Online Appendix to: Hierarchical Synopses with Optimal Error Guarantees

PANAGIOTIS KARRAS
National University of Singapore
and
NIKOS MAMOULIS
University of Hong Kong

A. AN OPTIMAL SOLUTION TO THE ERROR-BOUNDED HAAR$^+$ PROBLEM

In this Appendix, we examine how the ideas of Section 6.1 apply to the general case of a Haar$^+$ tree. The error-bounded Haar$^+$ problem we consider in this case is defined as follows:

Problem A.1. Given a data vector $\mathbf{D}$ and an error bound ϵ for a (weighted) maximum-error metric $\mathcal{L}_\infty^w$, construct a Haar$^+$ representation $\mathbf{H}$ of $\mathbf{D}$ that produces an approximation $\hat{\mathbf{D}}$, such that $\mathcal{L}_\infty^w(\|\mathbf{D} - \hat{\mathbf{D}}\|) \leq \epsilon$ and the number of occupied nodes B^* in $\hat{\mathbf{D}}$ is minimized.

As in Section 6.1, we are interested in the behavior of an $S(i, v)$ function: the minimum space budget needed by a Haar$^+$ triad C_i and its descendants in order to satisfy a $\mathcal{L}_\infty^w$-error bound ϵ with incoming value v at c_i. For a given i, $S(i, v)$ is defined for every $v \in \mathbb{R}$ and takes values in $\mathbb{N}$. In Theorem 4.2 we have shown that each triad in a Haar$^+$ representation $\mathbf{H}$ needs to contain at most one non-zero coefficient. We have used this property in order facilitate our computation of $E(i, v, b)$ in Section 5.2. Still, now we relax our observation of this property; it is not useful to us in this case; this relaxation allows us to delimit the value set of $S(i, v)$, as the following theorem shows.

THEOREM A.2. *Let s_i^* be the minimum value of $S(i, v)$, for any v, on a triad C of a Haar$^+$ tree representation $\mathbf{H}$, $v \in \mathbb{R}$. Then, $\forall v$, $S(i, v) \in \{s_i^*,\ s_i^* + 1,\ s_i^* + 2\}$.*

PROOF. Let $\tilde{v}$ be an incoming value with which the minimum of $S(i, v)$ is obtained: $\forall v, S(i, v) \geq S(i, \tilde{v}) = s_i^*$. Let p, q, r be the optimal values assigned to the head, left and right supplementary coefficients of C, respectively, with incoming value $\tilde{v}$, resulting in the state $C = [\tilde{v}, p, q, r]$ and the contribution vector $[\tilde{v} + p + q, \tilde{v} - p + r]$. For any other incoming value v', we may adjust the values assigned at the two supplementary coefficients in C so as to produce the same contribution vector (i.e., incoming values to the left and right subtrees of C). Specifically, the state $C = [v', p, q + \tilde{v} - v', r + \tilde{v} - v']$ produces the same contribution vector $[\tilde{v} + p + q, \tilde{v} - p + r]$. Hence the solution thereafter can be maintained as with incoming value $\tilde{v}$. The value adjustment increases the

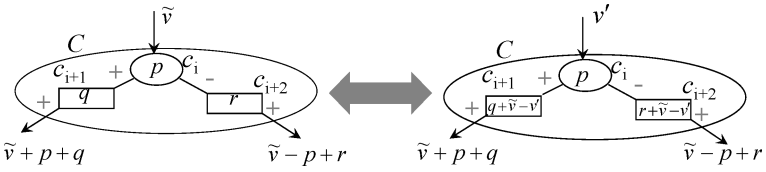

Fig. 1. Adjustment operation in Theorem A.2.

number of non-zero terms in C and its descendants by at most 2. Specifically, the number of non-zero terms is increased by 1 if exactly one of the performed value adjustments turns a zero value to a non-zero one, while it is increased by 2 in case both adjusted values turn from zero to non-zero. Hence, $S(i, v') \leq S(i, \tilde{v}) + 2 = s_i^* + 2$. In effect, $\forall v, S(i, v) \in \{s_i^*,\ s_i^* + 1,\ s_i^* + 2\}$. □

Figure 1 depicts the adjustment operation used in Theorem A.2; the right-side triad with incoming value v' in the figure produces the same contribution vector as the left-side one with the minimal-space incoming value $\tilde{v}$.

