Visualization-aware Timeseries Min-Max Caching with Error Bound Guarantees: Supplementary Material

I. Introduction

This document presents supplementary material supporting our research paper, ”Visualization-aware Time series Min-Max Caching with Error Bound Guarantees”. Due to space constraints in the main paper, some details were omitted and are included here for comprehensive understanding.

Specifically, we provide detailed algorithms for the error bound calculation, as well as for query evaluation.

For any queries or further information, feel free to reach out to us at the following emails:

stavmars@athenarc.gr(Stavros Maroulis); bstam@athenarc.gr (Vassilis Stamatopoulos); gpapas@athenarc.gr (George Papastefanatos); mter@athenarc.gr (Manolis Terrovitis);

II. Algorithms

A. Upper Error Bound Evaluation

By considering the definition of the error bound for the visualization and the two theorems related to inner- and inter-column errors, we can calculate the maximum error in a line chart visualization of a variable y within a time series T when it is generated using a grouping $\mathcal{G}^k(T)$. This analysis is conducted through an iterative examination of each pixel column, where we determine its potential inner-column and inter-column pixel errors. The details of this process are presented in Algorithm 1.

Initially, we iterate over the groups within $\mathcal{G}^k(T)$ and determine, for each group B_j^k, whether it overlaps with up to two pixel columns (line 4). We also ascertain whether the group is fully-contained within a single pixel column p_i or partially contained across two adjacent columns, p_i and p_{i+1}. In the case of full containment, we consider the min-max y range of the group and update the corresponding inner-column pixel range P_i with values that we can confidently determine (line 3). For partially contained groups, we adjust the pixel ranges P_r^i and P_l^{i+1} to account for the potential foreground pixels contributed by B_j^k to columns p_i and p_{i+1}, respectively (lines 9 - 10).

After establishing both the sets of correctly rendered inner-column pixels and the potential pixels that could be affected by the left and right partially overlapping groups in each pixel column, we proceed to iterate over each pixel column p_i. Within each of these columns, we determine the potential inner-column pixel errors (line 11) and inter-column pixel errors (lines 13 to 18). Specifically concerning the inter-column errors, for each pixel column and at each of its boundaries with neighboring columns, we rasterize the potentially false inter-column lines that would be rendered using $\mathcal{G}^k(T)$ across the $i-j$ intersection (line 16). Additionally, we calculate the maximum set of pixels that may be missing due to the absence of rendering the correct inter-column lines (line 17). By combining these sets of inner and inter-column errors, we can identify the maximum potential errors within the visualization, enabling us to compute the error bound ϵ (line 22).

The algorithm has a computational complexity of $O(k+w)$, since it iterates over k groups in $\mathcal{G}^k(T)$ and w pixel columns in the visualization.

B. Query Processing

Next, we provide a detailed query processing algorithm for MinMaxCache. Algorithm 2 is designed to evaluate visualization queries while adhering to error bounds. It takes a query Q and a maximum acceptable error bound ϵ as input and produces query results R for visualization.

Algorithm 1: Determining the Upper Error Bound in a Line Chart Visualization

Input: $\mathcal{G}^k(T)$, w, h
Output: Error bound ϵ

1 $E_{max} \leftarrow 0$
2 **for** $i \leftarrow 1$ **to** w **do**
3 $\quad P_i \leftarrow \emptyset$ `// Initialize the foreground pixel range that can be accurately determined by the fully-contained groups in pixel column` p_i
4 **for** $B_j^k \in \mathcal{G}^k(T)$ **do**
5 $\quad$ Determine the leftmost pixel column p_i that group B_j^k spans
$\quad$ `// Check if` B_j^k `is fully contained in` p_i
6 $\quad$ **if** $B_j^k \subseteq B_i^w$ **then**
7 $\quad\quad P_i \leftarrow P_i \cup [p_y(B_j^{min}), p_y(B_j^{max})]$
8 $\quad$ **else**
$\quad\quad$ `// For partial containment, set` p_i`'s right partially overlapping group and the left of its immediate right neighbor`
9 $\quad\quad P_r^i \leftarrow [p_y(B_j^{min}), p_y(B_j^{max})]$
10 $\quad\quad P_l^{i+1} \leftarrow [p_y(B_j^{min}), p_y(B_j^{max})]$
11 **for** $i \leftarrow 1$ **to** w **do**
$\quad$ `// Calculate inner-column errors for column` p_i
12 $\quad E_{inner}^i \leftarrow (P_l^i \cup P_r^i) \setminus P_i$
$\quad$ `// Initialize set of potential inter-column errors`
13 $\quad E_{inter}^i \leftarrow \emptyset$
$\quad$ `// Calculate inter-column errors for column` p_i
14 $\quad$ **for** *each neighbor* $p_j \in \{p_{i-1}, p_{i+1}\}$ **do**
15 $\quad\quad$ **if** p_j *exists* **then**
16 $\quad\quad\quad$ Rasterize the line segment between the min or max values of $\mathcal{G}^k(T)$ before and after the intersection between the $i - j$ columns to obtain the set $F_{i,j}$
17 $\quad\quad\quad$ Determine the maximum set of pixels that the actual missing correct line across $i - j$ would render to obtain the set $M_{i,j}$
18 $\quad\quad\quad E_{inter}^i \leftarrow E_{inter}^i \cup F_{i,j} \cup M_{i,j}$
$\quad$ `// Combine inner- and inter-column errors for column` p_i
19 $\quad E_{inter}^i \leftarrow E_{inter}^i \setminus P_i$
20 $\quad E^i \leftarrow E_{inner}^i \cup E_{inter}^i$
21 $\quad E_{max} \leftarrow E_{max} + |E^i|$
22 $\epsilon \leftarrow \frac{E_{max}}{w \times h}$
23 **return** ϵ

