Algorithmica (2013) 65:1–42
DOI 10.1007/s00453-011-9571-9

Computing Unrestricted Synopses Under Maximum Error Bound

Chaoyi Pang · Qing Zhang · Xiaofang Zhou · David Hansen · Sen Wang · Anthony Maeder

Received: 6 April 2009 / Accepted: 9 September 2011 / Published online: 5 October 2011

Abstract Constructing Haar wavelet synopses with guaranteed maximum error on data approximations has many real world applications. In this paper, we take a novel approach towards constructing unrestricted Haar wavelet synopses under maximum error metrics (L_∞). We first provide two linear time ($\log N$)-approximation algorithms which have space complexities of $O(\log N)$ and $O(N)$ respectively. These two algorithms have the advantage of being both simple in structure and naturally adaptable for stream data processing. Unlike traditional approaches for synopses construction that rely heavily on examining wavelet coefficients and their summations, the proposed methods are very compact and scalable, and sympathetic for online data

Part of the results in this paper appeared in Proceedings of the 12th International Conference on Extending Database Technology (EDBT) [14].

C. Pang (✉) · Q. Zhang · D. Hansen
The Australian e-Health Research Centre, ICT Centre, CSIRO, Brisbane, QLD, Australia
e-mail: chaoyi.pang@csiro.au

Q. Zhang
e-mail: qing.zhang@csiro.au

D. Hansen
e-mail: david.hansen@csiro.au

X. Zhou · S. Wang
School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, QLD 4072, Australia

X. Zhou
e-mail: zxf@itee.uq.edu.au

S. Wang
e-mail: sen.wang@uq.edu.au

A. Maeder
School of Computing and Mathematics, University of Western Sydney, Sydney, Australia
e-mail: a.maeder@uws.edu.au

processing. We then demonstrate that this technique can be extended to other findings such as Haar$^+$ tree. Extensive experiments indicate that these techniques are highly practical. The proposed algorithms achieve a very attractive tradeoff between efficiency and effectiveness, surpassing contemporary $(\log N)$-approximation algorithms in compressing qualities.

Keywords Approximation algorithm · Approximate query processing · Data compression · Data synopses · Haar wavelets

1 Introduction

Widely used in signal and image processing, the wavelet technique has been considered very promising for data compression and query approximation in database domains [11, 18]. Recent research has applied wavelet synopses to summarize data streams for approximate queries [4, 7]. The basic idea of constructing a wavelet synopsis of a data vector with size N, is to first transform the data vector into a representation with respect to a wavelet basis. The data vector is then approximated by retaining M coefficients as the wavelet synopsis and setting those remaining to 0 implicitly.

A conventional approach is to find M coefficients to minimize the overall mean squared error (L_2) [16]. This can be solved gracefully by applying Parseval's theorem [2]. However, the main drawback of this approach is that users cannot control the approximation error of individual elements in the data vector. This severely impedes further applications of the wavelet approximation. To alleviate this, researchers have made efforts to construct wavelet synopses with guaranteed maximum (absolute or relative) error in approximation [2]. Two approaches for constructing wavelet synopses on maximum error metric L_∞ have been taken: one is to construct bucket bound (size bound) synopses which would minimize the maximum approximation error of single data elements [2] whilst the other is to construct the smallest size of synopses such that the maximum approximation error does not exceed a given error bound [12, 13]. The details of each are explained below.

Bucket Bound Synopses The goal is to construct a synopsis of at most B Haar wavelet coefficients to minimize the maximum error L_∞.

The bucket bound (size-bound) synopses have been studied intensively [2, 3, 6, 7, 9]. Garofalakis et al. [2] show that the synopses can be constructed with $O(N^2)$ time/space. Several methods such as [3, 4, 7] have been proposed to improve the performance. Guha et al. [4, 5] indicated that the restriction to Haar wavelet coefficients for synopses is not a good strategy and extend to the construction of synopses not restricted to Haar wavelet coefficients. By using rounding techniques, basis vectors and resolution parameter δ on error bound Δ, they provide $(1+\delta)$-approximation algorithms to the error of the best possible unrestricted bucket bound synopses (while respecting the size B). These algorithms have sub-linear space complexity of $O(\delta^{-1}B^2\log^3 N)$ and time complexities of $O((\frac{\Delta}{\delta})^2 N \log N \log^2 B)$ and

$O(N\delta^{-2}\log^3 N\log^2 B)$. They also show $(\log N)$-approximation version of the algorithm runs in $O(\log N + B)$ space and works for all wavelets of compact support. Karras et al. [9] showed that bucket-bounded synopsis problems for maximum-error metrics can be more efficiently solved via their error-bounded counterparts and improve the result of [5] close to $O((\frac{\Delta}{\delta})^2 N\log N)$. Nevertheless, many real applications suggest the construction of error bound (Δ-bound) synopses as stated below.

Restricted Error-Bound Synopses Given an error bound Δ, the goal is to find M_{op}, a Haar wavelet synopsis with the smallest set of coefficients among all possible solutions, that would satisfy the Δ bound on the maximum error (i.e., $L_\infty < \Delta$) in approximation. M_{op} is called *R-optimal* as each element in the synopsis is restricted to be a coefficient.

The error bound synopses have been studied by Muthukrishnan and Guha [3, 12]. Their approaches for this problem are basically similar to those for bucket bound synopses. The construction of the optimal synopsis M_{op} has $O(N^2)$ time complexity. Furthermore, the construction of "unrestricted" error-bound synopses are yet to be fully studied, despite being quite beneficial in practice. Matias et al. [10] indicate that an optimal synopsis is a restricted synopsis (under an orthonormal basis) for overall mean squared error (L_2). However, this is not the case for maximum error (L_∞): a better compression results can often be achieved by using unrestricted synopses for maximum error.

Unrestricted Error-Bound Synopses Given error bound Δ, the goal is to find S_{op}, a synopsis with the smallest set of unrestricted coefficients among all possible solutions, that would satisfy the Δ bound on the maximum (absolute/relative) error. S_{op} is called *U-optimal* as each element in the synopsis is not restricted to be a coefficient.

Fortunately, extended from the idea of [4, 5] on unrestricted bucket bound synopses, Karras et al. proposed an efficient algorithm[1] for the U-optimal problem under two parameters: δ and Δ. This result (K algorithm) uses the dynamic-programming framework and has $O((\frac{\Delta}{\delta})^2 N)$ time complexity. Theoretically, their algorithm can obtain an U-optimal solution through (iteratively) using smaller δ values. The execution time of their algorithm greatly depends on the value of δ and Δ: It derives a smaller sized synopsis under a smaller δ value but requires a greater length of time.[2] Refer to Sect. 8 for detailed comparison tests. Proceeding the work of [9], Karras et al. [8] introduced the Haar$^+$ tree: a novel concept that merges a classical Haar tree with a compact hierarchical histogram [15] to simplify the construction process of synopses. Even through some properties in classical Haar tree may not properly be held in Haar$^+$ tree due to the supplementary nodes, the use of Haar$^+$ structure generally results higher compression qualities [8].

When compressing a high frequency time-series data such as the physiological data from surgery operations, users would not be concerned if the synopsis is re-

[1] The MinHaarSpace algorithm of [9]. It is termed as K algorithm in this paper.

[2] As stated in [9]: Smaller (δ) values burdened the running time without significant quality increase; larger (δ) values were undermining the quality of the synopses.

stricted to be wavelet coefficients. They may trade off the sizes of synopses for time efficiency. Any methods with high time cost may not be applicable effectively. They are more interested in the methods that can construct smaller-sized synopses efficiently. Examples of physiological data include Electrocardiogram (ECG), Arterial Blood Pressure (ABP), Central Venous Pressure (CVP) and respiration etc.

Given a data vector and an error bound Δ, the objective of this paper is to provide efficient algorithms for constructing an unrestricted synopsis S such that the maximum approximation error of each single data element is bounded by Δ and S is quite close to S_{op} in size.

Contributions We first propose two linear algorithms to construct the unrestricted error bound synopses: Fixed-value Shift (F-Shift) algorithm and Sliding-value Shift (S-Shift) algorithm. F-Shift is an approximation algorithm for the unrestricted absolute error-bound problem while S-Shift can be viewed as an optimal version of the former. That is, S-Shift always generates smaller (or equal) sized synopses than that of F-Shift. We then indicate that this technique can be extended to other findings. We especially present the S^+-Shift algorithm following the state-of-the-art Haar$^+$ structure [8]. Compared with conventional approaches, our contributions in this paper can be summarized as follows:

(1) We present novel approaches in the construction of unrestricted synopses under the maximum absolute error bound. The traditional methods usually work on the decomposed data (coefficients) and then find the retained coefficients either by checking the combinatorial path summations [3, 4, 9, 12] or by retaining the most "significant" coefficients [7]. Instead, we construct synopses by utilizing the shift transformation upon "data scope", which will be explained later in detail.

(2) We provide the first practicable approximation algorithms for construction of unrestricted error bound synopses. Let N be the size of the data vector. Our two algorithms, F-Shift and S-Shift, both have linear time complexity. F-Shift has $O(\log N)$ space complexity and S-Shift has $O(N)$ space complexity. They are $(\log N)$-approximation algorithms. Let S_F and S_S represent the synopses constructed by our F-Shift and S-Shift algorithms respectively. Then $|S_S| \leq |S_F| \leq |S_{op}| \log N$ holds. However, this property does not hold for R-optimal synopsis M_{op}. We indicate that the size of the restricted synopsis M_{op} can be very large even if the size of S_{op} is very small (Example 4.6). This result suggests that unrestricted synopses such as S_F or S_S can be preferable to restricted synopses in real applications. That is, in the approximation under maximum error bound (L_∞), a better compression results can often be achieved by using unrestricted synopses.

(3) The Shift algorithms are directed by and based on the two novel presentations and one novel concept along with their relevant properties proposed in the paper: shift transformation, shift range and data scope. These properties lead to the correctness of the Shift algorithms and the above-listed features. Shift transformations provide graphic explanations and meanings for the unrestricted synopses and their formations that are not available in the earlier work on unrestricted synopses. The idea of Data scope is very useful and makes the reasoning and the proofs easy to follow. We show the same methodology can be extended to

other findings such as the construction of synopses on maximum relative error, synopses on a given bucket size upon multidimensional streams [19], and the adaptation on Haar$^+$ structure. Especially, extended from S-Shift, we provide a new algorithm, termed as S$^+$-Shift, for the Haar$^+$ structure.

(4) The efficiencies and effectiveness of Shift algorithms have been demonstrated through extensive experiments on both synthetic data and real life data. In terms of effectiveness, the Shift algorithms generate error-bound synopses with good compression quality. Although the approximation ratio on synopses size is $\log N$ in theory, the practical tests indicate that $|S_S|$ is often quite close to the size of optimal synopses. In terms of efficiencies, the Shift algorithms dramatically improve the synopses construction time against the MinHaarSpace algorithm [9] under a same compression quality goal. Even over large data sets such as those with 2^{18} values, to construct the same compression quality synopses, the Shift algorithms take less than 1 second while the MinHaarSpace algorithm can take up to 10^7 seconds. Refer to Sect. 8.2 for details. We choose the state-of-the-art research results by Guha et al. [5] for the indirect comparisons with the Shift algorithms on synopsis quality under infinity-norm. Similar to the comparison results on multidimensional streams as demonstrated in [19], the approximation qualities of the Guha's algorithm in [5] on one-dimensional data are lower than that of Shift algorithms in all tested situations.

Similarly, for the Haar$^+$ structure, the comparison results on the S$^+$-Shift and the Haar$^+$ algorithm of [8] indicate that the S$^+$-Shift can be 10^4 times faster than the Haar$^+$ algorithm for producing similar sized synopses.

More importantly, due to $O(\log N)$ space complexity, the F-Shift algorithm can be used as an online compression algorithm for stream data: the algorithm processes incoming data progressively and does not require the Haar wavelet error tree to be pre-computed. In the process of compressing stream data, algorithms with linear (or above) space complexity cannot be used directly and require a mechanism to segment incoming data into sections (e.g., fixed windows). In contrast, unlike those existing compression methods, including that on L_2 measure, the F-Shift algorithm can compress stream data directly in a synchronized way and can be more efficient in reality as it does not need to wait for the incoming data to reach a certain volume. To the best of our knowledge, the F-Shift algorithm is the first compression algorithm with this property.[3]

The rest of the paper is organized as follows. Section 2 explains the basic terminologies. Section 3 describes the shift transformation and defines the data scope and the shift range concepts in conjunction with their related properties. Section 4 introduces the F-Shift algorithm and its properties. Section 5 is about the S-Shift algorithm. Section 6 is the extension to relative error. Section 7 is the extensions to other applications. Section 8 reports our experiment results. Section 9 concludes this paper.

[3]To compute the optimal size-bound or error-bound synopses under L_2, the conventional approaches require retaining some top most "significant" *normalized* coefficients on a data vector of N-size. Since the number of N needs to be prefixed for generating synopses, those approaches are based on the fixed-window mechanism and may not be regarded as progressive strictly.

Table 1 Notations

Symbol ($i \in \{0..N-1\}$)	Description
$D, [d_0; ..; d_{N-1}]$	data vector
W_D	Haar wavelet transformation on D
T	error tree
c_i, s_i	coefficient node, shift coefficient node
$d_i, \hat{d}_i$	leaf/data node and its reconstruction
$path(u)$	all ancestors of node u in T
$T(c)$	subtree rooted at c
$T_L(c)/T_R(c)$	left/right subtree of $T(c)$
Δ	error bound on approximation (>0)
S	set of shift coefficients
$T(c) \sqcap S$	the subset of S that locates in $T(c)$
S_F/S_S	synopsis obtained by F-Shift/S-Shift
M	set of Haar wavelet coefficients
M_{op}/S_{op}	restricted/unrestricted optimal synopsis
$T\langle S\rangle$	error tree after S shift transform on T

2 Haar Wavelet and Data Approximation

Table 1 summarizes the notations used throughout this paper.

Wavelets are a mathematical tool for hierarchically decomposing functions. Generally, the wavelet decomposition of a function consists of a coarse overall approximation together with detail coefficients that influence the function at various scales [16]. It can generally be computed in linear time.

The Haar wavelet is the simplest wavelet. The Haar wavelet decomposition of a data vector consists of a coefficient representing the overall average of the data values followed by detail coefficients in the order of increasing resolution. Each detail coefficient is the difference of a pair of averaged values from the lower level.

Suppose we are given data vector $D = [19; 17; 12; -4; 7; -1; -3; -7]$ containing $N = 8$ data values. The Haar wavelet decomposition of D can be computed as follows. We first average the data values, pairwise, to get a new lower-resolution data with values $[18; 4; 3; -5]$. That is, the first two values in the original data (19 and 17) average to 18, and the second two values 12 and -4 average to 4, and so on. We also store the pairwise differences of the original values (divided by 2) as detail coefficients: $[\frac{19-17}{2}; \frac{12+4}{2}; \frac{7+1}{2}; \frac{-3+7}{2}]$, which is $[1; 8; 4; 2]$. It is easy to see that the original values can be recovered from the averages and differences. By repeating this process recursively on the averages, we get the Haar wavelet decomposition in Fig. 1(i). We define the wavelet decomposition W_D of data vector D to be the coefficient representing the overall average of the original data, followed by the detail coefficients in the order of increasing resolution. That is, $W_D = [5; 6; 7; 4; 1; 8; 4; 2]$.

The Haar wavelet decomposition can be expressed in an error tree structure [11]. Depicted in Fig. 1(ii), error tree T is a hierarchical structure representing the nodes of

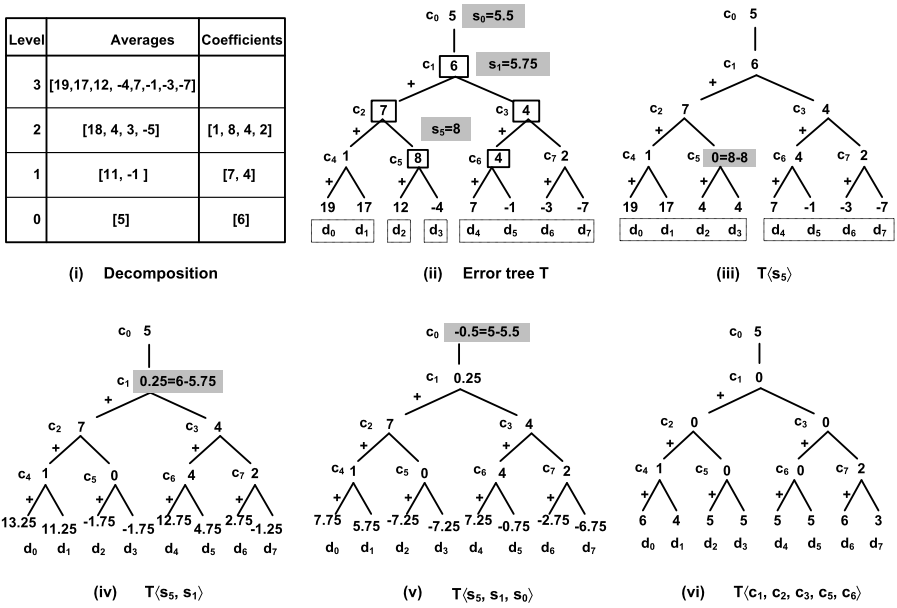


Fig. 1 Example ($\Delta = 8$): $M_{op} = \{c_1, c_2, c_3, c_5, c_6\}$ and $S_{op} = \{s_0, s_1, s_5\}$

W_D and D, where each internal node c_i represents a wavelet coefficient and each leaf node d_i represents an original data item. We use $T(u)$ to denote the subtree rooted at u, $T_L(u)$ and $T_R(u)$ as the subtrees rooted at the left and right child of u respectively.