Theorem A.2 implies that all possible incoming values $v \in \mathbb{R}$ to a triad C_i can be grouped in three sets: (i) the set of values with which the minimum space $S(i, v) = s_i^*$ is achieved, (ii) the set of those values for which $S(i, v) = s_i^* + 1$, and (iii), the rest, worst-case values, for which $S(i, v) = s_i^* + 2$. We can express each of these sets as a union of intervals of $\mathbb{R}$.

In our value adjustment operations, we have not affected the head coefficient (c_i in Figure 1). In fact, Lemma 4.1 implies that, when computing the value of $S(i, v)$, five options are available in each triad C_i: either all coefficients are left unoccupied (i.e., zero-value), or the head, left or right supplementary coefficient is occupied (i.e., nonzero-value), or both supplementary coefficients are occupied. For a given incoming value v, the option with which the minimum total space is required in C_i and its descendants is adopted as the value of $S(i, v)$. We now look at this computation more closely.

At the bottom-most Haar$^+$ tree level, the values of $S(i, v)$ are directly computed from the affected data. Since our purpose is to minimize a general-case, weighted maximum-error metric $\mathcal{L}_\infty^w$, each data item d_i with associated error weight w_i defines a tolerance interval $[d_i - \frac{\epsilon}{w_i}, d_i + \frac{\epsilon}{w_i}]$; approximation values within this interval satisfy the error bound ϵ for d_i. A bottom-level Haar$^+$ tree triad C_i approximates two data values. These two values define two tolerance intervals. Let the tolerance interval for the value at the left(right) leaf of C_i be $[a, b]$ ($[c, d]$). If $[a, b] \cap [c, d] \neq \emptyset$, then the minimum value of $S(i, v)$ is 0, obtained for $v \in [a, b] \cap [c, d]$; such an incoming value itself satisfies the error bound ϵ for both approximated data values. Otherwise, the minimum value of $S(i, v)$ is 1, obtained, for example, when $v \in [a, b]$ and a non-zero value in $z \in [c - v, d - v]$ is assigned to the right supplementary coefficient. Besides, $S(i, v) \leq 2$, since any incoming value can be accommodated by using two supplementary coefficients for the two approximated data values. We examine the cases in which $S(i, v) = 1$ in more detail. The value 1 is assumed by $S(i, v)$ when (i) a single non-zero supplementary coefficient suffices to satisfy the error bound ϵ on both data values at the leaves of C_i, and when (ii) a stand-alone (as Lemma 4.1

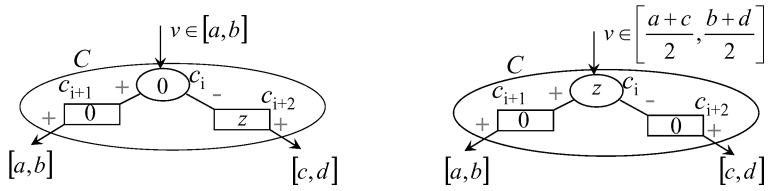

Fig. 2. Two cases in which $S(i, v) = 1$ in a bottom-level triad C.

prescribes), nonzero head coefficient is sufficient for that purpose. We analyze these cases.

—The former case (nonzero *supplementary* coefficient) occurs if and only if either $v \in [a, b]$ or $v \in [c, d]$, while $v \notin [a, b] \cap [c, d]$; that is, more succinctly, when $v \in [a, b] \triangle [c, d]$, where $A \triangle B$ is the *symmetric difference* of A and B, $A \triangle B = A \cup B - A \cap B = (A - B) \cup (B - A)$. If $v \in [a, b]$, then any value $z \in [c - v, d - v]$ suffices to be assigned to the right supplementary coefficient. Otherwise, if $v \in [c, d]$, then any value $z \in [a - v, b - v]$ should be assigned to the left supplementary coefficient.

