Proceedings of DMKD'03

8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery

13th June 2003, San Diego, California

in conjunction with
ACM SIGMOD International Conference on Management of Data

Sponsored by ACM SIGMOD

Editors

Mohammed J. Zaki
Computer Science Department
Rensselaer Polytechnic Institute
Troy, New York, USA

Charu C. Aggarwal
IBM T.J. Watson Research Center
Yorktown Heights, New York, USA

8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'03)

Mohammed J. Zaki
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180
zaki@cs.rpi.edu

Charu C. Aggarwal
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
charu@us.ibm.com

Foreword

This is the 8th workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'03), held annually in conjunction with ACM SIGMOD conference. The workshop aims to bring together data-mining researchers and experienced practitioners with the goal of discussing the next generation of data-mining tools. Rather than following a "mini-conference" format focusing on the presentation of polished research results, the DMKD workshop aims to foster an informal atmosphere, where researchers and practitioners can freely interact through short presentations and open discussions on their ongoing work as well as forward-looking research visions/experiences for future data-mining applications.

In addition to research on novel data-mining algorithms and experiences with innovative mining systems and applications, of particular interest in this year's DMKD workshop is the broad theme of "Future Trends in Data Mining." Topics of interest include:

- Privacy & Security issues in Mining: Privacy preserving data mining techniques are invaluable in cases where one may not look at the detailed data, but one is allowed to infer high level information. This also has relevance for the use of mining for national security applications.

- Mining Data Streams: In many emerging applications data arrives and needs to be processed on a continuous basis, i.e., there is need for mining without the benefit of several passes over a static, persistent snapshot.

- Data Mining in Bioinformatics and Biological Database Systems: High-performance data mining tools will play a crucial role in the analysis of the ever-growing databases of bio-sequences/structures.

- Semi/Un-Structured Mining for the World Wide Web: The vast amounts of information with little or no structure on the web raise a host challenging mining problems such as web resource discovery and topic distillation; web structure/linkage mining; intelligent web searching and crawling; personalization of web content.

- Future Trends/Past Reflections: What are the emerging topics/applications for next generation data mining? What lessons have been learned from over a decade of data mining? What areas need more attention?

Out of the 28 paper submissions we were able to select only 8 full papers (an acceptance rate of 28.5%). In addition 5 short papers were accepted. Each paper was reviewed by three members of the program committee. Along with an invited talk, we were able to assemble a very exciting program.

We would like to thank all the authors, invited speakers, and attendees for contributing to the success of the workshop. Special thanks to the program committee for help in reviewing the submissions, and ensuring a high quality program.

Sincerely,
Mohammed Zaki, Charu Aggarwal (editors)

Workshop Co-Chairs

Mohammed J. Zaki, Rensselaer Polytechnic Institute
Charu Aggarwal, IBM T.J. Watson Research Center

Program Committee

Roberto Bayardo, IBM Almaden Research Center, USA
Alok Choudhary, Northwestern University, USA
Gautam Das, Microsoft Research, USA
Venkatesh Ganti, Microsoft Research, USA
Minos N. Garofalakis, Bell Labs, USA
Dimitrios Gunopulos, University of California, Riverside, USA
Jiawei Han, University of Illinois at Urbana-Champaign, USA
Eamonn Keogh, University of California, Riverside, USA
Nick Koudas, AT&T Research, USA
Vipin Kumar, University of Minnesota, USA
Bing Liu, University of Illinois at Chicago, USA
Rosa Meo, University of Torino, Italy
Raymond Ng, University of British Columbia, Canada
Srini Parthasarathy, Ohio State University, USA
Rajeev Rastogi, Bell Labs, USA
Kyuseok Shim, Seoul National University, South Korea
Hannu Toivonen, University of Helsinki, Finland
Philip S. Yu, IBM T.J. Watson Research Center, USA

Workshop Program & Contents

The workshop will foster an informal atmosphere, where participants can freely interact through short presentations and open discussions on their ongoing work as well as discussion on current/future trends in data mining. Questions can be asked freely during any time, and discussion slots have been set aside in the morning and afternoon sessions. These slots are meant to focus on critical evaluation of current approaches and open problems. All speakers and participants are encouraged to think about these issues prior to the workshop and to take active part in the ensuing discussions.

Invited Talk
Analyzing Massive Data Streams: Past, Present, and Future

Minos Garofalakis
Bells Labs
600 Mountain Avenue
Murray Hill, NJ 07974, USA
minos@research.bell-labs.com

ABSTRACT

Continuous data streams arise naturally, for example, in the installations of large telecom and Internet service providers where detailed usage information (Call-Detail-Records, SNMP-/RMON packet-flow data, etc.) from different parts of the underlying network needs to be continuously collected and analyzed for interesting trends. Such environments raise a critical need for effective stream-processing algorithms that can provide (typically, approximate) answers to data-analysis queries while utilizing only small space (to maintain concise stream synopses) and small processing time per stream item. In this talk, I will discuss the basic pseudo-random sketching mechanism for building stream synopses and our ongoing work that exploits sketch synopses to build an approximate SQL (multi) query processor. I will also describe our recent results on extending sketching to handle more complex forms of queries and streaming data (e.g., similarity joins over streams of XML trees), and try to identify some challenging open problems in the data-streaming area.

A Symbolic Representation of Time Series, with Implications for Streaming Algorithms

Jessica Lin   Eamonn Keogh   Stefano Lonardi   Bill Chiu

University of California - Riverside
Computer Science & Engineering Department
Riverside, CA 92521, USA
{*jessica, eamonn, stelo, bill*}@*cs.ucr.edu*

ABSTRACT
The parallel explosions of interest in streaming data, and data mining of time series have had surprisingly little intersection. This is in spite of the fact that time series data are typically streaming data. The main reason for this apparent paradox is the fact that the vast majority of work on streaming data explicitly assumes that the data is discrete, whereas the vast majority of time series data is real valued.

Many researchers have also considered transforming real valued time series into symbolic representations, noting that such representations would potentially allow researchers to avail of the wealth of data structures and algorithms from the text processing and bioinformatics communities, in addition to allowing formerly "batch-only" problems to be tackled by the streaming community. While many symbolic representations of time series have been introduced over the past decades, they all suffer from three fatal flaws. Firstly, the dimensionality of the symbolic representation is the same as the original data, and virtually all data mining algorithms scale poorly with dimensionality. Secondly, although distance measures can be defined on the symbolic approaches, these distance measures have little correlation with distance measures defined on the original time series. Finally, most of these symbolic approaches require one to have access to all the data, before creating the symbolic representation. This last feature explicitly thwarts efforts to use the representations with streaming algorithms.

In this work we introduce a new symbolic representation of time series. Our representation is unique in that it allows dimensionality/numerosity reduction, and it also allows distance measures to be defined on the symbolic approach that lower bound corresponding distance measures defined on the original series. As we shall demonstrate, this latter feature is particularly exciting because it allows one to run certain data mining algorithms on the efficiently manipulated symbolic representation, while producing identical results to the algorithms that operate on the original data. Finally, our representation allows the real valued data to be converted in a streaming fashion, with only an infinitesimal time and space overhead.

We will demonstrate the utility of our representation on the classic data mining tasks of clustering, classification, query by content and anomaly detection.

Keywords
Time Series, Data Mining, Data Streams, Symbolic, Discretize

1. INTRODUCTION
The parallel explosions of interest in streaming data [4, 8, 10, 18], and data mining of time series [6, 7, 9, 20, 21, 24, 26, 34] have had surprisingly little intersection. This is in spite of the fact that time series data are typically streaming data, for example, stock value, medical and meteorological data [30]. The main reason for this apparent paradox is the fact that the vast majority of work on streaming data explicitly assumes that the data is discrete, whereas the vast majority of time series data is real valued [23].

Many high level representations of time series have been proposed for data mining. Figure 1 illustrates a hierarchy of all the various time series representations in the literature [2, 7, 14, 16, 20, 22, 25, 30, 31, 35]. One representation that the data mining community has not considered in detail is the discretization of the original data into symbolic strings. At first glance this seems a surprising oversight. In addition to allowing the framing of time series problems as streaming problems, there is an enormous wealth of existing algorithms and data structures that allow the efficient manipulations of symbolic representations. Such algorithms have received decades of attention in the text retrieval community, and more recent attention from the bioinformatics community [3, 13, 17, 29, 32, 33]. Some simple examples of "tools" that are not defined for real-valued sequences but are defined for symbolic approaches include hashing, Markov models, suffix trees, decision trees etc. As a more concrete example, consider the Jaccard coefficient [13], a distance measure beloved by streaming researchers. The Jaccard coefficient is only well defined for discrete data (such as web clicks or individual keystrokes) as thus cannot be used with real-valued time series.

There is a simple explanation for the data mining community's lack of interest in symbolic manipulation as a supporting technique for mining time series. If the data are transformed into virtually any of the other representations depicted in Figure 1, then it is possible to measure the similarity of two time series in that representation space, such that the distance is guaranteed to lower bound the true distance between the time series in the original space. This simple fact is at the core of almost all algorithms in time series data mining and indexing [14]. However, in spite of the fact that there are dozens of techniques for producing different variants of the symbolic representation [2, 11, 20], there is no known method for calculating the distance in the symbolic space, while providing the lower bounding guarantee.

In addition to allowing the creation of lower bounding distance measures, there is one other highly desirable property of any time series representation, including a symbolic one. Almost all time series datasets are very high dimensional. This is a challenging fact because all non-trivial data mining and indexing algorithms degrade exponentially with dimensionality. For example, above 16-20 dimensions, index structures degrade to sequential scanning [19].

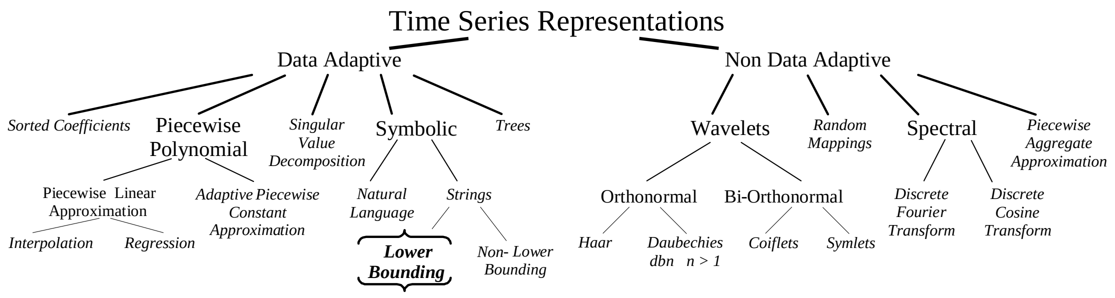


Figure 1: A hierarchy of all the various time series representations in the literature. The leaf nodes refer to the actual representation, and the internal nodes refer to the classification of the approach. The contribution of this paper is to introduce a new representation, the lower bounding symbolic approach

None of the symbolic representations that we are aware of allow dimensionality reduction [2, 11, 20]. There is some reduction in the storage space required, since fewer bits are required for each value; however, the intrinsic dimensionality of the symbolic representation is the same as the original data.

In [4], Babcock et. al ask if "*there is a need for database researchers to develop fundamental and general-purpose models… for data streams.*" The opinion of the authors is affirmative. In this work we take a step towards this goal by introducing a representation of time series that is suitable for streaming algorithms. It is dimensionality reducing, lower bounding and can be obtained in a streaming fashion.

As we shall demonstrate, the lower bounding feature is particularly exciting because it allows one to run certain data mining algorithms on the efficiently manipulated symbolic representation, while producing identical results to the algorithms that operate on the original data. In particular, we will demonstrate the utility of our representation on the classic data mining tasks of clustering [21], classification [16], indexing [1, 14, 22, 35], and anomaly detection [9, 24, 31].

The rest of this paper is organized as follows. Section 2 briefly discusses background material on time series data mining and related work. Section 3 introduces our novel symbolic approach, and discusses its dimensionality reduction, numerosity reduction and lower bounding abilities. Section 4 contains an experimental evaluation of the symbolic approach on a variety of data mining tasks. Finally, Section 5 offers some conclusions and suggestions for future work.

2. BACKGROUND AND RELATED WORK

Time series data mining has attracted enormous attention in the last decade. The review below is necessarily brief; we refer interested readers to [30, 23] for a more in depth review.

2.1 Time Series Data Mining Tasks

While making no pretence to be exhaustive, the following list summarizes the areas that have seen the majority of research interest in time series data mining.

- **Indexing:** Given a query time series Q, and some similarity/dissimilarity measure *D*(Q,C), find the most similar time series in database *DB* [1, 7, 14,22, 35].
- **Clustering:** Find natural groupings of the time series in database *DB* under some similarity/dissimilarity measure *D*(Q,C) [21,25].
- **Classification:** Given an unlabeled time series Q, assign it to one of two or more predefined classes [16].
- **Summarization:** Given a time series Q containing *n* datapoints where *n* is an extremely large number, create a (possibly graphic) approximation of Q which retains its essential features but fits on a single page, computer screen, executive summary, etc [26].
- **Anomaly Detection:** Given a time series Q, and some model of "normal" behavior, find all sections of Q which contain anomalies, or "surprising/interesting/unexpected/novel" behavior [9, 24, 31].

Since the datasets encountered by data miners typically don't fit in main memory, and disk I/O tends to be the bottleneck for any data mining task, a simple generic framework for time series data mining has emerged [14]. The basic approach is outlined in Table 1.

1.	Create an approximation of the data, which will fit in main memory, yet retains the essential features of interest.
2.	Approximately solve the task at hand in main memory.
3.	Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data.

Table 1: A generic time series data mining approach

It should be clear that the utility of this framework depends heavily on the quality of the approximation created in Step 1. If the approximation is very faithful to the original data, then the solution obtained in main memory is likely to be the same as, or very close to, the solution we would have obtained on the original data. The handful of disk accesses made in Step 3 to confirm or slightly modify the solution will be inconsequential compared to the number of disk accesses required had we worked on the original data. With this in mind, there has been great interest in approximate representations of time series, which we consider below.

2.2 Time Series Representations

As with most problems in computer science, the suitable choice of representation greatly affects the ease and efficiency of time series data mining. With this in mind, a great number of time series representations have been introduced, including the Discrete Fourier Transform (DFT) [14], the Discrete Wavelet Transform (DWT) [7], Piecewise Linear, and Piecewise Constant models (PAA) [22], (APCA) [16, 22], and Singular Value Decomposition (SVD) [22]. Figure 2 illustrates the most commonly used representations.

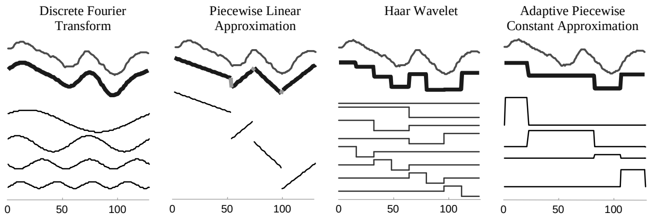


Figure 2: The most common representations for time series data mining. Each can be visualized as an attempt to approximate the signal with a linear combination of basis functions

Recent work suggests that there is little to choose between the above in terms of indexing power [23]; however, the representations have other features that may act as strengths or weaknesses. As a simple example, wavelets have the useful multiresolution property, but are only defined for time series that are an integer power of two in length [7].

One important feature of all the above representations is that they are real valued. This limits the algorithms, data structures and definitions available for them. For example, in anomaly detection we cannot meaningfully define the probability of observing any particular set of wavelet coefficients, since the probability of observing any real number is zero [27]. Such limitations have lead researchers to consider using a symbolic representation of time series.

While there are literally hundreds of papers on discretizing (symbolizing, tokenizing, quantizing) time series [2, 20] (see [11] for an extensive survey), none of the techniques allows a distance measure that lower bounds a distance measure defined on the original time series. For this reason, the generic time series data mining approach illustrated in Table 1 is of little utility, since the approximate solution to problem created in main memory may be arbitrarily dissimilar to the true solution that would have been obtained on the original data. If, however, one had a symbolic approach that allowed lower bounding of the true distance, one could take advantage of the generic time series data mining model, and of a host of other algorithms, definitions and data structures which are only defined for discrete data, including hashing, Markov models, and suffix trees. This is exactly the contribution of this paper. We call our symbolic representation of time series SAX (Symbolic Aggregate approXimation), and define it in the next section.

3. SAX: OUR SYMBOLIC APPROACH

SAX allows a time series of arbitrary length n to be reduced to a string of arbitrary length w, ($w < n$, typically $w << n$). The alphabet size is also an arbitrary integer a, where $a > 2$. Table 2 summarizes the major notation used in this and subsequent sections.

C	A time series $C = c_1,\ldots,c_n$
$\overline{C}$	A Piecewise Aggregate Approximation of a time series $\overline{C} = \overline{c}_1,\ldots,\overline{c}_w$
$\hat{C}$	A symbol representation of a time series $\hat{C} = \hat{c}_1,\ldots,\hat{c}_w$
w	The number of PAA segments representing time series C
a	Alphabet size (e.g., for the alphabet = {**a**,**b**,**c**}, a = 3)

Table 2: A summarization of the notation used in this paper

Our discretization procedure is unique in that it uses an intermediate representation between the raw time series and the symbolic strings. We first transform the data into the Piecewise Aggregate Approximation (PAA) representation and then symbolize the PAA representation into a discrete string. There are two important advantages to doing this:

- **Dimensionality Reduction:** We can use the well-defined and well-documented dimensionality reduction power of PAA [22, 35], and the reduction is automatically carried over to the symbolic representation.
- **Lower Bounding:** Proving that a distance measure between two symbolic strings lower bounds the true distance between the original time series is non-trivial. The key observation that allows us to prove lower bounds is to concentrate on proving that the symbolic distance measure lower bounds the *PAA distance measure*. Then we can prove the desired result by transitivity by simply pointing to the existing proofs for the PAA representation itself [35].

We will briefly review the PAA technique before considering the symbolic extension.

3.1 Dimensionality Reduction Via PAA

A time series C of length n can be represented in a w-dimensional space by a vector $\overline{C} = \overline{c}_1,\ldots,\overline{c}_w$. The i^{th} element of $\overline{C}$ is calculated by the following equation:

$$\overline{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j \qquad (1)$$

Simply stated, to reduce the time series from n dimensions to w dimensions, the data is divided into w equal sized "frames." The mean value of the data falling within a frame is calculated and a vector of these values becomes the data-reduced representation. The representation can be visualized as an attempt to approximate the original time series with a linear combination of box basis functions as shown in Figure 3.

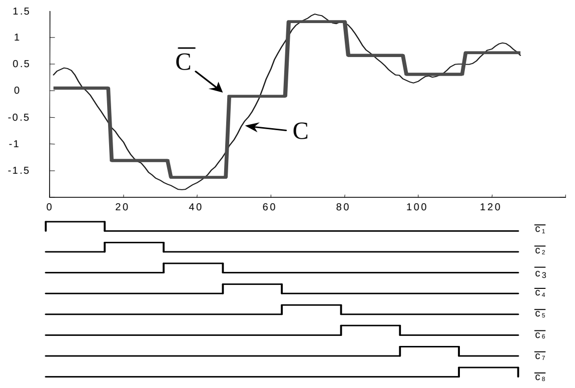


Figure 3: The PAA representation can be visualized as an attempt to model a time series with a linear combination of box basis functions. In this case, a sequence of length 128 is reduced to 8 dimensions

The PAA dimensionality reduction is intuitive and simple, yet has been shown to rival more sophisticated dimensionality reduction techniques like Fourier transforms and wavelets [22, 23, 35].

We normalize each time series to have a mean of zero and a standard deviation of one before converting it to the PAA representation, since it is well understood that it is meaningless to compare time series with different offsets and amplitudes [23].

3.2 Discretization

Having transformed a time series database into PAA, we can apply a further transformation to obtain a discrete representation. It is desirable to have a discretization technique that will produce symbols with equiprobability [3, 28]. This is easily achieved since normalized time series have a Gaussian distribution [27]. To illustrate this, we extracted subsequences of length 128 from 8 different time series and plotted a normal probability plot of the data as shown in Figure 4.

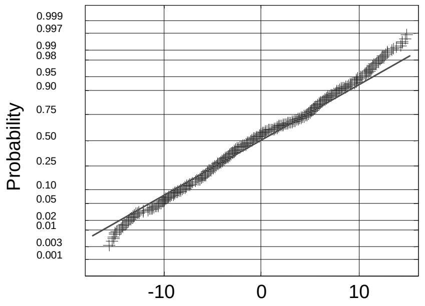


Figure 4: A normal probability plot of the cumulative distribution of values from subsequences of length 128 from 8 different datasets. The highly linear nature of the plot strongly suggests that the data came from a Gaussian distribution

Given that the normalized time series have highly Gaussian distribution, we can simply determine the "breakpoints" that will produce *a* equal-sized areas under Gaussian curve [27].

Definition 1. *Breakpoints*: breakpoints are a sorted list of numbers $B = \beta_1,\ldots,\beta_{a-1}$ such that the area under a $N(0,1)$ Gaussian curve from β_i to $\beta_{i+1} = 1/a$ (β_0 and β_a are defined as $-\infty$ and ∞, respectively).

These breakpoints may be determined by looking them up in a statistical table. For example, Table 3 gives the breakpoints for values of *a* from 3 to 10.

β_i \ *a*	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Table 3: A lookup table that contains the breakpoints that divide a Gaussian distribution in an arbitrary number (from 3 to 10) of equiprobable regions

Once the breakpoints have been obtained we can discretize a time series in the following manner. We first obtain a PAA of the time series. All PAA coefficients that are below the smallest breakpoint are mapped to the symbol "**a**," all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol "**b**," etc. Figure 5 illustrates the idea.

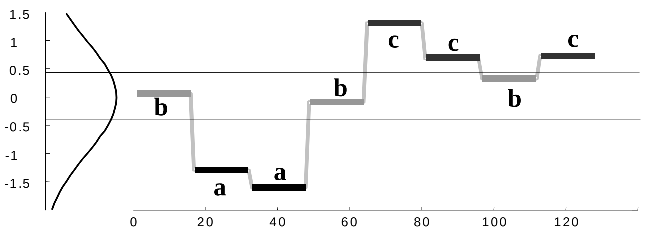


Figure 5: A time series is discretized by first obtaining a PAA approximation and then using predetermined breakpoints to map the PAA coefficients into SAX symbols. In the example above, with n = 128, w = 8 and a = 3, the time series is mapped to the word **baabccbc**

Note that in this example the 3 symbols, "**a**," "**b**," and "**c**" are approximately equiprobable as we desired. We call the concatenation of symbols that represent a subsequence a *word*.

Definition 2. *Word*: A subsequence C of length n can be represented as a *word* $\hat{C} = \hat{c}_1,\ldots,\hat{c}_w$ as follows. Let alpha$_i$ denote the ith element of the alphabet, i.e., alpha$_1$ = **a** and alpha$_2$ = **b**. Then the mapping from a PAA approximation $\overline{C}$ to a word $\hat{C}$ is obtained as follows:

$$\hat{c}_i = \quad alpha_j\ , \quad iif \quad \beta_{j-1} \le \overline{c}_i < \beta_j \qquad (2)$$

We have now defined SAX, our symbolic representation (the PAA representation is merely an intermediate step required to obtain the symbolic representation).

3.3 Distance Measures

Having introduced the new representation of time series, we can now define a distance measure on it. By far the most common distance measure for time series is the Euclidean distance [23, 29]. Given two time series Q and C of the same length n, Eq. 3 defines their Euclidean distance, and Figure 6.A illustrates a visual intuition of the measure.

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}\left(q_i - c_i\right)^2} \qquad (3)$$

If we transform the original subsequences into PAA representations, $\overline{Q}$ and $\overline{C}$, using Eq. 1, we can then obtain a lower bounding approximation of the Euclidean distance between the original subsequences by:

$$DR(\overline{Q},\overline{C}) \equiv \sqrt{\tfrac{n}{w}}\sqrt{\sum\nolimits_{i=1}^{w}\left(\overline{q}_i - \overline{c}_i\right)^2} \qquad (4)$$

This measure is illustrated in Figure 6.B. A proof that $DR(\overline{Q},\overline{C})$ lower bounds the true Euclidean distance appears in [22] (an alterative proof appears in [35]).

If we further transform the data into the symbolic representation, we can define a MINDIST function that returns the minimum distance between the original time series of two words:

$$MINDIST(\hat{Q},\hat{C}) \equiv \sqrt{\tfrac{n}{w}}\sqrt{\sum\nolimits_{i=1}^{w}\left(dist(\hat{q}_i,\hat{c}_i)\right)^2} \qquad (5)$$

The function resembles Eq. 4 except for the fact that the distance between the two PAA coefficients has been replaced with the sub-function *dist*(). The *dist*() function can be implemented using a table lookup as illustrated in Table 4.

	a	**b**	**c**	**d**
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

Table 4: A lookup table used by the MINDIST function. This table is for an alphabet of cardinality of 4, i.e. *a=4*. The distance between two symbols can be read off by examining the corresponding row and column. For example, *dist*(**a**,**b**) = 0 and *dist*(**a**,**c**) = 0.67.

The value in cell (*r*,*c*) for any lookup table can be calculated by the following expression.

$$cell_{r,c} = \begin{cases} 0, & if \ |r-c| \le 1 \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)}, & otherwise \end{cases} \tag{6}$$

For a given value of the alphabet size *a*, the table needs only be calculated once, then stored for fast lookup. The MINDIST function can be visualized in Figure 6.C.

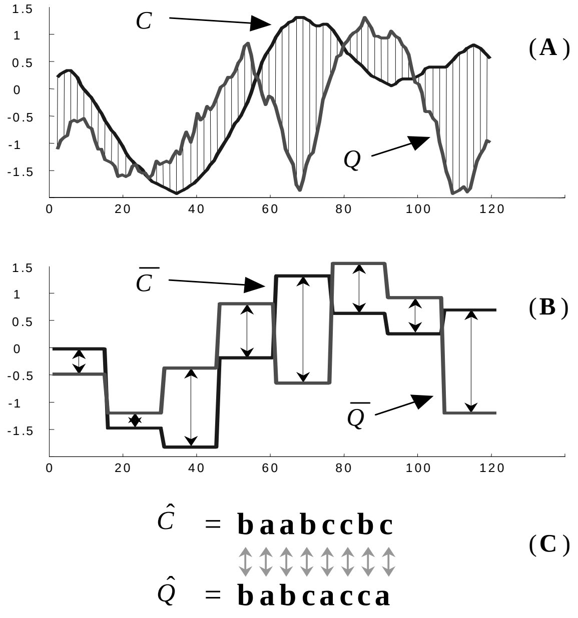


Figure 6: A visual intuition of the three representations discussed in this work, and the distance measures defined on them. **A**) The Euclidean distance between two time series can be visualized as the square root of the sum of the squared differences of each pair of corresponding points. **B**) The distance measure defined for the PAA approximation can be seen as the square root of the sum of the squared differences between each pair of corresponding PAA coefficients, multiplied by the square root of the compression rate. **C**) The distance between two SAX representations of a time series requires looking up the distances between each pair of symbols, squaring them, summing them, taking the square root and finally multiplying by the square root of the compression rate

There is one issue we must address if we are to use a symbolic representation of time series. If we wish to approximate a massive dataset in main memory, the parameters *w* and *a* have to be chosen in such a way that the approximation makes the best use of the primary memory available. There is a clear tradeoff between the parameter *w* controlling the number of approximating elements, and the value *a* controlling the granularity of each approximating element.

It is infeasible to determine the best tradeoff analytically, since it is highly data dependent. We can, however, empirically determine the best values with a simple experiment. Since we wish to achieve the tightest possible lower bounds, we can simply estimate the lower bounds over all possible feasible parameters, and choose the best settings.

$$Tightness\ of\ Lower\ Bound = \frac{MINDIST(\hat{Q},\hat{C})}{D(Q,C)} \tag{7}$$

We performed such a test with a concatenation of 50 time series databases taken from the UCR time series data mining archive. For every combination of parameters we averaged the result of 100,000 experiments on subsequences of length 256. Figure 7 shows the results.

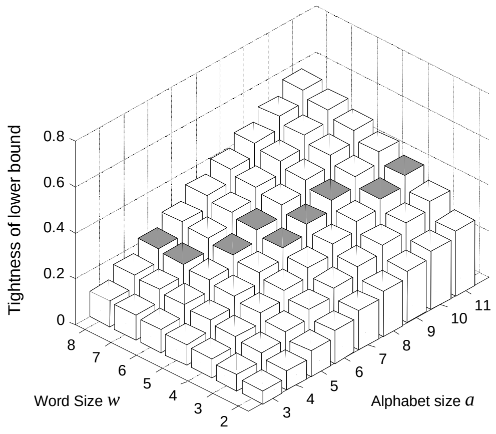


Figure 7: The empirically estimated tightness of lower bounds over the cross product of *a* = [3…11] and *w* = [2…8]. The darker histogram bars illustrate combinations of parameters that require approximately equal space to store every possible word (approximately 4 megabytes)

The results suggest that using a low value for *a* results in weak bounds, but that there are diminishing returns for large values of *a*. The results also suggest that the parameters are not too critical; an alphabet size in the range of 5 to 8 seems to be a good choice.

3.4 Numerosity Reduction

We have seen that, given a single time series, our approach can significantly reduce its dimensionality. In addition, our approach can reduce the numerosity of the data for some applications.

Most applications assume that we have one very long time series *T*, and that manageable *subsequences* of length *n* are extracted by use of a sliding window, then stored in a matrix for further manipulation [7, 14, 22, 35]. Figure 8 illustrates the idea.

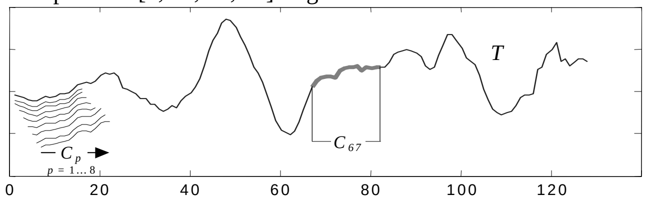


Figure 8: An illustration of the notation introduced in this section: A time series *T* of length 128, the *subsequence* C_{67} of length *n* = 16, and the first 8 subsequences extracted by a *sliding window*. Note that the sliding windows are overlapping

When performing sliding windows subsequence extraction, with any of the real-valued representations, we must store all |T| - n + 1 extracted subsequences (in dimensionality reduced form). However, imagine for a moment that we are using our proposed approach. If the first word extracted is **aabbcc**, and the window is shifted to discover that the second word is also **aabbcc**, we can reasonably decide not to include the second occurrence of the word in sliding windows matrix. If we ever need to retrieve all occurrences of **aabbcc**, we can go to the location pointed to by the first occurrences, and remember to slide to the right, testing to see if the next window is also mapped to the same word. We can stop testing as soon as the word changes. This simple idea is very similar to the run-length-encoding data compression algorithm.

The utility of this optimization depends on the parameters used and the data itself, but it typically yields a numerosity reduction factor of two or three. However, many datasets are characterized by long periods of little or no movement, followed by bursts of activity (seismological data is an obvious example). On these datasets the numerosity reduction factor can be huge. Consider the example shown in Figure 9.

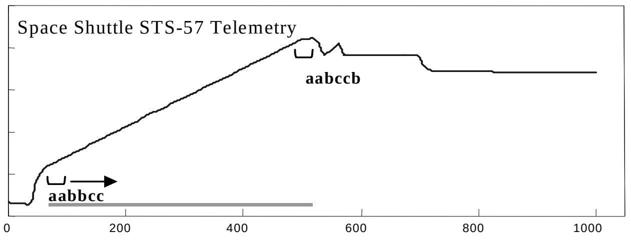


Figure 9: Sliding window extraction on Space Shuttle Telemetry data, with n = 32. At time point 61, the extracted word is **aabbcc**, and the next 401 subsequences also map to this word. Only a pointer to the first occurrence must be recorded, thus producing a large reduction in numerosity

There is only one special case we must consider. As we noted in Section 3.1, we normalize each time series (including subsequences) to have a mean of zero and a standard deviation of one. However, if the subsequence contains only one value, the standard deviation is not defined. More troublesome is the case where the subsequence is *almost* constant, perhaps 31 zeros and a single 0.0001. If we normalize this subsequence, the single differing element will have its value exploded to 5.48. This situation occurs quite frequently. For example, the last 200 time units of the data in Figure 9 appear to be constant, but actually contain tiny amounts of noise. If we were to normalize subsequences extracted from this area, the normalization would magnify the noise to large meaningless patterns.

We can easily deal with this problem, if the standard deviation of the sequence before normalization is below an epsilon ε, we simply assign the entire word to the middle-ranged alphabet (e.g. **cccccc** if a = 5).

We end this section with a visual comparison between SAX and the four most used representations in the literature (Figure 10).

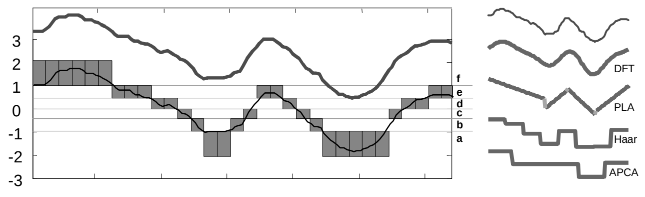


Figure 10: A visual comparison of SAX and the four most common time series data mining representations. A raw time series of length 128 is transformed into the word **ffffffeeeddcbaabceedcbaaaaacddee**. This is a fair comparison since the number of bits in each representation is the same

4. EXPERIMENTAL VALIDATION OF SAX

We performed various data mining tasks using SAX. For clustering and classification, we compared the results with the classic Euclidean distance, and with other previously proposed symbolic approaches. Note that none of these other approaches use dimensionality reduction. In the next paragraph we summarize the strawmen representations that we compare SAX to. We choose these two approaches since they are typical representatives of symbolic approaches in the literature.

André-Jönsson, and Badal [2] proposed the SDA algorithm that computes the changes between values from one instance to the next, and divide the range into user-predefined regions. The disadvantages of this approach are obvious: prior knowledge of the data distribution of the time series is required in order to set the breakpoints; and the discretized time series does not conserve the general shape or distribution of the data values.

Huang and Yu proposed the IMPACTS algorithm, which uses change ratio between one time point to the next time point to discretize the time series [20]. The range of change ratios are then divided into equal-sized sections and mapped into symbols. The time series is converted to a discretized collection of change ratios. As with SAX, the user needs to define the cardinality of symbols.

4.1 Clustering

Clustering is one of the most common data mining tasks, being useful in its own right as an exploratory tool, and also as a sub-routine in more complex algorithms [12,15, 21].

4.1.1 Hierarchical Clustering

Comparing hierarchical clusterings is a very good way to compare and contrast similarity measures, since a dendrogram of size N summarizes O(N^2) distance calculations [23]. The evaluation is typically subjective; we simply adjudge which distance measure appears to create the most natural groupings of the data. However, if we know the data labels in advance we can also make objective statements of the quality of the clustering. In Figure 11 we clustered nine time series from the Control Chart dataset, three each from the *decreasing trend*, *upward shift* and *normal* classes.

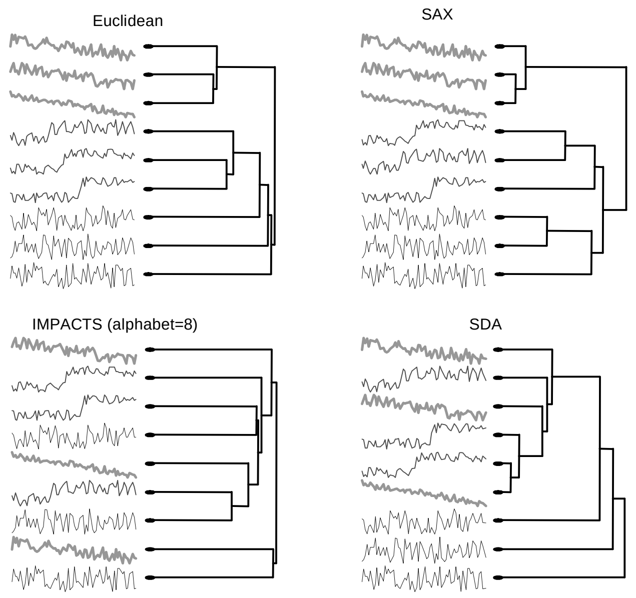


Figure 11: A comparison of the four distance measures' ability to cluster members of the Control Chart dataset. Complete linkage was used as the agglomeration technique

In this case we can objectively state that SAX is superior, since it correctly assigns each class to its own subtree. This is simply a side effect due to the smoothing effect of dimensionality reduction. More generally, we observed that SAX closely mimics Euclidean distance on various datasets.

4.1.2 Partitional Clustering

Although hierarchical clustering is a good sanity check for any proposed distance measure, it has limited utility for data mining because of its poor scalability. The most commonly used data mining clustering algorithm is *k*-means [15], so for completeness we will consider it here. We performed k-means on both the original raw data, and our symbolic representation. Figure 12 shows a typical run of *k*-means on a space telemetry dataset. Both algorithms converge after 11 iterations on average.

The results here are quite unintuitive and surprising; working with an approximation of the data gives better results than working with the original data. Fortunately, a recent paper offers a suggestion as to why this might be so. It has been shown that initializing the clusters centers on a low dimension approximation of the data can improve the quality [12], this is what clustering with SAX implicitly does.

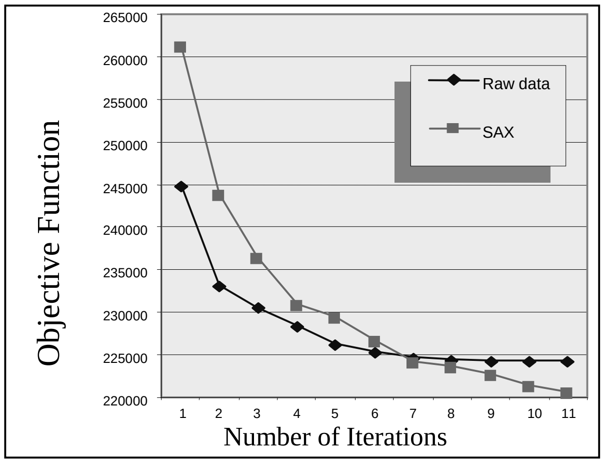


Figure 12: A comparison of the *k*-means clustering algorithm using SAX and the raw data. The dataset was Space Shuttle telemetry, 1,000 subsequences of length 512. Surprisingly, working with the symbolic approximation produces better results than working with the original data

4.2 Classification

Classification of time series has attracted much interest from the data mining community. The high dimensionality, high feature correlation, and typically high levels of noise found in time series provide an interesting research problem [23]. Although special-purpose algorithms have been proposed [25], we will consider only the two most common classification algorithms for brevity, clarity of presentations and to facilitate independent confirmation of our findings.

4.2.1 Nearest Neighbor Classification

To compare different distance measures on 1-nearest-neighbor classification, we used leaving-one-out cross validation. We compare SAX with Euclidean distance, IMPACTS, SDA, and LP_{∞}. Two classic synthetic datasets are used: the Cylinder-Bell-Funnel (CBF) dataset has 50 instances of time series for each of the three clusters, and the Control Chart (CC) has 100 instances for each of the six clusters [23].

Since SAX allows dimensionality and alphabet size as user input, and the IMPACTS allows variable alphabet size, we ran the experiments on different combinations of dimensionality reduction and alphabet size. For the other approaches we applied the simple dimensionality reduction technique of skipping data points at a fixed interval. In Figure 13, we show the results with a dimensionality reduction of 4 to 1.

Similar results were observed for other levels of dimensionality reduction. Once again, SAX's ability to beat Euclidean distance is probably due to the smoothing effect of dimensionality reduction; nevertheless, this experiment does show the superiority of SAX over the other approaches proposed in the literature.

4.2.2 Decision Tree Classification

Due to its poor scalability, Nearest Neighbor is unsuitable for most data mining applications; instead, decision trees are the most common choice of classifier. While decision trees are defined for real data, attempting to classify time series using the raw data would clearly be a mistake, since the high dimensionality and noise levels would result in a deep, bushy tree with poor accuracy.

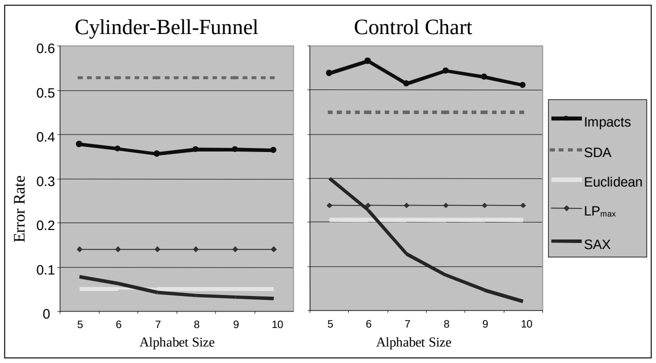


Figure 13: A comparison of five distance measures utility for nearest neighbor classification. We tested different alphabet sizes for SAX and IMPACTS. SDA's alphabet size is fixed at 5.

In an attempt to overcome this problem, Geurts [16] suggests representing the time series as a Regression Tree (RT) (this representation is essentially the same as APCA [22], see Figure 2), and training the decision tree directly on this representation. The technique shows great promise.

We compared SAX to the Regression Tree (RT) on two datasets; the results are in Table 5.

Dataset	SAX	Regression Tree
CC	3.04 ± 1.64	2.78 ± 2.11
CBF	0.97 ± 1.41	1.14 ± 1.02

Table 5: A comparison of SAX with the specialized Regression Tree approach for decision tree classification. Our approach used an alphabet size of 6; both approaches used a dimensionality of 8

Note that while our results are competitive with the RT approach, The RT representation is undoubtedly superior in terms of interpretability [16]. Once again, our point is simply that our "black box" approach can be competitive with specialized solutions.

4.3 Query by Content (Indexing)

The majority of work on time series data mining appearing in the literature has addressed the problem of indexing time series for fast retrieval [30]. Indeed, it is in this context that most of the representations enumerated in Figure 1 were introduced [7, 14, 22, 35]. Dozens of papers have introduced techniques to do indexing with a symbolic approach [2, 20], but without exception, the answer set retrieved by these techniques can be very different to the answer set that would be retrieved by the true Euclidean distance. It is only by using a lower bounding technique that one can guarantee retrieving the full answer set, with no false dismissals [14].

To perform query by content, we built an index using SAX, and compared it to an index built using the Haar wavelet approach [7]. Since the datasets we use are large and disk-resident, and the reduced dimensionality could still be potentially high (or at least high enough such that the performance degenerates to sequential scan if R-tree were used [19]), we use Vector Approximation (VA) file as our indexing algorithm. We note, however, that SAX could also be indexed by classic string indexing techniques such as suffix trees.

To compare performance, we measure the percentage of disk I/Os required in order to retrieve the one-nearest neighbor to a randomly extracted query, relative to the number of disk I/Os required for sequential scan. Since it has been forcibly shown that the choice of dataset can make a significant difference in the relative indexing ability of a representation, we tested on more than 50 datasets from the UCR Time Series Data Mining Archive. In Figure 14 we show 4 representative examples.

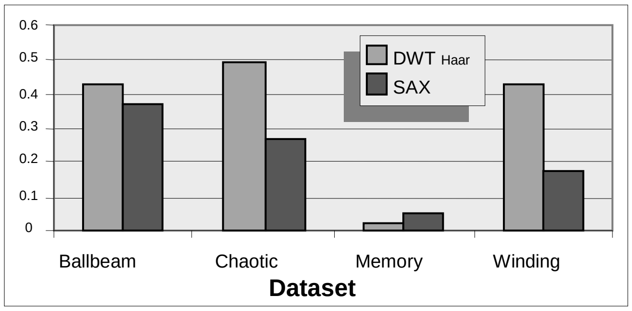


Figure 14: A comparison of indexing ability of wavelets versus SAX. The Y-axis is the percentage of the data that must be retrieved from disk to answer a 1-NN query of length 256, when the dimensionality reduction ratio is 32 to 1 for both approaches

Once again we find our representation competitive with existing approaches.

4.4 Taking Advantage of the Discrete Nature of our Representation

In the previous sections we showed examples of how our proposed representation can compete with real-valued representations and the original data. In this section we illustrate examples of data mining algorithms that take explicit advantage of the discrete nature of our representation.

4.4.1 Detecting Novel/Surprising/Anomalous Behavior

A simple idea for detecting anomalous behavior in time series is to examine previously observed normal data and build a model of it. Data obtained in the future can be compared to this model and any lack of conformity can signal an anomaly [9]. In order to achieve this, in [24] we combined a statistically sound scheme with an efficient combinatorial approach. The statistical scheme is based on Markov chains and normalization. Markov chains are used to model the "normal" behavior, which is inferred from the previously observed data. The time- and space-efficiency of the algorithm comes from the use of suffix tree as the main data structure. Each node of the suffix tree represents a pattern. The tree is annotated with a score obtained comparing the support of a pattern observed in the new data with the support recorded in the Markov model. This apparently simple strategy turns out to be very effective in discovering surprising patterns. In the original work we use a simple symbolic approach, similar to IMPACTS [20]; here we revisit the work using SAX.

For completeness, we will compare SAX to two highly referenced anomaly detection algorithms that are defined on real valued representations, the TSA-tree Wavelet based approach of Shahabi et al. [31] and the Immunology (IMM) inspired work of Dasgupta and Forrest [9]. We also include the Markov technique using IMPACTS and SDA in order to discover how much of the difference can be attributed directly to the representation. Figure 15 contains an experiment comparing all 5 techniques.

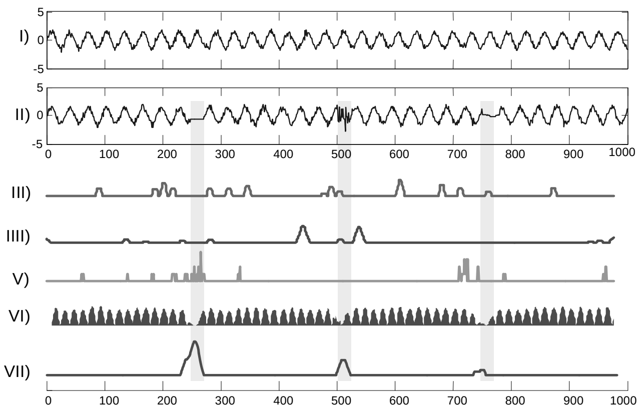


Figure 15: A comparison of five anomaly detection algorithms on the same task. **I)** The training data, a slightly noisy sine wave of length 1,000. **II)** The time series to be examined for anomalies is a noisy sine wave that was created with the same parameters as the training sequence, then an assortment of anomalies were introduced at time periods 250, 500 and 750. **III)** and **IIII)** The Markov Model technique using the IMPACTS and SDA representation did not clearly discover the anomalies, and reported some false alarms. **V)** The IMM anomaly detection algorithm appears to have discovered the first anomaly, but it also reported many false alarms. **VI)** The TSA-Tree approach is unable to detect the anomalies. **VII)** The Markov model-based technique using SAX clearly finds the anomalies, with no false alarms

The results on this simple experiment are impressive. Since suffix trees and Markov models can be used only on discrete data, this offers a motivation for our symbolic approach.

4.4.2 Motif discovery

It is well understood in bioinformatics that overrepresented DNA sequences often have biological significance [3, 13, 29]. A substantial body of literature has been devoted to techniques to discover such patterns [17, 32, 33]. In a previous work, we defined the related concept of "time series motif" [26]. Time series motifs are close analogues of their discrete cousins, although the definitions must be augmented to prevent certain degenerate solutions. The naïve algorithm to discover the motifs is quadratic in the length of the time series. In [26], we demonstrated a simple technique to mitigate the quadratic complexity by a large constant factor; nevertheless, this time complexity is clearly untenable for most real datasets.

The symbolic nature of SAX offers a unique opportunity to avail of the wealth of bioinformatics research in this area. In particular, recent work by Tompa and Buhler holds great promise [33]. The authors show that many previously unsolvable motif discovery problems can be solved by hashing subsequences into buckets using a random subset of their features as a key, then doing some post-processing search on the hash buckets[1]. They call their algorithm PROJECTION.

We carefully reimplemented the random projection algorithm of Tompa and Buhler, making minor changes in the post-processing step to allow for the fact that although we are hashing random projections of our symbolic representation, we actually wish to discover motifs defined on the original raw data. Figure 16 shows an example of a motif discovered in an industrial dataset [5] using this technique.

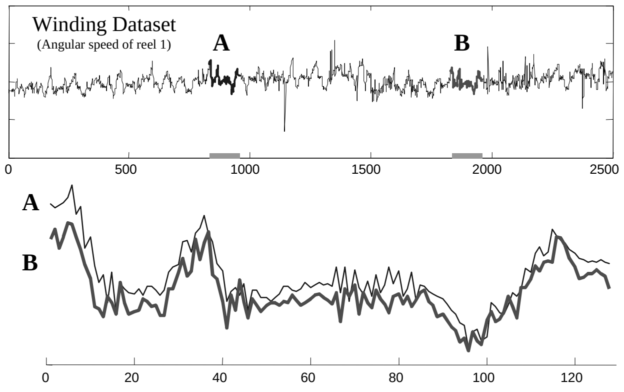


Figure 16: Above, a motif discovered in a complex dataset by the modified PROJECTION algorithm. Below, the motif is best visualized by aligning the two subsequences and "zooming in". The similarity of the two subsequences is striking, and hints at unexpected regularity

Apart from the attractive scalability of the algorithm, there is another important advantage over other approaches. The PROJECTION algorithm is able to discover motifs even in the presence of noise. Our extension of the algorithm inherits this robustness to noise.