Given a node u (internal or leaf), we define $path(u)$ as the set of nodes that lie on the path from the root node to u (excluding u). To reconstruct any leaf node d_i through the error tree T, we only need to compute the signed summation of nodes belonging to $path(d_i)$. That is,

$$d_i = \sum_{c_j \in path(d_i)} \delta_{ij} c_j, \tag{2.1}$$

where $\delta_{ij} = +1$ if $d_i \in T_L(c_j)$ and $\delta_{ij} = -1$ if $d_i \in T_R(c_j)$. From this property, we have the following formal definition of an error tree.

Definition 2.1 A binary tree T is an error tree iff each leaf node d_i equals the signed summation of all the internal nodes of $path(d_i)$. That is, each d_i satisfies (2.1).

Being reconstructed from W_D, data vector D can also be approximated with a subset of W_D. Let $\hat{d}_i$ denote the approximated data value of d_i on $M \subseteq W_D$. That is,

$$\hat{d}_i = \sum_{c_j \in path(d_i) \cap M} \delta_{ij} c_j.$$

Rather than obtaining the minimum set $M \subseteq W_D$ that satisfies

$$\max_{0 \le i < N} |\hat{d}_i - d_i| < \Delta$$

for a given error bound $\Delta > 0$, in this paper, we will investigate how to find a close-to-minimum-sized S such that $\max_{0 \le i < N} |\hat{d}_i^{(S)} - d_i| < \Delta$ where

$$\hat{d}_i^{(S)} = \sum_{c_j \in path(d_i) \wedge s_j \in S} \delta_{ij} s_j. \tag{2.2}$$

Generally, S is a set of s_i obtained from a subset of W_D by representing each c_j in the subset with s_j. Equation (2.2) is the signed summation of S nodes on $path(d_i)$. Detailed discussion on S and its graphic meaning will be given in the next section.

Example 2.2 In error tree T of Fig. 1(ii), $T(c_5) = T_R(c_2)$ and $T(c_4) = T_L(c_2)$. Data value d_2 can be reconstructed through the nodes of $path(d_2)$, i.e., $d_2 = 5+6+(-7)+8$. Let $M = \{c_1, c_2, c_3, c_5, c_6\}$ and $S = \{s_0, s_1, s_5\}$ where[4] $s_0 = 5.5$, $s_1 = 5.75$ and $s_5 = 8$. Then $\hat{d}_2 = 6 + (-7) + 8$, which is 7, and $\hat{d}_2^{(S)} = 5.5 + 5.75 + 8$, which is 19.25.

3 Concepts and Properties

In this section, three major concepts of this paper: *shift transformations*, *data scope* and *shift range* are defined. Some properties for each concept are given. We will firstly show how to use shift transformations to generate synopses. We then indicate that data scopes and shift ranges can be used to select shift coefficients and for assigning shift values. With these results, the general idea of Shift algorithms is given in Sect. 3.4.

3.1 Shift Transformation

Shift transformations supply graphic explanations and meanings for the unrestricted synopses and their formations. In this subsection, we first define shift transformations. We then show that a synopsis is a set of shift values generated from shift transformations that transform the error tree to "Δ-bound" (Property 3.3).

Let c_i be a coefficient of error tree T and s_i be a real value. An *s_i-shift transformation* at c_i of T transforms T into $T' = T\langle s_i \rangle$ through the following steps:

1. Replace coefficient c_i with $c_i - s_i$;
2. When $i > 0$,
   (a) replace each leaf node d of $T_L(c_i)$ with $d - s_i$;
   (b) replace each leaf node d of $T_R(c_i)$ with $d + s_i$.
3. When $i = 0$, replace each leaf node d of T with $d - s_0$.

s_i is called the *shift coefficient* (at c_i). In general, suppose that S is a set of shift coefficients on error tree T. $T\langle S \rangle$ is the tree transformed from T through applying the above steps on each $s_i \in S$ iteratively. Clearly, $T\langle S \rangle$ is well defined and irrelevant to the transformation order of the elements in S.

[4] We will explain why we chose these particular values for s_i in Sect. 3.3.

From the error tree definition, it can be proven that $T\langle S\rangle$ is the error tree on the updated data vector.

Property 3.1 *Let T be an error tree and S be a set of shift coefficients on T. Then $T\langle S\rangle$ is also an error tree.*

The above property assures that the Shift algorithms introduced in the paper work on an error tree at each iteration.

Example 3.2 (Continued from Example 2.2) Error trees of $T\langle\{s_5\}\rangle$, $T\langle\{s_1, s_5\}\rangle$ and $T\langle\{s_0, s_1, s_5\}\rangle$ are expressed from Figs. 1(iii)–(v) respectively.

A set of shift coefficients S is called a Δ-bound unrestricted synopsis (or simply *synopsis*) if $|diff_S(d_i)| < \Delta$ holds for each d_i of T where $diff_S(d_i) = d_i - \hat{d}_i^{(S)}$.

Generally, a shift coefficient s_i of S may not have the same value as c_i. A *restricted synopsis* is a synopsis where each $s_i = c_i$ in S. In this paper, a restricted synopsis will be denoted by M. An *optimal* (restricted/unrestricted) synopsis is a synopsis that has the smallest size among all possible synopses. By M_{op} (and S_{op}) we denote the optimal restricted (and unrestricted) synopsis respectively. Clearly, $|M_{op}| \leq |M|$ and $|S_{op}| \leq |S|$ hold.

Error tree T is Δ-*bound* if $|d_i| < \Delta$ holds for each d_i of T. The following results can be directly proven from the definition of synopses. It assures the correctness of Shift algorithms and indicates that a synopsis can be derived from shift transformations.

Property 3.3 *Let T be an error tree.*

1. *$T\langle S_{op}\rangle$ is Δ-bound.*
2. *Empty set $\emptyset$ is a synopsis iff T is Δ-bound.*
3. *S is a synopsis of T iff $T\langle S\rangle$ is Δ-bound.*
4. *If T is Δ-bound, then $|c_i| < \Delta$ holds for each $c_i \in T$.*

Property 3.3(4) can be proven by borrowing Property 1 of [13]. Roughly, if each $|d_i| < \Delta$, then the averages of any data values are less than Δ. Thus the averages of their differences are less than Δ. That is, $|c_i| < \Delta$ for each $c_i \in T$.

Example 3.4 (Continued from Example 3.2) Let $\Delta = 8$. It can be verified that $M_{op} = \{c_1, c_2, c_3, c_5, c_6\}$ and $S_{op} = \{s_0, s_1, s_5\}$ hold. $M = \{c_0, c_1, c_2, c_3, c_5, c_7\}$ is a restricted synopsis. Figure 1(v) is $T\langle S_{op}\rangle$ and Fig. 1(vi) is $T\langle M_{op}\rangle$. Clearly, $|d_i| < \Delta$ holds for each $d_i \in T\langle S_{op}\rangle$ (or $d_i \in T\langle M_{op}\rangle$).

3.2 Data Scope

Property 3.3 shows that a synopsis can be obtained through shift transformations. One of the remaining questions is which coefficient needs to be shifted or which s_i should be in a synopsis? In order to answer this question, we propose the *data scope* concept.

Definition 3.5 Let T' be a subtree of error tree T and S be a set of shift coefficients. The (data) scope of T' on S is defined as

$$scope\big(T', S\big) = \frac{1}{2} \max_{d_i, d_j \in T'} \big|diff_S(d_i) - diff_S(d_j)\big|.$$

Intuitively, $scope(T', S)$ describes the data extents of $T'\langle S\rangle$ and measures

$$\max_{d'_i, d'_j \in T'\langle S\rangle} \big|d'_i - d'_j\big|.$$

With this concept, we will explain when a coefficient of a subtree needs to be shifted.

Let S be a set of shift coefficients. For simple expressions, we denote $\{s_i | c_i \in T(c) \wedge s_i \in S\}$, the subset of S that locates in $T(c)$, by $T(c) \sqcap S$. The following property suggests that the exhaustive combinations on $path(d_i)$ as in [3, 12] can be alleviated by checking data scopes.

Property 3.6 *Let S be a set of shift coefficients. Then, for any coefficient c of T,*

(i) $scope(T(c), M_{op}) < \Delta$ *and* $scope(T(c), S_{op}) < \Delta$ *hold*;
(ii) $scope(T(c), S) = scope(T(c), T(c) \sqcap S)$;
(iii) *if* $scope(T(c), \emptyset) \geq \Delta$, *then* $T(c) \cap M_{op} \neq \emptyset$;
(iv) *if* $scope(T(c), \emptyset) \geq \Delta$, *then* $T(c) \sqcap S' \neq \emptyset$ *holds for any synopsis* S'.

Proof The proof of (i) can be obtained from the definitions of M_{op}, S_{op} and data scope directly.

The proof of (ii): Let d_i and d_j be any two data nodes in $T(c)$. $\hat{d}_i^{(S)}$ and $\hat{d}_j^{(S)}$, the reconstructed d_i and d_j on S, are built from the summation of two parts: (a) $\hat{d}_{i1}^{(S)}$ and $\hat{d}_{j1}^{(S)}$ from the nodes of $(T - T(c)) \sqcap S$ where $\hat{d}_{i1}^{(S)} = \hat{d}_{j1}^{(S)}$ and, (b) $\hat{d}_{i2}^{(S)}$ and $\hat{d}_{j2}^{(S)}$; where $\hat{d}_{i2}^{(S)}$ and $\hat{d}_{j2}^{(S)}$ from the nodes of $T(c) \sqcap S$. Since $\hat{d}_h^{(S)} = \hat{d}_{h1}^{(S)} + \hat{d}_{h2}^{(S)}$ and $diff_S(d_h) = d_h - \hat{d}_h^{(S)}$ for $h = i, j$,

$$\begin{aligned} diff_S(d_i) - diff_S(d_j) &= \big(d_i - \hat{d}_i^{(S)}\big) - \big(d_j - \hat{d}_j^{(S)}\big) \\ &= \big(d_i - \hat{d}_{i1}^{(S)}\big) - \big(d_j - \hat{d}_{j1}^{(S)}\big). \end{aligned}$$

That is, $|diff_S(d_i) - diff_S(d_j)| = |diff_{T(c)\sqcap S}(d_i) - diff_{T(c)\sqcap S}(d_j)|$. As d_i and d_j are arbitrary data of $T(c)$, (ii) is proven from the definition of data scope.

As the proofs of (iii) and (iv) are similar, we will only prove (iii) in the following.

Otherwise, assume that $T(c) \cap M_{op} = \emptyset$ holds. From (ii), we have $scope(T(c), M_{op}) = scope(T(c), \emptyset)$. Since $scope(T(c), \emptyset) \geq \Delta$, $scope(T(c), M_{op}) \geq \Delta$ holds. Therefore, (iii) is proven, as it is contradictory to (i). □

We will call subtree $T(c)$ *scope* Δ *bound* (or simply, *s-bound*) if $scope(T(c), \emptyset) < \Delta$, otherwise, it is *s-unbound*. $T(c)$ is called $\overline{s}$-bound (i.e., maximal s-bound) if it is s-bound and $T(c')$ is s-unbound where c' is the parent node of c in error tree T. Similarly, denote $T(c)$ to be $\underline{s}$-unbound if $T(c)$ is s-unbound but

each of its subtrees is s-bound. Clearly, if $T(c)$ is Δ-bound then it is s-bound. These concepts will be used in the Shift algorithms and for the size estimation on obtained synopses.

Property 3.6(ii) says that $scope(T(c), S)$ is unchangeable by shifting any nodes of $T - T(c)$. That is, shifting the coefficients that locate outside of $T(c)$ will not change the value of $scope(T(c), S)$. Property 3.6(iii) and (iv) imply that, in order to obtain a synopsis for the given Δ, some nodes inside $T(c)$ must be shifted whenever $T(c)$ is $\underline{s}$-unbound. This is the basic property for the Shift algorithms.

Example 3.7 (Continued from Example 3.4) For error tree T of Fig. 1(ii), $T(c_4)$, $T_L(c_5)$, $T_R(c_5)$ and $T(c_3)$ are $\overline{s}$-bound. $T(c_4)$ is also Δ-bound. The number of $\overline{s}$-bound subtrees of T is 4. Alternatively, $T(c_5)$ is $\underline{s}$-unbound. $T(c_2)$ is s-unbound as $scope(T(c_2), \emptyset) = (19 + 4)/2$ is greater than Δ. According to Property 3.6, at least one coefficient of $T(c_2)$ and $T(c_5)$ need to be shifted for a synopsis.

3.3 Shift Range

To obtain a synopsis, Property 3.6 indicates that a shift coefficient in a $\underline{s}$-unbound subtree should be retained or shifted. It is still not clear which specific coefficient should be chosen and in what value, considering that the shift coefficient s_i will usually not have the same value as coefficient c_i. These questions will be answered in this subsection. Explicitly, we will show how to convert an $\underline{s}$-unbound subtree into an s-bound subtree by applying a shift transformation on its root node. We then give the definition of the shift range. Generally, the shift range on coefficient c_i is a range of values for s_i to choose from.

In the following, Property 3.8 actually points out a method of converting an arbitrary error tree into a Δ-bound tree through shift transformations. Significantly, it shows that only node c_i where $T(c_i)$ is $\underline{s}$-unbound and/or node c_0 need to be shifted in the process of generating a synopsis from bottom-up.

Property 3.8 *Let $T' = T(c_i)$ be a subtree of error tree T rooted at node c_i ($i > 0$) and $\Delta > 0$ be a given error bound.*

(i) *If T' is $\underline{s}$-unbound for Δ, then there exists s_i such that $T'\langle s_i\rangle$ is s-bound.*
(ii) *If error tree T is s-bound but not Δ-bound, then there exists s_0 such that $T\langle s_0\rangle$ is Δ-bound.*

Proof The proof of (i). Let d_{L_g} and d_{L_s} (or d_{R_g} and d_{R_s}) be the largest and smallest data values of $T_L(c_i)$ (or $T_R(c_i)$) respectively. Since T' is $\underline{s}$-unbound, the following inequalities hold:

$$d_{L_g} - d_{L_s} < 2\Delta, \qquad d_{R_g} - d_{R_s} < 2\Delta, \tag{3.1}$$

$$\max\{d_{L_g}, d_{R_g}\} - \min\{d_{L_s}, d_{R_s}\} \geq 2\Delta. \tag{3.2}$$

In order to convert T' into an s-bound one, the shift coefficient s_i should satisfy:

$$\begin{cases} -2\Delta < d_{L_g} - s_i - (d_{R_s} + s_i) < 2\Delta, \\ -2\Delta < d_{R_g} + s_i - (d_{L_s} - s_i) < 2\Delta. \end{cases}$$

Since $d_{L_g} - d_{R_s} \geq -(d_{R_g} - d_{L_s})$, the above equation can be simplified into:

$$(d_{L_g} - d_{R_s})/2 - \Delta < s_i < -(d_{R_g} - d_{L_s})/2 + \Delta. \tag{3.3}$$

By (3.1), since

$$\begin{aligned}&\big[-(d_{R_g} - d_{L_s})/2 + \Delta\big] - \big[(d_{L_g} - d_{R_s})/2 - \Delta\big]\\ &= 2\Delta - \big[(d_{R_g} - d_{R_s}) + (d_{L_g} - d_{L_s})\big]/2\\ &> 0,\end{aligned}$$

the existence of s_i is proven.

The proof of (ii). Suppose that d_g and d_s are the largest and smallest data values of $T(c_0)$ respectively. Let $x_1 = (d_g + d_s)/2$ and $l_1 = (d_g - d_s)/2$. To convert T into a Δ-bound one, s_0 should satisfy

$$\begin{cases} -\Delta < x_1 - l_1 - s_0, \\ x_1 + l_1 - s_0 < \Delta, \end{cases}$$

or equivalently,

$$|x_1 - s_0| < \Delta - l_1. \tag{3.4}$$

From the assumption that T is s-bound, $\Delta - l_1 > 0$ holds. Therefore, (3.4) has solutions and $T\langle s_0\rangle$ is Δ-bound for each solution s_0 of (3.4). □

It should be noted that $T'\langle s_i\rangle$ is not guaranteed to be s-bound if we choose s_i to be c_i in Property 3.8(i). Similarly, in Property 3.8(ii), $T\langle s_0\rangle$ may not be guaranteed to be Δ-bound by setting s_0 to be c_0. That is, c_i (or c_0) may not satisfy (3.3) (or (3.4)).

The above proof is constructive. The proof also illustrates what values the shift coefficient needs to be chosen from. For instance, in the proof of Property 3.8(i), $T'\langle s_i\rangle$ is s-bound for any s_i that is in the range of (3.3).

