Effective and Efficient Variable-Length Data Series Analytics

Michele Linardi
Supervised by: Themis Palpanas
LIPADE, Université de Paris
michele.linardi@parisdescartes.fr

ABSTRACT

In the last twenty years, data series similarity search has emerged as a fundamental operation at the core of several analysis tasks and applications related to data series collections. Many solutions to different mining problems work by means of similarity search. In this regard, all the proposed solutions require the prior knowledge of the series length on which similarity search is performed. In several cases, the choice of the length is critical and sensibly influences the quality of the expected outcome. Unfortunately, the obvious brute-force solution, which provides an outcome for all lengths within a given range is computationally untenable. In this Ph.D. work, we present the first solutions that inherently support scalable and variable-length similarity search in data series, applied to sequence/subsequences matching, motif and discord discovery problems. The experimental results show that our approaches are up to orders of magnitude faster than the alternatives. They also demonstrate that we can remove the unrealistic constraint of performing analytics using a predefined length, leading to more intuitive and actionable results, which would have otherwise been missed.

1. INTRODUCTION

Data series (i.e., ordered sequences of points) are one of the most common data types[1], present in almost every scientific and social domain (such as meteorology, astronomy, chemistry, medicine, neuroscience, finance, agriculture, entomology, sociology, smart cities, marketing, operation health monitoring, human action recognition and others) [20].

Once the data series have been collected, the domain experts face the arduous tasks of processing and analyzing them [30, 6] in order to gain insights, e.g., by identifying similar patterns, and performing classification, or clustering. A core operation that is part of all these analysis tasks is similarity search, which has attracted lots of attention because of its importance [2]. Nevertheless, all existing scalable and index-based similarity search techniques are restricted in that they only support queries of a fixed length, and they require that this length is chosen at index construction [10, 24, 1, 25, 28, 29, 26, 21, 11]. The same observation holds for techniques proposed to discover motifs [12] and discords (i.e., anomalous subsequences) [27]: they all assume a fixed sequence length, which has to be predefined.

Evidently, this is a constraint that penalizes the flexibility needed by analysts, who often times need to analyze patterns of slightly different lengths (within a given data series collection) [7, 8, 5, 17, 16]. For example, in the *SENTINEL-2* mission data, oceanographers are interested in searching for similar coral bleaching patterns[2] of different lengths; at Airbus[3] engineers need to perform similarity search queries for patterns of variable length when studying aircraft take-offs and landings [19]; and in neuroscience, analysts need to search in Electroencephalogram (EEG) recordings for Cyclic Alternating Patterns (CAP) of different lengths (duration), in order to get insights about brain activity during sleep [22].

In our work, we focus on three core problems that are based on similarity search: subsequence matching, and motif and discord discovery, organized under the ULISSE and MAD methods:

1. ULISSE (ULtra compact Index for variable-length Similarity SEarch in data series) is the first indexing technique that supports variable-length subsequence matching for non Z-normalized and Z-normalized data series [15, 13, 14].

2. MAD (Motif and Discord discovery framework) implements two novel algorithms for variable-length motif and discord discovery in large data series [17, 4, 16].

2. VARIABLE-LENGTH ANALYTICS

In this section, we describe our proposed approaches to the aforementioned problems. In the next part we describe the notions and the elements used in our solutions.

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. Z-normalization is an essential operation in several applications, because it allows similarity search irrespective of shifting and scaling [5]. The Piecewise Aggregate Approximation (PAA) of a data series D, $PAA(D) = \{p_1, ..., p_w\}$,

[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, position in genome sequences in biology, etc.). We use the terms *data series*, *time series*, and *sequence* interchangeably.

Proceedings of the VLDB 2019 PhD Workshop, August 26th, 2019. Los Angeles, California.

[2] http://www.esa.int/Our_Activities/Observing_the_Earth

[3] http://www.airbus.com/

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 [9]. 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 $|alphabet|$ [24]. 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 $\{00, 01, 10, 11\}$).

