Approximate Search on Massive Spatiotemporal Datasets

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Ivan Brugere

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

VIPIN KUMAR

August, 2012

Acknowledgements

This work has been supervised by Professor Vipin Kumar, Karsten Steinhaeuser, and Shyam Boriah, all of whom have been instrumental in guiding my academic and intellectual growth. I have been fortunate to be a part of a collaborative group, all of whom have challenged me and contributed to this thesis in direct conversation or more intangible ways. Including the aforementioned, I would like to thank Ashish Garg, Varun Mithal, Yashu Chamber, Xi Chen, Lydia Manikonda, and Anuj Karpatne. Chapters 4 and 5 of this work have emerged in part from conversations with Professor Eamonn Keogh at the University of California – Riverside, I would like to thank him for his availability to students outside of UCR.

Abstract

Efficient time series similarity search is a fundamental operation for data exploration and analysis. While previous work has focused on indexing progressively larger datasets and has proposed data structures with efficient exact search algorithms, we motivate the need for approximate query methods that can be used in interactive exploration and as fast data analysis subroutines on large spatiotemporal datasets. This thesis formulates a simple approximate range query problem for time series data, and proposes a method that aims to quickly access a small number of high-quality results of the exact search resultset. We formulate an anytime framework, giving the user flexibility to return query results in arbitrary cost, where larger runtime incrementally improves search results. We propose an evaluation strategy on each query framework when the false dismissal class is very large relative to the query resultset and investigate the performance of indexing novel classes of time series subsequences.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Time series data is ubiquitous in contemporary scientific and commercial domains. Historical transaction data and item purchase histories help private firms forecast demand and personalize recommendation systems, while massive scientific data measures biological, astronomical, and ecological functions in different natural systems. The problem of time series similarity search aims to efficiently access time series data that is sufficiently 'similar' to a query time series object. Use cases for similarity search within earth science domains include grouping and characterizing phenology of vegetation, and studying the extent of change in ecosystems over time [16, 30].

For very large datasets, an exhaustive scan of the data is computationally prohibitive. Previous work in scalable time series indexing [8, 31] has overcome this by producing data structures which reduce comparisons to a small set of candidate data objects by a sub-linear search. The development of these data structures has used two broad strategies. One family exploits dimensionality reduction techniques to construct a projected data space which can be indexed by a spatial access method such as R-tree [4, 10, 19, 20]; the prototypical example is Generalized Multimedia Indexing (GEMINI) [15]. The second family exploits coarse time series representations to split the data space, which is indexable due to hierarchical fidelity of the approximation to the original data [11, 13, 37]. Both strategies are used to efficiently answer two basic queries defined relative to a specified distance measure: (1) the k-nearest neighbor search (k-NN), which returns the k time series objects of least distance to the query,

and (2) the exact range search, which, given a range threshold, returns *all* time series objects within this distance to the query. Most previous work has focused on the exact search problem because these methods provide guarantees on completeness and correctness.

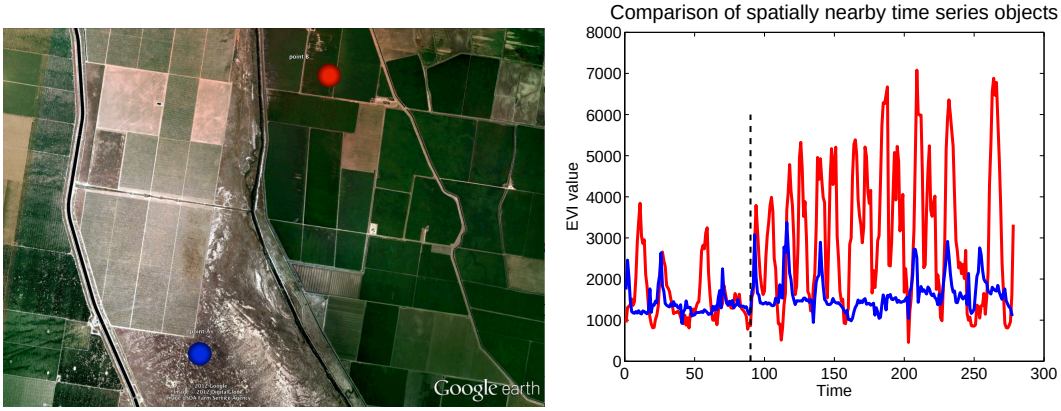

Figure 1.1: (Left) The earth imagery with red and blue points indicating time series location centroids, (Right) The time series for the red and blue locations. The dashed line at $t = 90$ denotes the time step where the temporal dynamics of the two time series diverge in similarity.

In some situations, exact queries are not relevant when a complete resultset is not necessary or is relatively expensive to find. For example, our motivation relates to a collaborative earth science research platform. Users should be able to launch *interactive queries* from an interface, and arbitrarily refine or loosen similarity constraints of the resultset. The user may only require the approximate prevalence of a particular time series pattern in a spatial region, and only expects the typical resultset is sufficiently large and of high quality. A second use case aims to build a 'null' distribution relative to a novel query segment, capturing the natural variation of the majority class in a spatial region. In this case, 'k' in a k-NN query may be so large as to cause poor performance with 'best-so-far' pruning strategies. We formulate the problem of approximate range query, which returns only a small portion of high quality results within a particular distance range γ, in much less time than exact search. In contrast to exact query methods, this method does not guarantee either (1) that the k-nearest neighbors are returned (for any k), nor (2) that the query has guarantees on recall, meaning it

returns an arbitrary portion of *possible* query results within the range. However, the proposed method experimentally provides sufficiently large resultsets with guarantees on similarity. While previous work has proposed the approximate query as subroutine to building an exact resultset [37], we formulate the problem as a stand-alone objective to be rigorously evaluated, and extend a previously proposed data structure to be queried on a specified range threshold.

In this work we build a time series indexing data structure for efficient, approximate range queries. We introduce the following contributions:

1. A simple formulation of the fast range query problem
2. An approximate range query method which returns high quality, partial resultsets relative to the exact range query
3. An evaluation methodology measuring the quality of query resultsets relative to the exact range query, in the presence of many false dismissals
4. An exploration of spatiotemporal earth science datasets using querying, and analysis illustrating challenges for query performance across different classes of time series segments
5. The proposal and analysis of a flexible searching paradigm which allows the user to spend an arbitrary computation cost to return a result

Section 1.1 gives a review of specific related work extended in this paper. Chapters 2 and 3 propose and evaluate an approximate range query paradigm focused on retrieving a small number of high-quality results on an interactive time scale, evaluating against a base approximate query and an exhaustive linear scan. Chapters 4 and 5 propose and evaluate an anytime query paradigm which has preference to rare time series sequences in the dataset. Section 6 outlines the limitations of our work and possible future directions. All defined symbols in this paper are summarized in Table 1.1.

1.1 Background

In this section, we outline related work in the areas of time series representations, distance measures, and similarity search indexing. We summarize in greater detail the

work we extend, so that our extension is sufficiently understandable.

1.1.1 Distance Measures and Time Series Representations

Time series distance measures are methods used to calculate the 'similarity' of two time series objects. Figure 1.1 (Right) illustrates that intuitively we might consider that the blue and red segments before the change event ($t = 90$) are more similar than the segments after the change event. This is captured by the Minkowski distance: $d(T,S) = (\sum_{i=1}^{|T|} |S^i - T^i|^p)^{\frac{1}{p}}$, where $p = 1$ defines the 'Manhattan' distance and $p = 2$ defines the 'Euclidean' distance. The choice of a particular distance measure corresponds to the assumptions of what time series characteristics constitute similarity. Lhermitte et al. [28] demonstrate that distance measures are differentially sensitive to scaling, shifting, or changes in shape of time series sequences.

Standard distance measures include Manhattan, Euclidean, correlation, and dynamic time warping (DTW) [6]. DTW in particular has been extensively used [3, 7, 21, 33, 35] and exhibits more flexibility for many data mining tasks [13, 14]. Research on dynamic time warping has focused on optimization using path constraints through the time warping matrix [18, 23, 26] or finding fast approximations of the time-warped distance [23, 36, 40].

Data access methods and data mining problems often rely on transformations of raw data to representations using approximations which can be quickly compared by exploiting bounding properties of a particular distance measure (e.g., Euclidean [25, 37]); common representations use discrete Fourier transforms [15], wavelets [10], piecewise linear approximation [22, 24], and piecewise aggregate approximation [39].

This initial work uses a simple distance measure of mean of Manhattan distance, because it is efficient to compute, the bounding of numerous data representations under this measure are straightforward, and it is less sensitive to outlier values than higher order L_p distances. In this work, the *mean* is used only for interpretability of distances. Non-lockstep measures such as DTW are computed on order $O(n^2)$ and thus are often approximated, and not as clearly applicable for the data structure in this work.

1.1.2 Time Series Indexing

Shieh and Keogh [37] propose using Symbolic Aggregate Approximation (SAX) representation [29] to index time series data for similarity search. We give a simplified overview of the building phase and structure of the index. The authors build an index tree which represents the data with a larger number of bits as the representation associated with that object is moved downward in the tree by split operations. Internal nodes yield search paths through the tree and leaf nodes store time series data. The authors use an equal-frequency discretization of k bins over the data space to create an alphabet of k symbols (referred to as cardinality k, where k are powers of two, $k = 2^l$). This representation is encoded by the function $W(S, k)$. The time series is thus transformed into a sequence of l-bit (k-cardinality) symbols, w_k^h (the h-th symbol encoding of S, with cardinality k). Using W, each time series object is encoded at a base cardinality (C^1) and grouped in nodes of common base encodings at the top of the tree (e.g. a node for each unique binary encoding given all time series objects, with $2^{|S|}$ possible nodes). For building the tree downward, there are exactly two symbols produced at a higher cardinality by 'promoting' a particular symbol (from l bits to $l + 1$ bits). This can be thought of as evenly bisecting the range of values represented by w_k in the original data space. The authors use this property to promote a particular time step t from the 2^l-th to the 2^{l+1}-th cardinality and split the grouped time series on the two possible encodings of $w_{2^{l+1}}^t$. The authors use a naive round-robin splitting policy and provide a threshold, α, serving as a stopping condition so that for every leaf L, $|L| \leq \alpha$. An exact 1-NN search is defined on the index, using best-so-far pruning. However, the authors report only non-interactive runtimes under 1-NN exact search (e.g. 5.8 minutes on datasets of comparable size to our evaluation).

Camerra et al. [8] extend the indexing of SAX representations of time series data. This method focuses on improving index build time to make feasible indexing on the order of a billion time series objects. Though these studies focus on producing index structures for progressively larger time series datasets, they do not focus on improving index performance for dataset sizes already surpassed. These methods provide no assurance of similarity for leaf nodes and have no mechanism for improving similarity in leaf nodes at the cost of further computation.

Previous work in similarity search aimed at spatiotemporal earth science datasets

has focused on exploiting the intrinsically spatial character of this time series data. Zhang et al. [41] propose a data structure designed on smooth fields with high spatial autocorrelation. In the presence of spatial heterogeneity as expressed in other datasets such as vegetation index (see Figure 1.1), the tree depth grows very large, and search resultsets are scattered across many small nodes requiring an exact search to traverse many branches. In contrast, our method groups spatially-distant segments with common SAX representations, and uses auxiliary spatial indexing to impose spatial constraints in spatially-oriented time series datasets. A family of object-based hierarchical grouping has sought to spatially segment data according to spatially-contiguous similarity at varying similarity thresholds, without building a search index [27]. These methods are also sensitive to spatial heterogeneity, which may produce fragmented groupings when similar time series are found non-contiguously in space.