Furthermore, a special situation of (3.3) is to minimize the scope of $T'\langle s_i\rangle$: in addition to being s-bound, shift the two intervals of $(d_{L_s} - s_i, d_{L_g} - s_i)$ and $(d_{R_s} + s_i, d_{R_g} + s_i)$ enclosed, i.e., one interval subsumes another. In this situation, it can be verified that s_i satisfies the following inequalities:

$$\begin{cases} \min\{(d_{L_g} - d_{R_g})/2,\ (d_{L_s} - d_{R_s})/2\} \leq s_i, \\ s_i \leq \max\{(d_{L_g} - d_{R_g})/2,\ (d_{L_s} - d_{R_s})/2\}. \end{cases} \tag{3.5}$$

Let $l_L = (d_{L_g} - d_{L_s})/2$, $l_R = (d_{R_g} - d_{R_s})/2$, $x_L = (d_{L_g} + d_{L_s})/2$ and $x_R = (d_{R_g} + d_{R_s})/2$. Inequalities (3.5) can be rewritten into

$$\big|(x_L - x_R) - 2s_i\big| \leq |l_L - l_R|. \tag{3.6}$$

The existence of s_i in (3.6) can be proven from (3.5). We will use (3.6) to select the values for s_i whenever $T(c_i)$ is $\underline{s}$-unbound rather than through (3.3). In Sect. 5.2, we will explain why we do not use (3.3) in our approach.

Definition 3.9 Let $T' = T(c_i)$ be a subtree of error tree T and $\Delta > 0$ be a given error bound. The shift range of c_i (i.e., $range(c_i)$) is defined as follows.

(a) if T' is $\underline{s}$-unbound, then $range(c_i) = [\underline{s_i}, \overline{s_i}]$ where

$$\underline{s_i} = \big[(x_L - x_R) - |l_L - l_R|\big]/2, \qquad \overline{s_i} = \big[(x_L - x_R) + |l_L - l_R|\big]/2,$$

derived from (3.6).

(b) if $T' = T(c_0)$ is s-bound but not Δ-bound, $range(c_0) = (\underline{s_0}, \overline{s_0})$ where

$$\underline{s_0} = x_1 - (\Delta - l_1), \qquad \overline{s_0} = x_1 + (\Delta - l_1),$$

derived from (3.4).

(c) if T' is s-bound, then $range(c_i) = [\underline{s_i}, \overline{s_i}]$ where $\underline{s_i} = \overline{s_i} = (x_L + x_R)/2$.

As we will explain in the latter sections, the S-Shift algorithm will employ "modified" shift ranges in the process of generating a synopsis. Clearly, $s_i = (x_L - x_R)/2$, which will be used in the F-Shift algorithm constantly, is a solution of (3.5).

Example 3.10 In Fig. 1(iii), $range(c_1) = [5.5, 6]$ from (3.6). That is, $T\langle\{s_1\}\rangle$ is s-bound for any $s_1 \in range(c_1)$. The data interval $[d_s, d_g]$ of $T(s_1)$ slides between $[-1.5, 13.5]$ and $[-2, 13]$ for $s_1 \in range(c_1)$ with the same data scope $l_2 = 7.5$.

In Fig. 1(iv), T is s-bound. $range(c_0) = (5.25, 6.25)$ by (3.4). $T\langle\{s_0\}\rangle$ is Δ-bound for any $s_0 \in range(c_0)$. Data interval (d_s, d_g) of $T\langle\{s_0\}\rangle$ slides between $(-7, 8)$ and $(-8, 7)$. Specifically, $T\langle\{s_0\}\rangle$ for $s_0 = 5.5$ is the error tree of Fig. 1(v).

3.4 Idea on Shift Algorithms

The properties of shift transformations, data scopes and shift ranges will be used to shape the Shift algorithms. They are used to decide which nodes of coefficients need to be retained in the synopsis and in what value. Generally, Shift algorithms work bottom-up from the data vector. It applies a shift transformation on the root node of a current $\underline{s}$-unbound subtree repeatedly. Each shift transformation converts the $\underline{s}$-unbound subtree into a s-bound subtree through properly chosen a shift value from the shift range. When the process comes to node c_0, the algorithms may shift node c_0 making the current $T(c_0)$ be Δ-bound. Eventually, the set of shift coefficients is the synopsis. For instance, let us consider the error tree T of Fig. 1(ii). The F-Shift algorithm first works on the $\underline{s}$-unbound subtree $T(c_5)$ and derives $T\langle\{s_5\}\rangle$ depicted in Fig. 1(iii). It then works on the next $\underline{s}$-unbound subtree, which is $T(c_1)$ in Fig. 1(iii), and derives $T\langle\{s_1, s_5\}\rangle$ depicted in Fig. 1(iv). Repeating these steps, the F-Shift algorithm derives $T\langle\{s_0, s_1, s_5\}\rangle$ ultimately.

The major difference between the F-Shift and the S-Shift is that they use different strategies on choosing shift values. Unlike the F-Shift where the shift values are set to be average values (i.e., $(x_L - x_R)/2$) from bottom up, the S-Shift algorithm generate synopsis S_S from two phases: the bottom-up phase and the top-down phase. In the bottom-up phase, it computes $range(c)$ from $range(c_L)$ on the left subtree and

range(c_R) on the right subtree in a "modified" way by generalized (3.6) and Definition 3.9. Its objective is to fuse the maximum overlapped data range on each subtree. In the top-down phase, it instantiates the values of selected shift coefficients within the ranges to compute synopsis S_S from top down. Details will be given in the next two sections.

4 F-Shift Algorithm

As depicted in Algorithm 1, the F-Shift (Fixed-value Shift) algorithm employs the data scope and shift technique to obtain a synopsis in bottom-up from the data vector. It does not need to construct W_D or T previously. This is because that a shift transformation on $c \in T$ actually only works on data of $T_L(c)$ and $T_R(c)$ in an opposite way and can be described by the data "intervals" of $T_L(c)$ and $T_R(c)$.

In this section, we first describe the F-Shift algorithm. We then study the complexity of F-Shift and discuss the upper bound on the synopses generated from it.

Input: D, the original data vector; Δ, the specified error bound
Output: A synopsis B. That is, the set of shift coefficients that satisfies Δ bound

$B \leftarrow \emptyset$; $F \leftarrow \emptyset$
define $f = (x, l, n)$ where x and l describe the current shift range; n is the number of data in the current subtree
for $i \leftarrow 0$ **to** $|D|$ **do**
    set $f = (d_i, 0, 1)$; add f to F
    while (there exist $f_l = (x_l, l_l, n_l)$ and $f_r = (x_r, l_r, n_r)$ in F, such that $n_l = n_r$) **do**
        set $d_g = \max\{x_l + l_l,\ x_r + l_r\}$ and $d_s = \min\{x_l - l_l,\ x_r - l_r\}$
        if $d_g - d_s \geq 2\Delta$ **then**
            set $b = (x_l - x_r)/2$, $l = \max\{l_l,\ l_r\}$ and $x = (x_l - b)$
        else
            set $b = 0$, $l = (d_g - d_s)/2$ and $x = (d_g + d_s)/2$
        end
        add $f = (x, l, 2n_r)$ to F
        delete f_l and f_r from F
        if $(b \neq 0)$ **then** add b to B
    end
end
Ensure: there is only one element $f = (x, l, n)$ in F
if $(|x| \geq |\Delta - l|)$ **then** add x to B
output B

Algorithm 1: F-Shift(D,Δ)

4.1 Description of F-Shift

Roughly, the F-Shift algorithm constructs a synopsis from d_0 up to d_{N-1} gradually onwards (line 3). This progressive feature is desirable in on-line processing of stream data.

Lines 7–11 is the key part of the algorithm. It uses a "fixed" value to convert an $\underline{s}$-unbound subtree into an s-bound one. It constructs $f = (x, l, 2n)$ from $f_L = (x_L, l_L, n)$ on the left subtree and $f_R = (x_R, l_R, n)$ on the right subtree. Suppose that d_{L_g} and d_{L_s} are the largest and smallest data values on the left subtree. Each data in the subtree is in the interval of (d_{L_s}, d_{L_g}). In $f_L = (x_L, l_L, n)$, $x_L = (d_{L_g} + d_{L_s})/2$ is the mean of d_{L_g} and d_{L_s}, $l_L = (d_{L_g} - d_{L_s})/2$ is the data scope, n is the number of data in the subtree.

If the current subtree is $\underline{s}$-unbound, a shift transformation is required (line 8). Let the shift value $b = (x_L - x_R)/2$. Clearly, such b satisfies (3.6). Line 17 decides s_0, the shift value of c_0. This step is based on (3.4). If $|x_1| \geq \Delta - l$ then shift c_0 into x_1 that satisfies (3.4).

Lines 14 and 18 indicate that each derived shift coefficient is added into B. In fact, each derived shift coefficients can be output directly as each early derived shift coefficient is not used for the construction of the later derived shift coefficients.

Property 3.1 guarantees that lines 5–15 work on an error tree at each iteration. Property 3.3 assures that the final B of the F-Shift algorithm is a Δ-bound synopsis. The value of each shift coefficient is assigned according to the designated shift range.

Example 4.1 Considering error tree T of Fig. 1(ii), F-shift algorithm generates $s_5 = 8$, $s_1 = 5.75$ and $s_0 = 5.75$ incrementally and eventually obtains the synopsis $S_F = \{s_5, s_1, s_0\}$ through the steps in Table 2.

4.2 Complexity and Upper Bound

In the F-Shift algorithm, the parameters on a node are derived from the parameters on the two children nodes with a constant number of operations. Since there are N number of nodes that need to be processed at most, the total time required to compute B (i.e., S_F) is $O(N)$.

As the F-Shift algorithm works in bottom up from the data vector, each s_i of S_F needs to be derived from the information on nodes $\{c_{2i}, c_{2^2i+2}, c_{2^3i+2^2+2}, \ldots\}$, one node at each level of $T(c_i)$, with at most $\log N$ number of nodes. If each retained coefficient s_i is accumulated into B as denoted in line 14 and line 18 of Algorithm 1, the space complexity of F-Shift is $O(\log N + |B|)$. However, since each early retained shift coefficient is not used to construct the later retained shift coefficients, the retained shift coefficient can be output directly at line 14 and line 18 of Algorithm 1. The space complexity of F-Shift is reduced to $O(\log N)$.

To estimate $|S_F|$, let the number of $\overline{s}$-bound subtrees of T be t_s. Since the F-Shift algorithm needs to shift the root node of two adjacent $\overline{s}$-bound subtrees to make it s-bound, it needs to shift at most $t_s - 1$ nodes to make T become s-bound. Again the

Table 2 Running steps of Example 4.1

Step	Line No.	f	F	Assigned values
1	4	$(19, 0, 1)$	$(19, 0, 1)$	
2	4	$(17, 0, 1)$	$(19, 0, 1), (17, 0, 1)$	
3	6			$d_g = 19, d_s = 17$
4	10			$l = 1, x = 18$
5	12–14		$(18, 1, 2)$	
6	4	$(12, 0, 1)$	$(18, 1, 2), (12, 0, 1)$	
7	4	$(-4, 0, 1)$	$(18, 1, 2), (12, 0, 1), (-4, 0, 1)$	
8	6			$d_g = 12, d_s = -4$
9	8			$\mathbf{b_5} = \mathbf{8}, l = 0, x = 4$
10	12–14		$(18, 1, 2), (4, 0, 2)$	
11	6			$d_g = 19, d_s = 4$
12	10			$l = 7.5, x = 11.5$
13	12–14		$(11.5, 7.5, 4)$	
14	4	$(7, 0, 1)$	$(11.5, 7.5, 4), (7, 0, 1)$	
15	4	$(-1, 0, 1)$	$(11.5, 7.5, 4), (7, 0, 1), (-1, 0, 1)$	
16	6			$d_g = 7, d_s = -1$
17	10			$l = 4, x = 3$
18	12–14		$(11.5, 7.5, 4), (3, 4, 2)$	
19	4	$(-3, 0, 1)$	$(11.5, 7.5, 4), (3, 4, 2), (-3, 0, 1)$	
20	4	$(-7, 0, 1)$	$(11.5, 7.5, 4), (3, 4, 2), (-3, 0, 1), (-7, 0, 1)$	
21	6			$d_g = -3, d_s = -7$
22	10			$l = 2, x = -5$
23	12–14		$(11.5, 7.5, 4), (3, 4, 2), (-5, 2, 2)$	
24	6			$d_g = 7, d_s = -7$
25	10			$l = 7, x = 0$
26	12–14		$(11.5, 7.5, 4), (0, 7, 4)$	
27	6			$d_g = 19, d_s = -7$
28	8			$\mathbf{b_1} = \mathbf{5.75}, l = 7.5, x = 5.75$
29	12–14		$(5.75, 7.5, 8)$	
30	17		$(5.75, 7.5, 8)$	$\mathbf{b_0} = \mathbf{5.75}$

F-Shift algorithm may need to adjust c_0 node at the last step to make T Δ-bound. Therefore, $|S_F| \leq 1 + t_s - 1$. That is, $|S_F| \leq t_s$. Thus the following lemma is proven.

Lemma 4.2 *Let the number of $\overline{s}$-bound subtrees of T is t_s. Then $|S_F| \leq t_s$.*

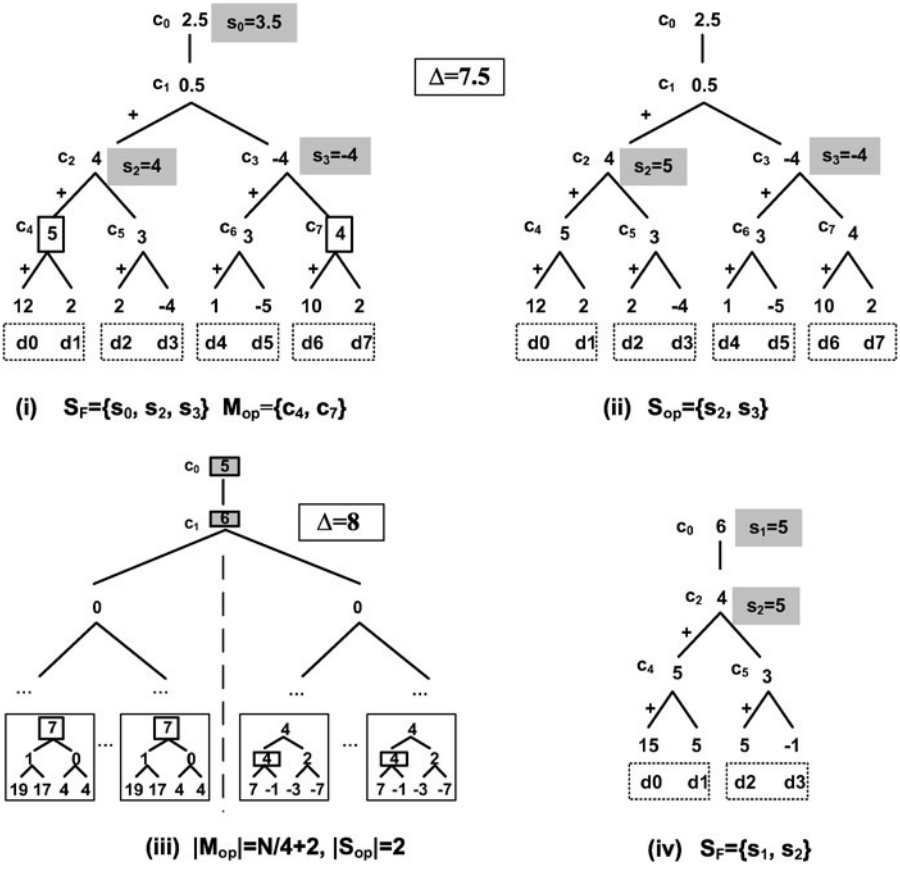


Fig. 2 Example

Moreover, the above bound on S_F is "tight": it can easily find a example such that $|S_F| = t_s$. However, this bound is not applicable to restricted-optimal synopsis M_{op}: It can be either $|M_{op}| \leq t_s$ or $|M_{op}| > t_s$.

Example 4.3 In Fig. 2(iv), $|S_F| = t_s = 2$ hold. In Fig. 1(ii), since $t_s = 4$ and $|M_{op}| = 5$, $|M_{op}| > t_s$ hold. In Fig. 2(i), since $t_s = 4$ and $|M_{op}| = 2$, $|M_{op}| < t_s$ hold.

Furthermore, we have $|S_F| \leq |S_{op}| \log N$. This is because, for each c_i that satisfies $|T(c_i) \sqcap S_F| = 1$, there exist s_j such that $s_j \in S_{op} \sqcap T(c_i)$ (Property 3.6(iii)). Let S_J be the set of all s_j. Then $S_J \subseteq S_{op}$. As the number of ancestors of s_j is bounded by $\log N$, $|S_F| \leq |S_J| \log N \leq |S_{op}| \log N$ holds. In summary, we have proven the following theorem for this section.

Theorem 4.4 *Let D be a data vector with size N and $\Delta > 0$. Suppose S_F is the set of shift coefficients obtained from F-Shift algorithm on D and Δ. Then*

- *S_F is a synopsis*;

- *the time and space complexity of F-Shift algorithm are* $O(N)$ *and* $O(\log N)$ *respectively*;
- *the size of* S_F *is bounded by* $\min\{t_s, |S_{op}| \log N\}$.

