the table times the size of a double/int. The space usage of the $O(nB)$ space V-Opt algorithm jumped from $2.8MB$ to $20MB$ as B was raised from 10 to 100. The algorithms VOPT-EO and VOPT-R used $1.4MB$ and SpaceOpt used $1.6MB$ for entire range of settings of B. The space benefit is quite clear and as is the theme of this paper, we focus only on the algorithms which have $O(n)$ space complexity.

Dependence on Skew: In Figure 6 we present the running times of the algorithms for $n = 4096$ for various settings of the skew in the synthetic data.

[†]See ftp://ftp.ics.uci.edu/pub/machine-learning-databases.
[‡]See http://lib.stat.cmu.edu/datasets/djdc0093.



(a) Showing VOPT-R



(b) Showing VOPT-EO vs Space-OPT

Figure 6: Running times for V-Opt and Space-Opt

Since the algorithm VOPT-R is much slower than the others to be attractive, we do not include that algorithm for any further experiments.

Dependence on n: We also investigate the dependence of n keeping $B = 50$ a fixed constant, this is presented in Figure 7. We show the graph for $skew = 1$. The separation between the lines is insufficient to show the results for more than one skew value in the graph, results for other skew settings are similar.

Real life data sets: We show the performance for fixed size real life data sets in Figure 8. The scale up experiments on the prefixes of DJIA data set is shown in Figure 9.

7.3 Summary: Histograms

- Figure 6 shows clearly that the running time of VOPT-R has quadratic dependence on B, it is $O(n^2B^2)$, and the other algorithms have linear dependence on B. The conclusion holds across different settings of skew and B. This shows that SpaceOpt is a significantly better algorithm compared to the previous space efficient algorithms.

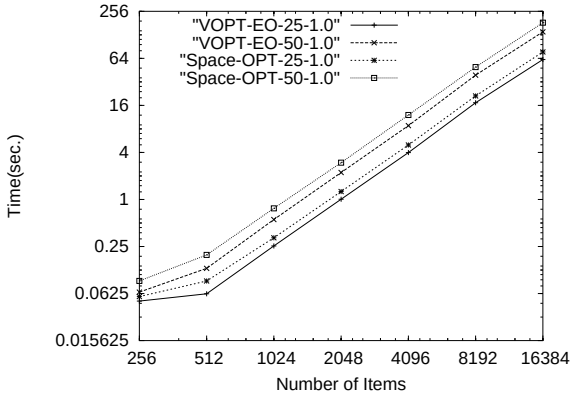


Figure 7: Running times for $B = 50$

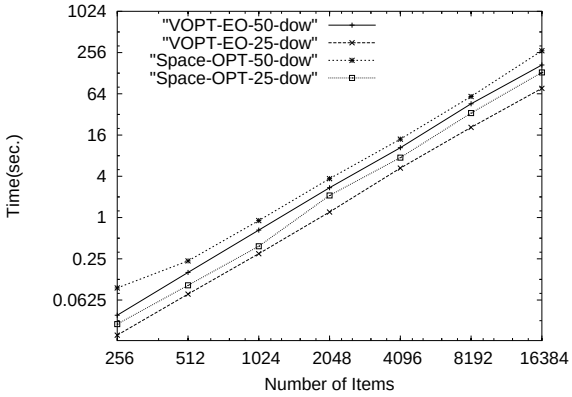


Figure 9: The DJIA dataset

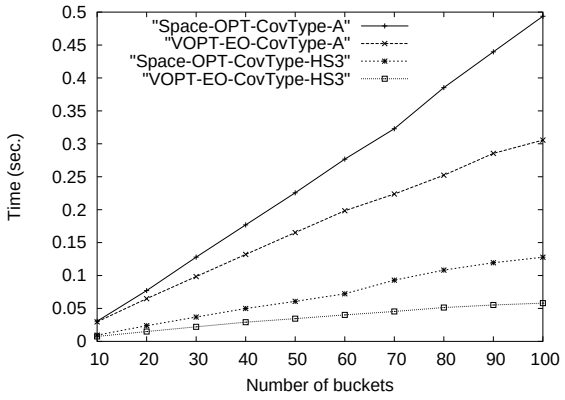


Figure 8: Running times for real life data sets

- Figures 7 and 9 show the running time of VOPT-EO and Space-OPT on a log-log scale, for synthetic and real life data the dependence is almost quadratic in n (as expected).
- As expected, for the same setting of B in synthetic (Figure 7) and real life (Figure 8) data, VOPT-EO was always faster than SpaceOpt.