5. CONCLUSIONS AND FUTURE DIRECTIONS

In this work we introduced the first dimensionality reduction, lower bounding, streaming symbolic approach in the literature. We have shown that our representation is competitive with, or superior to, other representations on a wide variety of classic data mining problems, and that its discrete nature allows us to tackle emerging tasks such as anomaly detection and motif discovery.

A host of future directions suggest themselves. In addition to use with streaming algorithms, there is an enormous wealth of useful definitions, algorithms and data structures in the bioinformatics literature that can be exploited by our representation [3, 13, 17, 28, 29, 32, 33]. It may be possible to create a lower bounding approximation of Dynamic Time Warping [6], by slightly modifying the classic string edit distance. Finally, there may be utility in extending our work to multidimensional time series [34].

6. REFERENCES

[1] Agrawal, R., Psaila, G., Wimmers, E. L. & Zait, M. (1995). Querying Shapes of Histories. In *proceedings of the 21st Int'l Conference on Very Large Databases*. Zurich, Switzerland, Sept 11-15. pp 502-514.

[2] André-Jönsson, H. & Badal. D. (1997). Using Signature Files for Querying Time-Series Data. In *proceedings of Principles of Data Mining and Knowledge Discovery*, 1st European Symposium. Trondheim, Norway, Jun 24-27. pp 211-220.

[3] Apostolico, A., Bock, M. E. & Lonardi, S. (2002). Monotony of Surprise and Large-Scale Quest for Unusual Words. In *proceedings of the 6th Int'l Conference on Research in Computational Molecular Biology*. Washington, DC, April 18-21. pp 22-31.

[4] Babcock, B, Babu, S., Datar, M., Motwani, R. & Widom, J. (2002). Models and Issues in Data Stream Systems. *Invited Paper in proceedings of the 2002 ACM Symp. On Principles of Database Systems*. June 3-5, Madison, WI.

[1] Of course, this description greatly understates the contributions of this work. We urge the reader to consult the original paper.

[5] Bastogne, T., Noura, H., Richard A. & Hittinger, J. .M. (2002). Application of Subspace Methods to the Identification of a Winding Process. In *proceedings of the 4th European Control Conference*, Vol. 5, Brussels.

[6] Berndt, D. & Clifford, J. (1994) Using Dynamic Time Warping to Find Patterns in Time Series. In *proceedings of the Workshop on Knowledge Discovery in Databases*, at the 12th Int'l Conference on Artificial Intelligence. July 31-Aug 4, Seattle, WA. pp 229-248.

[7] Chan, K. & Fu, A. W. (1999). Efficient Time Series Matching by Wavelets. In *proceedings of the 15th IEEE Int'l Conference on Data Engineering*. Sydney, Australia, Mar 23-26. pp 126-133.

[8] Cortes, C., Fisher, K., Pregibon, D., Rogers, A. & Smith, F. (2000). Hancock: a Language for Extracting Signatures from Data Streams. In *proceedings of the 6th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*. Aug 20-23, Boston, MA. pp 9-17.

[9] Dasgupta, D. & Forrest, S. (1996) Novelty Detection in Time Series Data using Ideas from Immunology. In *proceedings of The International Conference on Intelligent Systems*. June 19-21.

[10] Datar, M. & Muthukrishnan, S. (2002). Estimating Rarity and Similarity over Data Stream Windows. In *proceedings of the 10th European Symposium on Algorithms*. Sep 17-21, Rome, Italy.

[11] Daw, C. S., Finney, C. E. A. & Tracy, E. R. (2001). Symbolic Analysis of Experimental Data. *Review of Scientific Instruments*. (2002-07-22).

[12] Ding, C., He, X., Zha, & Simon., H. (2002). Adaptive Dimension Reduction for Clustering High Dimensional Data. In *proceedings of the 2nd IEEE International Conference on Data Mining*. Dec 9-12. Maebashi, Japan. pp 147-154.

[13] Durbin, R., Eddy, S., Krogh, A. & Mitchison, G. (1998). Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press.

[14] Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast Subsequence Matching in Time-Series Databases. *In proceedings of the ACM SIGMOD Int'l Conference on Management of Data*. May 24-27, Minneapolis, MN. pp 419-429.

[15] Fayyad, U., Reina, C. &. Bradley, P. (1998). Initialization of Iterative Refinement Clustering Algorithms. In *proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*. New York, NY, Aug 27-31. pp 194-198.

[16] Geurts, P. (2001). Pattern Extraction for Time Series Classification. In *proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*. Sep 3-7, Freiburg, Germany. pp. 115-127.

[17] Gionis, A. & Mannila, H. (2003). Finding Recurrent Sources in Sequences. In *proceedings of the 7th International Conference on Research in Computational Molecular Biology*. Apr 10-13, Berlin, Germany. To Appear.

[18] Guha, S., Mishra, N., Motwani, R. & O'Callaghan, L. (2000). Clustering Data Streams. In *proceedings of the 41st Symposium on Foundations of Computer Science*. Nov 12-14, Redondo Beach, CA. pp 359-366.

[19] Hellerstein, J. M., Papadimitriou, C. H. & Koutsoupias, E. (1997). Towards an Analysis of Indexing Schemes. In proceedings of the *16th ACM Symposium on Principles of Database Systems*. May 12-14, Tucson, AZ. pp 249-256.

[20] Huang, Y. & Yu, P. S. (1999). Adaptive Query Processing for Time-Series Data. In *proceedings of the 5th Int'l Conference on Knowledge Discovery and Data Mining*. San Diego, CA, Aug 15-18. pp 282-286.

[21] Kalpakis, K., Gada, D. & Puttagunta, V. (2001). Distance Measures for Effective Clustering of ARIMA Time-Series. In *proceedings of the 2001 IEEE International Conference on Data Mining*, San Jose, CA, Nov 29-Dec 2. pp 273-280.

[22] Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra, S. (2001). Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In *proceedings of ACM SIGMOD Conference on Management of Data*. Santa Barbara, CA, May 21-24. pp 151-162.

[23] Keogh, E. & Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *In proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. July 23 - 26, 2002. Edmonton, Alberta, Canada. pp 102-111.

[24] Keogh, E., Lonardi, S. & Chiu, W. (2002). Finding Surprising Patterns in a Time Series Database in Linear Time and Space. In the *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. July 23 - 26, 2002. Edmonton, Alberta, Canada. pp 550-556.

[25] Keogh, E. & Pazzani, M. (1998). An Enhanced Representation of Time Series Which Allows Fast and Accurate Classification, Clustering and Relevance Feedback. In *proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining*. New York, NY, Aug 27-31. pp 239-241.

[26] Lin, J., Keogh, E., Lonardi, S. & Patel, P. (2002). Finding Motifs in Time Series. In proceedings of the *2nd Workshop on Temporal Data Mining*, at the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining. Edmonton, Alberta, Canada, July 23-26. pp. 53-68.

[27] Larsen, R. J. & Marx, M. L. (1986). An Introduction to Mathematical Statistics and Its Applications. Prentice Hall, Englewood, Cliffs, N.J. 2nd Edition.

[28] Lonardi, S. (2001). Global Detectors of Unusual Words: Design, Implementation, and Applications to Pattern Discovery in Biosequences. PhD thesis, Department of Computer Sciences, Purdue University, August, 2001.

[29] Reinert, G., Schbath, S. & Waterman, M. S. (2000). Probabilistic and Statistical Properties of Words: An Overview. *Journal of Computational. Bio*logy. Vol. 7, pp 1-46.

[30] Roddick, J. F., Hornsby, K. & Spiliopoulou, M. (2001). An Updated Bibliography of Temporal, Spatial and Spatio-Temporal Data Mining Research. In *Post-Workshop Proceedings of the International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining*. Berlin, Springer. Lecture Notes in Artificial Intelligence. Roddick, J. F. and Hornsby, K., Eds. 147-163.

[31] Shahabi, C., Tian, X. & Zhao, W. (2000). TSA-tree: A Wavelet-Based Approach to Improve the Efficiency of Multi-Level Surprise and Trend Queries In *proceedings of the 12th Int'l Conference on Scientific and Statistical Database Management*. pp 55-68.

[32] Staden, R. (1989). Methods for Discovering Novel Motifs in Nucleic Acid Sequences. *Computer Applications in Biosciences.* Vol. 5(5). pp 293-298.

[33] Tompa, M. & Buhler, J. (2001). Finding Motifs Using Random Projections. In *proceedings of the 5th Int'l Conference on Computational Molecular Biology*. Montreal, Canada, Apr 22-25. pp 67-74.

[34] Vlachos, M., Kollios, G. & Gunopulos, G. (2002). Discovering Similar Multidimensional Trajectories. In *proceedings of the 18th International Conference on Data Engineering*. Feb 26-Mar 1, San Jose, CA.

[35] Yi, B, K., & Faloutsos, C. (2000). Fast Time Sequence Indexing for Arbitrary L*p* Norms. In *proceedings of the 26st Int'l Conference on Very Large Databases*. Sep 10-14, Cairo, Egypt. pp 385-394.

Clustering Binary Data Streams with K-means

Carlos Ordonez
carlos.ordonez@ncr.com
Teradata, a division of NCR
San Diego, CA 92127, USA

ABSTRACT

Clustering data streams is an interesting Data Mining problem. This article presents three variants of the K-means algorithm to cluster binary data streams. The variants include On-line K-means, Scalable K-means, and Incremental K-means, a proposed variant introduced that finds higher quality solutions in less time. Higher quality of solutions are obtained with a mean-based initialization and incremental learning. The speedup is achieved through a simplified set of sufficient statistics and operations with sparse matrices. A summary table of clusters is maintained on-line. The K-means variants are compared with respect to quality of results and speed. The proposed algorithms can be used to monitor transactions.

1. INTRODUCTION

Clustering algorithms partition a data set into several disjoint groups such that points in the same group are similar to each other according to some similarity metric [9]. Most clustering algorithms work with numeric data [3; 6; 14; 26], but there has been work on clustering categorical data as well [12; 15; 18; 23]. The problem definition of clustering categorical data is not as clear as the problem of clustering numeric data [9]. There has been extensive database research on clustering large and high dimensional data sets; some important approaches include [6; 3; 17; 26; 2; 7; 20]. Clustering data streams has recently become a popular research direction [13]. The problem is difficult. High dimensionality [17; 1; 2; 23], data sparsity [2; 14] and noise [3; 6; 7; 17] make clustering a more challenging problem.

This work focuses on clustering binary data sets. Binary data sets are interesting and useful for a variety reasons. They are the simplest form of data available in a computer and they can be used to represent categorical data. From a clustering point of view they offer several advantages. There is no concept of noise like that of quantitative data, they can be used to represent categorical data and they can be efficiently stored, indexed and retrieved. Since all dimensions have the same scale there is no need to transform the data set. There is extensive research on efficient algorithms that can manage large data sets [26; 14] and high dimensionality [2; 17]. Despite such efforts K-means remains one of the most popular clustering algorithms used in practice. The main reasons are that it is simple to implement, it is fairly efficient, results are easy to interpret and it can work under a variety of conditions. Nevertheless, it is not the final answer to clustering [6; 9]. Some of its disadvantages include dependence on initialization, sensitivity to outliers and skewed distributions and converging to poor locally optimal solutions. This article introduces several improvements to K-means to cluster binary data streams. The K-means variants studied in this article include the standard version of K-means [19; 9], On-line K-means [25], Scalable K-means [6], and Incremental K-means, a variant we propose.

1.1 Contributions and article outline

This article presents several K-means improvements to cluster binary data streams. Improvements include simple sufficient statistics for binary data, efficient distance computation for sparse binary vectors, sparse matrix operations and a summary table of clustering results showing frequent binary values and outliers. We study how to incorporate these changes into several variants of the K-means algorithms. Particular attention is paid to the Incremental K-means algorithm proposed in this article. An extensive experimental section compares all variants.

The article is organized as follows. Section 2 introduces definitions, the K-means algorithm and examples. Section 3 introduces the proposed improvements to cluster binary data streams and explains how to incorporate them into all K-means variants. Section 4 presents extensive experimental evaluation with real and synthetic data sets. Section 5 briefly discusses related work. Section 6 contains the conclusions and directions for future work.

2. PRELIMINARIES

2.1 Definitions

The input for the K-means clustering algorithm is a data set D having n d-dimensional points and k, the desired number of clusters. The output are three matrices, C, R, W, containing the means, the squared distances and weights respectively for each cluster, a partition of D into k subsets and a measure of cluster quality. Matrices C and R are $d \times k$ and W is a $k \times 1$ matrix. To manipulate matrices we use the following convention for subscripts. For transactions we use i; $i \in \{1, 2, \ldots, n\}$ (i alone is a subscript, whereas i_j refers to item j). For cluster number we use j; $j \in \{1, 2, \ldots, k\}$ and to refer to one dimension we use l: $l \in \{1, 2, \ldots, d\}$. Let $D_1, D_2, \ldots, D_k$ be the k subsets of D induced by clusters s.t. $D_j \cap D_{j'} = \emptyset$ for $j \neq j'$. To refer to a column of C or R we use the j subscript (i.e. C_j, R_j). So C_j and R_j refer to the jth cluster centroid and jth variance matrix respectively and W_j is the jth cluster weight. The *diag*[] notation will be used as a generic operator to obtain a diagonal matrix from a vector or consider only the diagonal of a matrix or to convert the diagonal of a matrix into a vector. This work assumes that a symmetric similarity measure [9], like Euclidean distance, on binary points is acceptable. That is, a 0/0 match is as important as a 1/1 match on some dimension. This is in contrast to asymmetric measures, like the Jaccard coefficient, that give no importance to 0/0 matches [15; 18].

Let $\mathcal{S} = [0,1]^d$ be a d-dimensional Hamming cube. Let $D = \{t_1, t_2, \ldots, t_n\}$ be a database of n points in $\mathcal{S}$. That is, t_i is a binary vector (treated as a transaction). Matrix D is a $d \times n$ sparse binary matrix. Let $T_i = \{l | D_{li} = 1, l \in \{1,2,\ldots,d\}, i \in \{1,2,\ldots,n\}\}$. That is, T_i is the set of non-zero coordinates of t_i; T_i can be understood as a transaction or an itemset to be defined below. Then the input data set D becomes a stream of integers indicating dimensions equal to 1 separated by an end of transaction marker. Since transactions are sparse vectors $|T_i| << d$. This fact will be used to develop an efficient distance computation. We will use T to denote average transaction size ($T = \sum_{i=1}^n |T_i|/n$). Matrices C and R are $d \times k$ and W is a $k \times 1$ matrix. To manipulate matrices we use the following convention for subscripts. For transactions we use i; $i \in \{1,2,\ldots,n\}$ (i alone is a subscript, whereas i_j refers to item j). For cluster number we use j; $j \in \{1,2,\ldots,k\}$ and to refer to one dimension we use l: $l \in \{1,2,\ldots,d\}$. Let $D_1, D_2, \ldots, D_k$ be the k subsets of D induced by clusters s.t. $D_j \cap D_{j'} = \emptyset$ for $j \neq j'$. To refer to a column of C or R we use the j subscript (i.e. C_j, R_j). So C_j and R_j refer to the jth cluster centroid and jth variance matrix respectively; R_j represents a diagonal matrix but it can be manipulated as a vector. W_j is the jth cluster weight. Since D contains binary points C_{lj} can be understood as the fraction of points in cluster j that have dimension l equal to 1. W_j is the fraction of the n points that belong to cluster j. The *diag*[] notation will be used as a generic operator to obtain a diagonal matrix from a vector or consider only the diagonal of a matrix or to convert the diagonal of a matrix into a vector. Points that do not adjust well to the clustering model are called outliers. K-means uses Euclidean distance; the distance from t_i to C_j is

$$\delta(t_i, C_j) = (t_i - C_j)^t (t_i - C_j). \tag{1}$$

2.2 The K-means clustering algorithm

K-means [19] is one of the most popular clustering algorithms [6; 10; 24]. It is simple and fairly fast [9; 6]. K-means is initialized from some random or approximate solution. Each iteration assigns each point to its nearest cluster and then points belonging to the same cluster are averaged to get new cluster centroids. Each iteration successively improves cluster centroids until they become stable.

Formally, the problem of clustering is defined as finding a partition of D into k subsets such that

$$\sum_{i=1}^{n} \delta(t_i, C_j) \tag{2}$$

is minimized, where C_j is the nearest cluster centroid of t_i. The quality of a clustering model is measured by the the sum of squared distances from each point to the cluster where it was assigned [26; 21; 6]. This quantity is proportional to the average quantization error, also known as distortion [19; 24]. The quality of a solution is measured as:

$$q(C) = \frac{1}{n} \sum_{i=1}^{n} \delta(t_i, C_j), \tag{3}$$

which can be computed from R as

$$q(R, W) = \sum_{j=1}^{k} W_j \sum_{l=1}^{d} R_{lj}. \tag{4}$$

2.3 Example

We now present an example with $d = 16, k = 4$ to motivate the need for a summary table. Assume items come as transactions containing integers in $\{1, \ldots, 16\}$. The clustering results for a store in terms of matrices are shown in Figure 1. If we consider a real database environment in which there are hundreds of product categories or thousands of products it is difficult to understand the output matrices C, W. We propose a summary of clusters as shown in Table 1. This summary is easier to understand than the matrix with floating point numbers. Another advantage is that we can see outliers. This table suggests associations [4; 16] among items in the same row.

3. CLUSTERING BINARY DATA STREAMS

3.1 Sparse matrix operations and simple sufficient statistics

The first concern when using a clustering algorithm with large data sets is speed. To accelerate K-means we use sparse distance computation and simpler sufficient statistics. We explain distance computation first. Sparse distance computation is made by precomputing the distances between the null transaction (zeroes on all dimensions) and all centroids C_j. Then only differences for non-null dimensions of each transaction are computed to determine cluster membership as transactions are being read. When D is a sparse matrix and d is high the distance formula is expensive to compute. In typical transaction databases a few dimensions may have non-zero values. So we precompute a distance from every C_j to the null vector $\bar{0}$. To that purpose, we define the k-dimensional vector Δ: $\Delta_j = \delta(\bar{0}, C_j)$. Based on Δ distances can be computed as:

$$\delta(t_i, C_j) = \Delta_j + \sum_{l=1,(t_i)_l \neq 0}^{d} ((t_i)_l - C_{lj})^2 - C_{lj}^2. \tag{5}$$

This computation improves performance significantly, but it does not affect result accuracy. A similar idea is applied to update clusters. This is discussed in more detail later.

K-means can use sufficient statistics [6; 26], which are summaries of $D_1, D_2, \ldots, D_k$ represented by the three matrices M, Q, N that contain sum of points, sum of squared points and number of points per cluster respectively. K-means is fast and the idea of using sufficient statistics is well known to make it faster [6; 10], but we can take a further step. Sufficient statistics can be simplified for clustering binary data. The following result states that the sufficient statistics for the problem of clustering binary vectors are simpler than the sufficient statistics required for clustering numeric data.

$$W = \begin{bmatrix} 0.40 \\ 0.05 \\ 0.20 \\ 0.35 \end{bmatrix} \quad C = \left[\begin{array}{lrrrr} & 1 & 2 & 3 & 4 \\ 1: beer & 0.12 & 0.80 & 0.09 & 0.21 \\ 2: bread & 0.95 & 0.21 & 0.45 & 0.78 \\ 3: coffee & 0.69 & 0.14 & 0.02 & 0.16 \\ 4: crackers & 0.31 & 0.87 & 0.50 & 0.07 \\ 5: ham & 0.21 & 0.89 & 0.21 & 0.14 \\ 6: jelly & 0.83 & 0.12 & 0.25 & 0.11 \\ 7: magazine & 0.11 & 0.09 & 0.03 & 0.03 \\ 8: meat & 0.02 & 0.11 & 0.13 & 0.99 \\ 9: milk & 0.91 & 0.45 & 0.77 & 0.23 \\ 10: notebook & 0.02 & 0.56 & 0.45 & 0.07 \\ 11: oil & 0.21 & 0.11 & 0.56 & 0.70 \\ 12: produce & 0.43 & 0.05 & 0.82 & 0.21 \\ 13: salsa & 0.25 & 0.01 & 0.21 & 0.11 \\ 14: soda & 0.10 & 0.61 & 0.70 & 0.07 \\ 15: water & 0.16 & 0.27 & 0.88 & 0.09 \\ 16: wine & 0.01 & 0.08 & 0.12 & 0.13 \end{array}\right]$$

Figure 1: Basket clusters

j	W_j	C_j (frequent dimensions)	Outliers (odd transactions)
1	40%	90-100%: bread milk 80-90%: jelly	{jelly wine notebook }
2	5%	80-90%: beer crackers ham	{crackers sauce }
3	20%	80-90%: produce water 70-80%: milk soda	{water coffee }
4	35%	90-100%: meat 80-90%: bread 70-80%: oil	{bread magazine }

Table 1: Transaction clusters summary table

Lemma 1 Let D be a set of n transactions of binary data. and $D_1, D_2, \ldots, D_k$ be a partition of D. Then the sufficient statistics required for computing C, R, W are only N and M.

Proof:
To compute W, k counters are needed for the k subsets of of D that are stored in the $k \times 1$ matrix N and then $W_j = N_j/n$. To compute C we use $M_j = \sum_{i=1}^{n} t_i, \forall t_i \in D_j$ and then $C_j = M_j/N_j$. To compute Q the following formula must be computed: $Q_j = \sum_{i=1}^{n} diag[t_i t_i^t] = \sum_{i=1^n} t_i = M_j, \forall t_i \in D_j$. Note that $diag[t_i t_i^t] = t_i$ because $x = x^2$ if x is binary and t_i is a vector of binary numbers. Elements off the diagonal are ignored for diagonal matrices. Then $Q = M$. Therefore, only N and M are needed.□

A consequence of the previous lemma is that R can be computed from C without scanning D or storing Q. Even further, Lemma 1 makes it possible to reduce storage to one half. However, it is not possible to reduce storage further because when K-means determines cluster membership it needs to keep a copy of C_j to compute distances and a separate matrix with M_j to add the point to cluster j. Therefore both C and M are needed for an Incremental or Scalable version, but not for an On-line version. The On-line K-means algorithm to be introduced later could keep a single matrix for centroids and sum of points. For the remainder of this article M is a $d \times k$ matrix and $M_j = \sum_{\forall t_i \in D_j} t_i$ and N is $k \times 1$ matrix and $N_j = |D_j|$. The update formulas for C, R, W are

$$C_j = \frac{1}{N_j} M_j, \tag{6}$$

$$R_j = diag[C_j] - C_j C_j^t, \tag{7}$$

and

$$W_j = \frac{N_j}{\sum_{j'=1}^{k} N_{j'}}. \tag{8}$$

A summary table G, as the one shown in Section 2.3, is necessary to understand very high dimensional binary data. Instead of trying to interpret a $d \times k$ matrix containing floating point numbers the user can focus in those matrix entries that are more interesting. This table should be cheap to maintain so that it introduces a negligible overhead. The user specifies a list of cutoff points and a number of top outliers per cluster. These cutoff points uniformly partition clusters means C_j. Dimension subscripts (items) are inserted into a rank every time C is updated. The table is constructed using a list of cutoff points $c_1, c_2, \ldots,$ s.t. $1 \geq c_{ij} > 0$, and a number of desired outliers per cluster $O_j \geq 0$ for cluster j. For each cutoff c_i there is a (possibly empty) list of dimensions L s.t. $l \in L$ and $c_i > C_{lj} \geq c_{i+1}$. Cutoff points are taken at equal interval lengths in decreasing order starting in 1 and are used for all k clusters for efficiency purposes. Having different cutoff points for each cluster or having them separated at different intervals would introduce a significant overhead to maintain G. Top outliers are inserted into a list in descending order according by distance to their nearest centroid; the outlier with smaller distance is deleted from the list. Outliers are inserted when cluster membership is determined. The cost to maintain this table is low. These are important considerations to update the summary table on-line. (1) It is expected that only a few dimensions will appear in some rank since the data sets are sparse. (2) Only high percentage ranks, closer to 1, are considered interesting. (3) All the cutoff points are separated at equal intervals. In this manner the rank for some dimension of C_j is determined in time $O(1)$ using C_{lj} as an index value.

Irregular intervals would make a linear search mandatory. (4) The dimensions are sorted in lexicographical order by dimension index l inside each rank. The insertion is done in time almost $O(1)$ because only a few dimensions appear in the list and the data set D is assumed to be a sparse matrix. When G becomes populated as transactions are scanned it is likely some dimension of C_j is already inserted and this can be determined in time $O(log(|L|)$ doing a binary search. This would not be the case if all the cutoffs points spanned the $[0, 1]$ interval because the d dimensions would have to be ranked. (5) Outlier insertion is done in time almost $O(1)$. This is the case because most input points are not outliers and it is assumed the desired number of outliers in the list is small.

3.2 K-means variants for binary data streams

This section presents the variants of K-means for binary data streams based on the improvements introduced above. Empty clusters are re-seeded with the the furthest neighbors of non-empty clusters as proposed in [6]. The re-seeding points are extracted from the outlier list stored in the summary table. We believe incorporating all improvements in all algorithms is more valuable than improving only one and then claiming that one is the best. This allows a fair comparison.

The input is $D = \{T_1, T_2, \ldots, T_n\}$, the binary data points given as transactions, as defined in Section 2.1, and k, the desired number of clusters. It is important to observe that points are assumed to come as lists of integers as defined in Section 2. The order of dimensions inside each transaction is not important. The order of transactions is not important as long as transactions do not come sorted by cluster. The output is the clustering model given by the matrices C, R, W, a partition of D into $D_1, D_2, \ldots, D_k$, a summary table G and a measure of cluster quality $q(R, W)$.

Let the nearest neighbor function be defined as

$$NN(t_i) = J,$$

such that

$$\delta(t_i, C_J) \leq \delta(t_i, C_j).$$

Let $\oplus$ be sparse addition of vectors. Mathematically this addition is a normal vector addition, but for implementation purposes only non-zero entries are added. $M_j \oplus t_i$ has complexity $O(T)$. In a similar manner we define a sparse division of matrices $\oslash$, and a sparse matrix subtraction $\ominus$ that only update matrix entries that change after reading a new point and assigning it to its nearest cluster. Empty clusters are re-seeded as follows. If $W_J = 0$ then $C_J \leftarrow t_o$ (outlier transaction), where t_o is a transaction s.t. $\delta(t_o, C_J) \geq \delta(t_i, C_j)$ for $j \neq J$ and $t_i \in D_j$. This outlier t_o is taken from the top outlier list in G.

The On-line K-means variant we use in this work is based on the On-line EM algorithm [22]. The basic difference is that distances are used to compute cluster memberships instead of weighted probabilities with Gaussian distributions. Initialization is based on a sample of k different points to seed C. The weights W_j are initialized to $1/k$ to avoid early re-seeding. A similar streamed version for continuous data is outlined in [21].

We implemented our improvements on the simplified Scalable K-means version proposed in [10]. In this work authors give convincing evidence that a simpler version of the original Scalable K-means [6] produces higher quality results in less time. The only critical parameter is the buffer size. Primary and secondary compression parameters as proposed in [6] are not needed. This version performs primary compression discarding all buffer points each time. Initialization is based on a sample of k different points to seed C. The weights W_j are initialized to $1/k$ to avoid early re-seeding.

Incremental K-means

This is our proposed variant of K-means. This version is a compromise between On-line K-means and the Standard K-means doing one iteration. A fundamental difference with both Standard K-means and Scalable K-means is that Incremental K-means does not iterate until convergence. Another important difference is that initialization, explained below, is done using global statistics of D instead of using a sample of k transactions. Doing only one iteration could be a limitation with continuous (numeric) data, but it is reasonable with binary data. A difference with On-line K-means is that it does not update C and W_j every transaction, but every n/L transactions (L times), and each time it touches the entirety of C and W. The setting for L is important to get a good solution. If $L = 1$ then Incremental K-means reduces to Standard K-means stopped early after one iteration. On the other hand, if $L = n$ then Incremental K-means reduces to On-line K-means. The setting we propose is $L = \sqrt{n}$. This is a good setting for variety of reasons: (1) It is independent from d and k. (2) A larger data set size n accelerates convergence since as $n \rightarrow \infty$, $L \rightarrow \infty$. (3) The number of points used to recompute centroids is the same as the total number of times they are updated. Cluster centroids are initialized with small changes to the global mean of the data set. This mean-based initialization has the advantage of not requiring a pass over the data set to get different seeds for different runs because the global mean can be incrementally maintained. In the pseudo-code r represents a random number in $[0, 1]$, μ is the global mean and $\sigma = diag[\sqrt{R_j}]$ represents a vector of global standard deviations. As $d \rightarrow \infty$ cluster centroids $C_j \rightarrow \mu$. This is based on the fact that the value of $\sum_{i=1}^{n} \delta(t_i, x)$ is minimized when $x = \mu$.

3.3 Suitability for binary data streams

Clustering data streams has become a popular research direction [13]. All the variants introduced above read each data point from disk just once. However, Scalable K-means iterates in memory which can slow it down for an incoming flow of transactions. Moreover, since it iterates until convergence it is necessary to have a threshold on the number of iterations to provide time guarantees. On-line K-means and Incremental K-means can keep up with the incoming flow of transactions since they do not iterate.

The proposed K-means variants have $O(Tkn)$ complexity; where T is the average transaction size; recall that $T << d$. Re-seeding using outliers is done in time $O(k)$. For sparse binary data updating G takes time almost $O(1)$. In the case of Scalable K-means there is an additional complexity factor given by the the number of iterations until convergence. Matrices M, C require $O(dk)$ space and N, W require $O(k)$. Since R is derived from C that space is saved. For all approaches there is an additional requirement $O(b)$ to hold b transactions in a buffer. In the case of Scalable K-means this size is explicitly used as a parameter to the clustering

```
Input: $\{T_1, T_2, \ldots, T_n\}$ and $k$
Output: $C, R, W$ and $q(R, W)$

FOR $j = 1$ TO $k$ DO
    $C_j \leftarrow \mu \pm \sigma r/d$
    $N_j \leftarrow 0$
    $M_j \leftarrow \bar{0}$
    $W_j \leftarrow 1/k$
END
$L = \sqrt{n}$
FOR $i = 1$ TO $n$ DO
    $j = NN(t_i)$
    $M_j \oplus t_i$
    $N_j \leftarrow N_j + 1$
    IF $(i \bmod (n/L)) = 0$ THEN
        $C_j \leftarrow M_j/N_j$
        $R_j \leftarrow C_j - C_j^t C_j$
        $W_j = N_j/i$
        FOR $j = 1$ TO $k$ DO
            IF $W_j = 0$ THEN
                $C_j \leftarrow t_o$
            END
        END
    END
END
```

Figure 2: The Incremental K-means algorithm

process. In any case the buffer space requirements are negligible compared to the clustering model space. In short, space requirements are $O(dk)$ and time complexity is $O(kn)$ for sparse binary vectors..

4. EXPERIMENTAL EVALUATION

This section contains extensive experimental evaluation with real and synthetic data sets. All algorithms were benchmarked on quality of results and performance. The main criterion for quality was $q(R, W)$. Most running times were measured in seconds. All experiments were done on a PC running at 600MHz with 64MB of main memory and a 20 GB hard disk. All algorithms were implemented in the C++ language. The Scalable K-means variant we used was modified from the code available from the authors of [10]. For the sake of completeness we also incorporated our performance improvements into the Standard K-means to compare quality of solutions and performance.

In general the main parameter is only k. Standard K-means and Scalable K-means had a tolerance threshold for $q(R, W)$ set to $\epsilon = 1.0e - 5$. Scalable K-means buffer size was set at 1% as recommended by the authors [6]. On-line and Incremental K-means had no parameters to set other than k. All algorithms updated the clustering summary table G on-line showing the cost to maintain it is low.

We do not show comparisons with the same implementations of Scalable K-means proposed in [6] and [10] because their times on sparse binary data sets are an order of magnitude higher; we believe this would not be interesting and time comparisons would be meaningless.

4.1 Experiments with Real Data Sets

Table 2 shows the best results out of 10 runs for several real data sets. The table shows the quantization error (average sum of squared distances) and elapsed time. Lower numbers are better. When we refer to d it is the number of binary dimensions, not to be confused with the original dimensionality of some data sets that may be smaller. We ran all algorithms a few times to find a good value for k. Such k produced clusters that had C values close to 1 in a few dimensions and mostly values close to 0 for the rest.

The *patient* data set had $n = 655$ and $d = 19$. This data set contained information for patient being treated for heart disease. We extracted categorical columns and each value was mapped to a binary dimension. Patients were already binned by their age in 10-year increments by medical doctors. These were clusters with some dimension $C_{lj} \geq 90\%$ One cluster had 18% of patients who were male, white and aged between 50-59. Another cluster had 15% of patients who were male, white and aged between 70-79; the interesting aspect is that the cluster with males aged 60-69 only had 6% of patients. Another cluster had about 10% of black male patients who were between 30-79 but 36% (most) of them were between 40-49. Another cluster had 14% of patients who were white females aged 70-79. Two outliers were two male patients younger than 40. Since n is very small times were a fraction of 1 second.

The *store* data set had a sample of $n = 234,000$ transactions from a chain of supermarkets with $d = 12$ binary attributes. Each transaction had categorical attributes from its store. The quantitative attributes were ignored. In this case binary dimensions were the values for each categorical variable describing a predefined classification of stores according to their profit performance (low, medium and high), the store layout (convenience, market, supermarket) and store size (small, medium, large, and very large). K-means was run with $k = 10$. In this case small convenience stores had the worst performance found in 5% of the data set. High performance was concentrated mostly in market town stores of small, medium and large size in 15% of transactions. One cluster revealed that most (66%) medium convenience stores but one third had high performance. A heavy cluster with 22% transaction with medium performance happened mostly in convenience and market town layouts, but of varying sizes. There were no interesting outliers in this case; they only included transactions with missing information.

The *bktdept* data set came from another chain of grocery stores. This data set sizes were $n = 1M$ and $d = 161$. The dimensions were the store departments indicating whether the customer bought in that department or not. This data set was challenging given its high dimensionality with most transactions having 5 departments or less and also because clusters had significant overlap. A small k did not produce good results. We had to go up to $k = 40$ to find acceptable solutions. The algorithms were able to identify some significant clusters. One cluster with 1% of baskets involved bread and pet products; a strange combination. Another 1% of baskets involved mostly biscuits and candy. 8% of baskets had cream as the main product with no other significant product bought together with it. Another interesting finding is that 5% mainly bought newspaper, but always with a variety of other products with milk being the most frequent one. Another 10% bought mostly cigarettes, but one fifth of them also bought milk. A couple of interesting outliers with odd combinations included a basket with more than 50 departments visited (too many compared to most baskets) including toys, cooking aids, pet products and food departments, and another basket with cigarettes, nuts and medicine among others. It was surprising that Scalable K-means was slower than Standard K-means.

Data set	k	std KM	online KM	scal KM	incr KM
patient $n = 655, d = 19$	8	0.0217 0	0.0247 0	0.0199 0	0.0181 0
store $n = 234k, d = 12$	10	0.0355 32	0.0448 13	0.0361 19	0.0389 12
bktdept $n = 1M, d = 161$	40	0.0149 823	0.0168 251	0.0151 842	0.0151 284
harddisk $n = 55k, d = 13$	5	0.0119 62	0.0279 36	0.0016 54	0.0016 31
phone $n = 743k, d = 22$	13	0.0116 472	0.0174 57	0.0090 91	0.0095 64

Table 2: Quality of results with real data sets

The *harddisk* data set contained data from a hard disk manufacturer. Each record corresponded to one disk passing or failing a series of tests coded in a categorical variable. The sizes for this data set were $n = 556, 390$ and $d = 13$. Most of the tests involved measurements where a zero would imply fail and a value greater than zero passing (with a degree of confidence). We mapped each measurement to one binary dimension setting a positive number to 1. This data set was hard to cluster because almost 100% of disks passed the tests. All algorithms found clusters in which failing disks were spread across all clusters. However, the best solution indicated that most failing disks (about 1.3%) failed a specific test. In this case outliers were passing disks that passed difficult two difficult tests and failed the common ones. Another outlier was a failing disk that failed the same tests as the outlier passing disk.
The *phone* data set contained demographic data about customers from a telephone company. We selected a few categorical dimensions that were already binary. This data set was very easy to cluster because customers concentrated on three well defined groups with the rest in small clusters. No clusters seemed to indicate anything particularly interesting. We omit further discussion.

4.2 Experiments with Transaction Data Sets

In this section we present experiments with transaction data sets created with the IBM data generator [4; 5]. The defaults we used were as follows. The number of transactions was $n = 100k$. The average transaction size T was 8,10 and 20. Pattern length (I) was one half of transaction length. Dimensionality d was 100, 1000 and 10,000. The rest of parameters were kept at their defaults (average rule confidence=0.25, correlation=0.75). These data sets represent very sparse matrices and very high dimensional data. We did not expect to find any significant clusters, but we wanted to try to the algorithms on them to see how they behaved. For most clusters solutions summarized in Table 3 centroids were below 0.5. According to our criterion they were bad. Clustering transaction files was difficult. Increasing k improved results by a small margin as can be seen. This fact indicated that clusters had significant overlap with the default data generation parameter settings. We further investigated how to make the generator produce clusters and it turned out that correlation and confidence were the most relevant parameters. In that case the quantization error showed a significant decrease for all algorithms. For these synthetic data sets all algorithms found solutions of similar quality. In general Scalable K-means found slightly better solutions. Then Standard and Incremental K-means came in second place. On-line K-means always found the worst solution.
Figure 3 shows performance graphs varying n, d and T with defaults $n = 100k$, $d = 1000$, $T = 10$. The program was run with $k = 10$. The left graph compares the four K-means variants. The center graph shows performance for Incremental K-means at two dimensionalities. The right graph shows Incremental K-means performance varying the average transactions size T. In this case the story is quite different compared to the easier synthetic binary data sets. Performance for Standard K-means and Scalable K-means degrades more rapidly than their counterparts with increasing n. All K-means variants are minimally affected by dimensionality when the average transaction size is kept fixed showing the advantage of performing sparse distance computation. All variants exhibit linear behavior.

4.3 Discussion

In general Incremental K-means and Scalable K-means found the best solutions. In a few cases Scalable K-means found slightly higher quality solutions than Incremental K-means, but in most cases Incremental K-means was as good or better than Scalable K-means. For two real data sets Scalable K-means found slightly better solutions than Incremental K-means. Performance-wise On-line K-means and Incremental K-means were the fastest. It was surprising that in a few cases Incremental K-means was faster than On-line K-means. It turned out that continuously computing averages for every transaction does cause overhead. Standard K-means was always the slowest. On its favor we can say that it found the best solution in a couple of cases with the real data sets. This suggests some clustering problems are hard enough to require several scans over the data set to find a good solution. In general there was not sensitivity to initialization, like when clustering numeric data, but more sophisticated initialization techniques may be employed. Such improvements can be added to this proposal.

5. RELATED WORK

Research on scaling K-means to cluster large data sets includes [6] and [10], where Scalable K-means is proposed and improved. An alternative way to re-seed clusters for K-means is proposed in [11]. We would like to compare that approach to the one used in this article. The problem of clustering data streams is proposed in [13]. There has been some work in that direction. Randomized clustering algorithms to find high-quality solutions on streaming data are proposed in [21].

Data set	*d*	*k*	std KM	online KM	scal KM	incr KM
T8I4D100k	1000	10	0.00758	0.00760	0.00746	0.00746
T8I4D100k	1000	20	0.00719	0.00727	0.00710	0.00720
T10I5D100k	100	10	0.07404	0.07545	0.07444	0.07456
T10I5D100k	100	20	0.07042	0.07220	0.07100	0.07151
T20I10D100k	10000	10	0.00194	0.00195	0.00192	0.00192
T20I10D100k	10000	20	0.00186	0.00191	0.00184	0.00184

Table 3: Quality of results with transaction data sets

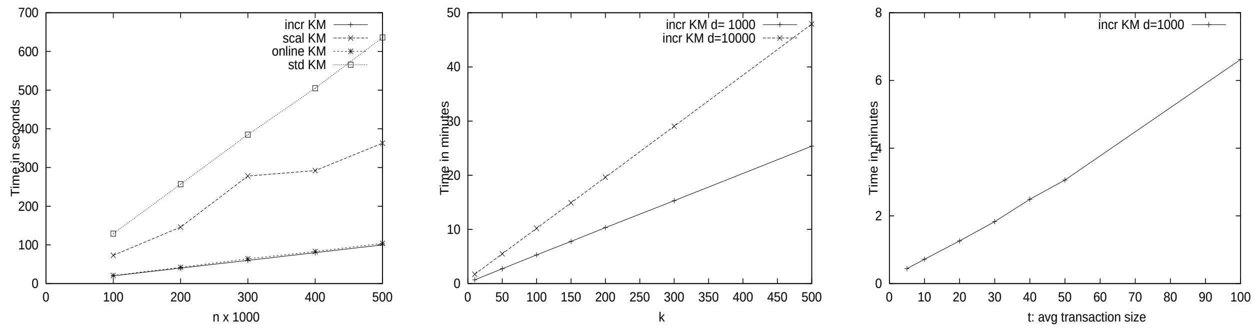


Figure 3: Performance varying n, k and T with high d

There is work on clustering categorical data sets from a Data Mining perspective. The K-modes algorithm is proposed in [18]; this algorithm is a variant of K-means, but using only frequency counting on 1/1 matches. Another algorithm is ROCK that groups points according to their common neighbors (links) in a hierarchical manner [15]. CACTUS is a graph-based algorithm that clusters categorical values using point summaries. These approaches are different from ours since they are not distance-based and have different optimization objectives. Also, ROCK is a hierarchical algorithm. One interesting aspect discussed in [15] is the error propagation when using a distance-based algorithm to cluster binary data in a hierarchical manner. But fortunately K-means is not hierarchical. An advantage of the K-means variants introduced above over purely categorical clustering algorithms, like K-modes, CACTUS and ROCK, is that they can be extended to cluster points with both numeric (continuous) and categorical dimensions (with categorical values mapped to binary). This problem is known as clustering mixed data types [9; 8]. There is some criticism on using distance similarity metrics for binary data [9], but our point is that a careful summarization and interpretation of results can make the approach useful. A fundamental difference with associations [4] is that associations describe frequent patterns found in the data matched only on 1/1 occurrences. Another difference is that associations do not summarize the transactions that support the discovered pattern.

6. CONCLUSIONS

This article proposed several improvements for K-means to cluster binary data streams. Sufficient statistics are simpler for binary data. Distance computation is optimized for sparse binary vectors. A summary table with best cluster dimensions and outliers is maintained on-line. The proposed improvements are fairly easy to incorporate. All variants were compared with real and synthetic data sets. The proposed Incremental K-means variant is faster than the already quite fast Scalable K-means and finds solution of comparable quality. In general these two algorithms find higher quality than On-line K-means.

Future work includes the following. Mining associations from clusters is an interesting problem. We plan to approximate association support from clusters avoiding scanning transactions. We want to introduce further processing using set-oriented metrics like the Jaccard coefficient or association support. Some of our improvements apply to continuous data but that is a harder problem for a variety of reasons. We believe a further acceleration of Incremental K-means is not possible unless approximation, randomization or sampling are used.

Acknowledgements

Special thanks to Chris Jermaine, from the University of Florida, whose comments improved the presentation of this article.

7. REFERENCES

[1] C. Aggarwal, C. Procopiuc, J. Wolf, P. Yu, and Jong Park. Fast algorithms for projected clustering. In *ACM SIGMOD Conference*, 1999.

[2] C. Aggarwal and P. Yu. Finding generalized projected clusters in dimensional spaces. In *ACM SIGMOD Conference*, 2000.

[3] R. Agrawal, J. Gehrke, D. Gunopolos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD Conference*, 1998.

[4] R. Agrawal, T. Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conference*, pages 207–216, 1993.

[5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB Conference*, 1994.

[6] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *ACM KDD Conference*, 1998.

[7] M. Breunig, H.P. Kriegel, P. Kroger, and J. Sander. Data bubbles: Quality preserving performance boosting for hierarchical clustering. In *ACM SIGMOD Conference*, 2001.

[8] R. Dubes and A.K. Jain. *Clustering Methodologies in Exploratory Data Analysis*, pages 10–35. Academic Press, New York, 1980.

[9] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*, pages 10–45. J. Wiley and Sons, 1973.

[10] F. Fanstrom, J. Lewis, and C. Elkan. Scalability for clustering algorithms revisited. *SIGKDD Explorations*, 2(1):51–57, June 2000.

[11] B. Fritzke. The LBG-U method for vector quantization – an improvement over LBG inspired from neural networks. *Neural Processing Letters*, 5(1):35–45, 1997.

[12] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus-clustering categorical data using summaries. In *ACM KDD Conference*, 1999.

[13] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *FOCS*, 2000.

[14] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *SIGMOD Conference*, 1998.

[15] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *ICDE Conference*, 1999.

[16] J. Han, J. Pei, and Yiwei Yun. Mining frequent patterns without candidate generation. In *ACM SIGMOD Conference*, 2000.

[17] A. Hinneburg and D. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality. In *VLDB Conference*, 1999.

[18] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3), 1998.

[19] J.B. MacQueen. Some method for classification and analysis of multivariate observations. In *Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[20] A. Nanopoulos, Y. Theodoridis, and Y. Manolopoulos. C2p: Clustering based on closest pairs. In *VLDB Conference*, 2001.

[21] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high quality clustering. In *IEEE ICDE*, 2002.

[22] C. Ordonez and E. Omiecinski. FREM: Fast and robust EM clustering for large data sets. In *ACM CIKM Conference*, 2002.

[23] C. Ordonez, E. Omiecinski, and Norberto Ezquerra. A fast algorithm to cluster high dimensional basket data. In *IEEE ICDM Conference*, 2001.

[24] S. Roweis and Z. Ghahramani. A unifying review of Linear Gaussian Models. *Neural Computation*, 1999.

[25] M. Sato and S. Ishii. On-line EM algorithm for the normalized Gaussian network. *Neural Computation*, 12(2), 2000.

[26] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *ACM SIGMOD Conference*, 1996.

Processing Frequent Itemset Discovery Queries by Division and Set Containment Join Operators

Ralf Rantzau
Department of Computer Science, Electrical Engineering and Information Technology
University of Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany
rrantzau@acm.org

ABSTRACT

SQL-based data mining algorithms are rarely used in practice today. Most performance experiments have shown that SQL-based approaches are inferior to main-memory algorithms. Nevertheless, database vendors try to integrate analysis functionalities to some extent into their query execution and optimization components in order to narrow the gap between data and processing. Such a database support is particularly important when data mining applications need to analyze very large datasets or when they need access current data, not a possibly outdated copy of it.
We investigate approaches based on SQL for the problem of finding frequent itemsets in a transaction table, including an algorithm that we recently proposed, called *Quiver*, which employs universal and existential quantifications. This approach employs a table schema for itemsets that is similar to the commonly used vertical layout for transactions: each item of an itemset is stored in a separate row. We argue that expressing the frequent itemset discovery problem using quantifications offers interesting opportunities to process such queries using set containment join or set containment division operators, which are not yet available in commercial database systems. Initial performance experiments reveal that Quiver cannot be processed efficiently by commercial DBMS. However, our experiments with query execution plans that use operators realizing set containment tests suggest that an efficient processing of Quiver is possible.

Keywords

association rule discovery, relational division, set containment join

1. INTRODUCTION

The frequent itemset discovery algorithms used in today's data mining applications typically employ sophisticated in-memory data structures, where the data is stored into and retrieved from flat files. This situation is driven by the need for high-performance ad-hoc data mining in the areas of health-care or e-commerce, see for example [12] for an overview. However, ad-hoc mining often comes at a high cost because huge investments into powerful hardware and sophisticated software are required.

1.1 Why Data Mining with SQL?

From a performance perspective, data mining algorithms that are implemented with the help of SQL are usually considered inferior to algorithms that process data outside the database system [21]. One of the most important reasons is that offline algorithms employ sophisticated in-memory data structures and try to scan the data as few times as possible while SQL-based algorithms either require several scans over the data or require many and complex joins between the input tables. However, DBMS already provide techniques to deal with huge datasets. These techniques need to be re-implemented in part if offline algorithms are used to analyze data that is so large that the in-memory data structures grow beyond the size of the computer's main-memory.
One can classify data mining model generation algorithms along the following list of features:

- On-line vs. off-line mining: While the algorithm is running, the user may change parameters and retrieve intermediate or approximate results as early as possible.
- In-place vs. out-of-place mining: The algorithm retrieves original data from the database itself or it operates on a copy of the original data, perhaps with a data layout that is optimal for the algorithm.
- Database-coupled vs. database-decoupled mining: The algorithm uses the DBMS merely as a file system, i.e., it stores and loads data using trivial queries or it exploits the query processor capabilities of the DBMS for nontrivial mining queries.