Example 4.5 In Fig. 2(i), it can be verified that $M_{op} = \{c_4, c_7\}$ and $S_F = \{s_0, s_2, s_3\}$ for $s_0 = 3.5$, $s_2 = 4$ and $s_3 = -4$. From (3.6), $range(c_2) = [3, 5]$ and $range(c_3) = [-4.5, -3.5]$. It can also show that $S_{op} = \{s'_2, s'_3\}$ for $s'_2 = 5$ and $s'_3 = -4$ in Fig. 2(ii).

Besides $|S_F| > |S_{op}|$, the above example also indicates that choosing proper shift values can reduce the size of S_F. This topic will be discussed in Sect. 5. In contrast, as indicated in the following example, the size of R-optimal $|M_{op}|$ can be very large even if $|S_{op}|$ is quite small.

Example 4.6 As depicted in Fig. 2(iii), D is the data vector that is formed by repeating $D_L = [19; 17; 4; 4]$ and $D_R = [7; -1; -3; -7]$ on its left and right subtree respectively. Let $\Delta = 8$. It can prove that $S_F = \{s_0 = 5.75, s_1 = 5.75\}$ for each $N = 2^n > 8$. It can also show that $|M_{op}| = N/4 + 2$. For instance, $M_{op} = \{c_0, c_1, c_4, c_5, c_{12}, c_{14}\}$ when $N = 16$.

5 S-Shift Algorithm

The S-Shift (Sliding-value Shift) algorithm is aimed at constructing a smaller sized synopsis. As indicated in (3.6), Sect. 3.3, there exist many shift values other than $b = (x_L - x_R)/2$ that used for the F-Shift algorithm. As an extension of the F-Shift algorithm, S-Shift algorithm employs shift ranges and "late binding" on shift values to compute a synopsis to diminish the number of shift coefficients. In particular, the strategy behind S-Shift is that first find the set of coefficients that need to be shifted along with their fused shift ranges from bottom-up and then instantiate a proper shift value for each selected coefficient from its range to obtain a synopsis in top-down. In this way, some shift coefficients in S_F can be evaporated away through fusing the shift ranges on their subtrees.

In this section, we will firstly describe the S-Shift algorithm in detail along with a running example. We then show that S-Shift is the "optimal" F-Shift and has linear time/space complexity in Sect. 5.2.

5.1 Description of S-Shift

Depicted in Algorithm 2, the S-Shift algorithm has two phases in function: Phase A (bottom-up phase) and Phase B (top-down phase). In Phase A, it marks the coefficients that need to be shifted and computes the reduced shift range from the ranges on the left and right subtrees. In Phase B, it locates a feasible shift value within the shift range for each marked coefficient generated in Phase A.

Input: D, the original data set; Δ, the specified error bound; T, the error tree
Output: B, the set of shift coefficients that satisfies Δ bound

1 $B \leftarrow \emptyset$
2 **for** $i \leftarrow 0$ **to** $|D|$ **do**
3 set $f_{N+i} = (c_{N+i}, d_i, d_i, 0, 0)$
4 **end**
5 *Bottom-Up Phase A: Compute B and* $f_i = (c_i, \underline{x_i}, \overline{x_i}, l_i, g_i)$ *from* f_{2i} *and* f_{2i+1} *from bottom-up* (Refer to Sect. 5.1.1)
6 *Top-Down Phase B: Compute each value of B,* $f_{2i} = (c_{2i}, x_{2i}, x_{2i}, l_{2i}, g_{2i})$ *and* $f_{2i+1} = (c_{2i+1}, x_{2i+1}, x_{2i+1}, l_{2i+1}, g_{2i+1})$ *from* $f_i = (c_i, x_i, x_i, l_i, g_i)$ (Refer to Sect. 5.1.2)
7 output B

Algorithm 2: S-Shift(D,Δ)

To obtain a synopsis,[5] the S-Shift algorithm computes and stores $f = (c, \underline{x}, \overline{x}, l, g)$ in bottom-up for each coefficient c: l is the scope of the current $T(c)$; g is a flag indicator that specifies c need to shift if g is assigned to 1. The interval $[\underline{x}, \overline{x}]$ indicates that the data interval $[d_s, d_g]$ of $T(c)$ can slide from $[\underline{x} - l, \underline{x} + l]$ to $[\overline{x} - l, \overline{x} + l]$ and can be denoted as $[x - l, x + l]$ for $x \in [\underline{x}, \overline{x}]$ in general.

The detailed descriptions and proof for the two phases are given in the next two subsections.

5.1.1 Phase A (Line 5) Description

The central part of this phase is to compute $f = (c, \underline{x}, \overline{x}, l, g)$ from left child $f_L = (c_L, \underline{x_L}, \overline{x_L}, l_L, g_L)$ and right child $f_R = (c_R, \underline{x_R}, \overline{x_R}, l_R, g_R)$ based on the following two facts:

(i) $l_L < \Delta$ and $l_R < \Delta$;
(ii) each data d_L of $T_L(c)$ is in interval $R(x_L) = [x_L - l_L, x_L + l_L]$ and each data d_R of $T_R(c)$ is in interval $R(x_R) = [x_R - l_R, x_R + l_R]$ for $x_L \in [\underline{x_L}, \overline{x_L}]$ and $x_R \in [\underline{x_R}, \overline{x_R}]$.

To obtain $f = (c, \_)$, the procedure first checks whether coefficient c need be shifted then computes the parameters accordingly. Clearly, coefficient c do not need to shift if and only if $\exists x_L, x_R$, where $x_L \in [\underline{x_L}, \overline{x_L}]$ and $x_R \in [\underline{x_R}, \overline{x_R}]$, such that[6]

$$\max_{\substack{d' \in R(x_L) \\ d'' \in R(x_R)}} |d' - d''| < 2\Delta. \tag{5.1}$$

Roughly, the idea of this phase is as follows:

[5] For easy descriptions, in S-Shift algorithm, we add c in $f(c, \_)$ and use $[\underline{x}, \overline{x}]$ rather than $[\underline{s}, \overline{s}]$ as in Definition 3.9.

[6] Or, equivalently, $|x_L - x_R| + (l_L + l_R) < 2\Delta$ as expressed similar to (5.2).

1. If (5.1) holds, obtain x such that $R(x)$ is enclosed in every $R(x_L) \cup R(x_R)$ that will be illustrated in Step A2 and Step A3 in the following.
2. Otherwise, shift coefficient c and obtain x such that $R(x_L - x)$ and $R(x_R + x)$ are enclosed that will be illustrated in Step A1 in the following.

More specifically, the computation of $f = (c, \underline{x}, \overline{x}, l, g)$ from $f_L = (c_L, \underline{x_L}, \overline{x_L}, l_L, g_L)$ and $f_R = (c_R, \underline{x_R}, \overline{x_R}, l_R, g_R)$ works as follows:

A1. Compute parameters when c needs to be shifted. Coefficient c need to be shifted if

$$\min |x_L - x_R| + (l_L + l_R) \geq 2\Delta. \tag{5.2}$$

This means that current $T(c)$ is $\underline{s}$-unbound no matter what values of $x_L \in [\underline{x_L}, \overline{x_L}]$ and $x_R \in [\underline{x_R}, \overline{x_R}]$ are assigned. In this situation, coefficient c need to be shifted into s. In order to minimize the scope of $T\langle s \rangle$, s should satisfy (3.6). Furthermore, s need to accumulate the shift ranges from $T_L(c)$ and $T_R(c)$. Therefore, it need to replace s_i in (3.6) with $x = x_L - s_i$ (or $x = x_R + s_i$) to acquire the shift range of x:

$$(x_L + x_R) - |l_L - l_R| \leq 2x \leq (x_L + x_R) + |l_L - l_R|.$$

By replacing $(x_L + x_R)$ with $(\underline{x_L} + \underline{x_R})$ on the left and with $(\overline{x_L} + \overline{x_R})$ on the right in the above formula,

$$(\underline{x_L} + \underline{x_R}) - |l_L - l_R| \leq 2x \leq (\overline{x_L} + \overline{x_R}) + |l_L - l_R|.$$

Based on this formula, set

$$\begin{cases} \underline{x} = [(\underline{x_L} + \underline{x_R}) - |l_L - l_R|]/2, \\ \overline{x} = [(\overline{x_L} + \overline{x_R}) + |l_L - l_R|]/2, \\ l = \max\{l_L, l_R\} \quad \text{and} \quad g = 1. \end{cases}$$

A2. Compute parameters when c does not need to shift and

$$|l_L - l_R| < \min |x_L - x_R| < 2\Delta - (l_L + l_R)$$

holds. In this situation, although $T(c)$ is s-bound, there exist a unique maximum overlapped data range (i.e., $\underline{x} = \overline{x}$) and the minimized scope of $T(c)$ (through adjusting x_L and x_R) is large than $\max\{l_L, l_R\}$. Set

$$l = (l_L + l_R + \min |x_L - x_R|)/2 \quad \text{and} \quad g = 2.$$

Since the upper data interval needs to move downwards and the lower interval needs to move upwards, two cases exist:

- If $\underline{x_L} > \underline{x_R}$, set $\underline{x}$ and $\overline{x}$ to be $(l_L + \underline{x_L} + \overline{x_R} - l_R)/2$ as $x + l = l_L + \underline{x_L}$ and $x - l = \overline{x_R} - l_R$.
- If $\underline{x_L} \leq \underline{x_R}$, set $\underline{x}$ and $\overline{x}$ to be $(l_R + \underline{x_R} + \overline{x_L} - l_L)/2$ as $x + l = \underline{x_R} + l_R$ and $x - l = \overline{x_L} - l_L$.

A3. Compute parameters when c does not need to shift and

$$\min|x_L - x_R| \le |l_L - l_R|$$

holds. Under this situation, the maximum overlapped range can be located in many places (i.e., $\underline{x} < \overline{x}$). Set

$$l = \max\{l_L,\ l_R\} \quad \text{and} \quad g = 3.$$

If $l_L > l_R$, move the shorter scope (l_R) to be included by the longer scope (l_L). Using x to replace x_L in the above formula,

$$\begin{cases} -|l_L - l_R| \le x - x_R \le |l_L - l_R|, \\ \underline{x_L} \le x \le \overline{x_L}. \end{cases}$$

This implies,

$$\max\{\underline{x_L},\ \underline{x_R} - |l_L - l_R|\} \le x \le \min\{\overline{x_L},\ \overline{x_R} + |l_L - l_R|\}.$$

Similarly, if $l_L \le l_R$,

$$\max\{\underline{x_R},\ \underline{x_L} - |l_L - l_R|\} \le x \le \min\{\overline{x_R},\ \overline{x_L} + |l_L - l_R|\}.$$

Therefore,

1. if $l_L > l_R$, set

$$\begin{cases} \underline{x} = \max\{\underline{x_L}, \underline{x_R} - |l_L - l_R|\}, \\ \overline{x} = \min\{\overline{x_L},\ \overline{x_R} + |l_L - l_R|\}. \end{cases}$$

2. If $l_L \le l_R$, set

$$\begin{cases} \underline{x} = \max\{\underline{x_R},\ \underline{x_L} - |l_L - l_R|\}, \\ \overline{x} = \min\{\overline{x_R},\ \overline{x_L} + |l_L - l_R|\}. \end{cases}$$

5.1.2 Phase B (Line 6) Description

The objective of this phase is to assign a proper shift value[7] for each $\{c_i \,|\, f_i = (c_i, \underline{x_i}, \overline{x_i}, l_i, g_i) \wedge (g_i = 1)\}$ and narrow down the range $[\underline{x_i}, \overline{x_i}]$ into an instance value x_i of $\underline{x_i} \le x_i \le \overline{x_i}$ from top-down starting at c_0.

First, it computes s_0, the shift coefficient at c_0 and an instance value from $[\underline{x_1}, \overline{x_1}]$ to satisfy the selected s_0. By replacing x_1 with $\underline{x_1}$ (and $\overline{x_1}$) in (3.4), it obtains

$$\underline{x_1} - (\Delta - l_1) < s_0 < \overline{x_1} + (\Delta - l_1).$$

If s_0's range includes 0, then c_0 does not need to shift and let $s_0 = 0$. Otherwise, set $s_0 = (\underline{x_1} + \overline{x_1})/2$. It can prove that $|x_1 - s_0| < |l_1 - \Delta|$. Add s_0 into B and set $sum(x) = s_0$.

[7] There may exist many feasible shift values for c_i. Without loss the generality, we assign a "middle" value in the range in our approach.

With the instantiated s_0, we need to instantiate data range $[\underline{x_1}, \overline{x_1}]$ into x to satisfy the selected s_0. Such x should satisfy,

$$\begin{cases} -(\Delta - l_1) < x < (\Delta - l_1), \\ \underline{x_1} - s_0 \le x \le \overline{x_1} - s_0. \end{cases}$$

In the above two formulas, the first one comes from (3.4) by replacing $x_1 - s_0$ with x while the second means that $x_1 = x + s_0$. Set

$$\underline{x} = \overline{x} = \big(\max\{-|l_1 - \Delta|, \underline{x_1} - s_0\} + \min\{|l_1 - \Delta|, \overline{x_1} - s_0\}\big)/2.$$

Next, update $f_L = (c_L, \_)$, $f_R = (c_R, \_)$ into,[8]

$$\begin{cases} f_L = (c_L, \underline{x_L} - sum(x), \overline{x_L} - sum(x), l_L, g_{L\_}), \\ f_R = (c_R, \underline{x_R} - sum(x), \overline{x_R} - sum(x), l_R, g_{R\_}). \end{cases}$$

Iteration Step: With the obtained $f = (c, \underline{x}, \overline{x}, l, g)$ and $sum(x)$, x_L in $f_L = (c_L, \_)$ and x_R in $f_R = (c_R, \_)$ will be instantiated from the following cases.

B1. When $f = (c, \underline{x}, \overline{x}, l, g) \wedge (g = 1)$ holds. Since $|x'_L - x'_R| < |l_L - l_R|$, $x'_L \in [\underline{x_L} - s_c, \overline{x_L} - s_c]$ and $x'_R \in [\underline{x_R} + s_c, \overline{x_R} + s_c]$,

$$\begin{cases} -|l_L - l_R| < x'_L - x'_R < |l_L - l_R|, \\ \underline{x_L} \le x'_L + s_c \le \overline{x_L}, \\ \underline{x_R} \le x'_R - s_c \le \overline{x_R}. \end{cases} \tag{5.3}$$

If $l_L \ge l_R$, set $x'_L = x$. First, derive the ranges of $(\underline{s_c}, \overline{s_c})$ by (5.3). Then set $s_c = (\underline{s_c} + \overline{s_c})/2$. Again from (5.3) and s_c, derives $(\underline{x'_R}, \overline{x'_R})$ and set $x'_R = (\underline{x'_R} + \overline{x'_R})/2$.

If $l_L < l_R$, set $x'_R = x$. Similar to the above, find x'_L and s_c.

B2. $f = (c, \underline{x}, \overline{x}, l, g) \wedge (g = 2)$ holds. Set $s_c = 0$, $x'_L = \underline{x_L}$ and $x'_R = \overline{x_R}$.

B3. $f = (c, \underline{x}, \overline{x}, l, g) \wedge (g = 3)$ holds. Set $v_c = 0$. Since $|x'_L - x'_R| < |l_L - l_R|$, $x_L = x'_L$ and $x_R = x'_R$,

$$\begin{cases} -|l_L - l_R| < x'_L - x'_R < |l_L - l_R|, \\ \underline{x_L} \le x'_L \le \overline{x_L}, \\ \underline{x_R} \le x'_R \le \overline{x_R}. \end{cases} \tag{5.4}$$

Similar to B1, except that $s_c = 0$ in this case.

Next, process the children nodes after doing the following:

$$\begin{cases} \text{Add } s_c \text{ into } B \text{ if } s_c \ne 0, \\ \text{Set } sum(x_L) = sum(x) + s_c \text{ and } sum(x_R) = sum(x) - s_c, \\ \text{Set } f_L = (c_L, x'_L, x'_L, l_L, g_L) \text{ and } f_R = (c_R, x'_R, x'_R, l_R, g_R). \end{cases}$$

[8] In fact, this step update f_2 and f_3.

A Running Example: The idea of S-Shift algorithm is illustrated in the following on the example of Fig. 2(i).

In Phase A: For each s-bound subtree $T(c_i)$, set $f_i = (c_i, (d_s + d_g)/2, (d_s + d_g)/2, (d_g - d_s)/2, 0)$ where d_g and d_s are the max and min data in $T(c_i)$ respectively. Therefore,

$$f_4 = (c_4, 7, 7, 5, 0), \qquad f_5 = (c_5, -1, -1, 3, 0),$$
$$f_6 = (c_6, -2, -2, 3, 0), \qquad f_7 = (c_7, 6, 6, 4, 0).$$

Next, the algorithm generates f_i from f_{2i} and f_{2i+1}. Using f_4 and f_5 to generate $f_2 = (c_2, \underline{x_2}, \overline{x_2}, l_2, g_2)$, a shift coefficient s_2 is required as $T(c_2)$ is $\underline{s}$-unbound. g_2 is set to 1 indicating that c_2 needs to shift. As indicated in Example 4.5, $s_2 \in range(c_2)$ where $range(c_2) = [3, 5]$. For $s_2 \in range(c_2)$, the data interval $[d_s, d_g]$ of $T(s_2)$ slides from $[-3, 7]$ to $[-1, 9]$ with data scope $l_2 = (d_g - d_s)/2 = 5$. $[\underline{x_2}, \overline{x_2}]$ is used to describe the data interval of $[-1, 9] \vee [-3, 7] = [-3, 9]$ and is set to $[-3 + l_2, 9 - l_2] = [2, 4]$. As a result, $f_2 = (c_2, 2, 4, 5, 1)$. Similarly, $f_3 = (c_3, 1.5, 2.5, 4, 1)$ can be derived.

