Data Series Progressive Similarity Search with Probabilistic Quality Guarantees

Anna Gogolou, Theophanis Tsandilas, Karima Echihabi, Anastasia Bezerianos, Themis Palpanas

▶ **To cite this version:**

Anna Gogolou, Theophanis Tsandilas, Karima Echihabi, Anastasia Bezerianos, Themis Palpanas. Data Series Progressive Similarity Search with Probabilistic Quality Guarantees. ACM SIGMOD International Conference on Management of Data, Jun 2020, Portland, United States. pp.1857-1873, 10.1145/3318464.3389751 . hal-02560760

HAL Id: hal-02560760
https://hal.science/hal-02560760

Submitted on 2 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Data Series Progressive Similarity Search with Probabilistic Quality Guarantees

Anna Gogolou
Université Paris-Saclay, Inria, CNRS, LRI & LIPADE, University of Paris
anna.gogolou@inria.fr

Theophanis Tsandilas
Université Paris-Saclay, Inria, CNRS, LRI
theophanis.tsandilas@inria.fr

Karima Echihabi
IRDA, Rabat IT Center, ENSIAS, Mohammed V University
karima.echihabi@gmail.com

Anastasia Bezerianos
Université Paris-Saclay, CNRS, Inria, LRI
anastasia.bezerianos@lri.fr

Themis Palpanas
LIPADE, University of Paris & French University Institute (IUF)
themis@mi.parisdescartes.fr

ABSTRACT

Existing systems dealing with the increasing volume of data series cannot guarantee interactive response times, even for fundamental tasks such as similarity search. Therefore, it is necessary to develop analytic approaches that support exploration and decision making by providing progressive results, before the final and exact ones have been computed. Prior works lack both efficiency and accuracy when applied to large-scale data series collections. We present and experimentally evaluate a new probabilistic learning-based method that provides quality guarantees for progressive Nearest Neighbor (NN) query answering. We provide both initial and progressive estimates of the final answer that are getting better during the similarity search, as well suitable stopping criteria for the progressive queries. Experiments with synthetic and diverse real datasets demonstrate that our prediction methods constitute the first practical solution to the problem, significantly outperforming competing approaches.

KEYWORDS

Data Series, Similarity Search, Progressive Query Answering

ACM Reference Format:
Anna Gogolou, Theophanis Tsandilas, Karima Echihabi, Anastasia Bezerianos, and Themis Palpanas. 2020. Data Series Progressive Similarity Search with Probabilistic Quality Guarantees. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3318464.3389751

SIGMOD'20, June 14–19, 2020, Portland, OR, USA
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA*, https://doi.org/10.1145/3318464.3389751.

1 INTRODUCTION

Data Series. Data series are ordered sequences of values measured and recorded from a wide range of human activities and natural processes [66], such as seismic activity, or electroencephalography (EEG) signal recordings. The analysis of such sequences[1] is becoming increasingly challenging as their sizes often grow to multiple terabytes [7, 64].

Data series analysis involves pattern matching [54, 91], anomaly detection [10, 11, 17, 24], frequent pattern mining [56, 72], clustering [48, 73, 74, 86], and classification [19]. These tasks rely on *data series similarity*. The data-mining community has proposed several techniques, including many similarity measures (or distance measure algorithms), for calculating the distance between two data series [26, 60], as well as corresponding indexing techniques and algorithms [30, 65], in order to address scalability challenges.

Data Series Similarity. We observe that data series similarity is often domain- and visualization-dependent [8, 37], and in many situations, analysts depend on time-consuming manual analysis processes. For example, neuroscientists manually inspect the EEG data of their patients, using visual analysis tools, so as to identify patterns of interest [37, 45]. In such cases, it is important to have techniques that operate within interactive response times [63], in order to enable analysts to complete their tasks easily and quickly.

In the past years, several visual analysis tools have combined visualizations with advanced data management and analytics techniques (e.g., [51, 71]), albeit not targeted to data series similarity search. Moreover, we note that even

[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 (angle in radial profiles, frequency in infrared spectroscopy, etc.). We use the terms *time series*, *data series*, and *sequence* interchangeably.

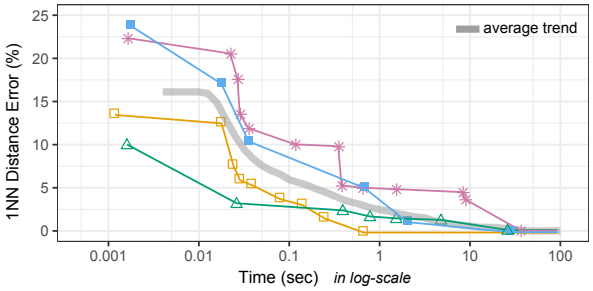

Figure 1: Progression of 1-NN distance error (Euclidean dist.) for 4 example queries (seismic dataset), using iSAX2+ [15]. The points in each curve represent approximate (intermediate points) or exact answers (last point) given by the algorithm. Lines end when similarity search ends. Thick grey line represents average trend over a random sample of 100 queries.

though the data series management community is focusing on scalability issues, the state-of-the-art indexes currently used for scalable data series processing [15, 50, 54, 85, 92] are still far from achieving interactive response times [29, 30].

Progressive Results. To allow for interactive response times when users analyze large data series collections, we need to consider progressive and iterative visual analytics approaches [6, 38, 79, 89]. Such approaches provide progressive answers to users' requests [35, 61, 77], sometimes based on algorithms that return quick approximate answers [25, 34]. Their goal is to support exploration and decision making by providing progressive (i.e., intermediate) results, before the final and exact ones have been computed.

Most of the above techniques consider approximations of aggregate queries on relational databases, with the exception of Ciaccia et al. [20, 21], who provide a probabilistic method for assessing how far an approximate answer is from the exact answer. Nevertheless, these works do not consider data series that are high-dimensional: their dimensionality (i.e., number of points in the series [30]) ranges from several hundreds to several thousands. We note that the framework of Ciaccia et al. [20, 21] does not explicitly target progressive similarity search. Furthermore, the approach has only been tested on datasets with up to 275K vectors with dimensionality of a few dozen, while we are targeting data series vectors in the order of hundreds of millions with dimensionality in the order of hundreds. Our experiments show that the probabilistic estimates that their methods [20, 21] provide are inaccurate and cannot support progressive similarity search.

In this study, we demonstrate the importance of providing progressive similarity search results on large time series collections. Our results show that there is a gap between the time the 1st Nearest Neighbour (1-NN) is found and the time when the search algorithm terminates. In other words, users often wait without any improvement in their answers. We further show that high-quality approximate answers are found very early, e.g., in less than one second, so they can support highly interactive visual analysis tasks.

Figure 1 presents the approximate results of the iSAX2+ index [15] for four example queries on a 100M data series collection with seismic data [36], where we show the evolution of the approximation error as a percentage of the exact 1-NN distance. We observe that the algorithm provides approximate answers within a few milliseconds, and those answers gradually converge to the exact answer, which is the distance of the query from the 1-NN. Interestingly, the 1-NN is often found in less than 1 sec (e.g., see yellow line), but it takes the search algorithm much longer to verify that there is no better answer and terminate. This finding is consistent with previously reported results [21, 38].

Several similarity-search algorithms, such as the iSAX2+ index [15] and the DSTree [85] (the two top performers in terms of data series similarity search [30]), provide very quick approximate answers. In this paper, we argue that such algorithms can be used as the basis for supporting progressive similarity search. Unfortunately, these algorithms do not provide any guarantees about the quality of their approximate answers, while our goal is to provide such guarantees.

Proposed Approach. We develop the first progressive approaches for data series similarity search with probabilistic quality guarantees, which are scalable to very large data series collections. Our goal is to predict how much improvement is expected when the search algorithm is still running. Communicating this information to users will allow them to terminate a progressive search early and save time.

The challenge is how to derive such predictions. If we further inspect Figure 1, we see that answers progressively improve, but improvements are not radical. The error of the first approximate answer (when compared to the final exact answer) is on average 16%, which implies that approximate answers are generally not very far from the 1-NN. We show that this behavior is more general and can be observed across different datasets and different similarity search algorithms [85, 92]. We further show that the distance of approximate answers can help us predict the time that it takes to find the exact answer. Our approach consists in describing these behaviors through statistical models. We then use these models to estimate the error of a progressive answer, assess the probability of an early exact answer, and provide upper bounds for the time needed to find the k-NN. We also explore query-sensitive models that predict a probable range of the k-NN distance before the search algorithm starts, and then is progressively improved as new answers arrive. We further provide reliable stopping criteria for terminating searches with probabilistic guarantees about the distance error or the number of exact answers. We note that earlier approaches [20, 21] do not solve the problem, since they

support bounds only for *distance* errors, they do not update their estimates during the course of query answering, and they do not scale with the data size.

Contributions. Our key contributions are as follows:

• We formulate the problem of *progressive data series similarity search*, and provide definitions specific to the context of data series.

• We investigate statistical methods, based on regression (linear, quantile, and logistic) and multivariate kernel density estimation, for supporting progressive similarity search based on a small number of training queries (e.g., 50 or 100). We show how to apply them to derive estimates for the NN distance (distance error), the time to find the NN, and the probability that the progressive NN answer is correct.

• We further develop stopping criteria that can be used to stop the search long before the normal query execution ends. These criteria make use of distance error estimates, probabilities of exact answers, and time bounds for exact answers.

• We perform an extensive experimental evaluation with both synthetic and real datasets. The results demonstrate that our solutions dominate the previous approaches, provide accurate probabilistic bounds, and lead to significant time improvements with well-behaved guarantees for errors. Source code and datasets are in [1].

2 RELATED WORK

Similarity Search. Several measures have been proposed for computing similarity between data series [26, 60]. Among them, Euclidean Distance (ED) [33], which performs a point-by-point value comparison between two time series, is one of the most popular. ED can be combined with data normalization (often *z-normalization* [39]), in order to consider as similar patterns that may vary in amplitude or value offset. Ding et al. [26] conducted an analysis of different measures (including elastic ones such as Dynamic Time Warping [9] and threshold based ones such as TQuEST [4]) and concluded that there is no superior measure. In our work, we focus on ED because [26, 30] it is effective, it leads to efficient solutions for large datasets, and is very commonly used.

The human-computer interaction community has focused on the interactive visual exploration and querying of data series. These querying approaches are visual, often on top of line chart visualizations [78], and rely either on the interactive selection of part of an existing series (e.g., [13]), or on sketching patterns to search for (e.g., [23, 58]). This line of work is orthogonal to our approach, which considers approximate and progressive results from these queries when interactive search times are not possible.

Optimized and Approximate Similarity Search. The database community has optimized similarity search methods by using index structures [14, 22, 33, 50, 54, 55, 65, 67–69, 85, 88, 92], or fast sequential scans [72]. Recently, Echihabi et al. [30, 31] compared the efficiency of these methods under a unified experimental framework, showing that there is no single best method that outperforms all the rest. In this study, we focus on the popular centralized solutions, though, our results naturally extend to parallel and distributed solutions [67–69, 88], since these solutions are based on the same principles and mechanisms as their centralized counterparts. Moreover, we focus on (progressive answers for) exact query answering. Given enough time, all answers we produce are exact, which is important for several applications [66]. In this context, progressive answers help to speed-up exact queries by stopping execution early, when it is highly probable that the current progressive answer is the exact one.