Obviously, some algorithms lie somewhere in-between these categories. For example, there are implementations of the frequent itemset discovery method that allow a user interaction only to some extent: a user can increase the minimum support threshold but maybe not decrease it because this would require revisiting some previously seen transactions. Hence, such an implementation realizes on-line mining only up to a certain degree.
We believe that there are datasets of high potential value that are so large that the only way to analyze them in their entirety (without sampling) is to employ algorithms that exploit the scalable query processing capability of a database system, i.e., these datasets require in-place, database-coupled mining, however, at the potential cost of allowing only off-line mining. On-line mining can be achieved if the query

execution plans are based on non-blocking operators along the "critical paths" of the data streams from the input tables to the root operator of the plan. A framework for such operators supporting queries within the knowledge discovery process has been investigated for example in [5].

1.2 The Problem of Frequent Itemset Discovery

We briefly introduce the widely established terminology for frequent itemset discovery. An *item* is an object of analytic interest, like a product of a shop or a URL of a document on a web site. An *itemset* is a set of items and a *k-itemset* contains k items. A *transaction* is an itemset representing a fact, like a basket of distinct products purchased together or a collection of documents requested by a user during a web site visit.

Given a set of transactions, the *frequent itemset discovery problem* is to find itemsets within the transactions that appear at least as frequently as a given threshold, called *minimum support*. For example, a user can define that an itemset is frequent if it appears in at least 2% of all transactions.

Almost all frequent itemset discovery algorithms consist of a sequence of steps that proceed in a bottom-up manner. The result of the kth step is the set of frequent k-itemsets, denoted as F_k. The first step computes the set of frequent items or 1-itemsets F_1. Each of the following step consists of two phases:

1. The *candidate generation phase* computes a set of potential frequent k-itemsets from F_{k-1}. The new set is called C_k, the set of candidate k-itemsets. It is a superset of F_k.
2. The *support counting phase* filters out those itemsets from C_k that appear more frequently in the given set of transactions than the minimum support and stores them in F_k.

All known SQL-based algorithms follow this "classical" two-phase approach. There are other, non-SQL-based approaches such as *frequent-pattern growth* [7], which do not require a candidate generation phase. However, the frequent-pattern growth algorithm employs a (relatively complex) main-memory data structure called frequent-pattern tree, which disqualifies it for a straightforward comparison with SQL-based algorithms.

1.3 Set Containment Tests

The *set containment join* (SCJ) $\bowtie_{\subseteq}$ is a join between two relations R and S on set-valued attributes $R.a$ and $S.b$, the join condition being that $a \subseteq b$. It is considered an important operator for queries involving set-valued attributes [8; 9; 11; 15; 14; 17; 18; 25]. For example, set containment test operations have been used for optimizing a workload of continuous queries, in particular for checking if one query is a subquery of another [4]. Another application area is content-based retrieval in document databases, when one tries to find a collection of documents containing a set of keywords.

Frequent itemset discovery, one of the most important data mining operations, is an excellent example of a problem that can be expressed using set containment joins: Given a set of transactions T and a single (candidate) itemset i, how many transactions $t \in T$ fulfill the condition $i \subseteq t$? If this number

transaction	*items*
1001	$\{A, D\}$
1002	$\{A, B, C, D\}$
1003	$\{A, C, D\}$

(a) *Transaction*

itemset	*items*
101	$\{A, B, D\}$
102	$\{A, C\}$

(b) *Itemset*

itemset	*items*	*transaction*	*items*
101	$\{A, B, D\}$	1002	$\{A, B, C, D\}$
102	$\{A, C\}$	1002	$\{A, B, C, D\}$
102	$\{A, C\}$	1003	$\{A, C, D\}$

(c) *Contains*

Figure 1: An example set containment join: *Itemset* $\bowtie_{items \subseteq items}$ *Transaction* = *Contains*

transaction	*item*
1001	A
1001	D
1002	A
1002	B
1002	C
1002	D
1003	A
1003	C
1003	D

(a) *Transaction* (single dividend)

itemset	*item*
101	A
101	B
101	D
102	A
102	C

(b) *Itemset* (several divisors)

transaction	*itemset*
1002	101
1002	102
1003	102

(c) *Contains* (several quotients)

Figure 2: An example set containment division: *Transaction* $\div_{\{item\} \supseteq \{item\}}$ *Itemset* = *Contains*

is beyond the minimum support threshold the itemset is considered frequent. In general, we would like to test a whole set of itemsets I for containment in T: *Find the number of tuples in* $I \bowtie_{\subseteq} T$ *for each distinct transaction in* T.

The set containment join takes unnormalized relations as input, i.e., the relations are not in the first normal form (1NF), which would require that the attributes have only atomic values. We have recently introduced the *set containment division* (SCD) operator $\div_{\supseteq}$, which is equivalent to SCJ but the relations are in 1NF [20]. It is a generalization of the well-known relational division operator. Given two relations $R(A \cup B)$ and $S(C \cup D)$, where $A = \{a_1, \ldots, a_m\}$, $B = \{b_1, \ldots, b_n\}$, $C = \{c_1, \ldots, c_n\}$, and $D = \{d_1, \ldots, d_p\}$ are sets of attributes and the attribute domains of b_i are compatible with c_i for all $1 \leq i \leq n$. The sets A, B, and C have to be nonempty, while D may be empty. The set containment division is defined by the algebra expression

$$\begin{aligned} R \div_{B \supseteq C} S &= \bigcup_{x \in \pi_D(S)} \left(\left(R \div \pi_C \left(\sigma_{D=x} \left(S \right) \right) \right) \times (x) \right) \\ &= T \left(A \cup D \right) . \end{aligned}$$

Note that the superset symbol ($\supseteq$) is used to contrast SCD from the traditional division operator and to show its similarity to the SCJ operator, which has a subset relation ($\subseteq$). It does not mean that SCD compares set-valued attributes, unlike SCJ.

If $D = \emptyset$, i.e., if S consists of a single group or "set" of elements, then the set containment division becomes the normal division operator:

$$R \div_{B \supseteq C} S = R \div S = T(A)\,,$$

where R is called *dividend*, S *divisor*, and T *quotient*.

To compare the two operators SCJ and SCD, Figures 1 and 2 sketch a simple example that uses equivalent input data for the operators. SCJ is based on unnormalized data while SCD requires input relations in 1NF. Each tuple of the result relation *Contains* delivers the information on which itemset is contained in which transaction.

In this paper, we argue that if SQL would allow expressing SCJ and SCD problems in an intuitive manner and if several algorithms implementing these operators were available in a DBMS, this would greatly facilitate the processing of queries for frequent itemset discovery.

1.4 Paper Overview

The remainder of this paper is organized as follows. In Section 2, we discuss frequent itemset algorithms that employ SQL queries. Query execution plans that realize the queries are discussed in Section 3. Section 4 explains algorithms for the SCJ and SCD operators and their implementation based on an open source Java class library for building query processors. Several experiments that assess the performance of the algorithms based on both synthetic and real-life datasets are presented in Section 5. We conclude the paper in Section 6 and give a brief outlook on future work.

2. FREQUENT ITEMSET DISCOVERY WITH SQL

Algorithms for deriving association rules with SQL have been studied in great detail in the past. Frequent itemset discovery, a preprocessing phase of association rule discovery, is more time-consuming than the subsequent rule generation phase [2]. Therefore, all association rule discovery algorithms described in the literature strive to optimize the frequent itemset generation phase.

2.1 Related Work

The *SETM* algorithm is the first SQL-based approach [10] described in the literature. Subsequent work suggested improvements of SETM. For example, in [24] views are used instead of some of the tables employed in SETM. The authors also suggest a reformulation of SETM using subqueries. The performance of SETM on a parallel DBMS has been studied [16]. The results have shown that SETM does not perform well on large datasets and new approaches have been devised, like for example *K-Way-Join*, *Three-Way-Join*, *Subquery*, and *Two-Group-Bys* [22]. These new algorithms differ only in the statements used for support counting. They use the same statement for generating C_k, as shown in Figure 3 for the example value $k = 4$. The statement creates a new candidate k-itemset by exploiting the fact that all of its k subsets of size $k-1$ have to be frequent. This condition is called *Apriori property* because it was originally introduced in the *Apriori* algorithm [2; 13]. Two frequent subsets are picked to construct a new candidate. These itemsets must have the same items from position 1 to $k-1$. The new candidate is further constructed by adding the kth items of both itemsets in a lexicographically ascending order. In addition, the statement checks if the $k-2$ remaining subsets of the new candidates are frequent as well, expressed by the two "skip item" predicates in Figure 3.

```
INSERT
INTO C4 (itemset, item1, item2, item3, item4)
SELECT newid(), item1, item2, item3, item4
FROM (
  SELECT a1.item1, a1.item2, a1.item3, a2.item3
  FROM   F3 AS a1, F3 AS a2, F3 AS a3, F3 AS a4
  WHERE  a1.item1 = a2.item1 AND
         a1.item2 = a2.item2 AND
         a1.item3 < a2.item3 AND
         -- Apriori property.
         -- Skip item1.
         a3.item1 = a1.item2 AND
         a3.item2 = a1.item3 AND
         a3.item3 = a2.item3 AND
         -- Skip item2.
         a4.item1 = a1.item1 AND
         a4.item2 = a1.item3 AND
         a4.item3 = a2.item3) AS temporary;
```

Figure 3: Candidate generation phase in SQL-92 for $k = 4$. Such a statement is used by all known algorithms that have a horizontal table layout.

Another well-known approach presented in [22] called *K-Way-Join* uses k instances of the transaction table and joins it k times with itself and with a single instance of C_k. The support counting phase of K-Way-Join is illustrated in Figure 4(a), where we contrast it to an equivalent approach using a vertical table layout in Figure 4(b) that is similar to our *Quiver* approach, discussed below.

The algorithms presented in [22] perform differently for different data characteristics. The authors report that Subquery is the best algorithm overall compared to the other approaches based on SQL-92. The reason is that it exploits common prefixes between candidate k-itemsets when counting the support.

More recently, an approach called *Set-oriented Apriori* has been proposed [23]. The authors argue that too much redundant computation is involved in each support counting phase. They claim that it is beneficial to save the information about which item combinations are contained in which transaction, i.e., Set-oriented Apriori generates an additional table $T_k(\mathit{transaction}, \mathit{item}_1, \ldots, \mathit{item}_k)$ in the kth step of the algorithm. The algorithm derives the frequent itemsets by grouping on the k items of T_k and it generates T_{k+1} using T_k. Their performance results have shown that Set-oriented Apriori performs better than Subquery, especially for high values of k.

2.2 The Quiver Algorithm

We have recently introduced a new SQL-based algorithm for deriving frequent itemsets called *Quiver (QUantified Itemset discoVERy)* [19]. The basic idea of the algorithm is to use a vertical layout for representing itemsets in a relation in the same way as transactions are represented, i.e., the relations storing candidate and frequent itemsets have the same schema $(\mathit{itemset}, \mathit{pos}, \mathit{item})$, similar to the schema for transactions $T(\mathit{transaction}, \mathit{item})$. Table 1 (taken from [19]) illustrates the difference between the horizontal and the vertical layout for transactions and itemsets.

Quiver's SQL statements employ *universal quantification* in both the candidate generation and the support counting phase. Quiver constructs a candidate $(k+1)$-itemset i from two frequent k-itemsets $f_1.\mathit{itemset}$ and $f_2.\mathit{itemset}$ by checking if for **all** *item* values of f_1 at position $1 \leq f_1.\mathit{pos} =$

```
INSERT
INTO     S4 (itemset, support)
SELECT   c1.itemset, COUNT(*)
FROM     C4 AS c1,
         T AS t1, T AS t2, T AS t3, T AS t4
WHERE    c1.item1       = t1.item        AND
         c1.item2       = t2.item        AND
         c1.item3       = t3.item        AND
         c1.item4       = t4.item        AND
         t1.transaction = t2.transaction AND
         t1.transaction = t3.transaction AND
         t1.transaction = t4.transaction
GROUP BY c.itemset
HAVING   COUNT(*) >= @minimum_support;

INSERT
INTO   F4 (itemset, item1, item2, item3, item4)
SELECT c1.itemset, c1.item1, c1.item2, c1.item3, c1.item4
FROM   C4 AS c1, S4 AS s1
WHERE  c1.itemset = s1.itemset;
```

(a) Original, horizontal version of K-Way-Join

```
INSERT
INTO     S4 (itemset, support)
SELECT   c1.itemset, COUNT(*)
FROM     C4 AS c1, C4 AS c2, C4 AS c3, C4 AS c4,
         T  AS t1, T  AS t2, T  AS t3, T  AS t4
WHERE    c1.itemset     = c2.itemset     AND
         c1.itemset     = c3.itemset     AND
         c1.itemset     = c4.itemset     AND
         t1.transaction = t2.transaction AND
         t1.transaction = t3.transaction AND
         t1.transaction = t4.transaction AND
         c1.item        = t1.item        AND
         c2.item        = t2.item        AND
         c3.item        = t3.item        AND
         c4.item        = t4.item        AND
         c1.pos         = 1              AND
         c2.pos         = 2              AND
         c3.pos         = 3              AND
         c3.pos         = 4
GROUP BY c1.itemset
HAVING   COUNT(*) >= @minimum_support;

INSERT
INTO   F4 (itemset, pos, item)
SELECT c1.itemset, c1.pos, c1.item
FROM   C4 AS c1, S4 AS s1
WHERE  c1.itemset = s1.itemset;
```

(b) Vertical version of K-Way-Join

Figure 4: Support counting phase of K-Way-Join for $k = 4$

$f_2.pos = i.pos \leq k - 1$ the condition $f_1.item = f_2.item = i.item$ holds. This is the prefix construction used in the Apriori algorithm mentioned in Section 2.1. Similar predicates are added to the WHERE clause of the SQL query to realize the Apriori property, i.e., the "skip item" predicates mentioned in the SQL query used for retrieving horizontal candidates in Figure 3. The SQL statement used for this phase is quite lengthy, therefore we do not present it in this paper. It is shown in [19] together with an equivalent query in tuple relational calculus to emphasize the use of universal quantification (the universal quantifier "$\forall$").

The frequent itemset counting phase of Quiver uses universal quantification as well. It is used to express the set containment test: For each candidate itemset i, find the number of transactions where for each transaction t there is a value $t.item = i.item$ for **all** values of $i.pos$. Note that this condition holds for itemsets of arbitrary size k. No further restriction involving the parameter k is necessary. Hence, the same SQL statement, depicted in Figure 5(a), can be used for any iteration of the frequent itemset discovery algorithm. The nested NOT EXISTS predicate realizes the universal quantifier.

Unfortunately, there is no universal quantifier defined in the SQL standard, not even in the upcoming standard SQL:2003. It is difficult for an optimizer to recognize that the query

Layout	Transactions	Itemsets
horizontal (single-row/ multi-column)		

transaction	$item_1$	$item_2$	$item_3$
1001	*A*	*C*	*NULL*
1002	*A*	*B*	*C*

itemset	$item_1$	$item_2$
101	*A*	*B*
102	*A*	*C*

Layout	Transactions	Itemsets
vertical (multi-row/ single-column)		

transaction	*item*
1001	*A*
1001	*C*
1002	*A*
1002	*B*
1002	*C*

itemset	*pos*	*item*
101	1	*A*
101	2	*B*
102	1	*A*
102	2	*C*

Table 1: Table layout alternatives for storing the items of transactions and itemsets

for support counting in Figure 5(a) is actually a division problem. Therefore, we suggest a set containment division operator in SQL, whose syntax is illustrated in Figure 5(b). Its semantics is equivalent to that of Figure 5(a).

Note that we cannot simply write

```
T AS t1 SET_CONTAINMENT_DIVIDE BY C AS c1
```

because the layout of the divisor table $C(itemset, pos, item)$ has the attribute pos, the lexicographical position of an item within an itemset, that is needed when candidates are created and that would incorrectly lead to grouping the divisor table on the attributes $(itemset, pos)$ instead of $(itemset)$. Hence, we omit this attribute in the SELECT clause of the subquery.

3. QUERY EXECUTION STRATEGIES

In this section, we show query execution plans (QEPs) produced by a commercial DBMS for some example SQL statements of the SQL-based frequent itemset discovery algorithms K-Way-Join and Quiver. In addition, we show how to realize the Quiver query for frequent itemset counting using a SCDs operator.

We compare Quiver to K-Way-Join because of their structural similarity. Remember that Quiver uses a vertical table layout for both itemsets and transactions while K-Way-Join uses a horizontal table layout for itemsets and a vertical layout for transactions. We are aware of the fact that K-Way-Join is not the best algorithm based on SQL-92 overall, even if our preliminary tests with a synthetic dataset has shown the opposite (see Section 5). However, we decided to derive first results by comparing these two approaches. Further performance experiments with other approaches will follow.

In Figures 6(a) and (b), we sketch the query execution plans that we derived during performance experiments with Microsoft SQL Server 2000 for several SQL algorithms discussed in Section 2.1. The plans (a) and (b) illustrate the execution strategies chosen by the optimizer for a given transaction table T and a candidate table C_4 for the queries shown in Figures 4(a) and 5(a). The operators in use are sort, row count, retrieval of the first row only (Top_1), rename (ρ), group (γ), join ($\bowtie$), left anti-semi-join ($\overline{\ltimes}$), index scan ($IScan$), and index lookup ($ISeek$).

The QEP (a) for the K-Way-Join query uses four hash-joins to realize the Apriori trick. It joins the candidate table C_4 four times with the transaction table.

The QEP (b) for the Quiver query using quantifications employs anti-semi-joins. Left anti-semi-join returns all rows of the left input that have no matching row in the right input. The top left anti-semi-join operator test for each combination of values $(c1.itemset, c1.transaction)$ on the left if at

```
INSERT
INTO     S (itemset, support)
SELECT   itemset, COUNT(DISTINCT transaction) AS support
FROM     (
  SELECT c1.itemset, t1.transaction
  FROM   C AS c1, T AS t1
  WHERE  NOT EXISTS (
    SELECT *
    FROM   C AS c2
    WHERE  NOT EXISTS (
      SELECT *
      FROM   T AS t2
      WHERE  NOT (c1.itemset = c2.itemset) OR
             (t2.transaction = t1.transaction AND
              t2.item        = c2.item)))
         ) AS Contains
GROUP BY itemset
HAVING   support >= @minimum_support;

INSERT
INTO   F (itemset, pos, item)
SELECT c1.itemset, c1.pos, c1.item
FROM   C AS c1, S AS s1
WHERE  c1.itemset = s1.itemset;
```

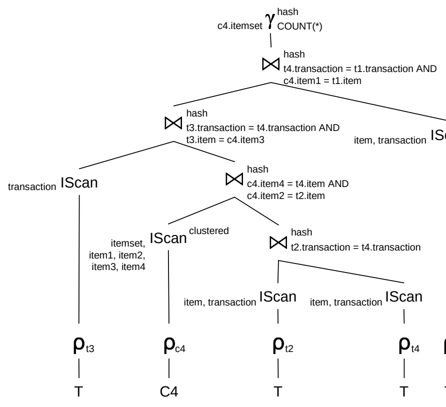
(a) Quiver using quantifications

```
INSERT
INTO     S (itemset, support)
SELECT   c1.itemset, COUNT(*)
FROM     T AS t1 SET_CONTAINMENT_DIVIDE BY (
           SELECT itemset, item
           FROM   C
         ) AS c1
         ON (t1.item = c1.item)
GROUP BY c1.itemset
HAVING   COUNT(*) >= @minimum_support;

INSERT
INTO   F (itemset, pos, item)
SELECT c1.itemset, c1.pos, c1.item
FROM   C AS c1, S AS s1
WHERE  c1.itemset = s1.itemset;
```

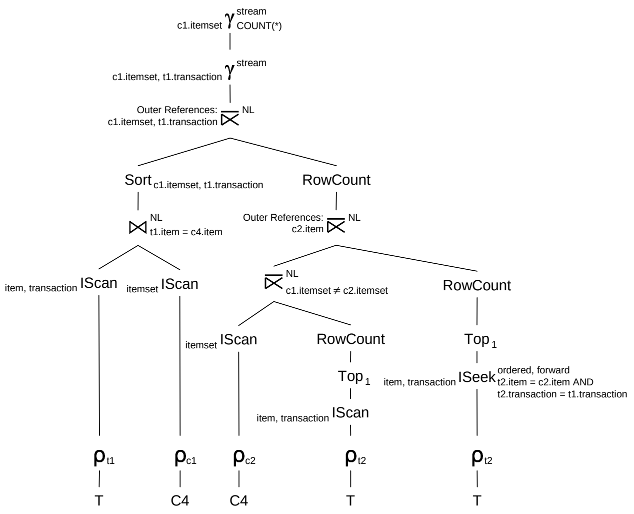
(b) Quiver using a set containment division operation, specified by hypothetical SQL keywords

Figure 5: Support counting phase of Quiver for any value of k. We write C (F) instead of C_k (F_k) for the candidate (frequent) itemset table because of the independence of the parameter k.



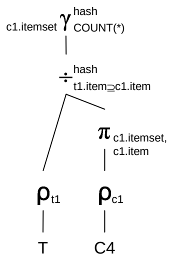
c4.itemset γ hash COUNT(*)
⋈ hash t4.transaction = t1.transaction AND c4.item1 = t1.item
⋈ hash t3.transaction = t4.transaction AND t3.item = c4.item3
item, transaction IScan
transaction IScan
⋈ hash c4.item4 = t4.item AND c4.item2 = t2.item
itemset, item1, item2, item3, item4 IScan clustered
⋈ hash t2.transaction = t4.transaction
item, transaction IScan
item, transaction IScan
ρt3
ρc4
ρt2
ρt4
ρt1
T
C4
T
T
T

(a) Original, horizontal version of K-Way-Join with hash-joins



c1.itemset γ stream COUNT(*)
c1.itemset, t1.transaction γ stream
Outer References: c1.itemset, t1.transaction NL
Sort c1.itemset, t1.transaction
RowCount
⋈ NL t1.item = c4.item
Outer References: c2.item NL
item, transaction IScan
itemset IScan
NL c1.itemset ≠ c2.itemset
RowCount
itemset IScan
RowCount
Top 1
Top 1
item, transaction ISeek ordered, forward t2.item = c2.item AND t2.transaction = t1.transaction
item, transaction IScan
ρt1
ρc1
ρc2
ρt2
ρt2
T
C4
C4
T
T

(b) Quiver with nested-loops left anti-semi-joins



(c) Quiver with hash-based SCD

Figure 6: Example query execution plans computing F_4, generated for the SQL queries in Figures 4(a), 5(a), and 5(b)

least one row can be retrieved from the right input. If no, then the combination qualifies for the subsequent grouping and counting, otherwise the left row is skipped. A similar processing is done for the left anti-semi-join with the outer reference *c2.item*. An interesting point to note is that the index scan (not the seek) of *t2* on *item, transaction* is uncorrelated. Every access to this table is used to check if the transaction table is empty or not. For our problem of frequent itemset discovery this table is non-empty by default.
In addition to the QEPs that have been derived for real SQL queries using a commercial DBMS, we illustrate a hypothetical QEP for the version of the Quiver algorithm that employs a set containment division operator in Figure 6(c). The corresponding query using hypothetical SQL keywords is specified in Figure 5(b).

4. ALGORITHMS FOR SCD AND SCJ

The term “hash” specified for the set containment division operator in the QEP in Figure 6(c) was used to illustrate that there may be several implementations (physical operators) in a DBMS realizing SCD, in this example based on the hash-division algorithm [6]. Since SCD is based on the division operator, as shown by the definition in Section 1.3, many implementations of division come into consideration for realizing SCD depending on the current data characteristics, e.g., the data may be grouped or even sorted on certain attributes, as discussed in [20].
The SCD operator that we used for the performance tests is based on the hash-division algorithm [6]. Hash-division uses two hash tables. The divisor hash table stores all rows of the divisor table (i.e., all rows of a single divisor group for SCD) and assigns to each row an integer number that is equal to $k-1$ for the kth row. The quotient hash table stores all distinct values of the dividend’s quotient attributes (i.e.,

attribute set A in Section 1.3). These values are a superset of the final quotients, i.e., they are quotient candidates. For each quotient candidate value, a bitmap is kept, whose length is the number of divisor rows. For each divisor, the dividend table is scanned once. For each dividend row, the algorithm looks up the value i in the divisor hash table as an index for the the quotient bitmap. Then, the quotient candidate is looked up in the quotient hash table. If it is found, the bit at position i is set to *true*, otherwise a new quotient candidate and bitmap is inserted with all bits set to *false*. After the scan, the quotient candidates in the quotient hash table are returned whose bits are all *true*.

We have realized the QEPs shown in Figure 6 using the open source Java class library *XXL (eXtensible and fleXible Library for data processing)* for building query processors [3]. Some example classes are *BTree*, *Buffer*, *HashGrouper*, *Join*, and *Predicate*. Interestingly, they provide a class called *SortBasedDivision* that implements the merge-division algorithm. However, several necessary operators for our purposes like nested-loops anti-semi-join and hash-join had to be built from scratch.

Several algorithms for SCJs have been proposed, as mentioned in Section 1.3. It may seem curious that we only show a QEP and experiments using Quiver realized with our home-made SCD instead of the better-known SCJ operator. In fact, we have realized an implementation of SCJ using the *Adaptive Pick-and-Sweep Join* algorithm [15], which is claimed to be the most efficient SCJ algorithm to date and made initial performance tests, presented in Section 5.2. However, we cannot yet report on a thorough comparison of different implementations for SCD and SCJ as it is part of our ongoing work. Note that an interesting recent publication [11] on SCJs did not investigate this algorithm.

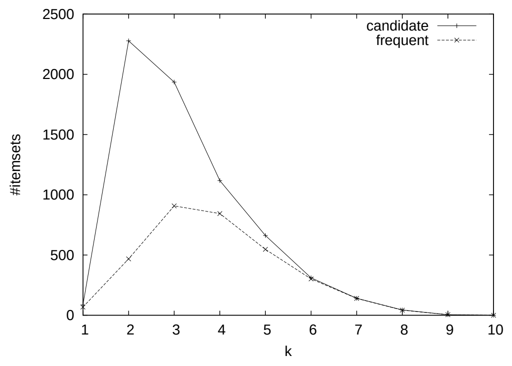
5. EXPERIMENTS

5.1 Commercial DBMS

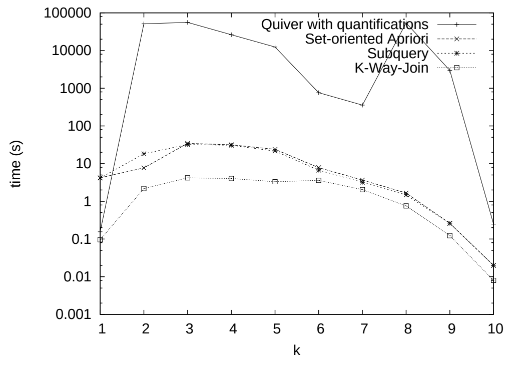
In Figure 7, we highlight some results of experiments on a commercial database system to assess the query execution plans and performance of the SQL-based algorithms K-Way-Join, Subquery, Set-oriented Apriori, and Quiver using quantifications. We used Microsoft SQL Server 2000 Standard Edition on a 4-CPU Intel Pentium-III Xeon PC with 900 MHz, 4 GB main memory, and Microsoft Windows 2000 Server. Due to lack of space, we cannot give the details on the indexes provided on the tables and what primary keys were chosen. However, it was surprising that K-Way-Join performed best for this (admittedly small) dataset, unlike reports in related work mentioned in Section 2.1. We provide more information on this comparison in [19].

5.2 XXL Query Execution Plans

In addition to the experiments on a commercial DBMS, we have executed the manual implementation of the QEPs with Java and XXL using one synthetic and one real-life dataset as the transactions table. Table 2 summarizes the characteristics of the datasets. The synthetic data has been produced using the *SDSU Java Library*, based on the well-known IBM data generation tool mentioned in [2]. The real-life transaction dataset called *BMS-WebView-2* by Blue Martini Software [26] contain several months of click-stream data of an e-commerce web site.



(a) Dataset characteristics



(b) Execution times. Note the logarithmic scale of the y-axis.

Figure 7: Experiments with SQL-based algorithms on a commercial DBMS for the synthetic dataset T5.I5.D10k with minimum support of 1% (100 transactions)

Type of data	Dataset	Distinct items	Rows	Trans-actions	Trans. size	
					avg.	max.
synthetic	T5.I5.D5k	86	30,268	5,000	6.05	18
real-life	BMS-WebView-2	3,340	358,278	77,512	4.62	161

Table 2: Overview of transaction datasets

5.2.1 *Set Containment Division*

The plans were executed on a dual-CPU Intel Pentium-III PC with 600 MHz, 256 MB main memory, Microsoft Windows 2000 Personal Edition, and JRE 1.4.1. The data resided on a single local hard disk.

We used subsets of the transaction datasets and subsets of the candidate 4-itemsets to derive frequent 4-itemsets for certain minimum support values. The synthetic (real-life) data candidates were generated for a minimum support of 1% (0.04%). Figure 8 shows some results of our experiments. One can observe that the execution plan using SCD performed best for small numbers of candidate sets. Of course, this result is disappointing but it is simply the result of our straightforward implementation of set containment division. Our implementation follows strictly the logical operator's definition, which is a union of several divisions: each division takes the entire transaction table as dividend and a single itemset group as divisor. We plan to develop more efficient implementations of the operator in the future.

The plan chosen by the commercial DBMS based on anti-

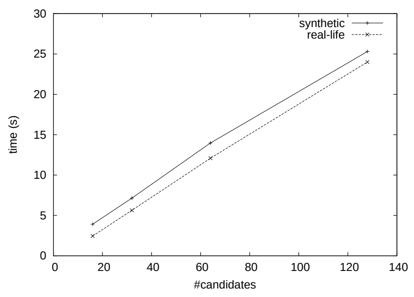


Figure 9: Experiments with set containment division for two types of data sets

semi-join was always a bad solution for the given data sets. K-Way-Join outperforms the other approaches for larger numbers of candidates.

5.2.2 Set Containment Join

In addition to the experiments comparing different query execution strategies, we have made initial performance tests using a set containment join operator instead of a set containment division, illustrated in Figure 9. We used the adaptive pick-and-sweep join algorithm. The query execution plan that we implemented using XXL is the same as in Figure 6(c), but a SCJ operator instead of the hash-based SCD. In the nested layout, the sets of items were represented by a Java vector of XXL tuples having a single item attribute. The test platform was a single-CPU Intel Pentium-4 PC with 1.9 GHz, 512 MB main memory, Microsoft Windows XP Professional SP 1, and JRE 1.4.1. The data resided on a single local hard disk. The experiments are based on the synthetic data set T5.I5.D10k and a subset of size 10,000 transactions of the real-life data set BMS-WebView-2. The candidate itemsets are similar to those used for the experiments before. For these small data sets, the operator processed the data with a linear performance for a growing number of candidate 4-itemsets.

6. CONCLUSIONS AND FUTURE WORK

We have shown that frequent itemset discovery is a prominent example for queries involving set containment tests. These tests can be realized by efficient algorithms for set containment join when the input data have set-valued attributes or by set containment division when the data are in 1NF. A DBMS has more options to solve the frequent itemset discovery problem optimally if it could choose to employ such operators inside the execution plans for some given data characteristics. No commercial DBMS offers an implementation of these operators to date. We believe that such a support would make data mining with SQL more attractive.

We currently realize query execution plans using algorithms for set containment joins and we compare them to plans involving set containment division. In future work, we will investigate also algorithms based on index structures supporting efficient containment tests, in particular the algorithms discussed in [8] and [11]. Furthermore, we will continue to study query processing techniques for switching between a vertical and a horizontal layout transparently (without changing the SQL statements) for data mining problems. Note that the investigation of the trade-offs of a horizontal and vertical data representation is not new. For example, the benefit of employing a vertical layout for querying e-commerce data has been observed in [1]. The techniques described there are generally applicable and we will study them in the context of the frequent itemset discovery problem.

7. ACKNOWLEDGMENTS

Daniel Böck's support of the performance experiments was greatly appreciated.

8. REFERENCES

[1] R. Agrawal, A. Somani, and Y. Xu. Storage and Querying of E-Commerce Data. In *Proceedings VLDB, Rome, Italy*, pages 149–158, September 2001.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings VLDB, Santiago, Chile*, pages 487–499, September 1994.

[3] J. V. d. Bercken, B. Blohsfeld, J.-P. Dittrich, J. Krämer, T. Schäfer, M. Schneider, and M. Seeger. XXL – A Library Approach to Supporting Efficient Implementations of Advanced Database Queries. In *Proceedings VLDB, Rome, Italy*, pages 39–48, September 2001.

[4] J. Chen and D. DeWitt. Dynamic Re-grouping of Continuous Queries. In *Proceedings VLDB, Hong Kong, China*, pages 430–441, August 2002.

[5] M. Gimbel, M. Klein, and P. Lockemann. Interactivity, Scalability and Resource Control for Efficient KDD Support in DBMS. In *Proceedings DTDM, Prague, Czech Republic*, pages 37–50, March 2002.

[6] G. Graefe and R. Cole. Fast Algorithms for Universal Quantification in Large Databases. *TODS*, 20(2):187–236, 1995.

[7] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[8] S. Helmer. *Performance Enhancements for Advanced Database Management Systems*. PhD thesis, University of Mannheim, Germany, December 2000.

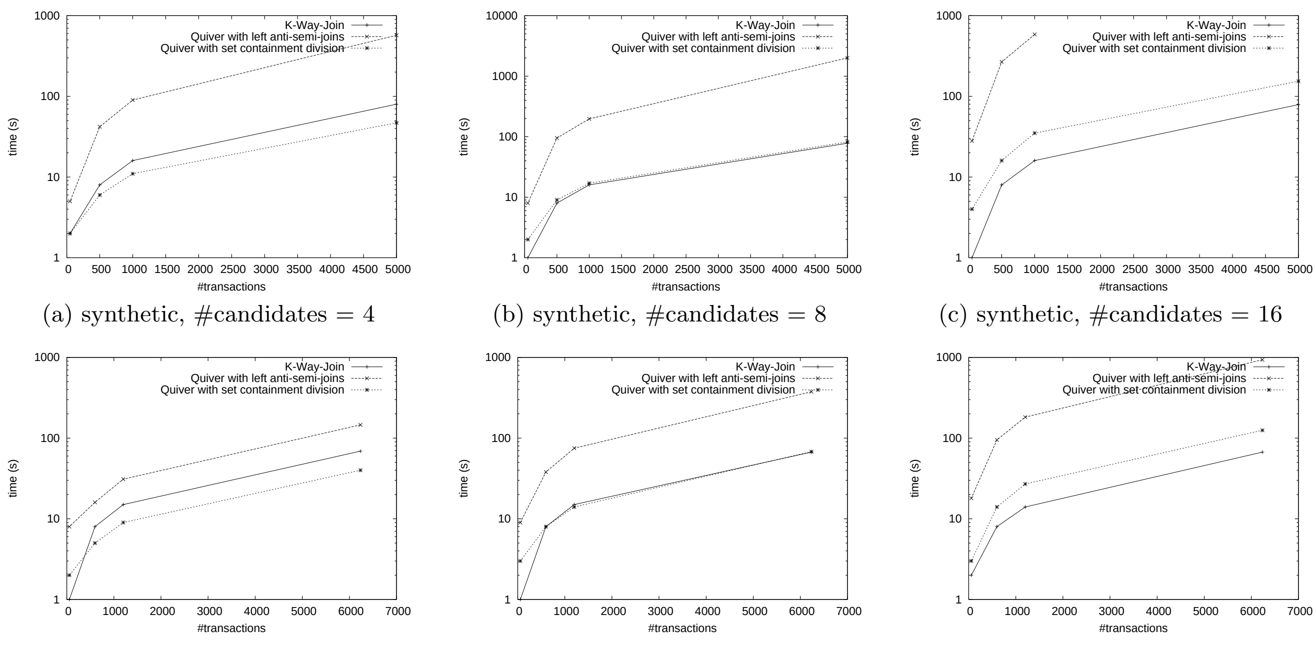
[9] S. Helmer and G. Moerkotte. Compiling Away Set Containment and Intersection Joins. Technical Report, University of Mannheim, Germany.

[10] M. Houtsma and A. Swami. Set-oriented Data Mining in Relational Databases. *DKE*, 17(3):245–262, December 1995.

[11] N. Mamoulis. Efficient Processing of Joins on Set-valued Attributes. In *Proceedings SIGMOD, San Diego, California, USA*, June 2003.

[12] W. Maniatty and M. Zaki. A Requirements Analysis for Parallel KDD Systems. In *Proceedings HIPS, Cancun, Mexico*, pages 358–365, May 2000.

[13] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient Algorithms for Discovering Association Rules. In *AAAI Workshop on Knowledge and Discovery in Databases, Seattle, Washington, USA*, pages 181–192, July 1994.



(a) synthetic, #candidates = 4 (b) synthetic, #candidates = 8 (c) synthetic, #candidates = 16

(d) real-life, #candidates = 4 (e) real-life, #candidates = 8 (f) real-life, #candidates = 16

Figure 8: Execution times of plans in Figure 6 for different numbers of candidate 4-itemsets and for two different datasets. Note the logarithmic scale of both axes.

[14] S. Melnik and H. Garcia-Molina. Divide-and-Conquer Algorithm for Computing Set Containment Joins. In *Proceedings EDBT, Prague, Czech Republic*, pages 427–444, March 2002.

[15] S. Melnik and H. Garcia-Molina. Adaptive Algorithms for Set Containment Joins. *TODS*, 28(1):56–99, March 2003.

[16] I. Pramudiono, T. Shintani, T. Tamura, and M. Kitsuregawa. Parallel SQL Based Association Rule Mining on Large Scale PC Cluster: Performance Comparison with Directly Coded C Implementation. In *Proceedings PAKDD, Beijing, China*, pages 94–98, April 1999.

[17] K. Ramasamy. *Efficient Storage and Query Processing of Set-valued Attributes*. PhD thesis, University of Wisconsin, Madison, Wisconsin, USA, 2002. 144 pages.

[18] K. Ramasamy, J. Patel, J. Naughton, and R. Kaushik. Set Containment Joins: The Good, The Bad and The Ugly. In *Proceedings VLDB, Cairo, Egypt*, pages 351–362, September 2000.

[19] R. Rantzau. Frequent Itemset Discovery with SQL Using Universal Quantification. In P. Lanzi and R. Meo, editors, *Database Support for Data Mining Applications*, volume 2682 of *LNCS*. Springer, 2003. To appear.

[20] R. Rantzau, L. Shapiro, B. Mitschang, and Q. Wang. Algorithms and Applications for Universal Quantification in Relational Databases. *Information Systems Journal, Elsevier*, 28(1):3–32, January 2003.

[21] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications. In *Proceedings SIGMOD, Seattle, Washington, USA*, pages 343–354, June 1998.

[22] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications. Research Report RJ 10107 (91923), IBM Almaden Research Center, San Jose, California, USA, March 1998.

[23] S. Thomas and S. Chakravarthy. Performance Evaluation and Optimization of Join Queries for Association Rule Mining. In *Proceedings DaWaK, Florence, Italy*, pages 241–250, August–September 1999.

[24] T. Yoshizawa, I. Pramudiono, and M. Kitsuregawa. SQL Based Association Rule Mining Using Commercial RDBMS (IBM DB2 UDB EEE). In *Proceedings DaWaK, London, UK*, pages 301–306, September 2000.

[25] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings SIGMOD, Santa Barbara, California, USA*, May 2001.

[26] Z. Zheng, R. Kohavi, and L. Mason. Real World Performance of Association Rule Algorithms. In *Proceedings SIGKDD, San Francisco, California, USA*, pages 401–406, August 2001.

Efficient OLAP Operations for Spatial Data Using Peano Trees

Baoying Wang Fei Pan Dongmei Ren Yue Cui Qiang Ding William Perrizo
Computer Science Department
North Dakota State University
Fargo, ND 58105
Tel: (701) 231-6257
{baoying.wang, fei.pan, Dongmei.ren, yue.cui, qiang.ding, william.perrizo}
@ndsu.nodak.edu

ABSTRACT
Online Analytical Processing (OLAP) is an important application of data warehouses. With more and more spatial data being collected, such as remotely sensed images, geographical information, digital sky survey data, efficient OLAP for spatial data is in great demand. In this paper, we build up a new data warehouse structure – PD-cube. With PD-cube, OLAP operations and queries can be efficiently implemented. All these are accomplished based on the fast logical operations of Peano Trees (P-Trees*). One of the P-tree variations, Predicate P-tree, is used to efficiently reduce data accesses by filtering out "bit holes" consisting of consecutive 0's. Experiments show that OLAP operations can be executed much faster than with traditional OLAP methods.

Keywords
Online analytical processing (OLAP), spatial data, P-Trees, PD-cube.

1. INTRODUCTION
Data warehouses (DWs) are collections of historical, summarized, non-volatile data, accumulated from transactional databases. They are optimized for Online Analytical Processing (OLAP) and have proven to be valuable for decision-making [9]. With more and more spatial data being collected, such as remote sensed images, geographical information, and digital sky survey data, it is important to study on-line analytical processing of spatial data warehouses.

The data in a warehouse are conceptually modeled as data cubes. Gray et al. introduce the data cube operator as a generalization of the SQL group by operator [3]. The chunk-based multi-way array aggregation method for data cube in OLAP was proposed in [4].

Typically, OLAP queries are complex and the size of the data warehouse is huge, which can cause queries to take very long to complete, if executed directly on raw data. This delay is unacceptable in most data warehouse environments.

Two major approaches have been proposed to efficiently process queries in a data warehouse: speeding up the queries by using index structures, and speeding up queries by operating on compressed data. Many different index structures have been proposed for high-dimensional data [1, 2]. Some special indexing techniques, such as bitmap index, join index and bitmap join index, have been introduced. The bitmap index uses a bit vector to represent the membership of tuples in a table. A significant advantage of bitmap index is that complex logical selection operations can be implemented very quickly by performing bit-wised *and*, *or*, and *complement* operators. However, bitmap indexes are space inefficient for high cardinality attributes. Bitmap indexes are only suitable for narrow domains, especially for membership functions with 0, or 1 as function values. The problem associated with sparse bitmap indexes is discussed in [5]. In order to overcome this problem, some improved bitmaps, e.g., encoded bitmap [7], have been proposed. The join indexes gained popularity from its use in relational database query processing. Join indexing is especially useful for maintaining the relationship between a foreign key and its matching primary keys, from the joinable relations.

Recently, a new data structure, Peano Tree (P-tree) [10, 11], has been introduced for spatial data. P-tree is a lossless, quadrant based compression data structure. It provides efficient logical operations that are well suited for multi-dimensional spatial data. One of the P-tree variations, Predicate P-tree, is used to efficiently reduce data accesses by filtering out "bit holes" which consist of consecutive 0's.

In this paper, we present a new data warehousing structure, PD-cube, to facilitate OLAP operations and queries. Fast logical operations on P-Trees are used to accomplish these OLAP operations. One of the P-tree variations, Predicate P-tree, is used to efficiently reduce data accesses by filtering out "bit holes" consisting of consecutive 0's. Experiments show that OLAP operations can be executed much faster this way than with traditional data cube methods.

This paper is organized as follows. In section 2, we review P-tree structure and its operations. In section 3, we propose a new data warehouse structure, PD-cube, and efficient OLAP operations using P-Trees. Finally, we compare our method with traditional data cube methods experimentally in section 4 and conclude the paper in section 5.

2. REVIEW OF PEANO TREES
A quadrant-based tree structure, called the Peano Tree (P-tree), was developed to facilitate compression and very fast logical operations of bit sequential (bSQ) data [10]. P-Trees can be 1-dimensional, 2-dimensional, 3-dimensional, etc.

The most useful form of a P-tree is the *predicate*-P-tree in which a 1-bit appears at those tree nodes corresponding to quadrants for

* Patents are pending on the P-tree technology. This work is partially supported by GSA Grant ACT#: K96130308.

which the predicate holds. The predicate can be a particular bit-position of a particular attribute or, more generally, a set of values for each of a set of attributes. Pure 1 P-tree (P1-tree) and Non Pure0 P-tree (NP0-tree) are two predicate P-Trees. In Figure 1, a bSQ file with 64 rows is shown, the file is rearranged into 2-D Peano or Z order in the left and the P-Trees (P1-tree and NP0-tree) are on the right. We identify quadrants using a Quadrant identifier, Qid - the string of successive sub-quadrant numbers (0, 1, 2 or 3 in Z or Peano order, separated by "." (as in IP addresses). Thus, the Qid of the bolded and underlined quadrant in Figure 1 is 2.2.

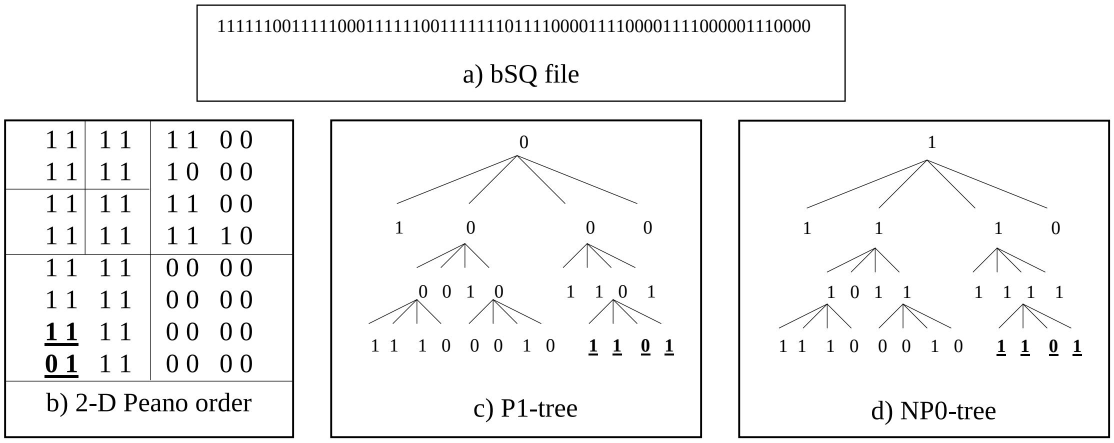
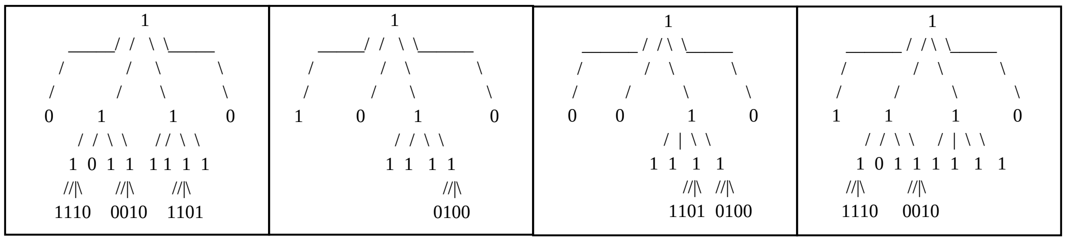


Figure 1. bSQ file, 2-D Peano order bSQ file, P1-tree, and NP0-tree

Logic *and* and *or* are the most important and frequently used P-tree logical operations. By using predicate P-tree operations, we filter "big holes" consisting of consecutive 0's and get the mixed quadrants. Then we only load the mixed quadrants of Peano sequence into main memory, thus reducing the data accesses. The *and* and *or* operations using NP0-tree are illustrated in Figure 2.



a). NP0-tree1 b). NP0-tree2 c). AND Result d). OR Result

Figure 2. P-tree Operations: and and or

In Figure 2, a) and b) are two NP0-trees, c) and d) are their *and* result and *or* result, respectively. From c), we know that bits at the range of position [0, 32] and [49, 64] are pure 0 since the bits in quadrant with Qid 0, 1 and 3 are 0. Quadrants with Qid 2 are non-pure0 parts. We only need to load quadrants with Qid 2. The logical operation results calculated in such way are the exactly same as anding two bit sequence with reduced data accesses. There are several ways to perform P-tree logical operations. The basic way is to perform operations level by level starting from the root level. The node operation is commutative.

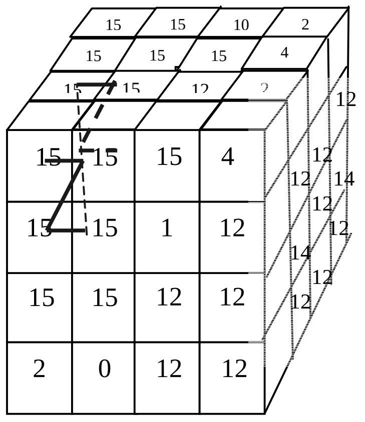
3. OLAP OPERATIONS USING P-TREES

In this section, we first develop a new data warehousing structure, PD-cube, which is the bit-wised representation of data cube. Then we show that OLAP operations, such as slice/dice and roll-up, based on PD-cube, can be efficiently implemented using P-Trees.

