A Visual Explorer for Geolocated Time Series

Georgios Chatzigeorgakidis
IMSI, Athena R.C., Greece
gchatzi@athenarc.gr

Kostas Patroumpas
IMSI, Athena R.C., Greece
kpatro@athenarc.gr

Dimitrios Skoutas
IMSI, Athena R.C., Greece
dskoutas@athenarc.gr

Spiros Athanasiou
IMSI, Athena R.C., Greece
spathan@athenarc.gr

ABSTRACT

We present spaTScope, a web application for visual exploration of geolocated time series. Analyzing such data is becoming increasingly important in many domains, such as energy demand management, geomarketing and geosocial networks. spaTScope allows users to visually explore large collections of geolocated time series and obtain insights about trends and patterns in their area of interest. The provided functionalities leverage a hybrid index that allows to navigate and group the available time series based not only on their similarity but also on spatial proximity. The results are visualized using linked plots combining maps and timelines.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**.

KEYWORDS

geolocated time series, hybrid indexing, visualization

ACM Reference Format:

Georgios Chatzigeorgakidis, Kostas Patroumpas, Dimitrios Skoutas, and Spiros Athanasiou. 2020. A Visual Explorer for Geolocated Time Series. In *28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20), November 3–6, 2020, Seattle, WA, USA*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397536.3422345

1 INTRODUCTION

Geolocated time series can be found in various domains and applications, especially related to geosocial networks and IoT. A typical example involves analyzing and forecasting water consumption measured by smart meters installed in urban households [4]. Analyzing such time series can provide valuable insights regarding trends and patterns of consumer behavior in daily life. These 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 be found in other domains, such as in geomarketing or mobile advertisement, where geolocated time series may represent the number of visitors or the business revenue generated at a certain location across time.

Extracting insights, trends, and patterns can be significantly facilitated by *linked visualizations* combining *map* and *timeline* views to summarize large collections of geolocated time series. For instance, 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, to enable interactive visualizations, queries and aggregations over geolocated time series need to be executed efficiently. Thus, an appropriate index structure becomes indispensable.

In recent work, we have presented *BTSR-Tree* [2], a hybrid index for geolocated time series that combines both spatial proximity and time series similarity. BTSR-Tree is an extension to the R-tree spatial index; in addition to the standard *Minimum Bounding Rectangle* (MBR) denoting the spatial extent of its contents, each node is augmented with a *Minimum Bounding Time Series* (MBTS), i.e., a pair of sequences that encloses all the 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 in order to provide response to queries against the dataset. To increase pruning effectiveness, time series indexed in a given node are further grouped into *bundles* on the basis of their similarity, hence achieving tighter bounds in the MBTS of these bundles. Each node can also store statistics, e.g., the number of time series pertaining to each bundle, in order to facilitate fast computation of aggregates, if necessary.

In this work, we demonstrate spaTScope[1], a web application that leverages the BTSR-Tree index to enable interactive visual exploration over large collections of geolocated time series. spaTScope takes advantage of the summarization approach we introduced in [3] and further enhanced in [1]. This technique can generate composite summaries on-the-fly for the geolocated time series within a selected area of interest in order to facilitate identification of significant patterns characterizing these time series as well as their distribution in space. More specifically, spaTScope offers an interactive UI consisting of two linked panels: (i) a *timeline* view that essentially groups similar time series together in a common plot as a band (i.e., *bundle*) representing the magnitude of their fluctuations across time, and (ii) a *map* view that displays MBRs with different colors and varying sizes to convey the spatial whereabouts of the locations for a selected bundle (i.e., group of time series). The application is supported by a back-end component, which communicates with the UI upon any user interaction (e.g., zoom in/out, pan). After focusing on the map area of interest, the user can trigger a search query against a BTSR-Tree index built

SIGSPATIAL '20, November 3–6, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-8019-5/20/11.
https://doi.org/10.1145/3397536.3422345

[1] https://github.com/smartdatalake/spaTScope

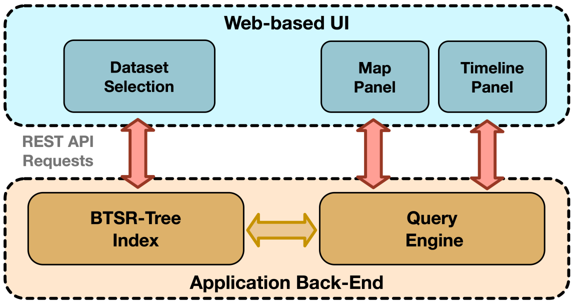


Figure 1: spaTScope architecture.

