ULISSE: ULtra compact Index for Variable-Length Similarity SEarch in Data Series

Michele Linardi #1, Themis Palpanas #2

# *LIPADE, Paris Descartes University*

1 michele.linardi@parisdescartes.fr

2 themis@mi.parisdescartes.fr

***Abstract*—Data series similarity search is an important operation and at the core of several analysis tasks and applications related to data series collections. Despite the fact that data series indexes enable fast similarity search, all existing indexes can only answer queries of a single length (fixed at index construction time), which is a severe limitation. In this work, we propose *ULISSE*, the first data series index structure designed for answering similarity search queries of *variable length*. Our contribution is two-fold. First, we introduce a novel representation technique, which effectively and succinctly summarizes multiple sequences of different length. Based on the proposed index, we describe efficient algorithms for approximate and exact similarity search, combining disk based index visits and in-memory sequential scans. We experimentally evaluate our approach using several synthetic and real datasets. The results show that *ULISSE* is several times (and up to orders of magnitude) more efficient in terms of both space and time cost, when compared to competing approaches.**

I. Introduction

Data sequences are one of the most common data types, and they are present in almost every scientific and social domain [1], [2]. This makes data series a data type of particular importance. Informally, a data series (a.k.a data sequence, or time series) is defined as an ordered sequence of points, each one associated with a position and a corresponding value[1].

Recent advances in sensing, networking, data processing and storage technologies have significantly facilitated the processes of generating and collecting tremendous amounts of data sequences from a wide variety of domains at extremely high rates and volumes. The *SENTINEL-2* mission [3] conducted by the European Space Agency (ESA) represents such an example of massive data series collection, producing over five trillion different data series that capture at fine grain the evolution of land use.

Once the data series have been collected, they need to be analyzed [4] in order to identify patterns, gain insights, detect abnormalities, and extract useful knowledge. Critical part of this process is the data series similarity search operation, which lies at the core of several analysis and machine learning algorithms.

[1]If the dimension that imposes the ordering of the sequence is time then we talk about time series. Though, a series can also be defined over other measures (e.g., angle in radial profiles in astronomy, mass in mass spectroscopy in physics, etc.). For the rest of this paper, we are going to use the terms *data series* and *sequence* interchangeably.

However, similarity search in very large data series collections is notoriously challenging [5], [6], [7], [8], due to the high dimensionality (length) of the data series. In order to address this problem, a significant amount of effort has been dedicated by the data management research community to data series indexing techniques, which lead to fast and scalable similarity search [9], [10], [11], [12], [13], [14], [5], [15], [16], [17], [18].

Despite the effectiveness and benefits of the proposed indexing techniques, which have enabled and powered many applications over the years, they all share a common restriction: they only support similarity search with queries of a fixed size. This size is predefined, and chosen at index construction time (since it also needs to be the size of the data series in the index).

A straightforward solution for answering such queries would be to use one of the available indexing techniques. However, in order to support (exact) results for variable-length similarity search, we would need to (i) create several distinct indexes, one for each possible query length; and (ii) for each one of these indexes, index all overlapping subsequences (using a sliding window).

Evidently, this is a constraint that penalizes the flexibility needed by analysts, who often times need to analyze patterns of variable lengths [12], [19], [20], [21]. This is true for several applications that produce datasets with a very large number of fixed length data series, on which analysts need to perform a large number of ad hoc similarity search queries of (slightly) different lengths.

Nevertheless, this solution is prohibitively expensive, in both space and time. Space complexity is increased, since we need to index a large number of subsequences for each one of the supported query lengths. If we consider the query length range (in points) $[96, 256]$ on a 20GB (500M points) dataset, we end up with a collection of subsequences (that need to be indexed) 5 orders of magnitude larger than the original dataset! Computational time is significantly increased as well, since we have to construct different indexes for each query length we wish to support.

In the current literature, a technique based on multi-resolution indexes [19], [12] has been proposed in order to mitigate this explosion in size, by creating a smaller number of distinct indexes and performing more post-processing. Nonetheless, this solution works exclusively for *non* Z-normalized series[2] (which means that it cannot return results with similar trends, but different absolute values), and thus, renders the solution useless for a wide spectrum of applications. Besides, it only mitigates the problem, since it still leads to a space explosion (albeit, at a lower rate), and therefore, it is not scalable, either.