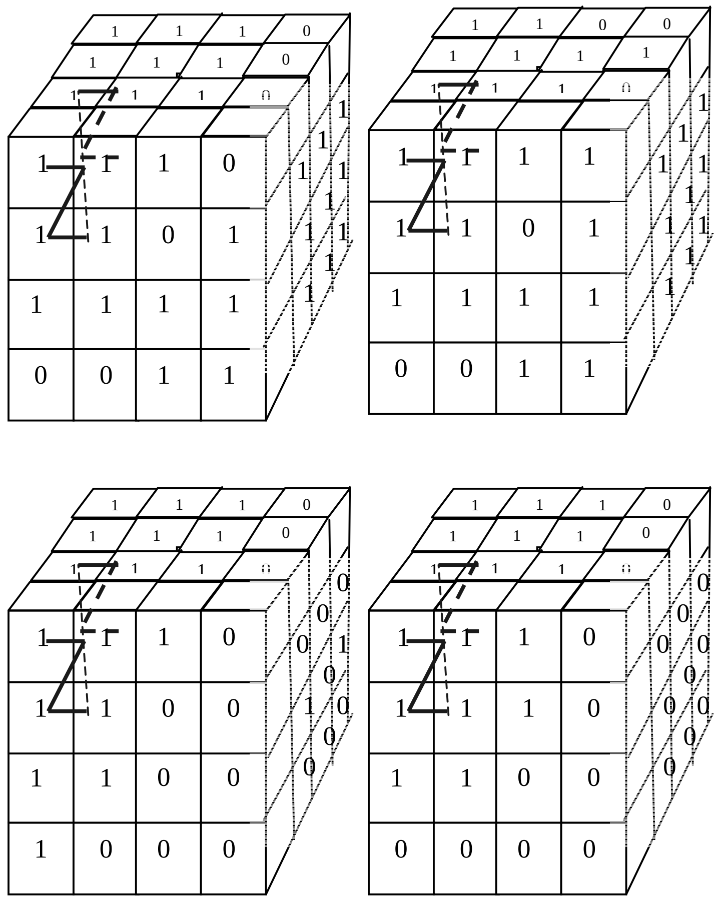
3.1 PD-cube

To take advantage of the continuity and sparseness in spatial data, we develop a new data warehouse structure, PD-cube. It is a bit-wised data cube in Peano order which is consistent with P-Trees in terms of data storage. We partition the data cube into bit level in order to facilitate fast logical operations of predicate P-Trees. Predicate P-Trees are tree-based bitmap indexes built on quadrants. By means of fast logical and, or and not operations of predicate P-Trees, PD-cubes achieve efficient data access, which is very important for massive spatial data set.

X	Y	T	Yeild
0	0	0	15 (1111)
1	0	0	15 (1111)
0	1	0	15 (1111)
1	1	0	15 (1111)
0	0	1	15 (1111)
1	0	1	15 (1111)
0	1	1	15 (1111)
1	1	1	15 (1111)
2	0	0	15 (1111)
3	0	0	4 (0100)
2	1	0	1 (0001)
3	1	0	12 (1100)
2	0	1	12 (1100)
3	0	1	2 (0010)
2	1	1	12 (1100)
3	1	1	12 (1100)
0	2	0	15 (1111)
1	2	0	15 (1111)
0	3	0	2 (0010)
1	3	0	0 (0000)
0	2	1	15 (1111)
1	2	1	15 (1111)
0	3	1	2 (0010)
1	3	1	0 (0000)
2	2	0	12 (1100)
3	2	0	12 (1100)
2	3	0	12 (1100)
3	3	0	12 (1100)
2	2	1	12 (1100)
3	2	1	12 (1100)
2	3	1	12 (1100)
3	3	1	12 (1100)
0	0	2	15 (1111)
1	0	2	15 (1111)
0	1	2	15 (1111)
1	1	2	15 (1111)
0	0	3	15 (1111)
1	0	3	15 (1111)
0	1	3	15 (1111)
1	1	3	15 (1111)
2	0	2	15 (1111)
3	0	2	10 (1010)
2	1	2	1 (0001)
3	1	2	14 (1110)
2	0	3	14 (1110)
3	0	3	2 (0010)
2	1	3	12 (1100)
3	1	3	12 (1100)
0	2	2	15 (1111)
1	2	2	15 (1111)
0	3	2	2 (0010)
1	3	2	0 (0000)
0	2	3	15 (1111)
1	2	3	15 (1111)
0	3	3	2 (0010)
1	3	3	0 (0000)
2	2	2	12 (1100)
3	2	2	12 (1100)
2	3	2	12 (1100)
3	3	2	12 (1100)
2	2	3	12 (1100)
3	2	3	12 (1100)
2	3	3	12 (1100)
3	3	3	12 (1100)



(b) Data Cube of Field Yield



(a) Fact Table of Yield in Peano Order

(c) PD-cubes

Figure 3. A Fact Table, its Data Cube and PD-cubes

Figure 3 shows an example of a 3-D data cube representing the crop yield. It has three dimensions: position x (X), position y (Y), and time (T). Since yield is a 4-bit value, we split the data cube of yield into four bit-wised cubes, which are stored in form of predicate P-Trees NP0 and P1.

3.2 OLAP Operations

The OLAP operations are the basis for answering spatial data warehouse questions like: “find all galaxies brighter than magnitude 22”, “find average yield of field”, or “find the total traffic flow during a period of time.” Slice/dice and roll-up operations will be described as follows.

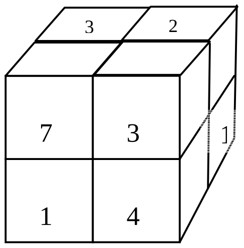
3.2.1 Slice/dice

The Slice operation is a selection along one dimension, while the dice operation defines a sub-cube by performing a selection on two or more dimensions. Slice and Dice are very efficiently accomplished using P-Trees. Typical select statements may have a number of predicates in their “where” clause that must be combined in a Boolean manner. The predicates may include “=”, “<” and “>”. These predicates lead to two different query scenarios, where “=” clause results in an equal query, and “<” or “>” in a range query.

3.2.1.1 Equal Select Slice

Suppose we have a 3-D data cube representing crop yield with dimensions X, Y and T, where X = {0, 1}, Y = {0, 1} and T = {0, 1}. Attribute “Yield” is converted from decimal to binary. The data cube and its corresponding relational table are shown in Figure 4.

From the table in Figure 4 we build 6 P-Trees as shown in Figure 5 according to the P-tree construction rule. Pi, j, is a P-tree for the jth bit of the ith attribute. For attribute of m bit, there are m P-trees, one for each bit position



X	Y	T	Yield
0	0	0	111
1	0	0	011
0	1	0	001
1	1	0	100
0	0	1	011
1	0	1	010
0	1	1	100
1	1	1	001

Figure 4. A 3-D Cube and Binary Relational Table

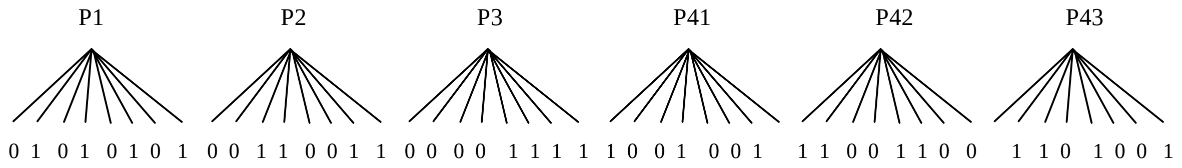


Figure 5. P-Trees for 3-D cube Quadrant

Suppose we want to know that yield at the place where Y = 1. The query is “get yield where Y = 1”. The process of selection is: first get P-tree masks, and then trim the P-Trees. The detailed process is discussed as follows.

First, create the P-tree mask, PM = P2. Then use PM to generate a new set of P-Trees. The new P-tree set is the result of selection.

Generation of new P-Trees is simple: just trim the nodes corresponding to the position of pure 0 in PM, and PM is P2 here. Take P43 for example, the trimming process is shown as Figure 6.

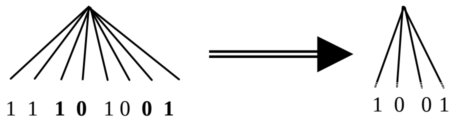


Figure 6. Trim Process of P-tree based on PM

After trimming the whole P-tree set, we can convert the trimmed P-tree set easily to a data cube as shown in Figure 7.

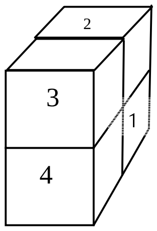


Figure 7. Result Cube of Selection

3.2.1.2 Range Slice

Typical range queries for spatial data warehouse include questions like: “Get crop yield within the range of X > 200 and Y > 150”. We analyze three range slice cases as follows.

a. Get yield where Y >1001.

Given the number of bits for the second attribute Y is fixed as 4, we can easily get the values that are greater than 1001. From Figure 8 , we can see that {"Y > 1001"} consists of two subsets {"Y = 11**"} and {"Y = 101*"}, where * is 1 or 0. Therefore the query clause can be written as "where Y = 11** || Y = 101*".

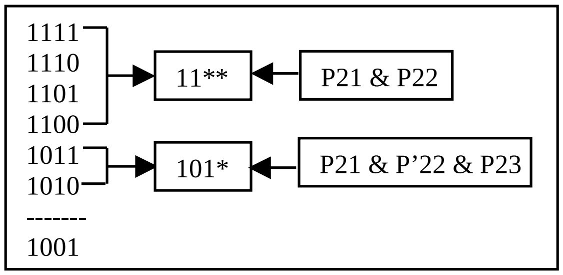


Figure 8. Range Query of the Second Attribute Y > 1001

The query is retrieved by the corresponding P-tree mask PM_{gt} = PM1 || PM2. PM1 is predicate P-tree for data that start with "11" and PM2 is for data that start with "101". Since Y is the second attribute of the Peano Cube, we have

PM1 = P21 & P22
PM2 = P21 & P'22 & P27.

where

P21 – the predicate P-tree for data that has value 1 in the first bit position of the second attribute

P'22 – the predicate P-tree for data that has value 0 in the second bit position of the second attribute

P22, P27 – follows the same rule.

b. Get yield where T > 01011001.

The retrieval process of this query is shown in Figure 9. Attribute T is the third dimension of the PD-Cube. Data set {T > 01011001} can be divided a number of subsets: {"1*******"}, {"011*****"}, {"010111**"} and {"0101101*"}, where * is 1 or 0. The retrieval process can be generalized as follows:

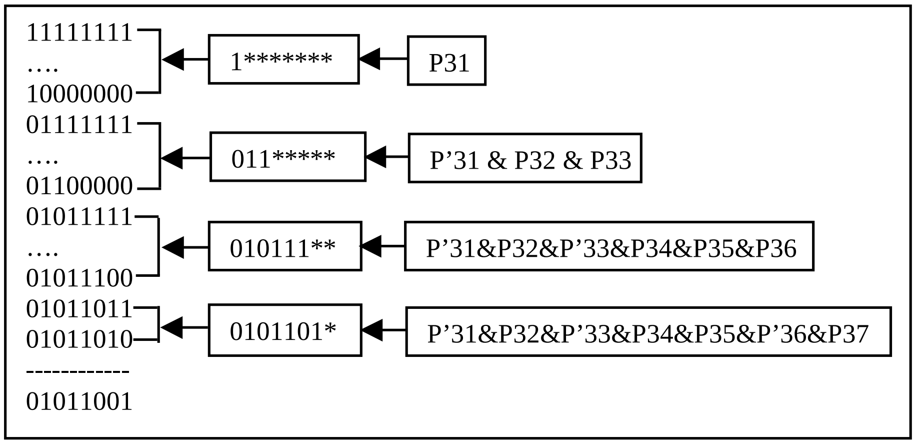


Figure 9. Range Query of the Third Dimension T > 01011001

- The number of subsets is equal to the number of 0 in the bit stream. Data set {T > 01011001} consists of four subsets.
- Start with the highest bit of string until the nth 0, revert it to 1 and append it with wildcard *, we get the first subset. The first subset of {T > 01011001} is {"1*******"}, the second subset is {"011*****"}, and so on.
- Generate mask P-tree for each subset according to bit pattern and position as discussed in previous example. For this example PM1 = P31, PM2 = P31' & P32 & P33, PM3 = P31' & P32 & P33' & P34 & P35 & P36 and PM4 = P31' & P32 & P33' & P34 & P35 & P36' & P37.
- AND all mask P-Trees together to get a general mask P-tree. For this example PM_{gl} = PM1 || PM2 || PM3 || PM4.

c. Combination of Equal Query and Range Query

It is often the case that "where" clause has ">=". For example, "get yield where T >= 01011001". We can easily divide data set {"T >=01011001"} into two subsets, {"T >01011001"} and {"T = 01011001"}. Therefore the mask P-tree PM_{ge} = PM_{gt} || PM_{eq}. PM_{gt} is related to range query and PM_{eq} is related to equal query.

For "where" clause like "01111011 > T > 01011001", we also need to divide set {"01111011 > T > 01011001"} into two subsets, {"01111011 > T"} and {"T > 01011001"}. The mask P-tree PM_{gl} = PM_{gt} || PM_{lt}.

d. Complement of Range Query

We have just analyzed "where" clause with ">" or ">=". But we also want "<" or "<=". For example, "get yield where T <= 01011001." Obviously, data set {"T <= 01011001"} is the complement of {"T > 01011001"}. With the result of query "Get yield where T > 01011001, we can easily retrieve query "Get yield where T <= 01011001" by making complement, i.e. PM_{le} = PM'_{gt}.

Dice operation is similar to slice operation. Only dice involves more than one dimension while slice is on one dimension. The example of dice can be "get yield where T <= 01011001" & "Y > 1001." We just need to *and* P-tree masks of slice on dimension T and Y to get the final P-tree for dice.

3.2.2 Rollup

PD-cube is stored in Peano order rather than in raster order. Therefore, the rollup of PD-cube is accomplished differently from the traditional data cube as a result of different storage models.

According to the Peano storage of PD-cube, we develop the rollup algorithm in Figure 11. With this algorithm, we can easily calculate aggregation along any dimension.

The rollup process we have discussed so far is for one cube. As described in section 3.1, PD-cubes are bitwise data cubes. The attribute "Yield" in Figure 4 are composed of three PD-cubes (See Figure 10).

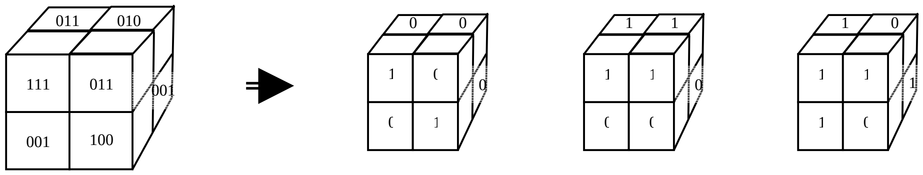


Figure 10. PD-cubes for Attribute "Yield"

```
// d – number of dimension of the data cube
// n – size of the data cube;  kd – the dimension to roll up along
// R[] – the result of roll up;  start – the start index of R[]
// val – the values in the data cube at the cursor position
// Skip - move cursor forward within the Peano cube
Algorithm rollup (d, n, kd, R[], R_start)
        n = 1:  R[start] += val
        n = $2^d$:
            For i = start TO (start + n/2-1) DO
                Skip (i-start) % $2^{kd-1}$ + $2^{kd}$ * $\lfloor (i-start)/2^{kd-1} \rfloor$
                R[i] += val
                Skip $\sum_{j=0}^{kd-2} 2^j$
                R[i] += val
        n > $2^d$:
            For i = 0 TO $2^{d-1}$-1 DO
                ssc = n / $2^d$          // ssc: size_sub_quadrant
                ddm = $\sqrt[d]{ssc}$          // ddm: domain_of_dimension
                Skip ((i + start) % $2^{kd-1}$ +
                        $2^{kd}$ * $\lfloor (i*+start)/2^{kd-1} \rfloor$ )*ssc
                rollup (d, n/$2^d$, kd, R[], start + i*ssc/ddm );
                Skip ssc + $\sum_{j=0}^{kd-2} 2^j * ssc$
                rollup (d, n/$2^d$, kd, R[], start + i*ssc/ddm );
End rollup
```

Figure 11. Rollup Algorithm for PD-cube

Suppose we want to aggregate "Yield" along dimension T (pointing down). First we need to rollup each cube using the algorithm in Figure 11, and get the results, S2 [ ], S1[ ] and S0[ ] as shown in Figure 12. With the rollup results of each cube, S2, S1, and S0, we can get rollup results, S, as follows:
$S[i] = S2[i] \times 2^2 + S1[i] \times 2^1 + S0[i] \times 2^0$
For i=1, $S[1] = 1 \times 2^2 + 1 \times 2^1 + 2 \times 2^0 = 8$. Similarly we can get S[2] = 7, S[3] = 7 and S[4] = 7.

4. PERFORMANCE ANALYSIS

We have implemented PD-cube and conducted a series of experiments to validate our performance analysis of the algorithm. Our experiments are implemented in C++ on a 1GHz Pentium PC machine with 1GB main memory. The test data includes the aerial TIFF images of the Oakes Irrigation Test Area in North Dakota. The data is prepared in five sizes, 128x128, 256x256, 512x512, 1024x1024, and 2048x2048. The data sets are available at [12].

In this experiment, we compare our algorithm with bitmap indexed data cube method. The response time is calculated based on average CPU time for 20 queries. The goal is to validate our claim that PD-cube and P-trees have high scalability in terms of data size. The experiment results are shown in 0.

As it is shown in Figure 13, when cube size > 1900KB, our method outperforms bitmap indexed data cube method. As the cube size increases, we can see the drastic increase in response time for bitmap indexed data cube method. The P-tree method shows its great strength for big cube sizes. In summary, our experiments show that our algorithm with PD-cube data structure and P-tree algorithm demonstrate significant improvement in performance and scalability over bitmap indexed data cube method for spatial data.

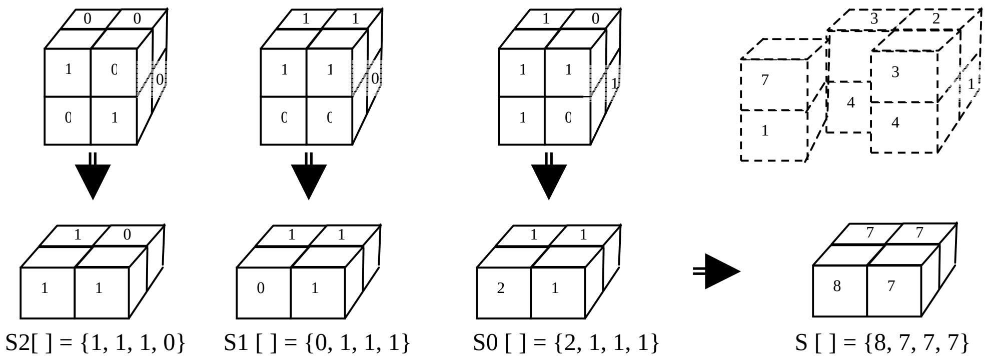


Figure 12. Rollup of "Yield" along Dimension T

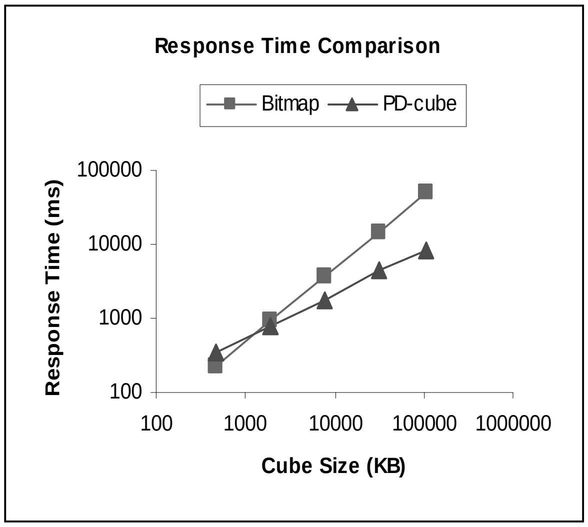


Figure 13. Response Time Comparison

5. CONCLUSION

In this paper, we present a general data warehousing structure, PD-cube, to facilitate OLAP operations. The fast logical operations of P-Trees are used to accomplish these operations. Predicate P-Trees are used to find the "big holes" of consecutive 0's by performing logical operations. Only the mixed chunks need to be loaded, thus the amount of data scans is reduced. Experiments indicate OLAP operations using P-Trees is much faster than traditional data cube methods.

One future research work is to extend our PD-cube into parallel data warehouse system. It appears particularly promising by partitioning a large cube horizontally or vertically into small cubes that can improve the query performance by parallelism. Besides, our method also has potential applications in other areas, such as DNA micro array and medical image analysis.

6. REFERENCES

[1] K. Lin, H. V. Jagadish, C. Faloutsos. *The TV-Tree: An Index Structure for High-Dimensional Data,* VLDB Journal, Vol. 7, pp. 517-542, 1995.

[2] S. Berchtold, D. Keim, H. P. Kriegel. *The X-Tree: An Index Structure for High-Dimensional Data*, Proc. 22nd Int. Conf. on Very Large Databases (VLDB), 1996, pp. 28-79.

[3] J. Gray, A. Bosworth, A. Layman, H. Pirahesh. *Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-tab, and Sub-Totals.* Data Mining and Knowledge Discovery 1, 29-57, 1997

[4] Y. Zhao, P. M. Deshpande, and J. F. Naughton. *An array-based algorithm for simultaneous multidimensional aggregates.* In Proc. 1997 ACM-SIGMOD Int, Conf. Management of Data (SIGMOD'97), pages 159-170, Tucson, AZ, May 1997.

[5] P. O'Neil, G. Graefe. *Multi-Table Joins Through Bitmapped Join Indices*, SIGMOD Record 24(7), September 1995

[6] P. Valduriez, *Join Indices*, ACM TODS, 12(2), June 1987.

[7] M.C. Wu, A. Buchmann. *Encoded Bitmap Indexing for Data Warehouses.* Proceedings of the 14th International Conference on Data Engineering, Orlando, Florida, USA, Feb. 1998, pp. 220--270.

[8] S. Amer-Yahia, T. Johnson. *Optimizing Queries on Compressed Bitmaps.* VLDB 2000: 729-778

[9] E. Codd, S. Codd, C. Salley. *Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate. Technical Report.* 1997, available at http://www.arborsoft.com/essbase/wht_ppr/coddps.zip.

[10] W. Perrizo, *Peano Count Tree Technology,* Technical Report NDSU-CSOR-TR-01-1, 2001.

[11] M. Khan, Q. Ding, W. Perrizo. *k-Nearest Neighbor Classification on Spatial Data Streams Using P-Trees*, PAKDD 2002, Spriger-Verlag, LNAI 2776, 2002, pp. 517-528.

[12] TIFF image data sets. Available at http://midas-10cs.ndsu.nodak.edu/data/images/.

Clustering Gene Expression Data in SQL Using Locally Adaptive Metrics

Dimitris Papadopoulos
UC Riverside
dimitris@cs.ucr.edu

Carlotta Domeniconi
George Mason University
carlotta@ise.gmu.edu

Dimitrios Gunopulos*
UC Riverside
dg@cs.ucr.edu

Sheng Ma
IBM T.J. Watson Research Center
shengma@us.ibm.com

ABSTRACT

The clustering problem concerns the discovery of homogeneous groups of data according to a certain similarity measure. Clustering suffers from the curse of dimensionality. It is not meaningful to look for clusters in high dimensional spaces as the average density of points anywhere in input space is likely to be low. As a consequence, distance functions that equally use all input features may be ineffective. We introduce an algorithm that discovers clusters in subspaces spanned by different combinations of dimensions via local weightings of features. This approach avoids the risk of loss of information encountered in global dimensionality reduction techniques. Our method associates to each cluster a weight vector, whose values capture the relevance of features within the corresponding cluster. In this paper we present an efficient SQL implementation of our algorithm, that enables the discovery of clusters on data residing inside a relational DBMS.

1. INTRODUCTION

The clustering problem concerns the discovery of homogeneous groups of data according to a certain similarity measure. It has been studied extensively in statistics [2], machine learning [5; 13], and database communities [14; 10; 20]. Clustering suffers from the curse of dimensionality problem in high dimensional spaces. In high dimensional spaces, it is highly likely that, for any given pair of points within the same cluster, there exist at least a few dimensions on which the points are far apart from each other. It is not meaningful to look for clusters in such a high dimensional space as the average density of points anywhere in input space is likely to be low. As a consequence, distance functions that equally use all input features may be ineffective.

The problem of high dimensionality can be addressed by requiring the user to specify a subspace (i.e., subset of dimensions) for cluster analysis. However, the identification of subspaces by the user is an error-prone process. More importantly, correlations that identify clusters in the data are likely not to be known by the user. Indeed, we desire such correlations, and induced subspaces, to be part of the findings of the clustering process itself.

In order to capture the local correlations of data, a proper feature selection procedure should operate locally in input space. In this paper we propose a *soft* feature selection procedure that assigns (local) weights to features according to the local correlations of data along each dimension. Dimensions along which data are loosely correlated receive a small weight, that has the effect of elongating distances along that dimension. Features along which data are strongly correlated receive a large weight, that has the effect of constricting distances along that dimension. Figures 1 give a simple example. The left plot depicts two clusters of data elongated along the x and y dimensions. The right plot shows the same clusters, where within-cluster distances between points are computed using the respective local weights generated by our algorithm. The weight values reflect local correlations of data, and reshape each cluster as a *dense spherical cloud*. This directional local reshaping of distances better separates clusters, and allows for the discovery of different patterns in different subspaces of the original input space.

The data-mining task of clustering ideally can be performed inside a relational DBMS. By relying on a DBMS to manage the data, the clustering application is freed from this task. Also, implementing a clustering algorithm in SQL, makes the programming task easier. Finally, since SQL is an industry standard and available on all major DBMS's, the implementation of the clustering algorithm is portable.

In this paper we formalize the problem of finding different clusters in different subspaces. Our algorithm (Locally Adaptive Clustering, or LAC) discovers clusters in subspaces spanned by different combinations of dimensions via local weightings of features. This approach avoids the risk of loss of information encountered in global dimensionality reduction techniques. We give a concrete SQL implementation of our algorithm, that exhibits good performance and scalability. These are the key considerations for deploying a clustering algorithm inside a DBMS, especially when dealing with very large datasets that must reside in disk space, or if we want to take advantage of computer clusters.

2. RELATED WORK

Local dimensionality reduction approaches for the purpose of efficiently indexing high dimensional spaces have been recently discussed in the database literature [12; 4; 17]. In general, the efficacy of these methods depends on how the clustering problem is addressed in the first place in the original feature space. A potential serious problem with such techniques is the lack of data to locally perform PCA on each cluster to derive the principal components. Moreover, for clustering purposes, the new dimensions may be diffi-

*Supported by NSF CAREER Award 9984729,NSF IIS-9907477, and the DoD.

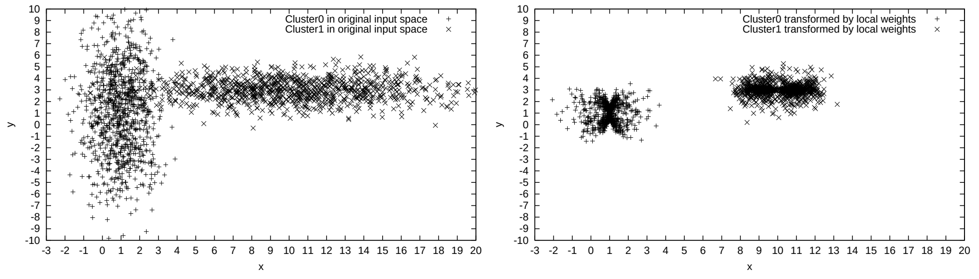


Figure 1: (Left)Clusters in original input space. (Right) Clusters transformed by local weights.

cult to interpret, making it hard to understand clusters in relation to the original space.

The problem of finding different clusters in different subspaces of the original input space has been addressed in [16]. While the work in [16] successfully introduces a methodology for looking at different subspaces for different clusters, it does not compute a partitioning of the data into disjoint groups. The reported dense regions largely overlap, since for a given dense region all its projections on lower dimensionality subspaces are also dense, and they all get reported. On the other hand, for many applications such as customer segmentation and trend analysis, a partition of the data is desirable since it provides a clear interpretability of the results.

Recently [15], another density-based projective clustering algorithm (DOC) has been proposed. This approach pursues an optimality criterion defined in terms of density of each cluster in its corresponding subspace. A Monte Carlo procedure is then developed to approximate with high probability an optimal projective cluster.

[9] also addresses the problem of feature selection to find clusters hidden in high dimensional data. The authors search through feature subset spaces, evaluating each subset by first clustering in the corresponding subspace, and then evaluating the resulting clusters and feature subset using the chosen feature selection criterion. We observe that dimensionality reduction is performed globally in this case. Therefore, the technique in [9] is expected to be effective when a data set contains some relevant features and some irrelevant (noisy) ones, across all clusters.

The problem of finding different clusters in different subspaces is also addressed in [1]. The proposed algorithm (PROjected CLUStering) seeks subsets of dimensions such that the points are closely clustered in the corresponding spanned subspaces. Both the number of clusters and the average number of dimensions per cluster are user-defined parameters. PROCLUS starts with choosing a random set of medoids, and then progressively improves the quality of medoids by performing an iterative hill climbing procedure that discards the 'bad' medoids from the current set. In order to find the set of dimensions that matter the most for each cluster, the algorithm selects the dimensions along which the points have the smallest average distance from the current medoid. The authors do not prove whether or not the algorithm converges to the optimality criterion they choose.

In contrast to the PROCLUS algorithm, our method does not require to specify the average number of dimensions to be kept per cluster. For each cluster, in fact, *all* features are taken into consideration, but properly weighted. The PROCLUS algorithm is more prone to loss of information if the number of dimensions is not properly chosen. For example, if data of two clusters in two dimensions are distributed as in Figure 1 (Left), PROCLUS may find that feature x is the most important for cluster 0, and feature y is the most important for cluster 1. But projecting cluster 1 along the y dimension doesn't allow to properly separate points of the two clusters. We avoid this problem by keeping both dimensions for both clusters, and properly weighting distances along each feature within each cluster. In addition, while the authors in [1] do not prove whether their algorithm converges to the chosen optimality criterion, we can formally show that our algorithm converges to a local minimum of the associated error function.

Generative approaches have also been developed for local dimensionality reduction and clustering. The approach in [11] makes use of maximum likelihood factor analysis to model local correlations between features.

[18] extends the single PCA model to a mixture of local linear sub-models to capture nonlinear structure in the data. A mixture of principal component analyzers model is derived as a solution to a maximum-likelihood problem. An EM algorithm is formulated to estimate the parameters.

3. BICLUSTERING OF GENE EXPRESSION DATA

Microarray technology is one of the latest breakthroughs in experimental molecular biology. Gene expression data are generated by DNA chips and other microarray techniques, and they are often presented as matrices of expression levels of genes under different conditions (e.g., environment, individuals, tissues). Each row corresponds to a gene, and each column represents a condition under which the gene is developed.

Biologists are interested in finding set of genes showing strikingly similar up-regulation and down-regulation under a set of conditions. To this extent, recently the concept of *bicluster* has been introduced [6]. A bicluster is a subset of genes and a subset of conditions with a high similarity score. A particular score that applies to expression data is the *mean squared residue score* [6]. Let I and J be subsets of genes and experiments respectively. The pair (I, J) specifies a submatrix A_{IJ} with a mean squared residue score defined as follows:

$$H(I,J) = \frac{1}{|I||J|} \sum_{i\in I, j\in J} (a_{ij} - a_{iJ} - a_{Ji} + a_{IJ})^2, \qquad (1)$$

where $a_{iJ} = \frac{1}{|J|}\sum_{j\in J} a_{ij}$, $a_{Ij} = \frac{1}{|I|}\sum_{i\in I} a_{ij}$, and $a_{IJ} =$

$\frac{1}{|I||J|}\sum_{i\in I,j\in J} a_{ij}$. They represent the row and column means, and the mean of the submatrix, respectively. The lowest score $H(I,J)=0$ indicates that the gene expression levels fluctuate in unison. The aim is then to find biclusters with low mean squared residue score (below a certain threshold).

We observe that the mean squared residue score is minimized when subsets of genes and experiments (or dimensions) are chosen so that the gene vectors (i.e., rows of the resulting bicluster) are close to each other with respect to the Euclidean distance. As a result, the LAC algorithm, and other subspace clustering algorithms, are well suited to perform simultaneous clustering of both genes and conditions in a microarray data matrix.

[19] introduces an algorithm (pCluster) for clustering similar patterns, that has been applied to DNA microarray data of a type of yeast. The pCluster model optimizes a criterion that is different from the mean squared residue score, as it looks for coherent patterns on a subset of dimensions (e.g., in an identified subspace, objects reveal larger values for the second dimension than for the first).

4. PROBLEM STATEMENT

We define what we call *weighted cluster.* Consider a set of points in some space of dimensionality N. A *weighted cluster* C is a subset of data points, together with a vector of weights $\mathbf{w}=(w_1,\ldots,w_N)$, such that the points in C are closely clustered according to the L_2 norm distance weighted using $\mathbf{w}$. The component w_j measures the degree of correlation of points in C along feature j. The problem becomes now how to estimate the weight vector $\mathbf{w}$ for each cluster in the data set.

In this setting, the concept of *cluster* is not based only on points, but also involves a weighted distance metric, i.e., clusters are discovered in spaces transformed by $\mathbf{w}$. Each cluster is associated with its own $\mathbf{w}$, that reflects the correlation of points in the cluster itself. The effect of $\mathbf{w}$ is to transform distances so that the associated cluster is reshaped into a dense hyper-sphere of points separated from other data.

In traditional clustering, the partition of a set of points is induced by a set of *representative* vectors, also called *centroids* or *centers.* The partition induced by discovering weighted clusters is formally defined as follows.

Definition: Given a set S of D points $\mathbf{x}$ in the N-dimensional Euclidean space, a set of k centers $\{\mathbf{c}_1,\ldots,\mathbf{c}_k\}$, $\mathbf{c}_j\in\Re^N$, $j=1,\ldots,k$, coupled with a set of corresponding weight vectors $\{\mathbf{w}_1,\ldots,\mathbf{w}_k\}$, $\mathbf{w}_j\in\Re^N$, $j=1,\ldots,k$, partition S into k sets $\{S_1,\ldots,S_k\}$:

$$S_j=\{\mathbf{x}|(\sum_{i=1}^N w_{ji}(x_i-c_{ji})^2)^{1/2}<(\sum_{i=1}^N w_{li}(x_i-c_{li})^2)^{1/2},\forall l\neq j\}, \tag{2}$$

where w_{ji} and c_{ji} represent the ith components of vectors $\mathbf{w}_j$ and $\mathbf{c}_j$ respectively (ties are broken randomly).

The set of centers and weights is *optimal* with respect to the Euclidean norm, if they minimize the error measure:

$$E_1(C,W)=\sum_{j=1}^k\sum_{i=1}^N w_{ji}e^{-X_{ji}} \tag{3}$$

subject to the constraints $\sum_{i=1}^N w_{ji}^2=1\ \forall j$. C and W are $(N\times k)$ matrices whose column vectors are $\mathbf{c}_j$ and $\mathbf{w}_j$ respectively, i.e. $C=[\mathbf{c}_1\ldots\mathbf{c}_k]$ and $W=[\mathbf{w}_1\ldots\mathbf{w}_k]$. X_{ji} represents the average distance from the centroid $\mathbf{c}_j$ of points in cluster j along dimension i, and is defined as follows:

$$X_{ji}=\frac{1}{|S_j|}\sum_{\mathbf{x}\in S_j}(c_{ji}-x_i)^2.$$

The exponential function in (3) has the effect of making the weights w_{ji} more sensitive to changes in X_{ji}, and therefore to changes in local feature relevance. This allows larger error improvements as we adapt the values of weights and centers, and therefore a faster computation to achieve spherically shaped clusters (separated from each other) in the space transformed by optimal weights (see Figure 1).

In the following we present an algorithm that finds a solution (set of centers and weights) that is a local minimum of the error function (3).

5. LOCALLY ADAPTIVE CLUSTERING ALGORITHM

This section describes our feature relevance estimation procedure that assigns local weights to features according to the local correlation of data along each dimension. Our technique progressively improves the quality of initial centroids and weights, by investigating the space near the centers to estimate the dimensions that matter the most, i.e. the dimensions along which local data are mostly correlated. Specifically, we proceed as follows.

We start with *well-scattered* points in D as the k centroids: we choose the first centroid at random, and select the others so that they are far from one another, and from the first chosen center. We initially set the values of weights to 1. Given the initial centroids $\mathbf{c}_j$, for $j=1,\ldots,k$, we compute the corresponding sets S_j as defined in (2), where $w_{ji}=1\ \forall j$ and $\forall i$. We then compute the average distance along each dimension from the points in S_j to $\mathbf{c}_j$. Let X_{ji} denote this average distance along dimension i. The smaller X_{ji} is, the larger is the correlation of points along dimension i. We use the value X_{ji} in an exponential weighting scheme to credit weights to features (and to clusters):

$$w_{ji}=exp(-h\times X_{ji})/(\sum_{l=1}^N(exp(-h\times 2\times X_{jl})))^{1/2} \tag{4}$$

where h is a parameter that can be chosen to maximize (minimize) the influence of X_{ji} on w_{ji}. When $h=0$ we have $w_{ji}=1/N$, thereby ignoring any difference between the X_{ji}. On the other hand, when h is large a change in X_{ji} will be exponentially reflected in w_{ji}. We empirically determine the value of h through cross-validation in our experiments with simulated data. We set the value of h to 9 in the experiments with real data. The exponential weighting is more sensitive to changes in local feature relevance [3] and gives rise to better performance improvement. In fact, it is more stable because it prevents distances from extending infinitely in any direction, i.e., zero weight. This, however, can occur when either linear or quadratic weighting is used.

The weights w_{ji} enable to elongate distances along less important dimensions, i.e. dimensions along which points are loosely correlated, and, at the same time, to constrict distances along the most influential ones, i.e. features along

which points are strongly correlated. Note that the technique is centroid-based because weightings depend on the centroid.
The computed weights are used to update the sets S_j, and therefore the centroids' coordinates. The procedure is iterated until convergence is reached, i.e. no change in centers' coordinates is observed.
The resulting algorithm, that we call LAC (Locally Adaptive Clustering), is summarized in the following.

Input: Set D of points $\mathbf{x} \in R^N$, and the number of clusters k.

1. Start with k initial centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$;
2. Set $w_{ji} = 1$, for each centroid $\mathbf{c}_j$, $j = 1, \ldots, k$ and each feature $i = 1, \ldots, N$;
3. For each centroid $\mathbf{c}_j$, $j = 1, \ldots, k$, and for each data point $\mathbf{x}$:
   - Set $S_j = \{\mathbf{x} | j = arg\min_l WDist(\mathbf{c}_l, \mathbf{x})\}$, where $WDist(\mathbf{c}_l, \mathbf{x}) = (\sum_{i=1}^N w_{li}(c_{li} - x_i)^2)^{1/2}$;
4. **Compute new weights**. For each centroid $\mathbf{c}_j$, $j = 1, \ldots, k$, and for each feature i:
   - Set $X_{ji} = \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2 / |S_j|$, where $|S_j|$ is the cardinality of set S_j (X_{ji} represents the average distance of points in S_j from $\mathbf{c}_j$ along feature i);
   - Set $w_{ji} = exp(-h \times X_{ji}) / (\sum_{l=1}^N exp(-h \times 2 \times X_{jl}))^{1/2}$;
5. For each centroid $\mathbf{c}_j$, $j = 1, \ldots, k$, and for each data point $\mathbf{x}$:
   - Recompute $S_j = \{\mathbf{x} | j = arg\min_l WDist(\mathbf{c}_l, \mathbf{x})\}$;
6. **Compute new centroids**. Set $\mathbf{c}_j = \frac{\sum_{\mathbf{x}} \mathbf{x} 1_{S_j}(\mathbf{x})}{\sum_{\mathbf{x}} 1_{S_j}(\mathbf{x})}$, for each $j = 1, \ldots, k$, where $1_S(.)$ is the indicator function of set S;
7. Iterate 3,4,5,6 until convergence (i.e., no change in centroids' coordinates)

The sequential structure of the LAC algorithm is analogous to the mathematics of the *EM* algorithm [7]. The hidden variables are the assignments of the points to the centroids. Step 3 constitutes the *E* step of the *EM* algorithm: it finds the values of the hidden variables S_j given the previous values of the parameters w_{ji} and c_{ji}. The following step (*M* step) consists in finding new matrices of weights and centroids that minimize the error function with respect to the current estimation of hidden variables.
We can also prove the following [8]:
Theorem. The LAC algorithm converges to a local minimum of the error function (3).

6. IMPLEMENTING LAC IN SQL

In the implementation of LAC in SQL we used the tables shown in Table 1. Next we describe in detail each step of our algorithm presented in Section 5, expressed in SQL.
In Figure 2 we give the query that populates table $S(id, cid)$, where *id* is the point id, and *cid* is the cluster id. This query implements steps 3 and 5 of the LAC algorithm. The query joins the dataset table $DATA$, with the centroids' table C and weights' table W, in order to calculate the weighted distances of each point to each cluster centroid. The result of the join goes to the temporary table $T1$. Then, in the second table $T2$, for each point, we calculate the minimum distance of the point at hand from all clusters. The final result (i.e. table S) is given through a join of $T1$ and $T2$. Table $T1$ has size of kD, while $T2$ has a size of D (and so does table S), where k is number of clusters and D is the number of points. Note that we can reduce the space used by the temporary tables by opening a cursor on table $DATA$, and for each tuple, calculate the k distances, and store the point, along with the cluster id corresponding to the minimum distance, directly into table S. Thus, the space requirement is the actual size of table S, i.e. D.

```
INSERT INTO S
WITH T1 (id, cid, wd) AS (
SELECT DATA.id,
C.cid,
SQRT(W.x1*(C.x1-DATA.x1)**2+..+W.xn*(C.xn-DATA.xn)**2)
FROM  DATA, C, W
WHERE C.cid = W.cid
),
T2 AS (
SELECT T1.id,  MIN(wd) AS mwd
FROM T1
GROUP BY T1.id
)
SELECT T2.id,  T1.cid
FROM T1, T2
WHERE T2.mwd = T1.wd AND T2.id = T1.id ;
```

Figure 2: SQL: Assign points (DATA) to clusters (S), calculating the weighted distances on the fly

```
INSERT INTO CARD_S
SELECT CID, COUNT(*)
FROM S
GROUP BY CID ;
```

Figure 3: SQL: Calculate the cardinalities of sets S_j.

Figure 3 shows the query that computes the cardinalities of sets $S_j, j = 1 \ldots k$. The sets are stored in table $S(id, cid)$. The computation is straightforward: it is a count of rows in S, grouped by the cluster id. The result is stored into table $CARD\_S$, and the values are used in steps 4 and 6 of LAC.

In Figure 4 we give the query that computes the average distance of points in S_j from c_j. This query implements the first part of step 4, where the values $X_{ji} = \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2 / |S_j|$ are computed. The result set of this query is kept into table $X(cid, d_1, \ldots, d_n, sum\_exp)$. Column sum_exp is calculated as a function of the n preceding columns, and it is used in second part of step 4, where the weights are updated. We chose to actually store this sum, in order to avoid having long expressions in the computation of the weights. The query joins tables C, $DATA$ and S on the cluster id and the point id, and presents the result by grouping the resulting rows on the cluster id.
The second part of step 4 of LAC is presented in Figure 5. The query uses the results stored in table X. Note that

Table	PK	Columns	Cardinality	Contents
$DATA$	id	$x1,\ldots,xn$	D	data points
W	cid	$x1,\ldots,xn$	k	weights
C	cid	$x1,\ldots,xn$	k	centroids
S	id, cid		D	sets S_j
X	cid	$d1,\ldots,dn, sum\_exp$	k	avg distances
$CARD\_S$	cid	$card$	k	card. of S_j

Table 1: Tables' descriptions

```
INSERT INTO X
WITH T1(cid, s1,..,sn) AS
(
SELECT S.cid ,
SUM((C.x1-DATA.x1)**2),...,SUM((C.xn-DATA.xn)**2)
FROM C, DATA, S
WHERE C.cid = S.cid
AND S.id = DATA.id
GROUP by S.cid
)
SELECT T1.cid ,
s1/CARD_S.card,
...
sn/CARD_s.card, 0
FROM T1, CARD_S
WHERE T1.cid = CARD_S.cid ;

UPDATE X
SET sum_exp = EXP(-h * X.d1) + ... + EXP( -h * X.dn) ;
```

Figure 4: SQL: Compute the distances X_{ji}

```
INSERT INTO W
SELECT X.cid,
EXP(-h * X.d1) / SQRT(X.sum_exp),
...
EXP( -h * X.dn) / SQRT(X.sum_exp)
FROM X;
```

Figure 5: SQL: Compute the weights (W)

parameter h is actually substituted by a scalar value, during the dynamic invocation of the queries by our control program.
In Figure 6 the query that computes the centroids is presented. This is step 6 of the LAC algorithm. The query first joins tables $DATA$, S and C, in order to compute the sums of the attributes of points that belong to the same cluster. This intermediate results goes to temporary table $SUMS$, and has size k. Then, tables $SUMS$ and $CARD\_S$ are joined, in order to compute the final result, i.e. the coordinates of the new centroids. Note that we cannot avoid the second join (which is performed on small tables, anyway), because all the free columns that appear in a SELECT clause and are not involved in an aggregation function (SUM in our case) have to appear in the GROUP BY clause. In our case, we perform a GROUP BY on column $S.cid$ only, so we cannot use column $CARD\_S.card$ in the SELECT clause that creates table $SUMS$. Hence the need for the second join between tables SUM and $CARD\_S$.

```
INSERT INTO C
WITH SUMS(cid, s1,..,sn) AS
(
SELECT C.cid, SUM(DATA.x1),...,SUM(data.xn)
FROM DATA, C, S
WHERE C.cid = S.cid
AND S.id = DATA.id
GROUP BY C.cid
)
SELECT  SUMS.cid ,
s1/CARD_S.card AS new_x1,
....
sn /CARD_S.card AS new_xn
FROM SUMS,  CARD_S
WHERE CARD_S.cid = SUMS.cid ;
```

Figure 6: SQL: Compute new centroids (C)

7. EXPERIMENTAL EVALUATION

In this section we present an experimental evaluation of the SQL implementation of LAC. The experiments were run on a desktop PC having a P4 processor running at 1.6GH and 1G of RAM. The hosting operating system was Linux Mandrake 8.1 (kernel ver. 2.4.17) and we we used DB2 v8.1.
We created synthetic datasets having dimensionality $N = 5$ and we embedded $k = 5$ clusters. The points' coordinates spanned within $[0, 100]$ on each dimension. The cardinalities of the datasets D were $100k$, $500K$ and $1M$ tuples. Figure 7 shows the execution times per iteration. The algorithm converges, on average, after 5 to 7 iterations. We can see that the execution times scale linearly with the size of the database. The algorithm's error (fraction of misclassified objects over total number of objects) for the three datasets (having $100k$, $500k$ and $1M$ tuples) was 6.16%, 6.09% and 6.5%, respectively.
In order to evaluate the performance of LAC with respect to the dimensionality of the data, we generated a second set of datasets having dimensionality N equal to 10, 20 and 30. We fixed the cardinality D to $100k$ tuples and we embedded $k = 5$ clusters. The execution times per iteration, for these datasets, is show in Figure 8. The algorithm's error for the three datasets was 8.4%, 4.16% and 4.06%, respectively.
Furthermore, in order to check the performance while varying the number of embedded clusters, datasets with k equal to 5 , 10 and 20 were used. The cardinality D was fixed to $100k$ tuples, while the dimensionality N of the data was set to 5. Figure 9 depicts the execution time per iteration for these sets of data. The algorithm's error for the three datasets was 6.16%, 4.26% and 3.77%, respectively.
The basic characteristic of microarray DNA expression data is the large number of conditions, under which the probes are done. In the form of a relation table, the genes correspond to rows and the conditions to columns (attributes). In order to examine the scalability of the SQL implementation

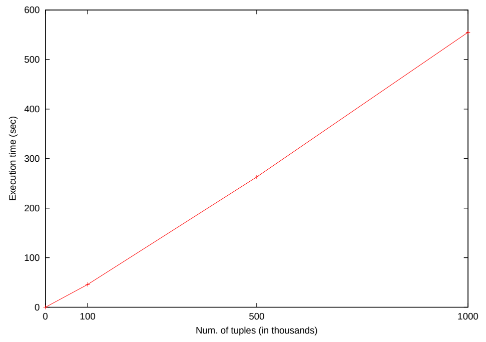


Figure 7: Synthetic datasets: time per iteration with varying no. of tuples, N=5, k=5

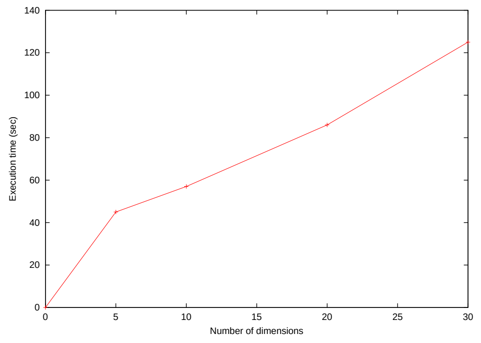


Figure 8: Synthetic datasets: time per iteration with varying no. of dimensions, D=100k, k=5

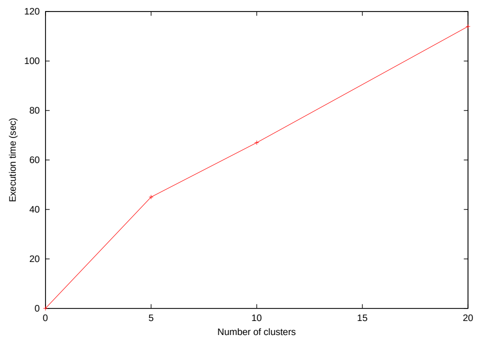


Figure 9: Synthetic datasets: time per iteration with varying no. of clusters, D=100k, N=5

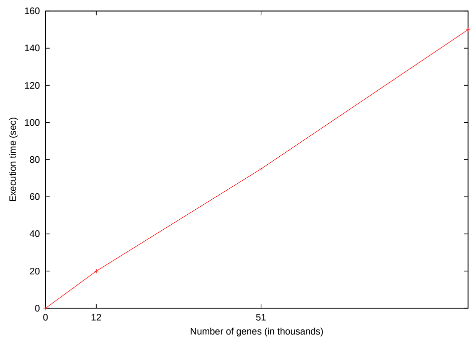


Figure 10: Biological datasets: time per iteration with varying no. of genes, N=22

of LAC, we did the following: we used a DNA microarray of gene expression profiles in hereditary breast cancer[1], which has 3226 genes and 22 conditions, as a basis, and we replicated it 4, 16 and 31 times, in order to get larger sets, having approximately 12K, 51K and 100K tuples, respectively. We ran LAC with k=4. The results show that within two biclusters five conditions (having id's 7,8,9,19,22 in the first one, and 7,8,9,13,22 in the second one) receive considerably larger weight than the others. The remaining biclusters contain fewer genes, and all conditions receive equal weights. This indicates that no correlation was found among those conditions. We observe that different combinations of conditions are selected for different biclusters, as also expected from a biological perspective. Figure 10 presents the execution times per iteration for this set of experiments. We observe that the algorithm scales up linearly with respect to database size, i.e. number of genes.