Now, it comes to compute $f_1 = (c_1, \underline{x_1}, \overline{x_1}, l_1, g_1)$ from f_2 and f_3. Coefficient c_1 will not need to be shifted if the data scope $l_1 < \Delta$. Furthermore, l_1 can be described by $[x_2 - l_2, x_2 + l_2] \vee [x_3 - l_3, x_3 + l_3]$ where $x_2 \in [2, 4]$ and $x_3 \in [1.5, 2.5]$. To minimize $[x_2 - l_2, x_2 + l_2] \vee [x_3 - l_3, x_3 + l_3]$, l_1 is set to $\max\{l_2, l_3\} = 5$ as the data scope of $T(c_1)$ can not be shorter than $\max\{l_2, l_3\}$ and $x_1 \in [\underline{x_1}, \overline{x_1}]$ satisfies,

$$\begin{cases} -|l_2 - l_3| \le x_1 - x_3 \le |l_2 - l_3|, \\ \underline{x_2} \le x_1 \le \overline{x_2}. \end{cases}$$

Intuitively, the first equation in the above means $[x_3 - l_3, x_3 + l_3] \subseteq [x_1 - l_1, x_1 + l_1]$. The following is then derived:

$$\underline{x_1} = \max\{\underline{x_2}, \underline{x_3} - |l_2 - l_3|\} = 2, \qquad \overline{x_1} = \min\{\overline{x_2},\ \overline{x_3} + |l_2 - l_3|\} = 3.5.$$

Up to now, this phase has selected the set of shift coefficients $\{s_2, s_3\}$. Their specific values will be determined in Phase B.

In Phase B: First, it decides s_0, the shift coefficient at c_0 and an instance value from $[\underline{x_1}, \overline{x_1}]$ to satisfy the selected s_0.

By replacing x_1 with $\underline{x_1}$ (and $\overline{x_1}$) in (3.4), it obtains

$$2 - (7.5 - 1) < s_0 < 3.5 + (7.5 - 1),$$

which is $s_0 \in (-4.5, 10)$. Since $0 \in (-4.5, 10)$, it means that c_0 does not need to shift. Next, it need to select $x \in [\underline{x_1} - s_0, \overline{x_1} - s_0]$ from

$$-|\Delta - l_1| < x < |\Delta - l_1|$$

to support the selected s_0 (equation (3.4)). Clearly, an instance of the above equation is

$$\underline{x} = \overline{x} = \big(\max\{-|\Delta - l_1|,\ \underline{x_1} - s_0\} + \min\{|\Delta - l_1|,\ \overline{x_1} - s_0\}\big)/2 = 2.25$$

that results $f_1 = (c_1, 2.25, 2.25, 5, \_)$.

Similarly, the following equations is used to derive the instances of x_2 in f_2 and x_3 in f_3.

$$\begin{cases} -|l_2 - l_3| < x_2 - x_3 < |l_2 - l_3|, \\ \underline{x_2} \le x_2 \le \overline{x_2}, \\ \underline{x_3} \le x_3 \le \overline{x_3} \end{cases} \Rightarrow \begin{cases} -1 < x_2 - x_3 < 1, \\ 2 \le x_2 \le 4, \\ 1.5 \le x_3 \le 2.5. \end{cases}$$

Set $x_2 = x_1 = 2.25$ as $l_2 > l_3$ and find one feasible solution is $x_3 = (1.5 + 2.5)/2 = 2$. Thus, $f_2 = (c_2, 2.25, 2.25, 5, 1)$ and $f_3 = (c_3, 2, 2, 4, 1)$.

To find the value of s_2 from $f_2 = (c_2, 2.25, 2.25, 5, 1)$, $f_4 = (c_4, 7, 7, 5, 0)$ and $f_5 = (c_5, -1, -1, 3, 0)$, we have the following equations:

$$\begin{cases} -|l_4 - l_5| < x_4 - x_5 < |l_4 - l_5|, \\ \underline{x_4} \le x_4 + s_2 \le \overline{x_4}, \\ \underline{x_5} \le x_5 - s_2 \le \overline{x_5} \end{cases} \Rightarrow \begin{cases} -2 < x_4 - x_5 < 2, \\ x_4 + s_2 = 7, \\ x_5 - s_2 = -1. \end{cases}$$

Set $x_4 = x_2 = 2.25$ as $l_4 > l_5$ and find one feasible solution for s_2 is 4.75. Similarly, it can derive that $s_3 = 4$.

5.2 The Property of S-Shift

In the bottom-up phase of the S-Shift algorithm, each $f = (c, \_)$, derived from $f_L = (c_L, \_)$ and $f_R = (c_R, \_)$ with a bounded number of operations, requires $O(1)$ time. Since there are N number of nodes that need to be processed, the time required in this phase is $O(N)$. The space complexity of this phase is also $O(N)$ as each $f = (c, \_)$ is required in the top-down phase. In the top-down phase, each $f = (c, \_)$ is used either to instantiate a value for an element of B or to narrow down shift ranges. This phase requires $O(N)$ in both memory and time for processing N number of nodes. Therefore, the total time/memory requirement of the S-Shift algorithm is $O(N)$.

With regard to the size of S_S, the synopsis constructed from the S-Shift algorithm relates a varied form of U-optimal problem, which is called *U-scope-based-optimal* problem that stated as follows.

Definition 5.1 Given data vector $D = [d_0; d_1; \ldots; d_{N-1}]$ and error bound Δ, the U-scope-shift-optimal (U_s-optimal) problem is to find S_{opt}, a synopsis with the smallest set of unrestricted coefficients among all possible solutions, such that

(i) S_{opt} is a synopsis. That is, $|d_i - \hat{d}_i^{(S_{opt})}| < \Delta$ holds for $0 \le i \le N - 1$.
(ii) $T_i\langle S_i\rangle$ is $\underline{s}$-unbound for each $s_i \in S_{opt}$ where $S_i = S_{opt} - \{s_i\}$ and $T_i = T(c_i)$ and $i > 0$.

Roughly, the difference between U-optimal problem and U_s-optimal problem is that s_i can be chosen from a s-bound subtree in a U-optimal problem but cannot be so in a U_s-optimal problem.

As mentioned early, Phase A minimizes the scope of each subtree and uses (3.6) rather than (3.3) to compute ranges. One issue of concern is that this approach of minimization may result in the lose of generality (i.e., a smaller sized synopsis may exist if uses (3.3)). The concern is clarified in Property 5.2.

Property 5.2 *Let $T_{i_L} = T_L(c_i)$ and $T_{i_R} = T_R(c_i)$ be the left and right subtree of $T_i = T(c_i)$.*

(1) *If $s_i \in S_{opt}$ and s_i does not lead to the minimal scope of $T_i\langle S_{opt}\rangle$ from $T_{i_L}\langle S_{opt}\rangle$ and $T_{i_R}\langle S_{opt}\rangle$. Then there exists s_i' that can minimize the scope of $T_i\langle S_{opt}\rangle$ such that $(S_{opt} - \{s_i\}) \cup \{s_i'\}$ is a synopsis.*

(2) *Suppose coefficient c_i is not shifted (i.e., $s_i \notin S_{opt}$). Let $S = T(c_i) \sqcap S_{opt}$, $S_L = T_L(c_i) \sqcap S_{opt}$ and $S_R = T_R(c_i) \sqcap S_{opt}$. We use d_s and d_g (d'_{L_s} and d'_{L_g}, d'_{R_s} and d'_{R_g}, respectively) to denote the smallest and largest data of $T_i\langle S\rangle$ ($T_L\langle S'_L\rangle$, $T_R\langle S'_R\rangle$, respectively) where S'_L (or S'_R) is obtained from S_L (or S_R) by assigning different values to some of its elements. If $d_s \le \min\{d'_{L_s}, d'_{R_s}\}$ and $\max\{d'_{L_g}, d'_{R_g}\} \le d_g$, then $(S_{opt} - S_L) \cup S'_L$, $(S_{opt} - S_R) \cup S'_R$ and $(S_{opt} - S_L \cup S_R) \cup S'_L \cup S'_R$ are synopses.*

Proof We only prove (1) as (2) can be proven similarly.

Let d_g and d_s (d_{L_g} and d_{L_s}, or d_{R_g} and d_{R_s}) be the largest and smallest data values of $T_i\langle S_{opt}\rangle$ ($T_L\langle S_{opt}\rangle$ or $T_R\langle S_{opt}\rangle$) respectively where $T_i = T(c_i)$. Since s_i is not maximum overlapping the scopes of $T_L\langle S_{opt}\rangle$ and $T_R\langle S_{opt}\rangle$, $(d_{L_g} - s_i > d_{R_g} + s_i) \wedge (d_{L_s} - s_i > d_{R_s} + s_i)$ or $(d_{L_g} - s_i < d_{R_g} + s_i) \wedge (d_{L_s} - s_i < d_{R_s} + s_i)$ holds.

If $(d_{L_g} - s_i > d_{R_g} + s_i) \wedge (d_{L_s} - s_i > d_{R_s} + s_i)$, let $s_i' = \min\{d_{L_g} - d_{R_g}, d_{L_s} - d_{R_s}\}$. It can be proven that

$$\begin{cases} \max\{(d_{L_g} - s_i'), (d_{R_g} + s_i')\} \le \max\{(d_{L_g} - s_i), (d_{R_g} + s_i)\}, \\ \min\{(d_{L_s} - s_i), (d_{R_s} + s_i)\} \le \min\{(d_{L_s} - s_i'), (d_{R_s} + s_i')\}. \end{cases}$$

That is,

$$\begin{cases} \max\{(d_{L_g} - s_i'), (d_{R_g} + s_i')\} \le d_g, \\ d_s \le \min\{(d_{L_s} - s_i'), (d_{R_s} + s_i')\}. \end{cases}$$

Let $d'_g = \max\{(d_{L_g} - s_i'), (d_{R_g} + s_i')\}$ and $d'_s = \min\{(d_{L_s} - s_i'), (d_{R_s} + s_i')\}$. The above formulae imply that $(|d'_g - \gamma| < \Delta) \wedge (|d'_s - \gamma| < \Delta)$ if $|d_g - \gamma| < \Delta$ and $|d_s - \gamma| < \Delta$. Therefore, $(S_{opt} - \{s_i\}) \cup \{s_i'\}$ is a synopsis by definition.

Similarly, it can be proven that the result for the situation of $(d_{L_g} - s_i < d_{R_g} + s_i) \wedge (d_{L_s} - s_i < d_{R_s} + s_i)$. □

Intuitively, Property 5.2 means that S_{opt} can always be constructed by minimizing the scopes of subtrees. Since the S-Shift algorithm checks all possible shift values that lead to the minimized scopes of subtrees, the following theorem is proven.

Theorem 5.3 *Let D be a data vector with size N and $\Delta > 0$. Suppose S_S is the set of shift coefficients obtained from S-Shift algorithm on D and Δ. Then*

- *S_S is an optimal synopsis for the U_s-optimal problem (i.e., $|S_S| = |S_{opt}|$ and $|S_S| \le |S_F|$);*
- *the time and space complexity of S-Shift algorithm are $O(N)$;*
- *the size of S_S is bounded by $\min\{t_s, |S_{op}| \log N\}$.*

In the above theorem, the bound on $|S_S|$ is directly obtained from Theorem 4.4.

6 On Relative Error

In this section, we will show that our idea can be used to construct the unrestricted synopses on maximum relative error. Minimizing the maximum relative error may be more essential compared to absolute error minimization in approximate query processing. This is because the same absolute error in two different data values may express huge differences in relative error. With the same idea used in the Shift algorithms, we shall briefly present the results on relative L_∞ in this section.

Let data vector D be $[d_0; d_1; \ldots; d_{N-1}]$ and approximation vector $\hat{D}$ be $[\hat{d}_0; \hat{d}_1; \ldots; \hat{d}_{N-1}]$. The relative error of d_i and $\hat{d}_i$ is defined as $\frac{|\hat{d}_i - d_i|}{\max\{|d_i|, \epsilon\}}$, where ϵ is a sanity bound in charge of preventing small data values from dominating the relative error. Given an error bound $\Delta > 0$, we say that approximation vector $\hat{D}$ is a Δ-approximation on maximum relative error if $\frac{|\hat{d}_i - d_i|}{\max\{|d_i|, \epsilon\}} < \Delta$ holds for $0 \le i \le N-1$.

Let $\vec{\Delta} = [\Delta_0; \ldots; \Delta_{N-1}]$ where $\Delta_i = \max\{|d_i|, \epsilon\}\Delta$ for $0 \le i \le N-1$. Clearly, $\vec{\Delta}$ can be obtained from D, Δ and sanity bound ϵ.

Therefore, finding unrestricted synopsis S_r from D on maximum relative error Δ can be described by using $\vec{\Delta}$. As depicted in Algorithm 3, the F_r-Shift algorithm computes unrestricted synopsis S_r from bottom-up on data vector D and $\vec{\Delta}$: Coefficient c_i $(i > 0)$ needs to be shifted if the range $(\underline{s}_L, \overline{s}_L)$ on the left subtree of $T(c_i)$ and the range $(\underline{s}_R, \overline{s}_R)$ on the right subtree of $T(c_i)$ does not share a common element.

To answer why Algorithm 3 works, we need to extend the concepts such as Δ-bound or s-bound to fit relative error $\vec{\Delta}$. Relevant to s-bound or s-unbound, we have the concepts on relative error.

Let $T' = T(c_i)$ be a subtree of error tree T rooted at node c_i. Subtree T' is $\vec{\Delta}$-*bound* if $|d_j| < \Delta_j$ holds for each d_j of T'; Subtree T' is r-*bound* if[9]

$$\bigcap_{d_j \in T'} (d_j - \Delta_j, d_j + \Delta_j) \neq \emptyset \tag{6.1}$$

holds or else T' is called r-*unbound*. Subtree T' is called $\underline{r}$-*unbound* if T' is r-unbound but its left subtree T'_L and right subtree T'_R are both r-bound. Intuitively, when T' is $\underline{r}$-unbound, then there exists an s_i-shift transformation such that $T'\langle s_i \rangle$ is r-bound. When T is r-bound but not $\vec{\Delta}$-bound, then there exists an s_0-shift transformation such that $T\langle s_0 \rangle$ is $\vec{\Delta}$-bound. These properties is detailed in Property 6.1, which is parallel to Property 3.8 and Property 3.6(iv) for the case of absolute error.

Property 6.1 *Let $T' = T(c_i)$ be a subtree of error tree T rooted at node c_i $(i > 0)$ and $\vec{\Delta}$ be a given relative error bound.*

(i) *If T' is $\underline{r}$-unbound for $\vec{\Delta}$, then there exists s_i such that $T'\langle s_i \rangle$ is r-bound.*
(ii) *If error tree T is r-bound but not $\vec{\Delta}$-bound, then there exists s_0 such that $T\langle s_0 \rangle$ is $\vec{\Delta}$-bound.*
(iii) *If $T(c_i)$ is $\underline{r}$-unbound for $\vec{\Delta}$, then $T(c_i) \sqcap S \neq \emptyset$ holds for any synopsis S.*

[9] It should be noted that the concept of r-bound extends the concept of s-bound. That is, when $\Delta_0 = \Delta_k$ holds for each Δ_k of $\vec{\Delta}$, it can be proven that $T(c_i)$ is r-bound if and only if $T(c_i)$ is s-bound.

