Map-Based Visual Exploration of Geolocated Time Series

Georgios Chatzigeorgakidis
University of Peloponnese
Greece
chgeorgakidis@uop.gr

Dimitrios Skoutas
Athena R.C.
Greece
dskoutas@imis.athena-innovation.gr

Kostas Patroumpas
Athena R.C.
Greece
kpatro@imis.athena-innovation.gr

Spiros Athanasiou
Athena R.C.
Greece
spathan@imis.athena-innovation.gr

Spiros Skiadopoulos
University of Peloponnese
Greece
spiros@uop.gr

ABSTRACT

The amount and significance of time series that are associated with specific locations, such as visitor check-ins at various places or sensor readings, have increased in many domains over the last years. Although several works exist for time series visualization and visual analytics in general, there is a lack of efficient techniques for geolocated time series in particular. In this work, we present an approach that relies on a hybrid spatial-time series index to allow for interactive map-based visual exploration and summarization of geolocated time series data. In particular, we use the BTSR-tree index, which extends the R-tree by maintaining bounds for the time series indexed at each node. We describe the structure of this index and show how it can be directly exploited to produce map-based visualizations of geolocated time series at different zoom levels efficiently. We empirically validate our approach using two real-world datasets, as well as a synthetic one that is used to test the scalability of our method.

KEYWORDS

time series visualization, geolocated time series, visual exploration

1 INTRODUCTION

Time series are generated and stored at a vastly increasing rate in many industrial and research applications, including the Web and the Internet of Things, public utilities, finance, astronomy, biology, and many more. A significant portion concerns *geolocated* time series, i.e., those generated at, or otherwise associated with specific locations. While indexing, mining and exploring time series data has attracted a lot of interest from the database and data mining communities [4, 12, 25, 28], studying of geolocated time series is still largely overlooked.

Geolocated time series can be found in various domains and applications. A typical example is encountered in the *DAIAD* project[1], where time series are used to represent water consumption measured by smart meters installed in urban households. Analyzing such time series can provide valuable insights regarding trends and patterns of consumer behavior in daily life. Such results can then be used to forecast and balance water demand, as well as to plan and prioritize interventions that can guide consumers towards more sensible water use. Similar use cases can also be found in other domains, such as in geomarketing or mobile advertisement, where geolocated time series may represent the number of visitors or the revenue generated at a certain location across time. Extracting insights, trends and patterns can be significantly facilitated by *map-based visualizations* of *summarized* time series data. For example, such visualizations can reveal which type of consumption patterns are most frequently observed among consumers in a certain area or what the spatial distribution of sales for a certain product looks like.

However, time series is an inherently complex data type, and such datasets can reach extremely large volumes, both horizontally (i.e., very long series across time) and vertically (i.e., time series generated by countless sources). Consequently, management, analysis and exploration of such Big Time Series Data is a task of excessive complexity, requiring efficient algorithms. In particular, visual exploration of geolocated time series needs to process the required information in real time, while the user interacts with the map. Whenever the user zooms in or scrolls the map, visual analytics and aggregates should be computed on-the-fly, e.g., identifying the predominant patterns in the time series and their spatial distribution within the map area.

Consider the example illustrated in Figure 1. When the user zooms the map into the red rectangle, the visualization tool should readily identify and present the two patterns (shown in blue and green color) appearing therein. To avoid cluttering the map, when the spatial distribution is very dense or the map area is too large, it is meaningful to display only aggregate information, e.g., the number of time series per identified pattern and their respective centroids; individual time series may be identified only at a greater zoom level.

Such fast computation and retrieval for large datasets of geolocated time series can be enabled by indexing. Several approaches have been proposed that efficiently index large amounts of plain time series data, either relying on *Discrete Wavelet Transform* like [5] to reduce dimensionality of time series, or the family of indices based on *Symbolic Aggregate Approximation* (SAX) over the time series [3, 4, 28, 31]. However, all aforementioned techniques index the data solely on the time series domain, not taking the spatial dimension into account. If the analyzed time series are inherently associated with a spatial attribute (e.g., locations of smart meters), such indexing does not efficiently support queries and visualizations that do not simply apply to the time series domain, but also involve spatial filters. As in the example of Figure 1, a user may need to explore time series similar to a specific pattern, but also having locations within the actual map area. For such mixed requests, it is inefficient to evaluate each predicate separately, e.g., using first the time series index to

[1] http://daiad.eu/

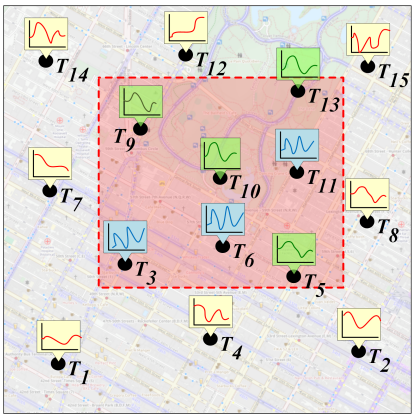

Figure 1: Exploring time series patterns in a spatial region.

retrieve a candidate set of time series similar to the pattern, and then applying a spatial filter against these candidates to retrieve only those qualifying within the specified query area.

To address this shortcoming, we have recently proposed an extension to the R-tree spatial index [16], offering efficient support to similarity search on geolocated time series. The idea behind our *BTSR-tree* hybrid index [7] is to combine both spatial proximity and time series similarity. To that end, in addition to the standard *Minimum Bounding Rectangle* (MBR) denoting the spatial bound of its contents, each node is augmented with a *Minimum Bounding Time Series* (MBTS), i.e., a band with upper and lower bounds that encloses all time series contained in its subtree. Maintaining both kinds of bounds per node enables pruning the search space simultaneously in the spatial and the time series domains while traversing the index. To increase pruning effectiveness, time series indexed in a given node are further distinguished into *bundles* on the basis of their similarity, hence achieving tighter bounds in the MBTS of these bundles.