In the initialization phase, the algorithm sets up a data structure to store information for each of the w pixel columns and for each variable y in Y_Q (line 1). Specifically, it maintains the fully contained min-max range for each variable y and keeps track of left and right partially overlapped groups for each pixel column and variable.

For each variable y in Y_Q, the algorithm identifies or initializes its corresponding interval tree IT_y (lines 3-5) and searches within each of them for groupings $G_{overlapping}^y$ that overlap the query interval I_Q (line 6).

If no overlapping groupings are found for any variable (line 7), indicating a complete cache miss, the algorithm fetches data from the database for the entire query interval I_Q using the initial default value for the aggregation factor AF (line 8).

If there are overlapping groupings for at least one variable, the algorithm proceeds with partial cache hits or cache

Algorithm 2: Query Evaluation in MinMaxCache

Input: $Q = (I_Q, Y_Q, w, h)$: visualization query; ϵ: acceptable error bound
Output: R: query results

Initialize w pixel columns, each maintaining left and right overlapped groups and fully contained min-max range for every $y \in Y_Q$
foreach $y \in Y_Q$ **do**
  Identify the interval tree IT_y based on $\mathcal{T}_{id}$ and y
  if IT_y *does not exist* **then**
    Initialize a new interval tree IT_y
  Find groupings $G^y_{overlapping}$ in IT_y that overlap with I_Q
if $\forall y \in Y_Q, G^y_{overlapping} = \emptyset$ **then**
  `// Complete cache miss`
  Fetch data from the database for the entire interval I_Q using the initial default value for AF
else
  foreach $y \in Y_Q$ **do**
    foreach $G \in G^y_{overlapping}$ **do**
      Update pixel columns based on G
    Calculate partial error bound ϵ'_y based on updated pixel columns for y
    Determine the aggregation factor AF_y, of the overlapping grouping with the largest coverage of I_Q
    $I^y_{missing} \leftarrow$ intervals in I_Q not covered by $G^y_{overlapping}$
    if $\epsilon'_y \leq \epsilon$ **then**
      if $I^y_{missing} \neq \emptyset$ **then**
        `// Partial cache hit for variable` y
        Fetch $I^y_{missing}$ from the database with AF_y
    else
      `// Cache miss for variable` y `because error exceeds bounds`
      Fetch data from the database for the entire interval I_Q for y with $AF'_y = 2 \times AF_y$ or use raw data if $\frac{\tau_{agg}}{\tau_s} < 4$
if *data was fetched from the database* **then**
  foreach $y \in Y_Q$ **do**
    Update pixel columns and IT_y with new groupings from fetched data for y
    Reevaluate ϵ'_y
if $\exists y \in Y_Q, \epsilon'_y > \epsilon$ **then**
  Issue an M4 query for I_Q for all $y \in Y_Q$ with $\epsilon'_y > \epsilon$
Prepare R as a set of points containing the min-max values and the ones with the first and last timestamps from each pixel column for each variable $y \in Y_Q$
return R

misses for each variable individually. For each variable y, it processes its overlapping groupings and updates pixel columns, which includes updating the fully contained min-max y range and tracking left and partially overlapped groups (line 12). It calculates a partial error bound based on the overlapped groups (line 13) and determines an aggregation factor AF_y for the variable by considering the aggregation factor of the overlapping grouping with the largest coverage over the query interval I_Q (line 14).

If a variable's partial error bound is lower than the acceptable error limit and there are missing intervals, this situation constitutes a partial cache hit. In such cases, the algorithm fetches only the missing data for this variable using the aggregation factor AF_y (line 18). However, if a variable's partial error bound exceeds the acceptable limit, indicating a cache miss for this variable, the algorithm retrieves data for the complete interval from the database. This retrieval is done with a doubled aggregation factor or, in cases where the aggregation interval τ_{agg} is very close to the sampling interval τ_s of the time series, raw data is retrieved (line 20). For data fetching, although not explicitly shown in the algorithm for simplicity, a single query to the database is issued to appropriately fetch the necessary data for each variable at the specific granularity needed for each of them.

When data needs to be fetched from the database, the algorithm proceeds to process the retrieved data for each variable. It performs updates on the corresponding pixel columns and reevaluates the error bounds (line 23). Subsequently, the algorithm conducts a verification step to ensure that error bounds are met for all variables. If any variable's error bound falls outside the predefined constraints, the algorithm initiates an M4 query for all variables that do not adhere to these constraints (line 26). In the final step, the algorithm compiles the query results, which include min-max values and timestamps for each variable (line 27), and proceeds to return the result set R (line 28).