Input: D, the original data vector; $\vec{\Delta}$, the specified error vector
Output: A synopsis S_r. That is, the set of shift coefficients that satisfies $\vec{\Delta}$ bound

$S_r \leftarrow \emptyset$; $F \leftarrow \emptyset$
define $f = (k, \underline{s}, \overline{s})$ where k is the number of data covered
for $(i \leftarrow 0)$ **to** $|D|$ **do**
  set $f = (1, d_i - \Delta_i, d_i + \Delta_i)$; add f to F
  while (there exists a $f_l = (k, \underline{s_l}, \overline{s_l})$ and $f_r = (k, \underline{s_r}, \overline{s_r})$ in F) **do**
    if $(\underline{s_r}, \overline{s_r}) \cap (\underline{s_l}, \overline{s_l}) = \emptyset$ **then**
      set $b = [(\underline{s_l} + \overline{s_l}) - (\underline{s_r} + \overline{s_r})]/4$;
      set $\overline{s} = \min\{\overline{s_l} - (\underline{s_l} - \overline{s_r})/2, \overline{s_r} + (\overline{s_l} - \underline{s_r})/2\}$;
      set $\underline{s} = \max\{\underline{s_l} - (\overline{s_l} - \underline{s_r})/2, \underline{s_r} + (\underline{s_l} - \overline{s_r})/2\}$
    else
      set $b = 0$;
      set $\overline{s} = \min\{\overline{s_r}, \overline{s_l}\}$;
      set $\underline{s} = \max\{\underline{s_r}, \underline{s_l}\}$
    end
    replace f_l and f_r with $f = (2k, \underline{s}, \overline{s})$ in F
    add b to S_r if $b \neq 0$
  end
end
Ensure: there is only one element $f = (n, \underline{s}, \overline{s})$ in F
if $0 \notin (\underline{s}, \overline{s})$ **then** set $b = (\underline{s} + \overline{s})/2$ and add b into S_r
output S_R

Algorithm 3: F_r-Shift(D, $\vec{\Delta}$): algorithm on relative error

Proof The proof of (i). Let T'_L and T'_R be the left and right subtree of T' respectively. Since T' is $\underline{r}$-unbound for $\vec{\Delta}$, T'_L and T'_R are r-bound for $\vec{\Delta}$. Therefore, according to (6.1), there exist values s_l and s_r satisfying

$$\begin{cases} d_L - \Delta_L < s_l < d_L + \Delta_L & \text{for each } d_L \in T'_L, \\ d_R - \Delta_R < s_r < d_R + \Delta_R & \text{for each } d_R \in T'_R. \end{cases}$$

Let $s_l + s_r = 2s$ and $s_l - s_r = 2s_i$. Then $s_l = s + s_i$ and $s_r = s - s_i$. Substitute them into the above equations and we have:

$$\begin{cases} d_L - s_i - \Delta_L < s < d_L - s_i + \Delta_L & \text{for each } d_L \in T'_L, \\ d_R + s_i - \Delta_R < s < d_R + s_i + \Delta_R & \text{for each } d_R \in T'_R. \end{cases} \tag{6.2}$$

That is, $T\langle s_i \rangle$ is r-bound for $\vec{\Delta}$ from the definitions of r-bound and shift transformations.

The proof of (ii). According to (6.1) and since T is r-bound, there exists value s_0 such that

$$d_j - \Delta_j < s_0 < d_j + \Delta_j$$

holds for each $d_j \in T$. That is, $T\langle s_0\rangle$ is $\vec{\Delta}$-bound as $-\Delta_j < d_j - s_0 < \Delta_j$ holds for each $d_j \in T$.

The proof of (iii). Otherwise, we assume that there exists a synopsis S such that $T' \sqcap S = \emptyset$ holds. Since $T'\langle S\rangle$ is $\vec{\Delta}$-bound,

$$-\Delta_k < d_k - \sum_{c_j \in path(c_i) \wedge s_j \in S} \delta_{ij} s_j < \Delta_k$$

holds for each $d_k \in T(c_i)$. The above equation implies that

$$t \in \bigcap_{d_k \in T(c_i)} (d_k - \Delta_k, d_k + \Delta_k)$$

for $t = \sum_{c_j \in path(c_i) \wedge s_j \in S} \delta_{ij} s_j$. Therefore, (6.1) holds for $T(c_i)$ and is contradictory to the hypothesis that $T(c_i)$ is $\underline{r}$-unbound for $\vec{\Delta}$. Thus, the result is proven. □

Similar to the F-Shift algorithm, the F_r-Shift algorithm also constructs a synopsis from d_0 up to d_{N-1} gradually onwards (lines 3).

Lines 5–17 construct $f = (2k, \underline{s}, \overline{s})$ from $f_l = (k, \underline{s_l}, \overline{s_l})$ on the left subtree and $f_r = (k, \underline{s_r}, \overline{s_r})$ on the right subtree. Lines 6–10 is the part that converts an $\underline{r}$-unbound subtree into an r-bound one through using a "fixed" value. In this case, the shift value is $b = (s_l - s_r)/2$ where $s_l = (\underline{s_l} + \overline{s_l})/2$ and $s_r = (\underline{s_r} + \overline{s_r})/2$. Refer to the value s_i in (6.2). $\overline{s}$ and $\underline{s}$ are computed from lines 8 and 9 respectively when a shift is required. Otherwise, $\overline{s}$ and $\underline{s}$ are computed from line 12 and line 13 respectively. Roughly, $(\underline{s}, \overline{s})$ is the intersection of $(\underline{s_l}, \overline{s_l})$ and $(\underline{s_r}, \overline{s_r})$ with/without a shift transformation.

With Property 6.1, it can prove that the F_r-Shift is a $(\log N)$-approximation algorithm and has $O(N)$ and $O(\log N)$ complexities on time/space respectively. In fact, the F-Shift algorithm is a special case of the F_r-Shift algorithm in which $\Delta = \Delta_i$ for $0 \le i \le N-1$.

Clearly, parallel to the S-Shift algorithm, the S_r-Shift algorithm, the optimal version of F_r-Shift, can be developed by using this paper's idea.

7 Other Applications

In this section, we will extend our methods to other applications. We first illustrate how to construct unrestricted bucket bound synopses through the Shift algorithms. We then briefly discuss the construction of multidimensional wavelet synopses on error-bound. Lastly, we discuss the extension on Haar$^+$ tree and provide the S$^+$-Shift algorithm.

7.1 Bucket Bound Synopses

Karras et al. [9] have provided a method on constructing unrestricted bucket bound synopses through finding error-bound synopses. The Shift algorithms can also be extended to construct bucket bounded synopses through calling the shift algorithms repetitively under different Δ values.

Let $S(\Delta)$ denote the synopsis that constructed from S-Shift algorithm on data vector D under error bound Δ. In order to construct bucket bounded synopsis B, we need to find $\Delta_{\max}$ and $\Delta_{\min}$ such that

$$|S(\Delta_{\max})| \leq |B| \leq |S(\Delta_{\min})|. \tag{7.1}$$

According to Property 3.3, (7.1) holds for the following $\Delta_{\max}$ and $\Delta_{\min}$:

$$\begin{cases} \Delta_{\max} \text{ is the largest value of } \{|d_i| \mid d_i \in D\}, \\ \Delta_{\min} \text{ is the } (|B|+1)\text{th-largest value of } \{|c_i| \mid c_i \in W_D\}. \end{cases}$$

We then do a binary search between $\Delta_{\max}$ and $\Delta_{\min}$ to find a Δ for synopsis B. The time cost of this approach is $O(N \log(\Delta_{\max} - \Delta_{\min}))$ by using the shift algorithms. The detailed test evaluation results of this approach are provided in Sect. 8.4.

7.2 Multidimensional Wavelet Synopses

The shift method mentioned in this paper can also be extended to construct multidimensional wavelet synopses under a given error bound. There are two common methods in building multidimensional Haar wavelet synopses, namely standard and non-standard methods. The standard method is a generalization of the one dimensional Haar wavelet construction. The non-standard method processes dimensions alternatively. To construct an error bound wavelet synopses, we use the non-standard synopses construction technique proposed in [1]. For simplicity reason, we only demonstrate how to apply the F-shift method on two dimensional wavelet synopses construction. Consider the original data as a two dimensional square array, the non-standard technique constructs a wavelet synopsis through periodically process data in a 2×2 box. The averages of data in each box form the lower-left quadrant of the original data, which will be processed recursively. Apply F-shift method, we shift the original data in each box to satisfy the 2Δ requirement, i.e. the data scope is less than Δ. Thus for each box, we compute three coefficients directly from the original data. The lower-left data of a box records the data scope of the approximation. We do this recursively on the lower-left quadrant, until the number of data in the quadrant is 1. Refer to paper [19] for details.

7.3 Haar$^+$ Synopses

Extended from Haar wavelet, Karras et al. introduced Haar$^+$ structure to improve data compression quality under general error metric L_p [8]. In a Haar$^+$ tree, as indicated in Fig. 3, the nodes are grouped in triads: the head coefficient behaves as a classical wavelet coefficient while the other two nodes, the left and right supplementary

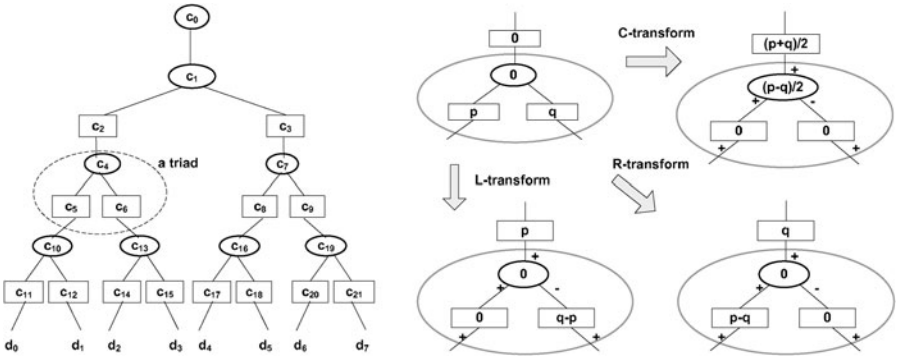


Fig. 3 Haar$^+$ tree and three transformations

coefficients, contribute their value positively to the data they affected. We called the head coefficient as e-node (ellipse node) and the other two nodes as r-node (rectangle node) as these nodes are represented by ellipse and rectangle in [8] respectively.

Because of using rectangle nodes, some properties in classical Haar tree may not properly be held in Haar$^+$ tree. For example, the average value of data in $T(c_i)$ is $\sum_{c_j \in path(c_i)} \delta_{ij} c_j$ in classical Haar trees. As a result, queries on the average data value can be more precisely answered with classical Haar synopses than with that of Haar$^+$. However, the Haar$^+$ structure generally enables higher compression quality at data sets with sharp discontinuities than that of Haar wavelet techniques [8]. For example, as indicated in Sect. 8, a Haar$^+$ synopsis can be the two-third size of a classical Haar synopsis.

In Haar$^+$ tree, let v be the reconstructed value from the root of the Haar+ tree up to the node where c_i resides and called incoming value at c_i [8], i.e.,

$$v = \sum_{s_j \in path(i) \cap S_H} \delta_{ji} s_j$$

where

$$\delta_{ij} = \begin{cases} -1 & ((i-1) \bmod 3 = 0) \wedge (d_j \text{ is on the right side of } c_i), \\ +1 & \text{otherwise.} \end{cases}$$

In this part, we extend the shift idea and propose the S$^+$-Shift algorithm for the elaborated Haar$^+$ structure. Like the S-shift algorithm, the S$^+$-Shift algorithm consists two phases, the bottom-up phase and the top-down phase, and uses three transformations on triads to eliminate the number of retained coefficients. The bottom-up phase is to generate a set of triads to retain along with their approximation ranges while the top-down phase is to retain a proper node with assigned value from a retained triad. Denoted in Fig. 3, the three transformations on triads enable to retain at most one node in a triad.[10]

[10]C-transform was used in S-shift algorithm and the algorithms of [8].

Input: D, the original data vector; Δ, the specified error bound
Output: A synopsis S_+, the set of Haar$^+$ shift coefficients that satisfies Δ bound

$S_+ \leftarrow \emptyset$; $F \leftarrow \emptyset$
define $f = (k, I)$ where k is the number of data covered
Bottom-up Phase (A): Find triads to retained;
for $(i \leftarrow 0)$ **to** $|D|$ **do**
  set $f = (1, I)$ where $I = [d_i - \Delta, d_i + \Delta]$; add f to F;
  while (there exists a $f_l = (k, I_l)$ and $f_r = (k, I_r)$ in F) **do**
    if $I_l \cap I_r = \emptyset$ **then**
      mark the triad as retained;
      let $J = \{[\frac{x_l + x_r}{2}, \frac{y_l + y_r}{2}] | \forall [x_l, y_l] \in I_l \wedge \forall [x_r, y_r] \in I_r\}$;
      set $I = I_l \cup J \cup I_r$;
    else
      set $I = I_l \cap I_r$;
    end
    replace f_l and f_r with $f = (2k, I)$ in F;
  end
end
Ensure: there is only one element $f = (n, I)$ in F
if $0 \notin I$ **then** set s be any value of I and retain this node **else** set $s = 0$;
set $v = s$;
Top-down Phase (B): Identify the retained node and assign a value for the triads obtained in Phase (A).;
foreach top-most retained triad **do**
  let v be the reconstructed value from the root of the tree up to the triad c where c is the e-node of the triad;
  let c_l and c_r be the left and right r-node of the triad respectively;
  let I_l and I_r denote ranges on c_l and c_r respectively;
  replace I_l with $\{[x_l - v, y_l - v] | \forall [x_l, y_l] \in I_l\}$; replace I_r with $\{[x_r - v, y_r - v] | \forall [x_r, y_r] \in I_r\}$;
  if $0 \in I_l$, then retain r-node c_r with a value of I_r;
  if $0 \in I_r$ then retain r-node c_l with a value of I_l;
  if $0 \notin I_l \cup I_r$, find s such that $s \in I_l$ and $-s \in I_r$; retain node c with value s;
end
output S_+, the set of retained nodes with assigned values

Algorithm 4: S$^+$-Shift (D,Δ): Haar$^+$-based shift algorithm

The idea behind S$^+$-Shift is to move the potentially retained values upwards so that they can be amortized by more data items. As specified in Sect. 3.3, an $\underline{s}$-unbound Haar tree can be converted into s-bound with a C-transform. This conversion could also be achieved by using either L-transform or R-transform in a Haar$^+$ tree. Suppose that the left and the right r-node in the triad have values p and q respectively. Intuitively, L-transform means to retain the right r-node with $q - p$ value under the input

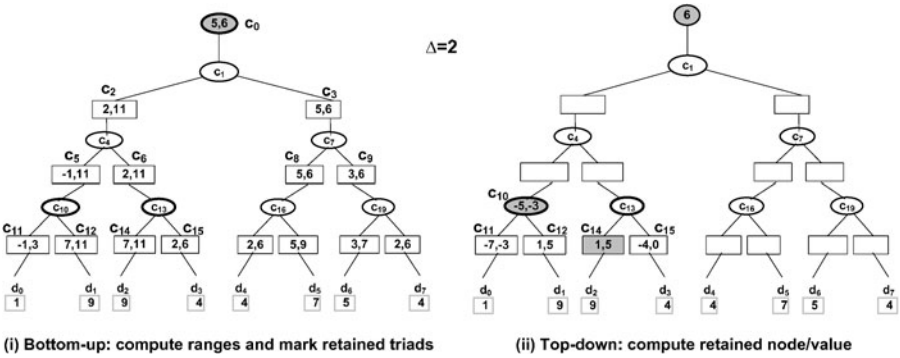


Fig. 4 Example on generating Haar$^+$ synopses

of p from the parent triad. Similarly, R-transform means to retain the left r-node with $p - q$ value under the input of q.

In the bottom-up phase, the S$^+$-Shift algorithm constructs approximation ranges from d_0 up to d_{N-1} gradually onwards: it marks the retained triads and keeps the approximation ranges at the r-nodes. Analogous to the $\underline{s}$-unbound situations in S-Shift, a triad needs to be retained if the left (I_l) and right (I_r) ranges does not share a common elements (lines 7–11). A node in this triad needs to be retained for any incoming value v of $I_l \cup J \cup I_r$ (line 10). Otherwise, this triad does not need to be retained for any incoming value v of $I_l \cap I_r$ (line 12).

In the top-down phase, the S$^+$-Shift algorithm decides a node to retain in a retained triad with a incoming value v of $I_l \cup J \cup I_r$ (lines 21–29). Intuitively, this part operates as follows: If $v \in I_l$, the right r-node c_r is retained with a value of $v_r - v$ for $v_r \in I_r$; If $v \in I_r$, the left r-node c_l is retained with a value of $v_l - v$ for $v_l \in I_l$; Otherwise, the e-node c is retained with a value of s satisfying $v + s \in I_l$ and $v - s \in I_r$. In the following, we use an example to illustrate the S$^+$-Shift algorithm.

Example 7.1 Let us consider the data set $D = \{1, 9, 9, 4, 4, 7, 5, 4\}$ and $\Delta = 2$. The constructed ranges in the bottom-up phase are depicted in the r-nodes in Fig. 4(i). For instance, with $I_{11} = [-1, 3]$ and $I_{12} = [7, 11]$ (line 5), $I_5 = I_{11} \cup J \cup I_{12}$ is derived for $J = [3, 7]$ as $I_{11} \cap I_{12} = \emptyset$ (lines 7–11). The triad headed at c_{10} is marked as retained triad (line 8). In the situation of deriving I_3 from I_8 and I_9, I_3 is set to be $I_8 \cap I_9$ (line 12). Retained triads are denoted by thick-line node. Since the range at c_0 is $[5, 6]$, $s_0 = 6$ can be retained.

The top-down phase is illustrated in Fig. 4(ii). With $v = 6$, the S$^+$-Shift algorithm computes the retained node and value from the top-most unprocessed retained triads, i.e., the triads headed at c_{10} or c_{13}. In the triad headed at c_{10}, the algorithm updates I_{11} and I_{12} into $I'_{11} = [-7, -3]$ and $I'_{12} = [1, 5]$ (line 25) for $v = 6$. Since $0 \notin I'_{11} \cup I'_{12}$ and $I'_{11} \cap [-5, -1] = [-5, -3]$, set s_{10}, the retained value at c_{10}, to be a value of $[-5, -3]$ (line 28). Similarly, for the triad headed at c_{13}, the algorithm updates $I_{14} = [7, 11]$ and $I_{15} = [2, 6]$ into $I'_{14} = [1, 5]$ and $I'_{15} = [-4, 0]$ (line 25). Since $0 \in I'_{15}$, set the retained value s_{14} to be a value of $[1, 5]$. Thus, $S_+ = \{s_0 = 6; s_{10} = -4, s_{14} = 3\}$, the set of shaded nodes, is a Haar$^+$ synopsis.