2.1 Subsequence Matching

The subsequence matching problem is defined as follows:

Given a data series collection $C = \{D^1, ..., D^C\}$, a series length range $[\ell_{min}, \ell_{max}]$, a query data series Q, where $\ell_{min} \leq |Q| \leq \ell_{max}$, and $k \in \mathbb{N}$, we want to find the set $R = \{D^i_{o,\ell} | D^i \in C \wedge \ell = |Q| \wedge (\ell + o - 1) \leq |D^i|\}$, where $|R| = k$. We require that $\forall D^i_{o,\ell} \in R \nexists D^{i'}_{o',\ell'}$ $s.t.$ $dist(D^{i'}_{o',\ell'}, Q) < dist(D^i_{o,\ell}, Q)$, where $\ell' = |Q|$, $(\ell' + o' - 1) \leq |D^{i'}|$ and $D^{i'} \in C$. We informally call R, the k *nearest neighbors* set of Q. Given two generic series of the same length, namely D and D' the function $dist(D, D')$ can be Euclidean Distance or Dynamic Time Warping.

Variable Length Subsequences. In a data series, 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 that is multiple of a segment.) We call *containment area* the space in between the segments that define the *Envelope*.

PAA Envelope. We 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.

Indexing 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

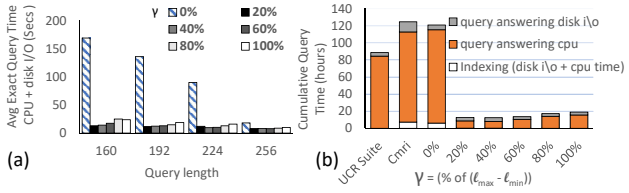

Figure 1: Query answering time performance, varying γ on non Z-normalized data series. *(a) ULISSE* average query time (CPU + disk I/O). *(b) ULISSE* average query disk I/O time. *(b)* Comparison of *ULISSE* to other techniques (cumulative indexing + query answering time).

Envelope uENV represents the principal building block of the *ULISSE* Index. In details, *ULISSE* is a tree structure, where each internal node stores the *Envelope uENV* representing 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. Each *Envelope* in leaf nodes point the the represented sequences in the original data series collection.

Approximate Subsequence Matching. Subsequence matching 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. 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 or DTW 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. 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.

Exact Subsequence Matching. 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 we propose 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.

Experiments. To evaluate *ULISSE*, we used synthetic and real data (but in the interest of space we only report results with the synthetic data). We record the average *CPU time*, *disk I/O* (time to fetch data from disk (Total time - CPU time)), for *100* queries, extracted from the datasets with the addition of Gaussian noise. We compare *ULISSE* with *UCR suite* [5] the non index-based state-of-the-art technique for answering similarity search queries. Concerning the competitor indexing techniques, the state-of-the-art is the Compact Multi Resolution Index [7] *CMRI*.

In Figure 1, we present results for subsequence matching queries on *ULISSE* when we vary γ, ranging from *0* to its maximum value in this dataset, 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 *Envelope* that is several orders of magnitude smaller than the one for $\gamma = 0\%$, where only a single master series is represented by each *Envelope*.

2.2 Motif and Discord Discovery

Motif and Discord are data mining primitives that represent frequent and rare (anomalous) patterns, respectively. Given a data series D, they are defined as follows:

- Data series motif: $D_{a,\ell}$ and $D_{b,\ell}$ is a motif pair iff $dist(D_{a,\ell}, D_{b,\ell}) \leq dist(D_{i,\ell}, D_{j,\ell})\ \forall i, j \in [1, 2, ..., |D| - \ell + 1]$, where $a \neq b$ and $i \neq j$, and dist is a function that computes the z-normalized Euclidean distance between the input subsequences.

- Data series discord: We call the k subsequences of D, with the k largest distances to their m^{th} Nearest Neighbor (according the Euclidean distance), the *Top-k* m^{th} discords.

Variable length motif and discord discovery. We provide solutions to the following problems:

- Variable-Length Motif Discovery: Given a data series D and a subsequence length-range $[\ell_{min}, ..., \ell_{max}]$, we want to find the data series motif pairs of all lengths in $[\ell_{min}, ..., \ell_{max}]$, occurring in D.