In parallel to our work, Li et al. [52] proposed a machine learning method, developed on top of inverted-file (IVF [43] and IMI [5]) and k-NN graph (HNSW [57]) similarity search techniques, that solves the problem of early termination of approximate NN queries, while achieving a target recall. In contrast, our approach employs similarity search techniques based on data series indices [31], and with a very small training set (up to 200 training queries in our experiments), provides guarantees with per-query probabilistic bounds along different dimensions: on the distance error, on whether the current answer is the exact one, and on the time needed to find the exact answer.

Progressive Visual Analytics. Fekete and Primet [34] provide a summary of the features of a progressive system; three of them are particularly relevant to progressive data series search: (i) progressively improved answers; (ii) feedback about the computation state and costs; and (iii) guarantees of time and error bounds for progressive and final results. Systems that support big data visual exploration include Zenvisage [76] that provides incremental results of aggregation queries, Falcon [62] that prefetches data for brushing and linking actions, and IncVisage [71] that progressively reveals salient features in heatmap and trendline visualizations.

Systems that provide progressive results are appreciated by users due to their quick feedback [6, 89]. Nevertheless, there are some caveats. Users can be mislead into believing false patterns [61, 79] with early progressive results. It is thus important to communicate the progress of ongoing computations [2, 75], including the uncertainty and convergence of results [2] and guarantees on time and error bounds [34]. Previous work provides such uncertainty and guarantees in relational databases and aggregation type queries [41, 44, 87].

Closer to the context of data series, Ciaccia and Patella [21] studied similarity search queries over general multi-dimensional spaces and proposed a probabilistic approach for computing the uncertainty of partial similarity search results. We discuss their approach in the following section.

3 PRELIMINARIES AND BACKGROUND

A *data series* $S(p_1, p_2, ..., p_\ell)$ is an ordered sequence of points with length n. A data series of length ℓ can also be represented as a single point in an ℓ-dimensional space. For this reason, the values of a data series are often called *dimensions*, and its length ℓ is called *dimensionality*. We use $\mathbb{S}$ to denote a *data series collection* (or *dataset*). We refer to the size $n = |\mathbb{S}|$ of a data series collection as *cardinality*. In this paper, we focus on datasets with very large numbers of data series.

Distance Measures. A data series *distance* $d(S_1, S_2)$ is a function that measures the dissimilarity of two data series S_1 and S_2, or alternatively, the dissimilarity of two data series subsequences. As mentioned in Sec 2, we chose Euclidean Distance (ED) as a measure due to its popularity and efficiency [26].

Similarity Search Queries. Given a dataset $\mathbb{S}$, a *query* series Q, and a distance function $d(\cdot, \cdot)$, a *k-Nearest-Neighbor (k-NN) query* identifies the k series in the dataset with the smallest distances to Q. The 1st Nearest Neighbor (1-NN) is the series in the dataset with the smallest distance to Q.

Similarity search can be *exact*, when it produces answers that are always correct, or *approximate*, when there is no such strict guarantee. A *δ-ϵ-approximate algorithm* guarantees that its distance results will have a relative error no more than ϵ with a probability of at least δ [30]. We note that only a couple of approaches [3, 21] provide such guarantees. Yet, their accuracy has never been tested on the range of dimensions and dataset sizes that we examine here.

Similarity Search Methods. Most data series similarity search techniques [14, 22, 33, 50, 54, 68, 85, 88, 92] use an index, which enables scalability. The index can offer quick approximate answers by traversing a single path of the index structure to visit the single most promising leaf, from where we select the *best-so-far (bsf)* answer: this is the candidate answer in the leaf that best matches (has the smallest distance to) the query. The bsf may, or may not be the final, exact answer: in order to verify, we need to either prune, or visit all the other leaves of the index. Having a good first bsf (i.e., close to the exact answer) leads to efficient pruning.

In the general case, approximate data series similarity search algorithms do not provide guarantees about the quality of their answers. In our work, we illustrate how we can efficiently provide such guarantees, with probabilistic bounds.

We focus on index-based approaches that support both quick approximate, and slower but exact, similarity search results. In this work, we adapt the state-of-the-art data series indexes iSAX2+ [15] and DSTree [85], which have been shown to outperform the other data series methods in query answering [30], and we demonstrate that our techniques are applicable to both indexes. We provide below a succinct description of the iSAX2+ and DSTree approaches.

The iSAX2+ [15] index organizes the data in a tree structure, where the leaf nodes contain the raw data and each internal node summarizes the data series that belong to its subtree using a representation called *Symbolic Aggregate Approximation (SAX)* [53]. SAX transforms a data series using Piecewise Aggregate Approximation (PAA) [47] into equi-length segments, where each segment is associated with the mean value of its points, then represents the mean values using a discrete set of symbols for a smaller footprint.

DSTree [85] is also a tree-based index that stores raw data in the leaves and summaries in internal nodes. Contrary to iSAX2+, DSTree does not support bulkloading, intertwines data segmentation with indexing and uses *Extended Adaptive Piecewise Approximation (EAPCA)* [85] instead of SAX. With EAPCA, a data series is segmented using APCA [16] into varying-length segments, then each segment is represented with its mean and standard deviation values.

Since the query answering time depends on the data distribution [93], and both iSAX2+ and DSTree can produce unbalanced index trees, we provide below an index-invariant asymptotic analysis on the lower/upper bounds of the query runtime. As we consider large on-disk datasets, the query runtime is I/O bound; thus we express complexity in terms of I/O [42, 46], using the dataset size N, the index leaf threshold th and the disk block size B. Consider an index over a dataset of size N such that each index leaf contains at most th series ($th \ll N$). We count one disk page access of size B as one I/O operation (for simplicity, we use B to denote the number of series that fit in one disk page). Note that both the iSAX2+ and DSTree indexes fit the entire index tree in-memory; the leaves point to the raw data on-disk.

Best Case. The best case scenario occurs when one of the children of the index root is a leaf, containing one data series. In this case, the approximate search will incur $\Theta(1)$ I/O operation. In the best case, exact search will prune all other nodes of the index and thus will also incur $\Theta(1)$ disk access.

Worst Case. Approximate search always visits one leaf. Therefore, the worst case occurs when the leaf is the largest possible, i.e., it contains th series, in which case approximate search incurs $\Theta(th/B)$ I/O operations. For exact search, the worst case occurs when the algorithm needs to visit every single leaf of the index. This can happen when the index tree has $N - th + 1$ leaves (i.e., each leaf contains only one series, except for one leaf with th series), as a result of each new series insertion causing a leaf split where only one series ends up in one of the children. Therefore, the exact search algorithm will access all the leaves, and will incur $\Theta(N)$ I/O operations. (Note that this is a pathological case that would happen when all series are almost identical: in this case, indexing and similarity search are not useful anyways.)

Baseline Approach. We briefly describe here the probabilistic approach of Ciaccia et al. [20–22]. Based on Ciaccia

et al. [22], a dataset $\mathbb{S}$ (a data series collection in our case) can be seen as a random sample drawn from a large population $\mathcal{U}$ of points in a high-dimensional space. Being a random sample, a large dataset is expected to be representative of the original population. Given a query Q, let $f_Q(x)$ be the probability density function that gives the relative likelihood that Q's distance from a random series drawn from $\mathcal{U}$ is equal to x. Likewise, let $F_Q(\cdot)$ be its cumulative probability function. Based on $F_Q(\cdot)$, we can derive the cumulative probability function $G_{Q,n}(\cdot)$ for Q's k-NN distances in a dataset of size $n = |\mathbb{S}|$. For 1-NN similarity search, we have:

$$G_{Q,n}(x) = 1 - (1 - F_Q(x))^n \quad (1)$$

We now have a way to construct estimates for 1-NN distances. Unfortunately, $f_Q(\cdot)$, and thus $F_Q(\cdot)$, are not known. The challenge is how to approximate them from a given dataset. We discuss two approximation methods:

1. *Query-Agnostic Approximation.* For high-dimensional spaces, a large enough sample from the overall distribution $f(\cdot)$ of pairwise distances in a dataset provides a reasonable approximation for $f_Q(\cdot)$ [22]. This approximation can then be used to evaluate probabilistic stopping-conditions by taking sampling sizes between 10% and 1% (for larger datasets) [21].

2. *Query-Sensitive Approximation.* The previous method does not take the query into account. A query-sensitive approach is based on a training set of reference queries, called *witnesses*. Witnesses can be randomly drawn from the dataset, or selected with the GNAT algorithm [12], which identifies the n_w points that best cover a multidimensional (metric) space based on an initial random sample of $3n_w$ points. Given that close objects have similar distance distributions, Ciaccia et al. [20] approximate $f_Q(\cdot)$ by using a weighted average of the distance distributions of all the witnesses.

The above methods have major limitations. First, since their 1-NN distance estimates are static, they are less appropriate for progressive similarity search. Second, a good approximation of $F_Q(\cdot)$ does not necessarily lead to a good approximation of $G_{Q,n}(\cdot)$. This is especially true for large datasets, as the exponent term n in Equation 1 will inflate even tiny approximation errors. Note that $G_{Q,n}(\cdot)$ can be thought of as a scaled version of $F_Q(\cdot)$ that zooms in on the range of the lowest distance values. If this narrow range of distances is not accurately approximated, the approximation of $G_{Q,n}(\cdot)$ will also fail. Our own evaluation demonstrates this problem. Third, they require the calculation of a large number of distances. Since the approximation of $G_{Q,n}(\cdot)$ is sensitive to errors in large datasets (see above), a rather large number of samples is required in order to capture the frequency of the very small distances.

Table 1: Table of symbols

Symbol	Description
S, Q	data series, query series
ℓ	length of a data series
$\mathbb{S}$	data series collection (or dataset)
$n = \|\mathbb{S}\|$	number of series in $\mathbb{S}$
$R(t)$	progressive answer at time t
k-NN, $knn(Q)$	k^{th} Nearest Neighbor of Q
$d_{Q,R}(t)$, $d(Q,R(t))$	distance between Q and $R(t)$
$d_{Q,knn}$, $d(Q,knn(Q))$	distance between Q and its k-NN
$\epsilon_Q(t)$	relative distance error of $R(t)$ from k-NN
$p_Q(t)$	probability that $R(t)$ is exact (i.e., the k-NN)
t_Q	time to find the k-NN
$\tau_{Q,\phi}$	time to find the k-NN with probability $1-\phi$
$\tau_{Q,\theta,\epsilon}$	time for which $\epsilon_Q(t) < \epsilon$ with confidence $1-\theta$
$\hat{\bullet}$	estimate of $\bullet$
$I_Q(t)$	information at time t
$h_{Q,t}(x)$	probability density function of Q's distance from its k-NN, given information $I_Q(t)$
$H_{Q,t}(x)$	cumulative distribution function of Q's distance from its k-NN, given $I_Q(t)$
$f_Q(x)$	probability density function of Q's distance from a random series in $\mathbb{S}$
$F_Q(x)$	cumulative distribution function of Q's distance from a random series in $\mathbb{S}$
$G_{Q,n}(x)$	cumulative distribution function of Q's distance from its k-NN
$\mathcal{W}$	set of witness series
$n_w = \|\mathcal{W}\|$	number of witnesses in $\|\mathcal{W}\|$

4 PROGRESSIVE SIMILARITY SEARCH

We define progressive similarity search for k-NN queries[2]. (Table 1 summarizes the symbols we use in this paper.)

