Primitives for Online Time Series Analysis

December 19, 2003

1 Summary

Data arriving in time order (a data stream) arises in fields ranging from physics to finance to medicine to music, just to name a few. Often the data comes from sensors (in physics and medicine for example) whose data rates continue to improve dramatically as sensor technology improves. Further, the number of sensors is increasing, so correlating data between sensors becomes ever more critical in order to distill knowlege from the data. On-line response is desirable in many applications (e.g., to aim a telescope at an activity of interest or to perform magnetic resonance-based real-time surgery). These three factors – data size, correlation, and fast response – motivate our proposal. Our intent is *to build a foundational library of primitives to perform online or near online correlation and burst detection on thousands or even millions of time series.* Besides their direct use, such primitives could provide a first level analysis of time series for online clustering and data mining systems.

Methods and Intellectual Merit

The construction of these primitives will call for the development of algorithmically efficient methods, both in main and secondary memory. These include new algorithms for compressing the data in time series and for finding interesting relationships among portions of these series. The techniques, therefore, may be transferrable to any research area involving online compression and comparison. To ensure concreteness, our activity will be motivated by our collaborations with physicists and other natural and biomedical scientists.

Broader Impact

These primitives will enhance research and the national cyberinfrastructure in any application requiring the fusion of time series information. Further, this work encompasses all kinds of time series, including time series from music and other media. This will facilitate the inclusion of faculty and students from K-12, both majority and minority, who may be attracted to science through applications such as query by humming. Finally, our analytical tools can help earth, cosmological, and medical scientists extract information from their data more effectively than at present.

2 Results from Prior and Current NSF Support

Shasha's most closely associated research grant has been NSF grant IIS-9988345, titled "ASES: an approximate search engine for structure" covering the period September 2000 to September 2003 with a total budget of $282,440. Jason Wang of NJIT is a separately funded but co-equal collaborator of that grant. The goal was to build engines for approximate search in collections of trees and graphs. These have resulted in software and algorithms available to the community. The graphgrep system [66, 34], done with doctoral student Rosalba Giugno, for searching and pattern recognition in graphs has about 50 academic and industrial users worldwide. It allows users to search for undirected graphs in large databases of graphs. The query graph can be a subset of the larger graph. The tree searching software has been downloaded or used by more than 500 users and is the basis for the search component of the TreeBase phylogenetic system [61, 72] accessible from GenBank and the San Diego Supercomputer center. We also note that his earlier work with Kaizhong Zhang on pattern recognition in trees [67] is the basis of IBM's XMLDiff package. All such basic engine software is available from Shasha's web site http://cs.nyu.edu/cs/faculty/shasha/papers/papers.html.

Shasha also collaborates with two biological groups working on the model plant Arabidopsis. In these collaborations, he has contributed to experimental design (the iterative use of combinatorial design) [65], the discovery of transcription factor-binding site pairs using microarray information [8], and the inference of the function of uncharacterized genes through the analysis of whole species traits [54]. Most recently, he has done the bioinformatics work for the first cell sorting work on a multicellular organism [9]. Other biological work includes [71, 73, 63].

In parallel with these efforts, Shasha has worked on core computer science topics. His ongoing interest in database tuning has led to a second book on the subject with Philippe Bonnet [64]. Doctoral student Alberto Lerner and he have designed a database query language for ordered data (in support of the time series work below) [52]. David Mazieres and Shasha have designed a cryptographic file system [57]. Shasha has also worked on publish-subscribe algorithms and systems [29, 28].

Shasha's work on the algorithmic aspects of time series, done primarily with doctoral student Yunyue Zhu, began with the Statstream [76] system for finding highly correlated moving windows in thousands of data streams. This work will be reviewed in the body of this proposal. A second effort callsed HumFinder [78, 79] attempts to match a person's humming to recorded music and was demonstrated in a noisy room at SIGMOD in 2003. Doctoral student Zhihua Wang has contributed to this work. Other recent work concerns algorithms for burst detection across multiple window sizes [77]. The starting point for the latter work is a collaboration with the Milagro group who are trying to detect gamma ray bursts through an array of detectors placed in an artificial lake near Los Alamos. The challenge they face is that gamma ray bursts may last from a few milliseconds up to several days. For this reason, it is interesting to discover bursts no matter what their time scale and to do so in one pass if possible (see the enclosed letter from Professor Allen Mincer).

His outreach efforts include being co-editor-in-chief of Information Systems, writing the "Puzzling Adventures" column for *Scientific American*, his role as science advisor to the New York Hall of Science where he helps with exhibit design mostly aimed at K-9 students, and two recent books [64, 62].

Cole's research is currently supported by NSF grant CCR-0105678, titled "Algorithms for Approximation and Graph Problems," covering the period August 2001 to July 2004 with a total budget of $299,999. The project is concerned with devising algorithms for finding approximate solutions to problems in string and pattern matching, to problems in graphs, and elsewhere. This continues a ten year focus by the PI on problems in string and pattern matching. Highlights of

the work on pattern matching include (i) the exact analysis of the Boyer-Moore string matching algorithm [12] (a simplified version of which appears in several textbooks, e.g., [25, 39]); (ii) the definition and solution of the subset matching problem [16] which is cental to providing efficient algorithms for tree pattern matching; and which was subsequently applied elsewhere by others [40, 44].

In recent work, Cole and Hariharan defined the sparse matching problem [17], a generalization of both geometric and subset matching. The efficiency of their approach came from a new way of encoding strings when applying convolution algorithms. This use of convolutions dates back to Fischer and Paterson [31] who used them to give an $O(n \log m log\sigma)$ algorithm for string matching with wildcards, where σ is the alphabet size, n the text size, and m the pattern size. The technique used in the sparse matching work also allowed this 30-year old bound to be improved to $O(n \log m)$. Convolution plays a significant role in our planned work.

Cole has also worked on cache-oblivious algorithms [7, 5, 6]. Cache-oblivious algorithms are meant to perform efficiently at all levels of the memory hierarchy without requiring special parameter settings such as block size [32]. Cole has worked a variety of searching problems in this setting including point location in the plane. Searching problems across the memory hierarchy is a significant concern in our planned work.

Finally, Cole and PhD student David Kandathil have designed a main memory partition sorting algorithm that works in-place and runs faster than the best implementations of quicksort [22].

Cole has worked on a number of other problems during the period of his current NSF grant [18, 14, 13, 4, 60, 19, 20, 15, 21], including an improvement to suffix trees (a central data structure in string matching) with PhD student Lee-Ad Gottlieb.

3 Motivation

Many applications generate multiple data streams. For example,

- In mission operations for NASA's Space Shuttle, approximately 20,000 sensors are telemetered once per second to Mission Control at Johnson Space Center, Houston [50].
- The earth observing system data project consists of 64,000 time series covering the entire earth [10] though the satellite covers the earth in swaths, so the time series have gaps.
- There are about 50,000 securities trading in the United States, and every second up to 100,000 quotes and trades (ticks) are generated.

These applications share the following characteristics:

- Updates come in the form of insertions of new elements rather than modifications of existing data.
- Data arrives continuously.
- One pass algorithms to filter the data are essential because the data is vast. However, if the filter does its job properly, there should be few enough candidates that detailed analysis can take more time per candidate with only a modest impact on the overall running time.

Particularly significant are multi-stream statistics, because they permit the fusion of information from multiple sources. Further we want to compute such statistics repeatedly, over closely spaced

moving windows. For example, we might compute a standard statistical (Pearson) correlation over a one hour moving window, but report the results for moving windows spaced every two minutes. We call this *periodic window processing*.

We contrast this work with the considerable recent body of work on massive data streams [26, 56, 36] where the assumption is that data can be read once and is not stored. In our applications, we assume an initial filtering step must be completed in one pass, but a second pass may select portions of data from a time-ordered historical log.

4 Problem Statement

A data stream, for our purposes, is a potentially unending sequence of data in time order. For specificity, we consider data streams that produce one data item each time unit.

Correlation over windows of different streams has many variations. Here we discuss a few paradigmatic problems.

- (All parameters fixed) Given N_s streams, a start time t_start, and a window size w, find, for each time window W of size w, all pairs of streams s_1 and s_2 such that s_1 during time window W is highly correlated (over 0.95 typically) with s_2 during the same time window. (Possible time windows are $[t_{start}..t_{start+w-1}]$, $[t_{start+1}..t_{start+w}]$, ... where t_{start} is some start time.)

- (Lagged correlation) Allow shifts in time. That is, given N_s streams and a window size w, find all time windows $W1$ and $W2$ where $|W1| = |W2| = w$ and all pairs of streams s_1 and s_2 such that s_1 during $W1$ is highly correlated with s_2 during $W2$.