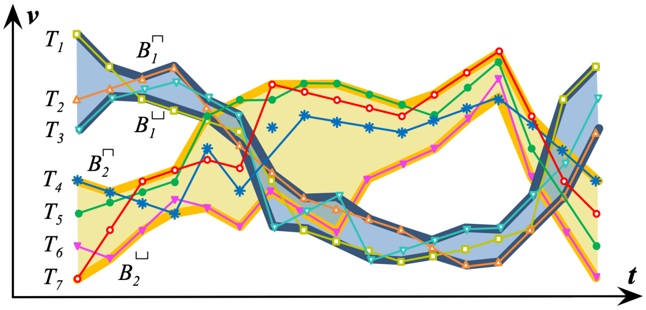
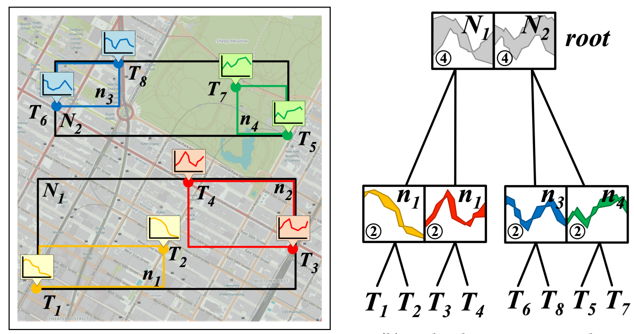


Figure 2: MBTS constructed for two sets of time series.

for the examined dataset. Summaries representing the qualifying results both in the spatial and time series domains are returned to the UI for rendering on the map and timeline panels, respectively.

Overall, spaTScope offers an interactive and user-friendly way to inspect and explore a dataset of geolocated time series. Through the panels, users can swiftly drill down to more detailed representations of the data or roll up to more concise renditions, depending on their area of interest and the degree of summarization applied on the underlying dataset.

The remainder of this paper proceeds as follows. Section 2 provides an overview of the application. Section 3 discusses the two back-end modules, and Section 4 presents the user interface. Section 5 outlines a demonstration scenario against a real-world dataset.



(a) Sample dataset (b) Hybrid BTSR-Tree index

Figure 3: The BTSR-Tree index.

2 SYSTEM OVERVIEW

Figure 1 presents an overview of spaTScope's architecture. This client-server application consists of three main components:

- *User Interface.* On the front-end, spaTScope offers a web-based interface that enables users to select a dataset of geolocated time series and explore it through the provided visualizations. Bundles are visualized in two linked panels: a *map* renders their spatial distribution and intensity, whereas on the *timeline panel* users can inspect their variations across time. Choosing a bundle from the timeline triggers a map update to focus on the time series belonging to that bundle. When interacting with the map, e.g., panning or zooming in, queries are sent to the back-end, and results are summarized and used to refresh or refine the depicted bundles. These changes are also reflected in the timeline panel.
- *Query Engine.* User interactions with the UI issue requests via a REST API to the query engine in spaTScope's back-end. Such requests are essentially search queries against the BTSR-Tree index of the explored dataset. Upon receiving the resulting bundle summaries, the query engine propagates them back to the UI for rendering them in the two panels.
- *BTSR-Tree Index.* To support efficient exploration and summarization in both the spatial and time series domains, this index structure is constructed in the back-end over the selected dataset of geolocated time series. The constructed BTSR-Tree is available for requests specified through the Query Engine module during data exploration. For efficiency, this index resides in memory for fast retrieval of results.

3 APPLICATION BACK-END

We first outline the BTSR-Tree, a state-of-the-art hybrid index tailored to efficient search against geolocated time series. We then present how the Query Engine module handles requests specified when users interact with the application.