Definition 4.1. Given a k-NN query Q, a data series collection $\mathbb{S}$, and a time *quantum* q, a ***progressive similarity-search algorithm*** produces results $R(t_1), R(t_2), ..., R(t_z)$ at time points $t_1, t_2, ..., t_z$, where $t_{i+1} - t_i \leq q$, such that $d(Q, R(t_{i+1})) \leq d(Q, R(t_i))$.

We borrow the quantum q parameter from Fekete and Primet [34]. It is a user-defined parameter that determines how frequently users require updates about the progress of their search. Although there is no guarantee that distance results will improve at every step of the progressive algorithm, the above definition states that a progressive distance will *never* deteriorate. This is an important difference of progressive similarity search compared to other progressive computation mechanisms, where results may fluctuate before they eventually converge, which may lead users to making wrong decisions based on intermediate results [18, 34, 40].

Clearly, progressive similarity search can be based on approximate similarity search algorithms – a progressive result is simply an approximate (best-so-far) answer that is updated

[2]We define the problem using k-NN, but for simplicity use $k = 1$ in the rest of this paper. We defer the discussion of the general case to future work.

over time. A progressive similarity search algorithm is also exact if the following condition holds:

$$\lim_{t \to \infty} d(Q, R(t)) = d(Q, knn(Q)) \quad (2)$$

where $knn(Q)$ represents the k-NN of the query series Q.

According to the above condition, the progressive algorithm will always find an exact answer. However, there are generally no strong guarantees about how long this can take. Ideally, a progressive similarity search algorithm will find good answers very fast, e.g., within interactive times, and will also converge to the exact answer without long delays. Even so, in the absence of information, users may not be able to trust a progressive result, no matter how close it is to the exact answer. In this paper, we investigate exactly this problem. Specifically, we seek to provide guarantees about: (i) How close is the progressive answer to the exact answer? (ii) What is the probability that the current progressive answer is the exact answer? (iii) When is the search algorithm expected to find the exact answer?

4.1 Progressive Distance Estimates

Given a progressive answer $R(t)$ to a k-NN query at time t, we are interested in knowing how far from the k-NN this answer is. For simplicity, we will denote the k-NN distance to the query as $d_{Q,knn} \equiv d(Q, knn(Q))$ and the distance between $R(t)$ and the query as $d_{Q,R}(t) \equiv d(Q, R(t))$. Then, the relative distance error is $\epsilon_Q(t)$, where $d_{Q,R}(t) = d_{Q,knn}(1 + \epsilon_Q(t))$. Given that this error is not known, our goal is to find an estimate $\hat{\epsilon}_Q(t)$. However, finding an estimate for the relative error is not any simpler than finding an estimate $\hat{d}_{Q,knn}(t)$ of the actual k-NN distance. We concentrate on this latter quantity for our analysis below. Though, since $d_{Q,R}(t)$ is known, deriving the distance error estimate $\hat{\epsilon}_Q(t)$ from the k-NN distance estimate $\hat{d}_{Q,knn}(t)$ is straightforward:

$$\hat{\epsilon}_Q(t) = \frac{d_{Q,R}(t)}{\hat{d}_{Q,knn}(t)} - 1 \quad (3)$$

We represent progressive similarity-search estimates as probability distribution functions.

Definition 4.2. Given a k-NN query Q, a data series collection $\mathbb{S}$, and a progressive similarity-search algorithm, a ***progressive k-NN distance estimate*** $\hat{d}_{Q,knn}(t)$ of the k-NN distance at time t is a probability density function:

$$h_{Q,t}(x) = Pr\{d_{Q,knn} = x \mid I_Q(t)\} \quad (4)$$

This equation gives the conditional probability that $d_{Q,knn}$ is equal to x, given information $I_Q(t)$.

We expect that progressive estimates will converge to $d_{Q,knn}$ (i.e., $\hat{\epsilon}_Q(t)$ will converge to zero). Evidently, the quality of an estimate at time t largely depends on the information $I_Q(t)$ that is available at this moment. In Section 5, we investigate different types of information we can use for this.

Given the probability density function in Equation 4, we can derive a point estimate that gives the *expected* k-NN distance, or an interval estimate in the form of a *prediction interval* (PI). Like a confidence interval, a prediction interval is associated with a confidence level. Given a confidence level $1 - \theta$, we expect that approximately $(1 - \theta) \times 100\%$ of the prediction intervals we construct will include the true k-NN distance. Note that although a confidence level can be informally assumed as a probability (i.e., what is the likelihood that the interval contains the true k-NN distance?), this assumption may or may not be strictly correct. Our experiments evaluate the frequentist behavior of such intervals.

To construct a prediction interval with confidence level $1 - \theta$ over a density distribution function $h_{Q,t}(\cdot)$, we derive the cumulative distribution function:

$$H_{Q,t}(x) = Pr\{d_{Q,knn} \leq x \mid I_Q(t)\} \quad (5)$$

From this, we take the $\theta/2$ and $(1 - \theta/2)$ quantiles that define the limits of the interval.

4.2 Guarantees for Exact Answers

Users may also need guarantees about the exact k-NN. We investigate two types of probabilistic guarantees for exact answers. First, at any moment t of the progressive search, we assess the probability $p_Q(t)$ that the exact answer has been found, given information $I_Q(t)$:

$$p_Q(t) = Pr\{d_{Q,R}(t) = d_{Q,knn} \mid I_Q(t)\} \quad (6)$$

Second, we estimate the time t_Q it takes to find the exact k-NN. As we already discussed, this time can be significantly faster than the time needed to complete the search. Let $\hat{t}_Q$ be its estimate. We express it as a probability density function:

$$r_{Q,t}(x) = Pr\{t_Q = x \mid I_Q(t)\} \quad (7)$$

which expresses the conditional probability that t_Q is equal to x, given information $I_Q(t)$. From this, we derive its cumulative distribution function $R_Q(\cdot)$. Then, given a confidence level $1 - \phi$, we can find a probabilistic upper bound $\tau_{Q,\phi}$ such that $R_Q(\tau_{Q,\phi}) = 1 - \phi$; ϕ represents the probability that the progressive answer at time $\tau_{Q,\phi}$ is not the exact, i.e., the proportion of bounds that fail to include the exact answer.

4.3 Stopping Criteria

Based on the provided estimates, users may decide to trust the current progressive result and possibly stop their search. Which *stopping criterion* to use is not straightforward and depends on whether users prioritize guarantees about the k-NN

Table 2: Experimental datasets

name	description	number of series	series length
synthetic	random walks	100M	256
seismic [36]	seismic records	100M	256
SALD [80]	MRI data	200M	128
deep1B [81]	image descriptors	267M	96

distance, about the relative error of the current progressive result, or about the exact answer itself.

An analyst may choose to stop query execution as soon as the prediction interval of the k-NN distance lies above a low threshold value. Unfortunately, this strategy raises some concerns. Recent work on progressive visualization [59] discusses the problem of confirmation bias, where an analyst may use incomplete results to confirm a "preferred hypothesis". This is a well-studied problem in sequential analysis [82]. It relates to the multiple-comparisons problem [90] and is known to increase the probability of a Type I error (false positives). We evaluate how such multiple sequential tests affect the accuracy of our methods, but discourage their use as stopping criteria, and instead propose the following three.

A sensible approach is to make use of the relative distance error estimate $\hat{\epsilon}_Q(t)$ (see Eq. 3). For instance, the analyst may decide to stop the search when the upper bound of the error's interval is below a threshold $\epsilon = 1\%$. An error-based stopping criterion offers several benefits: (i) the choice of a threshold does not depend on the dataset, so its interpretation is easier; (ii) this criterion does not inflate Type I errors as long as the threshold ϵ is fixed in advance; (iii) the error $\epsilon_Q(t)$ monotonically converges to zero (the same holds for the bounds of its estimates), thus there is a unique point in time $\tau_{Q,\theta,\epsilon}$ at which the bound of the estimated error reaches ϵ, where $1-\theta$ is our confidence level (here, θ determines the proportion of times for which the relative distance error of our result will be greater than ϵ).

A second approach is to use the $\tau_{Q,\phi}$ bound (see Section 4.2) to stop the search, which provides guarantees about the proportion of exact answers, rather than the distance error. It also depends on a single parameter, rather than two. To avoid the multiple-comparisons problem, we provide a single estimate of this bound at the very beginning of the search, allowing users to plan ahead their stopping strategy.

A third approach is to bound the probability $p_Q(t)$. Specifically, we stop the search when this probability exceeds a level $1-\phi$, where ϕ here represents the probability that the current progressive answer is not the exact. We experimentally assess the tradeoffs of these three stopping criteria.

5 PREDICTION METHODS

We now present our solutions. We use the 4 datasets shown in Table 2 to showcase our methods. We further explain and use these datasets in Section 6 to evaluate our methods.

Our goal is to support reliable prediction with small training sets of queries. We are also interested in expressing the uncertainty of our predictions with well-controlled bounds, as discussed in the previous section. We thus focus on statistical models that capture a small number of generic relationships in the data. We first examine methods that assume constant information ($I_Q(t) = I_Q$). They are useful for providing an initial estimate *before* the search starts. We distinguish between query-sensitive methods, which take into account the query series Q, and query-agnostic methods, which provide a common estimate irrespective of Q ($I_Q = I$). Inspired by Ciacca et al. [20, 22], these methods serve as baselines to compare to a new set of progressive methods. Our progressive methods update information during the execution of a search, resulting in considerably improved predictions.

To simplify our analysis, we focus on 1-NN similarity search. Our analysis naturally extends to k-NN search; a detailed study of this case is part of our future work.

5.1 Initial 1-NN Distance Estimates

We first concentrate on how to approximate the distribution function $h_{Q,0}(x)$ (see Equation 4), thus provide estimates before similarity search starts.

As Ciaccia et al. [20], we rely on witnesses, which are "training" query series that are randomly sampled from a dataset. Unlike their approach, however, we do not use the distribution of raw pairwise distances $F_Q(\cdot)$. Instead, for each witness, we execute 1-NN similarity queries with a fast state-of-the-art algorithm, such as iSAX2+ [15], or DSTree [85]. This allows us to derive directly the distribution of 1-NN distances and predict the 1-NN distance of new queries.

This approach has two main benefits. First, we use the tree structure of the above algorithms to prune the search space and reduce pre-calculation costs. Rather than calculating a large number of pairwise distances, we focus on the distribution of 1-NN distances with fewer distance calculations. Second, we achieve reliable and high-quality approximation with a relatively small number of training queries ($\approx 100-200$) independently of the dataset size (we report and discuss these results in Section 6).

Query-Agnostic Model (Baseline). Let $\mathcal{W} = \{W_j | j = 1..n_w\}$ be a set of $n_w = |\mathcal{W}|$ witnesses randomly drawn from the dataset. We execute a 1-NN similarity search for each witness and build their 1-NN distance distribution. We then use this distribution to approximate the overall (query-independent) distribution of 1-NN distances $g_n(\cdot)$ and its cumulative probability function $G_n(\cdot)$. This method is comparable to Ciaccia et al. [22] query-agnostic approximation method and serves as a baseline.