- (Largest correlating window size) Find the maximum window size w for which the above (lagged or unlagged) correlations are maximized.

- (Damping) Nearly orthogonal to the kind of correlation we seek is the question of whether recent data counts more than older data within a window of interesting size. If more recent data is more important, then it is appropriate to damp older values within the windows so their values count less in the correlation. Note that damping is not completely orthogonal to window size, because damping may allow windows to be unbounded in the past.

We characterize these variations by their parameter settings: fixed window size/find maximum window size, lag/no-lag, damp/no-damp.

In fact, the problem space is larger still. For example, here are two other problems that we are exploring.

- (Correlated Burst Detection) For this setting, we suppose that the data stream records events of one or more types. The task is to identify periods of burst, namely a window W during which the number of events of a given type exceeds some threshold value for window size $|W|$ [77]. We would like to be able to do this for many (thousands of) window sizes and many (thousands of) event types. The thresholds increase monotonically (but usually sub-linearly) with window size and may vary according to event type (e.g., lower for warnings and higher for failures). Then we would also like to correlate the bursts among different event types. This can be as simple as finding two time series $s1$ and $s2$ where the periods of burst correlate strongly over time perhaps with a lag. The opportunity here is that the digests are particularly simple because bursts can be characterized as starttime, duration pairs.

- (Time Warped Correlation) We are interested in comparing time series produced by imperfect sources, e.g., the time series of a Beatles song compared with the humming of the PI [49, 78]. In such an application, we may want to correlate windows of different sizes, implying that one of the time series may need to be compressed time-wise and this compression may have some jitter. We believe that many of the techniques developed in this proposal will contribute to the development of scalable algorithms for this problem, particularly our ideas for improving sketching.

4.1 Classifying the Problem

Given N_s streams and a window of size $winsize$, computing all pairwise correlations naively requires $O(winsize \times (N_s)^2)$ time. However, a classification of the time series into a specialized case and a general case leads to two general techniques, both of which enable substantial improvements over the naive approach.

- **Category 1:** The time series often exhibit a fundamental degree of regularity, at least over the short term, allowing long time series to be compressed to a few coefficients with little loss of information using data reduction techniques such as Fast Fourier Transforms and Wavelet Transforms. Using Fourier Transforms to compress time series data was originally proposed by Agrawal et al. [2]. This technique has been improved and generalized by [30, 55, 59]. Wavelet Transforms (DWT) [11, 33, 58, 74], Singular Value Decompositions (SVD) [51], and Piecewise Constant Approximations [48, 47, 69, 75] have also been proposed for similarity search. Keogh has pioneered many of the recent ideas in the indexing of dynamic time warping databases [46, 70]. The performance of these techniques varies depending on the characteristics of the datasets [68].

- **Category 2:** In the general case, such regularities are absent. However, sketch-based approaches [1, 41] can still give a substantial data reduction. These are based on the idea of taking the inner product of each time series window, considered as a vector, with a set of random vectors (or equivalently, this can be regarded as a collection of projections of the time series windows onto the random vectors). Thus, the guarantees given by the Johnson-Lindenstrauss lemma [45], hold. In time series data mining, sketch-based approaches have been used to identify representative trends [24, 42] and to compute approximate wavelet coefficients [33], for example.

5 Previous Work

In previous work [76], we showed how to solve the first correlation problem above online or almost online when high quality digests could be obtained using Fourier analysis (Category 1 above).

To be concrete, consider a setting in which each stream produces a data item every second. We want to compute the most highly correlated pairs (without lags or damping) over windows of length one hour and want to compute these every two minutes. On a Pentium 4 PC, the naive method (of comparing every stream with every other using a dot product) permits the identification of high correlations among 700 streams, whereas the method outlined here permits the computation of correlations for 10,000 streams. When more streams are present, the advantage will increase because the complexity of our algorithm increases linearly (in the number of streams) whereas the

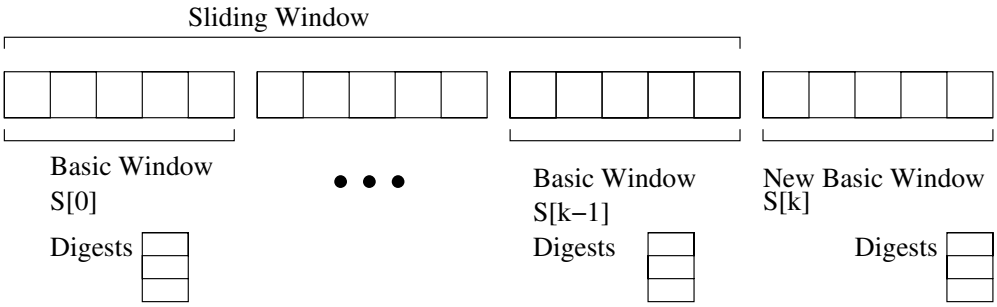

Figure 1: Sliding windows and basic windows

naive algorithm's complexity increases quadratically. Analogous speedups were previously obtained by [2] in a non-online setting.

Our approach begins by distinguishing among three time periods from smallest to largest.

- timepoint – the smallest unit of time over which the system collects data, e.g., a second.
- basic window – a consecutive subsequence of timepoints over which the system maintains a *digest* (i.e., a compressed representation) e.g., two minutes.
- sliding window – a user-defined consecutive subsequence of basic windows over which the user wants statistics, e.g., an hour. The user might ask, "which pairs of streams were correlated with a value of over 0.9 for the last hour?"

Figure 1 shows the relationship between sliding windows and basic windows.

The use of the intermediate time interval that we call the basic window yields two advantages:

1. (Near online response rates) Results of user queries need not be delayed more than the basic window time. In our example, the user will be told about correlations for the 2PM to 3PM window by 3:02 PM and correlations for the 2:02 PM - 3:02 PM window by 3:04 PM.[1]
2. (Free choice of window size) Maintaining stream digests based on the basic window allows the computation of correlations over windows of arbitrary size (chosen up front) with high accuracy.

The general strategy is just this: after a basic window ends, compute a digest comprising the first few coefficients of the Fourier Transform of the vector $(0, ..., 0, b)$ of length a sliding window, where b denotes the basic window contents. An entire sliding window consisting of k basic windows can be described by a vector of coefficients formed from a linear combination of the digests for the k basic windows. So, after computing the needed Fourier Transform of the latest basic window, it is simple to update the 4 to 8 coefficients for the entire sliding window. This takes constant time per coefficient.

Fourier Transforms do a good job of describing many time series, as pointed out by [2] and subsequently used in [30, 35, 55, 59]. Our empirical studies [76] using both random walk and stock market data show that the cost savings compared to a naive approach are significant and the

[1] One may wonder whether the basic window and therefore the delay can be reduced. The tradeoff is with computation time. Reducing the size of the basic window reduces the compression achieved and increases the frequency and hence expense of correlation calculations.

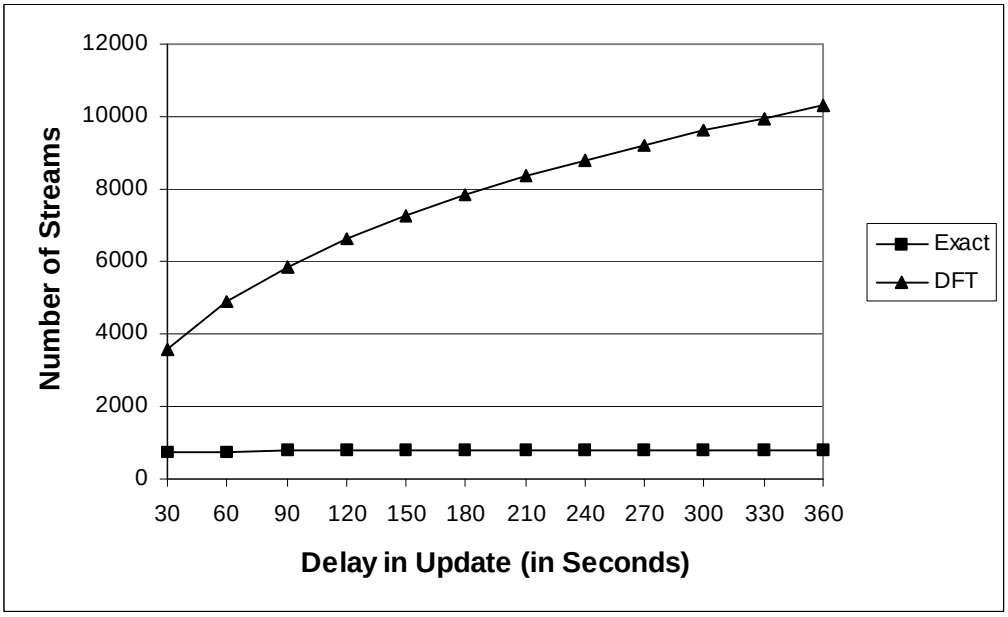

Figure 2: Comparison of the number of streams that the DFT Estimator and Exact methods can handle. The DFT results are checked for false positives in a post-processing step.

accuracy is excellent (no false negatives and few false positives). For speed, we were concerned with the following two questions:

1. How many streams can the system track continuously?
2. How long is the delay until the result is known?

Unsurprisingly, the net result is that the answers to questions (1) and (2) are related. We can increase the number of streams at the cost of increasing the delay in reporting correlations as illustrated in Figure 2. In both cases, we can ensure there are no false negatives by insisting that a slightly smaller correlation is sought on the DFT to account for the loss of information due to compression.