3.1 The BTSR-Tree Index

The BTSR-Tree index [2] is based on the notion of *Minimum Bounding Time Series* (MBTS). Like an MBR that contains a set of geometries in the spatial domain, an MBTS encloses *a set of time series* $\mathcal{T}$. Each MBTS consists of an *upper bounding time series* $B^{\sqcap}$ and a *lower bounding time series* $B^{\sqcup}$, constructed by respectively selecting the maximum and minimum of values at each timestamp $i \in \{1, \ldots, n\}$ among all time series in set $\mathcal{T}$. Figure 2 depicts an example of two MBTSs B_1, B_2 for two disjoint sets of time series.

A BTSR-Tree index is initialized as an R-tree built on the spatial attributes of the given geolocated time series dataset, as depicted in the example of Figure 3. Besides MBRs, each node is enhanced to also store MBTSs of similarly evolving time series, shown as colored strips per node in Figure 3b. This enables efficient pruning of the search space when evaluating hybrid queries combining time series similarity with spatial proximity.

For each child, a node stores a pre-specified number of MBTSs. 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

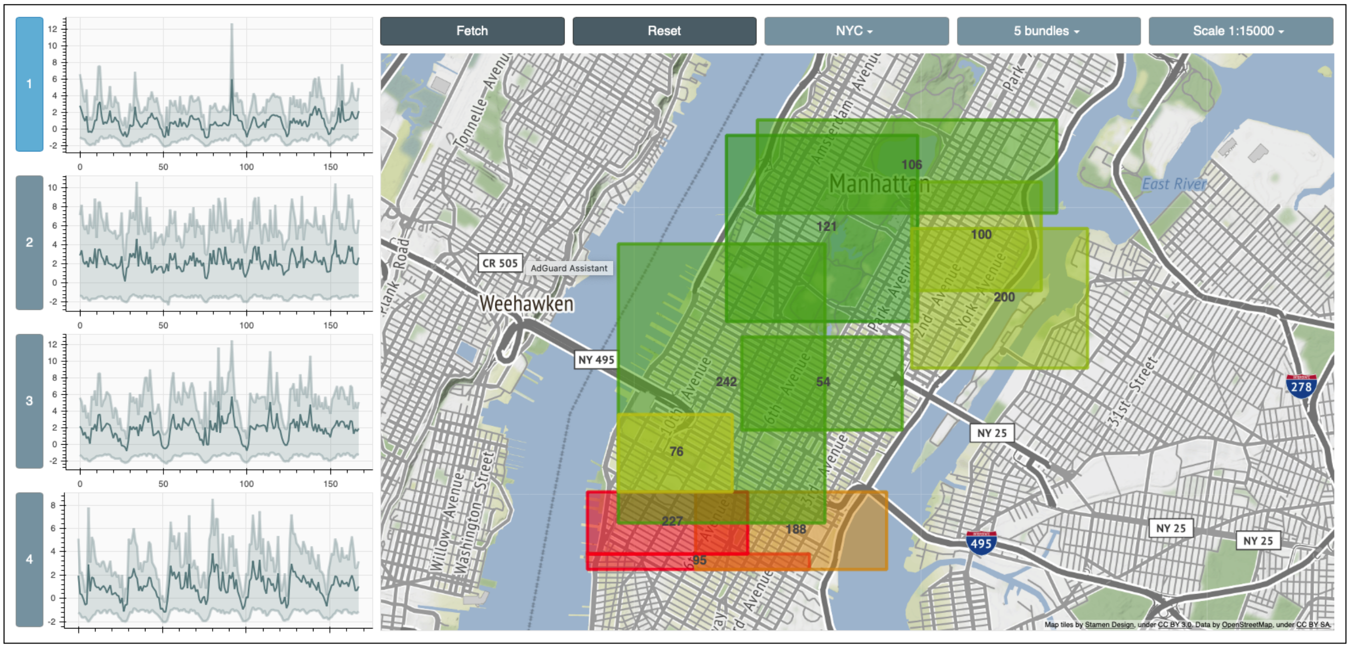


Figure 4: Application front-end visualizing taxi dropoff patterns in Manhattan, NYC.

any resulting changes are propagated upwards. Once the nodes have been populated, the MBTS of each node are calculated bottom-up, relying on *k-means clustering* according to their Euclidean distance in the time series domain, as exemplified for $k = 2$ in Figure 2. In a BTSR-Tree, each parent node receives all the MBTSs of its children and computes its own k MBTSs. To facilitate summarization, each node also retains the count of objects per MBTS indexed in the leaves of its subtree; this count is kept as a list $C = \{C_1, ..., C_k\}$. In Figure 2 such counts are shown within circles in each node. The process continues upwards, until reaching the root.