—The latter case (nonzero *head* coefficient) occurs if and only if the given incoming value v allows for an assigned value z at the head coefficient that satisfies the given error bound in both data leafs of C_i. This requirement translates to:

$$\left.\begin{array}{c} a \le v + z \le b \\ \wedge \\ c \le v - z \le d \end{array}\right\} \Leftrightarrow \left\{\begin{array}{c} \frac{a+c}{2} \le v \le \frac{b+d}{2} \\ \wedge \\ \max\{a - v, v - d\} \le z \le \min\{b - v, v - c\} \end{array}\right. \quad (1)$$

Naturally, if $[a, b] \cap [c, d] \neq \emptyset$, then it follows that $M = [\frac{a+c}{2}, \frac{b+d}{2}] \cap [a, b] \cap [c, d] \neq \emptyset$. Incoming values $v \in M$ give $S(i, v) = 0$, as explained before, hence should not be included in the present case of $S(i, v) = 1$. For convenience in notation, we define[1] $[a, b] \diamond [c, d] = [\frac{a+c}{2}, \frac{b+d}{2}] - [a, b] \cap [c, d]$. Hence, more succinctly, the case that a stand-alone nonzero head coefficient suffices to satisfy the error bound ϵ on both data values at the leaves of C_i occurs when $v \in [a, b] \diamond [c, d]$. Appropriate values of z are given in Equation (1).

Figure 2 presents the two cases in which $S(i, v) = 1$ on a bottom-level triad C_i with incoming value v; the left side of the figure depicts a case where a single nonzero (right) supplementary coefficient is assigned at C_i; the right side shows the case where a stand-alone nonzero head coefficient is assigned.

Summing up, $S(i, v)$ at a bottom-most-level triad C_i is defined as:

$$S(i, v) = \left\{\begin{array}{ll} 0, & v \in [a, b] \cap [c, d] \\ 1, & v \in [a, b] \triangle [c, d] \cup [a, b] \diamond [c, d] \\ 2, & v \notin [a, b] \cup [c, d] \cup [a, b] \diamond [c, d] \end{array}\right. \quad (2)$$

In order to represent the full value range of $S(i, v)$, we only need to store two sets: (i) the set $\mathbf{P}_i = [a, b] \cap [c, d]$, possibly empty, such that $v \in \mathbf{P}_i \Leftrightarrow S(i, v) = 0$,

[1]In fact, if $[a, b] \cap [c, d] \neq \emptyset$, then $[a, b] \diamond [c, d] \subseteq [a, b] \triangle [c, d]$.

and (ii) the union $\mathbf{Q}_i = [a,b] \triangle [c,d] \cup [a,b] \diamond [c,d] = [a,b] \cup [c,d] \cup [\frac{a+c}{2}, \frac{b+d}{2}] - [a,b] \cap [c,d]$, that is, a union of two or three distinct v-value intervals, such that $v \in \mathbf{Q}_i \Leftrightarrow S(i,v) = 1$. For the rest values of v it is inferred that $S(i,v) = 2$.