The experiments confirm that SpaceOpt is slower than the baseline algorithm. However, baseline algorithm only tells us the error whereas SpaceOpt returns us the entire histogram! We lose a small (but fixed) factor in running time and save space by a factor of B.

7.4 Experimental study: Wavelets

For the Wavelet synopsis construction problem we show only the results for the maximum relative error measure. We could not use the $O(n^2B)$ space algorithm described in [8] since we ran out of memory quickly. As mentioned earlier, choosing parameters of a working space based algorithm is always unclear and biases comparisons; we eschewed the approach. We therefore restrict ourselves to comparing the $O(B \log n)$ space *WaveOpt()* algorithm and the optimized algorithm *SpaceWave()*. As discussed earlier, the algorithm *WaveOpt()* serves as a lower bound/baseline for any algorithm that computes the same table as in [8]. We reiterate that the algorithms compute the same quantity and we only compare the running times.

- **WaveOpt:** This algorithm is described in Section 5.2. Since the algorithm is only a baseline, as mentioned earlier, we did not compute the coefficients/answer but the final error only.
- **SpaceWave:** It represents the space efficient algorithm with the optimization mentioned in Section 5.3. This algorithm computes the error as well as the coefficients, and the final reconstruction based on those coefficients.

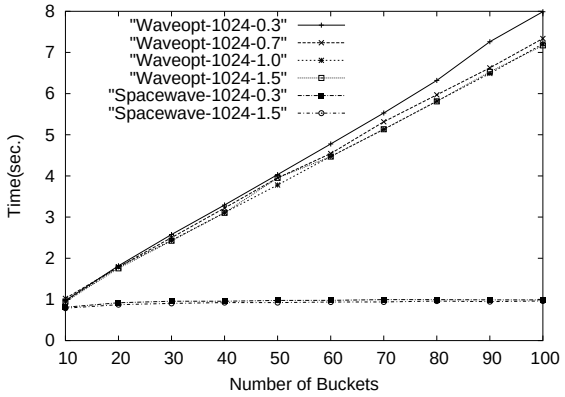
Dependence on Skew We present the comparison of running times for various settings of the skew parameter in Figure 10. We will only present the results for skew setting 1 in the rest of this section.

Dependence on n: We present the comparison of running times if B is fixed and we vary n Figure 11.

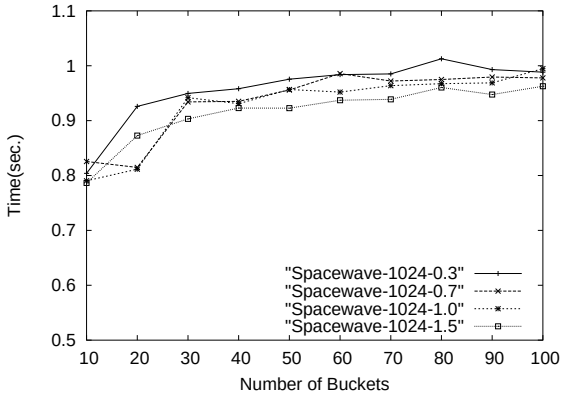
Real life data sets: We present the running times for the real life data sets in Figure 12. We present the result as B varies for CovType-A and as the size of the prefix n varies for the DJIA data. For the DJIA data we also show the time to compute the spectrum, i.e., the error for all $0 \leq B \leq n$. The spectrum of the datasets are shown in Figure 13.

7.5 Summary: Wavelets

- The Wavelet algorithms were far more insensitive to skew, Figures 10, compared to Histograms, Figure 6. Once n, B were fixed, the algorithm has little variation (except the binary search) that depends on the data. There are no obvious pruning



(a) Both WaveOpt and SpaceWave



(b) SpaceWave only

Figure 10: Running times as skew varies, $n = 1024$

strategies which we believe remains an open problem in this area, i.e., can the wavelet algorithms be made more data driven?.