- Variable-Length *Top-k* m^{th} Discord Discovery: Given a data series D, a subsequence length-range $[\ell_{min}, ..., \ell_{max}]$ and the parameters $a, b \in \mathbb{N}^+$ we want to enumerate the *Top-k* m^{th} discords for each $k \in \{1, .., a\}$ and each $m \in \{1, .., b\}$, and for all lengths in $[\ell_{min}, ..., \ell_{max}]$, occurring in D.

Fixed length motif and discord discovery. The state-of-the art algorithm for fixed length motif and discord discovery [3] requires the user to define the length of the desired motif or discord. This mining operation is supported by computation of the *Matrix profile*, which is a meta data series storing the z-normalized Euclidean distance between each subsequence and its nearest neighbor. The Matrix profile does not only derive the motif, but also ranks and filters out the other pairs, giving also a convenient and graphical representation of their occurrences and proximity. Unfortunately, this technique comes with an important shortcoming: it does not provide an effective solution for trying several different motif lengths. Therefore, the analyst is forced to run the algorithm using all possible lengths in a range of interest, and rank the various motifs discovered, picking eventually the patterns that contain the desired insight. Clearly, this possibility is not optimal for at least two reasons: the scalability, since finding motif of one fixed length takes $O(|D|^2)$ time, and also because it does not provide an effective way to compare motifs of different lengths.

MAD Framework. Our framework for Variable Length Motif and Discord Discovery (MAD) works by applying an *incremental computing* strategy, which aims to prune unnecessary distance computations for larger motif and discord lengths. Hence, given a data series D, we compute the Matrix profile using the smallest subsequence length, namely ℓ_{min}, within a specified input range $[\ell_{min}, \ell_{max}]$. The key idea of our approach is to minimize the work that needs to be done for succeeding subsequence lengths ($\ell_{min} + 1$, $\ell_{min} + 2$, ..., ℓ_{max}).

Matrix Profile Computation. We start the computation of the Matrix profile, considering all the contiguous subsequences of length ℓ_{min}, computing for each one the *Distance profile* in $O(|D|)$ time. This latter is a vector that contains the z-normalized Euclidean distances between a fixed subsequence and all the other in D (excluding trivial matches).

Lower Bound Subsequences of Different Length. We moreover introduce a new lower bounding distance [17], which lower bounds (is always smaller than) the true Euclidean distances between subsequences longer than ℓ_{min}. We initially compute this lower bound using the true Euclidean distances computation of subsequences with length ℓ_{min}. For the larger lengths, we update the lower bound, considering only the variation generated by the trailing points in the longer subsequences. This measure enjoys an important property: if we rank the subsequences according to this measure, the same rank will be preserved along all the lower bound updates for the subsequences of greater length. We exploit this property, in order to prune computations.

Pruning the Search Space. Once we compute motif and discords, with length greater than ℓ_{min}, instead of computing from scratch each distance profile, we update the true distances (in constant time) of the subsequences that have the p smallest lower bounding distances (computed in the previous step). These distances form what we call *partial distance profile*. In each partial distance profile, we also update the lower bound. After this operation, we may have two cases: if in a new computed distance profile the minimum true distance (*minDist*) is shorter than the maximum lower bound (*maxLB*), we know that no distances among those not computed can be smaller than minDist. In this case, a partial distance profile becomes a *valid distance profile*. On the other hand, when *maxLB* is smaller than *minDist*, this latter is not guaranteed to be the nearest neighbor distance. For discord discovery, we need to test this condition for the m smallest true distances in the partial distance profile. In this case a valid (partial) distance profile must contain the true m^{th} best match distances, which are smaller than, or equal to *maxLB*.