6 Preliminary Results

In our previous work, we explored the online correlation problem in the setting no-lag, no-damping, and fixed window size. To illustrate our approach to the several problem variants we mentioned, we discuss our ideas on the identification of lagged correlations. When a few Fourier (or Wavelet) coefficients effectively capture the data in the time series windows, we propose to develop methods using digests based on these coefficients. When these representations are not effective, we plan to attack the problem using digests based on sketches. For both approaches, we face two main issues:

1. How to compute the digests sufficiently quickly.
2. How to organize these digests so that they require less space than the original series.

Our preliminary work has focussed primarily on issue (1). We plan to build on other work in the literature for issue (2), as we discuss below.

6.1 Representing Data Compactly

We review the essence of the approaches to representing a sliding window $\mathbf{x}$=x_1,x_2,...x_m of real valued data. $\mathbf{x}$ can be thought of as a point in an m-dimensional space, R^m. The goal is to map $\mathbf{x}$ to a point $\mathbf{f}(\mathbf{x}) \in R^d$, a space of much smaller dimension, such that interpoint distances[2] are approximately preserved, i.e., with high probability, for the set of points at hand, for any pair of points $\mathbf{x}$, $\mathbf{y}$, $\| \mathbf{x}-\mathbf{y} \|_2 \approx \| \mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y}) \|_2$ (more precisely, for a given ϵ>0, $(1-\epsilon)\| \mathbf{x}-\mathbf{y} \|_2 \leq \| \mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y}) \|_2 \leq (1+\epsilon)\| \mathbf{x}-\mathbf{y} \|_2$).

Function $\mathbf{f}(\mathbf{x})$ might be the first k coefficients of the Discrete Fourier Transform (DFT), each coefficient being a complex number, or of the Discrete Wavelet Transform (DWT), or d values computed by a sketching method. (Recall that $f_i = f_{m-i}{}^*$, for Fourier coefficients $f_0, f_1, ..., f_{m-1}$ where z^* is the complex conjugate of z.)

Let $\mathbf{g}(\mathbf{x})$ denote the n-vector comprising the full DFT or DWT. It is well known that $\| \mathbf{g}(\mathbf{x}) \|_2 = \| \mathbf{x} \|_2$ and further there is a complementary transform $\mathbf{g}^{-1}$ such that $\mathbf{g}^{-1}\mathbf{g}(\mathbf{x}) = \mathbf{x}$. Clearly, if $\mathbf{f}(\mathbf{x})$ comprises the first few terms of $\mathbf{g}(\mathbf{x})$ then $\| \mathbf{f}(\mathbf{x}) \|_2 \leq \| \mathbf{g}(\mathbf{x}) \|_2$. This leads to the well-known notions of the power of $\mathbf{f}$ on $\mathbf{x}$.

Definition $\frac{\|\mathbf{f}(\mathbf{x})\|_2}{\|\mathbf{x}\|_2}$ is the power of $\mathbf{f}$ on $\mathbf{x}$.

Fact: If $\frac{\|\mathbf{f}(\mathbf{x})\|_2}{\|\mathbf{x}\|_2}$, $\frac{\|\mathbf{f}(\mathbf{y})\|_2}{\|\mathbf{y}\|_2} \geq 1-\delta$ then $\| x-y \|_2 - \delta(\| x \|_2 + \| y \|_2) \leq \| \mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y}) \|_2 \leq \| x-y \|_2 + \delta(\| x \|_2 + \| y \|_2)$.

Thus to obtain good distance estimate it suffices to have a transform that captures much of the power in a few coefficients. For slowly varying data, the DFT is known to be quite effective on time series data [2, 30, 68].

A transform closely related to the DFT is the Discrete Cosine Transform (DCT): on an m term series x_0,x_1,...,x_{m-1} it is the DFT of the series x_0,x_1,...,x_{m-1},x_{m-2},...,x_1, or similar mirror image series. The DCT is widely used in image and audio compression (JPEG and MP3), as it avoids the power losses using the DFT due to the mismatch between the values at the start and end of the series. Our preliminary experiments on stock price time series show that using 10 real coefficients there is only a 1.5% power loss, making it much more effective than the DFT (power loss of 14%) for these applications.

6.2 Time Lagged Correlations

The difficulty we face, both in using the DFT (or DCT or DWT) and in the sketch approach, is that while the basic approach is well understood, the obvious implementations of these ideas remain too expensive to be feasible. Overcoming this requires a mix of new algorithmic ideas and engineering. We discuss the transform and sketch approaches in turn.

6.2.1 Invertible Transforms

The basic method is the following. For each stream, for each possible position of the sliding window, (i) its digest is computed, (ii) the digest is stored in a near-neighbor data structure ND for points in R^d, (iii) its near neighbors in ND are reported along with the precise correlations to those neighbors. There are a number of issues to face.

[2] Algorithmically, distances can be treated interchangeably with correlations, because correlations are just normalized distances.

1. After each timepoint, we need to compute a digest for each sliding window ending at that timepoint.

2. To enable an efficient search in ND, the dimension R^d of the space for the digest points must be small.

3. We would like to guarantee no false negatives and relatively few false positives emerging from ND.

As noted in our prior work, to update the k leading coefficients of the DFT requires $O(k)$ time. This is also true for the Harr wavelet form of the DWT, and the DCT. Other instances of the DWT have to be considered on a case by case basis but in any event take no more than $O(k^2)$ time to update ([23], in preparation).

To address (2) we need to ensure that a few coefficients, say 5-10 real terms, capture most of the power. The DCT does an excellent job on certain kinds of series. But not all: for example, for series of stock price returns (the difference between a price of a share and the previous price of the share normalized by the average share price), 200 coefficients capture only 60% of the energy. For such series we will use sketches.

The reason we want a small dimension for the space R^d stored in ND is that near neighbor data structures quickly become less effective as the numbers of dimensions increase. In our prior work, a simple grid-based structure sufficed, but our research will include the evaluation of several multi-dimensional data structures to support nearest neighbor searching.

We note that if we are searching for points within distance $\delta(\| \mathbf{x} \|)_2$ of $\mathbf{x}$ (e.g., δ=5%) to avoid false negatives, we will need to retrieve points up to some distance $(\delta+\epsilon)\| \mathbf{f}(\mathbf{x}) \|_2$ from $\mathbf{f}(\mathbf{x})$, where the ϵ accounts for the power loss in $\mathbf{f}(\mathbf{x})$ and each of the candidate neighbors in ND (for a power loss of c%, $\epsilon = \frac{200c}{100-c}\% \approx 2c\%$ suffices.) However, to avoid being swamped with candidate false positives, there must be few points between distance $\delta\| \mathbf{f}(\mathbf{x}) \|_2$ and $(\delta+\epsilon)\| \mathbf{f}(\mathbf{x}) \|_2$ of $\mathbf{f}(\mathbf{x})$. Clearly this depends largely on the distribution of points $\mathbf{x}$ (i.e. the sliding window contents viewed as points in R^m), but our preliminary experiments on physics data suggest this holds for many applications. Formally, we capture the above by the following assumption.

Assumption: (*Bounded Expansion* [38]) Let $B_r(p)$ denote the ball of radius r around point p. We say a pont set S has expansion c if for all $p \epsilon S$ and all $r > 0$, there is an n_0 such that:
$\| B_r(p) \| \geq n_0$ implies that $\| B_{2r}(p) \| \leq c \| B_r(p) \|$.
Typically, c is a constant and n_0 is a small value such as $\log \| S \|$. This is exactly what is needed to keep the cost of handling false positives moderate.

For some of our applications the number of points stored in ND is very large (e.g. the physics observational data comprises some 64,000 streams, and each stream yields from 10,000 to 1,000,000 sliding window positions per stream, which is more than one billion points). Clearly, reducing the number of points that need to be stored will have a substantial effect on performance.