Query-Sensitive Model. Intuitively, the smaller the distance between the query and a witness, the better the 1-NN

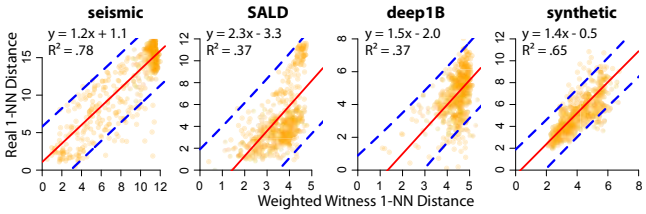

Figure 2: Linear models (red lines) predicting the real 1-NN distance $d_{Q,1nn}$ based on the weighted witness 1-NN distance dw_Q for $exp = 5$. All models are based on $n_w = 200$ random witnesses and $n_r = 100$ queries, and tested on 500 queries (in orange). The blue dashed lines show the range of their 95% prediction intervals.

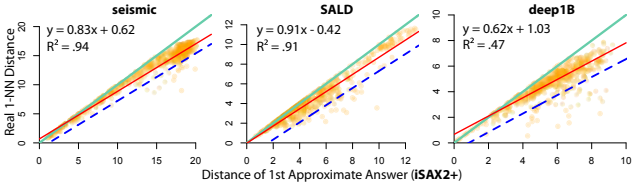

Figure 3: Linear models (red solid lines) predicting the real 1-NN distance $d_{Q,1nn}$ based on iSAX2+ *first* approximate answer distance. All models trained with 200 queries. The 500 (orange) points in each plot are queries out of the training set. Green (solid) lines ($y = x$) are hard upper bounds, set by the approximate answer. Blue lines show the range of one-sided 95% prediction intervals that form probabilistic lower bounds.

of this witness predicts the 1-NN of the query. We capture this relationship through a random variable that expresses the weighted sum of the 1-NN distance of all n_w witnesses:

$$dw_Q = \sum_{j=1}^{n_w} (a_{Q,j} \cdot d_{W_i,1nn}) \tag{8}$$

The weights $a_{Q,j}$ are derived as follows:

$$a_{Q,j} = \frac{d(Q,W_j)^{-exp}}{\sum_{i=1}^{n_w} d(Q,W_i)^{-exp}} \tag{9}$$

Our tests have shown optimal results for exponents exp that are close to 5. For simplicity, we use $exp = 5$ for all our analyses. Additional tests have shown that the fit of the model becomes consistently worse if witnesses are selected with the GNAT algorithm [12, 20] (we omit these results for brevity). Therefore, we only examine random witnesses here.

We use dw_Q as predictor of the query's real 1-NN distance $d_{Q,1nn}$ and base our analysis on the following linear model:

$$d_{Q,1nn} = \beta \cdot dw_Q + c \tag{10}$$

Figure 2 shows the parameters of instances of this model for the four datasets of Table 2. We conduct linear regressions by assuming that the distribution of residuals is normal (Gaussian) and has equal variance.

Since the model parameters (β and c) and the variance are dataset specific, they have to be trained for each individual dataset. To train the model, we use an additional random sample of n_r *training queries* that is different from the sample of witnesses. Based on the distance of each training query Q_i from the witnesses, we calculate dw_{Q_i} (see Equation 8). We also run similarity search to find its 1-NN distance $d_{Q_i,1nn}$. We then use all pairs $(dw_{Q_i}, d_{Q_i,1nn})$, where $i = 1..n_r$, to build the model. The approach allows us to construct both point estimates (see Equation 8) and prediction intervals (see Figure 2) that provide probabilistic guarantees about the range of the 1-NN distance.

5.2 Progressive 1-NN Distance Estimates

So far, we have focused on initial 1-NN distance estimates. Those do not consider any information about the partial results of a progressive similarity-search algorithm. Now, given Definition 4.1, the distance of a progressive result $d_{Q,R}(t_i)$ will never deteriorate and thus can act as an upper bound for the real 1-NN distance. The challenge is how to provide a probabilistic lower bound that is larger than zero.

Our approach relies on the observation that the approximate answers of index-based algorithms are generally close to the exact answers. Figure 3 illustrates the relationship between the true 1-NN distance and the distance of the first progressive (approximate) answer returned by iSAX2+ [15]. (The results for the DSTree index [85] that follows a completely different design from iSAX2+ are very similar; we omit them for brevity). We observe a strong linear relationship for both algorithms, especially for the DSTree index. We can express it with a linear model and then derive probabilistic bounds in the form of prediction intervals. As shown in Figure 3, the approach is particularly useful for constructing lower bounds. Those are clearly greater than zero and provide valuable information about the extent to which a progressive answer can be improved or not.

Since progressive answers improve over time and tend to converge to the 1-NN distance, we could take such information into account to provide tighter estimates as similarity search progresses. To this end, we examine different progressive prediction methods. They are all based on the use of a dataset of n_r training queries that includes information about all progressive answers of a similarity search algorithm to each query, including a timestamp and its distance.

Linear Regression. Let $t_1, t_2, ..., t_m$ be specific moments of interest (e.g., 100ms, 1s, 3s, and 5s). Given t_i, we can build a time-specific linear model:

$$d_{Q,1nn} = \beta_{t_i} \cdot d_{Q,R}(t_i) + c_{t_i} \tag{11}$$

where $d_{Q,R}(t_i)$ is Q's distance from the progressive answer at time t_i. As an advantage, this method produces models that are well adapted to each time of interest. On the downside, it requires the pre-specification of a discrete set of time points, which may not be desirable for certain application scenarios. However, building such models from an existing training dataset is inexpensive, so reconfiguring the moments of interest at use time is not a problem.

The above model can be enhanced with an additional term $\beta \cdot dw_Q$ (see Equation 8) that takes witness information into account. However, this term results in no measurable improvements in practice, so we do not discuss it further.

Kernel Density Estimation. A main strength of the previous method is its simplicity. However, linearity is a strong assumption that may not always hold. Other assumption violations, such as heteroscedasticity, can limit the accuracy of linear regression models. As alternatives, we investigate non-parametric methods that approximate the density distribution function $h_{Q,t}(\cdot)$ based on multivariate kernel density estimation [27, 83].

As for linear models, we rely on the functional relationship between progressive and final answers. We represent this relationship as a 3-dimensional density probability function $k_Q(x, y, t)$ that expresses the probability that the 1-NN distance from Q is x, given that Q's distance from the progressive answer at time t is y. From this function, we derive $h_{Q,t}(x)$ by setting $y = d_{Q,R}(t)$.

We examine two approaches for constructing the function $k_Q(\cdot, \cdot, \cdot)$. As for linear models, we specify discrete moments of interest t_i and then use bivariate kernel density estimation [84] to construct an individual density probability function $k_Q(\cdot, \cdot, t_i)$. Alternatively, we construct a common density probability function by using 3-variate kernel density estimation. The advantage of this method is that it can predict the 1-NN distance at any point in time. Nevertheless, this comes with a cost in terms of precision (see Section 6).

The accuracy of kernel density estimation highly depends on the method that one uses to smooth out the contribution of points (2D or 3D) in a training sample. We use gaussian kernels, but for each estimation approach, we select bandwidths with a different technique. We found that the plug-in selector of Wand and Jones [84] works best for our bivariate approach, while the smoothed cross-validation technique [27] works best for our 3-variate approach.

Measuring Time. So far, we have based our analysis on time. Nevertheless, time (expressed in seconds) is not a reliable measure for training and applying models in practice. The reason is that time largely depends on the available computation power, which can vary greatly across different hardware settings. Our solution is to use alternative measures that capture the progress of computation without being affected by hardware and computation loads. One can use either the number of series comparisons (i.e., the number of distance calculations), or the number of visited leaves. Both measures can be easily extracted from the iSAX2+ [15], the DSTree [85], or other tree-based similarity-search algorithms. Our analyses in this paper are based on the number of visited leaves (*Leaves Visited*). We should note that for a given number of visited leaves, we only consider a single approximate answer, which is the best-so-far answer after traversing the last leaf.

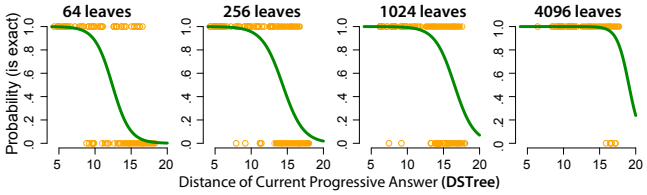

Figure 4: Estimating the probability of exact answers with 100 training queries based on the current progressive answer (seismic dataset). We show the logistic curves (in green) at different points in time (64, 256, 1024, and 4096 leaves) and 200 test queries (in orange).

5.3 Estimates for Exact Answers

We investigate two types of estimates for exact answers (see Section 4.2): (i) progressive estimates of the probability $p_Q(t)$ that the 1-NN has been found; and (ii) query-sensitive estimates of the time t_Q that it takes to find the exact answer. We base our estimations on the observation that queries with larger 1-NN distances tend to be harder, i.e., it takes longer to find their 1-NN. Now, since approximate distances are good predictors of their exact answers (see previous subsection), we can also use them as predictors of $p_Q(t)$ and t_Q.

Probability Estimation. Let $t_1, t_2, ..., t_m$ be moments of interest, and let $d_{Q,R}(t_i)$ be the distance of the progressive result at time t_i. We use logistic regression to model the probability $p_Q(t_i)$ as follows:

$$ln\frac{p_Q(t_i)}{1 - p_Q(t_i)} = \beta_{t_i} \cdot d_{Q,R}(t_i) + c_{t_i} \quad (12)$$

Again, we can use the number of visited leaves to represent time. Figure 4 presents an example for the seismic dataset, where we model the probability of exact answers for four points in time (when 64, 256, 1024, and 4096 leaves are visited). We observe that over time, the curve moves to the right range of distances, and thus, probabilities increase.

Note that we have considered other predictors as well (such as the time passed since the last progressive answer), but they did not offer any predictive value.

Time Bound Estimation. As we explained in Section 4.3, we provide a single estimate for t_Q at the very beginning of the search. Figure 5 illustrates the relationship between

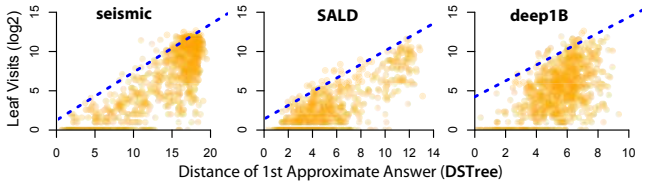

Figure 5: Upper time bounds for exact answers (ϕ = .05). Bounds (in blue) are constructed from 100 queries through quantile regression, where we estimate the 95% quantile of the logarithm of leaf visits as a function of the distance of the 1st approximate answer.

the distance of the first approximate answer and the number of leaves (in logarithmic scale) at which the exact answer is found. We observe that the correlation between the two variables is rather weak. However, we can still extract meaningful query-sensitive upper bounds, shown in blue (dashed lines). To construct such bounds, we use quantile regression [49]. This method allows us to directly estimate the $1-\phi$ quantile of the time needed to find the exact answer, i.e., derive the upper bound $\tau_{Q,\phi}$. As a shortcoming, the accuracy of quantile regression is sensitive in small samples. Nevertheless, we show that 100 training queries are generally enough to ensure high-quality results.