In this work, we propose *ULISSE* (ULtra compact Index for variable-length Similarity SEarch in data series), which is the first single-index solution that supports fast answering of variable-length similarity search queries for both non Z-normalized and Z-normalized data series collections. *ULISSE* produces exact (i.e., correct) results, and is based on the following key idea: a data structure that indexes data series of length ℓ, already contains all the information necessary for reasoning about any subsequence of length $\ell' < \ell$ of these series. Therefore, the problem of enabling a data series index to answer queries of variable-length, becomes a problem of how to reorganize this information that already exists in the index. To this effect, *ULISSE* proposes a new summarization technique that is able to represent contiguous and overlapping subsequences, leading to succinct, yet powerful summaries: it combines the representation of several subsequences within a single summary, and enables fast (approximate and exact) similarity search for variable-length queries.

II. Related Work

The literature includes several techniques for data series indexing [9], [11], [23], [5], which share a common limitation: they can only answer queries of a fixed, predetermined length, which has to be decided before the index creation.

In order to avoid this problem, two methods that allow one to answer queries of different lengths have been proposed [12], [19]. These techniques, following the search by prefix paradigm [9], try to mitigate the search space explosion problem by constructing a carefully selected, limited number of distinct indexes for series different lengths. When a query arrives, a subset of the constructed indexes are consulted and their results are combined, in order to generate the final answer to the query. Nevertheless, these techniques only work for non Z-normalized series, which is a show-stopper for several applications.

In contrast, our approach uses a single index that is able to answer similarity search queries of variable length, and works for both Z-normalized and non Z-normalized series.

Even though recent works have shown that sequential scans can be performed efficiently [24], such techniques are mostly applicable when the dataset consists of a single, very long data series, and queries are looking for potential matches in small subsequences of this long data series. Such approaches, in general, do not provide any benefit when the dataset is composed of a large number of small data series, like in our case. Therefore, indexing is required in order to efficiently support data exploration tasks, which involve ad-hoc queries, i.e., the query workload is not known in advance.

[2]Z-normalization transforms a series so that it has a mean value of zero, and a standard deviation of one. This allows similarity search to be effective, irrespective of shifting (i.e., offset translation) and scaling [22].

III. Preliminaries

Let a data series $D = d_1,...,d_{|D|}$ be a sequence of numbers $d_i \in \mathbb{R}$, where $i \in \mathbb{N}$ represents the position in D. We denote the length, or size of the data series D with $|D|$. The subsequence $D_{s,\ell}$=d_s,...,$d_{s+\ell-1}$ of length ℓ, is a contiguous subset of ℓ points of D starting at offset s, where $1 \leq s \leq |D|$ and $1 \leq \ell \leq |D|$. A subsequence is itself a data series. A data series collection, C, is a set of data series.

We say that a data series D is Z-normalized, denoted D^n, when its mean μ is 0 and its standard deviation σ is 1. The normalized version of $D = d_1, ..., d_{|D|}$ is computed as follows: $D^n = \{\frac{d_1-\mu}{\sigma}, ..., \frac{d_{|D|}-\mu}{\sigma}\}$. Z-normalization is an essential operation in several applications, because it allows similarity search irrespective of shifting and scaling [24].

Given two data series $D = d_1, ..., d_{|D|}$ and $D' = d'_1, ..., d'_{|D'|}$ of the same length (i.e., $|D| = |D'|$), we can calculate their Euclidean Distance as follows: $ED(D, D') = \sqrt{\sum_1^{|D|}(d_1 - d'_1)^2}$.

The Piecewise Aggregate Approximation (PAA) of a data series D, $PAA(D) = \{p_1, ..., p_w\}$, represents D in a w-dimensional space by means of w real-valued segments of length s, where the value of each segment is the mean of the corresponding values of D [25]. We denote the first k dimensions of $PAA(D)$, $(k \leq w)$, as $PAA(D)_{1,..,k}$.

The $iSAX$ representation of a data series D, denoted by $iSAX(D, w, |alphabet|)$, is the representation of $PAA(D)$ by w discrete coefficients, drawn from an alphabet of cardinality $|alphabet|$ [11]. The main idea of the $iSAX$ representation, is that the real-value space may be segmented by $|alphabet| - 1$ breakpoints in $|alphabet|$ regions, which are labeled by discrete symbols (e.g., with $|alphabet| = 4$ the available labels may be $\{00, 01, 10, 11\}$).