Table 1.1: Table of notation definitions. Where applicable, subscripts denote cardinality and superscripts denote dimensionality (time-step).

Symbol	*Definition*
$\mathbf{D}$	The global dataset of size $m \times n \times d$, where $m \times n$ constitute 2-dimensional space with time series objects each of length d.
Q, S, T	Time series objects, $Q, S \in \mathbf{D}$ where $S = (S^1, S^2, ..., S^k, ..., S^{\|S\|}), S^k$ the k-th observation in S and $\|S\|$ the length of S. S is an arbitrary time series object, and $S \neq T$, Q is a query time series object
$S^{k:l}$	The time series subsequence on S: $(S^k, ..., S^l), l > k$
$d(S,T)$	The distance calculation for objects S, T, $d(S,T) \geq 0$. Reported in mean of L_1: $d(T,S) = Mean(\sum \|S^i - T^i\|)$, $i = 1...\|T\|$
$d_p(D)$	The pairwise distance calculated over the collection of time series objects.
R	The resultset following the execution of a query using Q.
R^*	The exhaustive resultset using Q, relative to distance threshold γ, $R^* = \{\forall S \in \mathbf{D} : d(Q,S) \leq \gamma\}$.
N_k^i, N_k^j, L_k	Nodes $N, L \in I$ at depth k, N_k^i, N_k^j distinct, i.e. $i \neq j$. L denotes only leaf nodes.
$\|L\|$	The number of time series objects contained within leaf L, (note the overloading of this operator).
C, c	C, the cardinality expansion list for use in SAX indexing, associated with index I, e.g. $C = (2^1, 2^2, ..., 2^6)$; c, a cardinality vector indexing into C, $c_1, ..., c_k, ..., c_{\|S\|}$ where $1 \leq c_k \leq \|C\|$
$W(S,c)$	The SAX representation of time series S where W^k is S^k approximated in cardinality C^{c_k} (Or: interpret W as a $\|C\| \times \|S\|$ symbolization matrix of S which c indexes into)
w_k^i	The i-th symbol (or 'word') in a symbolized sequence, at cardinality k.
$\|w_k\|$	The range size in the unsymbolized space encompassed by symbol w_k, e.g. for a uniformly distributed data space, $\mathbf{D}$ with range $[0,1]$, $\|1_2\| = 0.5$, $\|2_2\| = 0.5$
b_w, b_p, g_m	$b_w = mean(\|w_{c^i}\|)$, $i = 1...\|S\|$, $b_p = max(d_p(N.data()))$ the word range sum (WRS) bound and pairwise bound on Node N; $g_m = median(d_p(N.data()))$ the *guidance* on Node N.
$Z(S)$	The z-normalized time series of S: $Z(S) = \frac{S^k - \mu_S}{\sigma_S}$, for $k = 1, ..., \|S\|$

Chapter 2

Approximate Search: Methods

2.1 Approximate Search

In previous work on approximate search, 'approximate' has meant to provide an error term or probability related to the approximate nearest neighbor result or the range boundary [5, 17, 32]. A well-performing method of these problem formulations aims to reduce these error terms. This is not suitable for the case where there may be many false dismissals, because error terms primarily evaluate the quality of the positive class irrespective of the false dismissals, and thus varies by the density of the distance-to-query values. Instead, we are interested in the *relative* quality of results against the false dismissal class, and a simple problem formulation which focuses on this purpose. We define the approximate range query on Q and distance threshold γ to be $R \subseteq R^*$, where R^* is the exact search resultset on Q and threshold γ.

An approximate search method proposed in Shieh and Keogh [37] follows the basic 'promotion' sequence given in Section 1.1, where the authors encode a query Q at the base cardinality and following the sequence until a leaf node is reached. This leaf constitutes the resultset. Though we use this method as a base search approach for comparison, it must be stressed that this method does not directly answer the approximate *range* query because the results may be arbitrarily dissimilar.

This approximate search method has a sensitivity to the splitting parameter α. Depending on the size and characteristics of the dataset, after the splitting phase using a large α, a node may be 'under-split', meaning the more natural splitting can be at

a higher cardinality (yielding dissimilar groupings within a node). Similarly, a small α 'over-splits' the data, yielding small resultsets and increased number of false dismissals since 'cousin' nodes at the same depth in the tree may be very similar and the search only returns a single node amongst these. Finding a natural α by experimentation is a poor strategy as the index build time is very large, and would only improve intra-node similarity in the average case, while some classes may remain unnaturally split. We address this issue by allowing α to be arbitrarily small and recovering the 'over-split' result using an approximate range query, where the traversal stopping condition is given by the desired range. This method yields higher performance, in terms of reduced false dismissals, with minor overhead cost.

Problem Definition 1 (Approximate Range Search on *I*)
Input: Query time series Q, distance threshold γ, query index structure I
Output: Result set of time series data R
Where: $\max(d(R, Q)) \leq \gamma$

2.2 Bounds on Query Index Structure

Our strategy to fulfill this search is that during the splitting phase, we store at each node the upper bound of the distance of any two time series objects of the node's SAX representation. This bound can then be used as a stopping condition in traversal. A SAX-encoded time series object S is some sequence of words at differing cardinalities, $S = (w_{c^1}, ..., w_{c^{|S|}})$. We can define a bound on the Manhattan distances at a node N as a sum of the 'range' of each word (comparing the maximum and minimum possible unsymbolized data values encompassed by that word). Therefore we have $b_w = Mean(|w_{c^i}|)$, $i = 1...|S|$, where we call b_w the word range sum (WRS) bound on node N. This bounding holds as an upper bound for any child node and further descendants, $b_w \geq b_w(N.children)$ because $|w^i_{2^l}| = 2|w^i_{2^{l+1}}|$.

If we assume that a query is a subsequence in the indexed data space, we can improve the tightness of this bounding. In this case, the pairwise distance bound on Node N is $b_p = max(d_p(N.data))$. This holds because $Q \in N.data$. In many use cases this may be a practical assumption, and assures no false positives are incurred. However, this bounding has limited scalability because the pairwise runtime is $O(|N|^2)$, and is

infeasible for large $|N|$.

We introduce a threshold κ, under which b_p is feasible. For nodes where $|N| \leq \kappa$ we do this pairwise computation, otherwise we approximate it with b_w. If the b_w is loose compared to b_p, the search stopping condition using the WRS bound will be met deeper in the tree, potentially incurring more false dismissals. This threshold can practically be set by the maximum targeted resultset size, (i.e. $\kappa = 1000, \kappa > |R|$). This gives the user flexibility to expend greater computational cost of the pairwise computation on larger nodes, yielding a tighter bound higher in the tree and more accurate stopping criteria earlier in the traversal.

The above bounding methods can be used to ensure that no false-positives are incurred. However, even in the case of using the tighter b_p, the bounding may be loose, requiring a large range threshold γ to be used in searching. To mitigate this, we define a median 'guidance' on the intra-node distance, which can be used as a finer threshold in searching. Though many of the elements in the resultset are considered false-positives relative to this guidance, it gives a more intuitive interpretation on the expected similarity of results. Where $|N| > \kappa$, the guidance is defined as $g_m = \frac{b_w}{2}$, using the WRS bound above. Otherwise, we define the guidance as $g_m = median(d_p(N.data))$. Note we use differing notation to stress that g_m is not a bound. For clarity, we summarize these strategies with the relation: $g_m \leq b_p \leq b_w$.

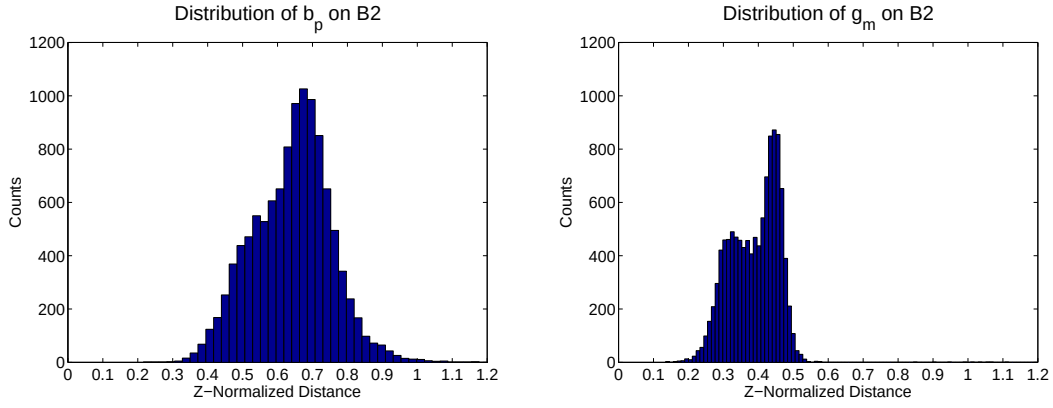

Figure 2.1: Distribution generated by b_p responses (Left) and g_m responses (right) over 20,000 queries following the promotion sequence to the leaf (i.e. base approximate search).

Figure 2.1 reports the distribution of b_p and g_m by traversing to the leaf on an input query. We observe that for z-normalized values, dissimilarity over 0.4 units degrades in practical use, yielding poor matching of the temporal dynamics of the query. Given this interpretation, the distribution of g_m responses (with $median = 0.4011$) would ensure that half the queries are practically useful, while the bounding b_p provides undesired stopping thresholds for nearly 90% of the data. For simplicity, we omit results using b_w because in our context b_p serves as a valid lower bound for it.

Given the above assurances on intra-node distance, we can provide an algorithm for the Approximate Range Search problem on the iSAX search index (Algorithm 1).

Algorithm 1: Approximate Range Search

Input: Query time series Q, stopping threshold γ, time series similarity index structure I with intra-node distance assurance.

Output: Resultset of time series data R

```
1  nodeStack = Stack();
2  N = I.get([c_base, W(Q, c_base)]);
3  while ¬N.isleaf & N.bound ≥ γ do
4  |   i = diffIdx(N.child.key, N.key);
5  |   c_new = node.key.c;
6  |   c^i_new += 1;
7  |   N = I.get([c_new, W(Q, c_new)]);
8  end
9  if N.isleaf then R = N.data else
10 |   nodeStack.push(N.children.key);
11 |   while ¬nodeStack.empty() do
12 |   |   N = I.get(nodeStack.pop());
13 |   |   if N.isleaf then R.concat(N.data) else nodeStack.push(N.children)
14 |   end
15 end
```

This method traverses the tree until either the γ similarity threshold is reached, or a leaf is found. To begin, it retrieves the node N at base cardinality, (line 2), and traverses until the stopping threshold γ is satisfied, (line 3; this node 'bound' can also be g_m). If a node is an interior node, it can recover the split index by comparing its cardinality vector to that of its children (line 4). It then promotes this time step and looks up the new key-string in the hash data structure until a satisfying node is reached (lines 5-7;

returns an empty resultset if the node is a leaf and does not satisfy γ). If the satisfying node N is a leaf, the method returns its data (line 9), otherwise it concatenates all node data of the subtree under N (lines 10-14). Using this method we can see that a smaller leaf size α does not increase false dismissals, it will only require more steps to concatenate split nodes (lines 10-14). Note that the situation can arise where the method returns an empty set, while positive results at range γ exist elsewhere in the tree.