5.4 Visualization Example

Figure 6 presents an example that illustrates how the above methods can help users assess how far from the 1-NN their current answers are. We use a variation of pirate plots [70] to visualize the 1-NN distance estimate $\hat{d}_{Q,1nn}(t)$ and the relative error estimate $\hat{\epsilon}_Q(t)$ by showing their probability density distribution and their 95% prediction interval. We also communicate the probability $p_Q(t_i)$ and a probabilistic bound $\tau_{Q,\phi}$ (ϕ = .05) after the first visited leaf.

Observe that the initial distance estimate is rather uncertain, but estimates become precise at the early stages of the search. The upper bound of the error estimate drops below 10% within 1.1sec, while the probability that the current answer is exact is estimated as 98% after 15.7sec (total query execution time for this query is 75.2sec). Such estimates can give confidence to the user that the current answer is very close to the 1-NN. In this example, the 1-NN is found in 3.8sec.

6 EXPERIMENTAL EVALUATION

Environment. All experiments were run on a Dell T630 rack server with two Intel Xeon E5-2643 v4 3.4Ghz CPUs, 512GB of RAM, and 3.6TB (2 x 1.8TB) HDD in RAID0.

Implementation. Our estimation methods were implemented in R. We use R's *lm* function to carry our linear regression, the *ks* library [28] for multivariate kernel density estimation, and the *quantreg* library [32] for quantile regression. We use a grid of 200×200 points to approximate a 2D density distribution and a grid of $60 \times 180 \times 180$ points to approximate a 3D density distribution. Source code and datasets are in [1].

Datasets. We used 1 synthetic and 3 real datasets from past studies [30, 92]. All are *100GB* in size with cardinalities and lengths reported in Table 2. Synthetic data series were generated as random walks (cumulative sums) of steps that follow a Gaussian distribution (0,1). This type of data has been extensively used in the past [15, 33, 93] and models the distribution of stock market prices [33]. The IRIS seismic dataset [36] is a collection of seismic instrument recordings from several stations worldwide (100M series of length 256). The SALD neuroscience dataset [80] contains MRI data (200M series of length 128). The image processing dataset, deep1B [81], contains vectors extracted from the last layers of a convolutional neural network (267M series of length 96).

Measures. We use the following measures to assess the estimation quality of each method and compare their results:

Coverage Probability: It measures the proportion of the time that the prediction intervals contain the true 1-NN distance. If the confidence level of the intervals is $1-\theta$, the coverage probability should be close to $1-\theta$. A low coverage probability is problematic. In contrast, a coverage probability that is higher than its nominal value (i.e., its confidence level) is acceptable but can hurt the intervals' precision. In particular, a very wide interval that always includes the true 1-NN distance (100% coverage) can be useless.

Prediction Intervals Width: It measures the size of prediction intervals that a method constructs. Tighter intervals are better. However, this is only true if the coverage probability of the tighter intervals is close to or higher than their nominal confidence level. Note that for progressive distance estimates, we construct one-sided intervals. Their width is defined with respect to the upper distance bound $d_{Q,R}(t)$.

Root-Mean-Squared Error (RMSE): It evaluates the quality of point (rather than interval) estimates by measuring the standard deviation of the true 1-NN distance values from their expected (mean) values.

To evaluate the performance of our stopping criteria, we further report on the following measures:

Exact Answers: It measures the number of exact answers as a percentage of the total number of queries.

Time Savings: Given a load of queries and a stopping criterion, it measures the time saved as a percentage of the total time needed to complete the search without early stopping.

Validation Methodology. To evaluate the different methods, we use a Monte Carlo cross-validation approach that consists of the following steps. For each dataset, we randomly draw two disjoint sets of data series $\mathcal{W}_{pool}$ and $\mathcal{T}_{pool}$ and precalculate all distances between the series of these two sets. The first set serves as a pool for drawing random sets of witnesses (if applicable), while the second set serves as a pool for

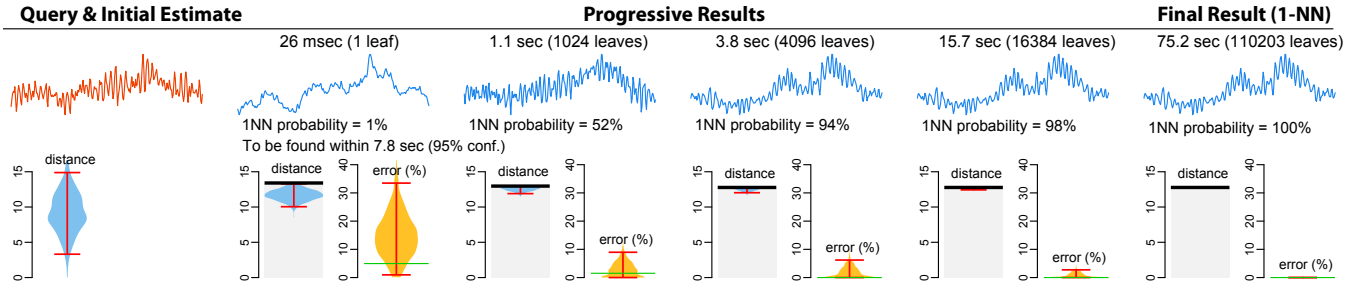

Figure 6: A query example from the seismic dataset showing the evolution of estimates based on our methods. The thick black lines show the distance of the current approximate answer. The red error bars represent 95% prediction intervals. The green line over the predicted distribution of distance errors shows the real error – it is unknown during the search and is shown here for illustration purposes. Estimates are based on a training set of 100 queries, as well as 100 random witnesses for initial estimates. We use the iSAX2+ index.

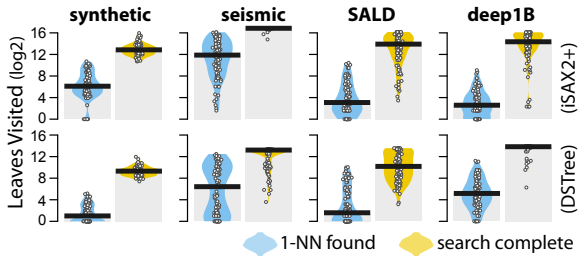

Figure 7: Distribution (over 100 queries) of the number of leaves visited (in log_2 scale) until finding the 1-NN (light blue) and completing the search (yellow). The thick black lines represent medians.

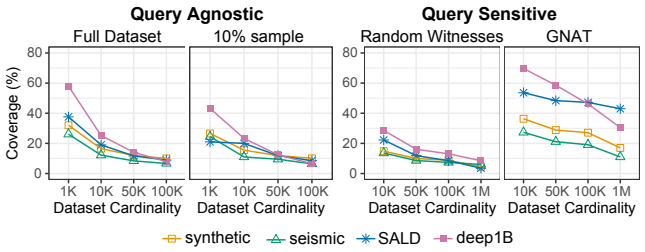

Figure 8: Coverage probabilities of query-agnostic (left) and query-sensitive (right) methods of Ciaccia et al. [20, 21] for 95% confidence level. We use 500 witnesses for the query-sensitive methods. We show best-case results (with the best *exp*: 3, 5, 12, or adaptive).

randomly drawing training (if applicable) and testing queries. At each iteration, we draw n_w witnesses ($n_w = 50$, 100, 200, or 500) and/or n_r training queries ($n_r = 50$, 100, or 200) from $\mathcal{W}_{pool}$ and $\mathcal{T}_{pool}$, respectively. We also draw $n_t = 200$ testing queries from $\mathcal{T}_{pool}$ such that they do not overlap with the training queries. We train and test the evaluated methods and then repeat the same procedure $N = 100$ times, where each time, we draw a new set of witnesses, training, and testing queries. Thus, for each method and condition, our results are based on a total of $N \times n_t = 20K$ measurements.

For all progressive methods, we test the accuracy of their estimates after the similarity search algorithm has visited 1 (2^0), 4 (2^2), 16 (2^4), 64 (2^6), 256 (2^8), and 1024 (2^{10}) leaves. Figure 7 shows the distributions of visited leaves for 100 random queries for all four datasets.

6.1 Results on Prediction Quality

Previous Approaches. We first evaluate the query-agnostic and query-sensitive approximation methods of Ciaccia et al. [20, 21]. To assess how the two methods scale with and without sampling, we examine smaller datasets with cardinalities of up to 1M data series (up to 100K for the query-agnostic approach). Those datasets are derived from the initial datasets presented in Table 2 through random sampling. Such smaller dataset sizes allow us to derive the full distribution of distances without sampling errors, while they are sufficient for demonstrating the behavior of the approximation methods as datasets grow.

Figure 8 presents the coverage probabilities of the methods. The behavior of query-agnostic approximation is especially poor. Even when the full dataset is used to derive the distribution of distances, the coverage tends to drop below 10% for larger datasets (95% confidence level). This demonstrates that the approximated distribution of 1-NN distances completely fails to capture the real one.

Results for the query-sensitive method are better, but coverage is still below acceptable levels. Figure 8 presents results for $n_w = 500$ witnesses. Note that our further tests have shown that larger numbers of witnesses result in no or very little improvement, while Ciaccia et al. [20] had tested a maximum of 200 witnesses. To weight distances (see Equation 9), we tested the exponent values $exp = 3$, 5, and 12, where the first two were also tested by Ciaccia et al. [20], while we found that the third one gave better results for some datasets. We also tested the authors' adaptive technique. Figure 8 presents the best result for each dataset, most often given by the adaptive technique.

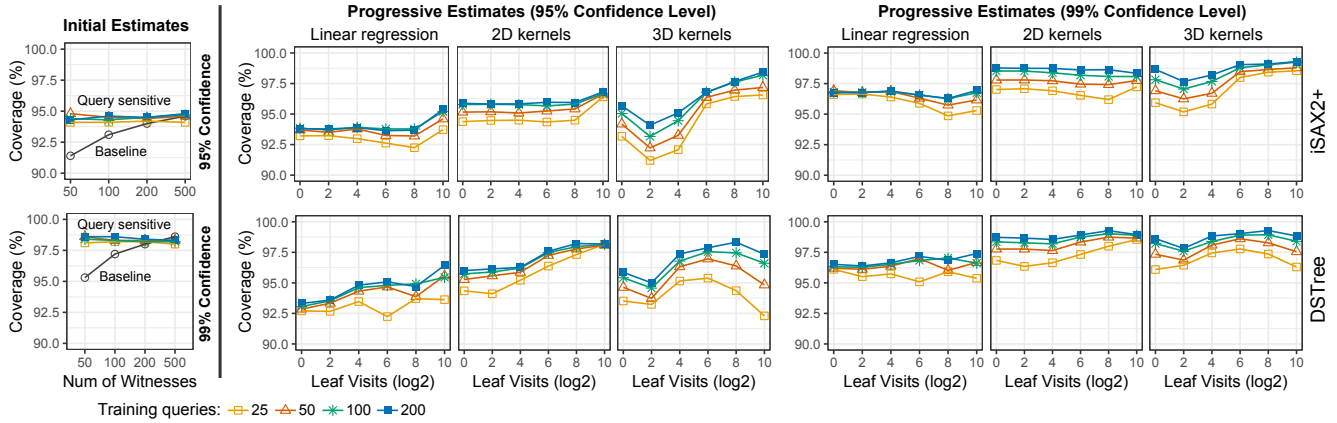

Figure 9: Coverage probabilities of our estimation methods for 95% and 99% confidence levels. We show averages for the four datasets (synthetic, seismic, SALD, deep1B) and for 25, 50, 100, and 200 training queries.