8. CONCLUSIONS

We presented an algorithm that discovers clusters in subspaces by different combinations of dimensions via local weightings of features. This method avoids the risk of loss of information encountered in global dimensionality reduction techniques. We implemented our algorithm using SQL in order to facilitate the discovery of clusters on datasets stored in a DBMS. The experimental evaluation shows the scalability of our algorithm.

9. REFERENCES

[1] C. Aggarwal, C. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast Algorithms for Projected Clustering. In *Proc. ACM SIGMOD*, 1999.

[2] P. Arabie and L. Hubert. An overview of combinatorial data analysis. Clustering and Classification. *World Scientific Pub.*, pages 5–63, 1996.

[3] L. bottou and V. Vapnik. Local Learning algorithms. *Neural Computation*, 4(6):888–900, 1992.

[4] K. Chakrabarti and S. Mehrotra. Local dimensionality reduction: A new approach to indexing high dimensional spaces. *In Proc. VLDB*, 2000.

[1]http://research.nhgri.nih.gov/microarray/NEJM_Supplement

[5] P. Cheeseman and J. Stutz. *Bayessian Classification (autoclass): Theory and Results*, chapter 6. AAAI/MIT Press, 1996.

[6] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proc. ISMB*, pages 93–103, 2000.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1997.

[8] C. Domeniconi. *Locally Adaptive Techniques for Patern Classification*. PhD thesis, UC Riverside, Aug 2002.

[9] J. G. Dy and C. E. Brodley. Feature Subset Selection and Order Identification for Unsupervised Learning. In *Proc. ICML*, 2000.

[10] M. Ester, H. P. Kriegel, and X. Xu. A database interface for clustering in large spatial databases. In *Proc. KDD*, 1995.

[11] Z. Ghahramani and G. E. Hinton. The EM Algorithm for Mixtures of Factor Analyzers. Technical Report CRG-TR-96-1, Dept. of Computer Science, Univ. of Toronto, 1996.

[12] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. of ACM SIGMOD*, 2001.

[13] R. Michalski and R. Stepp. *Machine Learning: An Artificial Intelligence Approach*, chapter 'Learning from observation: Conceptual Clustering'. IOGA Publishing Co., 1983.

[14] R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proc. of VLDB*, 1994.

[15] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A Monte Carlo algorithm for fast projective clustering. In *Proc. ACM SIGMOD*, 2002.

[16] D. G. R. Agrawal, J. Gehrke and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data For Data Mining Applications. In *Proceedings of ACM SIGMOD*, pages 94–105, June 1998.

[17] A. Thomasian, V. Castello, and C. S. Li. Clustering and singular value decomposition for approximate indexing in high dimensional spaces. In *Proc. CIKM*, 1998.

[18] M. E. Tipping and C. M. Bishop. Mixtures of Principal Component Analyzers. *Neural Computation*, 1(2), 1999.

[19] H. Wang, W. Wang, J. Yang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proc. ACM SIGMOD*, 2002.

[20] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proc. ACM SIGMOD*, 1996.

Graph-Based Ranking Algorithms for E-mail Expertise Analysis

Byron Dom[*]
Yahoo! Inc.
701 First Ave.
Sunnyvale, CA, USA
bdom@yahoo-inc.com

Iris Eiron
IBM Almaden Research Center
650 Harry Rd.
San Jose, CA, USA
iriss@us.ibm.com

Alex Cozzi
IBM Almaden Research Center
650 Harry Rd.
San Jose, CA, USA
cozzi@almaden.ibm.com

Yi Zhang[†]
Language Technology Institute
School of Computer Science
Carnegie Mellon University
yiz@cs.cmu.edu

ABSTRACT

In this paper we study graph–based ranking measures for the purpose of using them to rank email correspondents according to their degree of expertise on subjects of interest. While this complete expertise analysis consists of several steps, in this paper we focus on the analysis of digraphs whose nodes correspond to correspondents (people), whose edges correspond to the existence of email correspondence between the people corresponding to the nodes they connect and whose edge directions point from the member of the pair whose relative expertise has been estimated to be higher. We perform our analysis on both synthetic and real data and we introduce a new error measure for comparing ranked lists.

General Terms

Algorithms, Experimentation, Measurement

Keywords

Expert finding, Social network analysis, Digraph node ranking, Ordered list distance

1. INTRODUCTION

A problem faced on occasion by almost everyone is that of locating some desired information or source of knowledge. The latter is desirable when one isn't even sure what to ask for and can usually be satisfied by finding an expert in the topic of interest. This problem of finding an expert is what we address in this paper.

In large enterprises such as companies and government agencies a source of information that can be utilized in this search for experts is the aggregate email of the organization. Email is used today as a major means of communication in business. As such it contains precious information about the activities, interests and priorities of an individual or the organization. The email collection also includes both notes requesting information and notes providing it. It is reasonable to hope, therefore, that appropriate analysis of an organization's aggregate email can identify individuals with a high level of expertise in the topics of interest to the organization as a whole or some sufficiently large fraction of it.

In [3] an approach to performing this expertise analysis of email collections and a system embodying that approach are described. In this paper we concentrate on the third part of the following three–step process. This process is applied to the subset (of the complete collection) of the emails discussing a certain topic.

1. Quantitatively estimate the relative expertise between two people based on all the on–topic emails between them.
2. These pairwise relative–expertise assessments can be seen to implicitly form a weighted directed graph (*digraph*) in which the nodes correspond to people (senders and receivers of emails in the collection). The existence of an edge between nodes i and j indicates that there was at least one email between the associated people. The direction of the edge is from the person of greater expertise to the person of lesser expertise and the edge weight is the quantitive measure of relative expertise.
3. The described digraph is analyzed using a graph–based ranking algorithm whose output is a ranked list of all the nodes (people) according to their absolute expertise level.

The goal of the work described here is to compare various ranking algorithms. Our method for comparison is based on

[*]This research was done while in IBM.
[†]This research was done while on internship in IBM.

applying them to graphs where the ground truth ranking is known and then assessing how well the algorithms output agrees with the true ranking. Both simulations and real–data experiments are presented. We study a simulation of a fully connected "perfect" digraph in which every node pair is connected by a link with the correct direction and weight is formed. This graph is then systematically degraded by removing and/or reversing links. Additionally we analyze graphs based on real data, in which edges account for email exchange. These graphs contain only a small fraction of the possible edges. Furthermore the use of text classification to determine the direction of the edges may further degrade the quality of the graph by assigning incorrect directions and/or weights to the edges.

2. RELATED WORK

Motivated by the prevalence of the WWW, the task of expert finding has lately received growing attention from the research community [6; 4; 8]. For a review of work in this area, both research studies and commercial systems, we refer the reader to [15]. In contrast to our approach, the majority of the systems studies so far focus on connecting people to experts who posses some knowledge on a given topic. Our system in comparison is concerned with ranking people according to their expertise level rather than separating experts from non experts.

A system that attempts to provide ranking of experts by their expertise level is described in [9]. This system is an expert–finder system that exploits technical papers, presentations, resumes, home pages etc. on MITRE's corporate intranet to enable the location of relevant experts. The ranking uses a simple scheme based on the number of mentions of a term/phrase.

Another factor differentiating our system from many other expertise–location systems is the use of email for identifying experts. One of the few systems that are based on email is [13]. However this system does not consider the content of email messages but only flow pattern between the correspondence. Another system that uses email is presented in [14]. This system uses a pre-existing hierarchy of subject areas, characterized by word frequencies, to identify experts in specific areas by analyzing the word frequencies of the email messages written by each individual. However, like many of the systems mentioned above, this system does not deal with the question of the ranking of the identified experts.

3. RANKING MEASURES

As we stated in the introduction, our hope is to use both the texts of email messages and their distribution patterns to determine who the experts are. Because we represent the organization whose email we study as a graph whose nodes correspond to people and whose edges (links) correspond to the existence of email correspondence between the people whose corresponding nodes they connect, we think of and refer to this analysis as "link analysis".

The outcome of our analysis of the email between two people is the determination of which of the two knows more about the topic and a relative–expertise score, which characterizes the magnitude of that assessed difference. We can make an analogy between this comparison and a competition (match, game, etc.) between two sports teams. In this analogy the relative–expertise score corresponds to the difference in score between the two teams. We can further extend this analogy to a correspondence between finding experts and finding a champion by ranking teams, especially in cases where there haven't been competitions between every pair of teams. This problem has been studied extensively (see, for example [1; 2]). We will make use of this analogy in the following descriptions of the ranking approaches we evaluate.

3.1 Affinity (a.k.a Score)

Following our team–ranking analogy, perhaps the simplest way of ranking teams is by the number of wins or (closely related) the total point differential between the team and all of its opponents. We refer to this measure as "affinity" and it was referred to as "score" in [5]. This measure would be fine if every team played every other team or if we had relative–expertise assessments between every pair of people, but when that is not the case, it tends not to work well because it rates wins over all teams equally regardless of the records of those teams and who their opponents were.

3.2 Successor

This is related to the measures described in [5]. Think of directed edges pointing from winners to losers or from greater to lesser expertise. If all these edge directions are correct, all the people "downstream" (reachable by a directed path) from a given node are of lesser expertise. This one measure of expertise is simply the count of such people (nodes).

3.3 PageRank

One of the more popular team–ranking techniques is described in [5]. In this approach the obtained ranking is the principal eigenvector (corresponding to the largest eigenvalue) of the adjacency matrix of the digraph in which edges correspond to matches between teams and the direction of the edge is determined by who won. This approach turns out to be virtually identical to the well known PageRank[12] algorithm for ranking web pages, the only difference being that in PageRank low–weight edges are added between all nodes in both directions. This weight is an adjustable parameter. We refer to this step of adding low–weight links to all nodes as "smoothing", which we also perform.

3.4 Positional Power Function

One of the ranking measures we chose to evaluate is taken from [5]. It is closely related to what they call the "Positional Power Function" (PPF). For PPF the ranks $\{r_i | i = 1, 2, \ldots, n\}$ satisfy the following system of equations.

$$r_i = \sum_{j \in S_i} \frac{1}{n}(r_j + 1), \qquad (1)$$

where S_i denotes the set of *successor* nodes of node i.

A generalization of (1), which we evaluated as part of our experimental characterization is:

$$r_i = \sum_{j \in S_i} [\alpha + (1 - \alpha) r_j], \qquad (2)$$

We solved this system by the usual power–iteration scheme. The corresponding results in our experimental section are labelled "power".

3.5 HITS (authority)

The HITS[7] algorithm was devised with the thought of ranking web pages according to their degree of "authority". It is designed with the structure of the web in mind. In the associated digraph the directed edges correspond to hyperlinks. The HITS algorithm produces two scores/rankings – the "hub" score and the "authority" score. We present only the latter because it gave consistently better results than the hub score as expected. In terms of the digraph adjacency matrix A the authority score is given by the principal eigenvector of the matrix $A^T A$, where A^T represents the *transpose* of A.

4. ACCURACY EVALUATION

In this section we describe the measures we use to characterize the accuracy of ranking by characterizing the degree of agreement/disagreement between the computed ranking r and the correct (ground–truth) ranking ρ. In the analysis of synthetic data (described in Section 5) we used a novel error–distance measure, designed to exhibit qualitative behavior desirable for an expertise ranking accuracy measure. We we refer to this measure as "rnkerr" and describe it in the following section. The real–data analysis (described in Section 6), presents results using both the "rnkerr" measure and recall graphs.

4.1 Rank–Error Distance

This is a measure of the following form:

$$\epsilon(\rho, r) \;=\; \sum_{i=1}^{N} \eta(m_i)\,\zeta(\delta_i) \qquad (3)$$

where

$$m_i \;\triangleq\; \min(\rho_i, r_i) \qquad \text{and} \qquad \delta_i \;\triangleq\; |\rho_i - r_i|.$$

The general form of (3) embodies two notions about the desirable qualitative behavior of such a measure. One of these ideas is that the importance of an error (difference in the rank of the i^{th} object) increases with the highest rank the associated object received on either of the lists ($\min(\rho_i, r_i)$). If neither rank is high, the associated error (difference) should contribute very little to the overall score. This notion is captured in the function $\eta(\cdot)$, which decays monotonically with its argument, asymptotically approaching zero.

The second idea is that the magnitude of the error associated with a difference δ in rank should saturate at (asymptotically approach) some value as δ increases. The notion here is that beyond some value, all differences should carry approximately the same weight. This behavior is embodied in the function $\zeta(\delta)$.

In experiments we used the following specific functional forms for η and ζ each characterized by single adjustable parameters – λ and β respectively.

$$\eta(m; \lambda) \;=\; \frac{\exp(-\lambda m)}{\sum_{m=1}^{N} \exp(-\lambda m)}$$

and

$$\zeta(\delta; \beta) \;=\; \frac{1}{\beta}[1 - \exp(-\beta\delta)]$$

Where the parameter value $\lambda = 0.25$ and $\beta = 0.1$ were used in our experiments.

4.2 Recall

In our *recall*–based analysis we characterize the comparison between two rankings by a *recall* curve (see Figure 2 for example). In the figures depicting this the "query size" (let q represent this) refers to the top q ranked nodes (people) in the computed ranking r and the "recall" is the fraction of the top q nodes of the ground–truth ranking ρ that are contained in the top q members of r.

5. RESULTS FOR SYNTHETIC DATA

As part of our study of the behavior of the various ranking measures we ran them on synthetically generated expertise graphs. The general approach was to initially generate "perfect" graphs and then randomly degrade those graphs by removing and reversing edges. By "perfect" graphs we mean graphs for which there is an edge (link) between every pair of nodes and the direction of these edges correctly reflect the relative expertise of the nodes they connect. In what follows we present results for three such collections of experiments.

1. The first of these corresponds to only removing edges from a perfect 21-node graph.
2. The second corresponds to only reversing edges from a perfect 21-node graph.
3. The third corresponds to both removing and reversing edges from a 15-node perfect graph. That is, a number of edges is first removed and then a number of those remaining are reversed.

In the work presented here all edges have weight 1.

Figures 1(a) and 1(b)) are plots of the error distance ("rnkerr" = $\epsilon(\rho, r)$) described in Section 4 versus the number of nodes either removed or reversed. The individual points are average values from several trials as described below. There are three tables in this section, each table representing an experiments collection. Each table contains three columns for each measure. The following is a description of these columns.

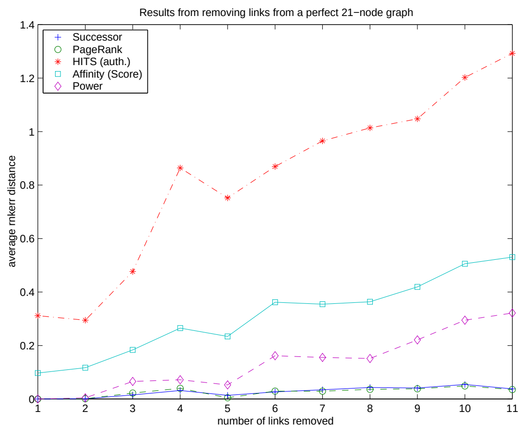
"avg. dist.": This is the average $\epsilon(\rho, r)$ between the ranking produced by the measure and the "ground truth" (i.e. the correct ranking). This average is over all trials in the particular experiment. Smaller values are better; a perfect score is zero.

"no. best": This is the number of trials for which the associated measure produced the best ranking as measured by $\epsilon(\rho, r)$. Larger numbers are better; a perfect score is equal to the total number of trials. The worst possible score is zero.

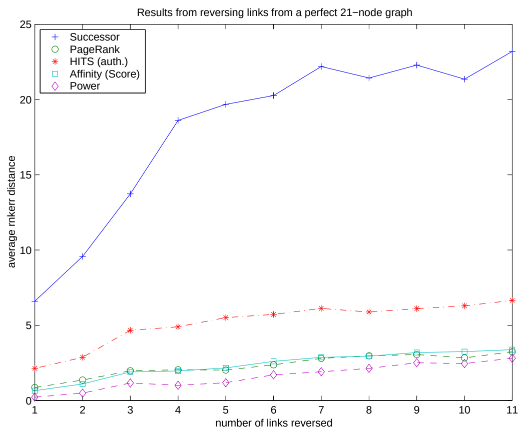
"no. worst": This is the number of trials for which the associated measure produced the worst ranking as measured by $\epsilon(\rho, r)$. Smaller numbers are better; a perfect score is zero. The worst possible score is equal to the total number of trials.

5.1 Removal of Edges from Perfect Graphs

The results presented here are for a set of experiments on 21-node graphs. It is assumed that the associated 21 people have expertise "scores" of $\{1, 2, 3, \ldots, 20, 21\}$. A directed link with weight 1 is placed between every pair of nodes going from higher to lower expertise score. This is the initial "perfect" graph, which has $|E| = \binom{21}{2} = 210$ edges. The number k_{rem} of edges removed from these graphs is varied from 1 through $\lceil n/2 \rceil$, which is 11 in this case. The results of these experiments are plotted in Figure 1(a) and tabulated in Table 1.



(a) Removing links.



(b) Reversing links.

Figure 1: Results from randomly removing/reversing links in a perfect 21-node.

measure	avg. dist.	no. best	no. worst
Successor	0.0342	321	0
PageRank	0.0317	332	0
HITS	0.9711	0	196
Affinity	0.3753	0	1
Power	0.1825	142	0

Table 1: Results of randomly removing links from a perfect 21-node graph.

5.2 Reversal of Edges in Perfect Graphs

In this set of experiments we reverse rather than remove edges in perfect graphs. The number of edges reversed and the scheme for determining that number and selecting the edges is exactly the same as that used in the removal experiments described in Section 5.1. The results of these experiments are plotted in Figure 1(b) and tabulated in Table 2.

measure	avg. dist.	no. best	no. worst
Successor	3.5258	1	369
PageRank	0.4594	62	0
HITS	0.9983	7	0
Affinity	0.4804	27	0
Power	0.3468	256	0

Table 2: Results of randomly reversing links in a perfect 21-node graph.

5.3 Combined Removal and Reversal from Perfect Graphs

The results presented here are for a set of experiments on 15-node graphs. It is assumed that the associated 15 people have expertise "scores" of $\{1, 2, 3, \ldots, 14, 15\}$. A directed link with weight 1 is placed between every pair of nodes going from higher to lower expertise score. This is the initial "perfect" graph, which has $|E| = \binom{15}{2} = 105$ edges. This graph is then degraded in various ways described as follows.

- number of edges removed k_{rem}: Varied from 1 through $\lceil n/2 \rceil$, which is 8 in this case.
- number of random sets of k_{rem} (remaining) edges removed for each value of k_{rem}:
$$\nu(k_{\mathrm{rem}}) = \left\lfloor \log_2 \binom{|E|}{k_{\mathrm{rem}}} \right\rfloor$$
Let $G_{k_{\mathrm{rem}}}$ be a random variable whose values are graphs formed by deleting random sets of k_{rem} edges from G.
- number of edges reversed k_{rev}: This is varied from 1 through k_{rem} for each $G_{k_{\mathrm{rem}}}$.
- number of random sets of k_{rev} edges reversed for each value of the pair $(G_{k_{\mathrm{rem}}}, k_{\mathrm{rev}})$:
$$\nu(k_{\mathrm{rem}}, k_{\mathrm{rev}}) = \left\lfloor \log_2 \binom{|E| - k_{\mathrm{rem}}}{k_{\mathrm{rev}}} \right\rfloor$$
- total no. graphs analyzed
$$\sum_{k_{\mathrm{rem}}=1}^{\lceil n/2 \rceil} \nu(k_{\mathrm{rem}}) \sum_{k_{\mathrm{rev}}=1}^{k_{\mathrm{rem}}} \nu(k_{\mathrm{rem}}, k_{\mathrm{rev}}) = 19307.$$

The results of this process are tabulated in Table 3.

measure	avg. dist.	no. best	no. worst
Successor	2.2315	457	15109
PageRank	0.6303	6372	14
HITS	1.1722	576	1890
Affinity	0.7225	1324	7
Power	0.5927	9012	16

Table 3: Results of randomly removing and reversing links from a perfect 15-node graph.

6. RESULTS FOR REAL DATA

For the second part of our experiments we collected a data set of internal email communication contributed by individuals in our organization. Fifteen people from a single workgroup contributed email messages from their personal email boxes, containing a total of 13417 messages. From this set we selected, using keyword search, messages related to ten topics. Five of the ten topics were generic, such as LaTeX, Java, and Perl, while the other five were more specific to the type of work done in the workgroup. The full list of people involved in correspondence on each topic was extracted. Human evaluators were asked to rate each person for his/her level of expertise per topic on a scale from 1 to 10. These ratings were then averaged across the evaluators. We used the resulting ranked list for each of the topic as the ground truth for our experiments.

Recall that in the graph representation of an email corpus nodes are associated with the people involved in the correspondence and a directed edge from node i to node j indicates that i is less of an expert on the subject than j is. With an email collection at hand this relative expertise information may be based on the analysis of the email messages exchanged between the parties. In the case where no correspondence exists between a pair of people, the relative expertise level cannot be directly determined and the corresponding edge will not exist in the graph. As a result, graphs based on real email correspondence are much sparser than the ideal fully connected graphs. Table 4 reveals that the number of edges in graphs representing the ten topics is on average only 20% of the number of edges in the equivalent fully connected graphs.

topic	#nodes	#edges	$\frac{\#\text{edges}}{\binom{\#\text{nodes}}{2}}$
java	37	84	12
latex	8	6	21
matlab	12	11	16
perl	12	11	16
xml	26	44	13
cognitive	19	21	12
eye	12	13	19
pong	15	13	12
wbi	28	98	25
wf	13	12	15
avg.	18	31	20

Table 4: Characteristics of real data base graphs.

While in the synthetic experiments (Section 5) we were randomly reversing edges, with an email corpus at hand we assume the direction of an edge is determined by a text classifier which analyzes the content of the messages exchanged between a pair of people. Like any analysis of this kind we cannot expect 100% accuracy from this process. Therefore in the resulting graphs some of the edges may be reversed. We experimented with four different analysis methods. For every pair of people i,j between which correspondence exists the various analysis methods will act as follows:

Ideal Classifier If the expertise level of i is higher than that of j and edge (i, j) is created with weight 1.

Dummy Classifier Creates two parallel edges in opposite directions between i and j, each with weight 1/2.

Random Classifier Flips an unbiased coin. If "tail" — creates an edge i, j with weight 1. Otherwise, creates an edge j, i with weight 1

Maximum Entropy We trained a Maximum Entropy (ME) [11] classifier from the Rainbow package [10]. We ran the classifier on the set of messages sent from i to j. We then create an edge (i, j) with weight p and an edge (j, i) where p is the probability the classifier gave to the event that i has higher expertise than j. If communication exists in both directions we run the classifier separately for each direction. We then assign a weight of $(p + 1 - q)/2$ to the edge (i, j) and $(q + 1 - p)/2$ to the edge (j, i) where p is as before and q is the output of the classifier when run on message on the opposite direction. In our experiments we used 9 out of the 10 queries to train the classifier and tested on the remaining topic.

	PageRank	Hits	Power	Affinity
Ideal	0.32	0.44	0.42	0.42
Dummy	0.53	0.53	0.53	0.56
Random	0.58	0.58	0.54	0.55
ME	0.56	0.55	0.55	0.57

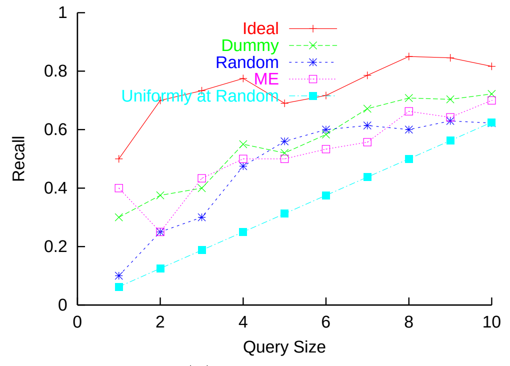
Table 5: Results on on real data based graphs.

A summary of the results of running the four ranking algorithms (excluding Sucessor) perviously mentioned on our email data is brought in table 5. The results are further detailed in Figures 2, 3 and 4.

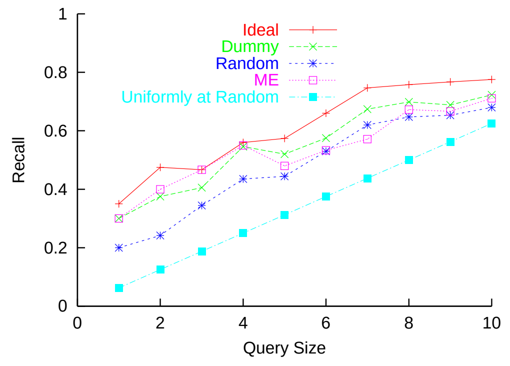
7. DISCUSSION OF RESULTS

In the synthetic results of Section 5 we examined three modes of degradation starting from perfect graphs. In the removal–only results the best (nearly perfect) results were obtained using PageRank and Successor, whereas HITS was significantly worse than the others. In the reversal–only results, on the other hand, the best results were obtained with Power, though the results obtained with PageRank and Affinity were nearly as good. The Successor results were much worse than the others and the HITS results were somewhat worse. Similar results were obtained in the removal–plus–reversal experiments. That is, the best results were obtained with Power, with PageRank and Affinity finishing a relatively close second and third respectively. The HITS algorithm was a more distant fourth and Successor was once again significantly worse than the others. As can be expected, measured accuracies obtained for the removal–only experiment were significantly better than those obtained for reversal–only and removal–plus–reversal results.

In the real data results of Section 6 we examined four modes of degradation starting from sparse graphs. The results obtained with "ideal" classifier support the synthetic removal–only results. PageRank performed significantly better whereas HITS was slightly worse than the other methods. Among the Dummy, Random, and ME classifiers very small differences in overall performance are observed. It should be noted, however, that Power performed slightly better than all the other methods. This is consistent with the results gathered for the removal–plus–reversal and removal–only

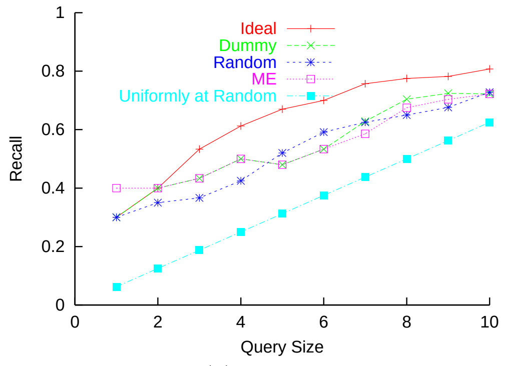


(a) PageRank.

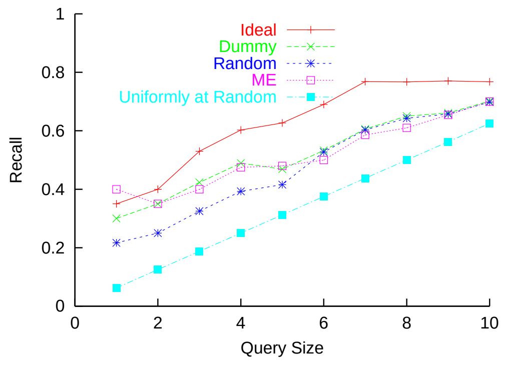


(b) HITS.

Figure 2: Impact of Classification.



(a) Power.



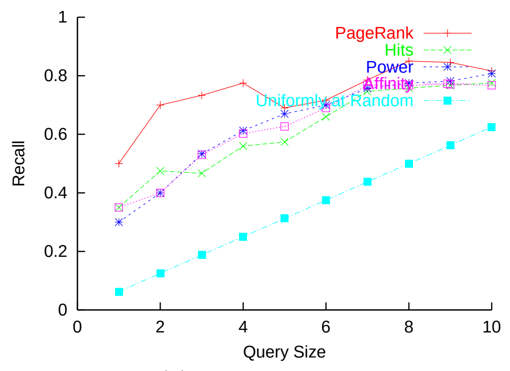
(b) Affinity.

Figure 3: Impact of Classification – contd.

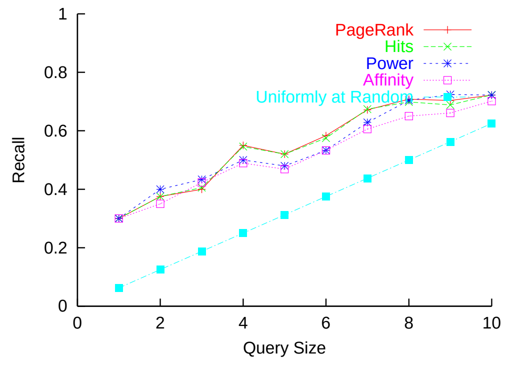
synthetic experiments. Note that the difference in performance between best performing algorithm and worst performing algorithm on synthetic data was much more significant than the difference between those algorithms obtained on real data. These experiments also confirm that using the ideal (i.e. correct) classification to generate the edge directions in the graph yielded significantly better results than those obtained when using the other classification methods.
As a calibration point, the graphs presenting real data results (Figures 2,3 and 4) include curves corresponding to picking a ranking uniformly at random. These are the straight–line curves appearing lowest in the four subfigures. Not surprisingly, all examined ranking algorithms perform significantly better than this random–ordering.
Unfortunately, we discovered that the ME classifier yielded ranking performance that is comparable if not worse than that of the Dummy and Random classifiers. This essentially means that this classifier failed to provide the ranking algorithm with any useful information from the content of the messages. This demonstrates the difficulties of the classification task. Two main factors contribute to the difficulty of this task: (1) Both messages authored by experts and those authored by lesser experts, discuss the same topic, and therefore use similar vocabulary. (2) Email messages vary in length and tend to be shorter than other documents – as short as a single word (e.g. "yes" or "no") – with much of the original context missing. Therefore, not every message contains sufficient information by itself to identify the relative expertise level of the parties involved.

7.1 Conclusion

The Successor algorithm, while dealing well with missing edges, is clearly not robust to errors in edge direction. For this reason it was ommited from the real data analysis. The HITS algorithm, while giving results of reasonable accuracy, gave noticeably worse results than the others (except Successor). The inferior results obtained with HITS may be attributable to the fact that it was conceived for the purpose of ranking web pages and is based on the notion of the existence of *hubs* and *authorities* which mutually reinforce each other during the ranking process. While one can hypothesize the existence of certain email correspondents that play a role analogous to that of hubs, it seems unlikely that many exist. A good hub would be someone who corresponds with a large number of experts on a certain topic. It is our feeling that most users of email will tend to have one (or at most a few) person they use as an expert consultant on a given topic.
The tournament–like model implicit in Power and PageRank seems more appropriate to the problem at hand. PageRank performs noticeably better than all other algorithms in the case edges are missing but the ones present in the graph point in the right direction. When noise is introduced in the edge directions Power becomes more advantageous. The final conclusion regarding which of the two is best depends on the accuracy of the available classifier. Affinity also gave reasonable results though somewhat inferior to Power and PageRank.
Examining the performance of the various algorithms operating on a graph constructed with the Dummy classifier



(a) Ideal Classifier.



(b) Dummy Classifier.

Figure 4: Best ranking algorithm.

(Figure 4(b)) reveals that all the examined algorithms perform significantly better than the "random ranking" curve. This result is encouraging because it implies that by only using the patten of the email exchange, independently of the link structure, some significant information was extracted. With a query size of only four we receive 50% recall. If the accuracy of the classifier is to be improved we can expect up to 70% recall with a query size of only two.

8. FUTURE WORK

In the future we may investigate other ranking algorithms and variations of those studied here. In addition we plan more extensive experimental investigations. In the synthetic–data investigations we will use graphs that are much sparser and therefore closer to those encountered in real data. We also plan to include weighted edges (i.e. not only 0/1) in our analysis. In the real–data experiments we will obtain more representative email collections.

Acknowledgments

We would like to thank Christopher Campbell for collecting subjective ranking data, and Nadav Eiron for his continuous support.

9. REFERENCES

[1] Bibliography on college football ranking systems. `http://www.cae.wisc.edu/~dwilson/rsfc/rate/biblio.html`.

[2] Overview of ranking literature from tbeck. `http://www.phys.utk.edu/sorensen/ranking/Data%20Import/overview_of_rank%ing_literature_f.htm`.

[3] A. Cozzi, B. Dom, I. Eiron, Y. Zhang, C. Campbell, and P. Maglio. Identifying experts by analyzing e-mail. submitted, 2003.

[4] L. N. Foner. Entertaining agents: A sociological case study. In *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 122–129, 1997.

[5] P. J.-J. Herings, G. van der Laan, and D. Talman. Measuring the power of nodes in digraphs. Technical Report 01-096/1, Tinbergen Institute, October 2001.

[6] H. Kautz, B. Selman, and M. Shah. Referral Web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3):63–65, 1997.

[7] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[8] B. Krulwich and C. Burkey. The ContactFinder agent: Answering bulletin board questions with referrals. In *AAAI-96*, 1996.

[9] D. Mattox, M. Maybury, and D. Morey. Enterprise expert and knowledge discovery. In *Proceedings of the 8th International Conference on Human-Computer Interaction (HCI International'99)*, pages 303–307, 1999.

[10] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[11] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification, 1999.

[12] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[13] M. F. Schwartz and D. C. M. Wood. Discovering shared interests using graph analysis. *Communications of the ACM*, 36(8):78–89, 1993.

[14] W. Sihn and F. Heeren. Xpertfinder - expert finding within specified subject areas through analysis of e-mail communication. In *EUROMEDIA 2001 : Sixth Annual Scientific Conference on Web Technology, New Media, Communications and Telematics Theory, Methods, Tools and Applications*, 2001.

[15] D. Yimam and A. Kobsa. Expert finding systems for organizations: Problem and domain analysis and the demoir approach. *"Journal of Organizational Computing and Electronic Commerce"*, 13(1):1–24, 2003.

Deriving Link-context from HTML Tag Tree

Gautam Pant
Department of Management Sciences
The University of Iowa
Iowa City, IA 52242
gautam-pant@uiowa.edu

ABSTRACT

HTML anchors are often surrounded by text that seems to describe the destination page appropriately. The text surrounding a link or the *link-context* is used for a variety of tasks associated with Web information retrieval. These tasks can benefit by identifying regularities in the manner in which "good" contexts appear around links. In this paper, we describe a framework for conducting such a study. The framework serves as an evaluation platform for comparing various link-context derivation methods. We apply the framework to a sample of Web pages obtained from more than 10,000 different categories of the ODP. Our focus is on understanding the potential merits of using a Web page's tag tree structure, for deriving link-contexts. We find that good link-context can be associated with tag tree hierarchy. Our results show that climbing up the tag tree when the link-context provided by greater depths is too short can provide better performance than some of the traditional techniques.

General Terms

Algorithms, Experimentation, Measurement

Keywords

Tag tree, DOM, Link-context

1. INTRODUCTION

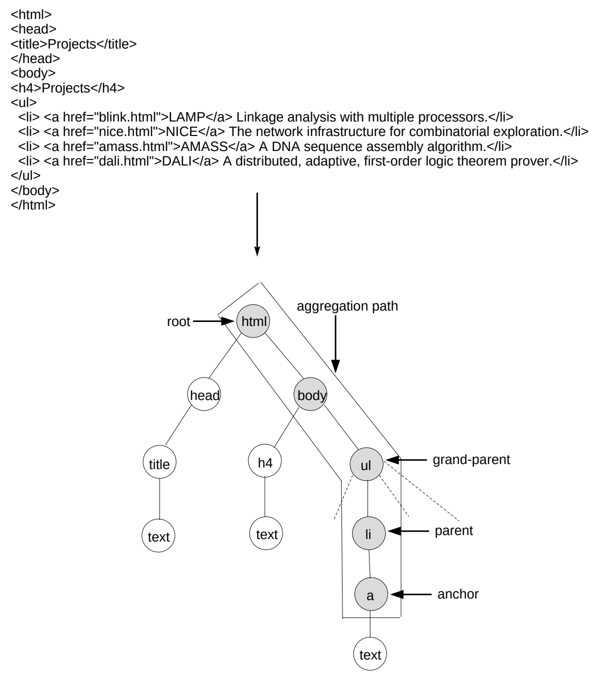
Link-context is used for many tasks related with Web information retrieval. In most cases, anchor text or text within an arbitrary size window around a link is used to derive the context of a link. While the importance of link-context has been noted in areas such as automatic classification [2] and search engines [3], *focused* or *topical* crawlers (e.g., [9; 6; 12; 1]) have a special reliance on it. Topical crawlers follow the hyperlinked structure of the Web while using the *scent of information* to direct themselves towards topically relevant pages. For deriving (sniffing) the appropriate scent they mine the content of pages that are already fetched to prioritize the fetching of unvisited pages. Unlike search engines that use contextual information to complement the content based retrieval, topical crawlers depend primarily on contextual information (since they score unvisited pages). Moreover, a search engine may be able to use contextual information about a URL (and the corresponding page) from a large number of pages that contain that URL. Such global information is hard to come by for a topical crawler that only fetches several thousand pages. Hence, the crawler must make the best use of the little information that it has about the unvisited pages. While, the study of combined contexts derived from many pages is important, in this paper we will concentrate on the quality of link-contexts derived from a single page.

A link-context derivation method takes in a hyperlink URL and the HTML page in which it appears as inputs. The output of the method is a context, if any, for the URL. Traditional context derivation methods view an HTML page as a flat file containing text interspersed with links. This view is a simple extension of techniques used with ordinary documents for obtaining contexts of citations included within the documents. However, recent methods have tried to use an HTML document's tag tree or Document Object Model (DOM) structure to derive contexts [5; 2]. These methods view an HTML page as a tree with `<html>` as its root and different tags and text forming the tree nodes. Figure 1 shows an example of an HTML page and its tag tree representation. As noted earlier, the use of text under the anchor tag (anchor text) as link-context has existed for some time. Do other tags and their hierarchy provide any additional information about the context of a link? The question forms one of the underlying themes of this paper.

When we use an anchor text as the context of a link, we are using all the text in the sub-tree rooted at the anchor tag as the link-context. We may extend the idea and consider all the text in the sub-tree rooted at any of the ancestors of an anchor tag as the context of the anchor URL. We call the node where the sub-tree is rooted as the *aggregation node.* Note that both the context and the corresponding link are within the same sub-tree. Hence, we may view an aggregation node as one that aggregates a link with potentially helpful context. There are many potential aggregation nodes for a single anchor.

2. RELATED WORK

Since the early days of Web many different applications have tried to derive contexts of links appearing on Web pages. McBryan [11] used the anchor text to index URLs in the WWW Worm. Iwazume *et. al.* [10] guided a crawler using the anchor text along with an ontology. SharkSearch [9], used not only the anchor text but some of the text in its "vicinity" to estimate the benefit of following the corresponding link. In a recent work, Chakrabarti *et. al.* [5] suggested the idea of using *DOM offset*, based on the dis-



```
<html>
<head>
<title>Projects</title>
</head>
<body>
<h4>Projects</h4>
<ul>
  <li> <a href="blink.html">LAMP</a> Linkage analysis with multiple processors.</li>
  <li> <a href="nice.html">NICE</a> The network infrastructure for combinatorial exploration.</li>
  <li> <a href="amass.html">AMASS</a> A DNA sequence assembly algorithm.</li>
  <li> <a href="dali.html">DALI</a> A distributed, adaptive, first-order logic theorem prover.</li>
</ul>
</body>
</html>
```

Figure 1: An HTML page and the corresponding tag tree

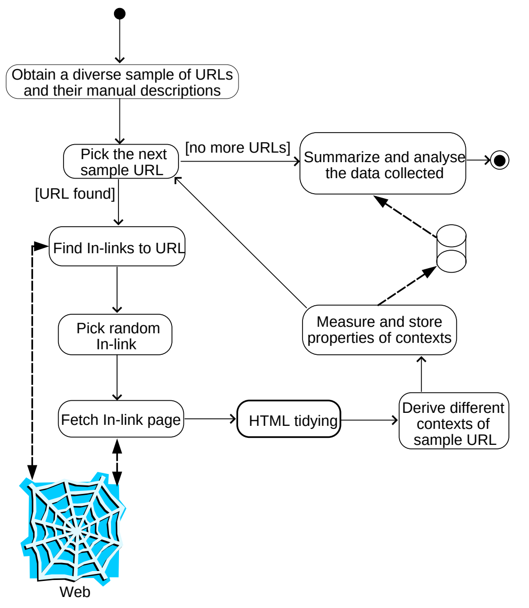

Obtain a diverse sample of URLs and their manual descriptions
Pick the next sample URL
[no more URLs]
Summarize and analyse the data collected
[URL found]
Find In-links to URL
Pick random In-link
Measure and store properties of contexts
Fetch In-link page
HTML tidying
Derive different contexts of sample URL
Web

Figure 2: The framework for study

tance of text tokens from an anchor on a tag tree, to score links for crawling. A DOM offset of zero corresponds to all the text tokens in the sub-tree rooted at the anchor tag. The text tokens to the left of the anchor tag on the tag tree are assigned negative DOM offset values. Similarly, positive DOM offset values are assigned to text tokens on the right of the anchor tag. The offset values are assigned in order — the first token on the right or left is assigned a value of ± 1 and so on. The authors trained a classifier in an attempt to identify the optimal DOM offset window to derive the link contexts. They found the offset window to be no larger than 10 (offset values between -5 and $+5$) for most cases. The mechanism for assigning the DOM offset values made no use of the inherent hierarchy in the tag tree. We will later describe link-context derivation methods that make explicit use of the tag tree hierarchy and compare them against some of the traditional techniques.

Attardi *et. al.* [2] proposed a technique that categorized a Web page based on the context of the URL corresponding to the page. In fact, they exploited the HTML structure to derive a sequence of context strings. However, they restricted the analysis to using only a few HTML tags and by obtaining the text within the tags in a selective and ad-hoc manner. We note that their concept of *context path* has similarities to the idea of *aggregation path* described in section 3.3.1. A context path is a sequence of text strings that are associated with a URL based on their appearance in certain tags and at specific positions in those tags along the hierarchy of the tag tree. The authors also associated a priori semantic meaning with some tags. For example, they argued that the title of a table column or row should be associated with the hyperlinks appearing in the corresponding row or column.

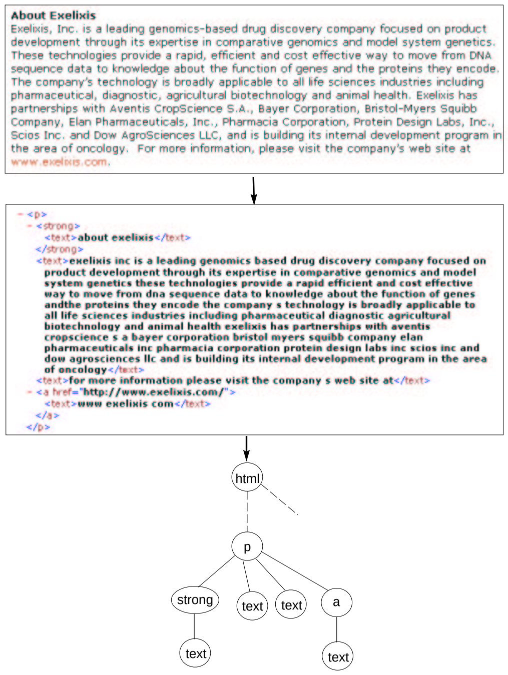
Brin and Page [3] have suggested the use of anchor text to index URLs in their Google search engine. Craswell *et. al.* [7] have shown that using anchor text can provide more effective rankings for site finding tasks. Chakrabarti *et. al.* [4] used a text window of size B bytes around the anchor to obtain the link-context. With an experiment using 5000 Web pages, the authors found that the word “Yahoo” was most likely to occur within 50 bytes of the anchor containing the URL `htpp://www.yahoo.com/`. Davison [8], while using a sample of the DiscoWeb, showed that anchor text (actually it includes words in the vicinity) has high similarity to the destination page as compared to a random page. The author also suggested (based on some results) that enlarging the context of a link by including more words should increase the chance of getting the important terms but at the cost of including more unimportant terms. While the focus of our study is different, we do validate some of the results shown by Davison.

3. FRAMEWORK

3.1 General Methodology

Figure 2 describes the framework in terms of the main steps needed to conduct the study. We first need a sample of URLs that represent a wide variety of topics. In addition, the semantics of the pages corresponding to the URLs must be understood and described by human experts. The manual descriptions will guide us in judging the quality of contexts that are derived for the corresponding URLs. We rely on the manual descriptions because the text content of a Web page may not describe its semantics accurately. Many Web pages are not designed to describe themselves and often contain little text. However, after viewing a page, a human expert can provide us with a good description of what the page is about.

Once we have a sample of Web pages (identified by *sample URLs*) and their manual descriptions, we obtain the in-links for those pages. For each sample URL we use a search engine to find its in-links. We must make sure that the in-link information from the search engine is not stale (i.e. the in-link


About Exelixis
Exelixis, Inc. is a leading genomics-based drug discovery company focused on product development through its expertise in comparative genomics and model system genetics. These technologies provide a rapid, efficient and cost effective way to move from DNA sequence data to knowledge about the function of genes and the proteins they encode. The company's technology is broadly applicable to all life sciences industries including pharmaceutical, diagnostic, agricultural biotechnology and animal health. Exelixis has partnerships with Aventis CropScience S.A., Bayer Corporation, Bristol-Myers Squibb Company, Elan Pharmaceuticals, Inc., Pharmacia Corporation, Protein Design Labs, Inc., Scios Inc. and Dow AgroSciences LLC, and is building its internal development program in the area of oncology. For more information, please visit the company's web site at www.exelixis.com.

```
- <p>
  - <strong>
      <text>about exelixis</text>
    </strong>
    <text>exelixis inc is a leading genomics based drug discovery company focused on
      product development through its expertise in comparative genomics and model
      system genetics these technologies provide a rapid efficient and cost effective
      way to move from dna sequence data to knowledge about the function of genes
      andthe proteins they encode the company s technology is broadly applicable to
      all life sciences industries including pharmaceutical diagnostic agricultural
      biotechnology and animal health exelixis has partnerships with aventis
      cropscience s a bayer corporation bristol myers squibb company elan
      pharmaceuticals inc pharmacia corporation protein design labs inc scios inc and
      dow agrosciences llc and is building its internal development program in the area
      of oncology</text>
    <text>for more information please visit the company s web site at</text>
  - <a href="http://www.exelixis.com/">
      <text>www exelixis com</text>
    </a>
  </p>
```

Figure 3: Mapping an HTML snippet (top) to a tag tree (bottom) via conversion of the HTML to a convenient XML format

points to the sample URL). We choose one random in-link for each sample URL. We fetch the page corresponding to the in-link and call it the *in-link page*. The page may provide us with content that describes the sample URL. Before we process the in-link page for deriving contexts, we "tidy" the HTML and convert it into a convenient XML format. After the conversion, we use multiple methods to derive contexts for the sample URL. The manual description of the sample URL helps us in judging different contexts and hence the methods that produced them. In the current study, we use one set of methods that analyze the HTML tag tree and another set that use arbitrary size text windows around the sample URL in `href`. We summarize the performance data obtained from our sample using a number of metrics. The framework allows for evaluating context derivation methods in general. Note that we only fetch the first `20KB` of each in-link page and we parse the tag tree up to a depth of 19 (root of the tree is at depth 0).