Thereafter, the computation proceeds in a recursive process, in which $S(i,v)$ at a triad C_i is defined from its values in the children triads of C_i. Specifically, let C_{i_L} be the left child triad of C_i and C_{i_R} be its right child triad. The function S applies on C_{i_L} for any incoming value $v \in \mathbb{R}$ and assumes a minimum value $s^*_{i_L} | \forall v \mathbb{R}, S(i_L, v) \geq s^*_{i_L}$. Similarly, applied on C_{i_R}, S assumes the minimum value $s^*_{i_R} | \forall v \mathbb{R}, S(i_R, v) \geq s^*_{i_R}$. Then, according to Theorem A.2, $S(i_L, v) \in \{s^*_{i_L},\ s^*_{i_L} + 1, s^*_{i_L} + 2\}$ and $S(i_R, v) \in \{s^*_{i_R},\ s^*_{i_R} + 1, s^*_{i_R} + 2\}$. We assume that the computation of $S(i_L, v)$ ($S(i_R, v)$) has recursively returned a union of l (m) v-value intervals in which $S(i_L, v)$ ($S(i_R, v)$) achieves its minimum value $s^*_{i_L}$ ($s^*_{i_R}$), as well as as the union of all v-value intervals in which it achieves the value $s^*_{i_L} + 1$ ($s^*_{i_R} + 1$); this assumption is valid in the bottom-level case; it will be recursively validated thereafter. The appropriate assigned values for each incoming value can also be derived from the stored information according to the preceding discussion. Let $\mathbf{L}_i = \bigcup_{j=1}^{l} L_j$ ($\mathbf{R}_i = \bigcup_{j=1}^{m} R_j$) be the former union of intervals, i.e. $v \in \mathbf{L}_i \Leftrightarrow S(i_L, v) = s^*_{i_L}$ ($v \in \mathbf{R}_i \Leftrightarrow S(i_R, v) = s^*_{i_R}$); likewise, let $\mathbf{L}^1_i = \bigcup_{j=1}^{s} L^1_j$ ($\mathbf{R}^1_i = \bigcup_{j=1}^{t} R^1_j$) be the union of all intervals such that $v \in \mathbf{L}^1_i \Leftrightarrow S(i_L, v) = s^*_{i_L} + 1$ ($v \in \mathbf{R}^1_i \Leftrightarrow S(i_R, v) = s^*_{i_R} + 1$).

By analogy to the bottom-level case, if $\mathbf{L}_i \cap \mathbf{R}_i \neq \emptyset$, then the minimum value of $S(i,v)$ is $s^*_{i_L} + s^*_{i_R}$, obtained for $v \in \mathbf{L}_i \cap \mathbf{R}_i$; such an incoming value is itself an optimal incoming value for both subtrees of C_i. Otherwise, the minimum value of $S(i,v)$ is $s^*_{i_L} + s^*_{i_R} + 1$. Besides, $S(i,v) \leq s^*_{i_L} + s^*_{i_R} + 2$, since two supplementary coefficients suffice to produce optimal incoming values to both C_{i_L} and C_{i_R} for any incoming value to C_i. Hence, if $\mathbf{L}_i \cap \mathbf{R}_i \neq \emptyset$, then $S(i,v) \in \{s^*_{i_L} + s^*_{i_R}, s^*_{i_L} + s^*_{i_R} + 1, s^*_{i_L} + s^*_{i_R} + 2\}$, otherwise $S(i,v) \in \{s^*_{i_L} + s^*_{i_R} + 1, s^*_{i_L} + s^*_{i_R} + 2\}$. In more detail, $S(i,v)$ assumes the value $s^*_{i_L} + s^*_{i_R} + 1$ in the following cases.

—When a single nonzero *supplementary* coefficient suffices to produce an optimal incoming value to both subtrees of C_i. It occurs so if and only if $v \in \mathbf{L}_i \cup \mathbf{R}_i - \mathbf{L}_i \cap \mathbf{R}_i = \mathbf{L}_i \triangle \mathbf{R}_i$. If $v \in \mathbf{L}_i$, then any value z such that $v + z \in \mathbf{R}_i$ suffices to be assigned as a right supplementary coefficient. Otherwise, if $v \in \mathbf{R}_i$, then any value z such that $v + z \in \mathbf{L}_i$ should be assigned as a left supplementary coefficient.