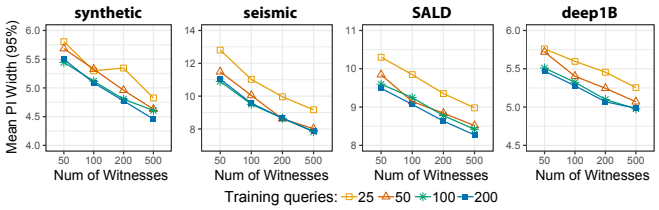

Figure 10: The mean width of the 95% PI for the witness-based query-sensitive method in relation to the number of witnesses and training queries.

We observe that the GNAT method results in clearly higher coverage probabilities than the fully random method. This result is somehow surprising because Ciacca et al. [20] report that the GNAT method tends to become less accurate than the random method in high-dimensional spaces with more than eight dimensions. Even so, the coverage probability of the GNAT method is largely below its nominal level. In all cases, it tends to become less than 50% as the cardinality of the datasets increases beyond 100K, while in some cases, it drops below 20% (synthetic and seismic).

For much larger datasets (e.g., 100M data series), we expect the accuracy of the above methods to become even worse. We conclude that they are not appropriate for our purposes, thus we do not study them further.

Quality of Distance Estimates. We evaluate the coverage probability of 1-NN distance estimation methods for confidence levels 95% ($\theta = .05$) and 99% ($\theta = .01$). Figure 9 presents our results. The coverage of the *Baseline* method reaches its nominal confidence level for $n_w = 200$ to 500 witnesses. In contrast, the *Query-Sensitive* method demonstrates a very good coverage even for small numbers of witnesses ($n_w = 50$) and training queries ($n_r = 25$). However, as Figure 10 shows, more witnesses increase the precision of prediction intervals, i.e., intervals become tighter while they still cover the same proportion of true 1-NN distances. Larger numbers of training queries also help.

The coverage probabilities of progressive estimates (Figure 9-Right) are best for the 2D kernel density approach, very close to their nominal levels. Linear regression leads to lower coverage, while the coverage of the 3D kernel density approach is more unstable. We observe that although the accuracy of the models drops in smaller training sets, coverage levels can still be considered as acceptable even if the number of training queries is as low as $n_r = 25$.

Figure 11 compares the quality of initial and early (i.e., based on first approximate answer) estimates provided by different techniques: (i) Baseline, (ii) Query-Sensitive method, (iii) 2D kernel density estimate for iSAX2+, and (iii) 2D kernel density estimate for DSTree. For all comparisons, we set $n_w = 500$ and $n_r = 100$. For these parameters, the coverage probability of all methods is close to 95%. We evaluate the width of their 95% prediction intervals and RMSE. We observe similar trends for both measures, where the query-sensitive method outperforms the baseline. We also observe that estimation based on the first approximate answer (at the first leaf) leads to radical improvements for all datasets. Overall, the DSTree index gives better estimates than iSAX2+.

As shown in Figure 12, progressive answers lead to further improvements. The RMSE is very similar for all three estimation methods, which means that their point estimates are equally good. Linear regression results in the narrowest intervals, which explains the lower coverage probability of this method. Overall, 2D kernel density estimation provides the best balance between coverage and interval width.

Sequential Tests. We assess how multiple sequential tests (refer to Section 4) affect the coverage probability of 1-NN distance prediction intervals. We focus on 2D kernel density estimation ($n_r = 100$), which gives the best coverage (see Figure 9). We examine the effect of (i) three sequential tests

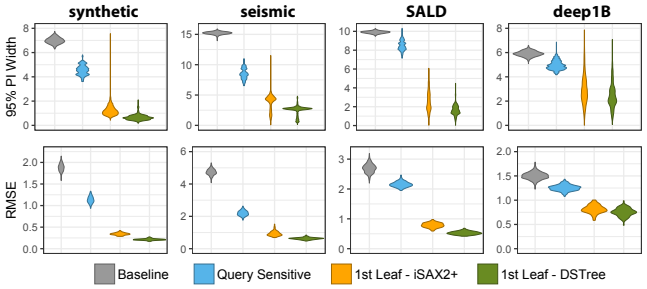

Figure 11: Violin plots showing the distribution of the width of 95% prediction intervals (top) and the distribution of the RMSE of expected 1-NN distances (bottom). We use $n_w = 500$ (baseline and query-sensitive method) and $n_r = 100$ (query-sensitive method and 2D kernel model for the 1st approximate answer).

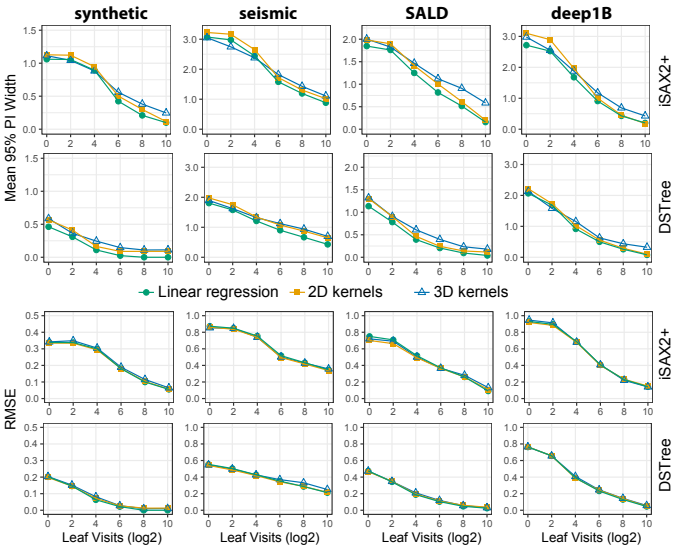

Figure 12: Progressive models: Mean width of 95% prediction intervals of 1-NN distance estimates and RMSE. Results are based on $n_r = 100$ training queries.

when visiting 1, 512, and 1024 leaves, and (ii) five sequential tests when visiting 1, 256, 512, 768, and 1014 leaves. We count an error if at least one of the three, or five progressive prediction intervals do not include the true 1-NN distance.

As results for DSTree and iSAX2+ were very close, we report on their means (see Figure 13(a)). The coverage of 95% prediction intervals drops from over 95% to about 90% for five tests (higher for seismic and lower for deep1B). Likewise, the coverage of their 99% prediction intervals drops to around 95%. These results provide rules of thumb on how to correct for multiple sequential tests, e.g., use a 95% level in order to guarantee a 90% coverage in 5 sequential tests. Notice, however, that such rules may depend on the estimation

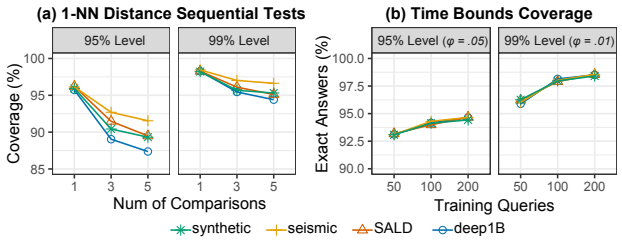

Figure 13: (a) Effect of 3 and 5 sequential tests on the coverage of 95% and 99% prediction intervals. We use 2D kernels with $n_r = 100$. (b) Coverage of exact answers for time upper bounds (95% and 99% conf. levels).

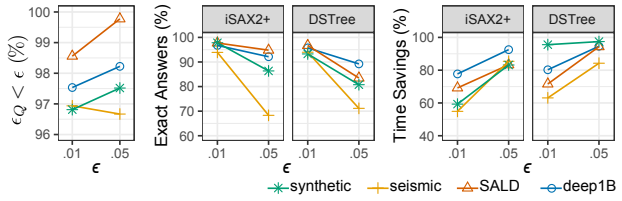

Figure 14: Evaluation of the stopping criterion that bounds the distance error ($\epsilon_Q < \epsilon$). We use 95% prediction intervals ($\theta = .05$) and $n_r = 100$ training queries.

method and the time steps at which comparisons are made. An in-depth study of this topic is part of our future work.

Time Bounds for Exact Answers. We are also interested in the quality of time guarantees for exact answers (refer to Section 4.2). We evaluate the coverage of our time bounds for 50, 100, and 200 training queries for confidence levels 95% ($\phi = .05$) and 99% ($\phi = .01$). Figure 13(b) summarizes our results. We observe that coverage is good for training samples of $n_r \geq 100$, but drops for $n_r = 50$.

6.2 Results on Time Savings

We compare our stopping criteria (see Section 4.3) and assess the time savings they offer. Figure 14 shows results for our first criterion that bounds the distance error. We consider 16 discrete and uniform moments t_i, where t_{16} is chosen to be equal to the maximum time it takes to find an exact answer in the training sample. For each t_i, we train an individual 2D kernel density and use 95% prediction intervals ($\theta = .05$) for estimation. The coverage (ratio of queries for which $\epsilon_Q < \epsilon$) exceeds its nominal level (95%) for all datasets, which suggests that results might be conservative. The reason is that stopping could only occur at certain moments. For higher granularity, one can use a larger number of discrete moments. The ratio of exact answers is close to 95% for $\epsilon = .01$ but becomes unstable for $\epsilon = .05$, dropping to as low as 70% for the seismic dataset. On the other hand, this results in considerable time savings, especially for DSTree: higher than 90% for the synthetic, SALD, and deep1B datasets.

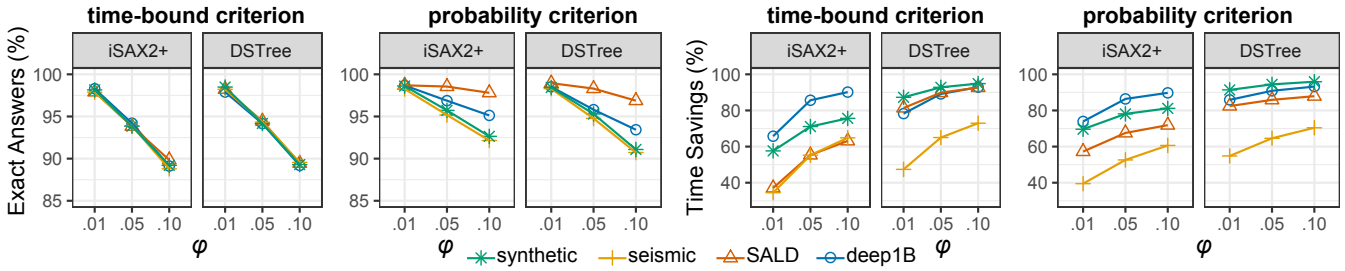

Figure 15: Evaluation of stopping criteria that bound (ϕ) the probability/ratio of non-exact answers. We measure their ratio of exact answers and their time savings (%). For all conditions, we use $n_r = 100$ training queries.