3.2 HTML Tag Tree

Many Web pages contain badly written HTML. For example, a start tag may not have an end tag, or the tags may not be properly nested. In many cases, the `<html>` tag or the `<body>` tag is all together missing from the HTML page. The process of converting a "dirty" HTML document into a well-formed one is called tidying an HTML page[1]. It includes insertion of missing tags and reordering of tags in the "dirty" page. Tidying an HTML page is necessary to make sure that we can map the content onto a tree structure with each node having a single parent. Hence, it is an essential precursor to analyzing an HTML page as a tag tree. Further, the text tokens are enclosed between `<text>...</text>` tags which makes sure that all text tokens appear as leaves on the tag tree. While this step is not necessary for mapping an HTML document to a tree structure, it does provide some simplifications for the analysis. Figure 3 shows the process through which an HTML snippet is converted into a convenient XML format and mapped onto a tag tree.

[1]http://www.w3.org/People/Raggett/tidy/

3.3 Context Derivation Methods

We will now describe two link-context derivation techniques. We later evaluate many versions of each and treat each version as a separate context derivation method.

3.3.1 *Context from Aggregation Nodes*

We fetch each in-link page, tidy it up as explained before and map it onto a tag tree structure. Somewhere, on the tree we will find the anchor tag that contains the sample URL. We call that anchor, the *sample anchor*. If there are more than one sample anchors in the in-link page, we consider only the first one for analysis. Next, we treat each node on the path from the root of the tree to the sample anchor as a potential aggregation node (see Figure 1 shaded nodes). Note that a particular context derivation method would identify one aggregation node for each sample URL. The path from the root of the tree to the sample anchor that contains potential aggregation nodes is called the aggregation path (see Figure 1). Once a node on the aggregation path is set to be an aggregation node the following data is collected from it:

- All of the text in the sub-tree rooted at the node is retrieved as the context of the sample URL. The similarity (described later) of the context to the manual description of the sample URL is measured and called the *context similarity*.

- Number of words in the context is counted. Large size contexts may be too "noisy" and burdensome for some systems.

When we decide on a strategy to assign an aggregation node on a given in-link page, it constitutes a context derivation method. Since we have collected the above data for the entire aggregation path for each of the in-links in our sample, we may evaluate many different methods.

For each in-link page we may have an *optimal* aggregation node that provides the highest context similarity for the corresponding sample URL. If there are multiple aggregation nodes with highest similarity, we pick the one closest to the sample anchor as the optimal one.

3.3.2 *Context from Text Window*

We consider the general technique of using arbitrary size windows of text around the sample URL in `href` as a baseline for comparison. Hence, for each sample URL, we use a window of T words around its first appearance on the in-link page as the context. Whenever possible, the window is symmetric with respect to the `href` containing the sample URL. The window may or may not include the entire anchor text. We derive context using multiple values of T starting from 2 and moving up to 100 with increments of 2 words. Each value of T corresponds to a separate context derivation method.

3.4 Performance Metrics

While manual evaluations of context derivation methods are ideal, such evaluation techniques will not scale easily to thousands of sample URLs and several methods applied to each sample. Hence, we depend on automated techniques. Stop words are removed before computing context similarities and stemming [13] is used to normalize the words. The (cosine) similarity between a context c and a description d is computed as:

$$sim(c, d) = \frac{\vec{v_d} \cdot \vec{v_c}}{\parallel \vec{v_d} \parallel \cdot \parallel \vec{v_c} \parallel}$$

where $\vec{v_c}$ and $\vec{v_d}$ are term frequency based vector representations of the context and the description respectively, $\vec{v_c} \cdot \vec{v_d}$ is the dot (inner) product of the two vectors, and $\parallel \vec{v} \parallel$ is the Euclidean norm of the vector $\vec{v}$.

We measure the performance of each of the context derivation methods using the following performance metrics:

Average context similarity: Given a context derivation method $\mathcal{M}$, we measure the context similarity for each of the sample URLs. The average context similarity for $\mathcal{M}$ is then computed as:

$$avg\_context\_sim(\mathcal{M}) = \frac{1}{n} \sum_{i=1}^{i=n} sim(c_i, d_i)$$

where n is the sample size, c_i is the context derived using the method $\mathcal{M}$ for sample URL i, and d_i is the manual description of the sample URL i. The average context similarity is a measure of average quality of contexts obtained using the method.

Zero-similarity frequency: The zero-similarity frequency measures the frequency with which a method obtains a context that has no similarity to the manual description. The frequency is measured as a percentage of the sample size. While zero similarity may not necessarily mean zero information (due to variable word usage), it does mean that the method failed to provide many of the important words that were used by an expert editor to describe a given page. Such failures could have an affect on the recall of a system based on the method. It is possible that a method provides us with high quality contexts for some URLs but provides us with little or no information about many others. The zero-similarity frequency is computed as:

$$zero\_sim\_freq(\mathcal{M}) = \frac{100}{n} \sum_{i=1}^{i=n} \delta(sim(c_i, d_i))$$

where the $\delta(x)$ is 1 if $x = 0$, and is 0 otherwise. Unlike average context similarity, we would like to have a low value of zero-similarity frequency.

Average context size: This measure can be viewed as the potential cost involved in processing or indexing contexts derived from a method. It is simply computed as:

$$avg\_context\_size(\mathcal{M}) = \frac{1}{n} \sum_{i=1}^{i=n} words(c_i)$$

where $words(c)$ counts the number of words in the context c. A desirable property of a context derivation method may be that it produces reasonable size link-contexts.

Note that we associate 95% confidence intervals with all average values.

4. SAMPLE

We obtain our sample URLs from the Open Directory Project[2] (ODP). The ODP contains a large list of categorized Web pages along with their manual descriptions provided by human editors. Due to the lack of commercial bias and easily available content dump, ODP is a suitable and practical source for obtaining our sample. First, we filter out all the categories of the ODP that have no external URLs since we want only those categories from which we can get sample URLs. Next, we remove the categories that appear under the "World", "Regional" or "International" top-level categories or sub-categories. The categories that appear to be an alphabetical category (e.g. Business: Healthcare: Consultancy: A) are also filtered out. These filters are needed to avoid many semantically similar categories as well as pages with non-English content. Once we have a filtered set of ODP categories (nearly a 100,000 in number), we randomly pick 20,000 categories from them. These categories represent a large spectrum of interests on the Web. From each of the 20,000 categories we pick one external URL randomly. The external URL thus obtained is used as a sample URL. We concatenate the title and the description (provided by the editors) of the external URL and use the combined text as the description of the sample URL. Hence, we get a sample of 20,000 URLs along with their manual descriptions from an equal number of ODP categories.

As mentioned earlier in-links to the sample pages are needed to evaluate different context derivation methods. We used the Google Web API[3] to collect the in-links. Using the API we were able to obtain 1-10 in-links for only 14,960 of the 20,000 sample URLs. Out of the in-links returned by the API, we filter out the ones that are from the `dmoz.org` or `directory.google.com` (a popular directory based on ODP) domains. Further, we fetch each page corresponding to the in-links and compare (ignoring case and non-alphanumeric characters) it against the description of the corresponding sample URL. If an in-link page contains the description we filter it out. The filtering process is necessary to avoid ODP pages or their partial or complete mirrors amongst the in-links. Such in-links, if not filtered, will corrupt our conclusions. Finally, we remove in-link pages that do not contain a link to the sample page (a case of stale information from the search engine). We randomly pick one in-link from the remaining in-links and associate it with the sample URL and its description. Table 1 shows a sample URL, its description and a random in-link. Due to the in-link filtering criteria and limited results from the API, our usable sample reduces to 10,549 sample URLs.

5. ANALYSIS

The sample anchor's absolute depth and the number of its ancestors will vary with in-link pages. However, due to the HTML tidying process, we are assured that each sample anchor will have a parent and a grand-parent node. This is because the tidying process inserts `<html>` and `<body>` tags when they are missing from an HTML page. Also, each tag tree will always have `<html>` tag at its root. For some in-link pages, grand-parent and root node will be the same. However, in many other cases the two will be entirely different.

[2]http://dmoz.org
[3]http://www.google.com/apis/

Table 1: Sample instance

Sample URL	*Description*	*In-link*
`http://www.biotechnologyjobs.com/`	biotechnology jobs database of candidate profiles. automatically matches search to most suitable job in the biotech, pharmaceutical, regulatory affairs or medical device industries.	`http://www.agriscape.com/careers/biotechnology/`

Figure 4 shows the performance of four different context derivation methods. The four methods correspond to setting the aggregation node at the anchor (sample anchor), parent, grand-parent and the root node respectively. In addition to the four methods, we also show the performance of a fictional method that would always find the optimal aggregation node. The error bars in this and the following plots show 95% confidence intervals. All the plots with the same performance metric are drawn to same scale. Hence, non-overlapping error bars indicate a result that is significant at $\alpha = 0.05$ significance level. Figure 4(a) shows that the fictional method will have significantly higher average context similarity than any of the other four methods. We also find that the method that uses anchor text gives the highest average context similarity amongst the other four methods. However, the same method leads to information failure for more than 40% of the sample URLs (Figure 4(b)). The average size of the anchor text is found to be 2.85 ± 0.05 words (slightly higher than that reported by Davison [8]). The method that picks context from the parent node, reduces the zero-similarity frequency by more than 11% at the cost of slightly reducing the average context similarity.

The method that sets the aggregation node at the root is using the entire in-link page as a context of the sample URL. As expected such a strategy produces low quality (similarity) contexts with large number of context words (Figure 4(a),(c)). However, it leads to contexts that are far more likely to contain some information about the sample URL (Figure 4(b)). Hence, our metrics show a trade-off which may manifest itself in form of precision and recall in a system that uses the context derivation methods. Davison's [8] results also show a similar trade-off by incrementally enlarging the size of a link-context.

For our baseline methods that use arbitrary size text windows we find that the best average context similarity is obtained when T is about 14 words. Yet, this best average context similarity is significantly worse than that of the method that just sets the aggregation node at sample anchor or its parent. However, the zero-similarity frequency at $T = 14$ is lower (better) than for the methods that uses the anchor or parent as the aggregation node.

We find that for 65% of the sample the optimal aggregation node can be found at the anchor, the parent or the grand-parent nodes. Also, it is most frequently found at the anchor and the frequency progressively decreases as we move up. This suggests that if we were to look for the optimal aggregation node, we should start from the anchor and make our way upwards if needed. Based on this observation, we suggest a simple *tree climbing* algorithm for setting the aggregation node. This algorithm strives for a minimum number of words in a context. We start from the sample anchor and if it does not have enough words (N) under it, we move a level above it in the tree. This process is continued until we have the minimum number (N) of words in our context or we reach the root of the tree. We test the algorithm for various values of N starting from 0 to 20 with increments of 1 word. The algorithm with each value of N can be considered to be a different context derivation method. For $N = 0$, the aggregation node would be trivially set to the sample anchor. Figure 5 shows the performance for various values of N. We find that the average context similarity with $N = 2, 3, 4$ is significantly higher than any of the methods seen before (*cf.* Figure 4(a) and Figure 5(a)). The best average context similarity is seen at $N = 2$ with average context size of about 53 words and zero-similarity frequency of 20% (half of the method using the anchor text). However, even for $N = 2$ the algorithm's average context similarity is much lower than that of the fictional method that finds the optimal aggregation node. Hence, there is large space for improvement that may come through more sophisticated data mining techniques.

6. CONCLUSION

We introduced a framework to study link-context derivation methods. The framework included a number of metrics to understand the utility of a given link-context. Using the framework, we evaluated a number of methods based on HTML tag tree and arbitrary size text windows. The importance of anchor text as link-context was reaffirmed. However, we also noticed its failure in providing information for large number of sample URLs. The parent of an anchor node, appeared to provide a good balance between the lack of information in the case of anchor node and lack of quality in case of root node. We observed that the optimal aggregation node is likely to be found at depths close to the sample anchor. A simple algorithm that utilizes the tag tree hierarchy and the above observation, provided us with higher quality contexts than any of the other methods considered.

We envisage a number of extensions to the current work. Large search engines have many pages that have several in-links. It is worth studying the performance of context derivation methods when they combine contexts from different source pages. In fact, our evaluation framework provides for such an analysis.

7. ACKNOWLEDGMENTS

The author would like to thank Filippo Menczer, Padmini Srinivasan, Alberto Segre and Robin McEntire for their valuable support and suggestions. Special thanks to Nick Street for providing the hardware that was used to conduct the experiments.

8. REFERENCES

[1] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu. Intelligent crawling on the World Wide Web with arbitrary predicates. In *WWW10*, Hong Kong, May 2001.

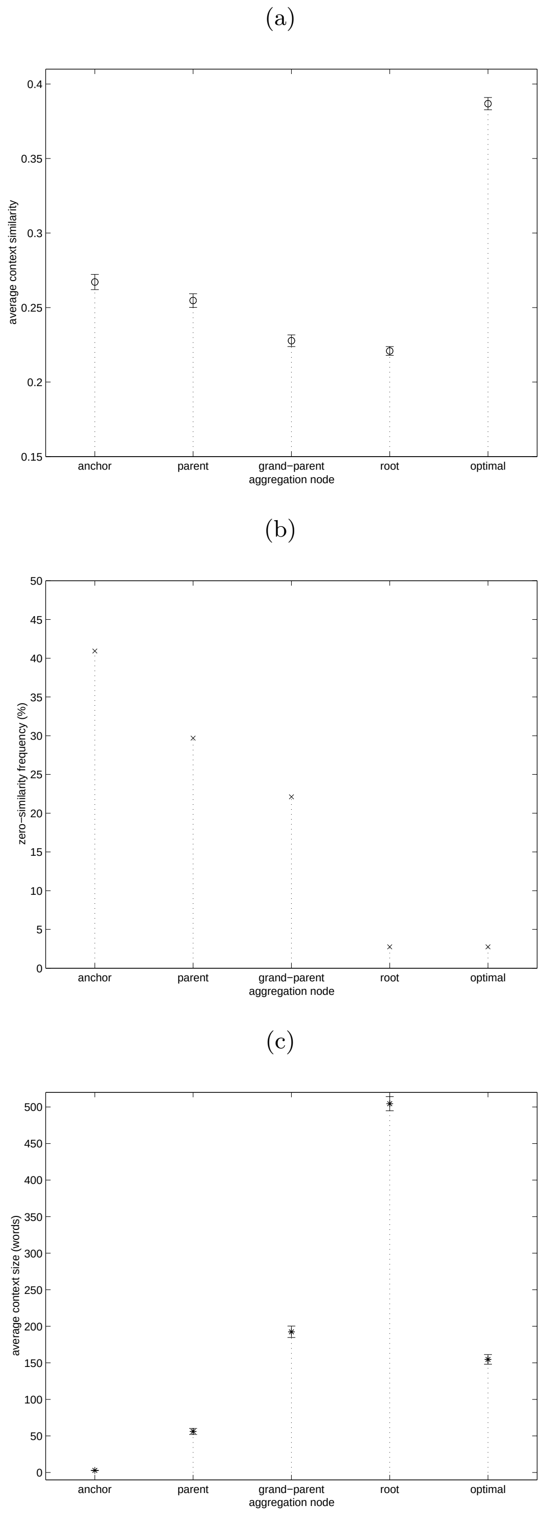


Figure 4: Performance of aggregation node based methods: (a) average context similarity (b) zero-similarity frequency (c) average context size

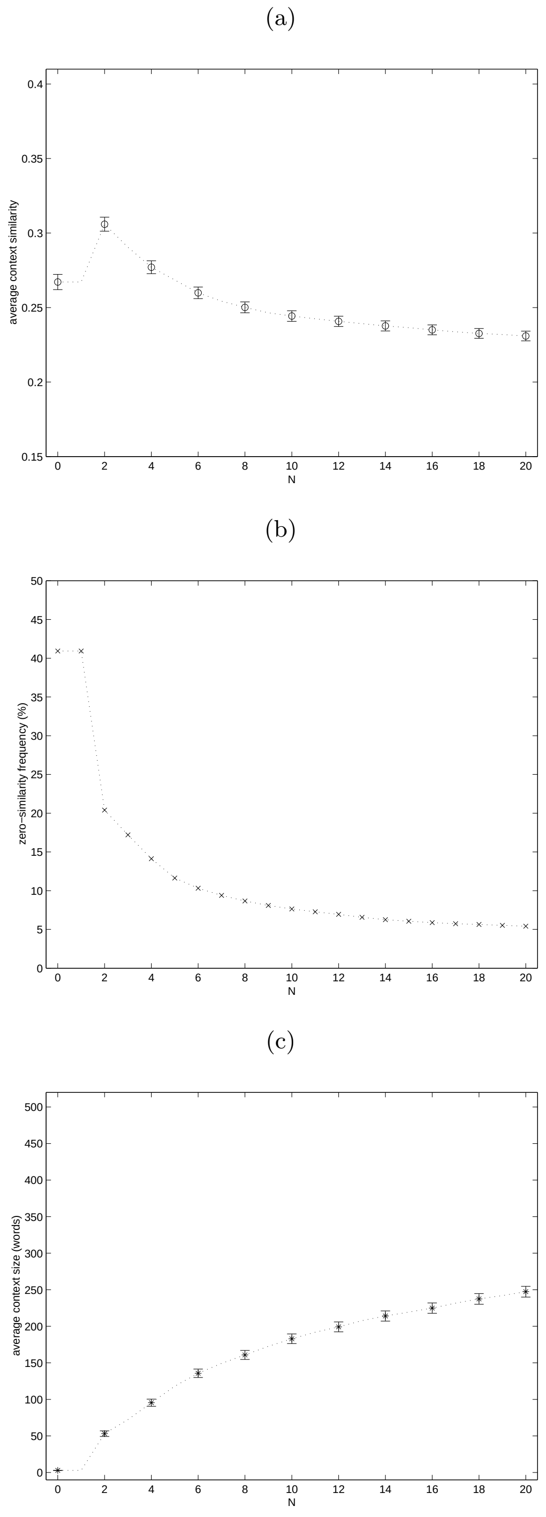


Figure 5: Performance of the tree climbing algorithm: (a) average context similarity (b) zero-similarity frequency (c) average context size

[2] G. Attardi, A. Gullí, and F. Sebastiani. Automatic Web page categorization by link and context analysis. In *Proceedings of THAI-99, 1st European Symposium on Telematics, Hypermedia and Artificial Intelligence*, 1999.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[4] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *WWW7*, 1998.

[5] S. Chakrabarti, K. Punera, and M. Subramanyam. Accelerated focused crawling through online relevance feedback. In *WWW2002*, Hawaii, May 2002.

[6] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

[7] N. Craswell, D. Hawking, and S. Robertson. Effective site finding using link anchor information. In *Proc. 24th Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2001.

[8] B. Davison. Topical locality in the web. In *Proc. 23rd Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2000.

[9] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. The shark-search algorithm — An application: Tailored Web site mapping. In *WWW7*, 1998.

[10] M. Iwazume, K. Shirakami, K. Hatadani, H. Takeda, and T. Nishida. Iica: An ontology-based internet navigation system. In *AAAI-96 Workshop on Internet Based Information Systems*, 1996.

[11] O. McBryan. Genvl and wwww: Tools for taming the Web. In *Proc. 1st International World Wide Web Conference*, 1994.

[12] F. Menczer and R. K. Belew. Adaptive retrieval agents: Internalizing local context and scaling up to the Web. *Machine Learning*, 39(2–3):203–242, 2000.

[13] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

Clustering of Streaming Time Series is Meaningless

Jessica Lin Eamonn Keogh Wagner Truppel

Computer Science & Engineering Department
University of California - Riverside
Riverside, CA 92521

{jessica, eamonn, wagner}@cs.ucr.edu

ABSTRACT

Time series data is perhaps the most frequently encountered type of data examined by the data mining community. Clustering is perhaps the most frequently used data mining algorithm, being useful in it's own right as an exploratory technique, and also as a subroutine in more complex data mining algorithms such as rule discovery, indexing, summarization, anomaly detection, and classification. Given these two facts, it is hardly surprising that time series clustering has attracted much attention. The data to be clustered can be in one of two formats: many individual time series, or a single time series, from which individual time series are extracted with a sliding window. Given the recent explosion of interest in streaming data and online algorithms, the latter case has received much attention.

In this work we make a surprising claim. Clustering of streaming time series is completely meaningless. More concretely, clusters extracted from streaming time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random. While this constraint can be intuitively demonstrated with a simple illustration and is simple to prove, it has never appeared in the literature.

We can justify calling our claim surprising, since it invalidates the contribution of dozens of previously published papers. We will justify our claim with a theorem, illustrative examples, and a comprehensive set of experiments on reimplementations of previous work. Although the primary contribution of our work is to draw attention to the fact that an apparent solution to an important problem is incorrect and should no longer be used, we also introduce a novel method which, based on the concept of time series motifs, is able to meaningfully cluster some streaming time series datasets.

Keywords

Time Series, Data Mining, Clustering, Rule Discovery, Data Streams

1. INTRODUCTION

Time series data is perhaps the most commonly encountered kind of data explored by data miners [26, 35]. Clustering is perhaps the most frequently used data mining algorithm [14], being useful in it's own right as an exploratory technique, and as a subroutine in more complex data mining algorithms [3, 5]. Given these two facts, it is hardly surprising that time series clustering has attracted an extraordinary amount of attention [3, 7, 8, 9, 11, 12, 15, 16, 17, 18, 20, 21, 24, 25, 27, 28, 29, 30, 31, 32, 33, 36, 38, 40, 42, 45]. The work in this area can be broadly classified into two categories:

- **Whole Clustering**: The notion of clustering here is similar to that of conventional clustering of discrete objects. Given a set of individual time series data, the objective is to group similar time series into the same cluster.
- **Subsequence Clustering**: Given a single time series, individual time series (subsequences) are extracted with a sliding window. Clustering is then performed on the extracted time series.

Subsequence clustering is commonly used as a subroutine in many other algorithms, including rule discovery [9, 11, 15, 16, 17, 20, 21, 30, 32, 36, 42, 45], indexing [27, 33], classification [7, 8], prediction [37, 40], and anomaly detection [45]. For clarity, we will refer to this type of clustering as STS (Subsequence Time Series) clustering.

In this work we make a surprising claim. Clustering streaming time series is meaningless! More concretely, clusters extracted from streaming time series are forced to obey a certain constraint that is pathologically unlikely to be satisfied by any dataset, and because of this, the clusters extracted by any clustering algorithm are essentially random. This simple observation has never appeared in the literature.

Our claim is surprising since it calls into question the contributions of dozens of papers. In fact, the existence of so much work based on STS clustering offers an obvious counter argument to our claim. It could be argued: "*Since many papers have been published which use time series subsequence clustering as a subroutine, and these papers produced successful results, time series subsequence clustering must be a meaningful operation.*"

We strongly feel that this is not the case. We believe that in all such cases the results are consistent with what one would expect from random cluster centers. We recognize that this is a strong assertion, so we will demonstrate our claim by reimplementing the most successful (i.e. the most referenced) examples of such work, and showing with exhaustive experiments that these contributions inherit the property of meaningless results from the STS clustering subroutine.

The rest of this paper is organized as follows. In Section 2 we will review the necessary background material on time series and clustering, then briefly review the body of research that uses STS clustering. In Section 3 we will show that STS clustering is meaningless with a series of simple intuitive experiments; then in Section 4 we will explain *why* STS clustering cannot produce useful results. In Section 5 we show that the many algorithms that use STS clustering as a subroutine produce results indistinguishable from random clusters. Since the main contribution of this paper may be considered "negative," we conclude in Section 6 with the demonstration of a simple algorithm that *can* find clusters in at least some trivial streaming datasets. This algorithm is not presented as the best way to find clusters in streaming time series; it is simply offered as an existence proof that such an algorithm exists, and to pave the way for future research.

2. BACKGROUND MATERIAL

In order to frame our contribution in the proper context we begin with a review of the necessary background material.

2.1 Notation and Definitions

We begin with a definition of our data type of interest, time series:

Definition 1. *Time Series*: A time series $T = t_1,\ldots,t_m$ is an ordered set of m real-valued variables.

Data miners are typically not interested in any of the global properties of a time series; rather, data miners confine their interest to subsections of the time series, called subsequences.

Definition 2. *Subsequence*: Given a time series T of length m, a subsequence C_p of T is a sampling of length $w < m$ of contiguous positions from T, that is, $C = t_p,\ldots,t_{p+w-1}$ for $1 \le p \le m - w + 1$.

In this work we are interested in the case where all the subsequences are extracted, and then clustered. This is achieved by use of a sliding window.

Definition 3. *Sliding Windows*: Given a time series T of length m, and a user-defined subsequence length of w, a matrix S of all possible subsequences can be built by "sliding a window" across T and placing subsequence C_p in the p^{th} row of S. The size of matrix S is $(m - w + 1)$ by w.

Figure 1 summarizes all the above definitions and notations.

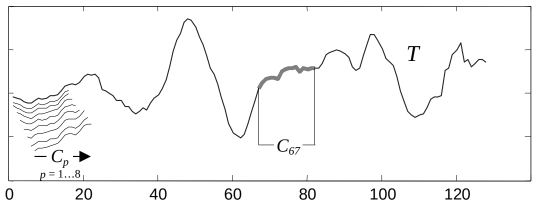


Figure 1. An illustration of the notation introduced in this section: a *time series T* of length 128, a *subsequence* of length w = 16, beginning at datapoint 67, and the first 8 subsequences extracted by a *sliding window*.

Note that while S contains exactly the same information as T, it requires significantly more storage space. This is typically not a problem, since, as we shall see in the next section, the limiting factor tends to be the CPU time for clustering.

2.2 Background on Clustering

One of the most widely used clustering approaches is hierarchical clustering, due to the great visualization power it offers [26, 29]. Hierarchical clustering produces a nested hierarchy of similar groups of objects, according to a pairwise distance matrix of the objects. One of the advantages of this method is its generality, since the user does not need to provide any parameters such as the number of clusters. However, its application is limited to only small datasets, due to its quadratic computational complexity. Table 1 outlines the basic hierarchical clustering algorithm.

```
Algorithm Hierarchical Clustering
1.  Calculate the distance between all objects. Store the
    results in a distance matrix.
2.  Search through the distance matrix and find the two
    most similar clusters/objects.
3.  Join the two clusters/objects to produce a cluster that
    now has at least 2 objects.
4.  Update the matrix by calculating the distances between
    this new cluster and all other clusters.
5.  Repeat step 2 until all cases are in one cluster.
```

Table 1: An outline of hierarchical clustering.

A faster method to perform clustering is k-means [5]. The basic intuition behind k-means (and a more general class of clustering algorithms known as iterative refinement algorithms) is shown in Table 2:

```
Algorithm k-means
1.  Decide on a value for k.
2.  Initialize the k cluster centers (randomly, if necessary).
3.  Decide the class memberships of the N objects by
    assigning them to the nearest cluster center.
4.  Re-estimate the k cluster centers, by assuming the
    memberships found above are correct.
5.  If none of the N objects changed membership in the last
    iteration, exit. Otherwise goto 3.
```

Table 2: An outline of the k-means algorithm.

The k-means algorithm for N objects has a complexity of O(kNrD), with k the number of clusters specified by the user, r the number of iterations until convergence, and D the dimensionality of time series (in the case of STS clustering, D is the length of the sliding window, *w*). While the algorithm is perhaps the most commonly used clustering algorithm in the literature, it has several shortcomings, including the fact that the number of clusters must be specified in advance [5, 14].

It is well understood that some types of high dimensional clustering may be meaningless. As noted by [1, 4], in high dimensions the very concept of nearest neighbor has little meaning, because the ratio of the distance to the nearest neighbor over the distance to the average neighbor rapidly approaches one as the dimensionality increases. However, time series, while often having high dimensionality, typically have a low intrinsic dimensionality [25], and can therefore be meaningful candidates for clustering.

2.3 Background on Time Series Data Mining

The last decade has seen an extraordinary interest in mining time series data, with at least one thousand papers on the subject [26]. Tasks addressed by the researchers include segmentation, indexing, clustering, classification, anomaly detection, rule discovery, and summarization.

Of the above, a significant fraction use streaming time series clustering as a subroutine. Below we will enumerate some representative examples.

- There has been much work on finding association rules in time series [9, 11, 15, 16, 20, 21, 30, 32, 26, 42, 45]. Virtually all work is based on the classic paper of Das et. al. that uses STS clustering to convert real valued time series into symbolic values, which can then be manipulated by classic rule finding algorithms [9].

- The problem of anomaly detection in time series has been generalized to include the detection of surprising or interesting patterns (which are not necessarily anomalies). There are many approaches to this problem, including several based on STS clustering [45].
- Indexing of time series is an important problem that has attracted the attention of dozens of researchers. Several of the proposed techniques make use of STS clustering [27, 33].
- Several techniques for classifying time series make use of STS clustering to preprocess the data before passing to a standard classification technique such as a decision tree [7, 8].
- Clustering of streaming time series has also been proposed as a knowledge discovery tool in its own right. Researchers have suggested various techniques to speed up the clustering [11].

The above is just a small fraction of the work in the area, more extensive surveys may be found in [24, 35].

3. DEMONSTRATIONS OF THE MEANINGLESSNESS OF STS CLUSTERING

In this section we will demonstrate the meaninglessness of STS clustering. In order to demonstrate that this meaninglessness is a product of the way the data is obtained by sliding windows, and not some quirk of the clustering algorithm, we will also do whole clustering as a control [12, 31].

3.1 K-means Clustering

Because k-means is a heuristic, hill-climbing algorithm, the cluster centers found may not be optimal [14]. That is, the algorithm is guaranteed to converge on a local, but not necessarily global optimum. The choices of the initial centers affect the quality of results. One technique to mitigate this problem is to do multiple restarts, and choose the best set of clusters [5]. An obvious question to ask is how much variability in the shapes of cluster centers we get between multiple runs. We can measure this variability with the following equation:

- Let $A=(\bar{a}_1,\bar{a}_2,...,\bar{a}_k)$ be the cluster centers derived from one run of k-means.
- Let $B=(\bar{b}_1,\bar{b}_2,...,\bar{b}_k)$ be the cluster centers derived from a different run of k-means.
- Let $dist(\bar{a}_i,\bar{a}_j)$ be the distance between two cluster centers, measured with Euclidean distance.

Then the distance between two sets of clusters can be defined as:

$$cluster\_distance(A,B) \equiv \sum_{i=1}^{k} \min\left[dist(\bar{a}_i,\bar{b}_j)\right] \;,\; 1 \le j \le k \qquad (1)$$

The simple intuition behind the equation is that each individual cluster center in *A* should map on to its closest counterpart in *B*, and the sum of all such distances tells us how similar two sets of clusters are.

An important observation is that we can use this measure not only to compare two sets of clusters derived for the same dataset, but also two sets of clusters which have been derived from *different* data sources. Given this fact, we propose a simple experiment.

We performed 3 random restarts of k-means on a stock market data set, and saved the 3 resulting sets of cluster centers into set **X**. We also performed 3 random restarts on random walk dataset, saving the 3 resulting sets of cluster centers into set **Y**.

We then measured the average cluster distance (as defined in equation 1), between each set of cluster centers in **X**, to each other set of cluster centers in **X**. We call this number *within_set_X_distance*. We also measured the average cluster distance between each set of cluster centers in **X**, to cluster centers in **Y**; we call this number *between_set_X_and_Y_distance*.

We can use these two numbers to create a fraction:

$$\text{clustering meaningfulness}(\mathrm{X},\mathrm{Y}) \equiv \frac{within\_set\_X\_distance}{between\_set\_X\_and\_Y\_distance} \qquad (2)$$

We can justify calling this number "*clustering meaningfulness*" since it clearly measures just that. If the clustering algorithm is returning the same or similar sets of clusters despite different initial seeds, the numerator should be close to zero. In contrast, there is no reason why the clusters from two completely different, unrelated datasets to be similar. Therefore, we should expect the denominator to be relatively large. So overall we should expect that the value of *clustering meaningfulness*(**X**,**Y**) should be close to zero when **X** and **Y** are sets of cluster centers derived from different datasets.

As a control, we performed the exact same experiment, on the same data, but using subsequences that were randomly extracted, rather than extracted by a sliding window. We call this whole clustering.

Since it might be argued that any results obtained were the consequence of a particular combination of *k* and *w*, we tried the cross product of *k* = {3, 5, 7, 11} and *w* = {8, 16, 32}. For every combination of parameters we repeated the entire process 100 times, and averaged the results. Figure 2 shows the results.

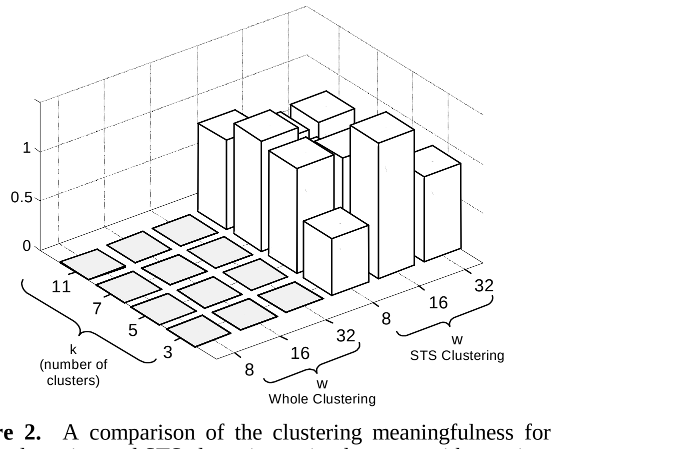


Figure 2. A comparison of the clustering meaningfulness for whole clustering and STS clustering, using k-means with a variety of parameters. The two datasets used were Standard and Poor's 500 Index closing values and random walk data.

The results are astonishing. The cluster centers found by STS clustering on any particular run of k-means on stock market dataset are not significantly more similar to each other than they are to cluster centers taken from random walk data! In other words, if we were asked to perform clustering on a particular stock market dataset, we could reuse an old clustering obtained from random walk data, and no one could tell the difference!

We reemphasize here that the difference in the results for STS clustering and whole clustering in this experiment (and all experiments in this work) are due exclusively to the feature extraction step. In particular, both are being tested on the same dataset, with the same parameters of *w* and *k*, using the same algorithm.

We also note that the exact definition of *clustering meaningfulness* is not important to our results. In our definition, each cluster center in *A* maps onto its closest match in *B*. It is possible therefore that two or more cluster centers from *A* map to one center in *B*, and some clusters in B have no match. However, we tried other variants of this definition, including pairwise matching, minimum matching and maximum matching, together with dozens of other measurements of clustering quality suggested in the literature [14]; it simply makes no significant difference to the results.

3.2 Hierarchical Clustering

The previous section suggests that k-means clustering of STS time series does not produce meaningful results, at least for stock market data. An obvious question to ask is, is this true for STS with other clustering algorithms? We will answer the question for hierarchical clustering here.

Hierarchical clustering, unlike k-means, is a deterministic algorithm. So we can’t reuse the experimental methodology from the previous section exactly; however, we can do something very similar.

First, we note that hierarchical clustering can be converted into a partitional clustering by cutting the first *k* links [29]. Figure 3 illustrates the idea. The resultant time series in each of the *k* subtrees can then be merged into single cluster prototypes. When performing hierarchical clustering, one has to make a choice about how to define the distance between two clusters, this choice is called the linkage method (cf. line 4 of Table 1).

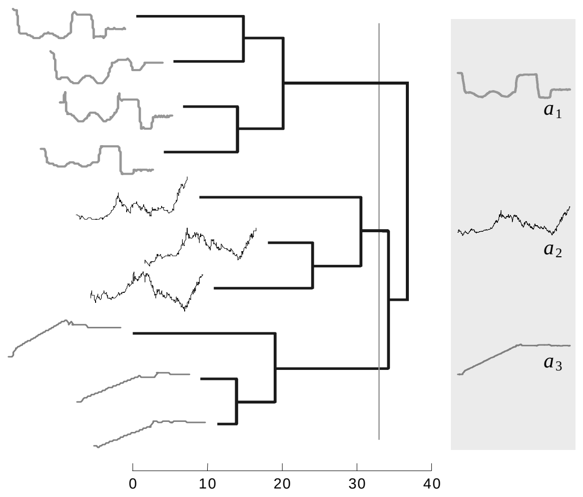


Figure 3. A hierarchical clustering of ten time series. The clustering can be converted to a *k* partitional clustering by “sliding” a cutting line until it intersects *k* lines of the dendrograms, then averaging the time series in the *k* subtrees to form *k* cluster centers (gray panel).

Three popular choices are complete linkage, average linkage and Wards method [14]. We can use all three methods for the stock market dataset, and place the resulting cluster centers into set *X*. We can do the same for random walk data and place the resulting cluster centers into set *Y*. Having done this, we can extend the measure of clustering meaningfulness in Eq. 2 to hierarchical clustering, and run a similar experiment as in the last section, but using hierarchical clustering. The results of this experiment are shown in Figure 4.

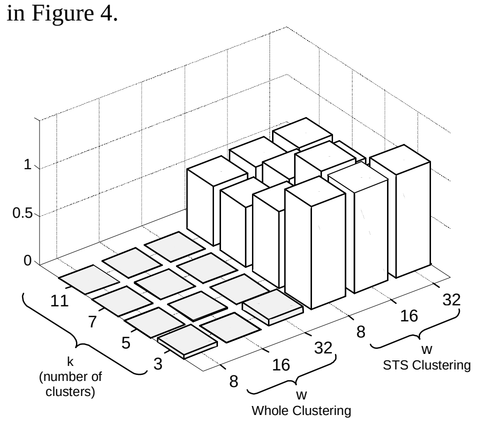


Figure 4. A comparison of the clustering meaningfulness for whole clustering and STS clustering using hierarchical clustering with a variety of parameters. The two datasets used were Standard and Poor's 500 Index closing values and random walk data.

Once again, the results are astonishing. While it is well understood that the choice of linkage method can have minor effects on the clustering found, the results above tell us that when doing STS clustering, the choice of linkage method has as much effect as the choice of dataset! Another way of looking at the results is as follows. If we were asked to perform hierarchical clustering on a particular dataset, but we did not have to report which linkage method we used, we could reuse an old random walk clustering and no one could tell the difference without re-running the clustering for every possible linkage method.

3.3 Other Datasets and Algorithms

The results in the two previous sections are extraordinary, but are they the consequence of some properties of stock market data, or as we claim, a property of the sliding window feature extraction? The latter is the case, which we can simply demonstrate. We visually inspected the UCR archive of time series datasets for the two time series datasets that appear the least alike [23]. The best two candidates we discovered are shown in Figure 5.

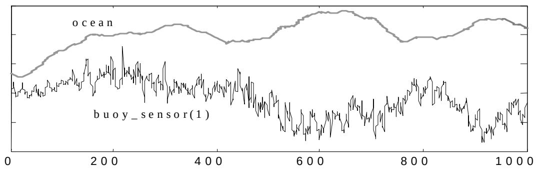


Figure 5. Two subjectively very dissimilar time series from the UCR archive. Only the first 1,000 datapoints are shown. The two time series have very different properties of stationarity, noise, periodicity, symmetry, autocorrelation etc.

We repeated the experiment of Section 3.2, using these two datasets in place of the stock market data and the random walk data. The results are shown in Figure 6.

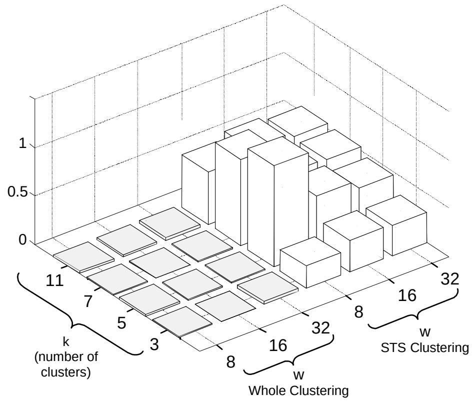


Figure 6. A comparison of the clustering meaningfulness for whole clustering and STS clustering, using k-means with a variety of parameters. The two datasets used were buoy_sensor(1) and ocean.

In our view, this experiment sounds the death knell for clustering of STS time series. If we cannot easily differentiate between the clusters from these two vastly different time series, then how could we possibly find meaningful clusters in any data?

In fact, the experiments shown in this section are just a tiny subset of the experiments we performed. We tested other clustering algorithms, including EM and SOMs [43]. We tested on 42 different datasets [24, 26]. We experimented with other measures of clustering quality [14]. We tried other variants of k-means, including different seeding algorithms. Although Euclidean distance is the most commonly used distance measure for time series data mining, we also tried other distance measures from the literature, including Manhattan, L_∞, Mahalanobis distance and dynamic time warping distance [12, 24, 31]. We tried various normalization techniques, including Z-normalization, 0-1 normalization, amplitude-only normalization, offset-only normalization, no normalization, etc. In every case we are forced to the inescapable conclusion: whole clustering of time series is usually a meaningful thing to do, but sliding window time series clustering is *never* meaningful.

4. WHY IS STS CLUSTERING MEANINGLESS?

Before explaining why STS clustering is meaningless, it will be instructive to visualize the cluster centers produced by both whole clustering and STS clustering. We will demonstrate on the classic Cylinder-Bell-Funnel data [26]. This dataset consists of random instantiations of the eponymous patterns, with Gaussian noise added. While each time series is of length 128, the onset and duration of the shape is subject to random variability. Figure 7 shows one instance from each of the three clusters.

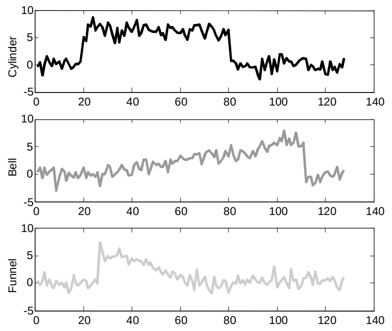


Figure 7. Examples of Cylinder, Bell, and Funnel patterns.

We generated a dataset of 30 instances of each pattern, and performed k-means clustering on it, with k = 3. The resulting cluster centers are show in Figure 8. As one might expect, all three clusters are successfully discovered. The final centers closely resemble the three different patterns in the dataset, although the sharp edges of the patterns have been somewhat "softened" by the averaging of many time series with some variability in the time axis.

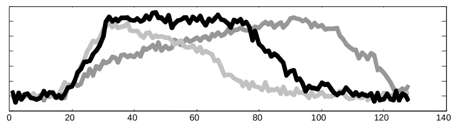


Figure 8. The three final centers found by k-means on the cylinder-bell-funnel dataset. The shapes of the centers are close approximation of the original patterns.

To compare the results of whole clustering to STS clustering, we took the 90 time series used above and concatenated them into one long time series. We then performed STS k-means clustering. To make it easy for the algorithm, we use the exact length of the patterns (w = 128) as the window length, and k = 3 as the number of desired clusters. The cluster centers are shown in Figure 9.

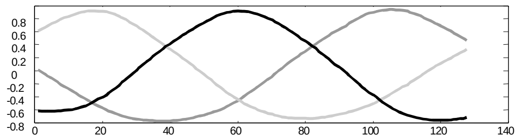


Figure 9. The three final centers found by subsequence clustering using the sliding window approach.

The results are extraordinarily unintuitive! The cluster centers look nothing like any of the patterns in the data; what's more, they appear to be perfect sine waves.

In fact, for $w << m$, we get approximate sine waves with STS clustering regardless of the clustering algorithm, the number of clusters, or the dataset used! Furthermore, although the sine waves are always exactly out of phase with each other by 1/k period, overall, their joint phase is arbitrary, and will change with every random restart of k-means.

This result completely explains the results from the last section. If sine waves appear as cluster centers for every dataset, then clearly it will be impossible to distinguish one dataset's clusters from another. Having demonstrated the inability of STS clustering to produce meaningful results, we have now revealed a new question: why do we always get cluster centers with this special structure?

4.1 A Hidden Constraint

To explain the unintuitive results above, we must introduce a new fact.

Theorem 1: For any time series dataset T, if T is clustered using sliding windows, and $w << m$, then the mean of all the data (i.e. the special case of $k = 1$) will be an approximately constant vector.

In other words, if we run STS k-means on *any* dataset, with $k = 1$ (an unusual case, but perfectly legal), we will always end up with a horizontal line as the cluster center. The proof of this fact is straightforward but long, so we have elucidated it in a separate technical report [41].

We content ourselves here with giving the intuition behind the proof, and offering a visual "proof" in Figure 10.

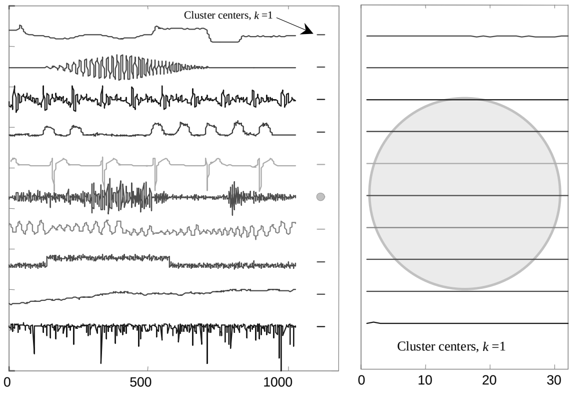


Figure 10: A visual "proof" of Theorem 1. Ten time series of vastly different properties of stationarity, noise, periodicity, symmetry, autocorrelation, etc, are shown at left. The cluster centers for each time series, for $w = 32$, $k = 1$ are shown next to the data. Far right shows a zoom-in that illustrates just how close to a straight line the cluster centers are. While the objects have been shifted for clarity, they have *not* been rescaled in either axis; note the light gray circle in both graphs. The datasets used are, reading from top to bottom: Space Shuttle, Flutter, Speech, Power_Data, Koski_ecg, Earthquake, Chaotic, Cylinder, Random_Walk, and Balloon.

The intuition behind Theorem 1 is as follows. Imagine an arbitrary datapoint t_i somewhere in the time series T, such that $w \leq i \leq m - w + 1$. If the time series is much longer than the window size, then virtually all datapoints are of this type. What contribution does this datapoint make to the overall mean of the STS matrix S? As the sliding window passes by, the datapoint first appears as the rightmost value in the window, then it goes on to appear exactly once in *every* possible location within the sliding window. So the t_i datapoints contribution to the overall shape is the same everywhere and must be a horizontal line. Only those points at the very beginning and the very end of the time series avoid contributing their value to all w columns of S, but these are asymptotically irrelevant. The average of many horizontal lines is clearly just another horizontal line.

The implications of Theorem 1 become clearer when we consider the following well documented fact. For any dataset, the weighted (by cluster membership) average of k clusters must sum up to the global mean. The implication for STS clustering is profound. If we hope to discover k clusters in our dataset, we can only do so if the weighted average of these clusters happen to sum to a constant line! However, there is no reason why we should expect this to be true of *any* dataset, much less *every* dataset. This hidden constraint limits the utility of STS clustering to a vanishing small set of subspace of all datasets.

4.2 The Importance of Trivial Matches

There are further constraints on the types of datasets where STS clustering could possibly work. Consider a subsequence C_p that is a member of a cluster. If we examine the entire dataset for similar subsequences, we should typically expect to find the best matches to C_p to be the subsequences …,C_{p-2}, C_{p-1}, C_{p+1}, C_{p+2} ,… In other words, the best matches to any subsequence tend to be just slightly shifted versions of the subsequence. Figure 11 illustrates the idea, and Definition 4 states it more formally.

Definition 4. *Trivial Match*: Given a subsequence C beginning at position p, a matching subsequence M beginning at q, and a distance R, we say that M is a *trivial match* to C of order R, if either $p = q$ or there does not exist a subsequence M' beginning at q' such that $D(C, M') > R$, and either $q < q' < p$ or $p < q' < q$.

The importance of trivial matches, in a different context, has been documented elsewhere [28]

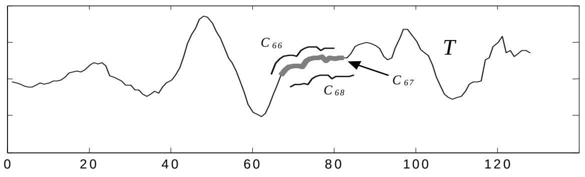


Figure 11: For almost any subsequence C in a time series, the closest matching subsequences are the subsequences immediately to the left and right of C.