—When the given incoming value v allows for a stand-alone (as Lemma 4.1 prescribes), nonzero *head* coefficient to produce an optimal incoming value to both subtrees of C_i. It occurs so if and only if:

$$\left.\begin{array}{c} v + z \in \mathbf{L}_i \\ \wedge \\ v - z \in \mathbf{R}_i \end{array}\right\} \Rightarrow v \in \bigcup_{\substack{j=1\ldots l \\ k=1\ldots m}} L_j \diamond R_k \tag{3}$$

The notation $L_i \diamond R_j$ excludes values $v \in L_j \cap R_k$. Still, in fact, *any* incoming value $v \in \mathbf{L}_i \cap \mathbf{R}_i$ (that is, in any intersecting pair of intervals) should be excluded from the present case of $S(i,v) = s^*_{i_L} + s^*_{i_R} + 1$, as it belongs to the case $S(i,v) = s^*_{i_L} + s^*_{i_R}$. Hence, for convenience in notation, we define the $\diamond$

operator for a union of l continuous intervals $\mathbf{A} = \cup_{i=1}^{l} A_i$ and a union of m continuous intervals $\mathbf{B} = \cup_{i=1}^{m} B_i$ as follows:

$$\mathbf{A} \diamond \mathbf{B} = \bigcup_{\substack{i=1\ldots l \\ j=1\ldots m}} A_i \diamond B_j - \mathbf{A} \cap \mathbf{B}. \tag{4}$$

As Equation (4) shows, the $\diamond$ operator, applied on the pair $\{\mathbf{A}, \mathbf{B}\}$, returns the union of all intervals $A_i \diamond B_j$, for all i, j pairs, minus the intersection of $\mathbf{A}$ and $\mathbf{B}$. In effect, the case that a stand-alone nonzero head coefficient suffices to produce an optimal incoming value to both subtrees of C_i occurs, more succinctly, when $v \in \mathbf{L}_i \diamond \mathbf{R}_i$. For $v \in L_j \diamond R_k = [a, b] \diamond [c, d] = [\frac{a+c}{2}, \frac{b+d}{2}] - [a, b] \cap [c, d]$, the appropriate assigned values z are $z \in [\max\{a - v, v - d\}, \min\{b - v, v - c\}]$ as in Equation (1).

—When the incoming value to C_i is itself an optimal incoming value to one subtree of C_i and a suboptimal incoming value that requires one space unit more than the minimum one in the other subtree. This case occurs so if and only if $v \in \mathbf{L}_i \cap \mathbf{R}_i^1$ or $v \in \mathbf{L}_i^1 \cap \mathbf{R}_i$. In both cases, $S(i, v) = S(i_L, v) + S(i_R, v) = s_{i_L}^* + s_{i_R}^* + 1$.

Putting it all together, $S(i, v)$ is expressed as:

$$S(i, v) = \begin{cases} s_{i_L}^* + s_{i_R}^*, & v \in \mathbf{L}_i \cap \mathbf{R}_i \\ s_{i_L}^* + s_{i_R}^* + 1, & v \in \mathbf{L}_i \triangle \mathbf{R}_i \cup \mathbf{L}_i \diamond \mathbf{R}_i \cup \mathbf{L}_i \cap \mathbf{R}_i^1 \cup \mathbf{L}_i^1 \cap \mathbf{R}_i \\ s_{i_L}^* + s_{i_R}^* + 2, & v \notin \mathbf{L}_i \cup \mathbf{R}_i \cup \mathbf{L}_i \diamond \mathbf{R}_i \cup \mathbf{L}_i \cap \mathbf{R}_i^1 \cup \mathbf{L}_i^1 \cap \mathbf{R}_i. \end{cases} \tag{5}$$