To this end, we observe that for plausible data streams the following assumption holds.

Assumption: (*Stream continuity*) Let p be the correlation of data streams s_1 and s_2 over a window of length w starting at times t_1 and t_2 respectively. Similarly, let $p\prime$ be the correlation starting at times t_1+1 and t_2+1, respectively. Then $| p - p\prime | = O(1/| w |)$.

The practical consequence is that if a high correlation pair of windows is found over streams s_1 and s_2 then the high correlation will continue for some interval of time $\triangle$ (possibly chosen adaptively as in the trails of [2]) and consequently correlations do not need to be sought at every timepoint. But recall our primary goal was to reduce the size of ND. Instead of storing a point for every position of the sliding window over each stream, it suffices to store points corresponding

to positionings of the sliding window at separations of $\triangle$. (The actual value of $\triangle$ depends on the various sources of error, but a value in the range 10-100 appears plausible.) We note, however, that there will be a need to perform a query for a point corresponding to each possible position of the sliding window, and not merely those at separations of $\triangle$.

6.2.2 Further Gains Using The DFT

When using the DFT it may be possible to both store only every $\triangle$th point and query every $\triangle$th point for each stream. This would have to be offset against the need for more coefficients in the DFT compared with the DCT to obtain a given bound on the power loss.

The key observation is that the DFT $f_0, ...f_{n-1}$ for the series $x_0, x_1, ...x_{m-1}$ and the DFT $g_0, g_1, ...g_{n-1}$ for the series $x_h, x_{h+1}, ..., x_{m-1}, x_0, ..., x_{h-1}$ are related as follows: $g_i = e^{-2\pi ih/n} f_i$. In addition, for small h, the series $x_h, ..., x_{m-1}, x_0, ..., x_{h-1}$ is a good approximation to $x_h, ..., x_{m-1}, x_m, ..., x_{m+h-1}$.

Let $\mathbf{x}$ denote the series $x_0, x_1, ..., x_{m-1}$ and $\mathbf{x}^h$ denote $x_h, ..., x_{m-1}, x_0, ..., x_{h-1}$. The search problem becomes the following: on a query $\mathbf{y}$ find those $\mathbf{x}$ such that $\mathbf{y}$ and $\mathbf{x}^h$ are nearly the same for small values of h (dependent on $\mathbf{x}$). Given $\mathbf{x}$ and $\mathbf{y}$, the value of h of a small bounded size that minimizes $\| \mathbf{y} - \mathbf{x}^h \|_2$ can be found easily: one uses the power series expansion of $\mathbf{f}(\mathbf{x}^h)$ as a function of h and solves for the minimum h.

Thus the remaining issue is to create a data structure supporting the retrieval of such points $\mathbf{x}$. One approach is to impose a canonical orientation on the Fourier coefficients, so that, say, the largest coordinate has a zero imaginary part.

An open question is whether something similar can be devised for the potentially better DCT.

6.2.3 The Sketch Approach

The sketch approach, as developed by Alon et al. [3], Indyk et al. [41, 43], and Achlioptus [1], provides a very nice guarantee: with high probability a random mapping taking points in R^m to points in $(R^d)^{2b+1}$ (the (2b+1)-fold cross-product of R^d with itself) approximately preserves distances.

Specifically, given a point $\mathbf{x} \in R^m$, we compute its dot product with d random vectors $\mathbf{r}_i \in \{1, -1\}^m$. The first random projection of $\mathbf{x}$ is given by $\mathbf{y}_1 = (\mathbf{x} * \mathbf{r_1}, \mathbf{x} * \mathbf{r_2}, ..., \mathbf{x} * \mathbf{r_d})$. We compute $2b$ more such random projections $\mathbf{y_1}, ..., \mathbf{y_{2b+1}}$. If $\mathbf{w}$ is another point in R^m and $\mathbf{z_1}, ..., \mathbf{z_{2b+1}}$ are its projections using dot products with the same random vectors then the median of $\|\mathbf{y_1} - \mathbf{z_1}\|, \|\mathbf{y_2} - \mathbf{z_2}\|, ...\|\mathbf{y_{2b+1}} - \mathbf{z_{2b+1}}\|$ is a good estimate of $\|\mathbf{x} - \mathbf{w}\|$. It lies within a $\theta(1/d)$ factor of $\|\mathbf{x} - \mathbf{w}\|$ with probability $1 - (1/2)^b$.

For our application, this leads to $2b + 1$ near-neighbor search structures over d-dimensional points. The retrieval method for finding all points within distance $\alpha\|\mathbf{p}\|$ of some query point $\mathbf{p}$ is to first retrieve all points within distance $2\alpha\|\mathbf{p}\|$ of $\mathbf{p}$ in each of the $2b + 1$ search structures, but only to report those that appear at least $b + 1$ times.

We need to limit the value of d needed to generate a chosen error bound ϵ in the equation $\epsilon = \theta(1/d)$. To ensure no false negatives, we require that distances of value $\alpha\|\mathbf{p}\|$ are reported as no more than $2\alpha\|\mathbf{p}\|$, i.e. $\epsilon \leq 1$. To bound the false positives, we require that distances of value more than $4\alpha\|\mathbf{p}\|$ are reported as more than $2\alpha\|\mathbf{p}\|$, i.e. $\epsilon \leq 1/2$. $d = 8/(\epsilon - (\epsilon^2/2))$ suffices. We believe that better bounds hold in practice and will test this experimentally.[3]

[3]It is known that the vectors $(\pm(1/\sqrt{d}))^n$ are the worst case for this type of projection, so we can simply test the effect of projections on these vectors. This will yield bounds on the performance of these methods on all vectors.

Computing the Sketches

For each random vector $\mathbf{r}$ of length equal to the sliding window length kw, we compute the dot product with each successive length kw portion of the stream (successive portions being one timepoint apart and w being the length of a basic window). As noted by Indyk [42], convolutions (computed via DFTs) can perform this efficiently off-line. The difficulty is how to do this efficiently online. One approach is to compute a series of dot products (using convolutions) for windows of doubling length (1,2,4,...,$kw/2$). This will yield the sought dot products online using $O(\log^2 n)$ work for each new item in each stream per dot product (i.e., per random vector).

If results can be delayed the length of a basic window, then the above computation need be done only for window lengths $(w, 2w, 4w, ..., |\text{sliding window}|\ /2)$. This entails $O(\log^2 kw - \log^2 2w)$ work per random vector per data item in each stream. Working out the constants, one arrives at roughly 750 floating point multiplies versus 3,600 integer additions in the naive method.

A further possible improvement is to use a "structured" random vector if you'll excuse the apparent oxymoron. The idea is to form each structured random vector $\mathbf{r}$ from the concatenation of k random vectors: $\mathbf{r} = \mathbf{s_1}, ..., \mathbf{s_k}$ where each $\mathbf{s_i}$ has length w. Further each $\mathbf{s_i}$ is either $\mathbf{u}$ or $-\mathbf{u}$, and $\mathbf{u}$ is a random vector in $\{1, -1\}^w$. This choice is determined by a random binary k-vector $\mathbf{b}$: if b_i=1, $\mathbf{s_i}$=$\mathbf{u}$ and if b_i=0, $\mathbf{s_i}$=$-\mathbf{u}$. The structured approach leads to an asymptotic performance of $O(k)$ integer additions and $O(\log w)$ floating point operations per datum and per random vector. For the example problem size this amounts to 30 integer additions, 84 floating point multiplications, and 42 floating point additions. Overall, this is a 30 to 40 factor improvement over the naive method.

In order to compute the dot products with structured random vectors, we first compute dot products with the random vector $\mathbf{u}$. We perform this computation by convolution once every w timesteps. Then each dot product with $\mathbf{r}$ is simply a sum of k already computed dot products.

Structured random vectors may influence the probability of an unduly inaccurate length estimate. We have ideas about how to bound that inaccuracy, but they are incomplete.

7 Other Challenges

Our discussion up to here has been to show that there are several avenues to explore that look very promising. In fact, however, they only begin to attack a large collection of interrelated problems. What follows are brief discussions of a sample of other challenging issues with only limited indications of approach.

7.1 Time Decay

A parameter that may be helpful in analyzing time series is to weight recent data more heavily than older ones. Recently, Cohen and Strauss [27] considered a variety of decay functions and how to maintain the time decayed sum and average of a series online. Analogously, we are interested in maintaining time-decayed distances both for digests and sketches.