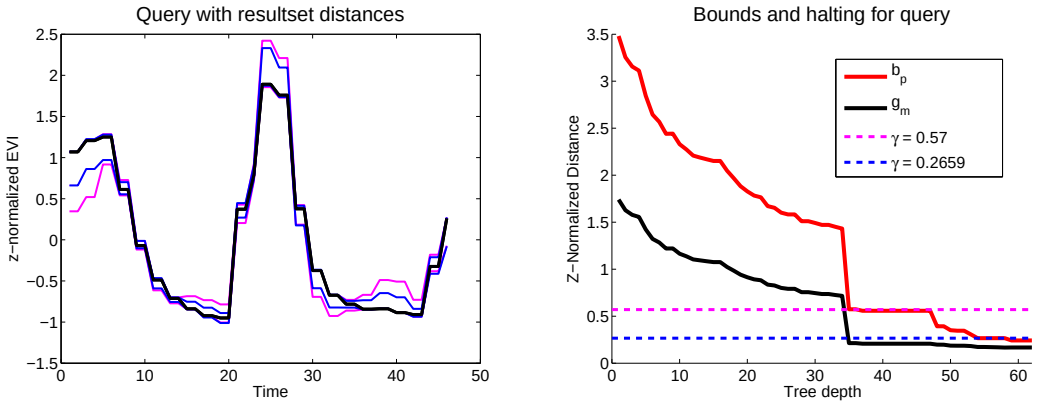

Figure 2.2: (Left) A query time series segment (black) and the average signed distance-to-query of the resultset, per time step, with specified stopping threshold on the bounding: $\gamma = 0.57$ and $\gamma = 0.2659$ for magenta and blue distances, respectively. (Right) The b_p (red) and g_m (black) values at each node in the traversal, with magenta and blue thresholds indicated. The abrupt decrease in bounding, near $depth = 35$ corresponds to changing from b_w to b_p according to the κ threshold.

We summarize this section by illustrating an example search. Figure 2.2 (left) compares an example query Q in black against the average signed distances of the query to the resultset, per time step. We plot: $Q + \epsilon^+$, $Q + \epsilon^-$ to show the average distance incurred per time step. The magenta and blue lines indicate the distances at stopping condition $\gamma = 0.57$ and $\gamma = 0.2659$, respectively. These γ values are chosen because they compare the resultset after the drop in the bounding, with the bounding at the leaf (maximum depth). So, we see that the query is under-estimated by R at the beginning of Q, and over-estimated near the center of Q, and generally the distances-to-query at $\gamma = 0.57$ are larger. Note that though these bounds were used, the maximum distance between the query and resultset may be lower: $max(d(Q, R)) \leq b_p$.

For example, $max(d(Q,R)) = 0.45$, $max(d(Q,R)) = 0.26$ for the respective thresholds. Figure 2.2 (right) shows the decreasing bound and guidance g_m as the tree is traversed. For Algorithm 1, γ can be interpreted as the value on the y-axis where traversal stops, intersecting b_p or g_m. At a particular depth, those elements where $g_m < d(Q,R) \leq b_p$ constitute the false positives when using g_m as the traversal stopping condition.

Chapter 3

Approximate Search: Performance and Evaluation

We present three evaluations on the proposed indexing framework. First, we compare the distribution of b_p, g_m, and final depth of searches over each dataset. This shows the fraction of data that we can expect to have search results of desired quality. Second, our evaluation takes into account that perhaps *many* false dismissals occur (often by orders of magnitude relative to the resultset) for a particular stopping threshold γ; our evaluation methodology provides a detailed comparison of the relative quality of the resultset against the set of false dismissals. Third, we show that we return larger resultsets than the base query method. Combining the three, we can conclude that for a class of queries, our method is more flexible and yields larger resultsets of high quality elements than in previous work.

3.1 Data and Preprocessing

For evaluation, we index an Enhanced Vegetation Index (EVI) product (MOD13A2) from NASA's Earth Observation System (EOS) [1] moderate resolution imaging spectroradiometer (MODIS) instrument. The dataset is freely distributed through the Land Processes Distributed Active Archive Center (LP DAAC) [2]. This dataset is a surrogate for the 'greenness' observed globally on the earth's surface, and is generated from daily observations processed to regular 16-day scenes (with an annual period $p = 23$), with

approximately 1-km spatial resolution. The dataset can be thought of as a data cube, with scenes of $m \times n$ pixels, stacked at a height of d. A particular location on earth has a time series of d historical observations. We adapt the piecewise aggregate approximation (PAA) [24] for fixed-length periodic data, so that k equal-length segments of length l are produced on each period. This method ensures values are consistently aggregated along annual boundaries. If the period p is not divisible by k, we aggregate the k-1 segments in the expected way. For the k^{th} segment, we overlap it appropriately with the previous segment so it is of length l. We informally observe that $l = 2$ and $l = 3$ is effective to reduces some fragmentation in the SAX grouping introduced by noise or boundary issues, without significant degradation in time series dynamics.

We index the z-normalized time series to provide a consistent way to define query range thresholds for time series with high or low average vegetation response. Z-normalization is used only to provide a more principled evaluation; indexing without z-normalization differentiates similar shapes at different mean values. We use a splitting threshold $\alpha = 20$, and analyze spatial 'tiles' defined on many MODIS data products across three scenes of two tiles each. We informally call these three scenes 'California,' 'Brazil,' and 'Zimbabwe,' respectively, though the data is neither complete, nor contained within, each political boundary. We choose these scenes to illustrate the spatial heterogeneity of the data, which makes a 'global' indexing non-trivial. Each scene contains 2400×1200 time series objects. We index 2-year overlapping subsequences starting with the first complete year of data, 2001, and generate further subsequences by shifting at steps of a year (i.e., 2001:2003, 2002:2004,...,2008:2010, 2009:2011), yielding $28.8M$ time series segments of length 46 for each scene.

3.1.1 Querysets and Runtime Analysis

We present a brief runtime analysis comparing the base, proposed, and exhaustive linear-scan search methods. Presenting a large analysis of false dismissals is non-trivial because the exact search is executed with relatively large runtime, even on only $30M$ time series segments. We therefore cannot build a queryset of all subsequences generated on each scene (though our index could complete the approximate search in large but tractable time). We generate a simple test to show the overhead cost of the aggregation step in the proposed approximate range search. We execute 40 searches, ensuring $|R| > \alpha$ in

the proposed search method, meaning that resultset aggregation has occurred within each approximate search. The total cumulative runtime for the base and proposed methods is 3.47 seconds and 11.13 seconds, respectively; we have generated a single subsequence dataset to do an in-memory, single matrix-wise similarity computation in MATLAB, at 732.8 seconds, (averages are $0.08, 0.27$, and 18.3 seconds per search). This linear scan runtime is internally optimized in MATLAB; implementing the linear scan with a conventional loop yields 925 seconds average per search. In experimentation, the aggregation step for arbitrarily small α still yields queries on interactive time scales (e.g. < 1 second).

Due to the limitations of the exact search runtime, we generate three randomly sampled query subsets from each scene at a size of $10K$ and $20K$ two-year subsequences. We have verified that results are sufficiently consistent across repeated random samples. For brevity, when reporting results, these querysets will be referred to by scene letter and number. For example, $B1, Z2$ identify Brazil and Zimbabwe, respectively. For simplicity, the query source and searched scene will always match, so these querysets also identify the scene for each result.

3.2 Results

3.2.1 Bounding and Depth

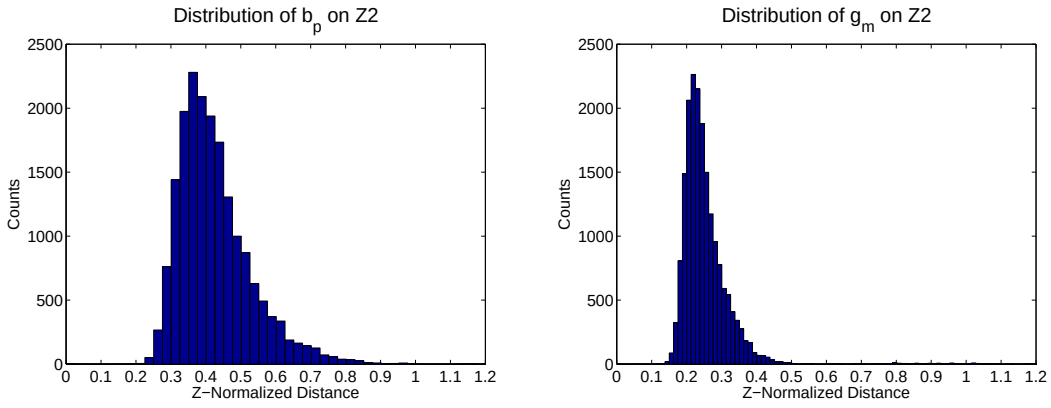

Figure 3.1: Distribution of pairwise distance bound b_p responses (Left), and pairwise median guidance g_m responses (Right) on $Z2$.

Figure 2.1 reports the distribution of b_p (Left) and g_m (Right) on $B2$. Comparatively, Figure 3.1 illustrates that searches on $Z2$ report much better (lower) b_p and g_m. The median of each distribution of g_m responses across queries $B2$ and $Z2$ are 0.4, and 0.24, respectively. These results are summarized in Table 3.1 (rows 1 and 7). Figure 3.2 shows that the depth distributions of $B2$ (left) and $Z2$ (right) differ dramatically. On $B2$, many traversals are only at a nominal depth (< 5). This means grouping often occurs on binary SAX representations in Brazil, yielding poor similarity. We can use this analysis to hypothesize on the relative heterogeneity or noise characteristics between scenes, or the effectiveness of pre-processing such as aggregation or time series smoothing. Whatever the cause, Brazil intuitively seems a 'harder' scene to search than Zimbabwe. This is consistent with our experience with data quality of tropical regions in vegetation index datasets.

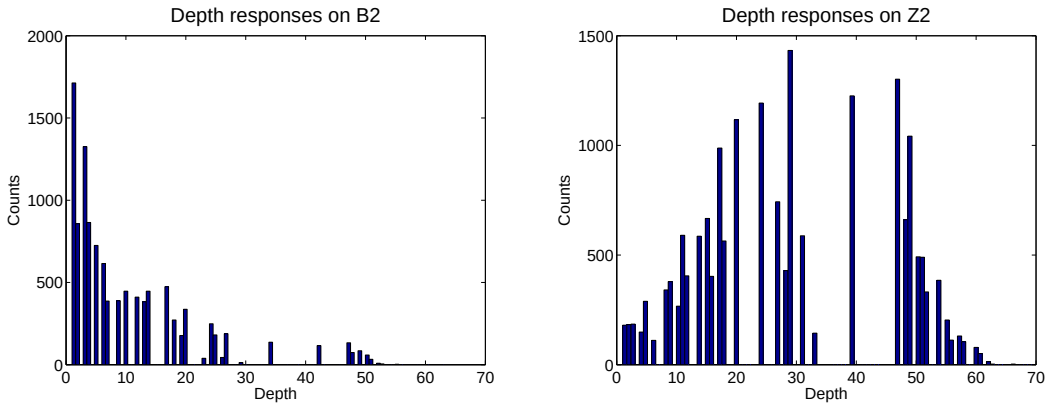

Figure 3.2: Distribution of depth responses for base search method (to leaf), on queryset $B2$ (Left), and $Z2$ (Right).