An important observation is the fact that different subsequences can have vastly different numbers of trivial matches. In particular, smooth, slowly changing subsequences tend to have many trivial matches, whereas subsequences with rapidly changing features and/or noise tend to have very few trivial matches. Figure 12 illustrates the idea. The figure shows a time series that subjectively appears to have a cluster of 3 square waves. The bottom plot shows how many trivial matches each subsequence has. Note that the square waves have very few trivial matches, so all three taken together sit in a sparsely populated region of w-space. In contrast, consider the relatively smooth Gaussian bump centered at 125. The subsequences in the smooth ascent of this feature have more than 25 trivial matches, and thus sit in a dense region of w-space; the same is true for the subsequences in the descent from the peak. So if clustering this dataset with k-means, $k = 2$, the two cluster centers will be irresistibly drawn to these two "shapes", simple ascending and descending lines.

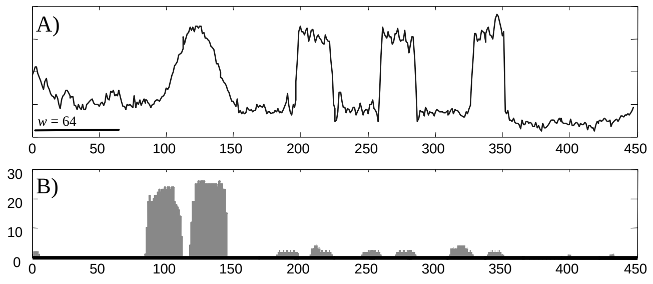


Figure 12: A) A time series T that subjectively appears to have a cluster of 3 noisy square waves. B) Here the i^{th} value is the number of trivial matches for the subsequence C_i in T, where $R = 1$, $w = 64$.

The importance of this observation for STS clustering is obvious. Imagine we have a time series where we subjectively see two clusters: equal numbers of a smooth slowing changing pattern, and a noisier pattern with many features. In w-dimensional space, the smooth pattern is surrounded by many trivial matches. This dense volume will appear to any clustering algorithm an extremely promising cluster center. In contrast, the highly featured, noisy pattern has very few trivial matches, and thus sits in a relatively sparse space, all but ignored by the clustering algorithm. Note that it is not possible to simply remove or "factor out" the trivial matches since there is no way to know beforehand the true patterns.

We have not yet fully explained why the cluster centers for STS clustering degenerate to sine waves (cf Figure 9). However, we have shown that for STS "clustering", algorithms do not really cluster the data. If not clustering, what are the algorithms doing? It is instructive to note that if we perform singular value decomposition on time series, we also get shapes that seem to approximate sine waves [25]. This suggests that STS clustering algorithms are simply returning a set of basis functions that can be added together in a weighted combination to approximate the original data.

An even more tantalizing piece of evidence exists. In the 1920's, "data miners" were excited to find that by preprocessing their data with repeated smoothing, they could discover trading cycles. Their joy was shattered by a theorem by Evgeny Slutsky (1880-1948), who demonstrated that any noisy time series will converge to a sine wave after repeated applications of moving window smoothing [22]. While STS clustering is not exactly the same as repeated moving window smoothing, it is clearly highly related. For brevity we will defer future discussion of this point to future work.

4.3 Is there a Simple Fix?

Having gained an understanding of the fact that STS clustering is meaningless, and having developed an intuition as to why this is so, it is natural to ask if there is a simple modification to allow it to produce meaningful results. We asked this question, not just among ourselves, but also to dozens of time series clustering researchers with whom we shared our initial results. While we considered all suggestions, we discuss only the two most promising ones here.

The first idea is to increment the sliding window by more than one unit each time. In fact, this idea was suggest by [9], but only as a speed up mechanism. Unfortunately, this idea does not help. If the new step size s is much smaller than w, we still get the same empirical results. If s is approximately equal to, or larger than w, we are no longer doing subsequence clustering, but whole clustering. This is not useful, since the choice of the offset for the first window is a critical parameter, and choices that differ by just one timepoint can give arbitrarily different results.

The second idea is to set k to be some number much greater than the true number of clusters we expect to find, then do some post-processing to find the *real* clusters. Empirically, we could not make this idea work, even on the trivial dataset introduced at the beginning of this section. We found that even if k is extremely large, unless it is a significant fraction of T, we still get arbitrary sine waves as cluster centers. In addition, we note that the time complexity for k-means increases with k.

It is our belief that there is no simple solution to the problem of STS-clustering; the definition of the problem is itself intrinsically flawed.

4.4 Necessary Conditions for STS Clustering to Work

We conclude this section with a summary of the conditions that must be satisfied for STS clustering to be meaningful.

Assume that a time series contains k approximately or exactly repeated patterns of length w. Further assume that we happen to know k and w in advance. A necessary (but not necessarily sufficient) condition for a clustering algorithm to discover the k patterns is that the weighted mean of the patterns must sum to a horizontal line, and each of the k patterns must have approximately equal numbers of trivial matches.

It is obvious that the chances of both these conditions being met is essentially zero.

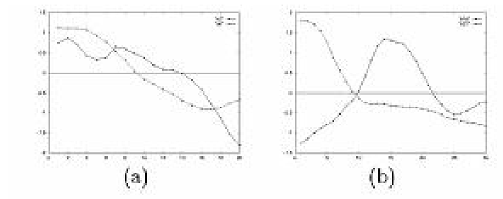
5. A CASE STUDY ON EXISTING WORK

As we noted in the introduction, an obvious counter argument to our claim is the following. "*Since many papers have been published which use time series subsequence clustering as a subroutine, and these papers produce successful results, time series subsequence clustering must be a meaningful operation.*" To counter this argument, we have reimplemented the most influential such work, the Time Series Rule Finding algorithm of Das et. al. [9] (the algorithm is not named in the original work, we will call it TSRF here for brevity and clarity).

5.1 (Not) Finding Rules in Time Series

The algorithm begins by performing STS clustering. The centers of these clusters are then used as primitives that are feed into a slightly modified version of a classic association rule algorithm [2]. Finally, the rules are ranked by their J-measure, an entropy based measure of their significance.

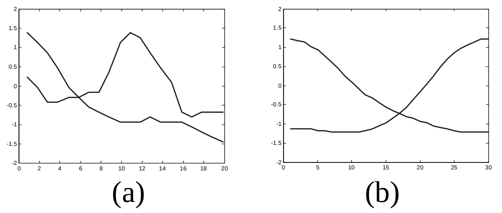
The rule finding algorithm found the rules shown in Figure 13 using 19 months of NASDAQ data. The high values of support, confidence and J-measure are offered as evidence of the significance of the rules. The rules are to be interpreted as follows. In Figure 13 (b) we see that "***if*** *stock rises then falls greatly, follow a smaller rise,* ***then*** *we can expect to see within 20 time units, a pattern of rapid decrease followed by a leveling out*." [9].



w	d	Rule	Sup %	Conf %	J-Mea.	Fig
20	5.5	$7 \Rightarrow^{15} 8$	8.3	73.0	0.0036	(a)
30	5.5	$18 \Rightarrow^{20} 21$	1.3	62.7	0.0039	(b)

Figure 13: Above, two examples of "significant" rules found by Das et. al. (This is a capture of Figure 4 from their paper). Below, a table of the parameters they used and results they found.

What would happen if we used the TSRF algorithm to try to find rules in random walk data, using exactly the same parameters? Since no such rules should exist by definition, we should get radically different results[1]. Figure 14 shows one such experiment; the support, confidence and J-measure values are essentially the same as in Figure 13!



w	d	Rule	Sup %	Conf %	J-Mea	Fig
20	5.5	$11 \Rightarrow^{15} 3$	6.9	71.2	0.0042	(a)
30	5.5	$24 \Rightarrow^{20} 19$	2.1	74.7	0.0035	(b)

Figure 14: Above, two examples of "significant" rules found in *random walk* data using the techniques of Das et. al. Below, we used identical parameters and found near identical results.

This one experiment might have been an extraordinary coincidence; we might have created a random walk time series that happens to have some structure to it. Therefore, for every result shown in the original paper we ran 100 recreations using different random walk datasets, using quantum mechanically generated numbers to insure randomness [44]. In every case the results published cannot be distinguished from our results on random walk data.

The above experiment is troublesome, but perhaps there are simply no rules to be found in stock market. We devised a simple experiment in a dataset that does contain known rules. In particular we tested the algorithm on a normal healthy electrocardiogram. Here, there is an obvious rule that one heartbeat follows another. Surprisingly, even with much tweaking of the parameters, the TSRF algorithm cannot find this simple rule.

The TSRF algorithm is based on the classic rule mining work of Agrawal et.al. [2]; the only difference is the STS step. Since the work of [2] has been carefully vindicated in 100's of experiments on both real and synthetic datasets, it seems reasonable to conclude that the STS clustering is at the heart of the problems with the TSRF algorithm.

These results may appear surprising, since they invalidate the contributions of dozens of papers [9, 11, 15, 16, 17, 20, 21, 30, 32, 36, 42, 45]. However, in retrospect, this result should not really be too surprising. Imagine that a researcher claims to have an algorithm that can differentiate between three types of Iris flowers (Setosa, Virginica and Versicolor) based on petal and sepal length and width [10]. This claim is not so extraordinary, given that it is well known that even amateur botanists and gardeners have this skill [6]. However, the paper in question is claiming to introduce an algorithm that can find rules in stock market time series. There is simply no evidence that any human can do this, in fact, the opposite is true: every indication suggests that the patterns much beloved by technical analysts such as the "calendar effect" are completely spurious [19, 39].

6. A TENTATIVE SOLUTION

The results presented in this paper thus far are somewhat downbeat. In this section we modify the tone by introducing an algorithm that *can* find clusters in some streaming time series. This algorithm is not presented as the best way to find clusters in streaming time series; for one thing, its time complexity is untenable for massive datasets. It is simply offered as an existence proof that such an algorithm exists, and to pave the way for future research.

Our algorithm is motivated by the two observations in Section 4 that attempting to cluster every subsequence produces an unrealistic constraint, and that considering trivial matches causes smooth, low-detail subsequences to form pseudo clusters.

We begin by considering motifs, a concept highly related to clusters. Motifs are overrepresented sequences in discrete strings, for example, in musical or DNA sequences [34]. Classic definitions of motifs require that the underling data be discrete, but in recent work the present authors have extended the definitions to real valued time series [28]. Figure 15 illustrates a visual intuition of a motif, and definition 5 defines the concept more concretely.

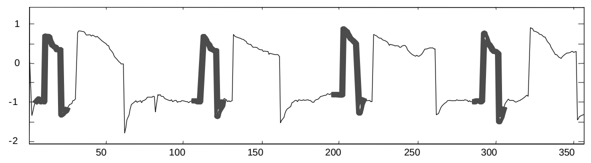


Figure 15: An example of a motif that occurs 4 times in a short section of winding(4) dataset.

Definition 5. *K-Motifs*: Given a time series *T*, a subsequence length *n* and a distance range *R*, the most significant motif in *T* (called *1-Motif*) is the subsequence C_1 that have the highest count of non-trivial matches. The K^{th} most significant motif in *T* (called *K-Motif*) is the subsequence C_K that has the highest count of non-trivial matches, and satisfies $D(C_K, C_i) > 2R$, for all $1 \leq i < K$.

Although motifs may be considered similar to clusters, there are several important differences, a few of which we enumerate here.

- When mining motifs, we must specify an additional parameter *R*.

[1] Note that the shapes of the patterns in Figures 13 and 14 are only very approximately sinusoidal. This is because the time series are relatively short compared the window length. When the experiments are repeated with longer time series, the shapes converge to pure sine waves.

- Assuming the distance *R* is defined as Euclidean, motifs always define circular regions in space, whereas clusters may have arbitrary shapes[2].
- Motifs generally define a small subset of the data, and not the entire dataset.
- The definition of motifs explicitly eliminates trivial matches.

Note that while the first two points appear as limitations, the last two points explicitly counter the two reasons that STS clustering cannot produce meaningful results.

We cannot simply run a *K*-motif detection algorithm in place of STS clustering, since a subset of the motifs discovered might really be a group that should be clustered together. For example, imagine a true cluster that sits in a hyper-ellipsoid. It might be approximated by 2 or 3 motifs that cover approximately the same volume. However, we could run a *K*-motif detection algorithm, with $K >> k$, to extract promising subsequences from the data, then use a classic clustering algorithm to cluster only these subsequences. This idea is formalized in Table 3.

```
Algorithm motif-based-clustering
1.  Decide on a value for $k$.
2.  Discover the $K$-motifs in the data, for $K = k \times c$
    (c is some constant, in the region of about 2 to 30)
3.  Run $k$-means, or $k$ partitional hierarchical clustering, or any
    other clustering algorithm on the subsequences covered by $K$-motifs
```

Table 3: An outline of the motif-based-clustering algorithm.

Line two of the algorithm requires a call to a motif discovery algorithm; such an algorithm appears in [28].

6.1 Experimental Results

We have seen in Section 6 that the clusters centers returned by STS have been mistaken for meaningful clusters by many researchers. To eliminate the possibility of repeating this mistake, we will demonstrate the proposed algorithm on the dataset introduced in Section 4, which consists of the concatenation of 30 examples each of the Cylinder, Bell, Funnel shapes in random order. We would like our clustering algorithm to be able to find clusters centers similar to the ones shown in Figure 8.

We ran the motif based clustering algorithm with $w = 128$, and $k = 3$, which is fair since we also gave these two correct parameters to all the algorithms above. We needed to specify the value of *R*; we did this by simply examining a fraction of our dataset, finding ten pairs of subsequences we found to be similar, measuring the Euclidean distance between these pairs, and averaging the results. The cluster centers found are shown in Figure 16.

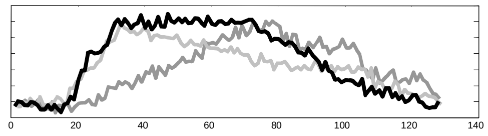


Figure 16: The cluster centers found by the motif-based-clustering algorithm on the concatenated Cylinder-Bell-Funnel dataset. Note the results are very similar to the prototype shapes shown in Figure 7, and the cluster centers found by the whole matching case, shown in Figure 8.

[2] It is true that k-means favors circular clusters, but more generally, clustering algorithms can define arbitrary spaces.

These results tell us that on at least some datasets, we can do meaningful streaming clustering. The fact that this is achieved by working with only a subset of the data, and explicitly excluding trivial matches, further supports our explanations in Sections 4.1 and 4.2, of why STS clustering is meaningless.

7. CONCLUSIONS

We have shown that a popular technique for data mining does not produce meaningful results. We have further explained the reasons why this is so.

Although our work may be viewed as negative, we have shown that a reformulation of the problem can allow clusters to be extracted from streaming time series. In future work we intend to consider several related questions; for example, whether or not the weaknesses of STS clustering described here have any implications for model-based, streaming clustering of time series, or streaming clustering of nominal data [13].

Acknowledgments: We gratefully acknowledge the following people who looked at an early draft of this paper and made useful comments and suggestions: Michalis Vlachos, Frank Höppner, Howard Hamilton, Daniel Barbara, Magnus Lie Hetland, Hongyuan Zha, Sergio Focardi, Xiaoming Jin, Shoji Hirano, Shusaku Tsumoto, Mark Last, and Zbigniew Struzik.

8. REFERENCES

[1] Aggarwal, C., Hinneburg, A., & Keim, D. A. (2001). On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *proceedings of the 8th Int'l Conference on Database Theory*. London, UK, Jan 4-6. pp 420-434.

[2] Agrawal, R., Imielinski, T. & Swami, A. (1993). Mining Association Rules Between Sets of Items in Large Databases. In *proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*. Washington, D.C., May 26-28. pp. 207-216.

[3] Bar-Joseph, Z., Gerber, G., Gifford, D., Jaakkola, T. & Simon, I. (2002). A New Approach to Analyzing Gene Expression Time Series Data. In *proceedings of the 6th Annual Int'l Conference on Research in Computational Molecular Biology*. Washington, D.C., Apr 18-21. pp 39-48.

[4] Beyer, K., Goldstein, J., Ramakrishnan, R. & Shaft, U. (1999). When is Nearest Neighbor Meaningful? In *proceedings of the 7th Int'l Conference on Database Theory*. Jerusalem, Israel, Jan 10-12. pp 217-235.

[5] Bradley, P. S. & Fayyad, U.M. (1998). Refining Initial Points for K--Means Clustering. In *proceedings of the 15th Int'l Conference on Machine Learning*. Madison, WI, July 24-27. pp. 91-99.

[6] British Iris Society, Species Group Staff. (1997). A Guide to Species Irises: Their Identification and Cultivation. *Cambridge University Press*. March, 1997.

[7] Cotofrei, P. (2002). Statistical Temporal Rules. In *proceedings of the 15th Conference on Computational Statistics* - Short Communications and Posters. Berlin, Germany, Aug 24-28.

[8] Cotofrei, P. & Stoffel, K (2002). Classification Rules + Time = Temporal Rules. In *proceedings of the 2002 Int'l Conference on Computational Science*. Amsterdan, Netherlands, Apr 21-24. pp 572-581.

[9] Das, G., Lin, K., Mannila, H., Renganathan, G. & Smyth, P. (1998). Rule Discovery from Time Series. In *proceedings of the 4th Int'l Conference on Knowledge Discovery and Data Mining*. New York, NY, Aug 27-31. pp 16-22.

[10] Fisher, R. A. (1936). The Use of Multiple Measures in Taxonomic Problems. *Annals of Eugenics.* Vol. 7, No. 2, pp 179-188.

[11] Fu, T. C., Chung, F. L., Ng, V. & Luk, R. (2001). Pattern Discovery from Stock Time Series Using Self-Organizing Maps. *Workshop*

Notes of KDD2001 Workshop on Temporal Data Mining. San Francisco, CA, Aug 26-29. pp 27-37.

[12] Gavrilov, M., Anguelov, D., Indyk, P. & Motwani, R. (2000). Mining the Stock Market: Which Measure is Best? In *proceedings of the 6th ACM Int'l Conference on Knowledge Discovery and Data Mining*. Boston, MA, Aug 20-23. pp 487-496.

[13] Guha, S. Mishra, N. Motwani, R. & O'Callaghan, L (2000). Clustering Data Streams. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*. Redondo Beach, CA. Nov 12-14. pp. 359-366.

[14] Halkidi, M., Batistakis, Y. & Vazirgiannis, M. (2001). On Clustering Validation Techniques. *Journal of Intelligent Information Systems* (JIIS), Vol. 17, No. 2-3. pp. 107-145.

[15] Harms, S. K., Deogun, J. & Tadesse, T. (2002). Discovering Sequential Association Rules with Constraints and Time Lags in Multiple Sequences. In *proceedings of the 13th Int'l Symposium on Methodologies for Intelligent Systems*. Lyon, France, June 27-29. pp 432-441.

[16] Harms, S. K., Reichenbach, S. Goddard, S. E., Tadesse, T. & Waltman, W. J. (2002). Data Mining in a Geospatial Decision Support system for Drought Risk Management. In *proceedings of the 1st National Conference on Digital Government*. Los Angeles, CA, May 21-23. pp. 9-16.

[17] Hetland, M. L. & Sætrom, P. (2002). Temporal Rules Discovery Using Genetic Programming and Specialized Hardware. In *proceedings of the 4th Int'l Conference on Recent Advances in Soft Computing*. Nottingham, UK, Dec 12-13.

[18] Honda, R., Wang, S., Kikuchi, T. & Konishi, O. (2002). Mining of Moving Objects from Time-Series Images and its Application to Satellite Weather Imagery. *The Journal of Intelligent Information Systems*, Vol. 19, No. 1, pp. 79-93.

[19] Jensen, D. (2000). Data Snooping, Dredging and Fishing: The dark Side of Data Mining. SIGKDD99 panel report. *ACM SIGKDD Explorations*, Vol. 1, No. 2. pp. 52-54.

[20] Jin, X., Lu, Y. & Shi, C. (2002). Distribution Discovery: Local Analysis of Temporal Rules. In *proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Taipei, Taiwan, May 6-8. pp 469-480.

[21] Jin, X., Wang, L., Lu, Y. & Shi, C. (2002). Indexing and Mining of the Local Patterns in Sequence Database. In *proceedings of the 3rd International Conference on Intelligent Data Engineering and Automated Learning*. Manchester, UK, Aug 12-14. pp 68-73.

[22] Kendall, M. (1976) Time-Series. 2nd Edition. Charles Griffin and Company, Ltd., London.

[23] Keogh, E. (2002). The UCR Time Series Data Mining Archive [http://www.cs.ucr.edu/~eamonn/TSDMA/index.html]. Riverside CA. University of California - Computer Science & Engineering Department

[24] Keogh, E. (2002). Exact Indexing of Dynamic Time Warping. *In proceedings of the 28th International Conference on Very Large Data Bases*. Hong Kong, Aug 20-23. pp 406-417.

[25] Keogh, E. Chakrabarti, K. Pazzani, M & Mehrotra, S. (2001). Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Journal of Knowledge and Information Systems. Vol. 3, No. 3, pp. 263-286.

[26] Keogh, E. & Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *In proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. July 23 - 26, 2002. Edmonton, Alberta, Canada. pp 102-111.

[27] Li, C., Yu, P. S. & Castelli, V. (1998). MALM: A Framework for Mining Sequence Database at Multiple Abstraction Levels. In *proceedings of the 7th ACM CIKM Int'l Conference on Information and Knowledge Management*. Bethesda, MD, Nov 3-7. pp 267-272.

[28] Lin, J. Keogh, E. Patel, P. & Lonardi, S. (2002). Finding motifs in time series. In *the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. July 23 - 26, 2002. Edmonton, Alberta, Canada.

[29] Mantegna., R. N. (1999). Hierarchical Structure in Financial Markets. *European. Physical Journal*. B11, pp. 193-197.

[30] Mori, T. & Uehara, K. (2001). Extraction of Primitive Motion and Discovery of Association Rules from Human Motion. In *proceedings of the 10th IEEE Int'l Workshop on Robot and Human Communication*, Bordeaux-Paris, France, Sept 18-21. pp 200-206.

[31] Oates, T. (1999). Identifying Distinctive Subsequences in Multivariate Time Series by Clustering. *In proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*. San Diego, CA, Aug 15-18. pp 322-326.

[32] Osaki, R., Shimada, M. & Uehara, K. (2000). A Motion Recognition Method by Using Primitive Motions, Arisawa, H. and Catarci, T. (eds.) *Advances in Visual Information Management, Visual Database Systems*, Kluwer Academic Pub. pp 117-127.

[33] Radhakrishnan, N., Wilson, J. D. & Loizou, P. C. (2000). An Alternate Partitioning Technique to Quantify the Regularity of Complex Time Series. *International Journal of Bifurcation and Chaos*, Vol. 10, No. 7. World Scientific Publishing. pp 1773-1779.

[34] Reinert, G., Schbath, S. & Waterman, M. S. (2000). Probabilistic and statistical properties of words: An overview. J. *Comput. Bio*., Vol. 7, pp 1-46.

[35] Roddick, J. F. & Spiliopoulou, M. (2002). A Survey of Temporal Knowledge Discovery Paradigms and Methods. *Transactions on Data Engineering*. Vol. 14, No. 4, pp 750-767.

[36] Sarker, B. K., Mori, T. & Uehara, K. (2002). Parallel Algorithms for Mining Association Rules in Time Series Data. CS24-2002-1 Tech report.

[37] Schittenkopf, C., Tino, P. & Dorffner, G. (2000). The Benefit of Information Reduction for Trading Strategies. *Report Series for Adaptive Information Systems and Management in Economics and Management Science*, July. Report #45.

[38] Steinback, M., Tan, P.N., Kumar, V., Klooster, S. & Potter, C. (2002). Temporal Data Mining for the Discovery and Analysis of Ocean Climate Indices. In the *2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Alberta, Canada. July 23.

[39] Timmermann, A., Sullivan, R. & White, H. (1998). The Dangers of Data-Driven Inference: The Case of Calendar Effects in Stock Returns. *FMG Discussion Papers dp0304*, Financial Markets Group and ESRC.

[40] Tino, P., Schittenkopf, C. & Dorffner, G. (2000). Temporal Pattern Recognition in Noisy Non-stationary Time Series Based on Quantization into Symbolic Streams: Lessons Learned from Financial Volatility Trading. *Report Series for Adaptive Information Systems and Management in Economics and Management Science*, July. Report #46.

[41] Truppel, Keogh, Lin (2003). A Hidden Constraint When Clustering Streaming Time Series. UCR Tech Report.

[42] Uehara, K. & Shimada, M. (2002). Extraction of Primitive Motion and Discovery of Association Rules from Human Motion Data. *Progress in Discovery Science 2002, Lecture Notes in Artificial Intelligence*, Vol. 2281. Springer-Verlag. pp 338-348.

[43] Van Laerhoven, K. (2001). Combining the Kohonen Self-Organizing Map and K-Means for On-line Classification of Sensor data. *Artificial Neural Networks*, Dorffner, G., Bischof, H. & Hornik, K. (Eds.), Vienna, Austria; Lecture Notes in Artificial Intelligence. Vol. 2130, Springer Verlag, pp.464-470.

[44] Walker, J. (2001). HotBits: Genuine Random Numbers Generated by Radioactive Decay. www.fourmilab.ch/hotbits/

[45] Yairi, T., Kato, Y. & Hori, K. (2001). Fault Detection by Mining Association Rules in House-keeping Data. In *proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space*. Montreal, Canada, June 18-21.

A Learning-Based Approach to Estimate Statistics of Operators in Continuous Queries: a Case Study

Like Gao[*] Min Wang[†] X. Sean Wang[*] Sriram Padmanabhan[†]
[*]Dept. of Information and Software Engineering, George Mason University, VA
{lgao, xywang}@gmu.edu
[†]IBM T.J. Watson Research Center, NY, {min, srp}@us.ibm.com

ABSTRACT

Statistic estimation such as output size estimation of operators is a well-studied subject in the database research community, mainly for the purpose of query optimization. The assumption, however, is that queries are ad-hoc and therefore the emphasis has been on capturing the data distribution. When long standing continuous queries on a changing database are concerned, a more direct approach, namely building an estimation model for each operator, is possible. In this paper, we propose a novel learning-based method. Our method consists of two steps. The first step is to design a dedicated feature extraction algorithm that can be used incrementally to obtain feature values from the underlying data. The second step is to use a data mining algorithm to generate an estimation model based on the feature values extracted from the historical data. To illustrate the approach, this paper studies the case of similarity-based searches over streaming time series. Experimental results show this approach provides accurate statistic estimates with a low overhead.

1. INTRODUCTION

Long standing queries, also called continuous queries [28], are those that are typically issued once and then evaluated continuously, every time when the database is updated or some pre-defined events occur. These queries appear in a wide range of applications, especially in applications that need to monitor certain events from data sources. The database research community has shown great interest in these queries over the last decade [6; 18]. Recently, this interest has increased sharply due to the emerging needs of data stream management [3; 9; 10; 11; 19].

As in traditional database systems, it is important to estimate related statistics of the operators used in a continuous query. For example, in cost-based query optimization, an optimizer needs to estimate the cost and output size of operators to determine the most efficient execution plan. In continuous queries, these statistic estimates are more crucial since the database is usually updated frequently and fast query response is required.

A variety of techniques exist to estimate statistics of operators. Among all the statistics, output size estimation has attracted the most attention. Methods in the literature include non-parameric methods (including histograms [15; 21]), parametric methods [25], curve fitting [26], sampling [16], and methods based on query feedback [5]. Among them, histogram method is most commonly used in database systems due to its computational efficiency and independence of data distribution. However, despite technical differences, a common feature of these techniques is that they all aim at capturing the underlying data distribution as precisely as possible under certain storage constraint. Such captured data distributions are then used to estimate the output sizes (selectivities) of operators.

When dealing with continuous queries, we may use a different approach due to the difference between a traditional query and a continuous query. In a traditional query, the database is assumed to be static and the queries are ad-hoc, i.e., the system needs to handle any possible query. This is why most existing techniques focus on capturing the entire underlying data distribution. In a continuous query, however, the query is long standing and the database changes each time the query is evaluated. To estimate the statistics of a given operator in a continuous query, we only need to know the part of the data that is relevant to the specific operator (instead of the entire underlying data distribution) and capture the evolution of it when the database changes. This leads to a different estimation method that does not require the entire data distribution.

In this paper, we reconsider the estimation problem in the context of continuous queries. Also, we do not limit this study to standard operators (e.g., selection and join) and the standard statistics (e.g., output size). Consider the problem of estimating statistics of an operator O in a continuous query, where the query is fixed and the database is changing. The fixed query assumption implies that the output statistic, say $stat_o$, can solely be determined by the input data D, say $stat_o = f_o(D)$, where f_o is a fixed estimation function for operator O. In other words, we only need to know the input data to obtain the statistic directly. In contrast, most existing estimation methods capture the data distribution in advance and determine the statistic of a specific operator at the query evaluation time using the data distribution.

The advantage of this direct method is its potential to get more accurate estimates. However, the continuous query may involve a large volume of input data and data types can be complex. It is not clear how to establish the estimation function f_o directly. In this paper, we decompose the estimation function f_o into two components, namely *fea-*

ture extractor and *statistic estimator*. More specifically, let $stat_o = s_o(e_o(D))$, where e_o is the feature extractor that obtains feature values from the underlying data, and s_o is a statistic estimator that uses the obtained feature values as input. The feature extractor not only reduces the data volume, but also extracts the relevant part of the underlying data. In addition, it converts the underlying data into a data type that will be used for building the statistic estimator. In some cases, the extractor can work incrementally when database is updated to increase the efficiency. The statistic estimator can be built through the use of traditional methods, such as decision tree algorithms, polynomial functions and even histograms.

The feature extractor is usually determined manually and in many cases, a dedicated incremental procedure is developed to obtain feature values in order to reduce the overhead. Strategies to establish the statistic estimator can be classified into two categories: analytic approaches and empirical approaches. In an analytic approach, by analyzing the underlying evaluation procedure of an operator, we may be able to derive the statistic estimator which takes the features as input. However, when the operator is complicated or it involves multiple resources, an empirical approach may be used to establish the statistic estimator, if enough experimental data, i.e., a collection of feature data and corresponding actual statistics, are available. For complex situation, the empirical approach is more suitable because we can directly use historical evaluation data of the continuous query as experimental data, and apply an available data mining algorithm to analyze these data to build the statistic estimator. This is why we call our proposed approach *a learning-based* one and the statistic estimator is also called *estimation model*.

To validate our approach, we study a special kind of continuous query, namely similarity-based search over streaming time series. We design a feature extractor by using data approximations via DFT (Discrete Fourier Transform), and use a decision tree algorithm to establish the estimation model. We then use experiments to assess the performance of our proposed approach. The experimental results show that this approach provides accurate estimates with a low overhead.

The rest of this paper is organized as follows. We introduce the similarity-based search over streaming time series in Section 2 and describe two important statistics in Section 3. We present the detailed learning-based approach in Section 4. In Section 5, we report our experimental results and, in Section 6, give the related work. We conclude this paper in Section 7.

2. SIMILARITY-BASED SEARCH OVER STREAMING TIME SERIES

In this section, we define the notion of a similarity-based search over streaming time series.

There are two types of data involved in such a search, namely *time series* and *streaming time series*. A *time series*, denoted as $\mathsf{P} = \langle p_1, p_2, \ldots, p_n \rangle$, is a finite sequence of n real numbers, with its length n denoted $len(\mathsf{P})$. A *streaming time series*, $\mathsf{S} = \langle s_1, \ldots, s_{t-1}, s_t, \ldots \rangle$, is an infinite sequence of real numbers with the assumption that its values arrive at the database system sequentially. The subscript t indicates the data arrival time of s_t. Given a streaming time series S, the *streaming time series* S *up to time* t denotes the subseries of S before time t, inclusively, i.e., $\langle s_1, \ldots, s_{t-1}, s_t \rangle$.

Given two time series, there are different ways to measure the similarity between them [12]. In this paper, the weighted Euclidean distance,

$$sim(\mathsf{Q}, \mathsf{P}) = \sqrt{\sum_{i=0}^{N-1} (q_{len(\mathsf{Q})-i} - p_{len(\mathsf{P})-i})^2 / N},$$

where $N = min(len(\mathsf{Q}), len(\mathsf{P}))$, is used to define the similarity between time series Q and P. More specifically, this measure is defined on the common N-suffix of two time series, where N is the length of the shorter time series. Under this definition, the more similar the two time series, the smaller the distance between them. This definition can be applied directly to a time series and a streaming time series up to current time without any modification. [1]

Assume Q is a given query time series, $\mathcal{P}$ is a set of time series $\{\mathsf{P}_1, \mathsf{P}_2, \ldots, \mathsf{P}_m\}$ where each P_i ($1 \le i \le m$) in $\mathcal{P}$ is called a pattern, an integer $k > 0$ is called *similarity rank* and a real number $\alpha \ge 0$ is called *similarity threshold*. A pattern P_i in $\mathcal{P}$ is a *k-nearest & α-near neighbor* of Q if there exist at most $k-1$ patterns $\mathsf{P}_j \in \mathcal{P}$ such that $sim(\mathsf{Q}, \mathsf{P}_i) > sim(\mathsf{Q}, \mathsf{P}_j)$, and $sim(\mathsf{Q}, \mathsf{P}_i) \le \alpha$. A k-nearest & α-near neighbor is also called a k-α-near neighbor for short. When α or k is set to some special value, the k-α-near neighbor is exactly the *k-nearest neighbor* (when $\alpha = \infty$) or the *α-near neighbor* (when $k = \infty$).

Definition Given a streaming time series S, a set of pattern time series $\mathcal{P}$, a similarity rank k and a threshold α, the *similarity-based search over stream* S is to find, at each data arrival time t, all k-α-near neighbors of S up to time t from set $\mathcal{P}$.

A continuous query may contain one or more similarity-based searches as its operators. The query optimizer needs to find the most efficient query execution plan so that the system can finish the query evaluation as soon as possible. It is important to provide accurate statistic estimates of each involved similarity-based search to the optimizer.

3. TWO IMPORTANT STATISTICS FOR OPTIMIZING CONTINUOUS QUERIES

In the following, we describe two important statistics that are useful in optimizing a continuous query, namely *cost* and *emptiness* of a similarity-based search. Cost is the execution time of the search, and emptiness is a boolean value to state the search result size being zero or not.

Consider a continuous query that contains the conjunction of two predicates, each involving a similarity-based search. For example, predicate p_1 is to test if a streaming time series

[1]Note our definition of *weighted Euclidean distance* is different from that in [14], where the two time series involved in the distance definition have finite length.

S_1 has a nearest neighbor that is within a given threshold in pattern set $\mathcal{P}_1$ at the current time position.

It is easy to see that the optimal evaluation order of these two predicates can be determined as follows: 1) If the two similarity-based searches have similar evaluation cost and one of them is more likely to result in an empty set (thus the corresponding predicate is false), this search should be evaluated first; 2) If the two similarity-based searches are equally likely to generate an empty set, the one that is less costly should be evaluate first.

From this simple example, we can see that the optimal plan is determined by the cost and emptiness of the two searches. We thus choose cost and emptiness of a similarity-based search as the statistics to be studied and see how to estimate their values.

We do not study output size of a similarity-based search in this paper, although it is one of the most important statistics in query optimization with relational operators. This follows from two observations in a similarity-based search. Firstly, a small similarity rank k may be given, e.g., $k = 1$. This implies that the output size is usually not greater than this value k. Secondly, an optimizer may only need to know the emptiness of a search, and does not care about the actual result size. This happens in many applications with continuous queries, such as monitoring streams for abnormal event detection. Therefore, we decide to study emptiness instead of output size in this paper.

Although we do not study how to estimate the output size of a similarity-based search, we can still apply the strategies discussed hereafter for output size estimation.

4. STATISTIC ESTIMATION

As mentioned in the introduction, we need to consider the execution procedure of the operator to determine the feature extractor. In this paper, we assume a specific procedure is used to evaluate the similarity-based searches. By considering this procedure, we present a strategy below to design the feature extractors. We then describe how to use decision tree methods to establish the estimation models, and discuss how to use these models to obtain the statistic estimates.

The specific procedure for similarity-based search is a direct computation method to find k-α-near neighbors. More specifically, in this method, we first determine a number of pattern time series where the k-α-near neighbors must be within them. We then calculate the distance from each of these patterns to the streaming time series by directly applying the similarity formula. Having all these distances, we can quickly determine which patterns are the actual k-α-near neighbors.

4.1 Feature Extractors

Feature extractor obtains feature values from the underlying data with the goal of minimizing estimation overhead. In order to reduce the overhead, we develop an incremental approach to design the feature extractor by using data approximations of time series and streaming time series.

We first describe how to approximate time series and streaming time series. The approximation method that we use in this paper is similar to that in [8], where a small number of significant DFT coefficients are used to represent a time series or a subseries of a streaming time series. We call these coefficients as the *approximations via DFT*.

For the static pattern time series, i.e., all patterns are stored in a database and do not change during the continuous query evaluation, we can pre-process them to find their approximations off-line. Since patterns in the database may have different lengths, we need to use DFT with the corresponding length to perform the approximation for each pattern.

For a streaming time series, we can only obtain its approximation in an on-line fashion when each new data value arrives. To reduce the overhead of computing DFT at each time position, we use an incremental approach to get the approximation of the N-suffix of the streaming time series, where N is the length of a pattern time series. Specifically, we use Sliding DFT [27], given as the following proposition, to quickly calculate the DFT coefficients. We can see both the space requirement and computation cost of this incremental DFT calculation are very small.

Proposition Given a DFT length N and a streaming time series $\mathsf{S} = \langle s_1, \ldots \rangle$, let $Y = \langle y_0, y_1, \ldots, y_{N-1} \rangle$ be the DFT coefficients of the N-suffix of S at time $t-1$, and $Y' = \langle y'_0, y'_1, \ldots, y'_{N-1} \rangle$ be those coefficients of S at time t, then Y' can be found by $y'_n = e^{j2\pi n/N}[y_n + (s_t - s_{t-N})/N]$ for $n = 0 \ldots N-1$.

Two facts should be noticed about the approximation. The first is that corresponding to each distinct pattern length, there is an approximation of the streaming time series. This means that a streaming time series may have multiple versions of approximation at any given time position. The second is that we can calculate a distance lower bound for two time series using their approximations via DFT, as given in the following proposition.

Proposition For two time series Q and P where $N = min(len(\mathsf{Q}), len(\mathsf{P}))$, if let $\hat{\mathsf{Q}}$ and $\hat{\mathsf{P}}$ denote the approximations of the N-suffix of Q and P, respectively, and let n denote the dimensionality of the approximations, then

$$\sqrt{\frac{n}{N}} sim(\hat{\mathsf{Q}}, \hat{\mathsf{P}}) \leq sim(\mathsf{Q}, \mathsf{P}).$$

Proof: Let the N-suffix of Q and P be $\langle x_1, x_2, \ldots, x_N \rangle$ and $\langle y_1, y_2, \ldots, y_N \rangle$, respectively. Let their corresponding DFT coefficients Q_N and P_N be $\langle X_1, X_2, \ldots, X_N \rangle$ and $\langle Y_1, Y_2, \ldots, Y_N \rangle$, respectively. Using Parseval's Theorem [22], we have:

$$sim(\mathsf{Q}, \mathsf{P}) = \sqrt{\frac{\sum_{i=1}^{N}(x_i - y_i)^2}{N}} = \sqrt{\frac{\sum_{i=1}^{N}|X_i - Y_i|^2}{N}}.$$

For the approximations $\hat{\mathsf{Q}}$ and $\hat{\mathsf{P}}$ with dimensionality of n, $n \leq N$, we have

$$sim(\hat{\mathsf{Q}}, \hat{\mathsf{P}}) = \sqrt{\frac{\sum_{i=1}^{n}|X_{mi} - Y_{mi}|^2}{n}} \leq \sqrt{\frac{\sum_{i=1}^{N}|X_i - Y_i|^2}{n}}.$$

The proposition follows. $\square$

In this paper, to simplify the illustration, we only use the first DFT coefficient to approximate the pattern time series and streaming time series, i.e., $n = 1$ in the above inequation.

We now consider how to extract feature values for cost estimation by using these approximations. First, we pre-process the pattern time series by sorting all of them by their approximations (the first DFT coefficients) in increasing order, and we re-assign the corresponding ranks as their new indices. We perform this task off-line since no streaming time series is involved. We then determine as features two pattern indices, namely $LowerIndex$ and $UpperIndex$, whenever a new value of the streaming time series arrives. These two indices determine a range of patterns. The reason that we use these two indices as features is because the number of patterns in the range is closely related to cost of the similarity-based search.

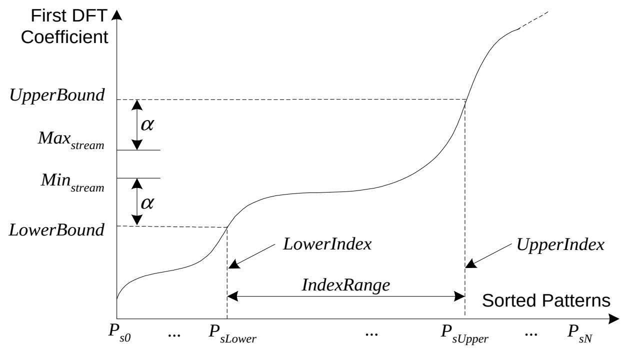


Figure 1: Cost feature extractor.

In Figure 1, we illustrate the detailed steps of obtaining these two feature values in an on-line fashion. In this figure, the x-axis represents the indices (ranks) of the pattern time series, the y-axis represents the approximation values and the monotonic increasing curve represents the first DFT coefficients of all patterns in the pattern set after we perform the sorting. We carry out the feature extraction as follows. First, we use the incremental DFT method to get multiple approximations of the streaming time series up to current time. (Each approximation corresponds to a distinct length for the pattern in the pattern set.) Assume these approximations have a minimum value of Min_{stream} and a maximum value of Max_{stream}. Now we can determine a distance range by computing $LowerBound = Min_{stream} - \alpha$ and $UpperBound = Max_{stream} + \alpha$. Assume p is a pattern in the pattern set and p is an α-near neighbor of the streaming time series at current time position. Based on the proposition on distance lower bound, we can see that the first DFT coefficient of p must be in the range of $[LowerBound, UpperBound]$. Finally, we use binary search to quickly find the corresponding index range $[LowerIndex, UpperIndex]$.

We use the similar procedure to extract features for the emptiness estimation. We also choose these two pattern indices as features and also use the process in Figure 1 to get their values, since the emptiness of the search should be closely related to the index range. In addition, we consider a unique property, namely continuity [10] of a similarity-based search over streaming time series. This property says that successive evaluations of a search are likely to result in the same output. This implies that if one evaluation of the search is empty (non-empty), then the following one is also likely to be empty (non-empty). Hence, we can use previous evaluation emptiness to estimate for current one. So besides the above two pattern indices, we also use a number of most recent emptiness as features.

In summary, we use as features two pattern indices and the emptiness of a number of previous evaluations. In the implementation, we use an additional feature, $IndexRange = UpperIndex - LowerIndex$. Although this feature seems to be redundant, it does make the estimation models more concise and more accurate. We tabulate all the features for each extractor in Figure 2.

Extractor	Features
Cost	$LowerIndex$, $UpperIndex$ and $IndexRange$
Emptiness	$LowerIndex$, $UpperIndex$, $IndexRange$ and a number of previous $Emptiness$

Figure 2: Features used.

4.2 Estimation Models

Building and using estimation models are straightforward. It is a typical data mining application.

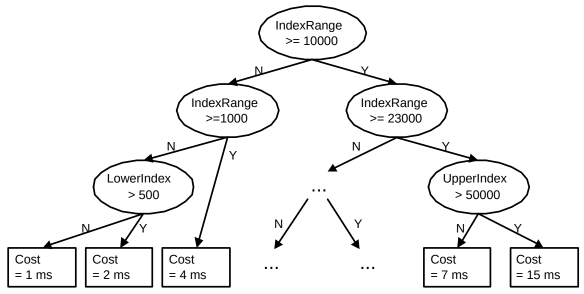


Figure 3: A sample cost estimation model

We first look at how to establish the estimation models. We first run the similarity-based search for a sufficient number of times as the streaming time series come in. During each evaluation, we use feature extractors to get the feature values and collect the actual evaluation cost and emptiness. Having these collected data as training data set, we then apply data mining algorithms such as C4.5 decision tree [23] to establish the estimation models. We give a sample decision tree for the cost estimation in Figure 3.

Using the estimation models to estimate the statistics is also straightforward. When new values of the streaming time series arrive, we use the feature extractors incrementally to obtain the feature values, and then feed them into corresponding estimation models to get the estimates.

5. EXPERIMENTAL RESULTS

In this section, we report our experimental results that demonstrate the effectiveness of the proposed learning-based approach. The experiment is performed on a desktop box with Pentium III 1.2GHz CPU and 512M memory, running Windows XP Professional, and the code is written in programming language C++. The data mining algorithm used to build estimation models is C4.5 decision tree algorithm [23]. All datasets are loaded into memory in advance and hence the cost measured only refers to the computational cost.

Datasets
Both pattern time series and the streaming time series are synthetic data. The 10^5 pattern time series are generated independently via a random-walk function. Each pattern has a length between 50 and 300. The streaming time series is constructed by randomly picking up some pattern time series from the above and concatenating them into one long sequence. In order to fill the gap between two successive patterns, some values are interpolated in between via PCHIP (Piecewise Cubic Hermite Interpolating Polynomial) functions. In addition, some patterns appear more often than others in order to simulate periodic phenomena in real world streams. Finally, white noise is added to the streaming time series.

Pattern time series are pre-processed to obtain their approximations via DFT and they are sorted by their first DFT coefficients. The approximations of streaming time series are calculated on fly with the incremental DFT approach given in Section 4.1. We will only use the first DFT coefficients of streaming time series and pattern time series to derive the features.

Performance measures
The performance (accuracy) of cost and emptiness estimation models is given as follows:

- The accuracy of the cost estimation model: the ratio that the estimated cost is within a given tolerance of the actual cost [2].
- The accuracy of the emptiness estimation model: the ratio that the estimation model correctly predicts the emptiness.

Continuous queries
We use similarity-based searches that ask for 1-α-near neighbors of the streaming time series as the continuous query, where α varies from 0.05 to 0.1, step by step. Here we fix the similarity rank $k = 1$ for two reasons. The first is that a small change of k does not make much difference to the evaluation cost. This can be seen from our feature extraction procedure, where only α is used. The second is that emptiness estimation of a search is independent of the rank value k. Indeed, if and only if a 1-α-near neighbor search results in an empty (non-empty) set, any k-α-near neighbor search will also result in an empty (non-empty) set.

Experimental results
In the experiments, we evaluate the above similarity-based search for 2,000 time positions, which we call a *run*. For each run, we choose the similarity threshold α between 0.05 and 0.1. In each run, we collect feature values, actual cost

[2]The tolerance is 1ms in the experiments reported, which is less 10% of the average cost.

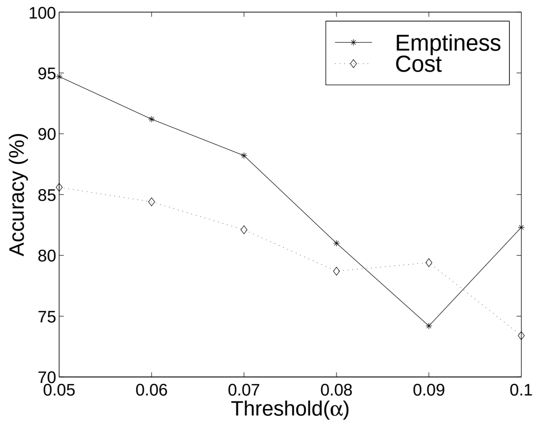


Figure 4: Performance of the models.

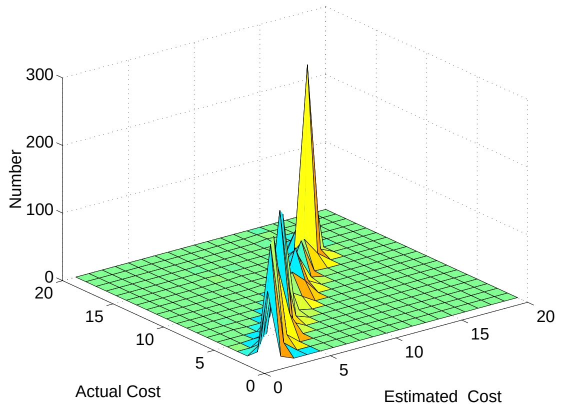


Figure 5: Distribution of estimation (cost).

and emptiness of the search at each time position. We then apply the C4.5 algorithm on 70% of the collected data to find out the two estimation models. We use the remaining 30% data to test the accuracy of the estimation models. The results are given in Figure 4.

From Figure 4, we can see that the emptiness estimation model has a relatively high accuracy. An interesting trend is that as the threshold increases, the accuracy goes down first and then goes up. This follows the fact that when threshold is very small (very big), the search is very likely to result in an empty (non-empty) output. In these cases, the estimation model can accurately estimate the emptiness. When the threshold takes values in between, the emptiness is more sensitive to the feature values. In this case, the estimation model has a relatively low accuracy.