IV. Proposed approach

The key idea of the *ULISSE* approach is the succinct summarization of *sets* of series, namely, overlapping subsequences. In this section, we present this summarization method.

A. Representing Multiple Subsequences

When we consider, contiguous and overlapping subsequences of different lengths within the range $[\ell_{min}, \ell_{max}]$, we expect the outcome as a bunch of similar series, whose differences are affected by the misalignment and the different number of points.

Given a data series D, and a subsequence length range $[\ell_{min}, \ell_{max}]$, we define the master series as the subsequences of the form $D_{i,min(|D|-i+1,\ell_{max})}$, for each i such that $1 \leq i \leq |D| - (\ell_{min} - 1)$, where $1 \leq \ell_{min} \leq \ell_{max} \leq |D|$. We observe that for any master series of the form $D_{i,\ell'}$, we have that $PAA(D_{i,\ell'})_{1,..,k} = PAA(D_{i,\ell''})_{1,..,k}$ holds for each ℓ'' such that $\ell'' \geq \ell_{min}$, $\ell'' \leq \ell' \leq \ell_{max}$ and $\ell', \ell''\%k = 0$. Therefore, by computing only the PAA of the master series in D, we are able to represent the PAA prefix of any subsequence of D.

When we zero-align the PAA summaries of the master series, we compute the minimum and maximum PAA values (over all the subsequences) for each segment: this forms what we call an *Envelope*. (When the length of a master series is not a multiple of the PAA segment length, we compute the PAA coefficients of the longest prefix, which is multiple of a segment.) We call *containment area* the space in between the segments that define the Envelope.

B. PAA Envelope

We now formalize the concept of the *Envelope*, introducing a new series representation. We denote by L and U the PAA coefficients, which delimit the lower and upper parts, respectively, of a containment area. Furthermore, we introduce a parameter γ, which permits to select the number of master series we represent by the Envelope.

We refer to it using the following signature: $paaENV_{[D,\ell_{min},\ell_{max},a,\gamma,s]} = [L,U]$. It delimits the containment area generated by the PAA coefficients of the master series.

If we want to build an Envelope, containing Z-normalized sequences, we need also to take into account the shifted coefficients of the Z-normalized subsequences, which are not master series. Hence, each PAA coefficient of a master series will be represented by the set of values resulting from the Z-normalizations. Even though this additional step adds complexity, it turns out that we can follow the same procedure as before (i.e., with non Z-normalized sequences), in order to get the Envelope for the Z-normalized sequences, as well.

C. Indexing the Envelopes

Here, we define the procedure used to index the Envelopes.

Given a $paaENV$, we can translate its PAA extremes into the corresponding iSAX representation: $uENV_{paaENV_{[D,\ell_{min},\ell_{max},a,\gamma,s]}} = [iSAX(L), iSAX(U)]$, where $iSAX(L)$ ($iSAX(U)$) is the vector of the minimum (maximum) PAA coefficients of all the segments corresponding to the subsequences of D. The *ULISSE* Envelope, $uENV$, represents the principal building block of the *ULISSE* index.

We are now ready to construct the *ULISSE* tree index. Each *ULISSE* internal node stores the Envelope $uENV$ that represents all the sequences in the subtree rooted at that node. Leaf nodes contain several Envelopes, which by construction have the same $iSAX(L)$. On the contrary, their $iSAX(U)$ varies, since it get updated with every new insertion in the node. (Note that, inserting by keeping the same $iSAX(U)$ and updating $iSAX(L)$ represents a symmetric and equivalent choice.)

V. Similarity Search with *ULISSE*

In this section, we present the building blocks of the similarity search algorithms we developed for the *ULISSE* index.

A. Lower Bounding Euclidean Distance

The iSAX representation allows the definition of a distance function that lower bounds the true Euclidean distance [11]. This function compares the PAA coefficients of the first data series, D, against the breakpoints that delimit the regions of the second data series, D', where $|D| = |D'|$. As a result, we get a lower bounding function $mindist_{PAA\_iSAX}(PAA(D), iSAX(D'))$. Then, it holds that $mindist_{PAA\_iSAX}(PAA(D), iSAX(D')) \leq ED(D, D')$.