In this paper, we take advantage of this novel BTSR-tree index and propose an interactive method for map-based visual exploration and summarization over large datasets of geolocated time series. Intuitively, when the user interacts with the map, i.e., either zooms in/out or moves the visible area, this technique can identify the most significant patterns characterizing the time series located in this area. These patterns are abstracted by the MBTSs of the bundles contained in the nodes of the BTSR-tree index within this area, summarizing the various time series therein. In addition, the corresponding MBRs and the number of geolocated time series per pattern can be depicted on the map.

For providing prompt visualizations of summaries over geolocated time series data and minimizing latency when drawing the relevant graphic elements, we need early access to both spatial and time series information while traversing the index. For this purpose, we adapt the BTSR-tree index so as to also include aggregates per node, i.e., the number of time series pertaining to each bundle. Subsequently, we introduce a new traversal algorithm for efficient retrieval of a given number of bundles that are the most representative in the map area. To the best of our knowledge, this is the first work that considers visual exploration and summarization of geolocated time series.

In summary, our main contributions are as follows:

- We propose an adapted variant of the BTSR-tree index as well as a novel algorithm for its traversal in order to quickly retrieve summaries (bundles) of geolocated time series within a given area.
- Thanks to robust index support, we introduce a method that can cluster bundles according to time series similarity, calculate their support and identify their spatial extent, thus enabling their interactive map-based exploration.
- We exemplify the proposed visualization method with two use cases based on real-world datasets.
- We also empirically evaluate the performance of index traversal, confirming its low execution cost against a large synthetic dataset of geolocated time series.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 outlines basic concepts and formulates the problem. Section 4 describes the BTSR-tree index and introduces our approach on summarization of geolocated time series for their efficient visual exploration. Section 5 presents indicative use cases with map visualizations and also reports performance results. Finally, Section 6 concludes the paper and outline future research directions.

2 RELATED WORK

As our approach suggests visual exploration of time series and is based on indexing of such data, next we briefly survey both of these research topics.

Visual Exploration of Time Series. In constrast to declarative visualization specifications suggested in [29], a recent tutorial [21] advocates the use of example-based methods in exploration of large relational, textual, and graph datasets. Such a *query-by-example* approach has been applied in [13] so as to explore relevance feedback for retrieval from time series databases. Instead of returning the top matching time series, this technique incorporates diversity into the results, which are presented to the user for feedback and refined in several rounds.

RINSE [32] is a Recursive Interactive Series Explorer specifically designed for exploration of data series. Built on top of ADS+ [31], a special adaptive index structure for data series, it can progressively build parts of the index on demand at query time, concerning only those chunks of the data involved in users' queries. In terms of visualization, users can get those series qualifying to range or nearest-neighbor queries interactively drawn on screen, as well as monitor various statistics regarding the index footprint (e.g., RAM and disk usage) as it gets updated.

In contrast, ATLAS [6] is a visual analytics tool specifically geared towards interactivity when ad hoc filters, arbitrary aggregations, and trend exploration are applied against massive time series data. This client-server architecture employs a column store as its backend equipped with indexing, and preemptively caches data that may be required in queries so as to reduce latency when *panning*, *scrolling*, and *zooming* over time series.

Recently, the ONEX paradigm [22] concerns online exploration of time series. It first constructs compact similarity groups over time series for specific lengths based on Euclidean distance, and then can efficiently support exploration of these groups with the Dynamic Time-Warping (DTW) method over their representatives of different lengths and alignments.

Besides, *smoothing* can be applied to streaming time series to remove noise in visualizations while preserving large-scale deviations [27]. To highlight important phenomena without harming representation quality from oversmoothing, this approach introduces quantitative metrics involving variance of first differences and kurtosis to automatically calibrate smoothing parameters.

ForeCache [2] leverages two prefetching mechanisms to facilitate exploration of large geospatial, multidimentional and time

series data stored in a DBMS. By predicting the user's behavior, it fetches the necessary data as the user interacts with the application.

None of the aforementioned methods and systems provides map-based visual exploration of *geolocated* time series, as is the goal of our work in this paper.

Indexing of Time Series. Earlier approaches towards indexing of time series data were based on leveraging multi-resolution representations. For instance, the Discrete Wavelet Transform [15] is used in [5] to gradually reduce the dimensionality of time series data via the *Haar wavelet* [17] and generate an index using the coefficients of the transformed sequences. In [26], it is further observed that, other than orthonormal wavelets, biorthonormal ones can also be used for efficient similarity search over wavelet-indexed time series data, demonstrating several such wavelets that outperform the Haar wavelet in terms of precision and performance. In addition, an alternative approach regarding k-nearest neighbor search over time series data is introduced in [18]. The proposed method accesses the coefficients of Haar-wavelet-transformed time series through a sequential scan over step-wise increasing resolutions.

State-of-the-art approaches for time series indexing comprise methods based on the *Symbolic Aggregate Approximation* (SAX) representation [20]. This is derived from the *Piecewise Aggregate Approximation* (PAA) representation of a time series [19, 30], by quantizing the segments of its PAA representation on the y-axis. The first attempt to leverage the potential of the SAX representation was presented in [28], introducing the indexable Symbolic Aggregate Approximation (*i*SAX), capable of a multi-resolution representation for time series. The iSAX index was further extended to *i*SAX 2.0 [3] by enabling bulk loading of time series data. Its next version is the *i*SAX2+ index [4], which handles better the expensive I/O operations caused by the aggressive node splitting while building the index. Finally, the ADS+ index [31] is another extension of *i*SAX, which overcomes the still significantly expensive index build time by adaptively building the index while processing the workload of queries issued by the user. A comprehensive overview of the time series indexing approaches based on the SAX representation is presented in [23].