A decay function simply multiplies the terms of time series to produce a new time series on which the same statistics as before are computed. Thus given a decay function d, and a time series x_1, x_2.... x_m, one obtains the new series, $x_1 d(m-1)$, $x_2 d(m-2)$,...$x_{m-1} d(2)$, $x_m d(1)$. The classic decay function is exponential decay, $d(x) = c^{-x}$ for some constant $c > 1$. This is generalized to PolyExp decay [27], $d(x) = poly(x)c^{-x}$. Cohen and Strauss also introduced Polynomial Decay, $d(x) = 1/x^\alpha$ for $\alpha \geq 2$.

Maintaining a sketch in this setting requires performing the dot product of a time-decayed series and a random vector $\mathbf{r}$. We note that applying the decay function to $\mathbf{r}$ instead of the time series would yield the same result. So it seems likely that the sketching approaches outlined earlier in this proposal (as well as other sketching approaches) will carry over to the present setting.

Computing digests of time-decayed series is not quite as simple. For exponential and poly-exponential decay, digests can still be maintained in $O(k)$ time per new datum in the series. For polynomial decay, our only result to date is a solution requiring $O(k \log n)$ time per new datum (using a bucketing approach along the lines of Cohen and Strauss). This adds a new source of approximation error (beyond that inherent in the digest): the bucketing error. It remains an open question whether there is a more efficient and accurate solution.

Finally, note that time decay functions permit unbounded length windows, so long as $\sum_{k \geq 1} d(i)$ is bounded. This introduces new challenges. Fortunately, as a practical matter, the windows are effectively bounded, since there is some bound b such that for $i \geq b$, $d(i)$ is negligible.

7.2 Finding Longest High Correlation Windows

As already noted in the stream continuity assumption of Section 6.2.1, it is quite likely that if the windows of length l starting at times t_1 and t_2 over streams s_1 and s_2 respectively have a high correlation, then so do the windows of length l starting at time t_1+1 and t_2+1 respectively. For this reason, reporting sequences of such successive high correlation window pairs would often be useless. It might be better to report a maximal such pair.

Let W_1 and W_2 be windows over streams s_1 and s_2 starting at times t_1 and t_2, respectively with $|W_1| = |W_2|$. We define the h-extension of w to be the window ${W_1}^h = (t_1, ..., t_1 + |W_1| - 1 + h)$ over s_1; ${W_2}^h$ is defined analogously.

Definition: ${W_1}^k$ and ${W_2}^k$ are maximal correlation-c windows if ${W_1}^h$ and ${W_2}^h$ have correlation at least c for all $0 \leq h \leq k$, but ${W_1}^{k+1}$ and ${W_2}^{k+1}$ have correlation less than c.

By this definition, high correlation windows must maintain their high correlation at every time-point starting with window size $\| W_1 \|$. Such highly correlated maximal pairs can be computed by tracking the extensions of high correlation windows which start at length the size of a sliding window.

We note that there can still be overlapping pairs of high correlation windows under this definition, and indeed a window pair wholly contained in a larger window pair. Which to report is application dependent. Keeping track of all of them efficiently is a subject for future work.

A related open question comes from the fact that somewhat lower correlations would be interesting for larger windows. Would different correlations for different length windows simply require separate computations, or is it possible to reuse portions of the computation? In the case of sketches, the structured random vector approach again appears promising.

7.3 Secondary Memory

In our proposed solution scheme the near neighbor search structure ND can grow very fast. For example, using the sketches for a hundred million points in R^m (Just 1000 streams for 3 hours with a new datum each second), requires storing some 10^{10} points. As we seek further order of magnitude increases in the number and length of streams we can handle, it is evident ND will be kept in secondary storage.

Because we can reduce our searches to fairly small dimensionality, we will start with standard multidimensional structures (e.g. R-trees with STR packing [53]). In addition, new data structures

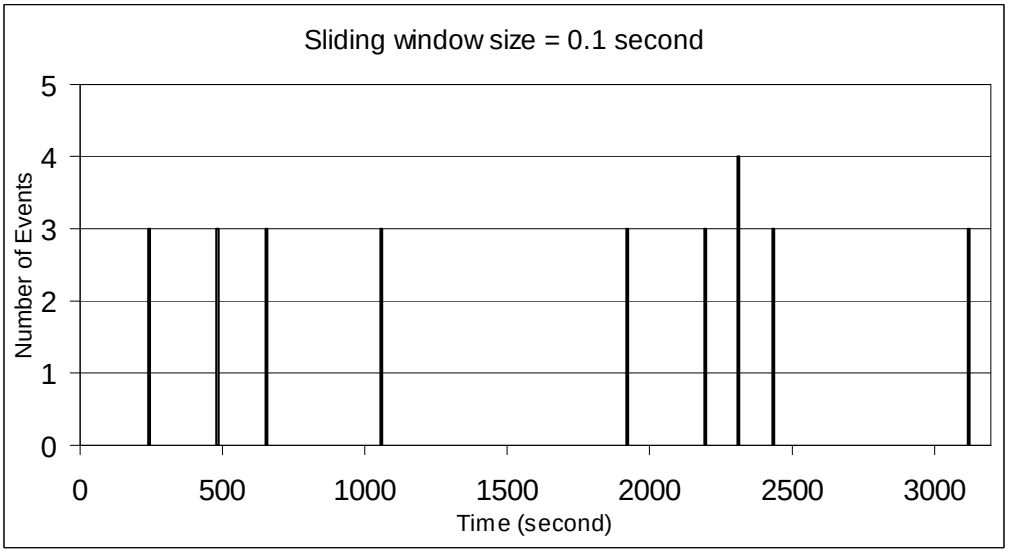

Figure 3: A 0.1 second window size and the number of events per window at moments of burst

giving low-distortion embeddings of high dimensional structures may turn out to be very useful [37, 38].

One natural question is whether it could be helpful to batch queries to ND; after all, we have already proposed computing correlations only every basic time window interval, so why not batch the ensuing queries?

Another useful technique may be to store only representative points in ND. If there are several nearby points, it would seem to be sufficient to store just one of them in ND. If need be, the others can be retrieved from a simple (say, hash) structure whose key is the representative point in ND. One would expect this situation to be increasingly common as the data size increases. The alternative, of course, is that the target correlation we seek would increase. This would further increase storage needs, as more accurate approximations in the digests or sketches would be required.

7.4 Labeled Time Series

Time series, such as those associated with specific locations (e.g., climate statistics), can be viewed as being labeled (e.g. by longitude and latitude). Those labels may influence the use of correlation, either to forbid reporting nearby correlates or focusing only on those.

7.5 Bursts

In our prior work on detection of gamma ray bursts, the goal was to detect events having a 10^{-6} probability, assuming background radiation occurred according to a Poisson distribution. We gave a highly efficient method that works simultaneously over a wide range of window lengths. Figures 3 and Figure 4 show records of burst at two window sizes that differ by a factor of 100. Windows of such different sizes may uncover bursts at different times.

Now we ask whether we can detect similarities among bursts occurring in different data streams. The problem is rather different here, for at this point it may be best to consider the bursts as forming a collection of very sparse data streams.

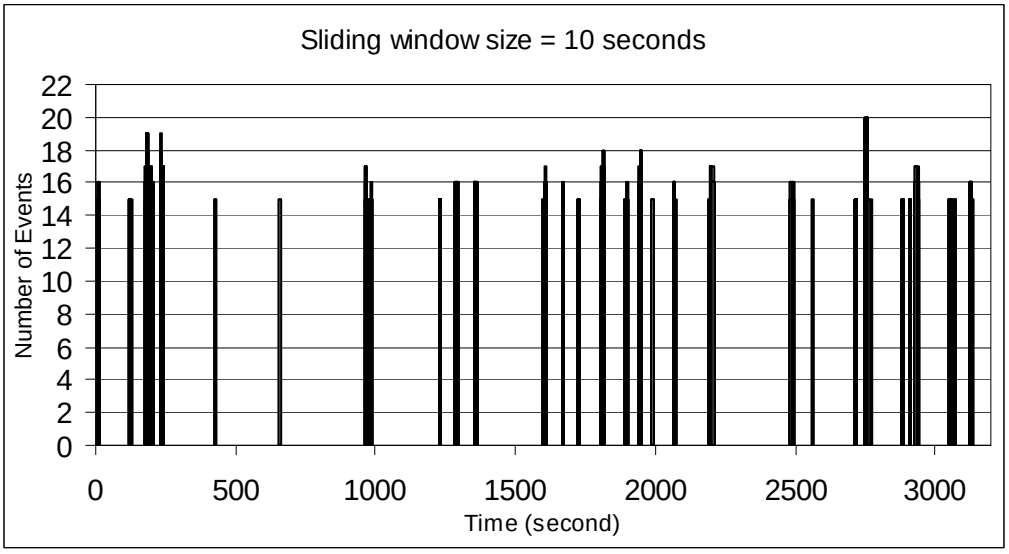

Figure 4: A 10 second window size and the number of events per window at moments of burst

We imagine that a single stream would have relatively few bursts so there is little difficulty in finding even the best lagged correlations among a pair of streams. However, if there are many streams, it remains open how to find stream pairs (or portions of stream pairs) with lagged correlations efficiently. The sketching approach appears promising, but we would need to avoid considering all possible positions for the random vectors, for otherwise the efficiency gains due to sparsity would be lost.