Figure 3.3 (Left) further shows this intuition by plotting the decreasing node sizes of base nodes (i.e. binary, $k = 2^1$) across each scene on logarithmic scales. The top-ranked node in Zimbabwe contains over $1M$ time series segments. This means that over $1M$ time series segments map to the same binary encoding. The top-100 largest nodes all contain more than $100K$ segments. Because of the structure of the tree, common time series segments which share a base encoding are more likely to be grouped with higher granularity after index construction. Using $\alpha = 20$, the number of time series segments in 'unsplit' and singleton nodes in Brazil is $4.12M$ in comparison to $247K$ in Zimbabwe.

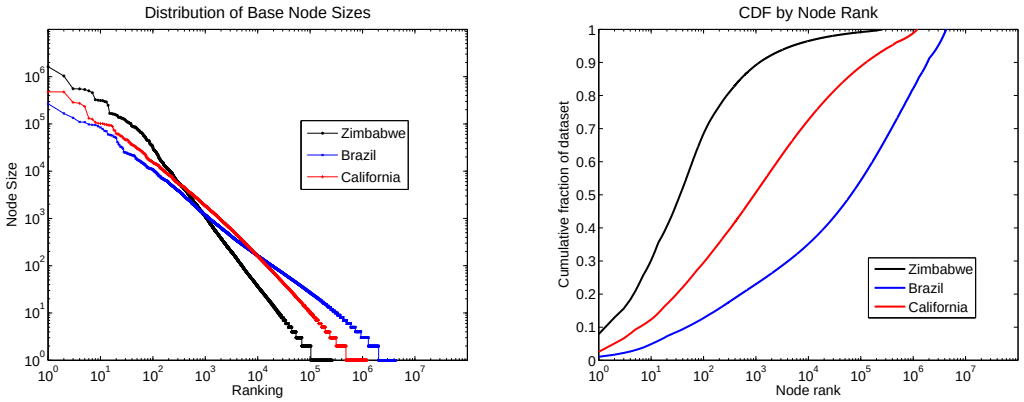

Figure 3.3: (Left) Comparison of decreasing node sizes across scenes on logarithmic scales. (Right) The cumulative fraction of data in the top-k largest nodes.

Figure 3.3 (Right) gives an alternate perspective of these distributions, plotting the cumulative fraction of the top k nodes to the complete dataset. For the top 1000 nodes, approximately 20%, 50% and 90% of data is contained within these nodes for Brazil, California, and Zimbabwe, respectively.

3.2.2 Rank depth test

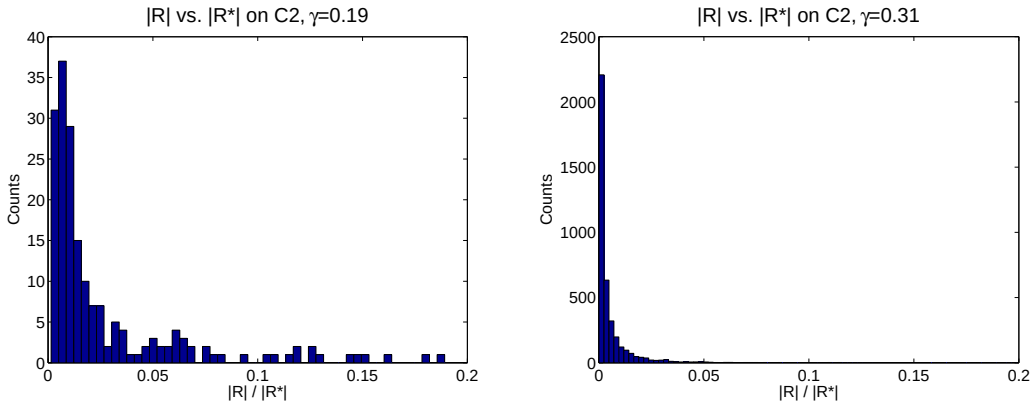

Figure 3.4: A comparison of the size of satisfied rank depth test results ($|R| > 50$ for specified γ) compared with the size of the exhaustive resultset, $|R^*|$, showing values of $\frac{|R|}{|R^*|}$ for $C2$, $\gamma = 0.19$ (Left) and $\gamma = 0.31$ (Right). The results in R^* not in R correspond to false dismissals.

We show that a traditional analysis of false dismissal rate is not suitable in our context because the upper bound is often not tight and not characteristic of the quality of relevant grouped sequences. If we extend an envelope of 0.4 around the query, the number of false dismissals is large. Figure 3.4 (Left) illustrates the relative size of the resultset $|R|$, compared to the size of the resultset from the linear scan $|R^*|$, reporting $\frac{|R|}{|R^*|}$. Satisfiability is given by $\gamma = 0.19$ with a resultset size $|R| > 50$. This criteria yield 191 hits on $C2$. Yet the false dismissals are massive, with the median quotient of 0.0119, resulting in false dismissals on the order of thousands. We find this order of false dismissals is typical across scenes. In our use-case we would not be interested in retrieving such a large search resultset, therefore, false dismissals are not a good measure of our method since we would be most interested in obtaining a lesser number of *highest quality* results.

The nature of grouping on exact SAX subsequence representation is a stronger constraint to be satisfied than a fixed distance threshold calculated cumulatively across the sequence. The cardinality promotion operation fixes a progressively smaller envelope around each observation of a time series pattern in order to be assigned a particular word w_k^i, at time step i. In contrast, a distance-to-query of 0.4 does not impose constraints on any particular distance per observation (i.e. $|Q^i - T^i| < \epsilon$, for $i = 1...|S|$), only that their sum is below the threshold: $\sum_i |Q^i - T^i| < \gamma$. The grouping by SAX representation implicitly imposes the constraint $Q^i \in w_k^i$, bounding the distance *at every observation.* This motivates the hypothesis that grouping by exact matching of SAX subsequences (with sufficient frequency) will yield, on average, higher quality results relative to the false dismissal set, which may satisfy the distance threshold but are not similar to the query at each time step.

We test this hypothesis by comparing the distance ranking of each result in R within the exhaustive resultset R^*. The rank depth test is defined on each resultset of sufficient size ($|R| > 50$) satisfying the given γ. Given each S in R, we return the percentile at which $d(Q, S)$ exists in the distance list of the exhaustive resultset R^* satisfying γ, $d(Q, R^*)$, where lower is better. This measure is invariant to the relative size of resultsets, however we note that this test is only suitable when $|R| << |R^*|$. When $|R| = |R^*|$, the rank depth is a permutation of the original rank ordering, with median 0.5. Figure 3.5 (Left) aggregates all true positive rank depth responses across 546

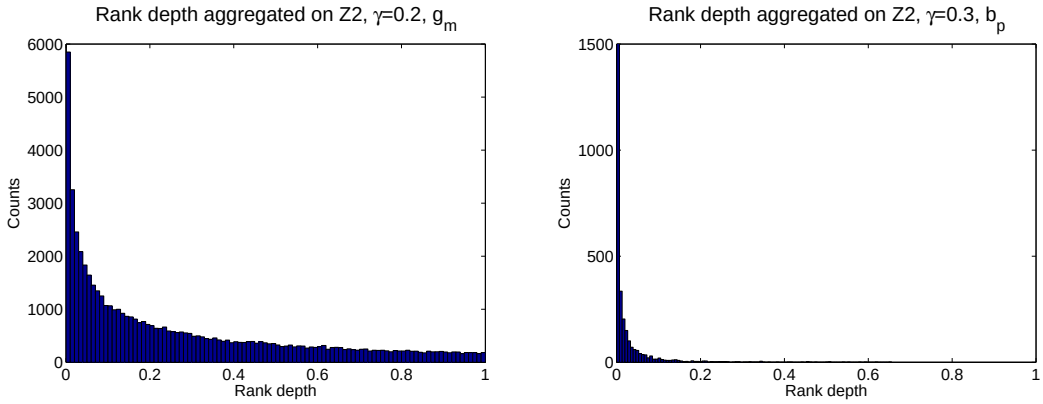

Figure 3.5: Rank depth test on Z2 showing aggregate search responses of using stopping condition $\gamma = 0.2$ for g_m (Left), and $\gamma = 0.3$ for b_p (Right). Each count corresponds to a time series subsequence object.

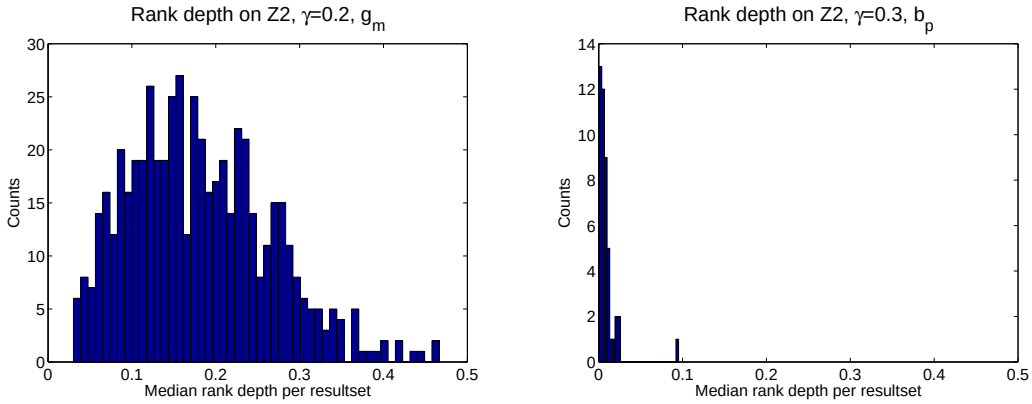

Figure 3.6: Frequency of medians of rank depth test on $Z2$. One count corresponds to a search resultset, (lower is better).

resultsets at $\gamma = 0.2$, using g_m. Figure 3.5 (Right) reports the same over 46 hits at $\gamma = 0.3$, using b_p. We see that the rank depth results using b_p are relatively better. This is surprising, because we expect b_p to be loose, while g_m by definition is set to the center of the pairwise distance distribution. Yet the bounded stopping condition allows for relatively fewer, higher quality hits.

There is a limitation to this analysis; perhaps in Figure 3.5 (Left) we observe the result of a small number of very large, high quality resultsets while the typical search returns poor results (i.e. there is a skew hidden in the aggregation of resultsets). Figure

3.6 shows the median rank depth per resultset. In the case that there is no favor to the approximate search compared to the exhaustive method, we should observe a distribution around 0.5 (a random sampling of rankings). We see that the *median* rank depth per resultset is 0.17. We interpret this to mean that relatively high quality results are returned at these γ thresholds; however, we note that few results are found in the top $|R|$ elements of R^*. Across all scenes, we observe that the median recall of top-$|R|$ results of R^* is under 20%. This can be explained by the low-recall problem setting given the strictness of the grouping criterion: only one dissimilar observation is required in order for a pair of sequences to have different SAX representations.