Unfortunately, none of the abovementioned access methods can inherently support geolocated time series, i.e., time series inextricably associated with a location. To the best of our knowledge, the only index in the literature that supports such time series is the BTSR-tree index [7]. This hybrid index follows a similar rationale set by *spatio-textual indices* [8, 9, 11, 14] that have been proposed to speed up evaluation of queries combining location-based predicates with keyword search. Essentially, this paradigm implies combining a spatial index structure (e.g., R-tree, Quadtree, Space-Filling Curve) with a textual index (e.g., inverted file, signature file). Depending on their structure, these variants can be characterized either as *spatial-first* or *textual-first* indices [10]. In a similar spirit, our BTSR-tree is a spatial-first index based on the R-tree that can additionally abstract similarity of time series instead of a textual one. As a result, it can offer analogous improvements when searching against geolocated time series data, as we discuss in more detail in Section 4.1.

3 PROBLEM DEFINITION

Next, we introduce notation and definitions used in our approach, and we formally define the problem addressed in this paper.

A *time series* is a time-ordered sequence of values $T = \{v_1, \ldots, v_w\}$, where v_i is the value at the i-th time point and w is the length of the series. In particular, we deal with time series that are additionally characterized by a *location*, denoted by $T.loc$. Assuming a 2-dimensional space, we further use the notation $T.loc_x$, $T.loc_y$ to refer to the (x, y) coordinates of T's location.

In the *spatial domain*, the distance between two geolocated time series T and T' of equal length w is calculated using the Euclidean distance of their respective locations. Furthermore, we normalize this distance with $maxDist_{sp}$, i.e., the maximum spatial distance of any pair of objects in the dataset, to obtain a measure in the interval $[0, 1]$. Thus:

$$dist_{sp}(T, T') = \frac{\sqrt{(T.loc_x - T'.loc_x)^2 + (T.loc_y - T'.loc_y)^2}}{maxDist_{sp}} \tag{1}$$

In the *time series domain*, similarly to other prior works (e.g., [28]), we also apply the Euclidean distance to measure the similarity of a pair of objects. In future work, we plan to make use of more complex distance measures [24]. More specifically, we calculate the distance between two time series T and T' as follows:

$$dist_{ts}(T, T') = \frac{\sqrt{\sum_{i=1}^{w} (v_i - v'_i)^2}}{maxDist_{ts}} \tag{2}$$

where $maxDist_{ts}$ denotes the maximum distance of any pair of objects in the dataset and is used for normalization, as above.

To index and summarize time series, we use the notion of *Minimum Bounding Time Series* (MBTS). An MBTS is a summarization of a set of time series $\mathcal{T}$, defined by a pair of upper and lower time series bounds that contain them. Formally, given a set of time series $\mathcal{T}$, its MBTS consists of an *upper bounding time series* T_{up} and a *lower bounding time series* T_{lo}, constructed by selecting the maximum (for T_{up}) and minimum (for T_{lo}) of values at each time point among all time series of the set as follows:

$$\begin{aligned} T_{up} &= \{\max_{T \in \mathcal{T}} T[0], \ldots, \max_{T \in \mathcal{T}} T[w-1]\} \\ T_{lo} &= \{\min_{T \in \mathcal{T}} T[0], \ldots, \min_{T \in \mathcal{T}} T[w-1]\} \end{aligned} \tag{3}$$

We can now formally introduce our problem. Given a set of geolocated time series and an area, our goal is to produce a summary for visualization comprising the following two parts:

- *Time series summary*: A collection of MBTSs (*bundles*), summarizing the time series located within the given area.
- *Spatial summary*: A set of MBRs, each one associated with an object counter for each identified bundle.

The bundles provide a summarization of the time series that are contained within their MBTSs. Figure 2 depicts an example of two time series bundles for two different sets of time series. Regarding the spatial summary, for each MBR associated with a certain bundle, the counter denotes the number of time series contained in it.

4 APPROACH

We propose a visualization method for geolocated time series that draws on a map the time series and spatial summaries for the current visible area. Using this process, a user can select the bundle of her preference and the proper spatial summary will appear on the map after acquiring the necessary MBRs from

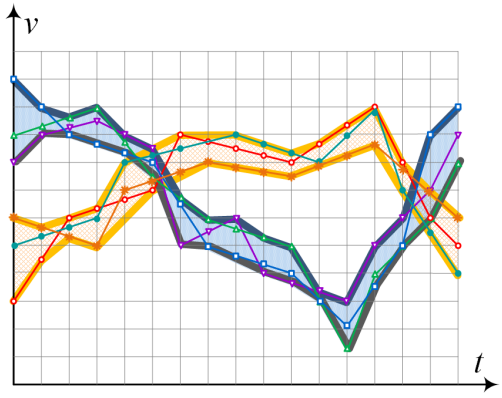

Figure 2: Example illustrating the resulting bundles for two sets of time series.

the BTSR-tree index. Whenever the user zooms in/out or moves around the map, the BTSR-tree is traversed, and the corresponding bundles, MBRs and object counts are obtained to drive the visualization. In each case, the rectangle corresponding to the visible part of the map is used to feed a traversal algorithm that efficiently gathers the results. In the following, we describe this process in detail, after providing some necessary background information on the BTSR-tree index.

4.1 The BTSR-tree Index

To efficiently generate real-time visualizations of geolocated time series data, we need early access to both spatial and time series related information while traversing the index, in order to maintain low latency levels when drawing the required graphic elements. However, none of the approaches presented in Section 2 supports geolocated time series indexing. To the best of our knowledge, the recently proposed BTSR-tree index [7] is the only one that provides the desired functionality.