We also need to determine suitable definitions for a correlation. Is it simply a normalized sum of the overlap lengths of the bursts? Or would it be better to weight each burst by the number of burst events per time step? (Note that a large burst will need to have fewer such events per time step.)

8 Roadmap

While our approaches promise to improve performance by two orders of magnitude, the job is far from done. To find the highest correlating pairs among millions of time series will require significant further breakthroughs in data structures and approximations. In particular we believe that a synthesis of sketches and digests will prove helpful, perhaps by dynamically measuring the accuracy of the Fourier or Wavelet approximation. In addition, our current algorithms are highly and simply ("embarassingly") parallelizable because both the digest and sketch formation and comparison can be done independently for each stream. We will want to ensure that any algorithmic improvements retain the parallelizability.

- **Year 1**

  Algorithmic: Find effective algorithms for lagged correlation using both linear digests and sketches. Identify the best multidimensional data structures for the batch query mode that our application requires. This may require some data structure enhancements.

Implementation: Enhance our currently downloadable software beyond the no-lag, fixed window size, no-damp case to allow lags.

Collaboration: Complete integration and testing of our burst detection software in the Milagro project.

- **Year 2**

  Algorithmic: Study the best methods for the no-lag, no-damp case to extend correlations to the maximum possible window size. This will entail both the use of approaches such as hierarchical random vectors and methods for extending short term matches to longer term matches.

  Implementation: Study and implement cleaning and preparation algorithms for the data from the Jet Propulsion Laboratory (see letter). Establish a parallel platform.

  Collaboration: Find long time course data in biology.

- **Year 3**

  Algorithmic: Study the correlation problem in the no-lag, fixed window case with a view to extending the scale to up to millions of time series. This will entail careful management of secondary memory and it is here that enhancements on existing multidimensional data structures may be required.

  Implementation: Implement algorithms for the online correlation analysis of NASA and other massive online data sets, but starting with smaller data sets.

  Collaboration: Set up a web site that provides scientists who require our services easy access to all of our tools. Gather feedback to improve both our algorithms and software.

- **Year 4**

  Algorithmic: Study the correlation problem for bursts in all its variants. Study the basic correlation problem in full generality and at maximum scale again for millions of time series.

  Implementation: Implement algorithms for the simpler correlation problem on massive online data sets.

  Collaboration: Form an active user's group.

9 Management Plan

PI Shasha will perform project management and reporting to NSF, liaisons with application areas, system architecture, and will be a co-participant in algorithmic development. Co-PI Cole will serve as an algorithmic inventor and advisor to the implementors.

Our collaborations with domain scientists will give us both related problems to solve and test cases of utility.

10 Human Resources

Projects related to this proposal have already led to one doctorate, several well-received papers, and a book. Currently, there are four students (one PhD and three masters) working on Query by Humming and Burst Detection. This coming summer, high school students will join us. We believe

that the appeal of working with scientists and musicians will draw undergraduates (one will start next semester)and high school students into learning the mathematics of our techniques.

References

[1] D. Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281. ACM Press, 2001.

[2] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient Similarity Search In Sequence Databases. In D. Lomet, editor, *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO)*, pages 69–84, Chicago, Illinois, 1993. Springer Verlag.

[3] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *ACM Symposium on Theory of Computing*, pages 20–29, 1996.

[4] A. Amir, R. Cole, and E. P. M. Lewenstein. Function matching: algorithms, applications and a lower bound. In *30th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 929–942, 2003.

[5] M. Bender, R. Cole, E. Demaine, and M. Farach-Colton. Scanning and traversing: maintaining data for traversals in a memory hierarchy. In *European Symposium on Algorithms (ESA)*, pages 139–151, 2002.

[6] M. Bender, R. Cole, E. Demaine, and M. Farach-Colton. Two simplified algorithms for maintaining order in a list. In *European Symposium on Algorithms (ESA)*, pages 152–164, 2002.

[7] M. Bender, R. Cole, and R. Raman. Exponential search structures for efficient oblivious algorithms. In *29th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 195–207, 2002.

[8] K. Birnbaum, P. N. Benfey, and D. E. Shasha. cis element/transcription factor analysis (cis/tf): A method for discovering transcription factor/cis element relationships. *Genome Res.*, (11):1567–1573, 2001.

[9] K. Birnbaum, D. E. Shasha, J. Y. Wang, J. W. Jung, G. M. Lambert, D. W. Galbraith, and P. N. Benfey. A gene expression map of the arabidopsis root. *Science*, pages 1956–1960.

[10] A. Braverman. Personal communication. 2003.

[11] K.-P. Chan and A. W.-C. Fu. Efficient time series matching by wavelets. In *Proceedings of the 15th International Conference on Data Engineering, Sydney, Australia*, pages 126–133, 1999.

[12] R. Cole. Tight bounds on the complexity of the Boyer-Moore string matching algorithm. *SIAM Journal on Computing*, pages 1075–1091, 1994.

[13] R. Cole, Y. Dodis, and T. Roughgarden. How much can taxes help selfish routing? In *4th Electronic Commerce Conference*, pages 98–107, 2003.

[14] R. Cole, Y. Dodis, and T. Roughgarden. Pricing network edges for heterogeneous selfish users. In *35th Symposium on Theory of Computing (STOC)*, pages 521–530, 2003.

[15] R. Cole, L. Gottlieb, and M. Lewenstein. Dictionary matching and indexing with errors and don't cares. Submitted for publication.

[16] R. Cole and R. Hariharan. Tree pattern matching and subset matching in randomized $O(n \log^3 m)$ time. In *29th Symposium on Theory of Computing (STOC)*, pages 66–75, 1997.

[17] R. Cole and R. Hariharan. Verifying candidate matches in sparse and wildcard matching. In *34th Symposium on Theory of Computing (STOC)*, pages 592–601, 2002.

[18] R. Cole and R. Hariharan. A fast algorithm for computing Steiner edge connectivity. In *35th Symposium on Theory of Computing (STOC)*, pages 167–176, 2003.

[19] R. Cole and R. Hariharan. Faster suffix tree construction with missing suffix links. In *SIAM J. on Computing, Vol. 33, No. 1*, pages 26–42, 2003.

[20] R. Cole and R. Hariharan. Tree pattern matching to subset matching in linear time. In *SIAM J. on Computing, Vol. 32, No. 4*, pages 1056–1066, 2003.

[21] R. Cole, C. Iliopoulos, M. Mohamed, W. Smyth, and L. Yang. Computing the minimum k-cover of a string. In *Prague Stringology Conference*, 2003.

[22] R. Cole and D. Kandathil. The average case analysis of partition sorts. Submitted for publication.

[23] R. Cole, D. Shasha, and Y. Zhu. Fast algorithms for time-lagged correlation. 2004, in preparation.

[24] G. Cormode, P. Indyk, N. Koudas, and S. Muthukrishnan. Fast mining of massive tabular data via approximate distance computations. In *ICDE 2002, 18th International Conference on Data Engineering, February 26-March 1, 2002, San Jose, California*, 2002.

[25] M. Crochemore and W. Rytter. *Text and Algorithms*, pages 59–62. Oxford University Press, 1994.

[26] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA. ACM/SIAM, 2002*, pages 635–644, 2002.

[27] M. S. Edith Cohen. Maintaining time-decaying stream aggregate. volume PODS, pages 1–11, 2003.

[28] F. Fabret, F. Llirbat, J. Pereira, A. Jacobsen, and D. Shasha. Webfilter: A high-throughput xml-based publish and subscribe system. In *Very Large Database Systems*, pages 511–520, 2001.

[29] F. Fabret, F. Llirbat, J. Pereira, K. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe. In *ACM Sigmod*, pages 115–126, 2001.

[30] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 419–429, 1994.

[31] M. Fischer and M. Paterson. String matching and other products. In *Complexity of computation. SIAM-AMS proceedings, ed. R.M. Karp*, pages 113–125, 1974.

[32] M. Frigo, C. Leiserson, H. Prokop, and S. Ramachandran. Cache-oblivious algorithms. In *Symposium on Foundations of Computer Science (FOCS)*, pages 285–297, 1999.