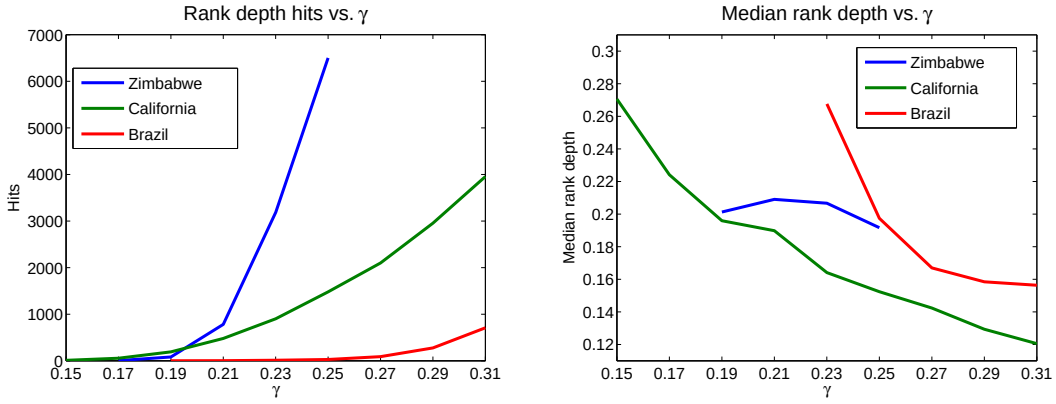

Figure 3.7: (Left) The number of hits for rank depth test while varying γ. (Right) The median rank depth across scenes with varying γ, and *hits* > 10 (to ensure a reliable median). The few plotted points for Z2 are due to the sharp rise in hits.

Comparison over γ values

Figure 3.7 shows the rank depth results across each scene as γ increases. Figure 3.7 (Left) shows the number of hits which satisfy the specified γ and $|R| > 50$. We see that Zimbabwe has over 15% of queries satisfied at $\gamma = 0.23$, while California has roughly the same number of hits at $\gamma = 0.29$. Figure 3.7 (Right) shows that California and Brazil increase in performance as γ increases and the quality of the exhaustive resultset is loosened. Therefore even queries at higher g_m can perform well *relative* to exhaustive queries at that range. Figure 3.4 (Right) shows the distribution of relative size of resultsets and false dismissals at an increased $\gamma = 0.31$. We see the quotient

of size difference has decreased from 0.0119 to 0.0020, yielding false dismissals on the order of tens of thousands. This demonstrates the effectiveness of the SAX constraint in producing high-quality results, while the distance threshold-based method degrades in relative quality quicker over increased γ.

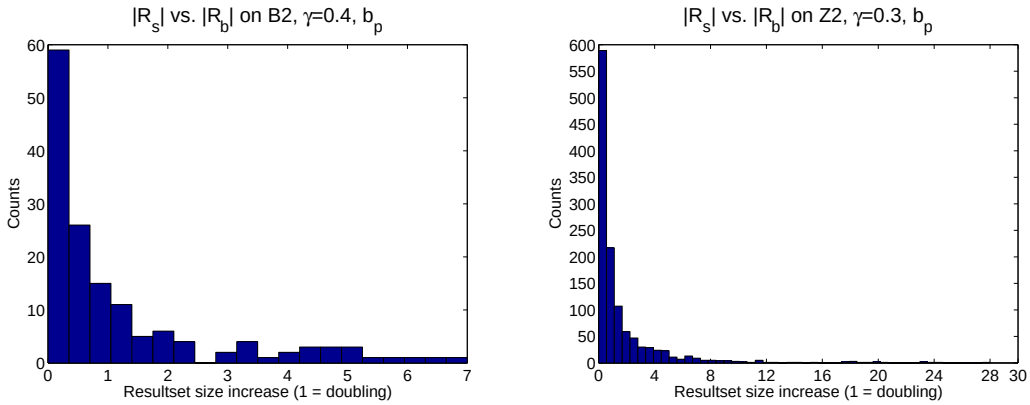

Figure 3.8: The increase in size from the base resultset R_b to proposed approximate search resultset R_s, measured as $\frac{|R_s|-|R_b|}{|R_b|}$ for stopping condition $\gamma = 0.4$, using b_p on $B2$ (Left), and with $\gamma = 0.3$, using b_p on $Z2$ (Right).

3.2.3 Resultset size increase

Figure 3.8 compares the size of the base approximate search resultset R_b, and the proposed approximate range query resultset R_s using $\frac{|R_s|-|R_b|}{|R_b|}$ with stopping threshold $\gamma = 0.4$ and $\gamma = 0.3$, on $B2$ (Left) and $Z2$ (Right). In this case, no false positives are within either resultset, and $R_b \subseteq R_s$. Therefore, the difference in size, $|R_s| - |R_b|$ constitutes false dismissals in R_b. The median increase is 0.55, 0.62, over 152 and 1233 hits, respectively. The relative increase using g_m is larger: 3.64, and 2.35, at $\gamma = 0.3$, $\gamma = 0.2$ respectively. Because the base search resultset size is bounded by α, and resultset sizes of the proposed search method are consistent for larger α at a fixed γ. Therefore, this measure necessarily increases as α decreases. Our previous measures: depth, b_p, g_m also necessarily improve on increased α, balancing the desired quality of results with given computational resources.

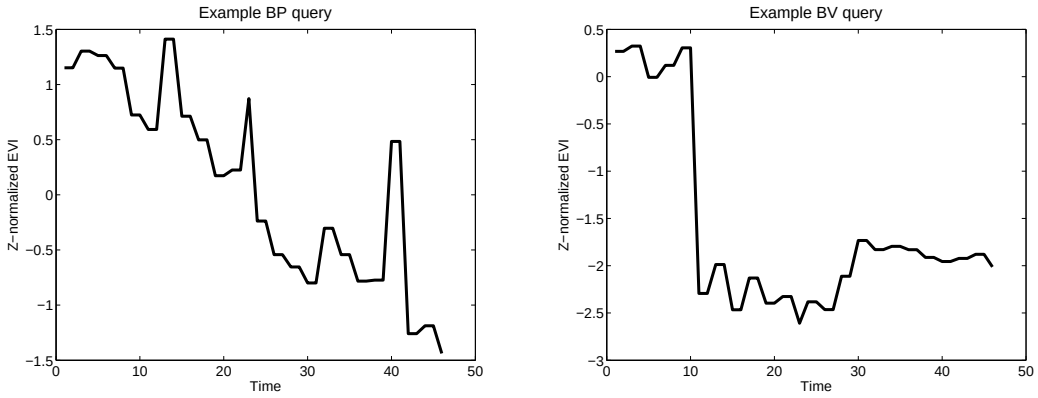

Figure 3.9: Example time series subsequences exhibiting and gradual change in temporal dynamics (Left), abrupt change in temporal dynamics (Right).

Searching for rare time series segments

In the above analysis, we used querysets of randomly selected subsequences from each scene. However, in many use cases, users have a specific rare class of subsequence to study that may not be represented in that sample. Previous work has focused on time series exhibiting a change in temporal dynamics [9, 16, 30] as shown in Figure 1.1 (red time series), and Figure 3.9. In the EVI dataset, these segments are rare because generally they require human intervention or natural events, such as fire or flood, which affect a relatively small area of earth. We construct a queryset for the persistent delta algorithm (PD) [9] (Figure 3.9 (Left)) and the Vegetation-Independent Delta algorithm (VID) [30] (Figure 3.9 (Right)). For PD, we choose the 5000 top-ranked time series objects according to total vegetation loss over a duration of between 2 and 3 years. For VID we take the top-ranked 0.5% of time series objects in Brazil and California, and top 0.2% in Zimbabwe (because changes are less prevalent in this scene). For each algorithm, we create a two-year segment starting from January where the first 'changed' observation occurred according to the algorithm. Though these methods are not flawless and incur their own false positives, on visual inspection each queryset is intuitively skewed to relevant, rare subsequences corresponding to land-cover changes.

Table 3.1 shows that in *every* scene, the median g_m of search responses increases, and median depth decreases for the PD and VID querysets. We also see that relatively few queries satisfy criteria for the rank depth test. Above, we saw that the bounding

Table 3.1: Summary of evaluations across different scenes. The suffixes 'V', 'P' refer to querysets assembled from change detection algorithms in [9, 30]. '-' entries denote no satisfying search of the test on the dataset.

Descriptor			Distance & depth Medians			Rank depth - Medians				Resultset relative size	
Row	Data	γ	Hits	g_m	Depth	Hits	Min	Median	Max	Hits	Median
1	B2	0.20	11162	0.40	6	2	0.18	0.25	0.33	30	0.52
2	BP	0.20	2687	0.41	5	-	-	-	-	1	1.72
3	BV	0.20	8585	0.46	5	-	-	-	-	-	-
4	C2	0.20	16675	0.32	14	311	0.03	0.19	0.53	1167	3.21
5	CP	0.20	3232	0.39	6	118	0.05	0.18	0.49	332	4.74
6	CV	0.20	10155	0.35	9	66	0.08	0.35	0.64	1084	14.20
7	Z2	0.20	18511	0.24	27	546	0.03	0.17	0.47	2697	2.35
8	ZP	0.20	3596	0.31	14	9	0.07	0.16	0.32	91	1.88
9	ZV	0.20	2679	0.36	5	-	-	-	-	78	0.34

Table 3.2: Evaluation results on Brazil scene over increasing γ threshold

Descriptor			Distance & depth Medians			Rank depth - Medians				Resultset relative size	
Row	Data	γ	Hits	g_m	Depth	Hits	Min	Median	Max	Hits	Median
1	B2	0.21	11162	0.40	6	3	0.14	0.36	0.38	40	0.78
2	B2	0.23	11162	0.40	6	12	0.09	0.27	0.44	100	1.11
3	B2	0.25	11162	0.40	6	28	0.03	0.20	0.36	228	1.35
4	B2	0.27	11162	0.40	6	89	0.02	0.17	0.45	514	1.83
5	B2	0.29	11162	0.40	6	277	0.01	0.16	0.49	1056	2.90

Table 3.3: Evaluation results on California scene over increasing γ threshold

Descriptor			Distance & depth Medians			Rank depth - Medians				Resultset relative size	
Row	Data	γ	Hits	g_m	Depth	Hits	Min	Median	Max	Hits	Median
1	C2	0.21	16675	0.32	14	480	0.00	0.19	0.69	1552	3.61
2	C2	0.23	16675	0.32	14	902	0.00	0.16	0.71	2450	4.70
3	C2	0.25	16675	0.32	14	1479	0.01	0.15	0.69	3615	5.45
4	C2	0.27	16675	0.32	14	2101	0.00	0.14	0.64	4892	5.70
5	C2	0.29	16675	0.32	14	2955	0.00	0.13	0.69	6243	6.93

distribution of Zimbabwe tended lower than Brazil significantly. But comparing the ZV queryset with $Z2$ (rows 7, 9), we see that it is now more similar to $B2$ (row 1) in terms of median g_m and depth. This demonstrates scope for work in indexing for rare classes of interest in heterogeneous datasets.

Table 3.4: Evaluation results on Zimbabwe scene over increasing γ threshold

Descriptor			Distance & depth Medians			Rank depth - Medians				Resultset relative size	
Row	Data	γ	Hits	g_m	Depth	Hits	Min	Median	Max	Hits	Median
1	Z2	0.21	18511	0.24	27	782	0.03	0.21	0.49	4320	3.50
2	Z2	0.23	18511	0.24	27	3187	0.02	0.21	0.59	7917	7.61
3	Z2	0.25	18511	0.24	27	6503	0.01	0.19	0.66	11044	13.80

Chapter 4

Anytime Framework: Methods

In Chapter 3, we demonstrated that a single approximate search traversal performs poorly on querysets of rare time series subsequences, relative to a randomly selected queryset. This is because rare sequences are grouped higher in the index tree structure, on coarser approximations. We also showed that an approximate query of this nature can be executed in very little time relative to an exact search (0.1 seconds, compared with 18 seconds).