The BTSR-tree is based on the R-tree [16] for the spatial indexing part. The R-tree organizes a hierarchy of nested d-dimensional rectangles. Each node corresponds to a disk page and represents the MBR of its children or, for leaf nodes, the MBR of its contained geometries. The number of entries per node (excluding the root) is between a lower bound m and a maximum capacity M. Query execution in R-trees starts from the root. MBRs in any visited node are tested for intersection against the search region. Qualifying entries are recursively visited until the leaf level or until no further overlaps are found. Several paths may be probed, as multiple sibling entries could overlap with the search region. The BTSR-tree extends the information stored within each node of the R-Tree with bundles of MBTSs. This allows to efficiently prune the search space when evaluating hybrid queries combining time series similarity with spatial proximity.

As in the standard R-tree, each node of the BTSR-tree has at least m and at most M entries and stores the MBRs of its children. Additionally, for each child, a node stores a pre-specified number of time series bundles, each consisted of an MBTS that encloses all the time series indexed in its subtree. Each bundle is calculated using Equation 3. Construction and maintenance of the BTSR-tree follow the procedures of the R-tree for data insertion, deletion and node splitting. Objects (i.e., geolocated time series) are inserted into leaf nodes, and any resulting changes are propagated upwards. Once the nodes have been populated, the bundles of each node are calculated bottom-up. To construct the time series bundles within each node, we rely on *k-means clustering*. The objects contained in each node are clustered according to their Euclidean distance on the time series domain. The example in Figure 2 depicts the bundles (the two bands with a thick outline) obtained for a set of time series (shown as thin polylines) when the number of clusters is set to $k = 2$.

For inner nodes, the bundles are constructed bottom-up. First, in each leaf node, the contained time series are clustered into k bundles. Then, the MBTS of each bundle is computed and stored in the node. As a next step, each parent node receives all the MBTSs of its children and computes its own k bundles and respective set of MBTS by clustering them. The process continues upwards, until reaching the root of the tree. Optionally, *Piecewise Aggregate Approximation* [19, 30] can be applied over the time series. As detailed in [7], this allows a trade off between the number of bundles per node and the MBTS resolution, thus permitting a larger number ($> k$) of bundles in nodes at higher levels in the tree hierarchy.

In addition, to support the required functionality of our visualization method, we further extend here the information stored in each node with the *count* of geolocated time series that are fully contained within each bundle. This is also done bottom-up, while the index is traversed to calculate the bundles. At each leaf node, after the clustering, we propagate the number of members of each cluster to its parent, which, in turn calculates its clusters and aggregates the counts it has received for each bundle's members. This procedure continues up to the root of the tree.

4.2 Summary Construction for Map-Based Visualization

We now present our summarization approach for producing map-based visualizations of geolocated time series. The process is outlined in Algorithm 1. It takes as input the *query rectangle* (q), i.e., the area of the map for which the visualization is produced, and the number of bundles k to be generated. The process comprises three distinct steps. Initially, the BTSR-tree index is traversed to obtain the MBRs contained in the query rectangle, along with their bundles and the number of objects per bundle (Line 1). Next, k-means clustering is applied using the average time series per bundle as centroids (Line 2). Finally, the new bundles are calculated and the proper MBRs and corresponding object counts are assigned to each bundle (Line 3). Next, we describe each step in more detail.

Step 1: BTSR-tree Traversal. During this step, the BTSR-tree index is traversed, with the target being the fast provision of a predefined number k of geolocated time series bundles contained within the given area q, along with the MBRs where these bundles can be found and the total number of geolocated time series that reside within each MBR. All required information is stored within the nodes of the BTSR-tree, thus, when a node that is contained within the query rectangle is found, the relevant information is retrieved and added to the intermediate results, without any need to continue searching in its sub-tree. The output of this step is passed to the next step of the procedure.

In more detail, the traversal is performed as follows. After initializing a queue with the root's children (Line 7), we loop over it (Line 8) until it's empty. For each inner node's child N', we check whether its MBR is contained within the given query rectangle q (Lines 11–12). If so, its MBR, time series bundles and

the number of objects per bundle are added to the intermediate results (Line 13) as a *tuple* with the following components:

$$\langle mbr, \{(mbts_1, cnt_1), ..., (mbts_k, cnt_k)\}\rangle$$

Each such tuple indicates the MBR of a node (*mbr*), consisting of the coordinates of the lower left and upper right point, as well as k pairs denoting the bundles of the node along with the corresponding number of objects per bundle. If the MBR is not contained in the query rectangle, we check whether it overlaps with it and if so, we add the child node to the queue (Line 15). If not, this MBR is located outside the query rectangle, and thus we can skip searching this child. Once no more nodes are left to search, the intermediate results are finally returned (Line 16).

Step 2: Bundles Clustering. The traversal algorithm returns tuples, each containing the bundles residing in the query rectangle, the corresponding nodes' MBRs and the number of objects per bundle. During step 2, k-means clustering is executed on the average time series of each bundle.

Line 2 of Algorithm 1 calls the clustering procedure. Initially, for each tuple (Line 20), we loop over its bundles (Line 21) and generate a new tuple per bundle of the following format:

$$\langle T_{avg}, mbts, cnt, mbr\rangle$$

This new tuple contains an average time series, the bundle itself (*mbts*), the number *cnt* of objects enclosed in this bundle, and the MBR (*mbr*) this bundle belongs to (Line 23). The average time series T_{avg} is calculated by averaging the upper and lower bound of each bundle (Line 22), i.e., average value at each time point. The resulting collection of tuples (Line 24) is fed to the k-means algorithm in order to return the required number k of bundles to be created. This clustering generates a clustered collection of tuples using the calculated average time series (Line 25). These results are then forwarded to step 3 (Line 26).