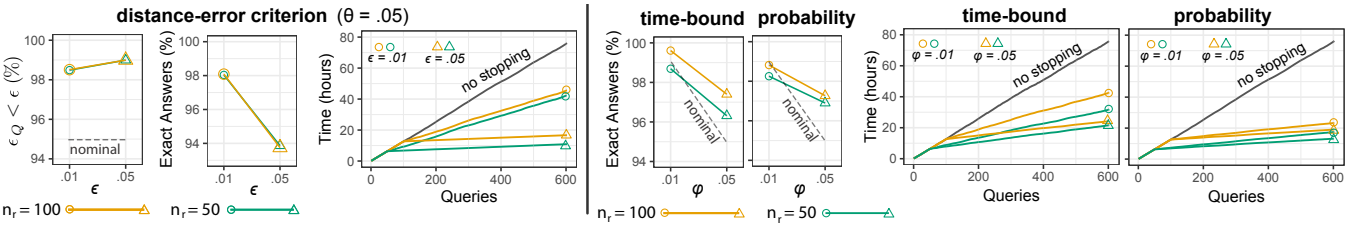

Figure 16: Performance of our stopping criteria for a real workload of 600 queries (deep1B dataset and DSTree). We draw $n_r = 50$ or 100 random queries for training. We then apply a criterion to the remaining queries. Answers with $\epsilon_Q < \epsilon$ and exact ones (%) are measured for those "testing" queries. (We report means over 100 repetitions.)

Figure 15 compares the two stopping criteria (time bound and probability) that control the ratio of exact answers. For the probability criterion, we consider again 16 discrete moments to stop the search, as above. The time-bound criterion results in mean exact answer ratios that are very close to nominal levels, while the probability criterion is rather conservative. However, the time gains of the two techniques are comparable. For iSAX2+, the probability criterion achieves both a higher accuracy and higher time savings than the probability criterion. In contrast, both criteria lead to similar time savings for DSTree, reducing query times by up to 95%.

Training Costs vs. Gains. Training linear models with 100 queries is instantaneous, while learning 16 – 20 density functions with 2D kernel density estimation takes no more than 4-6 seconds on a regular laptop. Of course, our approach requires the full execution of the training queries. For a detailed analysis of the costs of exact similarity search with iSAX2+ and DSTree, we refer the reader to the results of Echihabi et al. [30]. Depending on the size and type of the dataset, processing 100 queries can take some dozens of minutes (50 GB datasets), or several hours (250 GB datasets). Nevertheless, the higher this initial training cost, the higher the benefit is when users later execute their queries.

Figure 16 shows the results for the first 600 queries extracted from a real query workload that comes with the deep1B dataset. (Experiment conducted on a server with two Intel Xeon E5-2650 v4 2.2GHz CPUs, 75GB of RAM.) Results are based on 100 repetitions; each time we draw at random 50, or 100 queries for training. We then apply our stopping criteria to accelerate the remaining queries.

The results show that our approach leads to significant performance improvements, while coverage (exact answers, or answers with $\epsilon_Q < \epsilon$) is very close to, or higher than the nominal levels, even with training sizes of only 50 queries. For example, this workload of 600 queries would normally take 76 hours to execute with the DSTree index, but we can execute it in less than 20 hours (probability criterion; including training time), while achieving an average coverage of more than 95% exact answers. Finally, we note that, as the trends in the graphs show, the time savings and speedup offered by our progressive similarity search techniques will increase as the size of the query workload increases.

7 CONCLUSIONS

Providing progressive answers for data series similarity search along with probabilistic quality guarantees is an important problem. We describe the first scalable and effective solutions to this problem, and demonstrate their applicability using synthetic and real datasets from diverse domains.

Acknowledgments. Work partially supported by program Investir l'Avenir and Univ. of Paris IDEX Emergence en Recherche ANR-18-IDEX-0001, EU project NESTOR (MSCA #748945), and FMJH Program PGMO with EDF-THALES.

REFERENCES

[1] 2020. Suplmentary Material. http://helios.mi.parisdescartes.fr/~themisp/progrss/

[2] Marco Angelini, Giuseppe Santucci, Heidrun Schumann, and Hans-Jörg Schulz. 2018. A Review and Characterization of Progressive Visual Analytics. *Informatics* 5 (2018), 31.

[3] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. 1998. An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions. *J. ACM* 45, 6 (Nov. 1998), 891–923. https://doi.org/10.1145/293347.293348

[4] Johannes Aßfalg, Hans-Peter Kriegel, Peer Kröger, Peter Kunath, Alexey Pryakhin, and Matthias Renz. 2006. Similarity Search on Time Series Based on Threshold Queries. In *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006, Proceedings*. 276–294. https://doi.org/10.1007/11687238_19

[5] Artem Babenko and Victor S. Lempitsky. 2015. The Inverted Multi-Index. *IEEE Trans. Pattern Anal. Mach. Intell.* 37, 6 (2015), 1247–1260.

[6] Sriram Karthik Badam, Niklas Elmqvist, and Jean-Daniel Fekete. 2017. Steering the Craft: UI Elements and Visualizations for Supporting Progressive Visual Analytics. *Comput. Graph. Forum* 36, 3 (June 2017), 491–502. https://doi.org/10.1111/cgf.13205

[7] Anthony J. Bagnall, Richard L. Cole, Themis Palpanas, and Konstantinos Zoumpatianos. 2019. Data Series Management (Dagstuhl Seminar 19282). *Dagstuhl Reports* 9, 7 (2019).

[8] Gustavo E. Batista, Eamonn J. Keogh, Oben Moses Tataw, and Vinícius M. Souza. 2014. CID: An Efficient Complexity-invariant Distance for Time Series. *Data Min. Knowl. Discov.* 28, 3 (2014).

[9] Donald J Berndt and James Clifford. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *AAAIWS*. 359–370.

[10] Paul Boniol, Michele Linardi, Federico Roncallo, and Themis Palpanas. 2020. Automated Anomaly Detection in Large Sequences. In *ICDE*.

[11] Paul Boniol and Themis Palpanas. 2020. Series2Graph: Graph-based Subsequence Anomaly Detection for Time Series. *PVLDB* (2020).

[12] Sergey Brin. 1995. Near Neighbor Search in Large Metric Spaces. In *Proceedings of the 21th International Conference on Very Large Data Bases (VLDB '95)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 574–584. http://dl.acm.org/citation.cfm?id=645921.673006

[13] Paolo Buono and Adalberto Lafcadio Simeone. 2008. Interactive Shape Specification for Pattern Search in Time Series. In *AVI*.

[14] Alessandro Camerra, Themis Palpanas, Jin Shieh, and Eamonn J. Keogh. 2010. iSAX 2.0: Indexing and Mining One Billion Time Series. In *ICDM*. IEEE Computer Society, 58–67.

[15] Alessandro Camerra, Jin Shieh, Themis Palpanas, Thanawin Rakthanmanon, and Eamonn J. Keogh. 2014. Beyond One Billion Time Series: Indexing and Mining Very Large Time Series Collections with iSAX2+. *Knowl. Inf. Syst.* 39, 1 (2014), 123–151.

[16] Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. 2002. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. *ACM Trans. Database Syst.* 27, 2 (June 2002), 188–228. https://doi.org/10.1145/568518.568520

[17] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *ACM Computing Surveys (CSUR)* 41, 3 (2009), 15.

[18] Surajit Chaudhuri, Bolin Ding, and Srikanth Kandula. 2017. Approximate Query Processing: No Silver Bullet. In *SIGMOD*.

[19] Yihua Chen, Eric K. Garcia, Maya R. Gupta, Ali Rahimi, and Luca Cazzanti. 2009. Similarity-based Classification: Concepts and Algorithms. *J. Mach. Learn. Res.* 10 (June 2009), 747–776. http://dl.acm.org/citation.cfm?id=1577069.1577096

[20] Paolo Ciaccia, Alessandro Nanni, and Marco Patella. 1999. A Query-sensitive Cost Model for Similarity Queries with M-tree. In *In Proc. of the 10th ADC*. Springer Verlag, 65–76.

[21] Paolo Ciaccia and Marco Patella. 2000. PAC Nearest Neighbor Queries: Approximate and Controlled Search in High-Dimensional and Metric Spaces. In *ICDE*. 244–255.

[22] Paolo Ciaccia, Marco Patella, and Pavel Zezula. 1998. A Cost Model for Similarity Queries in Metric Spaces. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '98)*. ACM, New York, NY, USA, 59–68. https://doi.org/10.1145/275487.275495

[23] Michael Correll and Michael Gleicher. 2016. The Semantics of Sketch: Flexibility in Visual Query Systems for Time Series Data. In *VAST*.

[24] Michele Dallachiesa, Themis Palpanas, and Ihab F. Ilyas. 2014. Top-k Nearest Neighbor Search in Uncertain Data Series. *Proc. VLDB Endow.* 8, 1 (Sept. 2014), 13–24. https://doi.org/10.14778/2735461.2735463

[25] Bolin Ding, Silu Huang, Surajit Chaudhuri, Kaushik Chakrabarti, and Chi Wang. 2016. Sample + Seek: Approximating Aggregates with Distribution Precision Guarantee. In *SIGMOD*.

[26] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. 2008. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1542–1552.

[27] Tarn Duong and Martin L. Hazelton. 2005. Cross-validation Bandwidth Matrices for Multivariate Kernel Density Estimation. *Scandinavian Journal of Statistics* 32, 3 (2005), 485–506. https://doi.org/10.1111/j.1467-9469.2005.00445.x

[28] Tarn Duong, Matt Wand, Jose Chacon, and Artur Gramacki. 2019. ks: Kernel Smoothing. https://cran.r-project.org/web/packages/ks/.

[29] Karima Echihabi. 2019. Truly Scalable Data Series Similarity Search. In *VLDB PhD Workshop*.

[30] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2018. The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. *PVLDB* 12, 2 (2018), 112–127.

[31] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2019. Return of the Lernaean Hydra: Experimental Evaluation of Data Series Approximate Similarity Search. *PVLDB* 13, 3 (2019), 402–419.

[32] Roger Koenker et al. 2019. quantreg: Quantile Regression. https://cran.r-project.org/web/packages/quantreg.

[33] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. 1994. Fast Subsequence Matching in Time-Series Databases. In *SIGMOD*. ACM, New York, NY, USA, 419–429. https://doi.org/10.1145/191839.191925

[34] Jean-Daniel Fekete and Romain Primet. 2016. Progressive Analytics: A Computation Paradigm for Exploratory Data Analysis. *CoRR* abs/1607.05162 (2016). http://arxiv.org/abs/1607.05162

[35] Danyel Fisher, Steven M. Drucker, and A. Christian König. 2012. Exploratory Visualization Involving Incremental, Approximate Database Queries and Uncertainty. *IEEE CG&A* 32 (2012).

[36] Incorporated Research Institutions for Seismology. 2014. IRIS Seismic Data Access. http://ds.iris.edu/data/access/.

[37] Anna Gogolou, Theophanis Tsandilas, Themis Palpanas, and Anastasia Bezerianos. 2018. Comparing Similarity Perception in Time Series Visualizations. *IEEE TVCG* 25 (2018).

[38] Anna Gogolou, Theophanis Tsandilas, Themis Palpanas, and Anastasia Bezerianos. 2019. Progressive Similarity Search on Time Series Data. In *Proceedings of the Workshops of the EDBT/ICDT 2019 Joint Conference, EDBT/ICDT 2019, Lisbon, Portugal, March 26, 2019*. http://ceur-ws.org/Vol-2322/BigVis_5.pdf

[39] Dina Q. Goldin and Paris C. Kanellakis. 1995. On Similarity Queries for Time-Series Data: Constraint Specification and Implementation. In *CP*.