Roughly, the number of intervals in an r-node at level $(\log N - h)$ is bounded by a^h where $a \geq 3$ is a constant value and $0 \leq h \leq \log N$. Since the space of S$^+$-Shift algorithm is bounded by

$$N + \frac{N}{2}a + \frac{N}{2^2}a^2 + \cdots + \frac{N}{2^{\log N}}a^{\log N},$$

the space complexity of S$^+$-Shift algorithm is bounded by $O(N^2)$. Similarly, since sorting and merging a^h number of intervals requires $O(a^h h)$, the time cost of S$^+$-Shift algorithm is bounded by

$$N + \frac{N}{2}a + \frac{2N}{2^2}a^2 + \cdots + \frac{hN}{2^h}a^h + \cdots + \frac{N\log N}{2^{\log N}}a^{\log N}.$$

Thus, the time complexity of S$^+$-Shift algorithm is bounded by $O(N^2 \log N)$.

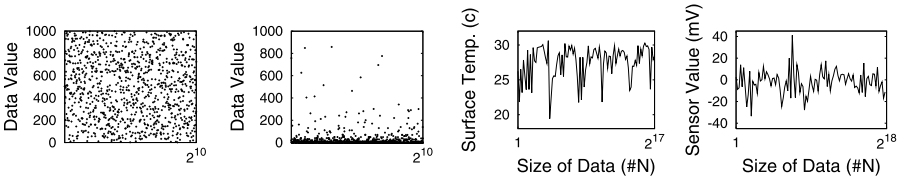
As the S$^+$-Shift algorithm is very similar to the S-Shift algorithm, we shall leave detailed analyses and the optimization of this algorithm in our future work. Experimental tests on S$^+$-Shift are provided in the next section.

8 Experimental Evaluation

In this section, we present some of our experimental results on the Shift algorithms to illustrate their compression quality and time efficiency. We compare the results of the F-Shift algorithm (F), the S-Shift algorithm (S) and the S$^+$-Shift algorithm (S^+) to those achieved with the restricted error bound synopses construction algorithm (R) in [12], the MinHaarSpace algorithm (K) in [9], the Haar+ algorithm (K^+) in [8] and the $(\log N)$-approximation algorithm (G) in [5]. For the K and K$^+$ algorithms, we use the source code that borrowed from the authors of [9] and [8]. Table 3 summarizes the various algorithms' complexity and their specialties. All the algorithms in this article are implemented in GNU C++ and all the experiments are performed on an Intel Xeon 3.0 GHz Linux box with 2 GB memory.

Table 3 Summary of various wavelet synopses

	Ref.	Time	Space	Spec.
Space Bound	IK [9]	$O(R^2 n(\log \varepsilon^* + \log n))$	$O(R \log n + n)$	Indirect Unrestrict
	G [5]	$O(n)$	$O(B + \log n)$	Approx. Unrestrict
Error Bound	R [12]	$O(n^2/\log n)$	$O(n^2)$	Optimal Restrict
	K, K$^+$ [8, 9]	$O((\frac{\varepsilon}{\delta})^2 n)$	$O(\frac{\varepsilon}{\delta}\log n + n)$	Approx. Unrestrict
	G [5]	$O(n)$	$O(\log n)$	Approx. Unrestrict
	F, S [14]	$O(n)$	$O(\log n)$, $O(n)$	Approx. Unrestrict



(a) Uni (b) Zipf (c) TEM (d) EEG

Fig. 5 Distributions of various data sets

Experiment Bed[11] The experiments are conducted on two synthetic data sets and two real life data sets. The two synthetic data sets, as depicted in Figs. 5(a) and (b), are *Uni* and *Zipf* respectively with their values distributed in [0, 1000]. Uni contains values following uniform distribution. Zipf contains values following zipf distribution with a zipf parameter of 2.0, i.e. a highly skewed distribution. The two real data sets used in this section are acquired from UCI KDD Archive [17]: *TEM* and *EEG* as depicted in Figs. 5(c) and (d) respectively. *TEM* is sea surface temperatures extracted from the El Nino Data set, taken from a series of buoys positioned throughout the equatorial Pacific. This data set is also used in [9] for its experiments. *EEG* includes values, in microvolt (mV), acquired from a large study to examine EEG correlates of genetic predisposition to alcoholism.

8.1 Compression Quality

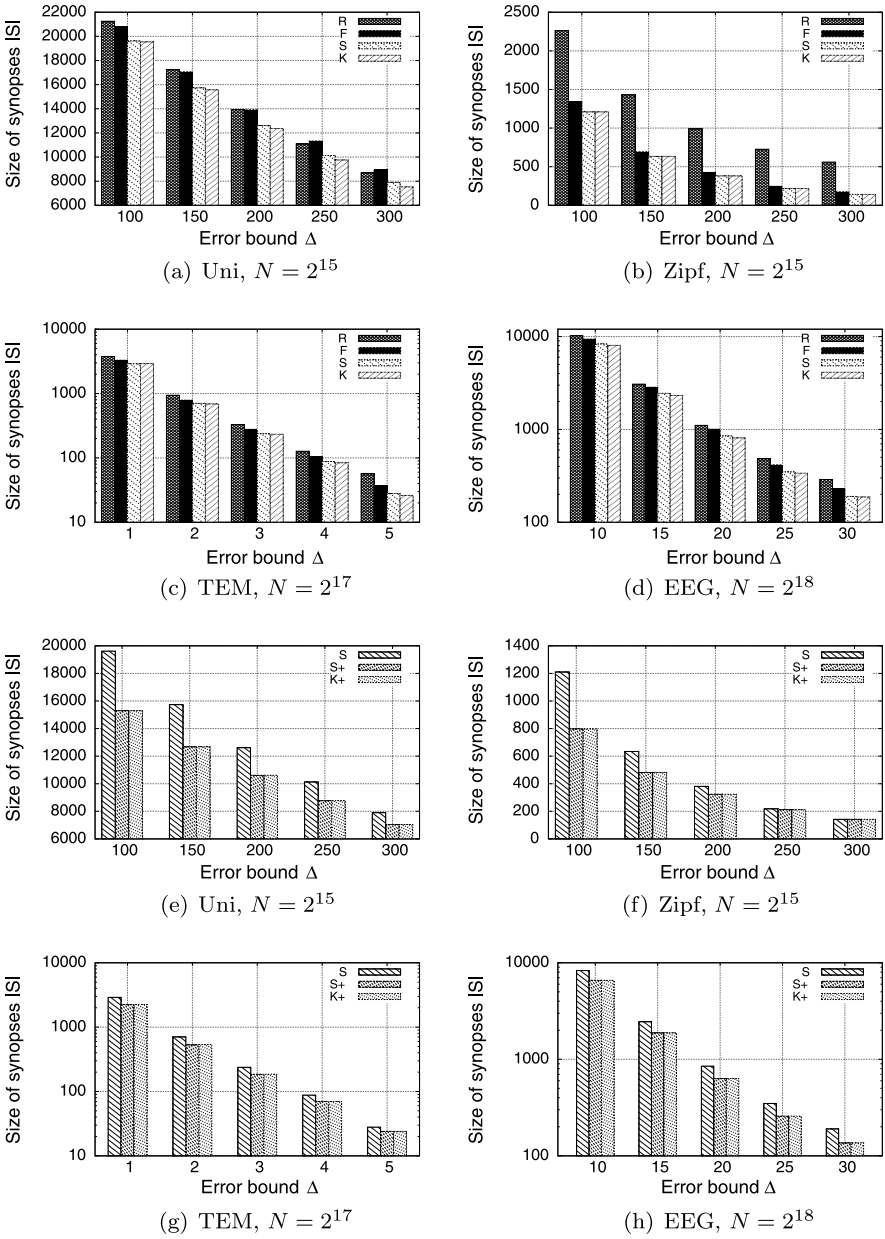
Since the size of synopses reflect the compression quality, we shall construct error bound synopses for each of the algorithms F, S, and R and compare these obtained synopses against the optimal one in this subsection. As stated in [9] that a smaller δ in K algorithm leads to a smaller sized synopsis under a sacrifice of construction time, we shall use the synopses generated from K algorithm under smaller δ as the substitutes of optimal synopses. With this intention in mind, we always set $\Delta/\delta = 1000$ in both K and K^+ algorithms for each considered data set and assume these generated synopses are quite close to the optimal in the experiments. The average time on construction synopses by K is between 1 to 2 hours under this situation.

Figure 6 presents the experiment results on the four data sets. As indicated in Figs. 6(a)–(d) are for Haar synopses. It shows that F and S achieve better compression quality than that of R. S performs quite well and is very close to K. Figures 6(e)–(h) are for Haar$^+$ synopses. It indicates that S^+ always performs better than S, in terms of Haar$^+$ synopsis sizes. Interestingly, S^+ also achieves same or better compression quality than that of K^+.

8.2 Time Efficiency

Unlike the running time of F, S, S^+ and R that solely depends on N, the running time of K and K^+ is also heavily related to $(\Delta/\delta)^2$: A big δ results in a quick construction

[11] We also generate another moderate skew data set with values following normal distribution in our experiments. The test results on this data set lies between that of Uni and Zipf. Due to space limitation, we will not report the details in this version of this paper.



(a) Uni, $N = 2^{15}$ (b) Zipf, $N = 2^{15}$

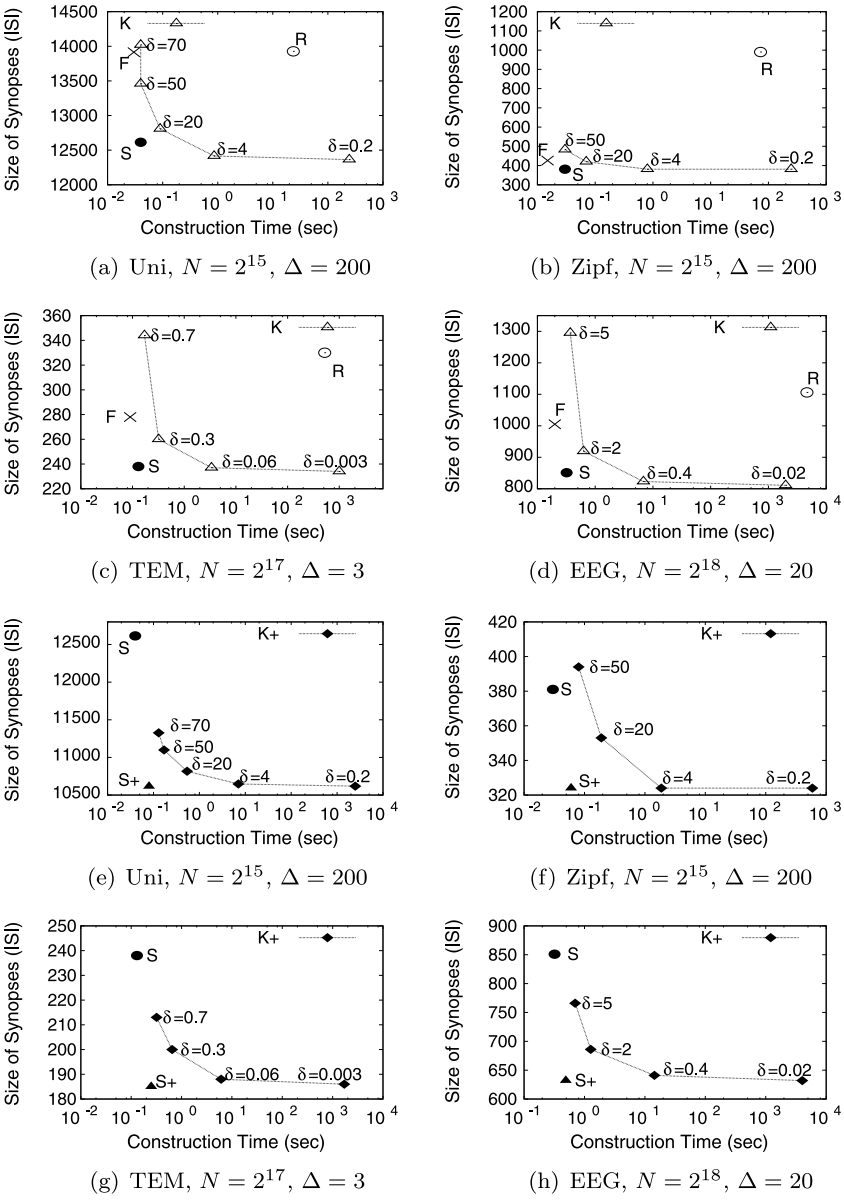
(c) TEM, $N = 2^{17}$ (d) EEG, $N = 2^{18}$

(e) Uni, $N = 2^{15}$ (f) Zipf, $N = 2^{15}$

(g) TEM, $N = 2^{17}$ (h) EEG, $N = 2^{18}$

Fig. 6 Sizes of synopses on various data sets

with poor compression quality while a small δ results in a slow construction with a good compression quality. In this subsection, we evaluate the synopses construction time under a fixed error bound (Δ) and various δ values on K and K^+. From the following discussion, we conclude that the S-Shift algorithms can be 1–7 orders of



(a) Uni, $N = 2^{15}$, $\Delta = 200$ (b) Zipf, $N = 2^{15}$, $\Delta = 200$

(c) TEM, $N = 2^{17}$, $\Delta = 3$ (d) EEG, $N = 2^{18}$, $\Delta = 20$

(e) Uni, $N = 2^{15}$, $\Delta = 200$ (f) Zipf, $N = 2^{15}$, $\Delta = 200$

(g) TEM, $N = 2^{17}$, $\Delta = 3$ (h) EEG, $N = 2^{18}$, $\Delta = 20$

Fig. 7 Time cost of synopses construction on various data sets

magnitudes faster than K algorithm when constructs a similar sized Haar synopsis and that the S^+-Shift algorithms can be 1–4 orders of magnitudes faster than K^+ algorithm when constructs a similar sized Haar$^+$ synopsis.

Figure 7 shows the synopses construction time on the four data sets. As indicated in Figs. 7(a)–(d), K demonstrates a trade-off between construction time and synopses sizes. Table 4 lists the construction time ratios with K upon constructing

Table 4 Synopses construction time comparison

	Uni	Zipf	TEM	EEG
R/K	341	–	625	4367
F/K	0.12	3×10^{-3}	5×10^{-4}	1×10^{-3}
S/K	0.01	1×10^{-7}	4×10^{-4}	7×10^{-4}