Step 3: Bundles Calculation and MBR Assignment. During step 3, the clustered tuples received from step 2 are used to calculate the final bundles, corresponding MBRs and total number of objects per MBR are assigned to each bundle. The final bundles are calculated in a similar manner to the MBTS bundles during BTSR-tree construction. More specifically, at each time point, we obtain the maximum and minimum value among the corresponding upper and lower bounds for the bundles of each cluster (see Section 4.1). In case two MBRs that belong to the same final bundle are the same, their number of objects is aggregated. The final result is then forwarded to the visualization layer.

Line 3 of Algorithm 1 calls the corresponding procedure. For each cluster of tuples received from step 2 (Line 29), we loop over its members (Line 32) and we use each tuple's bundle and MBR to update the upper and lower bounds and the collection of MBRs that are contained in the final bundle (Lines 33–34). Once the bounds and the corresponding list of MBRs for the current bundle have been calculated, we issue an aggregated tuple to the final result (Line 35). This tuple has the following components:

$$\langle mbts', \{(mbr_1, cnt_1), ..., (mbr_n, cnt_n)\}\rangle$$

where $mbts'$ is a resulting bundle, along with the MBRs associated with it. Note that the number n of MBRs (as well as their shape) may be varying per bundle, reflecting the spatial distribution of the respective pattern. Each MBR is accompanied with the corresponding number of objects (i.e., raw time series) therein. The final result with all such tuples is then returned in order to generate the visualization (Line 36).

Algorithm 1: Summarization of Geolocated Time Series

Input: The query rectangle q; the number of bundles to be generated k
Output: A list R containing tuples of bundles, MBRs and object counts

```
1  R ← IndexTraversal(q)                                   // Step 1
2  R_c ← BundlesClustering(R,k)                            // Step 2
3  R_f ← BundlesCalculation(R_c)                           // Step 3
4  return R_f
5  Procedure IndexTraversal(q)
6      R ← ∅
7      Q ← Root.getChildren()
8      while Q ≠ ∅ do
9          N ← Q.getNext()
10         if N is not leaf then
11             foreach N' ∈ N.getChildren() do
12                 if q.contains(N'.mbr) then
13                     R ← R ∪ {⟨N'.mbr, {N'.mbts}, {N'.cnt}⟩}
14                 else if q.overlaps(N'.mbr) then
15                     Q ← Q ∪ N'.getChildren()
16     return R
17 Procedure BundlesClustering(R,k)
18     R_c ← ∅
19     C ← ∅
20     foreach t ∈ R do
21         foreach b ∈ t.mbts do
22             T_avg ← avg(b.up, b.lo)
23             t' ← ⟨T_avg, b, t.cnt(b), t.mbr⟩
24             C ← C ∪ {t'}
25     R_c ← kmeans(C.avg, k)
26     return R_c
27 Procedure BundlesCalculation(R_c)
28     R_f ← ∅
29     foreach Cl ∈ R_c.clusters do
30         B ← ∅
31         M ← ∅
32         foreach t ∈ Cl do
33             B ← updateMBTS(B, t.mbts)
34             M ← updateMBRs(M, t.mbr, t.cnt)
35         R_f ← R_f ∪ {⟨{B}, {M}⟩}
36     return R_f
```

5 EXPERIMENTAL EVALUATION

In this section, we first describe our experimental setup, followed by indicative examples of map-based visualizations of real-world geolocated time series, as well as scalability results using a synthetic dataset containing 4 million time series. The experiments were conducted on a Dell PowerEdge M910 with 4 Intel Xeon E7-4830 CPUs, each containing 8 cores clocked at 2.13GHz, 256 GB RAM and a total storage space of 900 GB. Finally, we assume that the index fits in memory.

5.1 Experimental Setup

5.1.1 Datasets. We use two real-world datasets selected from different application domains and with diverse characteristics. In addition, we generated a synthetic dataset to test the scalability of our method. Table 1 lists a summary of the characteristics of each dataset.

Table 1: Datasets used in the experiments.

Dataset	Area (km^2)	Number of time series	Length w of each time series
Water	114	822	168
Taxi	2,500	417,960	168
Synthetic	114	4,000,000	168

DAIAD Water Consumption (Water). Courtesy of the DAIAD project, we acquired a geolocated time series dataset of hourly water consumption for 822 households in Alicante, Spain from 1/1/2015 to 20/1/2017. In order to get a more representative dataset for our tests, we calculated the average weekly time series per household, which is the average consumption value per hour of the week. Thus, the length of each resulting time series is $24 \times 7 = 168$ values across the week.

NYC taxi dropoffs (Taxi). This dataset contains time series extracted from yellow taxi rides in New York City during 2015. The original data[2] provide pick-up and drop-off locations, as well as corresponding timestamps for each ride. For each month, we generated time series by applying a uniform spatial grid over the entire city (cell side was 200 meters) and counting all drop-offs therein for each day of the week at the time granularity of one hour. Thus, we obtained the number of drop-offs for 24×7 time intervals in every cell, which essentially captures the weekly fluctuation of taxi destinations there. The centroid of each cell is used as the geolocation of the corresponding time series.

Synthetic Water Consumption (Synthetic). To examine the scalability of our method, we generated a synthetic dataset comprising 4 million geolocated time series by inflating the water consumption dataset. This was achieved by using the original time series as seeds and introducing some random variations in their location and pattern. We chose the water dataset so as to generate a more densely populated dataset (Alicante is a medium-sized city), in order to stress-test our visualization method.

5.1.2 Parameters. We built the BTSR-Tree index setting the minimum and maximum number of entries per node to $m = 60$ and $M = 200$, respectively. Regarding the number of bundles, we set $k_0 = 5$ for its leaf nodes. The number of bundles for the traversal algorithm is set to be equal to the number of bundles at the leafs, i.e., $k = k_0 = 5$. For an evaluation of the BTSR-Tree index under different parameter settings, please refer to [7].