This large difference in runtime, and often poor quality of results of a single traversal is a natural motivation for an anytime framework for executing similarity queries. An anytime framework describes a class of methods which allow the user to arbitrarily halt execution. While not halted by the user, this solution monotonically improves in quality, converging to the exact solution in some tractable time. Xu et al. [38] proposes a method for anytime k-nearest neighbor search and in Chapter 5 we adapt portions of the authors' evaluation methodology for an anytime range query on the proposed data structure.

The presented data structure has an intuitive anytime interpretation. Given the set of base nodes, and a lower-bounding pruning strategy using the distance-to-query, for each base node, an anytime method calculates the bounded distance and traverses into the node if necessary. If the node is a leaf and has not been pruned, the distance computation is performed on the data and satisfying results are included in the resultset. At any time we can terminate and return the current resultset.

Shieh and Keogh [37] presents a lower-bounding between a time series and the SAX

encoding representing the time series segments in a node. We simplify this formulation, and provide its interpretation under the mean L_1 norm. Let $B_u(w^i), B_l(w^i)$ be the upper and lower discretization bounds on SAX word w^i (i.e. the maximum and minimum possible value mapped to w^i). The lower bound of distance between time series T, and words $w^{1...|S|}$ is $Mean(LB(T, w))$ where:

$$LB(T, w^i) = \begin{cases} 0 & B_l(w^i) \leq T^i < B_u(w^i) \\ Min(|B_u(w^i) - T^i|, |B_l(w^i) - T^i|) & \text{otherwise} \end{cases}$$

We define the anytime exact range query on the search index I as:

Algorithm 2: Anytime Range Search

Input: Query time series Q, Base node keys B, similarity threshold γ, runtime limit t, time series similarity index structure I

Output: Resultset of time series data R

```
1  nodeStack = Stack();
2  while i ≤ |B| & runtime < t do
3  |  nodeStack.push(B(i));
4  |  while ¬nodeStack.empty() & runtime < t do
5  |  |  N = I.get(nodeStack.pop());
6  |  |  if N.isleaf then
7  |  |  |  dists = D(Q, N.data) ;
8  |  |  |  R.concat(N.data[dists ≤ γ]);
9  |  |  else
10 |  |  |  d = Mean(LB(Q, N.children(1)));
11 |  |  |  if d ≤ γ then nodeStack.push(N.children(1));
12 |  |  |  d = Mean(LB(Q, N.children(2)));
13 |  |  |  if d ≤ γ then nodeStack.push(N.children(2));
14 |  |  end
15 |  end
16 |  i = i + 1;
17 end
```

Algorithm 2 iterates over base node keys until completion or allotted runtime is expended (line 2; timekeeping details hidden for brevity). The current node key is pushed onto the stack (line 2). While the stack is not empty and runtime not exhausted (line 4), the method pops a key from the stack and retrieves the node from the index

(line 5). If the node is a leaf, the method computes the distance and includes the satisfying results (lines 6-8), otherwise it computes the lower bounded distance between the query and the encoding associated with the node's children keys, pushing these keys onto the stack if they are not above the desired threshold γ (lines 10-13). Note that the base node keys have an implicit ordering; this allow for varying *search strategies* using different orderings so that the method makes best use of its allotted time. One practical strategy we implement is to order the nodes in decreasing size (as shown in Figure 3.3). This strategy aims to maximize search *coverage* by pruning the largest base nodes first.

The above method is an *exact* range query because $Min(|B_u(w^i) - T^i|, |B_l(w^i) - T^i|)$ monotonically increases as the cardinality of any w^i increases. This can be thought of as the minimal bound 'moving away' from the observation T^i as w^i is bisected. So, if $LB(T, w) > \gamma$, then after any promotion sequence producing w^*, $LB(T, w^*) \geq LB(T, w) > \gamma$. Therefore, with a sufficient runtime limit t, the method will compute the distance against all time series objects which can possibly be at a distance of γ. The number of distance computations required is not straightforward, and depends on the prevalence of the query time series, and the quality of the leaf node groupings.

Due to these characteristics, we hypothesize that in contrast to the approximate range query presented in Chapters 2 and 3, the anytime query has preference to novel time series sequences at lesser depth in the tree. Allotted a fixed time, the search method prunes much of the tree at the base level for novel query sequences, and finds matches at very shallow depth. In contrast, common query sequences spend time traversing deep into the tree, maintaining the stack, and searching numerous candidate nodes. We evaluate this hypothesis below.

Chapter 5

Anytime Framework: Performance and Evaluation

5.1 Evaluation Setup

An anytime query will generally converge to the exact query solution after some runtime, and this runtime may be very large. We observe that due to the necessary implementation details of tree traversals in our chosen development environment (MATLAB), an anytime query can execute for very long indeed (> 200 seconds), relative to an optimized single-pass exhaustive search (≈ 18 seconds) on $28.8M$ subsequences. Even the latter runtime limits the scope of quantitative evaluation of index performance. Running the anytime query to completion and reporting the final runtime is nearly intractable for querysets large enough to give characteristic insights about the search index and data scene.

We therefore choose to 'interrupt' the anytime query after some runtime of execution. Since the index should compete against the best our platform can provide (otherwise, we can opt for the optimized linear scan), we limit the allotted runtime under $t = 18$ seconds, even though this is a very aggressive constraint for our implementation. We report performance at $t = 5$ and $t = 10$ seconds by taking the first 500 queries of each of the rare subsequence querysets formed by the change algorithm, and random $20K$ querysets, both presented in Chapter 3. For simplicity, we refer to these subsets by the same identifiers, (e.g: $C2$, CV). Because few queries complete in this allotted time,

these are approximate queries as described in Chapter 2. Below, we propose measuring performance using three evaluations: (1) coverage, (2) minimum (nearest-neighbor) and median distances comparison of R with R^*, and (3) rank depth test.

5.2 Results

5.2.1 Coverage

We are interested in measuring the 'coverage' of a query. For a given query, *coverage* measures the proportion of the dataset which the query is able to 'see' over the allotted runtime, either by pruning the segments in the sub-tree associated with a node (see: Algorithm 2, lines 10-13), or by comparing against the data in the node (lines 7-8). Figure 5.1 (Left) demonstrates that for a randomly selected queryset, little of the dataset is examined after $t = 10$ seconds of execution, while Figure 5.1 (Right) shows that rare query segments see around 50% of the data in the same time. Due to the ordering of nodes by decreasing size, common queries tend to traverse deep into a large ('early') node in this list, while rare query segments avoid traversing into these nodes by pruning operations, and thus accumulate a relatively large 'coverage' measure. Figure 3.3 (Right) shows that for the California scene, 50% of the data is contained within the top 900 nodes. This means that on average, the query scans 900 base nodes by the end of execution (or $< 0.1\%$ of all nodes). Table 5.1 shows that the CV queryset has much greater coverage than $C2$, and summarizes most results in this chapter.

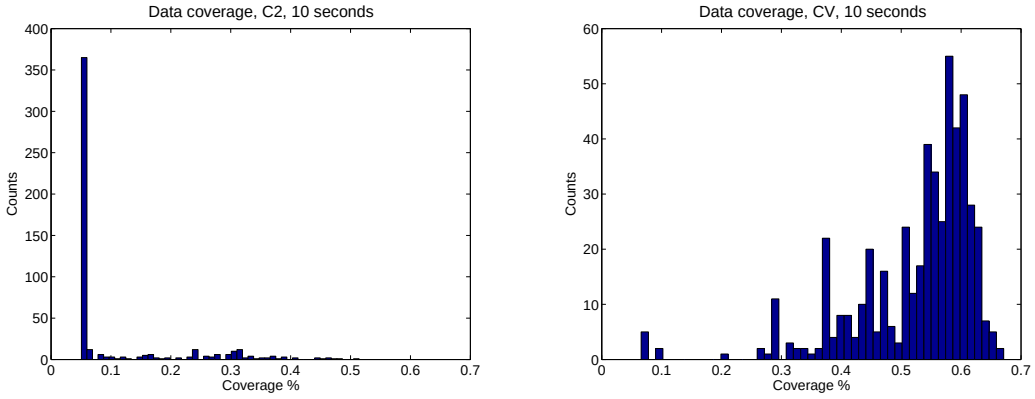

Figure 5.1: Distribution of coverage percentage for (Left) C2, and (Right) CV.

We also consider the number of distance calculations against node sequences to evaluate the performance of the index. This value is related to the density of the dataset, and the quality of the groupings formed in the index building phase. If a subsequence is very common, or the similarity threshold γ is high, few nodes can be pruned. When the distance calculation is performed on these numerous nodes, the bounding may be poor, causing many 'false positive' distance calculations where $d(Q, N.data) > \gamma$. Table 5.1 shows that for Brazil and California, the random querysets have less comparisons than the rare queryset at the same runtime allowance. Looking at the timing profile closer, we see that since the random querysets tend to traverse into deeper subtrees, the query spends more time maintaining the stack. Therefore, fewer computations are performed even though many nodes may be similar within these large subtrees.

Figure 5.2 (Left) shows the number of hits, $|R|$ on CV. Figure 5.2 (Right) reports the hits on the exhaustive linear scan. The threshold, $\gamma = 0.25$, provides a challenge because the exact resultset size is still small, with a mean resultset size of 6778 and 314 for $C2$ and CV, respectively. On this scene, there are no empty exhaustive resultsets R^* at $\gamma = 0.25$, however nearly half the resultsets on CV are empty in the allotted time $t = 10$, and on $C2$, only 7% are non-empty on the same allotment. This demonstrates the difficulty of overcoming the internal matrixwise-scan optimizations available in the development environment. Recall that a naive approach for exhaustive scan, which is necessarily more similar in implementation to the anytime query method above, returns results in approximately 925 seconds per query.

5.2.2 Nearest-neighbor and median distance comparison

Figure 5.3 (Left) shows the distribution of minimum distance per search resultset, comparing each exhaustive resultset R^* (blue) with the queried resultset R (red). Specifically, we show $Min(d(Q_i, R_i))$ for resultsets $i = 1...500$. The blue distribution corresponds to the distance to the actual 1-NN for each query. Similarly, Figure 5.3 (Right) reports the *median* distance-to-query for each resultset. We summarize these distributions by comparing the distance between their medians. Table 5.1 reports this as the 'ΔDistances', which shows that queries across all scenes are able to better retrieve results at the median of the exhaustive distribution than the actual 1-NN.