In the case of the *ULISSE* index, we cannot use the $mindist$ function described above. Since our index contains Envelope representations, we need a new lower bounding function, in order to lower bound the distances between a data series Q (i.e., the query), and a set of subsequences, whose iSAX symbols are described by the Envelope $uENV_{paaENV_{[D,\ell_{min},\ell_{max},a,\gamma,s]}} = [iSAX(L), iSAX(U)]$. This leads to a new lower bounding function, $mindist_{ULiSSE}(PAA(Q), uENV_{paaENV_{[D,\ell_{min},\ell_{max},a,\gamma,s]}})$.

B. Approximate search

Similarity search performed on *ULISSE* index relies on the $mindist_{ULiSSE}()$ lower bounding function to prune the search space. This allows to navigate the tree in order, visiting first the most promising nodes. Note that in the comparison, we use the largest prefix of the query, which is a multiple of the PAA segment length, used at the index building stage. As soon as a leaf node is discovered, we can load the raw data series pointed by the Envelopes in the leaf. Each time we compute the true Euclidean distance between the series in a leaf, the best-so-far distance (bsf) is updated, along with a vector containing the k best matches, where k refers to the k nearest neighbors (for a k-NN query). Since priority is given to the most promising nodes, we can terminate our visit, when at the end of a leaf visit the k bsf's have not improved.

C. Exact search

Note that the approximate search described above may not visit leaves that contain answers better than the approximate answers already identified, and therefore, it will fail to produce exact, correct results.

The exact nearest neighbor search algorithm finds the k sequences with the absolute smallest distances to the query. In this case, the search algorithm may visit several leaves: the process stops after it has either visited, or pruned (when the lower bounding distance to the node is greater than the bsf) all the nodes of the index, guaranteeing the correctness of the results.

VI. Experimental Evaluation

We implemented the *ULISSE* algorithms (indexing and query answering) in C, and used the gcc 4.8.2 compiler. All experiments were run on an Intel Xeon E5-2403 (4 cores @ 1.9GHz), using the x86_64 GNU/Linux OS environment.

For the experiments, we used synthetic and real data (but in the interest of space only report results with the synthetic

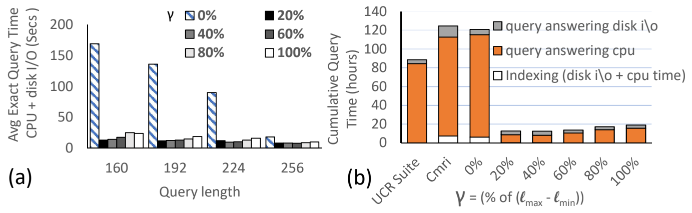

Fig. 1. Query answering time performance, varying γ on non Z-normalized data series. *(a) ULISSE* average query time (CPU + disk I/O). *(b) ULISSE* average query disk I/O time. *(b)* Comparison of *ULISSE* to other techniques (cumulative indexing + query answering time).

data). We produced the synthetic, random walk datasets with a generator, where a random number is drawn from a Gaussian distribution $N(0, 1)$, then at each time point a new number is drawn from this distribution and added to the value of the last number. This kind of data generation has been extensively used in the past [26], and has been shown to effectively model real-world financial data [9]. The synthetic datasets contain 1.25 billion points of size 4 bytes each, which is the equivalent of 23 bilion overlapping series of lengths between $[160, 256]$.

We record the average *CPU time*, *disk I/O* (time to fetch data from disk (Total time - CPU time)), and *pruning power* (percentage of the total number of Envelopes in the index that do not need to be read), for *100* queries, extracted from the datasets with the addition of Gaussian noise. For each index used, the *building time* and the relative *size* are also reported. We compare *ULISSE* with *UCR suite* [24] the non index-based state-of-the-art technique for answering similarity search queries, also for Z-normalized sequences. Concerning the competitor indexing techniques, the state-of-the-art is the Compact Multi Resolution Index [12] *CMRI*, which works only for *non* Z-normalized series.

Varying γ. We first present results for similarity search queries on *ULISSE* when we vary γ, ranging from *0* to its maximum value, i.e., $\ell_{max} - \ell_{min}$. In Figure 1, we report the results concerning non Z-normalized series We observe that grouping contiguous and overlapping subsequences under the same summarization (Envelope) by increasing γ, affects positively the performance of index construction, as well as query answering.