5.2 Map Visualizations

Our visualization method depicts the MBTS derived for the most representative patterns of time series at the currently visible area of the map. Once our summarization method returns the results, the corresponding MBRs contained in the current view and zoom level are drawn on the map, along with the number of the geolocated time series that belong to the selected bundle. This number is depicted using circles, colored green for small numbers, yellow for larger and red for more densely populated MBRs, thus easily conveying the local intensity of this pattern. The bundles are listed on the left of the map, using confidence bands to indicate their upper and lower bounds. The average time series of each bundle is also depicted. A user can scroll this list and select the bundle of their preference. Once a bundle is selected, the contents of the map are updated accordingly with the respective MBRs and aggregates.

Figure 3 shows an example of such visualization using the water dataset. The depicted area is in the center of Alicante, in the most densely populated zone of the city. In this example, `Bundle 4` is selected (indicated with a green colored frame) and the relevant MBRs are shown on the map (using red colored frames). This indicates that inside each depicted MBR there exists a specific number of geolocated time series that have been clustered to the chosen bundle. As mentioned, each geolocated time series in this dataset represents hourly water consumption of a household across one week. Different consumption behaviors have been grouped together and a daily pattern for each bundle can be noticed which is due to the Circadian rhythmic way that people consume water [1]. The rather large number of geolocated time series in the bundle, considering the zoom level and the extent of the MBRs, intuitively suggests that neighboring families tend to have similar water consumption behavior.

Figure 4 illustrates another example, this time using the taxi dataset in New York City. This dataset is significantly larger, and the zoom level selected in this example is lower (a larger geographic area is visible), hence the MBRs contain a larger number of time series. In this figure, we choose `Bundle 1`, which represents the rather quieter taxi dropoff zones in Manhattan, as the total number of dropoffs there is rarely over 60 during any hour of the week. In this example, there is also a clear daily routine in all bundles, with the dropoffs reaching a local maximum twice per day, suggesting the rush hours in New York City, when people commute to and from their work. In almost all bundles, the daily pattern is significantly different on Saturdays and Sundays, which confirms the intuition that during weekends people do not tend to commute in a routinely fashion. Overall, such visual representations of information digested from massive time series data can easily catch users' attention to important phenomena and ongoing trends, confirming the usefulness of our approach.

5.3 Performance Results

In order to evaluate the performance of our approach on larger datasets, we built the index using the synthetic dataset and examined its response time for different zoom levels on the map. Since this is intended as an interactive application, where the summarization method is triggered as soon as the user moves the map, response times must be adequately small. Ideally, the response time should be in the order of milliseconds. In our method, this is facilitated by the fact that the search along a path stops once it encounters a node whose MBR is contained in the actual map extent (rectangle). In this experiment, we measure the response time for different zoom levels, since zooming-in requires deeper traversal of the BTSR-tree index in order to locate the relevant nodes. We use map scales to indicate the different zoom levels.

Figure 5 depicts traversal costs for different map scales over the areas covered by the three datasets. More specifically, the water and synthetic datasets cover the area of the city of Alicante, Spain, whereas the taxi dataset the wider metropolitan area of New York City. Response time in all cases is equal or lower than one second, which makes this method suitable for interactive visualization. The synthetic dataset, due to its very high density is significantly slower than the rest, however still the results are

[2] http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

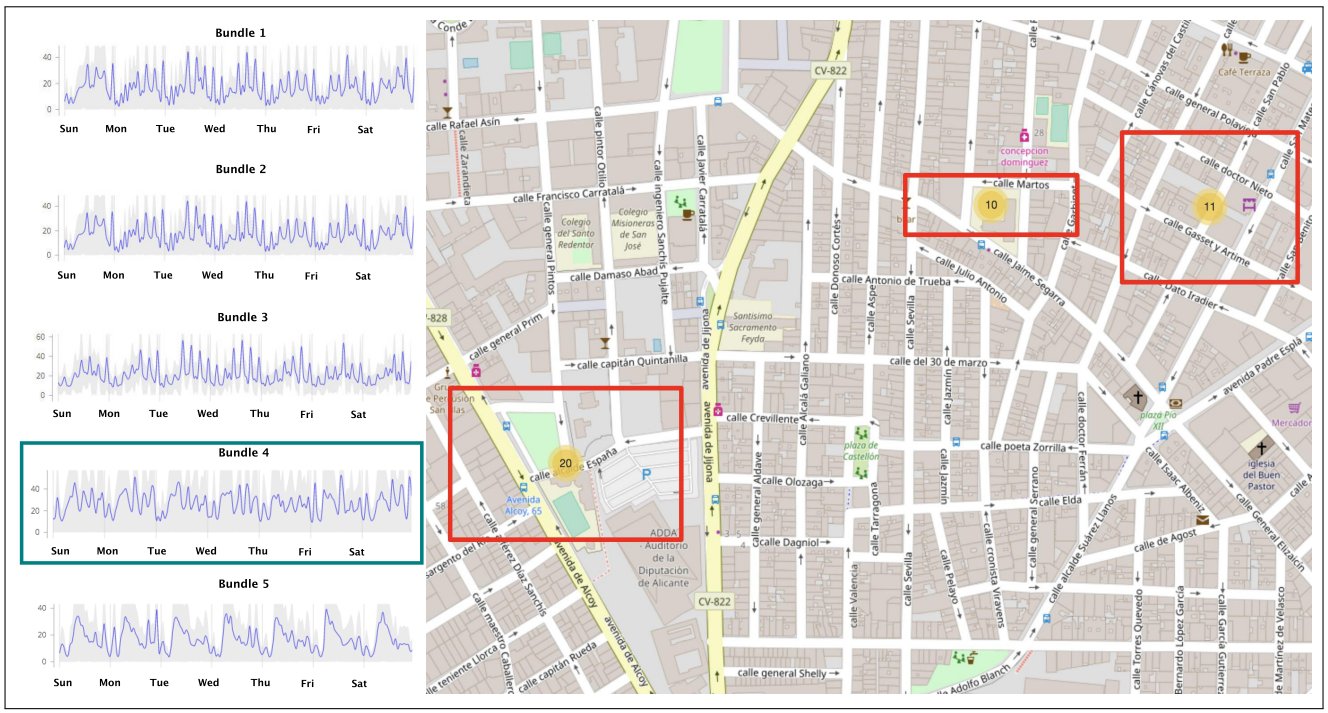

Figure 3: Visualizing water consumption patterns in the city center of Alicante (map scale 1:5000).