- All the algorithms are (approximately) quadratic in n as can be seen from the slopes of the lines in the log-log plot in Figure 11 and 12(b).

- Recall that the running time of Waveopt is $O(n^2 B \log B)$. Figures 10, 12(a) & (b) verify that the running time is almost linear in B.

- SpaceWave is the clear winner and its running time is almost independent of B as the analysis suggested. It took less less time to compute the spectrum than to evaluate the coefficients for a fixed B since there were no recursion/recomputation involved.

8 Summary

In this paper we took a fresh look at DP techniques for synopsis construction problems. We provided an algorithmic framework using recursion, divide and conquer and DP to give $O(n)$ space algorithms for Wavelet and

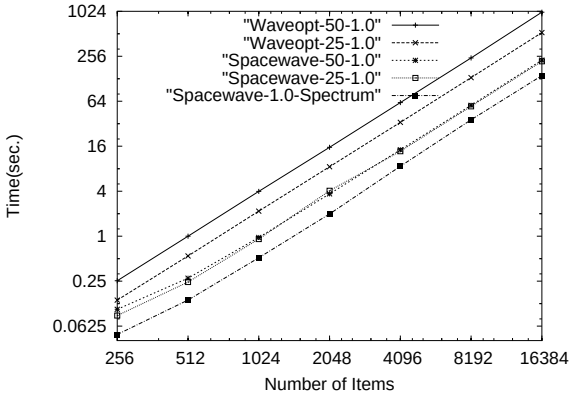


Figure 11: Running times as function of n, $B = 25, 50$

Histogram synopsis construction problems. We improved the state-of-the-art in many problems simultaneously and eliminated the use of working space notion in those problems. We indicated how the ideas affect other synopsis construction problems based on DP.

Acknowledgments

We thank Kyuseok Shim and Hyoungmin Park for discussions and providing the datasets. We also thank Sampath Kannan, Rajeev Motwani, S. Muthukrishnan for comments at various stages of the work. The comments from the anonymous referees were very helpful towards improving the presentation.

References

[1] S. Acharya, P. Gibbons, V. Poosala, and S. Ramaswamy. The Aqua Approximate Query Answering System. *SIGMOD Conference*, 1999.

[2] L. Amsaleg, P. Bonnet, M. J. Franklin, A. Tomasic, and T. Urhan. Improving responsiveness for wide-area data access. *IEEE Data Eng.*, 20(3):3–11, 1997.

[3] L. Arge. External memory data structures. *Proc. of ESA*, pages 1–29, 2001.

[4] K. Chakrabarti, M. N. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. *VLDB Conference*, 2000.

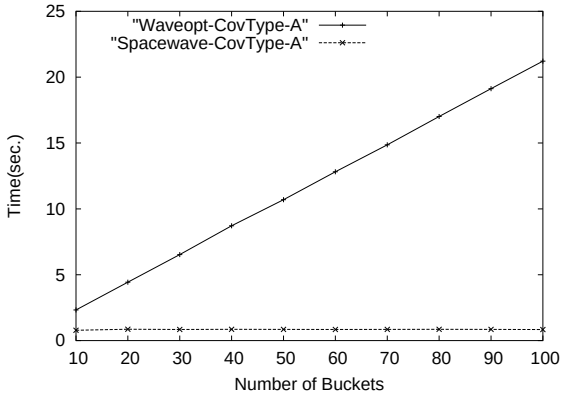
[5] K. Chakrabarti, E. J. Keogh, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM TODS*, 27(2):188–228, 2002.

[6] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, 1992.

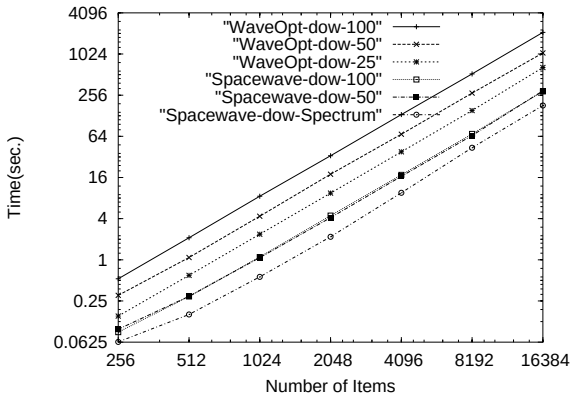
[7] A. Deligiannakis and N. Roussopoulos. Extended wavelets for multiple measures. *SIGMOD Conference*, 2003.