3.2 The Query Engine

In [3] we introduced an efficient approach for constructing summaries over a BTSR-Tree index. This was further enhanced in [1] for advanced visualizations of geolocated time series. In the sequel, we briefly outline this framework, which serves as the Query Engine in spaTScope. Query parameters include an *area of interest*, abstracted as the rectangle q of the currently visible area on map, and the number k of bundles to be retrieved from the index. The process comprises three successive steps:

Step 1: BTSR-Tree Traversal. For a given query, we search the index in order to fetch bundles having their MBRs within the specified area of interest q. A breadth-first search traversal of the index examines its nodes level by level. Once M nodes are fully contained within rectangle q, the traversal terminates, and their summarized data (i.e., MBR, MBTS, and the counts C per MBTS) are retrieved and added to the intermediate results. M is a system parameter that controls the maximum number of regions to be displayed on the map in order to avoid cluttering.

Step 2: Bundle Clustering. The set $\{\langle MBR, MBTS, C\rangle\}$ of intermediate results qualifying to the query criteria are further summarized in this stage. In particular, we apply a k-means clustering based on the obtained MBTS's, abstracting each one by the average of its upper $B^{\sqcap}$ and lower bounding series $B^{\sqcup}$.

Step 3: Bundle Calculation and MBR Assignment. Each of the k clusters produced in the previous step will represent the bundles returned for visualization. Thus, for each cluster we need to construct its respective MBTS from its members. Indeed, the overall MBTS per bundle is derived with the same process used during BTSR-Tree construction, i.e., taking the maximum (minimum) value per timestamp among all upper (respectively, lower) bounds among the members of the corresponding cluster. The MBRs per cluster are not combined, unless two of them coincide; in that case, we keep a single MBR and sum up the total number of objects in the original two MBRs. Finally, each bundle is represented with a single MBTS, multiple (possibly overlapping) MBRs, and each MBR denotes the number of geolocated time series contained therein.

4 USER INTERFACE

Figure 4 illustrates the user interface of spaTScope, which consists of two interactive panels. On the left side, a *timeline* visualizes bundle summaries fetched from the back-end. On the right side, a *map* displays the spatial distribution of a selected bundle. The two panels are linked, meaning that changes in one are also reflected to the other. For instance, if the user picks another bundle from the timeline panel, the corresponding MBRs appear on the map. Moreover, if the user focuses the map to another area by zooming or panning, a new search request is invoked to the back-end to refresh the bundles that correspond to the new area of interest.

Once the application is launched, the user can select the file containing the dataset for exploration. If the BTSR-Tree index for this dataset exists, it is automatically loaded; otherwise, it is constructed, and also stored for future use. The UI also contains buttons and drop-down lists for fetching the summaries from the back-end, resetting the map to its initial position, selecting the dataset file and selecting a specific scale value for the map.

Timeline panel. This container displays the collection of k bundle summaries fetched from the back-end. Users may specify a desired value k for bundles from a drop-down list; once this request is processed in the back-end, the map is updated accordingly to mirror the new bundle summaries. Each bundle in the timeline panel illustrates its MBTS as a band that fully encloses the time series summarized by this bundle. The average time series of each bundle is also drawn (in dark blue color in Figure 4) to indicate the general trend. Intuitively, as the displayed bundles have been derived after clustering (Section 3.2), the user will be able to identify diverse patterns in their fluctuation across time. By scrolling down this list, the user can choose a bundle for inspection by clicking on the corresponding button, which automatically draws its spatial distribution as MBRs on the map. Note that the bundles in this panel become progressively refined (i.e., bands get narrower) as the spatial area of interest shrinks. At the finest granularity, the timeline panel displays the raw time series, each one corresponding to a particular location on map. Selecting a time series in this panel highlights its location on map, and vice versa.