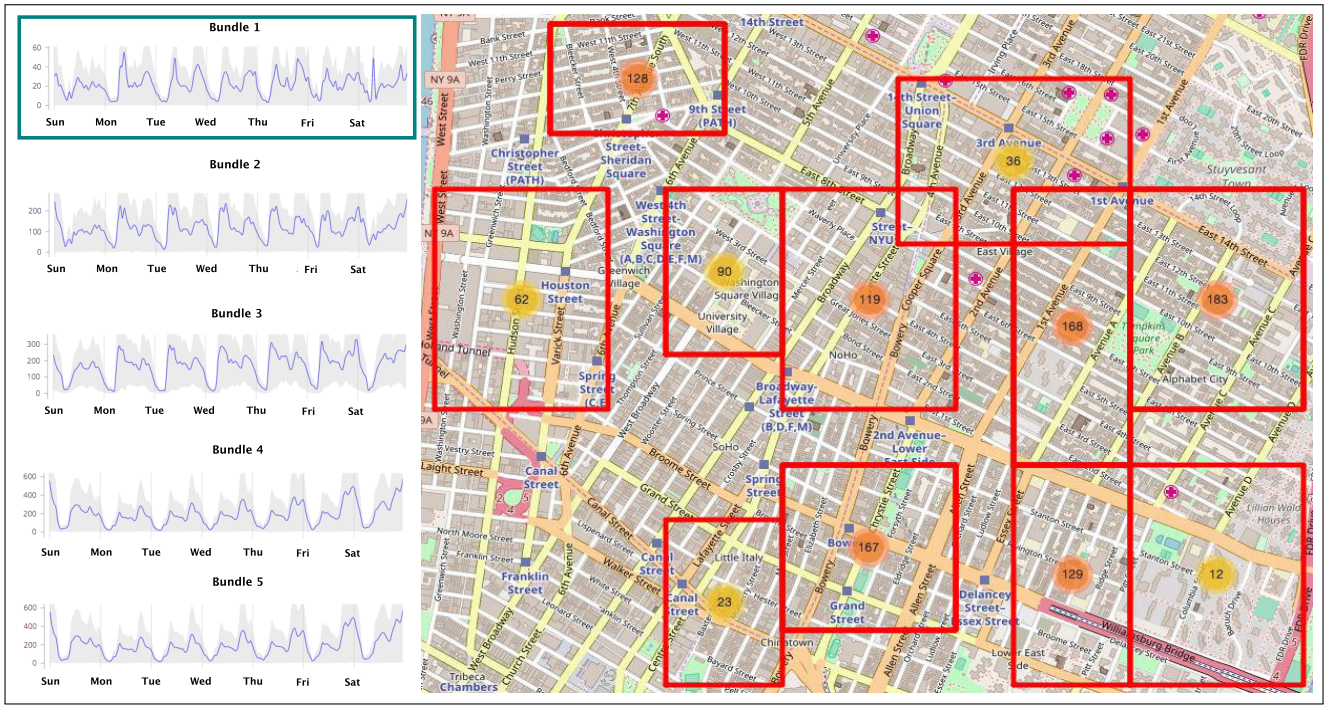

Figure 4: Visualization of taxi dropoff patterns in Manhattan, NYC (map scale 1:10000).

obtained in less than a second. The response for the water dataset is almost instant due to its small size and very low density.

Initially, in all cases, at the largest scale, the visible area of the map contains all the time series in the dataset, thus it only has to retrieve information from the root of the index. Then, as we zoom in, more nodes have to be visited, as the MBRs of the accessed nodes begin to overlap with the map rectangle and their children have to be retrieved. The worst case for the synthetic dataset is at scale 1:5000, which roughly corresponds to a large neighborhood of the city, where many time series are located. For the taxi dataset, the worst case is at 1:20000, which corresponds to the wider Manhattan area and then the response time gradually drops due to the lower dataset density. The number of nodes accessed in each case is proportional to the response times, ranging from

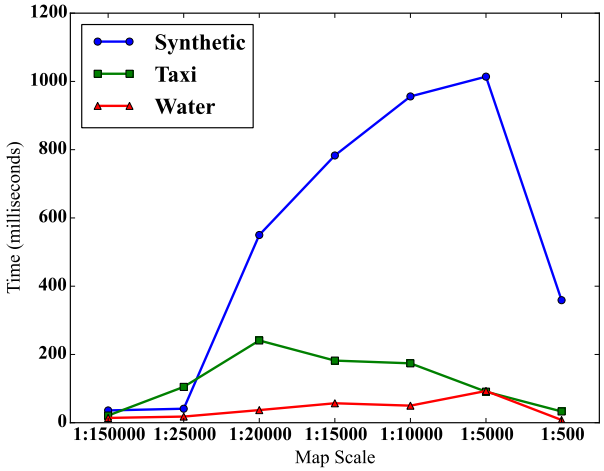

Figure 5: Execution time for different map scales.

one node (the root) in case of the smaller map scale (all city) up to 165 at scale of 1:5000 for the synthetic dataset, one up to 53 for the taxi dataset and one up to 15 for the water dataset. Interestingly, fewer node accesses are required in all cases at the very large scale of 1:500, since the respective small map area overlaps with fewer nodes and most of the search space is pruned.