Equation (5) defines $S(i, v)$ recursively throughout a Haar$^+$ tree. Again, in order to represent the full value range of $S(i, v)$, we only need to store two sets: (i) a set $\mathbf{P}_i$, such that $v \in \mathbf{P}_i \Leftrightarrow S(i, v) = s_i^*$, and (ii) a set $\mathbf{Q}_i$, such that $v \in \mathbf{Q}_i \Leftrightarrow S(i, v) = s_i^* + 1$. For the set of the rest values of v, if nonempty, it is inferred that $S(i, v) = s_i^* + 2$. This representation of the value range of $S(i, v)$ verifies the inductive step of our approach; we assumed that the value ranges of $S(i_L, v)$ and $S(i_R, v)$ were so represented at two children triads, and we have shown that $S(i, v)$ is then so represented at the parent triad; the assumption holds at the bottom Haar$^+$ tree level; hence, this representation is inductively propagated through the Haar$^+$ tree in a bottom-up fashion with our recursive scheme. The root case of the recurrence is as follows:

$$S(0, 0) = \begin{cases} s_1^*, & 0 \in \mathbf{C_0} \\ s_1^* + 1, & 0 \notin \mathbf{C_0}, \end{cases} \tag{6}$$

where $\mathbf{C_0}$ is the set of incoming values v that achieve the minimum space $S(1, v) = s_1^*$ at triad C_1. The computation of $S(0, 0)$ derives the minimum-space solution. We only need to trace backwards through the choices made at each triad after the solution at the top of the Haar$^+$ tree has been established. Besides, we may follow the space-efficiency paradigm suggested by Guha [2008]; after the solution is established a the topmost level, we solve the two half-size problems at the two subtrees of the root, and recursively recompute the respective solutions, by the same strategy.

Complexity Analysis. The space required to store the sets (unions of intervals) $\mathbf{P}_i$ and $\mathbf{Q}_i$ representing the value domain of $S(i, v)$ for a triad C_i grows with the level of the Haar$^+$ tree in which C_i resides. In a bottom-level triad C_i, exactly three distinct value intervals need to be stored. Each interval is represented by the pair of its defining values, along with auxiliary information that assists in the computation of appropriate assigned values z at C_i for an incoming value v in that interval. If $[a, b] \cap [c, d] \neq \emptyset$, then these distinct intervals are $[a, b] \cap [c, d] = \mathbf{P}_i$ and $[a, b] - [c, d]$ with $[c, d] - [a, b]$ whose union makes $\mathbf{Q}_i$; otherwise, if $[a, b] \cap [c, d] = \emptyset$, then the distinct intervals $[a, b]$, $[c, d]$ and $[\frac{a+c}{2}, \frac{b+d}{2}]$ whose union makes up $\mathbf{Q}_i$, need to be stored. In a worst-case scenario, the parent triad C_i at the next-to-bottom level receives a union of three intervals from both its bottom-level children triads C_{i_L} and C_{i_R} as the sets of incoming values $\mathbf{L}_i$, $\mathbf{R}_i$ that achieve the minimum space at them. The space required to store the respective sets representing the value range of $S(i, v)$ at C_i is dominated[2] by the results of the $\mathbf{L}_i \diamond \mathbf{R}_i$ operation; in the worst case, this operation returns a union of $3 \times 3 = 3^2 = 9$ intervals. By induction on the remaining levels, it follows that the space required to represent the value range of $S(i, v)$ at a triad C_i in level ℓ of the Haar$^+$ tree (where for the bottom level $\ell = 0$) is $O(3^{2^\ell})$. This complexity also expresses the time needed to compute these intervals and also absorbs the time for the auxiliary intersection and symmetric difference set operations. Since at most $\log n + 1$ triad value domains arrays need to be concurrently stored (one at each level on a root to leaf path, plus for the last triad's sibling), the space complexity is $O(\sum_{\ell=0}^{\log n} 3^{2^\ell}) = O(3^n)$. Similarly, level ℓ contains $\frac{n}{2^{\ell+1}}$ triads, hence the total time complexity is $O(\sum_{\ell=0}^{\log n} 3^{2^\ell} \frac{n}{2^{\ell+1}}) = O(3^n)$. That is also the space complexity of the algorithm without use of the space-efficiency technique. We observe that the space-efficiency technique does not create a complexity difference; however, it does prune the required storage space in practice.

[2]The other, intersection and symmetric difference set operations are linearly dependent on the sizes of their operand sets.