This result agrees with the intuition of a low-recall scenario where the query is more

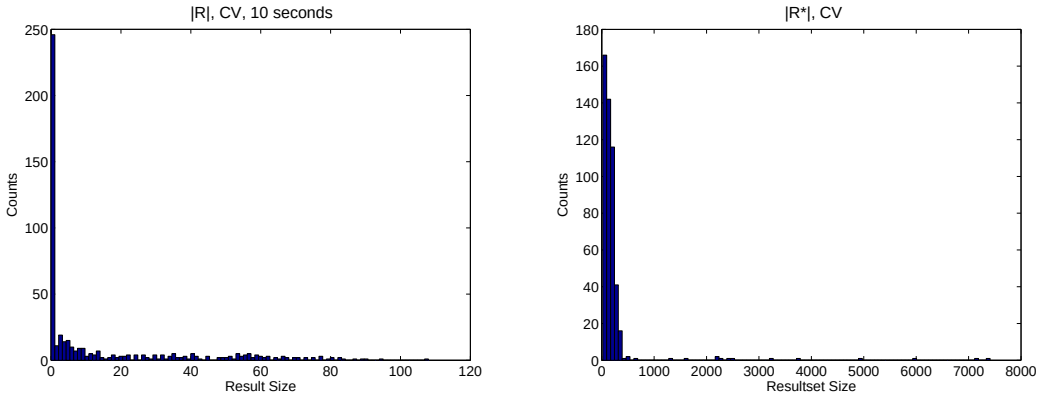

Figure 5.2: Distribution of size of CV resultsets (Left) vs. exhaustive resultsets R^* (Right).

likely to retrieve results near the center of the distribution. Also, the ordering of nodes by decreasing size suggests that results are unordered relative to distance-to-query. So in contrast to the approximate search evaluation in Chapter 3, we expect no strong skew to the quality of results as runtime increases. Different base-ordering heuristics such as in increasing distance-to-encoding, or combining the single approximate traversal with the anytime framework may introduce a stronger skew to the anytime framework. This could yield a setting of diminishing returns, meaning that result quality would improve to lesser degrees for further expended runtime, which would yield a practical stopping condition where marginal improvement falls below a given threshold.

5.2.3 Rank depth

Figure 5.4 shows the median rank depth on the BV queryset. As in Chapter 3, for each resultset R, a point in this distribution corresponds to the median rank depth across all results in R, where $|R| \geq 25$. The 'hits' reports this number of resultsets of sufficient size. In Table 5.1, we clearly see that for rows with suitably large rank depth test hits, the rank depth is near 0.5, suggesting a random selection amongst the rank ordering of the exhaustive resultset.

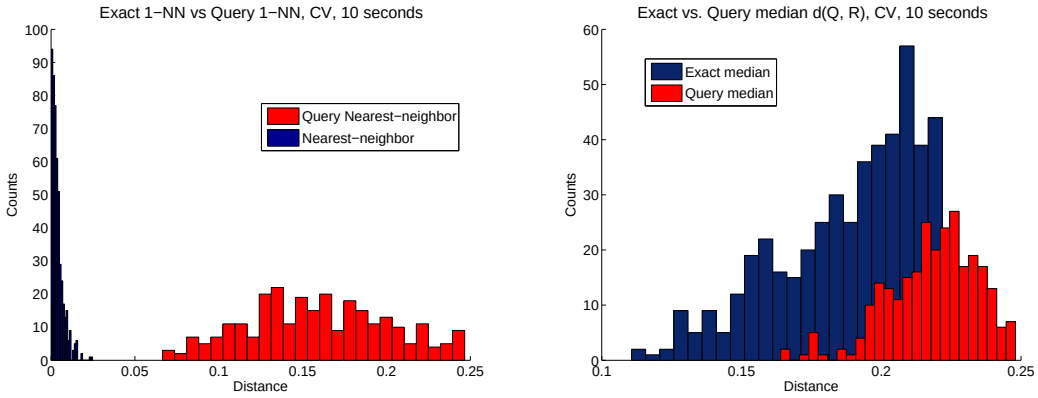

Figure 5.3: Comparison between exhaustive (Blue) and query (Red) for minimum (Left) and median (Right) distance responses $d(Q, R)$ under range $\gamma = 0.25$. Each count corresponds a resultset.

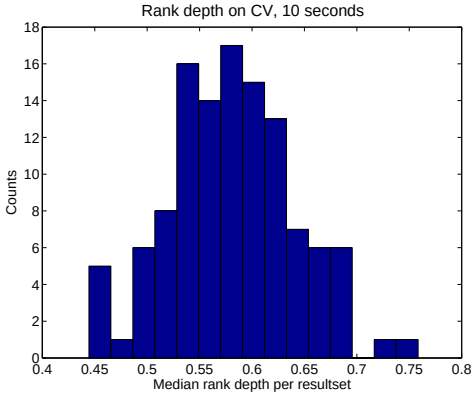

Figure 5.4: Median rank depth per resultset on BV.

5.2.4 Analysis of scenes

The Brazil scene is characterized by a large number of smaller nodes, where many similar encodings can occur due to noise shows very unintuitive performance in Table 5.1. Row 2 shows that B2 has even larger coverage than BV. Coverage of 0.20 corresponds to approximately 1000 nodes processed (by Figure 3.3), meaning that each queryset on the Brazil scene likely stays very shallow in the traversal. Row 8 demonstrates that the random queryset on the California scene traverses deep into large, 'early' nodes. Rows 5 and 6 show that in Brazil, the BP queryset perform best in all measures. Examining

closer we see this is not due to outliers in the results. Our evaluation strategy effectively penalizes for failed queries. Excluding 372 empty resultsets on Row 6, the median hits and recall is 26 and 0.2835, respectively.

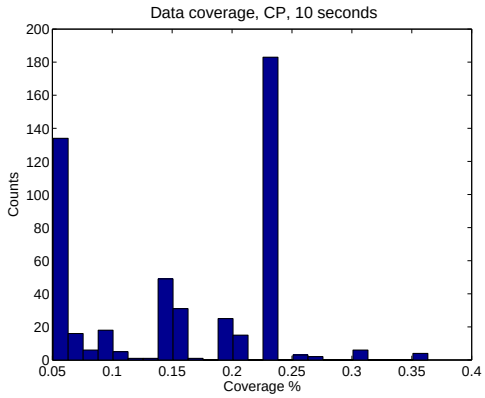

Figure 5.5: Coverage distribution for queryset CP. Each count corresponds to a query.

The California scene has been studied in previous work with respect to land cover change related to fire events, due to access of high-quality validation data generated and made available by the state of California. The coverage results for this scene are predictable: Row 8 shows recall is generally small, meaning that traversal enters large, 'early' nodes and returns relatively few results (0.00 mean recall, with only 17 non-empty resultsets within 10 seconds). In contrast, Row 10 shows CV queries scan much further into the ordered node-list, and return relatively more results. These queries also best approximate the ΔDistances minimum. This is likely because a sequence exhibiting an abrupt decrease has a distinct coding even at the base cardinality such that few dissimilar segments are mapped within these nodes. Furthermore, temporal events such as fire occuring in different time steps yield different encodings, meaning that only fire events within approximately the same date are generally grouped. Therefore, if any results are returned, they are generally of high similarity.

Figure 5.5 shows the coverage results for CP. We observe that there is some bi-modal behavior where many queries cover approximately 0.05 of the dataset, or 0.23. Observing query traversals we informally find that for 'expected' gradually decreasing segments as shown in Figure 3.9 (Left) traverse into some large nodes, but more shallowly than the $C2$ queryset. In contrast, on the CV queryset, these traversals tend to be pruned at

the base.

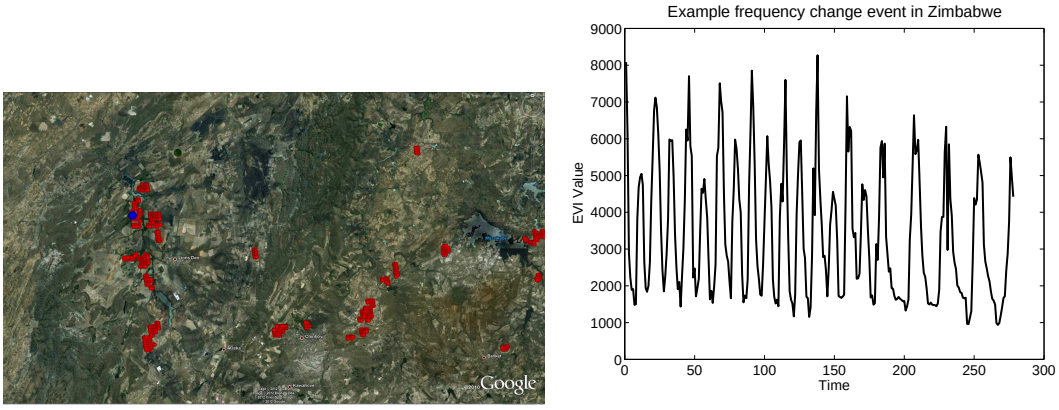

Figure 5.6: (Left) Example land cover and land cover changes prevalent in Zimbabwe scene. (Right) Time series at blue point exhibiting a change after 2006.

The Zimbabwe scene was chosen for study because it has informally been observed to contain few land cover types, and due to political circumstances in the state of Zimbabwe we observe many land cover changes relating to agricultural deintensification which may on suitable scale for indexing. Figure 5.6 (Left) shows an example of the land cover in Zimbabwe, where approximately 90% of data is within the largest 1000 base nodes. Furthermore, on investigation many of these largest nodes contain similar base representations. Rows 13 and 14 show that the mean size of exhaustive resultsets is 82,842, and a maximum exhaustive resultset size of $1.34M$. We see that Zimbabwe uniformly has poor search coverage. This is expected. Even though agricultural deintensification subsequences are represented in the ZP queryset, the nature of change (double-to-single annual peaks, as shown in Figure 5.6 (Right)), means pruning is not achieved as easily as on CV. On inspection, the ZV queryset does not exhibit clear abrupt changes as shown in Figure 3.9 (Right), so the characterization of patterns within this queryset is not simple. However, Row 16 shows that the mean size of exhaustive resultsets is very small relative to $Z2$, so the sampling is not simply random.

Table 5.1: Summary of evaluations across different scenes. The suffixes 'V', 'P' refer to querysets assembled from change detection algorithms in [9, 30]. All 'coverage' results aggregated by mean; ΔDistances and rank depth aggregated by median.

Descriptor			Coverage (All reported in mean)					ΔDistances		Rank depth	
Row	Data	Time	% Coverage	Total Hits	Hits	Compared	Recall	Min	Median	Hits	Median
1	B2	5	0.20	533	1.19	11364	0.00	0.20	0.00	8	0.50
2	B2	10	0.24	533	3.58	22474	0.00	0.20	0.00	13	0.51
3	BV	5	0.18	31	0.16	8855	0.00	0.22	0.01	1	0.60
4	BV	10	0.19	31	0.45	18057	0.01	0.19	0.01	5	0.67
5	BP	5	0.28	81	14.31	6482	0.05	0.19	0.00	41	0.55
6	BP	10	0.28	81	26.37	12979	0.09	0.18	0.00	66	0.54
7	C2	5	0.05	6778	4.59	6612	0.00	0.21	0.01	4	0.46
8	C2	10	0.05	6778	8.35	13980	0.00	0.23	0.01	5	0.46
9	CV	5	0.52	314	7.48	8040	0.04	0.16	0.02	63	0.57
10	CV	10	0.56	314	14.74	17425	0.08	0.16	0.02	116	0.58
11	CP	5	0.16	1980	0.08	7510	0.00	0.22	0.08	0	-
12	CP	10	0.20	1980	0.38	16210	0.00	0.23	0.07	2	0.68
13	Z2	5	0.05	82842	46.35	7985	0.00	0.20	0.00	89	0.54
14	Z2	10	0.05	82842	100.34	15313	0.00	0.20	0.00	118	0.56
15	ZV	5	0.05	219	0.00	6124	0.00	0.24	0.01	0	-
16	ZV	10	0.05	219	0.00	11917	0.00	0.24	0.01	0	-
17	ZP	5	0.05	386	0.28	6908	0.00	0.21	0.03	0	-
18	ZP	10	0.05	386	0.48	13649	0.01	0.21	0.04	3	0.61

Chapter 6

Conclusion and Future Work

In this work, we have shown an effective extension to previous work on time series indexing for similarity search. Using this extension, we can reduce sensitivity to the splitting parameter α at a greater storage and computational cost (according to the resources of the user). Using strategies of bounding on the intra-node pairwise distance, we proposed an approximate range query method which had the flexibility of fast queries that can be easily refined or loosened based on a stopping condition γ.