The latter may seem counterintuitive, since inserting more master series into a single $Envelope$ is likely to generate large containment areas, which are not tight representations of the data series. On the other hand, it leads to an overall number of $Envelopes$ that is several orders of magnitude smaller than the one for $\gamma = 0\%$. In this last case, when $\gamma = 0$, the algorithm inserts in the index as many records as the number of master series present in the dataset (*485*M).

Comparison to Alternatives. In Figure 1(b) we show the cumulative time performance (i.e., $4,000$ queries in total), comparing *ULISSE*, *CMRI*, and *UCR Suite*. Note that in this experiment, *ULISSE* indexing time is negligible w.r.t. the query answering time. *ULISSE*, outperforms both *UCR Suite* and *CMRI*, achieving a speed-up of up to 12x.

VII. Conclusions

In this work, we proposed *ULISSE*, the first index able to answer similarity search queries of variable-length. We experimentally evaluated, our indexing and similarity search algorithms, demonstrating the effectiveness and efficiency of the proposed solution.

References

[1] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco, "Practical data prediction for real-world wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, 2015.
[2] T. Palpanas, "Data series management: The road to big sequence analytics," *SIGMOD Rec.*, 2015.
[3] ESA. SENTINEL-2 mission. [Online]. Available: https://sentinel.esa.int/web/sentinel/missions/sentinel-2
[4] K. Zoumpatianos and T. Palpanas., "Data series management: Fulfilling the need for big sequence analytics." in *ICDE*, 2018.
[5] Y. Wang, P. Wang, J. Pei, W. Wang, and S. Huang, "A data-adaptive and dynamic segmentation index for whole matching on time series," *PVLDB*, 2013.
[6] K. Zoumpatianos, S. Idreos, and T. Palpanas, "Indexing for interactive exploration of big data series," in *SIGMOD*, 2014.
[7] T. Palpanas, "Big sequence management: A glimpse of the past, the present, and the future," in *SOFSEM*, 2016.
[8] ——, "The parallel and distributed future of data series mining," in *High Performance Computing & Simulation (HPCS)*, 2017.
[9] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *SIGMOD*, 1994.
[10] E. J. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle, "Indexing large human-motion databases," in *VLDB*, 2004.
[11] J. Shieh and E. J. Keogh, "*i*sax: indexing and mining terabyte sized time series," in *KDD*, 2008.
[12] S. Kadiyala and N. Shiri, "A compact multi-resolution index for variable length queries in time series databases," *KAIS*, 2008.
[13] A. Camerra, T. Palpanas, J. Shieh, and E. J. Keogh, "isax 2.0: Indexing and mining one billion time series," in *ICDM 2010*, 2010.
[14] 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 isax2+," *KAIS*, 2014.
[15] K. Zoumpatianos, S. Idreos, and T. Palpanas, "RINSE: interactive data series exploration with ADS+," *PVLDB*, 2015.
[16] ——, "ADS: the adaptive data series index," *VLDB J.*, 2016.
[17] D. E. Yagoubi, R. Akbarinia, F. Masseglia, and T. Palpanas, "Dpisax: Massively distributed partitioned isax," in *ICDM*, 2017, pp. 1135–1140.
[18] H. Kondylakis, N. Dayan, K. Zoumpatianos, and T. Palpanas, "Coconut: A scalable bottom-up approach for building data series indexes," in *PVLDB*, 2018.
[19] T. Kahveci and A. Singh, "Variable length queries for time series data," in *ICDEF*, 2001.
[20] M. Linardi, Y. Zhu, T. Palpanas, and E. J. Keogh, "Matrix profile X: Valmod - scalable discovery of variable-length motifs in data series," 2018.
[21] ——, "Valmod: A suite for easy and exact detection of variable length motif in data series," 2018.
[22] E. J. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Min. Knowl. Discov.*, 2003.
[23] Y. Bu, T. wing Leung, A. W. chee Fu, E. Keogh, J. Pei, and S. Meshkin, "Wat: Finding top-k discords in time series database," in *SDM*, 2007.
[24] T. R. et al., "Searching and mining trillions of time series subsequences under dynamic time warping," in *SIGKDD*, 2012.
[25] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *KAIS*, vol. 3, 2000.
[26] K. Zoumpatianos, Y. Lou, T. Palpanas, and J. Gehrke, "Query workloads for data series indexes," in *KDD*, 2015.