[8] M. Garofalakis and A. Kumar. Deterministic wavelet thresholding for maximum error metric. *Proc. of PODS*, 2004.

[9] M. N. Garofalakis and P. B. Gibbons. Probabilistic wavelet synopses. *ACM TODS (See also SIGMOD 2002)*, 29:43–90, 2004.



(a) CovType-A (Aspect), $n = 512$



(b) DJIA as prefix size varies

Figure 12: Wavelets on real-life datasets

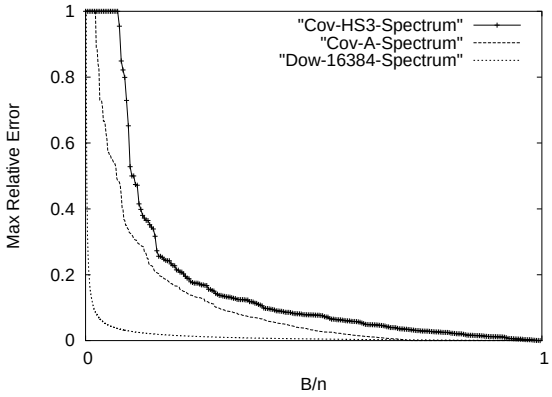


Figure 13: Spectrum of the real life datasets

[10] P. Gibbons and Y. Matias. Synopsis data structures for massive data sets. *Proc. of SODA*, 1999.

[11] P. Gibbons, Y. Matias, and V. Poosala. Fast Incremental Maintenance of Approximate Histograms. *VLDB Conference*, 1997.

[12] A. Gilbert, Y. Kotadis, S. Muthukrishnan, and M. Strauss. Surfing Wavelets on Streams: One Pass Summaries for Approximate Aggregate Queries. *VLDB Conference*, 2001.

[13] S. Guha. Space efficiency in optimal, approximation and streaming algorithms for synopsis construction problems. *Manuscript (email for copy).*, 2005.

[14] S. Guha and B. Harb. Wavelet synopsis for data streams: Minimizing non-euclidean error. *Proc. of SIGKDD Conference*, 2005.

[15] S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Histogramming data streams with fast per-item processing. *Proc. of ICALP*, 2002.

[16] S. Guha, N. Koudas, and K. Shim. Data Streams and Histograms. *Proc. of STOC, (see also the full version of [16], available at http://www.cis.upenn.edu/∼sudipto/mypapers/histjour.pdf.gz)*, 2001.

[17] S. Guha, K. Shim, and J. Woo. REHIST: Relative error histogram construction algorithms. *VLDB Conference*, 2004.

[18] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. *SIGMOD Conference*, 1997.

[19] Y. E. Ioannidis. Universality of serial histograms. *VLDB Conference*, 1993.

[20] Y. E. Ioannidis and V. Poosala. Balancing Histogram Optimality and Practicality for Query Result Size Estimation. *ACM SIGMOD Conference*, 1995.

[21] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel. Optimal Histograms with Quality Guarantees. *VLDB Conference*, 1998.

[22] P. Karras and N. Mamoulis. One pass wavelet synopis for maximum error metrics. *This proceeding*, 2005.

[23] Y. Matias and D. Urieli. Manuscript. 2004.

[24] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-Based Histograms for Selectivity Estimation. *SIGMOD Conference*, 1998.

[25] M. Muralikrishna and D. J. DeWitt. Equi-depth histograms for estimating selectivity factors for multidimensional queries. *SIGMOD Conference*, 1998.

[26] S. Muthukrishnan. Workload optimal wavelet synopsis. *DIMACS TR*, 2004.

[27] S. Muthukrishnan and M. Strauss. Approximate histogram and wavelet summaries of streaming data. *DIMACS TR 52*, 2003.

[28] V. Poosala, Y. E. Ioannidis, P. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. *SIGMOD Conference*, 1996.

[29] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. *SIGMOD Conference*, 1979.

[30] J. S. Vitter. External memory algorithms and data structures: dealing with massive data. *ACM Comput. Surv.*, 33(2):209–271, 2001.