This work provides an exploratory analysis formulating several future directions specifically addressing the challenges found in earth science datasets, shared in heterogenous spatiotemporal datasets more broadly. EVI and other earth science datasets are characterized by a variety of largely stable or gradually changing, periodic temporal patterns and a minority class of abrupt change-event sequences. We have deeply explored these challenges in an effort to quickly retrieve 'interesting' subsequences, and shown the contrasting performance of these querysets under the approximate search.

6.1 Data Sparsity

Because exact matching of SAX encodings is leveraged in the data structure, data sparsity limits the index performance, especially in grouping sufficiently many similar, rare subsequences. Figure 3.3 (Left) illustrates that a large fraction of data resides in small nodes (e.g. $|N| \leq 10$) at a base encoding. Incorporating domain knowledge to reduce dimensionality (like our simple adaptation of PAA for periodic time series)

and choosing promotion time steps in a more informed way might practically stretch performance in this instance. More generally, aggregation and smoothing techniques are not well understood given the objective of preserving time series characteristics of interest such as sudden change events [12]. Future work would develop and evaluate specialized aggregation techniques for the preservation of these characteristics.

We attempted to index a new scene in Wyoming as a target to study widespread forest degradation from pine beetle infestation, consisting of $1.44M$ time series objects and $11.52M$ *3-year* subsequences (note we previously evaluated on 2-year subsequences). Figure 6.1 shows the distribution of cumulative top-k node sizes as a fraction of all subsequences (as in Figure 3.3) between Wyoming and Brazil. Wyoming is plotted using 2, 3, and 4-year subsequences. For 2-year subsequences, we see it has a higher fraction of large nodes than Brazil. Wyoming contains less than half the subsequences of Brazil, but indexing 3-year subsequences, Wyoming has more nodes, meaning that on average, a larger proportion of time series segments are contained in smaller nodes. Indexing 4-year segments, the curve flattens even more, approaching the plot of all singleton nodes (in green). We intended to use this grouping to build a null model for the purpose of other land-cover change detection techniques, which look for stability (tightly grouped, 'unchanged' subsequences) three years prior to the scoring time step, but the data sparsity of three-year segments (46 compared to 69 time steps) means that few reliable groupings are returned either under the approximate or anytime framework.

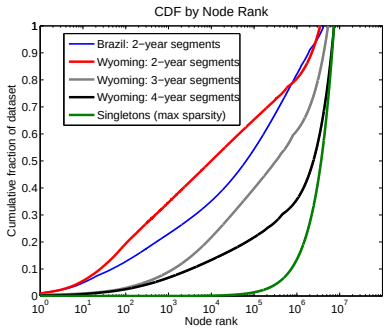

Figure 6.1: The cumulative fraction of data in top-k ranked nodes comparing Brazil and Wyoming.

The authors, Shieh and Keogh [37] likely did not experience these limitations of sparsity because they evaluate on relatively simple random-walk data which does not exhibit

heterogeneity from numerous underlying models, or temporal events such as land-cover change. For exact search, the authors' nearest-neighbor query is an intrinsically easier query to compute. Using a best-so-far, the 1-NN query on the data structure progressively constrains the threshold γ through the traversal, meaning that the runtime of the method is affected by how soon in execution a fairly good best-so-far match is found. This means the pruning rate accelerates as γ is reduced to tighter best-so-far. In contrast, the range query visits many 'similar' nodes without progressively tighter constraints. Therefore a scene yielding more, shallower nodes slows the anytime range query greater than the 1-NN query.

6.2 Search Index Structures

Time series exhibiting a change in temporal dynamics (as shown in Figure 1.1) are rarely grouped with suitable granularity. In the EVI dataset, vegetation dynamics tend to be stable in absence of relatively rare natural events such as fires and floods, or human interventions such as conversion to cropland, while the indexing method implicitly assumes all classes of interest will be of sufficient frequency. The size of data required to sufficiently index rare classes becomes intractable. Future work would combine the symbolization indexing strategy with grouping criteria that have some flexibility to mismatches and warping. Future work will also further develop evaluation methodologies for low-recall problem settings, on heterogeneous datasets with potentially many query classes.

There is great scope for future work in fast, approximate query methods. Though exact nearest neighbor search, even under DTW, has been shown to scale to previously unachievable trillions of data points [34], non-indexed query methods have the strict requirement of accessing each time series object at least once, even to apply a fast pruning operation (such as $O(1)$) and are more applicable to more constrained k-NN queries, while approximate, indexed query methods have the advantage of accessing the structured auxiliary data generated over the dataset, then accessing only a small portion of the time series objects in the data space. This advantage is further highlighted in a database environment where large, sequential reads are not preferred. Future work should focus on scaling-up fast approximate search even further.

References

[1] NASA Earth Observing System. `http://eospso.gsfc.nasa.gov`.

[2] Land Processes Distributed Active Archive Center. `http://lpdaac.usgs.gov`.

[3] J. Aach and G. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495, June 2001.

[4] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *Conference on Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[5] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science*, pages 459 –468, oct. 2006.

[6] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, pages 359–370, 1994.

[7] E. Caiani, A. Porta, G. Baselli, M. Turiel, S. Muzzupappa, F. Pieruzzi, C. Crema, A. Malliani, and S. Cerutti. Warped-average template technique to track on a cycle-by-cycle basis the cardiac filling phases on left ventricular volume. In *Computers in Cardiology*, volume 25, pages 73–76. IEEE, 1998.

[8] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh. iSAX 2.0: Indexing and Mining One Billion Time Series. *IEEE International Conference on Data Mining*, pages 58–67, 2010.

[9] Y. Chamber, A. Garg, V. Mithal, I. Brugere, V. Kumar, M. Lau, M. Steinbach, C. Potter, S. A. Klooster, V. Krishna, and S. Boriah. A novel time series based approach to detect gradual vegetation changes in forests. In *Conference on Intelligent Data Understanding*, pages 248–262, 2011.

[10] K. Chan and A. Fu. Efficient time series matching by wavelets. In *ICDE 1999*, pages 126–133. IEEE, 1999.

[11] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. Yu. Indexable PLA for efficient similarity search. In *VLDB 2007*, pages 435–446, 2007.

[12] X. Chen, V. Mithal, S. R. Vangala, I. Brugere, S. Boriah, and V. Kumar. A study of time series noise reduction techniques in the context of land cover change detection. *Department of Computer Science and Engineering - Technical Reports*, (TR 11-016), August 2011.

[13] S. Chu, E. Keogh, D. Hart, M. Pazzani, and Others. Iterative deepening dynamic time warping for time series. In *SDM 2002*, pages 195–212, 2002.

[14] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *VLDB 2008*, 1(2):1542–1552, August 2008.

[15] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *ACM SIGMOD Record*, 23(2):419–429, 1994.

[16] A. Garg, L. Manikonda, S. Kumar, V. Krishna, S. Boriah, M. Steinbach, D. Toshnival, V. Kumar, C. Potter, and S. A. Klooster. A model-free time series segmentation approach for land cover change detection. In *Conference on Intelligent Data Understanding*, pages 144–158, 2011.

[17] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB 1999*, pages 518–529, 1999.

[18] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):67–72, February 1975.

[19] T. Kahveci and A. K. Singh. Efficient index structures for string databases. In *VLDB*, pages 351–360, 2001.

[20] K. V. R. Kanth, D. Agrawal, and A. K. Singh. Dimensionality reduction for similarity searching in dynamic databases. In L. M. Haas and A. Tiwary, editors, *ACM SIGMOD International Conference on Management of Data*, pages 166–176. ACM Press, 1998.

[21] A. Kassidas and J. MacGregor. Synchronization of batch trajectories using dynamic time warping. *AIChE Journal*, 44(4):864–875, Apr. 1998.

[22] E. Keogh and M. Pazzani. A simple dimensionality reduction technique for fast similarity search in large time series databases. *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pages 122–133, 2000.

[23] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, May 2004.

[24] E. Keogh, K. Chakrabarti, and M. Pazzani. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, Aug. 2001.

[25] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, 30(2):151–162, 2001.

[26] S. Kim, S. Park, and W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *International Conference on Data Engineering 2001*, pages 607–614. IEEE, 2001.

[27] S. Lhermitte, J. Verbesselt, I. Jonckheere, K. Nackaerts, J. A. van Aardt, W. W. Verstraeten, and P. Coppin. Hierarchical image segmentation based on similarity of NDVI time series. *Remote Sensing of Environment*, 112(2):506 – 521, 2008.

[28] S. Lhermitte, J. Verbesselt, W. Verstraeten, and P. Coppin. A comparison of time series similarity measures for classification and change detection of ecosystem dynamics. *Remote Sensing of Environment*, 115(12):3129 – 3152, 2011.

[29] J. Lin, E. J. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007.

[30] V. Mithal, A. Garg, I. Brugere, S. Boriah, V. Kumar, M. Steinbach, C. Potter, and S. A. Klooster. Incorporating natural variation into time series-based land cover change detection. In *Conference on Intelligent Data Understanding*, pages 45–59, 2011.

[31] V. Niennattrakul, P. Ruengronghirunya, and C. A. Ratanamahatana. Exact indexing for massive time series databases under time warping distance. *Data Mining and Knowledge Discovery*, 21(3):509–541, Nov. 2010.

[32] M. Patella and P. Ciaccia. Approximate similarity search: A multi-faceted problem. *Journal of Discrete Algorithms*, 7(1):36 – 48, 2009.

[33] L. Rabiner, A. Rosenberg, and S. Levinson. Considerations in dynamic time warping algorithms for discrete word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(6):575–582, 1978.

[34] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. In *Proceedings of ACM SIGKDD 2012*, 2012.

[35] H. Sakoe. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, Feb. 1978.

[36] S. Salvadore and P. Chan. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Proceedings of the 3rd Workshop on Mining Temporal and Sequential Data*, 2004.

[37] J. Shieh and E. Keogh. iSAX: disk-aware mining and indexing of massive time series datasets. *Data Mining and Knowledge Discovery*, 19(1):24–57, Feb 2009.

[38] W. Xu, D. P. Miranker, R. Mao, and S. Ramakrishnan. Anytime k-nearest neighbor

search for database applications. *International Workshop on Similarity Search and Applications*, 0:139–148, 2008.

[39] B. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *Proceedings of the 26th International Conference on Very Large Data Bases*, pages 385–394, 2000.

[40] B. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of 14th International Conference on Data Engineering*, pages 201–208. IEEE, 1998.

[41] P. Zhang, Y. Huang, S. Shekhar, and V. Kumar. Exploiting spatial autocorrelation to efficiently process correlation-based similarity queries. In T. Hadzilacos, Y. Manolopoulos, J. Roddick, and Y. Theodoridis, editors, *Advances in Spatial and Temporal Databases*, volume 2750 of *Lecture Notes in Computer Science*, pages 449–468. Springer Berlin / Heidelberg, 2003.