[40] Yue Guo, Carsten Binnig, and Tim Kraska. 2017. What you see is not what you get!: Detecting Simpson's Paradoxes during Data Exploration. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics, HILDA@SIGMOD*.

[41] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. 1997. Online Aggregation. In *SIGMOD*.

[42] Joseph M. Hellerstein, Elias Koutsoupias, and Christos H. Papadimitriou. 1997. On the Analysis of Indexing Schemes. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '97)*. Association for Computing Machinery, New York, NY, USA, 249–256. https://doi.org/10.1145/263661.263688

[43] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 1 (2011), 117–128.

[44] Chris Jermaine, Subramanian Arumugam, Abhijit Pol, and Alin Dobra. 2008. Scalable approximate query processing with the DBO engine. *ACM Trans. Database Syst.* 33, 4 (2008), 23:1–23:54.

[45] J. Jing, J. Dauwels, T. Rakthanmanon, E. Keogh, S.S. Cash, and M.B. Westover. 2016. Rapid Annotation of Interictal Epileptiform Discharges via Template Matching under Dynamic Time Warping. *Journal of Neuroscience Methods* 274 (2016).

[46] Paris C. Kanellakis, Sridhar Ramaswamy, Darren E. Vengroff, and Jeffrey S. Vitter. 1993. Indexing for Data Models with Constraints and Classes (Extended Abstract). In *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS '93)*. Association for Computing Machinery, New York, NY, USA, 233–243. https://doi.org/10.1145/153850.153884

[47] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems* 3, 3 (2001), 263–286. https://doi.org/10.1007/PL00011669

[48] Eamonn Keogh and M. Pazzani. 1998. An Enhanced Representation of Time Series which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In *Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98)*. ACM Press, New York City, NY, 239–241.

[49] Roger Koenker. 2005. *Quantile Regression*. Cambridge University Press. https://doi.org/10.1017/CBO9780511754098

[50] Haridimos Kondylakis, Niv Dayan, Kostas Zoumpatianos, and Themis Palpanas. 2018. Coconut: A Scalable Bottom-Up Approach for Building Data Series Indexes. *PVLDB* 11, 6 (2018), 677–690. https://doi.org/10.14778/3184470.3184472

[51] Tim Kraska. 2018. Northstar: An Interactive Data Science System. *PVLDB* 11, 12 (2018), 2150–2164.

[52] Conglong Li, Minjia Zhang, David G. Andersen, and Yuxiong He. 2020. Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In *SIGMOD*.

[53] Jessica Lin, Eamonn J. Keogh, Stefano Lonardi, and Bill Yuan-chi Chiu. 2003. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, DMKD 2003, San Diego, California, USA, June 13, 2003*. 2–11. https://doi.org/10.1145/882082.882086

[54] Michele Linardi and Themis Palpanas. 2019. Scalable, Variable-Length Similarity Search in Data Series: The ULISSE Approach. *PVLDB* (2019).

[55] Michele Linardi and Themis Palpanas. 2020. Scalable Data Series Subsequence Matching with ULISSE. *VLDBJ* (2020).

[56] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn J. Keogh. 2018. Matrix Profile X: VALMOD - Scalable Discovery of Variable-Length Motifs in Data Series. *SIGMOD*.

[57] Yury A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[58] Miro Mannino and Azza Abouzied. 2018. Expressive Time Series Querying with Hand-Drawn Scale-Free Sketches. In *CHI*.

[59] Luana Micallef, Hans-Jörg Schulz, Marco Angelini, Michaël Aupetit, Remco Chang, Jörn Kohlhammer, Adam Perer, and Giuseppe Santucci. 2019. The Human User in Progressive Visual Analytics. In *Short Paper Proceedings of EuroVis'19*. Eurographics Association, 19–23. https://doi.org/10.2312/evs.20191164

[60] Katsiaryna Mirylenka, Michele Dallachiesa, and Themis Palpanas. 2017. Data Series Similarity Using Correlation-Aware Measures. In *SSDBM*.

[61] Dominik Moritz, Danyel Fisher, Bolin Ding, and Chi Wang. 2017. Trust, but Verify: Optimistic Visualizations of Approximate Queries for Exploring Big Data. In *CHI*.

[62] Dominik Moritz, Bill Howe, and Jeffrey Heer. 2019. Falcon: Balancing Interactive Latency and Resolution Sensitivity for Scalable Linked Visualizations. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, Article 694, 11 pages. https://doi.org/10.1145/3290605.3300924

[63] J. Nielsen. [n.d.]. Response times: The 3 important limits. https://www.nngroup.com/articles/response-times-3-important-limits/.

[64] Themis Palpanas. 2015. Data Series Management: The Road to Big Sequence Analytics. *SIGMOD Record* 44, 2 (2015), 47–52. https://doi.org/10.1145/2814710.2814719

[65] Themis Palpanas. 2020. Evolution of a Data Series Index - The iSAX Family of Data Series Indexes. *Communications in Computer and Information Science (CCIS)* (2020).

[66] Themis Palpanas and Volker Beckmann. 2019. Report on the First and Second Interdisciplinary Time Series Analysis Workshop (ITISA). *SIGMOD Rec.* 48, 3 (2019).

[67] Botao Peng, Panagiota Fatourou, and Themis Palpanas. 2020. MESSI: In-Memory Data Series Indexing. In *ICDE*.

[68] Botao Peng, Themis Palpanas, and Panagiota Fatourou. 2018. ParIS: The Next Destination for Fast Data Series Indexing and Query Answering. *IEEE BigData* (2018).

[69] Botao Peng, Themis Palpanas, and Panagiota Fatourou. 2020. ParIS+: Data Series Indexing on Multi-core Architectures. *TKDE* (2020).

[70] Nathaniel Phillips. 2017. A Companion to the e-Book "YaRrr!: The Pirate's Guide to R". https://github.com/ndphillips/yarrr.

[71] Sajjadur Rahman, Maryam Aliakbarpour, Ha Kyung Kong, Eric Blais, Karrie Karahalios, Aditya Parameswaran, and Ronitt Rubinfield. 2017. I'Ve Seen "Enough": Incrementally Improving Visualizations to Support Rapid Decision Making. *Proc. VLDB Endow.* 10, 11 (Aug. 2017), 1262–1273. https://doi.org/10.14778/3137628.3137637

[72] Thanawin Rakthanmanon, Bilson J. L. Campana, Abdullah Mueen, Gustavo E. A. P. A. Batista, M. Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn J. Keogh. 2012. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. In *KDD*. ACM, 262–270.

[73] Thanawin Rakthanmanon, Eamonn J Keogh, Stefano Lonardi, and Scott Evans. 2011. Time Series Epenthesis: Clustering Time Series Streams requires Ignoring Some Data. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 547–556.

[74] Pedro Pereira Rodrigues, João Gama, and João Pedro Pedroso. 2006. ODAC: Hierarchical Clustering of Time Series Data Streams. In *SDM*. SIAM, 499–503.

[75] Hans-Jörg Schulz, Marco Angelini, Giuseppe Santucci, and H Schumann. 2016. An Enhanced Visualization Process Model for Incremental Visualization. *IEEE Transactions on Visualization and Computer Graphics* 22 (07 2016), 1830–1842. https://doi.org/10.1109/TVCG.2015.2462356

[76] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless Data Exploration with Zenvisage: An Expressive and Interactive Visual Analytics System. *Proc. VLDB Endow.* 10, 4 (Nov. 2016), 457–468. https://doi.org/10.14778/3025111.3025126

[77] Charles D. Stolper, Adam Perer, and David Gotz. 2014. Progressive Visual Analytics: User-Driven Visual Exploration of In-Progress Analytics. *IEEE TVCG* 20 (2014).

[78] Edward R. Tufte. 1986. *The Visual Display of Quantitative Information.*

[79] Cagatay Turkay, Erdem Kaya, Selim Balcisoy, and Helwig Hauser. 2017. Designing Progressive and Interactive Analytics Processes for High-Dimensional Data Analysis. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (Jan. 2017), 131–140. https://doi.org/10.1109/TVCG.2016.2598470

[80] Southwest University. 2017. Southwest University Adult Lifespan Dataset (SALD). http://fcon_1000.projects.nitrc.org/indi/retro/sald.html.

[81] Skoltech Computer Vision. 2018. Deep billion-scale indexing. http://sites.skoltech.ru/compvision/noimi.

[82] Abraham Wald. 1945. Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics* 16, 2 (06 1945), 117–186. https://doi.org/10.1214/aoms/1177731118

[83] Matt P. Wand and Michael C. Jones. 1993. Comparison of Smoothing Parameterizations in Bivariate Kernel Density Estimation. *J. Amer. Statist. Assoc.* 88, 422 (1993), 520–528. https://doi.org/10.1080/01621459.1993.10476303

[84] Matt P. Wand and Michael C. Jones. 1994. Multivariate plug-in bandwidth selection. *Computational Statistics* 9, 2 (1994), 97–116. http://oro.open.ac.uk/28244/

[85] Yang Wang, Peng Wang, Jian Pei, Wei Wang, and Sheng Huang. 2013. A Data-adaptive and Dynamic Segmentation Index for Whole Matching on Time Series. *PVLDB* 6, 10 (2013), 793–804.

[86] T. Warren Liao. 2005. Clustering of Time Series Data — A Survey. *Pattern Recognition* 38, 11 (2005), 1857–1874.

[87] Sai Wu, Beng Chin Ooi, and Kian-Lee Tan. 2013. Online Aggregation. In *Advanced Query Processing, Volume 1: Issues and Trends.* 187–210.

[88] Djamel-Edine Yagoubi, Reza Akbarinia, Florent Masseglia, and Themis Palpanas. 2020. Massively Distributed Time Series Indexing and Querying. *TKDE* 32, 1 (2020).

[89] E. Zgraggen, A. Galakatos, A. Crotty, J. Fekete, and T. Kraska. 2017. How Progressive Visualizations Affect Exploratory Analysis. *IEEE Transactions on Visualization and Computer Graphics* 23, 8 (Aug 2017), 1977–1987. https://doi.org/10.1109/TVCG.2016.2607714

[90] Emanuel Zgraggen, Zheguang Zhao, Robert C. Zeleznik, and Tim Kraska. 2018. Investigating the Effect of the Multiple Comparisons Problem in Visual Analysis. In *CHI.*

[91] Kostas Zoumpatianos, Stratos Idreos, and Themis Palpanas. 2015. RINSE: Interactive Data Series Exploration with ADS+. *PVLDB* 8, 12 (2015), 1912–1915. https://doi.org/10.14778/2824032.2824099

[92] Kostas Zoumpatianos, Stratos Idreos, and Themis Palpanas. 2016. ADS: The Adaptive Data Series Index. *VLDB J.* 25, 6 (2016), 843–866. https://doi.org/10.1007/s00778-016-0442-5

[93] Kostas Zoumpatianos, Yin Lou, Themis Palpanas, and Johannes Gehrke. 2015. Query Workloads for Data Series Indexes. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015.* 1603–1612. https://doi.org/10.1145/2783258.2783382