Exact Motif and Discord Discovery. Once the partial distance profiles are computed, we pick the absolute smallest lower bounding value from all the non-valid distance profiles, namely *minLBAbs* (if any). Therefore, the global minimum (true) distance of all the valid (partial) distance profiles, which is smaller than *minLBAbs* is guaranteed to be the distance between the motif pair subsequences. Symmetrically, we consider the valid (partial) distance profiles to find the true m^{th} best match distances, which are the greatest nearest neighbor distances that are larger than *maxLBAbs*. This latter is the largest lower bounding distance of the non-valid distance profiles.

In the motif discovery task, if no nearest neighbor distance is smaller than *minLBAbs*, we recompute only the distance profiles that have the *maxLB* distance smaller than the smallest true distance computed.

On the other hand, for discord discovery, if no true nearest neighbor distances are found we need to iterate the non

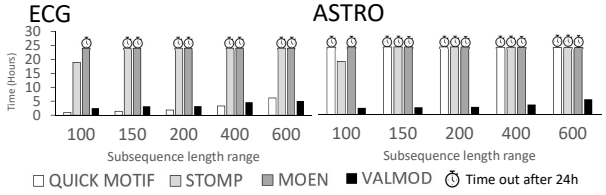

Figure 2: Time over motif length ranges (default ℓ_{min}=*1024*, data series length= *0.5M*.

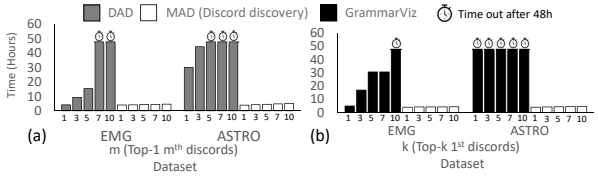

Figure 3: (a) $Top-1$ m^{th} discords discovery, and (b) $Top-k$ 1^{st} discords discovery time performance.

valid (partial) distance profiles, which contain the *maxLB* distance greater than the largest m^{th} best match distance.

We keep extracting in this manner the motif and the discord subsequences of each length, until ℓ_{max}.

Motif Discovery Experimental Evaluation. To benchmark the MAD framework, we used several different real datasets. Concerning the motif discovery problem, the competitors we considered are: QUICKMOTIF [12], STOMP [3], and MOEN [18]. We report in Figure 2 a sample of the experiments we conducted (detailed experimental results on several datasets are reported elsewhere [17]). Here, we show the results of MAD, which finds motifs in different real datasets. In the plots, we report the total execution time varying motif length ranges. From this experiment, we observe that VALMOD maintains a good and stable performance across datasets and parameter settings, quickly producing results, even in cases where the competitors do not terminate within a reasonable amount of time.

Discord Discovery Experimental Evaluation. We identify two state-of-the-art competitors to compare to our approach, the Motif And Discord (MAD) framework. The first one, DAD (Disk aware discord discovery) [27], implements an algorithm suitable to enumerate the *fixed-length* $Top-1$ m^{th} discords. The second approach, GrammarViz [23], is the most recent technique, which discovers *Top-k* 1^{st} discords. In Figure 3.(a) we report the results of $Top-1$ m^{th} discord discovery, varying m. We note that MAD gracefully scales over the number of discords to enumerate and is up to one order of magnitude faster than DAD. In Figure 3.(b), we show the result of $Top-k$ 1^{st} discords discovery. Once again, MAD scales better over the number of discovered discords, as its execution time remains almost constant. A different trend is observed for GrammarViz, whose performance significantly deteriorates as k increases.

3. CONCLUSIONS

Even though much effort has been dedicated for developping techniques for data series analytics, existing solutions for subsequence matching, motif and discord discovery are limited to fixed length queries/results. In this Ph.D. work, we propose the first scalable solutions to the variable-length version of these problems: *ULISSE* is the first index that supports variable-length subsequence matching over both Z-normalized and non Z-normalized sequences [15, 13, 14], while MAD is the first framework that implements variable-length motif and discord discovery [17, 4, 16].

References

[1] A. Camerra, T. Palpanas, J. Shieh, and E. J. Keogh. isax 2.0: Indexing and mining one billion time series. In *ICDM 2010*, 2010.

[2] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim. The lernaean hydra of data series similarity search: An experimental evaluation of the state of the art. *PVLDB*, 12(2), 2018.