[33] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *VLDB 2001*, pages 79–88. Morgan Kaufmann, 2001.

[34] R. Giugno and D. Shasha. Graphgrep: A fast and universal method for querying graphs. In *Proceeding of the IEEE International Conference in Pattern recognition (ICPR)*, Quebec, Canada, August 2002.

[35] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP'95)*, 1995.

[36] M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 58–66, 2001.

[37] A. Gupta, R. Krauthgamer, and J. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Symposium on Foundations of Computer Science (FOCS)*, pages 534–543, 2003.

[38] A. Gupta and M. Ruhl. Nearest neighbors in growth-restricted metrics. In *Symposium on the Theory of Computing (STOC)*, pages 741–750, 2002.

[39] D. Gusfield. *Algorithms on Strings*. Cambridge University Press.

[40] P. Indyk. Deterministic superimposed coding with application to pattern matching.

[41] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *40th Symposium on Foundations of Computer Science*, pages 189–197, 2000.

[42] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 363–372. Morgan Kaufmann, 2000.

[43] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM Press, 1998.

[44] P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric matching under noise: combinatorial bounds and algorithms. In *Symposium on Discrete Algorithms (SODA)*, pages 457–465, 1999.

[45] W. B. Johnson and J. Lindenstrauss. Extensions of lipshitz mapping into hilbert space. *Contemp. Math.*, 26:189–206, 1984.

[46] E. Keogh. Exact indexing of dynamic time warping. In *In 28th International Conference on Very Large Data Bases*, pages 406–417, 2002.

[47] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. ACM SIGMOD International Conf. on Management of Data*, 2001.

[48] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series. In *Databases. Knowledge and Information Systems 3(3)*, pages 263–286, 2000.

[49] E. Keogh and M. J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289, 2000.

[50] E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In *the third conference on Knowledge Discovery in Databases and Data Mining*, 1997.

[51] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 289–300, 1997.

[52] A. Lerner and D. Shasha. Aquery: Query language for ordered data, optimization techniques, and experiments. In *VLDB*, 2003.

[53] S. Leutenegger, M. Lopez, and J. Edgington. STR: A simple and efficient algorithm for R-tree packing. *Proc. of the 1997 International Conference on Data Engineering*, pages 497–506, 1997.

[54] M. Levesque, D. Shasha, W. Kim, M. G. Surette, and P. N. Benfey. Trait-to-gene: A computational method for predicting the function of uncharacterized genes. *Current Biology*, 13:129–133, Janurary 2003. Discussed in: http://www.the-scientist.com/yr2003/jun/hot_030603.html.

[55] C.-S. Li, P. S. Yu, and V. Castelli. Hierarchyscan: A hierarchical similarity search algorithm for databases of long sequences. In *ICDE*, pages 546–553, 1996.

[56] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Random sampling techniques for space efficientonline computation of order statistics of large datasets. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 251–262, 1999.

[57] D. Mazieres and D. Shasha. Building secure file systems out of byzantine storage. In *Twenty-First ACM Symposium on Principles of Distributed Computing*, 2002.

[58] I. Popivanov and R. J. Miller. Similarity search over time series data using wavelets. In *ICDE*, 2002.

[59] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 13–25, 1997.

[60] R.Cole, Z. Galil, R. Hariharan, S. Muthukrishnan, and K. Park. Parallel two dimensional witness computation. In *Information and Computation*, to appear.

[61] H. Shan, K. Herbert, W. Piel, D. Shasha, and J. T. L. Wang. A structure-based search engine for phylogenetic databases. In *SSDBM (Scientific Database Management)*, pages 7–10, 2002.

[62] D. Shasha. *Dr. Ecco's Cyberpuzzles : 36 Puzzles for Hackers and Other Mathematical Detectives*. W. W. Norton, isbn 0-393-05120-x edition, June 2002. Hardcover, 231 Pages.

[63] D. Shasha. Plant systems biology: Lessons from a fruitful collaboration. *Plant Physiology*, 132:1–2, June 2003.

[64] D. Shasha and P. Bonnet. *Database Tuning: Principles, Experiments, and Troubleshooting Techniques*. Morgan Kaufmann Publishers, isbn 1-55860-753-6 edition, June 2002. Paper, 464 Pages.

[65] D. Shasha, A. Kouranov, L. Lejay, M. Chou, and G. Coruzzi. Using combinatorial design to study regulation by multiple input signals. a tool for parsimony in the post-genomics era. *Plant Physiology*, (127(4)):1590–1594, Dec 2001.

[66] D. Shasha, J. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *ACM PODS*, pages 39–52, May 2002. Invited.

[67] D. Shasha and K. Zhang. *Approximate Tree Pattern Matching*, pages 341–371. Oxford University Press, 1997. Pattern Matching in Strings, Trees, and Arrays by A. Apostolico and Z. Galil (eds.), ISBN 0-19-511367-5.

[68] D. Shasha and Y. Zhu. *Fast Discovery in Massive Time Series*. Springer Verlag, 2004.

[69] E. K. D. G. W. T. T. Palpanas, M. Vlachos. Online amnesic approximation of streaming time series. In *In ICDE*, 2004.

[70] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multi-dimensional time-series with support for multiple distance measures. In *In the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 216–225, August 24 - 27, 2003.

[71] J. T. L. Wang, Q. Ma, D. Shasha, and C. H. Wu. New techniques for extracting features from protein sequences. *IBM Systems Journal, Special Issue on Deep Computing for the Life Sciences*, 40(2):426–441, 2001. Invited, accessible at http://www.research.ibm.com/journal/sj40-2.html.

[72] J. T. L. Wang, H. Shan, D. Shasha, and W. H. Piel. Treerank: A similarity measure for nearest neighbor searching in phylogenetic databases. In *Scientific and Statistical Database Management (SSDBM)*, Cambridge, MA (USA), July 2003.

[73] X. Wang, J. T.-L. Wang, D. Shasha, B. Shapiro, I. Rigoutsos, and K. Zhang. Finding patterns in three dimensional graphs: Algorithms and applications to scientific data mining. *IEEE Transactions on Knowledge and Data Engineering*, pages 731–749, 2002.

[74] Y.-L. Wu, D. Agrawal, and A. ElAbbadi. A comparison of dft and dwt based similarity search in time-series databases. In *Proceedings of the 9th International Conference on Information and Knowledge Management*, 2000.

[75] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 385–394. Morgan Kaufmann, 2000.

[76] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB 2002,Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 358–369, 2002.

[77] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *KDD 2003, Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 24-27, 2003,Washington, DC, USA*. ACM, 2003.

[78] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In A. Y. Halevy, Z. G. Ives, and A. Doan, editors, *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, pages 181–192. ACM, 2003.

[79] Y. Zhu, D. Shasha, and X. Zhao. Query by humming - in action with its technology revealed. In A. Y. Halevy, Z. G. Ives, and A. Doan, editors, *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, page 675. ACM, 2003.

Dennis Shasha — biographical sketch

Education

1977	B.S.	Yale University
1980	M.S.	Syracuse University (overlapped work at IBM Data Systems Division)
1984	Ph.D.	Harvard University

Positions

1984– Assistant Professor, Associate Professor, and Full Professor of Computer Science
Courant Institute of Mathematical Sciences
New York University

1993– Co-Editor-in-Chief (with Matthias Jarke) of *Information Systems*
Publisher: Elsevier

1991– Database tuning, design, and pattern discovery consulting for Bell Labs, Telcordia, Novell, finance, gaming, and biotech companies.

Publications Most Closely Related to the Proposed Project

1. "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time" Yunyue Zhu and Dennis Shasha, VLDB 2002.
2. "Query by Humming - in Action with its Technology Revealed" Yunyue Zhu, Dennis Shasha, and Xiaojian Zhao, ACM SIGMOD 2003.
3. "Efficient Elastic Burst Detection in Data Streams," Yunyue Zhu and Dennis Shasha, ACM KDD 2003.
4. "AQuery: Query Language for Ordered Data, Optimization Techniques, and Experiments," Alberto Lerner and Dennis Shasha, VLDB 2003.
5. *Fast Discovery in Massive Time Series*, Dennis Shasha and Yunyue Zhu Springer-Verlag, June 2004, in press.

Other Significant Publications

1. "A gene expression map of the Arabidopsis root" Kenneth Birnbaum, Dennis E. Shasha, Jean Y. Wang, Jee W. Jung, Georgina M. Lambert, David W. Galbraith, and Philip N. Benfey *Science*, Dec 12 2003: 1956-1960
2. "Trait-To-Gene: A Computational Method for Predicting the Function of Uncharacterized Genes," Mitchell Levesque, Dennis Shasha, Wook Kim, Michael G. Surette, and Philip N. Benfey, *Current Biology* 2003, vol. 13, pp. 129-133.