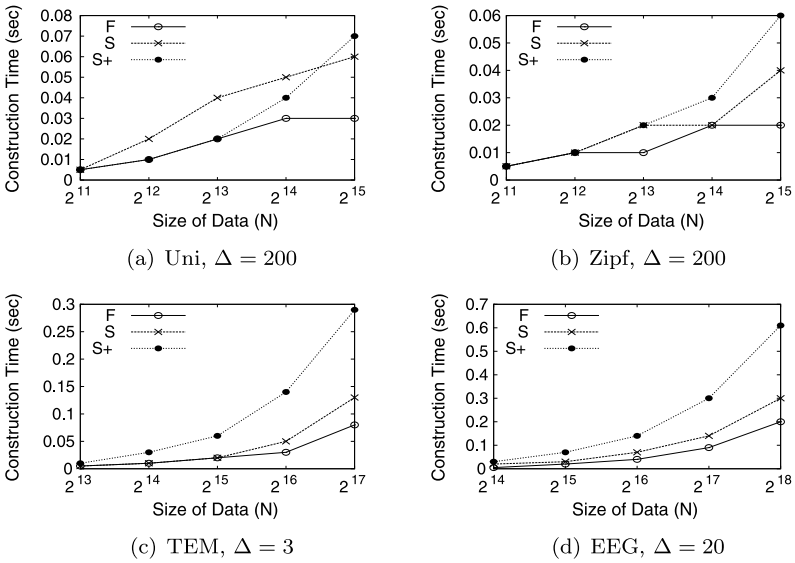


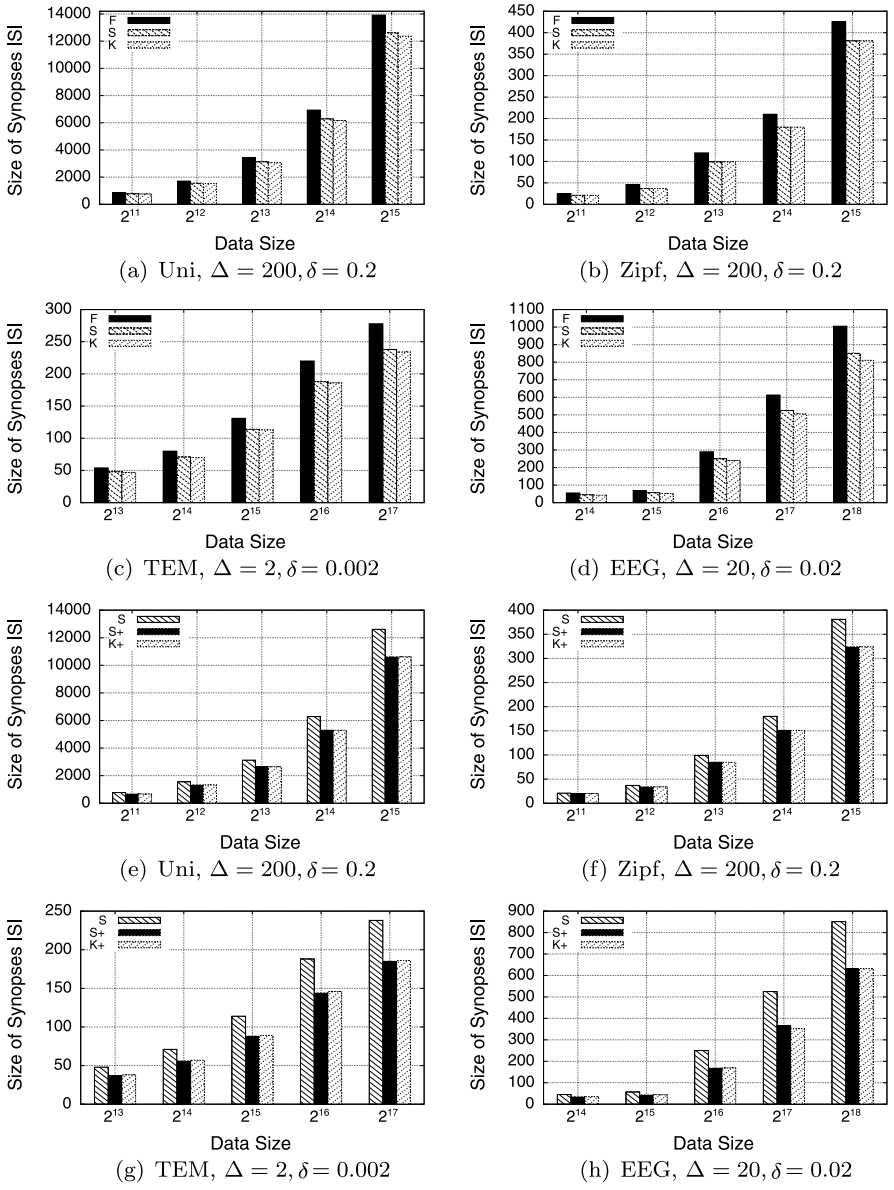
Fig. 8 Scalability on synopses construction

similar sized synopses. It indicates that R is far more slower than K on constructing a similar quality synopsis. The shift algorithms, F and S both perform better than K under a smaller δ. That is, when constructs a similar quality synopsis, F improves the synopses construction time 1–4 orders of magnitudes. S improves the synopses construction time 2–7 orders of magnitudes. This can be explained from $O((\frac{\Delta}{\delta})^2 N)$ time complexity on smaller δ. It is also interesting to note that the performances of F and S are especially well when constructing synopses on data sets with skewed distribution. Figures 7(e)–(h) depict the comparisons on S, S^+ and K^+. It shows that S^+ can be 1–4 orders of magnitudes faster than K^+ algorithm.

8.3 Scalability

To evaluate the time and compression quality performance trends of F and S on large data sets, the scalability test is twofold: (A) evaluating the synopses construction time on various data sizes and (B) assessing the compression quality (synopses size) on various data sizes. For the K in (B), we set small δ values as we did in Sect. 8.1 to generate near optimal results. We omit the tests of R in this part.

Figure 8 presents the results for (A). It shows that the running time of F and S scales well with larger data sizes. In these tests, the three shift algorithms can generate



(a) Uni, $\Delta = 200, \delta = 0.2$ (b) Zipf, $\Delta = 200, \delta = 0.2$

(c) TEM, $\Delta = 2, \delta = 0.002$ (d) EEG, $\Delta = 20, \delta = 0.02$

(e) Uni, $\Delta = 200, \delta = 0.2$ (f) Zipf, $\Delta = 200, \delta = 0.2$

(g) TEM, $\Delta = 2, \delta = 0.002$ (h) EEG, $\Delta = 20, \delta = 0.02$

Fig. 9 Scalability on compression quality

arbitrary error-bound synopses in less than 1 second, even for large data sets with up to 2^{18} values.

Figure 9 presents the results for (B). It shows that the shift algorithms have good compression quality on various data sizes. The S^+ performs especially well and beat K^+ in all our experiment results.

8.4 Test on Bucket Bound

In this subsection, we will demonstrate the compression quality and efficiency of our approach on the generation of Bucket Bound Synopses.

As discussed in Sect. 7, the Shift algorithms can be used indirectly to construct unrestricted bucket bound synopses. Let us denote the indirect algorithm based on F-shift, S-shift and S^+-shift as IF, IS and IS^+ respectively. We also denote the IndirectHaar(B) algorithm of [9] as IK and the bucket bound algorithm of [8] as IK^+. In the comparisons, we include the greedy algorithm, termed as G, from Guha et al. [5]. As indicated in [5], G has linear time complexity $O(N)$ and $O(B + \log N)$ space complexity with $O(\log N)$ approximation ratio.

Figure 10 shows the time costs when constructing the synopses of size 300 on four data sets. It indicates that IS and IS^+ can be 1–3 orders of magnitudes faster than K and K^+ respectively to achieve a similar maximum error bound. It also shows that G is slightly faster than IF but with a larger maximum error bound. The compression quality of G is further reported in Fig. 11. Consistent with the comparison results for multidimensional synopses [19], these results indicate that the Shift algorithms achieve better quality of synopses (i.e., smaller in size or in errors) compared to those indirectly constructed by G.

8.5 Workability

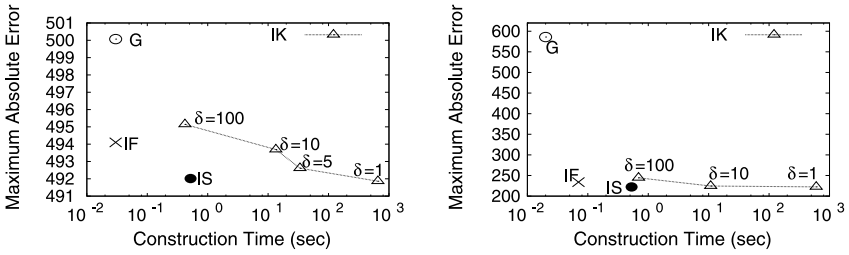
Our last test focuses on the workability of our two shift-based algorithms. As mentioned early, one application of our techniques is to compress the time series data acquired through clinical operations. The generated synopses will be used to reconstruct the original data values under some error bounds. We use the synopses generated by S to reconstruct the original data. Specifically, the synopses reconstruct the EEG data set. Figure 12 shows the reconstructed approximation data of the EEG data set where the two synopses have error bounds 5 and 20 respectively. Obviously, the two reconstructed data set are quite similar to the original data set, while the synopses size is only 10% to 0.6% of the original data size.

8.6 Summary

In general, all the experiments indicate that the Shift algorithms construct small size of synopses for both synthetic and real data sets. Both the F-Shift and S-Shift algorithms construct synopses in less than 0.1 second, even when the size of original data reaches up to 2^{18}. The S^+-Shift is generally 4000 times faster than that of KS^+ algorithm on generating similar sized $Haar^+$ synopses. This demonstrates that our algorithms are efficient and practically capable for large data sets.

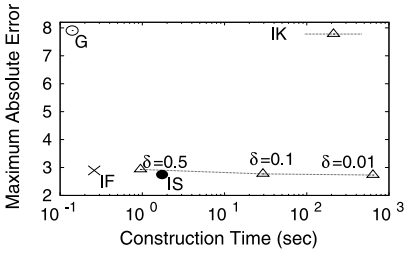
9 Conclusion

In this paper, we have proposed Shift algorithms for constructing the unrestricted error bound synopses. We indicate our methods can be extended to other findings

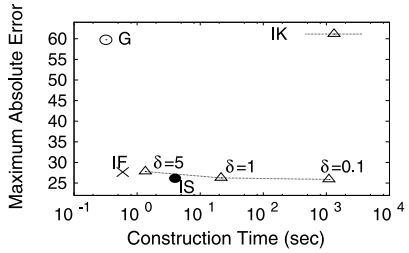


(a) Uni, $N = 2^{15}$, $B = 300$

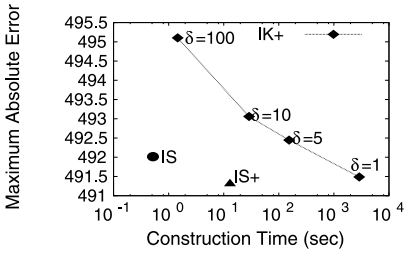
(b) Zipf, $N = 2^{15}$, $B = 300$



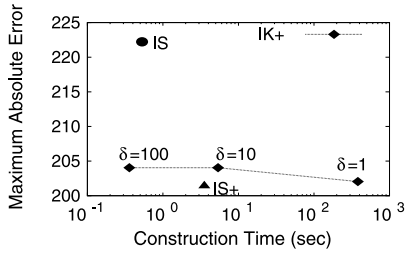
(c) TEM, $N = 2^{17}$, $B = 300$



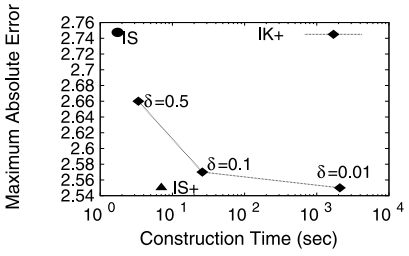
(d) EEG, $N = 2^{18}$, $B = 300$



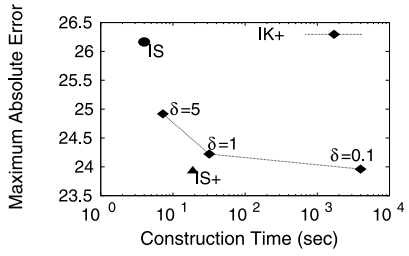
(e) Uni, $N = 2^{15}$, $B = 300$



(f) Zipf, $N = 2^{15}$, $B = 300$



(g) TEM, $N = 2^{17}$, $B = 300$

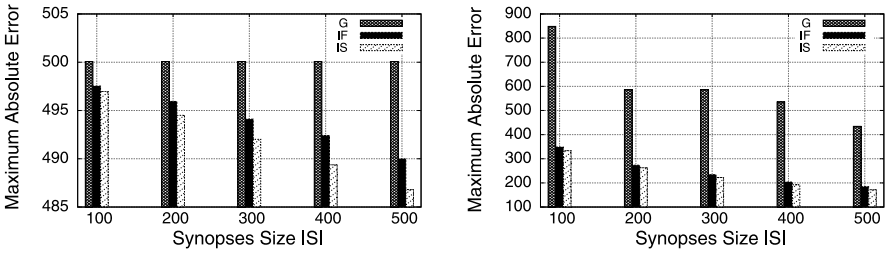


(h) EEG, $N = 2^{18}$, $B = 300$

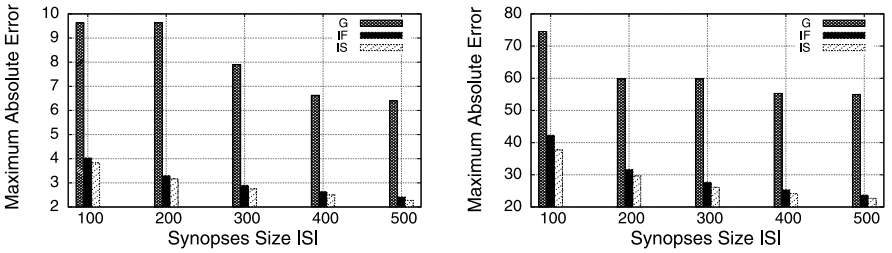
Fig. 10 Time cost of synopses construction on a bucket bound

such as the construction of bucket bound synopses and the synopses on Haar$^+$ structure. The algorithms are highly practical, cheap to run and generate smaller-sized synopses.

However, the paper leaves many attractive open problems to be answered in the future work. There exist gaps between F-Shift and S-Shift algorithms: If better com-



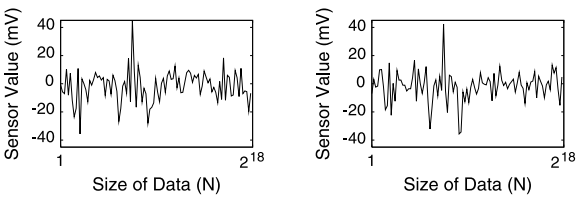
(a) Uni, $N = 2^{15}$ (b) Zipf, $N = 2^{15}$



(c) TEM, $N = 2^{17}$ (d) EEG, $N = 2^{18}$

Fig. 11 Synopses on various data sizes and errors

Fig. 12 Approximation of the EEG data set



(a) $\Delta = 5$, $|S| = 28168$ (b) $\Delta = 20$,$|S| = 1690$

pressing outcomes can be achieved through using other fixed values rather than the ones that used in F-Shift? More importantly, it is desirable to find a more efficient algorithm on Haar$^+$ through combining the techniques of S$^+$-Shift and the Haar$^+$ algorithm of [8]. Our future work will consider these problems.

Acknowledgements The authors would like to thank Panagiotis Karras for his generous giving us his source code for the comparison tests in this paper. The authors are also grateful to Sudipto Guha, Panagiotis Karras and the anonymous referees for their helpful comments. The work reported in this article is partially supported by an ARC research grant DP0987557.

References

1. Chakrabarti, K., Garofalakis, M., Rastogi, R., Shim, K.: Approximate query processing using wavelets. VLDB J. **10**(2–3), 199–223 (2001)
2. Garofalakis, M., Gibbons, P.B.: Probabilistic wavelet synopses. ACM Trans. Database Syst. **29**(1), 43–90 (2004). doi:10.1145/974750.974753

3. Guha, S.: Space efficiency in synopsis construction algorithms. In: VLDB '05: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 409–420. ACM, New York (2005)
4. Guha, S., Harb, B.: Wavelet synopsis for data streams: minimizing non-Euclidean error. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD '05, pp. 88–97. ACM, New York (2005). doi:10.1145/1081870.1081884
5. Guha, S., Harb, B.: Approximation algorithms for wavelet transform coding of data streams. IEEE Trans. Inf. Theory **54**(2), 811–830 (2008). doi:10.1109/TIT.2007.913569
6. Guha, S., Shim, K., Woo, J.: Rehist: relative error histogram construction algorithms. In: VLDB '04: Proceedings of the Thirtieth International Conference on Very Large Data Bases, pp. 300–311. Morgan Kaufmann, San Mateo (2004)
7. Karras, P., Mamoulis, N.: One-pass wavelet synopses for maximum-error metrics. In: VLDB '05: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 421–432. ACM, New York (2005)
8. Karras, P., Mamoulis, N.: Hierarchical synopses with optimal error guarantees. ACM Trans. Database Syst. **33**(3), 1–53 (2008). doi:10.1145/1386118.1386124
9. Karras, P., Sacharidis, D., Mamoulis, N.: Exploiting duality in summarization with deterministic guarantees. In: KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 380–389. ACM, New York (2007). doi:10.1145/1281192.1281235
10. Matias, Y., Urieli, D.: Optimal workload-based weighted wavelet synopses. In: Proceedings of International Conference on Database Theory (ICDT), pp. 368–382 (2005)
11. Matias, Y., Vitter, J.S., Wang, M.: Wavelet-based histograms for selectivity estimation. SIGMOD Rec. **27**(2), 448–459 (1998). doi:10.1145/276305.276344
12. Muthukrishnan, S.: Subquadratic algorithms for workload-aware Haar wavelet synopses. In: Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), pp. 285–296 (2005)
13. Pang, C., Zhang, Q., Hansen, D., Maeder, A.: Building data synopses within a known maximum error bound. In: APWeb/WAIM'07: Proceedings of the Joint 9th Asia-Pacific Web and 8th International Conference on Web-Age Information Management Conference on Advances in Data and Web Management, pp. 463–470. Springer, Berlin (2007)
14. Pang, C., Zhang, Q., Hansen, D., Maeder, A.: Unrestricted wavelet synopses under maximum error bound. In: EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology, pp. 732–743. ACM, New York (2009). doi:10.1145/1516360.1516445
15. Reiss, F., Garofalakis, M., Hellerstein, J.M.: Compact histograms for hierarchical identifiers. In: Proceedings of the 32nd International Conference on Very Large Data Bases, VLDB '06, pp. 870–881. ACM, New York (2006). http://portal.acm.org/citation.cfm?id=1182635.1164202
16. Stollnitz, E.J., Derose, T.D., Salesin, D.H.: Wavelets for Computer Graphics: Theory and Applications. Morgan Kaufmann, San Francisco (1996)
17. UCI KDD archive. http://kdd.ics.uci.edu
18. Vitter, J.S., Wang, M., Iyer, B.: Data cube approximation and histograms via wavelets. In: CIKM '98: Proceedings of the Seventh International Conference on Information and Knowledge Management, pp. 96–104. ACM, New York (1998). doi:10.1145/288627.288645
19. Zhang, Q., Pang, C., Hansen, D.: On multidimensional wavelet synopses for maximum error bounds. In: DASFAA '09: Proceedings of the 14th International Conference on Database Systems for Advanced Applications, pp. 646–661 (2009)