[3] C. M. Y. et al. Matrix profile I: all pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *ICDM*, 2016.

[4] M. L. et al. Matrix profile goes mad: Variable-length motif and discord discovery in data series. In *Under Submission 2019*.

[5] T. R. et al. Searching and mining trillions of time series subsequences under dynamic time warping. In *SIGKDD*, 2012.

[6] A. Gogolou, T. Tsandilas, T. Palpanas, and A. Bezerianos. Progressive similarity search on time series data. In *BigVis, in conjunction with EDBT/ICDT*, 2019.

[7] S. Kadiyala and N. Shiri. A compact multi-resolution index for variable length queries in time series databases. *KAIS*, 2008.

[8] T. Kahveci and A. Singh. Variable length queries for time series data. In *ICDEF*, 2001.

[9] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *KAIS*, 3, 2000.

[10] E. J. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle. Indexing large human-motion databases. In *VLDB*, 2004.

[11] H. Kondylakis, N. Dayan, K. Zoumpatianos, and T. Palpanas. Coconut: A scalable bottom-up approach for building data series indexes. In *PVLDB*, 2018.

[12] Y. Li, L. H. U, M. L. Yiu, and Z. Gong. Quick-motif: An efficient and scalable framework for exact motif discovery ICDE, 2015.

[13] M. Linardi and T. Palpanas. Scalable data series subsequence matching with ulisse. In *Under Submission 2019*.

[14] M. Linardi and T. Palpanas. ULISSE: ULtra compact Index for Variable-Length Similarity SEarch in Data Series. In *ICDE 2018*.

[15] M. Linardi and T. Palpanas. Scalable, variable-length similarity search in data series: The ULISSE approach. *PVLDB*, 11(13):2236–2248, 2018.

[16] M. Linardi, Y. Zhu, T. Palpanas, and E. J. Keogh. VALMOD: A suite for easy and exact detection of variable length motifs in data series. In *SIGMOD Conference 2018*.

[17] M. Linardi, Y. Zhu, T. Palpanas, and E. J. Keogh. Matrix profile X: Valmod - scalable discovery of variable-length motifs in data series. In *SIGMOD*, 2018.

[18] A. Mueen and N. Chavoshi. Enumeration of time series motifs of all lengths. *Knowl. Inf. Syst.*, 2015.

[19] A. G. H. of Operational Intelligence Department Airbus. Personal communication., 2017.

[20] T. Palpanas. Data series management: The road to big sequence analytics. *SIGMOD Rec.*, 2015.

[21] B. Peng, P. Fatourou, and T. Palpanas. Paris: The next destination for fast data series indexing and query answering. In *IEEE Big Data*, 2018.

[22] A. Rosa, L. Parrino, and M. Terzano. Automatic detection of cyclic alternating pattern (cap) sequences in sleep: preliminary results. *Clinical Neurophysiology*, 1999.

[23] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandhi, A. P. Boedihardjo, C. Chen, and S. Frankenstein. Time series anomaly discovery with grammar-based compression. In *EDBT*, 2015.

[24] J. Shieh and E. J. Keogh. *i*sax: indexing and mining terabyte sized time series. In *KDD*, 2008.

[25] 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.

[26] D. E. Yagoubi, R. Akbarinia, F. Masseglia, and T. Palpanas. Dpisax: Massively distributed partitioned isax. In *ICDM*, 2017.

[27] D. Yankov, E. J. Keogh, and U. Rebbapragada. Disk aware discord discovery: finding unusual time series in terabyte sized datasets. *Knowl. Inf. Syst.*, 2008.

[28] K. Zoumpatianos, S. Idreos, and T. Palpanas. RINSE: interactive data series exploration with ADS+. *PVLDB*, 2015.

[29] K. Zoumpatianos, S. Idreos, and T. Palpanas. ADS: the adaptive data series index. *VLDB J.*, 2016.

[30] K. Zoumpatianos and T. Palpanas. Data series management: Fulfilling the need for big sequence analytics. In *ICDE*, 2018.