3. "Using Combinatorial Design to Study Regulation by Multiple Input Signals. A Tool for Parsimony in the Post-Genomics Era," Dennis Shasha, Andrei Kouranov, Laurence Lejay, Michael Chou, and Gloria Coruzzi, *Plant Physiology*, Dec. 2001 127(4):1590-1594.

4. "Building secure file systems out of Byzantine storage," David Mazieres and Dennis Shasha, Principles of Distributed Computing, 2002, pp. 108-117.

5. *Database Tuning: principles, experiments, and troubleshooting techniques*, Dennis Shasha and Philippe Bonnet, Morgan Kaufmann Publishers, June 2002, ISBN 1-55860-753-6, Paper, 464 Pages.

Synergistic Activities

Much of my work has to do with providing software for tree matching and searching, graph searching, and time series analysis. The software is available from http://www.cs.nyu.edu/cs/faculty/shasha/papers/papers.h Other synergistic activities have mostly to do with conveying the ways in which computer scientists think. (i) Write monthly puzzle column for Scientifc American as well as a three puzzle books detailing the adventures of Dr. Ecco, the latest of which is called *Dr. Ecco's Cyberpuzzles: 36 puzzles for hackers and other mathematical detectives*. (ii) Wrote book: *Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists, (with C. Lazere)* to interest people in the thought patterns of our field. (iii) Advise the New York Hall of Science on exhibits.

List of Collaborators or Potential Collaborators

I have collaborated with the following people during the last 48 months (or may collaborate with them due to my series editorship at Oxford for Genomics and Bioinformatics and Information Systems) in addition to those listed in the publications list: Michael Rabin, Charles Cantor, David Botstein, Lee Hood, Raju Kucherlapati, Michael Ashburner, Minoru Kanehisa, Nicole Bidoit-Tollu, Ricardo Baeza-Yates, Klaus R. Dittrich, Yannis Ioannidis, Matthias Jarke, Gottfried Vossen, Maurizio Lenzerini, Peri Loucopoulos, Patrick O'Neil, Felipe Carino Jr., Nick Koudas, Masatoshi Yoshikawa, Kenneth A. Ross, Bettina Kemme, and Amr El Abbadi.

Names of Graduate and Post-Graduate Advisors and Advisees

Nathan Goodman was my dissertation advisor. The following lists my advisees (their graduation year, doctoral dissertation title, and current affiliation are in parentheses): Kaizhong Zhang (1989, *The Editing Distance Between Trees: Algorithms and Applications*, Full Professor at the University of Western Ontario, Canada), Theodore Johnson (1990, *The Performance of Concurrent Data Structure Algorithms*, Research scientist at AT&T Laboratories in Florham Park), Jose Perez-Carballo (1990, *Design and Implementation of HyTeK: A Knowledge-Based Hypertext System*, Assistant Professor at Rutgers, New Brunswick), Jason Tsong-Li Wang (1991, *Query Optimization in Database and Information Retrieval Systems*, Full Professor at the New Jersey Institute of Technology, Newark), Vladimir Lanin (1991, *Semantically-Based Concurrent Data Structure Algorithms*, Member of Technical Staff at a startup company), Brian Anderson (1991, *Persistent LINDA: Design and Implementation of a System to Add Transactions to LINDA*, Technical lead at a startup company in California), John Turek (1991, *Algorithms for Robust Parallel Computation*, Manager at IBM Research, Hawthorne), Steve Rozen (1993, *Automating Physical Database Design: An Extensible Approach*, Research Scientist at the Whitehead Institute, MIT), Gilad Koren

(1993, *Competitive On-Line Scheduling for Overloaded Real-Time Systems*, Professor at Natanyu College, Israel), Karpjoo Jeong (1995, *PLinda 2.0: Fault Tolerant Parallel Computation on Idle Workstations*, Professor at Konkuk University, Korea), Bin Li (1998, *Free Parallel Data Mining*, Vice President at Citibank, New York), Peter Wyckoff (1998, *Fault Tolerant Parallel Computing on Networks of Non-Dedicated Workstations*, Research Scientist at Yahoo), Peter Piatko (1998, *Thinksheet: A Tool for Information Navigation*, Research Scientist at SAIC), David Tanzer (2000, *Queryable Expert Systems*, Current Position: start-up. Rosalba Giugno (2003, Graphgrep) *Searching Algorithms and Data Structures for Combinatorial, Temporal and Probabilistic Databases* Current position: Post-doc, University of Catania. Alberto Lerner (2003) *Querying Ordered Data with AQuery* Current position: Researcher, IBM T. J. Watson Research Center. Yunyue Zhu (2003) *High Performance Discovery in Time Series: techniques and case studies*

Mailing Address

Professor Dennis Shasha, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012 (shasha@cs.nyu.edu).

Richard John Cole – biographical sketch

Education

1978	B.A.	Oxford University, Mathematics
1980	M.S.	Cornell University, Computer Science
1982	Ph.D.	Cornell University, Computer Science

University Positions

1982–	Assistant Professor, Associate Professor, and Full Professor of Computer Science Courant Institute of Mathematical Sciences New York University
1994–2000	Chair, Computer Science Department, NYU
1998	ACM Fellow
2002	Member of editorial board: *SIAM Journal of Computing*

Publications Most Closely Related to the Proposed Project

1. "Verifying candidate matches in sparse and wildcard matching," Richard Cole and Ramesh Hariharan, STOC 2002: 592-601

2. "Dictionary matching and indexing with errors and don't cares," Richard Cole, Lee-Ad Gottlieb, and Moshe Lewenstein, Submitted for publication

3. "Exponential Structures for Efficient Cache-Oblivious Algorithms," Michael A. Bender, Richard Cole, Rajeev Raman, ICALP 2002: 195-207

4. "The Average Case Analysis of Partition Sorts" Richard Cole and David Kandathil Submitted for publication.

5. "Approximate String Matching: A Simpler Faster Algorithm." Richard Cole, Ramesh Hariharan, SIAM J. Comput. 31(6): 1761-1782 (2002)

Other Significant Publications

1. "Tight bounds on the complexity of the Boyer–Moore string matching algorithm," Richard Cole, *SIAM Journal on Computing*, 23(1994), 1075–1091

2. "Parallel merge sort" Richard Cole, *SIAM Journal on Computing*, 17(1988), 770-785

3. "On the dynamic finger conjecture for splay trees. Part II: the proof." Richard Cole, *SIAM Journal on Computing*, 29(2000), 44-85.

4. "Pricing network edges for heterogeneous selfish users" Richard Cole, Yevgeniy Dodis, Tim Roughgarden STOC 2003: 521-530

5. "A fast algorithm for computing Steiner edge connectivity" Richard Cole, Ramesh Hariharan STOC 2003: 167-176

Synergistic Activities

Educational material: description plus animation of binary search trees and splay trees. See www.cs.nyu.edu/algvis.

List of Collaborators or Potential Collaborators

Amihood Amir, Michael Bender, Eric Demaine, Yevgeniy Dodis, Martin Farach-Colton, Zvi Galil, Lee-Ad Gottlieb, Piotr Indyk, Ramesh Hariharan, David Kandathil, Moshe Lewenstein, Bruce Maggs, Bud Mishra, Kirsten Ost, Kunsoo Park, Ely Porat, Rajeev Raman, Therese Pzzytycha, Tim Roughgarden, Wajciech Rytter, Jeannette Schmidt, Alan Siegel, Stefan Schirra, Ramesh Sitaraman, Mikhal Thorup, T. Lecroq, W. Plandowski, Manal Mohamed, W. F. Smyth, Lee Yang, and Naila Rahman.

Co-editors at Siam Journal on Computing: Bernard M. Chazelle, David A. Eppstein, Steven Fortune, Michael L. Fredman, Harold N. Gabow, Oded Goldreich, R.L. Graham, Vassos Hadzilacos, Neil Immerman, A. Karlin, Ker I. Ko, S. Rao Kosaraju, Ming Li, John C. Mitchell, Rajeev Motwani, Prabhakar Raghavan, Vijaya Ramachandran, Michael Saks, David B. Shmoys, Peter Shor, Igor Shparlinksi, Janos Simon, Madhu Sudan, Eva Tardos, Eli Upfal, Moshe Y. Vardi, Umesh V. Vazirani, David P. Williamson, and Mihalis Yannakakis

Mailing Address

Professor Richard Cole, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, NY 10012 (shasha@cs.nyu.edu).