Map panel. This component has the typical functionalities of a web-based map application (zoom, pan, full extent). The user can move, or zoom in/out the map using her mouse. Each time the user clicks the "Fetch" button, a new search request is triggered to the query engine and the newly fetched results are rendered. Over a backdrop of raster map tiles (e.g., OpenStreetMap[2]), `spaTScope` can display the spatial distribution of the bundle selected in the timeline panel in the form of MBRs, as obtained by BTSR-Tree index. The corresponding total number of objects (i.e., raw geolocated time series) contained within each MBR is also shown. To convey the local density of objects, each MBR is colored accordingly using the spectral palette, i.e., ranging from blue for less dense MBRs, to yellow for moderately densed ones, up to red for the most densely populated MBRs.

When the user clicks on an MBR, the map automatically zooms in to its extend and a new search request is triggered in the back-end using the visible map extent as the new area of interest. In case the zoom level has reached the MBR of a leaf in the underlying BTSR-Tree, the map will eventually pinpoint the exact locations of the corresponding raw time series (and these time series will be shown in the timeline panel).

Finally, by pressing the *reset button*, the user can reset the application and render in both panels the same contents shown when the dataset was first chosen by the user (i.e., full spatial extent).

5 DEMONSTRATION SCENARIO

To demonstrate `spaTScope`, we have prepared a scenario based on geolocated time series extracted from taxi dropoff data in New York City. Specifically, this dataset contains time series extracted from yellow taxi rides in NYC during 2015[3]. Based on the timestamped locations for drop-off per ride, we generated time series per cell of a uniform spatial grid over the entire city (cell side was 200 meters). In each cell, we counted all drop-offs 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. In total, this dataset contains 417,960 geolocated time series. Performance-wise, the time between issuing a request to the application back-end and rendering the results on the UI for this dataset does not exceed two seconds.

Once the user connects to the web application, she is presented with its basic visual interface. First, she clicks on the file selection button to pick a dataset for exploration. The respective BTSR-Tree index is then loaded. The default number of bundles is set to $k = 5$; thus, 5 bundles that summarize the time series of the entire dataset are shown in the timeline panel. The first of those bundles is chosen and its corresponding MBRs are displayed on the map. Each bundle captures a distinct pattern of fluctuations in the time series domain, so the user will be able to directly identify on map the spatial distribution and intensity separately for each such pattern. This first-cut insight may be further refined by clicking on an MBR for closer inspection. The map zooms automatically to fully cover this MBR and both panels are refreshed: in the timeline, more fine-grained bundles are drawn capturing the patterns of time series located in the visible map area only. In addition, new MBRs are shown on map, corresponding to one of the bundles just retrieved. In case the user specifies another value for k, e.g., wishing to display $k = 10$ bundles, more refined ones are fetched from the back-end and the map is refreshed to show the new MBRs. Naturally, each of those fresh MBRs now summarizes fewer objects, hence the MBR extents and counts are adjusted accordingly. Thus, the user is able to progressively proceed her analysis into more detail, until reaching the finest zoom level. Then, individual locations are depicted on map along with their corresponding time series in the timeline. If the user zooms in/out or pans the map, fresh summaries are fetched from the back-end to illustrate the bundles and their distribution in the new map extent. Thus, when choosing a different area, the user can identify differing patterns in drop-offs and also locate where each one has strong presence.

ACKNOWLEDGMENTS

This work was supported by the EU H2020 project SmartDataLake (Grant No. 825041).

REFERENCES

[1] G. Chatzigeorgakidis, K. Patroumpas, D. Skoutas, S. Athanasiou, and S. Skiadopoulos. Visual exploration of geolocated time series with hybrid indexing. *Big Data Research*, 15:12–28, 2019.

[2] G. Chatzigeorgakidis, D. Skoutas, K. Patroumpas, S. Athanasiou, and S. Skiadopoulos. Indexing geolocated time series data. In *SIGSPATIAL*, pages 19:1–19:10, 2017.

[3] G. Chatzigeorgakidis, D. Skoutas, K. Patroumpas, S. Athanasiou, and S. Skiadopoulos. Map-based visual exploration of geolocated time series. In *BigVis (EDBT/ICDT Workshops)*, pages 92–99, 2018.

[4] P. Chronis, G. Giannopoulos, and S. Athanasiou. Open issues and challenges on time series forecasting for water consumption. In *DAMASCA (EDBT/ICDT Workshops)*, 2016.

[2] https://www.openstreetmap.org

[3] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page