Consequently, we deem that our method performs adequately fast even against a heavily dense synthetic dataset, where a large number of time series are contained within a small area.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a method for map-based visual exploration of geolocated time series data. To that end, we proposed a summarization approach over geolocated time series, which allows a visual analytics application to retrieve the required information. Such retrieval can be achieved at low latency, thus being suitable for interactive exploration of large volumes of such data. The results can be displayed on a map, depicting the relevant MBRs and the number of time series contained in each one, for a selected pattern detected in the time series data. Thanks to the support of a robust hybrid indexing technique, the patterns detected at a given zoom level are calculated via k-means clustering over the time series that reside in the currently visible part of the map. Our experiments on a large-scale synthetic dataset indicated that the visualization can be rendered adequately fast for use in interactive map-based applications. Additionally, we presented indicative demonstrations of the visualizations generated on two real-world datasets from different domains, confirming that these visualizations are helpful in revealing patterns both on the time series themselves as well as their geographic distribution.

Our ongoing and future work focuses on supporting more detailed visual analytics and identifying more fine-grained patterns through visual exploration. One possible extension would be to enable zooms along time, so that the user can identify patterns and their spatial distribution, not only over the entire time series, but also over particular intervals. Further, it would be interesting to drill-down in a particular summarized result and discover whether there are differentiations in the spatial distributions of its constituent, more detailed patterns.

ACKNOWLEDGMENTS

This work has been partially funded and supported by the General Secretariat for Research and Technology (GSRT), the Hellenic Foundation for Research and Innovation (HFRI), the EU Project *City.Risks* (grant No 653747), and the NSRF 2014-2020 project *HELIX* (grant No 5002781).

REFERENCES

[1] J. Aschoff. Circadian rhythms in man. *Science*, 148(3676):1427–1432, 1965.
[2] L. Battle, R. Chang, and M. Stonebraker. Dynamic prefetching of data tiles for interactive visualization. In *SIGMOD*, pages 1363–1375, 2016.
[3] A. Camerra, T. Palpanas, J. Shieh, and E. J. Keogh. iSAX 2.0: Indexing and mining one billion time series. In *ICDM*, pages 58–67, 2010.
[4] A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. J. Keogh. Beyond one billion time series: indexing and mining very large time series collections with i SAX2+. *Knowl. Inf. Syst.*, 39(1):123–151, 2014.
[5] K. Chan and A. W. Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.
[6] S. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining interactivity while exploring massive time series. In *IEEE VAST*, pages 59–66, 2008.
[7] G. Chatzigeorgakidis, D. Skoutas, K. Patroumpas, S. Athanasiou, and S. Skiadopoulos. Indexing geolocated time series data. In *SIGSPATIAL*, pages 19:1–19:10, 2017.
[8] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: An experimental evaluation. *PVLDB*, 6(3):217–228, 2013.
[9] Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *SIGMOD*, pages 277–288, 2006.
[10] M. Christoforaki, J. He, C. Dimopoulos, A. Markowetz, and T. Suel. Text vs. space: efficient geo-search query processing. In *CIKM*, pages 423–432, 2011.
[11] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.
[12] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. J. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *PVLDB*, 1(2):1542–1552, 2008.
[13] B. Eravci and H. Ferhatosmanoglu. Diversity based relevance feedback for time series search. *Proc. VLDB Endow.*, 7(2):109–120, 2013.
[14] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.
[15] A. Graps. An introduction to wavelets. *IEEE Comput. Sci. Eng.*, 2(2):50–61, 1995.
[16] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.
[17] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910.
[18] S. Kashyap and P. Karras. Scalable kNN search on vertically stored time series. In *SIGKDD*, pages 1334–1342, 2011.
[19] E. J. Keogh, K. Chakrabarti, M. J. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowl. Inf. Syst.*, 3(3):263–286, 2001.
[20] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, 2007.
[21] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. New trends on exploratory methods for data analytics. *Proc. VLDB Endow.*, 10(12):1977–1980, 2017.
[22] R. Neamtu, R. Ahsan, E. Rundensteiner, and G. Sarkozy. Interactive time series exploration powered by the marriage of similarity distances. *Proc. VLDB Endow.*, 10(3):169–180, 2016.
[23] T. Palpanas. Big sequence management: A glimpse of the past, the present, and the future. In *SOFSEM*, pages 63–80, 2016.
[24] J. Paparrizos and L. Gravano. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*, pages 1855–1870, 2015.
[25] P. Paraskevopoulos, G. Pellegrini, and T. Palpanas. Tweeloc: A system for geolocalizing tweets at fine-grain. In *ICDM Workshops*, pages 1178–1183, 2017.
[26] I. Popivanov and R. J. Miller. Similarity search over time-series data using wavelets. In *ICDE*, pages 212–221, 2002.
[27] K. Rong and P. Bailis. Asap: Prioritizing attention via time series smoothing. *Proc. VLDB Endow.*, 10(11):1358–1369, 2017.
[28] J. Shieh and E. J. Keogh. *i*SAX: indexing and mining terabyte sized time series. In *SIGKDD*, pages 623–631, 2008.
[29] E. Wu, L. Battle, and S. R. Madden. The case for data visualization management systems: Vision paper. *Proc. VLDB Endow.*, 7(10):903–906, 2014.
[30] B. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *VLDB*, pages 385–394, 2000.
[31] K. Zoumpatianos, S. Idreos, and T. Palpanas. Indexing for interactive exploration of big data series. In *SIGMOD*, pages 1555–1566, 2014.
[32] K. Zoumpatianos, S. Idreos, and T. Palpanas. Rinse: Interactive data series exploration with ads+. *Proc. VLDB Endow.*, 8(12):1912–1915, 2015.