The major reason for performance drop is that we only use one DFT coefficient to derive features. This introduces errors to the feature values. These errors have different impact on estimate accuracy for different similarity threshold values. When the impact becomes large, the accuracy of the estimation drops and so does the performance.

The accuracy of the cost estimation model is around 70% to 80%, as shown in Figure 4. Note that a mis-classified cost does not necessarily mean a bad estimate, since it may be still very close to the actual cost. To assess this, we run the similarity-based search with threshold $\alpha = 0.1$ for another 2,000 time positions, and study the cost estimation distribution. The result is shown in Figure 5. In this figure, each pair of actual cost and estimated cost corresponds to one evaluation. The vertical value indicates the total number of times that a pair of estimated cost and actual cost appears. Since all non-zero height values are clustered to the diagonal in this figure, this means that in each evaluation, the estimated cost is very close to the actual cost.

For the search with $\alpha = 1$, the averaged time to find the two statistic estimates is 0.05ms. Compared to the overall search time of 15ms on average, this is a very small overhead. Experiments with other thresholds also confirm that our proposed approach has very low overhead.

6. RELATED WORK

Continuous queries have long been attracting the attention of database researchers. Terry et al. [28] perhaps were the first one to introduce the notion of "continuous queries" for a class of queries that are issued once and then run "continuously" over databases. Later Liu et al. [17; 18] also presented a similar concept, termed "continual query". Recently, due to the emerging needs of data stream management systems, the interest in continuous queries, especially in continuous queries over streams, has increased sharply. One direction in this field is to design stream data process systems that support continuous queries, i.e., Chen et al. presented the design of the NiagraCQ system [6; 7]; Babu and Widom [4], Madden and Franklin [19] also reported system architecture and related issues for dealing with continuous queries. Another direction is query plan optimization for continuous queries [20; 29]. Some more related details of continuous queries and streams can be found in the tutorial [3; 11].

In this paper, we studied the estimation problem for similarity-based searches over streaming time series. Most previous work [1; 2; 24] on similarity-based searches was concentrated on ad-hoc queries over stationary data sources, and in most cases, the time series stored in databases are supposed to have the same length. Although [9; 10] also study similarity-based searches over streaming time series, their main contribution is on how to efficiently evaluate similarity-based search on a single streaming time series. In contrast, this paper considers the scenario that a continuous query involves multiple streaming time series and multiple similarity-based searches, and focuses on estimating statistics that are necessary for optimizing such a query.

This paper is also closely related to the literature on statistic estimation, which has been addressed in the introduction section. Besides, another interesting work [13] used machine learning algorithms on the training queries to "re-vitalize" the existing output size estimation methods whose performance previously deteriorated due to high update loads. However, all these work is based on modeling the data distribution. In contrast, in this paper, our learning-based approach is a direct estimation method without the step of modeling data distribution.

7. CONCLUSIONS

In this paper, we introduced a learning-based framework to estimate statistics of an operator in a continuous queries. The framework consists of extracting features and building learning model using a training workload. We studied the case of similarity-based query over streaming time series to validate this approach. Our experiments demonstrated the effectiveness of our proposed approach.

It should be noted that there may be different ways to design a feature extractor. It is important for a feature extractor to be well constructed so that the chosen method can lead to accurate estimation models. It will be interesting to study the strategy on how to find an "optimal" pair of feature extractor and estimation model, which may yield the best estimation accuracy under given resource constraint. Another interesting research direction is to apply this learning-based approach to other types of continuous queries.

8. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *Procs. of FODO*, pages 69–84, 1993.

[2] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *The VLDB Journal*, pages 490–501, 1995.

[3] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Procs. of PODS*, pages 1–16, 2002.

[4] S. Babu and J. Widom. Continuous queries over data streams. In *SIGMOD Record, Sept. 2001*, 2001.

[5] Chungmin Melvin Chen and Nick Roussopoulos. Adaptive selectivity estimation using query feedback. In *Procs. of ACM-SIGMOD*, pages 161–172, 1994.

[6] J. Chen, D. J. DeWitt, and J. F. Naughton. Design and evaluation of alternative selection placement strategies in optimizing continuous queries. In *ICDE Conference*, 2002.

[7] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagaraCQ: a scalable continuous query system for Internet databases. In *Procs. of ACM-SIGMOD*, pages 379–390, 2000.

[8] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Procs. of ACM-SIGMOD*, pages 419–429, 1994.

[9] L. Gao and X. S. Wang. Continually evaluating similarity-based pattern queries on a streaming time series. In *Procs. of ACM-SIGMOD*, 2002.

[10] L. Gao, X. S. Wang, and Z. Yao. Evaluating continuous nearest neighbor queries for streaming time series via pre-fetching. In *CIKM*, pages 485–492, 2002.

[11] Minos N. Garofalakis, Johannes Gehrke, and Rajeev Rastogi. Querying and mining data streams: You only get one look. In *Procs. of ACM-SIGMOD*, 2002.

[12] D. Gunopulos and G. Das. Time series similarity measures. In *Procs. of KDD*, pages 243–307, 2000.

[13] Banchong Harangsri, John Shepherd, and Anne H. H. Ngu. Query size estimation using machine learning. In *Database Systems for Advanced Applications*, pages 97–106, 1997.

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

[15] Robert Kooi. *The Optimization of Queries in Relational Databases*. PhD thesis, Case Western Reserve University, 1980.

[16] Richard J. Lipton, Jeffrey F. Naughton, and Donovan A. Schneider. Practical selectivity estimation through adaptive sampling. In *Procs. of ACM-SIGMOD*, pages 1–11, 1990.

[17] L. Liu, C. Pu, R. S. Barga, and T. Zhou. Differential evaluation of continual queries. In *International Conference on Distributed Computing Systems*, pages 458–465, 1996.

[18] L. Liu, C. Pu, and W. Tang. Continual queries for internet scale event-driven information delivery. *IEEE TKDE*, 11(4):610–628, 1999.

[19] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *ICDE Conference*, 2002.

[20] Samuel Madden, Mehul Shah, Joseph M. Hellerstein, and Vijayshankar Raman. Continuously adaptive continuous queries over streams. In *Procs. of ACM-SIGMOD*, pages 49–60, 2002.

[21] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In *Procs. of ACM-SIGMOD*, pages 448–459, 1998.

[22] A.V. Oppenheim and R.W. Schafer. *Digital Signal Processing*. Prentice-Hall, Inc, 1975.

[23] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgann Kaufmann, 1993.

[24] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *Procs. of ACM-SIGMOD*, pages 13–25, 1997.

[25] P.G. Selinger, M.M. Astrahan, D.D. Chainberlin, R.A. Lorie, and T.G. Price. Access path selection in a relational database management system. In *Procs. of ACM-SIGMOD*, pages 23–34, 1979.

[26] W. Sun, Y. Ling, N. Rishe, and Y. Deng. An instant and accurate size estimation method for joins and selection in a retrieval-intensive environment. In *Procs. of ACM-SIGMOD*, pages 79–88, 1993.

[27] Jeffrey D. Taft. The Sliding DFT Page. On-line. http://www.nauticom.net/www/jdtaft/DFT_increm.htm.

[28] D. Terry, D. Goldberg, D. Nichols, and B. Oki. Continuous queries over append-only databases. In *Procs. of ACM-SIGMOD*, pages 321–330, 1992.

[29] S. D. Viglas and Jeffrey F. Naughton. Rate-based query optimization for streaming information sources. In *Procs. of ACM-SIGMOD*, pages 37–48, 2002.

Using transposition for pattern discovery from microarray data

François Rioult
GREYC CNRS UMR 6072
Université de Caen
F-14032 Caen, France
frioult@info.unicaen.fr

Jean-François Boulicaut
LIRIS CNRS FRE 2672
INSA Lyon
F-69621 Villeurbanne, France
jfboulic@lisi.insa-lyon.fr

Bruno Crémilleux
GREYC CNRS UMR 6072
Université de Caen
F-14032 Caen, France
bruno@info.unicaen.fr

Jérémy Besson
LIRIS CNRS FRE 2672
INSA Lyon
F-69621 Villeurbanne, France
jbesson@lisi.insa-lyon.fr

ABSTRACT

We analyze expression matrices to identify a priori interesting sets of genes, e.g., genes that are frequently co-regulated. Such matrices provide expression values for given biological situations (the lines) and given genes (columns). The frequent itemset (sets of columns) extraction technique enables to process di cult cases (millions of lines, hundreds of columns) provided that data is not too dense. However, expression matrices can be dense and have generally only few lines w.r.t. the number of columns. Known algorithms, including the recent algorithms that compute the so-called condensed representations can fail. Thanks to the properties of Galois connections, we propose an original technique that processes the transposed matrices while computing the sets of genes. We validate the potential of this framework by looking for the closed sets in two microarray data sets.

1. INTRODUCTION

We are now entering the post-genome era and it seems obvious that, in a near future, the critical need will not be to generate data, but to derive knowledge from huge data sets generated at very high throughput. Di erent techniques (including microarrays and SAGE) enable to study the simultaneous expression of (tens of) thousands of genes in various biological situations. The data generated by those experiments can then be seen as expression matrices in which the expression level of genes (the columns) are recorded in various biological situations (the lines). Various knowledge discovery methods can be applied on such data, e.g., the discovery of sets of co-regulated genes, also known as synexpression groups [12]. These sets can be computed from the frequent sets in the boolean matrices coding for the expression data (see Table 1).

One attribute a_i is attributed the value true (1) to represent the over- (or under-) expression of gene i in that particular situation.

Discretization procedures (true is assigned above a threshold value) are used to derive boolean matrices from a raw expression matrix. Discretization can obviously have a large in uence on the nature of the extracted sets. It is thus essential that, in exploratory contexts, one can study di erent threshold values and proceed with a large number of analysis.

What we would like to do is to compute **all** sets of genes that have the true value in a su cient number (frequency threshold) of biological situations. Extracting frequent sets is one of the most studied data mining techniques since the description of the APRIORI algorithm [1] and tens of algorithms have been published. Nevertheless, the gene expression matrices, obtained through microarrays, raise new di culties, due to their "pathological" dimensions (i.e. few lines and a huge number of columns). This is a very di cult problem since the overall complexity is exponential in the number of genes. Furthermore the size of the solutions (i.e. collection of extracted sets) is huge whatever the frequency threshold since there is a very limited number of lines.

In Section 2, we present the problems raised by the extraction of frequent sets. Section 3 proposes a solution that combines the power of the closed set extraction with an original use of the properties of the Galois connection [17; 9]. In Section 4 we provide experimental results on two matrices built from microarray data [2; 16]. It establishes the spectacular gains allowed by our approach. Section 5 concludes.

2. FREQUENT SET EXTRACTION

2.1 De£nitions

Let $\mathcal{S}$ denote a set of biological situations and $\mathcal{A}$ denote a set of attributes. In the example from Table 1, $\mathcal{S} = \{s_1, \ldots s_5\}$ and $\mathcal{A} = \{a_1, \ldots a_{10}\}$. Each attribute denotes a property about the expression of a gene. The encoded expression data is represented by the matrix of the binary relation $R \subset \mathcal{S} \quad \mathcal{A}$ de ned for each situation and each attribute. $(s_i, a_j) \in R$ denotes that situation i has the property j, i.e., that gene j is over-expressed or under-expressed in situation i. A database r to be mined is thus a 3-tuple $(\mathcal{S}, \mathcal{A}, R)$. $\mathcal{L}_\mathcal{A} = 2^\mathcal{A}$ is the power set of attributes. For the sake of clarity, sets of attributes are often called sets of genes. $\mathcal{L}_\mathcal{S} = 2^\mathcal{S}$ is the power set of situations.

De nition 1. Given $T \quad \mathcal{S}$ and $X \quad \mathcal{A}$, let $f(T) = \{a \in \mathcal{A} \mid \forall s \in T, (s,a) \in R\}$ and $g(X) = \{s \in \mathcal{S} \mid \forall a \in X, (s,a) \in R\}$. f provides the set of over-expressed or under-expressed genes that are common to a set of situations and g provides the set of situations that share a given set of attributes (expression properties). (f, g) is the so-called Galois connection between $\mathcal{S}$ and $\mathcal{A}$. We use the classical notations $h = f \circ g$ and $h' = g \circ f$ to denote the Galois closure operators.

	Attributes									
Situations	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
s_1	1	1	1	1	0	1	1	0	0	0
s_2	1	1	1	1	0	0	0	0	1	1
s_3	1	1	1	1	0	0	0	0	1	1
s_4	0	0	0	0	1	1	1	1	1	1
s_5	1	0	1	0	1	1	1	1	0	0

Table 1: Example of a boolean matrix $\mathbf{r}_1$

De nition 2. A set of genes X $\mathcal{A}$ is closed i $h(X) = X$. We say that X satis es the $\mathcal{C}_{Close}$ constraint in $\mathbf{r}$: $\mathcal{C}_{Close}(X, \mathbf{r}) \equiv h(X) = X$. A set of situations T $\mathcal{S}$ is closed i $h'(T) = T$.

De nition 3. The *frequency* of a set of genes X $\mathcal{A}$ denoted $\mathcal{F}(X, \mathbf{r})$ is the size of $g(X)$. Constraint $\mathcal{C}_{\text{freq}}$ enforces a minimal frequency: $\mathcal{C}_{\text{freq}}(X, \mathbf{r}) \equiv \mathcal{F}(X, \mathbf{r})$ where is the user-de ned frequency threshold.

Example 1. Given $\mathbf{r}_1$, we have $\mathcal{F}(\{a_1, a_3, a_5\}) = 1$ and $\mathcal{F}(\{a_1, a_2\}) = 3$. If $= 3$, $\{a_9, a_{10}\}$ and $\{a_1, a_2, a_3, a_4\}$ satisfy $\mathcal{C}_{\text{freq}}$ in $\mathbf{r}_1$ but $\{a_1, a_5\}$ does not. $h(\{a_1, a_2\})$ in $\mathbf{r}_1$ is $f(g(\{a_1, a_2\})) = f(\{s_1, s_2, s_3\}) = \{a_1, a_2, a_3, a_4\}$. $\{a_1, a_2\}$ does not satisfy $\mathcal{C}_{Close}$ in $\mathbf{r}_1$ but $\{a_1, a_2, a_3, a_4\}$ satis es it.

Mining task. We want to compute the collection of the frequent sets of genes $FS = \{\varphi \in \mathcal{L}_{\mathcal{A}} \mid \mathcal{C}_{\text{freq}}(\varphi, \mathbf{r}) \text{ satis ed}\}$ where $\mathcal{C}_{\text{freq}}$ is the minimal frequency constraint and $\mathbf{r}$ is a boolean expression matrix. Furthermore, we need the frequencies of each frequent itemset to, e.g., derive interesting association rules from them.
The closure of a set of genes X, $h(X)$, is the maximal (w.r.t. set inclusion) superset of X which has the same frequency than X. A closed set of genes is thus a maximal set of genes whose expression properties (true values) are shared by a set of situations. E.g., the closed set $\{a_1, a_3\}$ in the data of Table 1, is the largest set of genes that are over-expressed (or under-expressed) simultaneously in situations s_1, s_2, s_3 and s_5.
The concept of free set has been introduced in [7] as a special case of the -free sets and has been proposed independently in [3] under the name of key pattern. This concept characterizes the closed set generators [13] but is also useful for non redundant association rule computation (see, e.g., [5] for an illustration).

De nition 4. A set of genes X $\mathcal{A}$ is free i X is not included in the closure (i.e., $h = f \circ g$) of one of its strict subsets. We say that X satis es the $\mathcal{C}_{\text{free}}$ constraint in $\mathbf{r}$. An alternative de nition is that X is free in $\mathbf{r}$ i the frequency of X in $\mathbf{r}$ is strictly lower than the frequency of every strict subset of X.

Example 2. $\{a_1, a_6\}$ satis es $\mathcal{C}_{\text{free}}$ in $\mathbf{r}_1$ but $\{a_1, a_2, a_3\}$ does not.

It is easy to adapt these de nitions to sets of situations. An important result is that the closures of the free sets are closed sets. The size of the collection of the free sets is, by construction, greater or equal to the size of the collection of the closed sets (see, e.g., [8]).
It is well known that $\mathcal{L}_{\mathcal{A}}$ can be represented by a lattice ordered by set inclusion. On top of the lattice, we have the empty set, then the singletons, the pairs, etc. The last level for our example from Table 1 contains the unique set of size 10. A classical framework [11; 10] for an e cient exploration of such a search space is based on the monotonicity of the used constraints w.r.t. the specialization relation, i.e., set inclusion.

De nition 5. A constraint $\mathcal{C}$ on sets is said anti-monotonic when $\forall X, X'$: $(X'$ $X \wedge X$ satis es $\mathcal{C}) \Rightarrow X'$ satis es $\mathcal{C}$. A constraint $\mathcal{C}$ is said monotonic when $\forall X, X'$: $(X$ $X' \wedge X$ satis es $\mathcal{C}) \Rightarrow X'$ satis es $\mathcal{C}$.

Example 3. $\mathcal{C}_{\text{freq}}$, $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{\text{freq}} \wedge \mathcal{C}_{\text{free}}$ are anti-monotonic. $\mathcal{C}_{size}(X) \equiv |X| > 3$ is monotonic.

The negation of a monotonic (resp. anti-monotonic) constraint is an anti-monotonic (resp. monotonic) constraint. Anti-monotonic constraints can be pushed e ciently into the extraction process: when a set X does not satisfy an anti-monotonic constraint, we can prune large parts of the lattice since no superset of X can satisfy it. For instance, the Apriori algorithm [1] computes all the frequent sets by a levelwise search on the lattice, starting from the most general sentences (the singletons) until it reaches the most speci c sentences that are frequent (the maximal frequent sets w.r.t. set inclusion). Apriori and its variants work well on very large boolean matrices (millions of lines, hundreds or thousands of columns) that are not dense and for lowly correlated data. Notice that such algorithms have to count the frequency of at least every frequent set.

2.2 Extraction tractability

The computation of sets that satisfy a given constraint $\mathcal{C}$ is a very hard problem. Indeed, as soon as we have more than a few tens of columns, only a quite small subset of the search space can be explored. Then, the size of the solution, i.e., the collection of the sets that satisfy $\mathcal{C}$ can be so huge that none algorithm can compute them all. When the used constraint is $\mathcal{C}_{\text{freq}}$, it is possible to take a greater frequency threshold to decrease a priori the size of the solution and thus provide the whole collection of the frequent sets. The used threshold can however be disappointing for the biologist: extracted patterns are so frequent that they are already known.
In the expression matrices we have to analyze, the number of the frequent sets can be huge, whatever is the frequency threshold. It comes from the rather low number of lines and thus the small number of possible frequencies. Clearly, Apriori and its variants can not be used here. Since we need for the frequencies of every frequent set, e.g., for deriving valid association rules, algorithms that compute only the maximal frequent itemsets, e.g., [4] do not solve the problem. We decided to investigate the use of the so-called *condensed representations* of the frequent sets by the frequent closed

sets, i.e., $CFS = \{\varphi \in \mathcal{L}_\mathcal{A} \mid \mathcal{C}_{\text{freq}}(\varphi, \mathbf{r}) \wedge \mathcal{C}_{Close}(\varphi, \mathbf{r})$ satis ed$\}$ because FS can be e ciently derived from CFS [13; 6]. CFS is a compact representation of the information about every frequent set and its frequency. Furthermore, several recent algorithms can compute e ciently the frequent closed sets [13; 7; 8; 14; 18; 3]. To be e cient, these algorithms can not use the properties of $\mathcal{C}_{Close}$ which is neither anti-monotonic nor monotonic. However, we can compute the frequent free sets and provide their closures, i.e., $\{h(\varphi) \in \mathcal{L}_\mathcal{A} \mid \mathcal{C}_{\text{freq}}(\varphi, \mathbf{r}) \wedge \mathcal{C}_{\text{free}}(\varphi, \mathbf{r})$ satis ed$\}$ [7; 8; 3]. The lattice is still explored levelwise. At level k, the data is accessed to compute the frequency and the closure of each candidate set. The infrequent sets can be pruned. Thanks to pruning at level k-1, the frequent sets are free sets. Candidates for the next level can be generated from two free sets (using an Apriori-like generation procedure [1]) and candidates for which at least one subset is not frequent ($\mathcal{C}_{\text{freq}}$ is violated) or that are included in the closure of one their subsets (i.e., $\mathcal{C}_{\text{free}}$ is violated) are pruned before the next iteration can start. At the end, we compute $h(X)$ for each frequent free set that has been extracted. It turns out that the anti-monotonicity of a constraint like $\mathcal{C}_{\text{freq}} \wedge \mathcal{C}_{\text{free}}$ is used in two phases. First (Criterion 1), we avoid the computation of supersets that do not satisfy the constraint thanks to the Apriori-like generation procedure. Next (Criterion 2), we prune the sets for which some subsets do not satisfy the constraint. The number of pruned candidates in the second phase, i.e., failures for Criterion 2, can be huge for matrices with a number of lines that is small w.r.t. the number of columns and it can lead to intractable extractions. In other terms, even though these approaches have given excellent results on large matrices for transactional data (e.g., correlated and dense data in WWW usage mining applications), they can fail on expression matrices because of their "pathological" dimensions. Furthermore, we want to enable the use of various discretization procedures and thus the analysis of more or less dense matrices. It appears crucial to us that we can achieve a breakthrough w.r.t. extraction intractability and it has lead to the following original method.

3. A NEW METHOD

We have considered the extraction from a transposed matrix using the Galois connection to infer the results that would have been extracted from the initial matrix. Indeed, one can associate to the lattice on genes the lattice on situations. Elements from these lattices are linked by the Galois operators. The Galois connection gives rise to concepts [17] that associate sets of genes with sets of situations, or in the transposed matrix, sets of situations with sets of genes. When we have only few situations and many genes, the transposition enables to reduce the complexity of the search.

De nition 6. If $X \in \mathcal{L}_\mathcal{A}$ and $T \in \mathcal{L}_\mathcal{S}$, we consider the so-called concepts (X, T) where $T = g(X)$ and $X = f(T)$. By construction, concepts are built on closed sets and, each closed set of genes (resp. situations) is linked to a closed set of situations (resp. genes).

De nition 7. The concept theory w.r.t. $\mathbf{r}$, $\mathcal{L} = \mathcal{L}_\mathcal{A} \quad \mathcal{L}_\mathcal{S}$, and a constraint $\mathcal{C}$ is denoted $Th_c(\mathcal{L}, \mathbf{r}, \mathcal{C})$. It is the collection of concepts (X, T) such that $X \in \{\varphi \in \mathcal{L}_\mathcal{A} \mid \mathcal{C}(\varphi, \mathbf{r})$ satis ed$\}$.

On Figure 1, we provide the so-called Galois lattice for the concepts in the data from Table 1. The specialization relation on the sets of genes which is oriented from the top towards the bottom of the lattice is now associated to a specialization relation on sets of situations which is oriented in the reverse direction. Indeed, if $X \subset Y$ then $g(X) \quad g(Y)$. The collection of the maximally speci c sets of genes (e.g., the maximal frequent itemsets) has been called the positive border in [10]. A dual concept is the one of negative border, i.e., the minimally general sets (e.g., the smallest infrequent sets whose every subset is frequent). The lattice is thus split in two parts. On the top, we have the solution which is bordered by the positive border. On the bottom, we have the sets that do not belong to the solution. The minimal elements of this part constitute the negative border. This duality is interesting: borders are related to a specialisation relation and an anti-monotonic constraint. The bottom part of the lattice can be considered as the solution for the negated constraint, the former positive border becomes the negative border and vice versa.

On the Galois lattice, it is possible to perform two types of extraction: one on the gene space (1), starting from the top of the lattice and following the specialisation relation on the genes, and the other one on the biological situations (2), starting from the bottom of the lattice and following the specialisation relation on the situations. We now de ne the matrix transposition for a matrix $\mathbf{r}$, the constraint transposition for a constraint $\mathcal{C}$ and we state the central result of the complementarity of the extractions.

De nition 8. If $\mathbf{r} = (\mathcal{S}, \mathcal{A}, R)$ is an expression matrix, the transposed matrix is ${}^t\mathbf{r} = (\mathcal{A}, \mathcal{S}, {}^tR)$ where $(a, s) \in {}^tR \iff (s, a) \in R$.

Whereas the matrix transposition is quite obvious, it is not the same for the transposition of constraints. In the case of the minimal frequency constraint $\mathcal{C}_{\text{freq}}$, the dual notion of the frequency for the sets of genes is the length of the corresponding sets of situations.

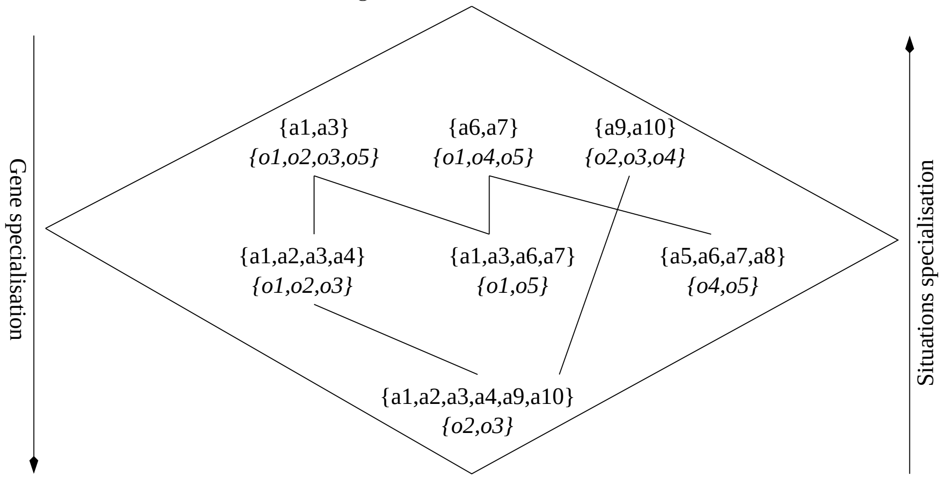
De nition 9. Let $\mathcal{C}$ be a constraint on $\mathcal{L}_\mathcal{A}$, its transposed constraint ${}^t\mathcal{C}$ is de ned on $\mathcal{L}_\mathcal{S}$ by $\forall T \in \mathcal{L}_\mathcal{S}$, ${}^t\mathcal{C}(T, \mathbf{r}) \iff \mathcal{C}(f(T), \mathbf{r})$ where f is the Galois operator. Thus, ${}^t\mathcal{C}_{\text{freq}}(T, \mathbf{r}) \equiv |T| \quad$ if $\quad$ is the frequency threshold for $\mathcal{C}_{\text{freq}}$.

With respect to gene specialization, ${}^t\mathcal{C}$ is monotonic (resp. anti-monotonic) if $\mathcal{C}$ is monotonic (resp. anti-monotonic). However, if $\mathcal{C}$ is anti-monotonic (e.g., $\mathcal{C}_{\text{freq}}$) following the gene specialization relation, ${}^t\mathcal{C}$ is monotonic according to the specialization relation on the situations: it has to be negated to get an anti-monotonic constraint that can be use e ciently.

Property 1. If $\mathcal{C}$ is anti-monotonic w.r.t. gene specialization, then $\neg{}^t\mathcal{C}$ is anti-monotonic w.r.t. situation specialization.

We have an operation for the transposition of the data and a new anti-monotonic constraint w.r.t. to the specialization relation on the situations. However, to obtain this new anti-monotonic constraint, we had to transpose the original constraint and take its negation: the new extraction turns to be complementary to the collection we would get with the standard extraction.

Figure 1: A Galois lattice



De nition 10. Given ${}^t\mathbf{r}$ and $\neg{}^t\mathcal{C}$, the transposed theory $Th_c(\mathcal{L}, {}^t\mathbf{r}, \neg{}^t\mathcal{C})$ is the transposition of $Th_c(\mathcal{L}, \mathbf{r}, \mathcal{C})$.

Property 2. The concept theory $Th_c(\mathcal{L}, \mathbf{r}, \mathcal{C})$ and its transposed theory $Th_c(\mathcal{L}, {}^t\mathbf{r}, \neg{}^t\mathcal{C})$ are complementary w.r.t. the whole collection of concepts.

Example 4. On the data from Table 1, the sets of genes with a frequency of at least 3 are $\{a_1, a_3\}$, $\{a_6, a_7\}$, $\{a_9, a_{10}\}$, and $\{a_1, a_2, a_3, a_4\}$. A closed set of genes has a frequency greater than 3 if the size of the corresponding situation set is greater than 3. When taking the negation of this constraint, we look for the sets of situations whose size are at most 3 (anti-monotonic constraint w.r.t. the situation specialization). The sets $\{s_1, s_5\}$, $\{s_4, s_5\}$ and $\{s_2, s_3\}$ are extracted. Clearly, the two collections are complementary (see Figure 2).

The correctness of this extraction method for nding the closed sets of genes from the extractions on transposed matrices relies on this complementary property. Due to the lack of space, we consider only a straightforward application of this framework that concerns the computation of concepts. If we compute the closed sets from the gene space, the Galois connection allows to infer the closed sets of situations. Reciprocally, the extraction on the transposed matrix provides the closed sets on the situations and we can infer the closed sets of genes. Thus, the same collection of closed sets can be extracted from a matrix or its transposed. The choice between one or the other method can be guided by the dimension of the matrix. On the data from Table 1, the smallest dimension concerns the situations (5 elements) and it leads to $2^5 = 32$ possible sets. Among these 32 elements, only 10 are closed. However extracting the closed sets from the original matrix, which contains 10 columns, leads to a search space of $2^{10} = 1024$ sets of genes whereas there is still 10 closed sets. To compute the closed sets, we output the closures of the free sets. This is an e cient solution since $\mathcal{C}_{\text{free}}$ is anti-monotonic. Several free sets can however generate the same closed set. On the data from Table 1, the free set extraction provides 41 sets which generate the 10 closed sets, whereas the transposed matrix extraction provides only 17 free sets. We provide real examples in the next section.

4. APPLICATIONS

We have been working on data sets produced with cDNA microarrays at the Stanford Genome Technology Center (Paolo Alto, CA 94306, USA). The rst data set is described in [16]. It concerns the study of human insulino-resistance. From 6 cDNA microarrays (around 42 557 spots for 29 308 UniGene clusters), a typical preprocessing for microarray data has given an expression matrix with 6 lines (situations) and 1 065 columns (genes). It is denoted as the `inra` matrix. The second data set concerns gene expression during the development of the drosophila [2]. With the same kind of preprocessing, we got an expression matrix with 162 lines and 1 230 columns denoted `droso`. To derive boolean matrices, we have encoded the over-expression of genes: for each gene i, we have computed a threshold $_i$ under which the attribute boolean value is false, and true otherwise. Different methods can be used for the de nition of threshold $_i$ and we have done as follows [2]: $_i = Max_i \quad (1 \quad \_discr)$ where Max_i is the maximal expression value for gene i and $\_discr$ is a parameter that is common to every genes.

We used the prototype `mv-miner` implemented by F. Rioult and have extracted the closed sets under the frequency threshold 1 to get all of them. These experiments have been performed on a 800MHz processor with RAM 4GB and 3GB for swap (linux operating system). We have used parameter $\_disc$ to study the extraction complexity w.r.t. the density of the boolean matrices (ratio of the number of true values on the number of values).

First, we compare the extraction in `inra` and ^{t}`inra` for a given value of $\_disc$ (Table 2).

We have 41 closed sets in these boolean matrices. The free set extraction on `inra` provides 667 831 free sets of genes whereas the extraction on ^{t}`inra` provides 42 free sets of situations. In Section 2.2, we have seen that algorithms use anti-monotonic constraints in two ways. Criterion 1 avoids to generate some candidates that would have to be pruned. Criterion 2 enables to prune candidates for which the constraint is not satis ed. Checking Criterion 2 is expensive because it needs to store the sets and check the properties of all their subsets. Table 2 provides the number of sets (for each level in the levelwise search) which satisfy these two criteria and the number of sets that have been exam-

Figure 2: Complementarity of the extractions

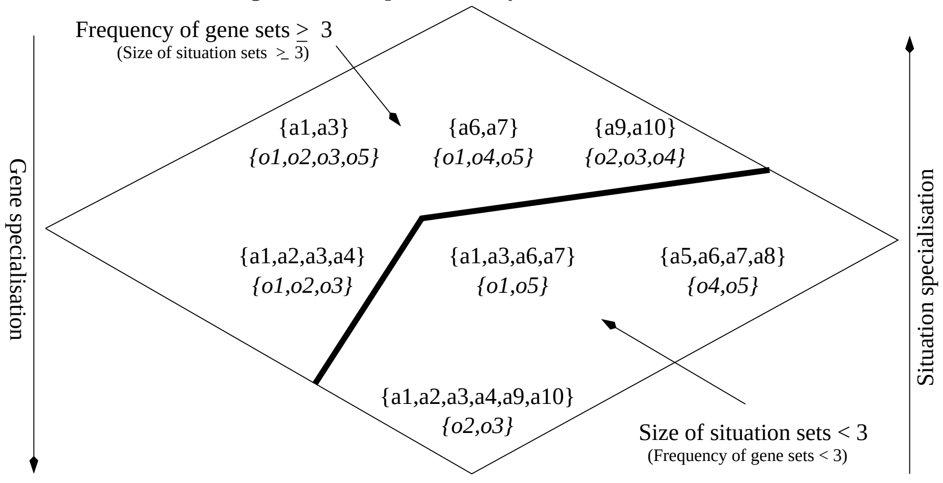


Table 2: Failure/success in pruning for `inra` and ^{t}`inra`

	^{t}`inra`		`inra`	
size	success	failure	success	failure
1	6	0	777	0
2	15	0	172 548	128 928
3	16	4	2 315 383	4 713 114
4	6	9	2 965 726	9 371 325
5	0	2	0	1 544 485
Total	43	15	5 454 434	15 757 852
Nb free sets	42		667 831	
Nb closed sets	41			

ined when processing `inra` and ^{t}`inra`. Extraction in ^{t}`inra` is clearly more e cient. Not only it provides less candidate sets to test (43 vs. 5 454 434) but also it leads to far less failures: 15 vs. 15 757 852.

We have performed experiments on the two microarray data sets for various discretization thresholds. Considering `droso`, Table 3 shows that extraction becomes feasible for larger densities. It enables that the biologist explore alternatives for discretizations.

Results on `inra` con rm the observations (see Table 4). The di erence between standard extraction and transposed matrix extraction is even more spectacular. Indeed, extraction time on the transposed matrix can become negligible w.r.t. the standard extraction time (e.g., 120 ms vs. 368 409 ms). Notice that the number of free sets of genes can be very large w.r.t. the number of closed sets to be found, e.g., 51 881 free sets for only 34 closed sets.

Also, the method has been applied on very large expression matrices derived from human SAGE data (matrix 90 12 636) [15]. In these matrices and for di erent discretization techniques, none of the standard extractions have been tractable while extractions on the transposed matrix have been easy.

5. USING THE CLOSED SETS

An algorithm like `mv-miner` takes a boolean matrix and provides the free sets on the columns and the closed sets on both the lines and the columns with their frequencies in the data. From the closed sets of genes and their frequencies, it is indeed possible to select the frequent closed sets provided a frequency threshold. When using $\mathcal{C}_{\text{freq}}$ with > 1, it is possible to use its transposed constraint and make use of the transposed extractions.

Let us discuss informally how to use the closed sets to derive some knowledge. It is possible to regenerate the whole collection of the frequent sets of genes (resp. situations) from the frequent closed sets of genes (resp. situations). So, the multiples applications of the frequent itemsets are available (e.g., association rule mining, class characterization, some types of clustering, approximation of the joint distribution). Notice also that the extracted collections can be represented as concepts [17] and thus many knowledge discovery tasks based on concept lattices can be considered.

It is quite possible that the number of frequent sets of genes is too huge and that a "blind" regeneration process is not feasible. It is possible to lter at regeneration time, e.g., to take into account some syntactical constraints on the sets of interests. Notice also that the free sets that have been proved useful for non redundant association rule computation are missing. When mining the transposed matrix, we get the free sets on situations but not the free sets on genes.

Let us however sketch typical uses of the extracted patterns. Part of this has been already validated on SAGE data analysis in [5]. It is useful to look at the patterns extracted after several discretizations on the same expression matrix. These extractions can provide di erent sets of co-regulated genes for the same expression data. So, after such computations, the biologist want to compare di erent closed set

Table 3: Results for the drosophila data

	_discr	density	time (ms)	nb free sets	nb closed sets
^{t}droso	0.02	0.08	160	965	434
droso	0.02	0.08	1 622	5 732	434
^{t}droso	0.075	0.015	420	3 667	1 508
droso	0.075	0.015	35 390	60 742	1 508
^{t}droso	0.1	0.019	721	6 890	2 569
droso	0.1	0.019	146 861	162 907	2 569
^{t}droso	0.15	0.032	4 526	36 309	10 447
droso	0.15	0.032	failure	-	-
^{t}droso	0.2	0.047	36 722	410 666	4 6751
droso	0.2	0.047	failure	-	-
^{t}droso	0.25	0.067	455 575	1 330 099	259 938
droso	0.25	0.067	failure	-	-
^{t}droso	0.3	0.09	failure	-	-
droso	0.3	0.09	failure	-	-

collections, looking at the common patterns, the dissimilarities, etc. Browsing these collections of closed sets can lead to the selection of some of them, e.g., the one that are almost always extracted.
Then, the biologists can select some concepts for an in-depth study of the interactions between the involved genes. Clearly, one objective criterion for selection can be based on the frequency. One important method concerns the use of information sources about gene functions. Quite often, one of the rst post-processing is to look for the homogeneity of the sets (e.g., they all share the same function). It is then quite interesting to focus on almost homogeneous sets of genes and look at the outliers. This approach has been used in the SAGE data analysis described in [5] and has provided a valuable result: one EST (Expressed Sequence Tag) was always co-regulated with a set of around 20 genes that had the same function and it is reasonable to suspect that this EST has that function. For this type of post-processing, it is possible to use the various ontologies that are available, e.g., http://www.geneontology.org/, and, e.g., study the homogeneity of the selected sets of genes at different levels (biological process, molecular function, cellular function).
Last but not the least, the biologist can chose a given closed set of genes X and then project the original expression matrix on X. Since the size of a closed set will be generally small w.r.t. the size of whole collection of genes, it is then possible to mine this restricted matrix. For instance, it becomes possible to extract the whole collection of non redundant association rules (free sets of genes in the left-hand side) from this non transposed restricted matrix.

6. CONCLUSION

We have been studying the extraction of groups of genes found to be frequently co-regulated in expression matrices. This type of data raises di cult problems due to the huge size of the search space and to the huge size of the solutions. In [5], it has been shown that the use of condensed representations as described in e.g. [6; 7], was useful, at least when the number of biological situations is not too small in light of the number of genes. Unfortunately this situation is rarely observed in most of the available gene expression data. We therefore explored the possibility to process the transposed matrices by making use of properties of the Galois connections. This resulted in a very spectacular improvement of the extraction procedure, allowing to work in context where previous approaches failed. [5] has validated the interest of frequent closed sets in biological terms on a reduced set of genes. We are pretty con dent that given the algorithmic breakthrough, biological signi cant information will be extracted from the expression data we have to mine. We are furthermore exploring the transposition of other constraints.

Acknowledgements This work has been partially funded by the EU contract cInQ IST-2000-26469 (FET arm of the IST programme). F. Rioult is supported by the IRM department of the Caen CHU, the comite de la Manche de la Ligue contre le Cancer and the Conseil Regional de Basse-Normandie. The authors thank S. Blachon, O. Gandrillon, C. Robardet, and S. Rome for exciting discussions.

8. REFERENCES

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, 1996.

[2] M. Arbeitman, E. Furlong, F. Imam, E. Johnson, B. Null, B. Baker, M. Krasnow, M. P. Scott, R. Davis, and K. P. White. Gene expression during the life cycle of drosophilia melanogaster. *Science*, 297, 2002.

[3] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explorations*, 2(2):66 – 75, Dec. 2000.

[4] R. J. Bayardo. E ciently mining long patterns from databases. In *Proceedings ACM SIGMOD'98*, pages 85–93, Seattle, Washington, USA, 1998. ACM Press.

[5] C. Becquet, S. Blachon, B. Jeudy, J.-F. Boulicaut, and O. Gandrillon. Strong association rule mining for large gene expression data analysis: a case study on human SAGE data. *Genome Biology*, 12, 2002.

Table 4: Results for the human data

	_*discr*	density	time (ms)	nb free sets	nb closed sets
^{t}**inra**	0.01	0.193	80	19	19
inra	0.01	0.193	16 183	11 039	19
^{t}**inra**	0.05	0.21	90	29	29
inra	0.05	0.21	42 991	23 377	29
^{t}**inra**	0.085	0.023	110	33	33
inra	0.085	0.023	83 570	38 659	33
^{t}**inra**	0.1	0.24	120	36	34
inra	0.1	0.24	368 409	51 881	34
^{t}**inra**	0.15	0.268	120	40	38
inra	0.15	0.268	failure	-	-

[6] J.-F. Boulicaut and A. Bykowski. Frequent closures as a concise representation for binary data mining. In *Proceedings PAKDD'00*, volume 1805 of *LNAI*, pages 62–73, Kyoto, JP, Apr. 2000. Springer-Verlag.

[7] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by mean of free-sets. In *Proceedings PKDD'00*, volume 1910 of *LNAI*, pages 75–85, Lyon, F, Sept. 2000. Springer-Verlag.

[8] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery journal*, 7(1):5–22, 2003.

[9] R. Godin, R. Missaoui, and H. Alaoui. Incremental algorithms based on galois (concepts) lattices. *Computational Intelligence*, 11(2):246 – 267, 1995.

[10] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery journal*, 1(3):241–258, 1997.

[11] T. M. Mitchell. Generalization as search. *Arti cial Intelligence*, 18:203–226, 1982.

[12] C. Niehrs and N. Pollet. Synexpression groups in eukaryotes. *Nature*, 402:483–487, 1999.

[13] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Ef- cient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, Jan. 1999.

[14] J. Pei, J. Han, and R. Mao. CLOSET an e cient algorithm for mining frequent closed itemsets. In *Proceedings ACM SIGMOD Workshop DMKD'00*, Dallas, USA, May 2000.

[15] C. Robardet, F. Rioult, S. Blachon, B. Cremilleux, O. Gandrillon, and J.-F. Boulicaut. Mining closed sets of genes from SAGE expression matrices: a spectacular improvement. Technical report, LIRIS INSA Lyon, F-69622 Villeurbanne, March 2003.

[16] S. Rome, K. Clement, R. Rabasa-Lhoret, E. Loizon, C. Poitou, G. S. Barsh, J.-P. Riou, M. Laville, and H. Vidal. Microarray pro ling of human skeletal muscle reveals that insulin regulates 800 genes during an hyperinsulinemic clamp. *Journal of Biological Chemistry*, March 2003. In Press.

[17] R. Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470. Reidel, 1982.

[18] M. J. Zaki. Generating non-redundant association rules. In *Proceedings ACM SIGKDD'00*, pages 34 – 43, Boston, USA, Aug. 2000. AAAI Press.

Weave Amino Acid Sequences for Protein Secondary Structure Prediction

Xiaochun Yang
Department of Computer Science,
Brigham Young University,
Provo, Utah, 84602 U.S.A.
xiaochuny@cs.byu.edu

Bin Wang
Institute of Software,
Northeastern University,
Shenyang, 110004 China.P.R.
wang_b2001@hotmail.com

ABSTRACT

Given a known protein sequence, predicting its secondary structure can help understand its three-dimensional (tertiary) structure, i.e., the folding. In this paper, we present an approach for predicting protein secondary structures. Different from the existing prediction methods, our approach proposes an encoding schema that weaves physio-chemical information in encoded vectors and a prediction framework that combines the context information with secondary structure segments. We employed Support Vector Machine (SVM) for training the CB513 and RS126 data sets, which are collections of protein secondary structure sequences, through sevenfold cross validation to uncover the structural differences of protein secondary structures. Hereafter, we apply the sliding window technique to test a set of protein sequences based on the group classification learned from the training set. Our approach achieves 77.8% segment overlap accuracy (SOV) and 75.2% three-state overall per-residue accuracy (Q_3), which outperform other prediction methods.

General Terms

Design, Experimentation

Keywords

Protein structure prediction, protein secondary structure, SVM, encoding schema

1. INTRODUCTION

Understanding the function and the physiological role of proteins is a fundamental requirement for the discovery of novel medicines and protein-based products with medical and industrial applications. To enhance their understanding, many researchers focus on the functional analysis of genome these days. This functional analysis is a difficult problem, and the determination of protein three-dimensional structures is essential in the analysis of the genome functions. A protein three-dimensional structure can help understand its function. Unfortunately, it happens that three-dimensional protein structures cannot be accurately predicted from protein (amino acid) sequences. An intermediate, but useful and fundamental, alternative step is to predict protein secondary structures since a protein three-dimensional structure results partially from its secondary structure.

The protein secondary structures can be categorized into several states: a-helix (H), 3_{10} helix (G), pi helix (I), isolated β-bridge (B), extended β-strand (E), hydrogen bonded turn (T), bend (S) and random coil (-) [1]. In the previous work on predicting protein secondary structures, researchers focused on classifying these states into three consolidated classes: helix, strand and coil classes, among which the helix and strand classes are the major repetitive secondary structures [11]. These days the number of known protein sequences has far exceeded the number of known protein secondary structures, which have been obtained through chemical and biological methods such as the crystallography and NMR methods [14], and the rapid pace of the discovery of genomic sequences has further widened the gap. Thus, computational tools for predicting protein secondary structures from protein sequences are needed and they are essential in narrowing the widening gap [14]. After four decades of research work, the prediction of the secondary structures has not yet attained the satisfying accuracy rate. (See Table 1.)

The secondary structure prediction methods developed in the past [8; 17; 19] use local information of a single sequence, but they exhibit major drawbacks. For example, the accuracy measures (Q3) on these methods are relatively low (57% for the 1st generation and 63% for the 2nd generation). A usual criticism on using the neural network prediction approach [17] is that the methods are "black boxes." The neural network approach may be an effective classifier; however, it is hard to explain why a given pattern is classified as a helix rather than a strand, and thus it is difficult to derive a conclusion on the merit of the approach. Recently, a more promising approach on solving the predicting problem appears in [7]. Hua et al. introduce a prediction approach based on the SVM [18], which separated the input data, i.e., the protein secondary structure segments, into two classes to uncover the protein secondary structural classes.

However, most of the existing works did not consider the roles of physio-chemical properties of each amino acid residue, and to our best knowledge, none of them consider the roles of contexts of helices, strands, or coils in protein sequences which results in the lower accuracy prediction. Our prediction approach builds up an *encoding schema* that weaves the properties of different chemical groups into secondary structure segments and a *two dimensional prediction framework* that considers not only interested segments themselves (e.g. helices and strands) but also the contexts of those interested

Table 1: Prediction of protein secondary structures during the three generations

	1st Generation	2nd Generation	3rd Generation
Time Period	1960s - 1970s	1980s	1990s
Standard Method	GOR1 [8]	GOR3 [13; 19]	PHD [17]
Approach	Statistical Information	Local Interactions Among Amino Acids	Homologous Protein Sequences
Prediction Accuracy (Q3)	57%	63%	72%

segments. Encoding schema includes: (i) n-gram for determining the length of training segments, (ii) cardinal vectors for catching similar physio-chemical properties of amino acid residues, and (iii) probability of different residues for capture the tendency of a residue location has a conformation of helical (strand or coil) structure, while two dimensional prediction framework uses sliding window technique to exhaustively exam all the possible protein sequence segments and synthesizes classifiers derived from the helical (strand or coil) segments and context segments to the final prediction. We use two data sets CB513 [3] and RS126 [16], which contain 513 and 126 sequences of protein secondary structures, respectively, that have been discovered by using different chemical and biological experiments and employ SVM to train samples (segments) in the data sets. Hereafter, we use the two dimensional prediction framework to compare each predicted segments with known protein classes. Our prediction approach gets encouraging experimental results by adopting various accuracy measures, and it has some advantages over other prediction methods, which are listed as follows.

- It weaves part of biological information, such as chemical groups of each residue, and frequency of residues into the encoding schema which combines the classical encoding schema, matrix calculation, and statistical method, where we use classical encoding schema to represent the order of residues, matrix calculation to combine biological information and order of segments together, and statistical method to capture the frequency of each residue in the training set.
- Our two dimensional prediction is correlated, i.e. each prediction is not made in isolation, taking account of the predictions for neighboring residues. It allows protein classification based on not only the local regions of similarity but also the context regions of similarity in contrast to the detection of global similarities used in the traditional search techniques. Each prediction is based on multiple decision functions that get rid of the inherent bias/noise.
- The approach is theoretically sound since the SVM method has been adopted by researchers in mathematic and computer science since its discovery in 1997 in solving the classification problems, and it has been proved to be efficient in the determination of the decision function in separating two different (in our work, the protein) classes, which is exactly what we need in solving the prediction of protein secondary structures.

The rest of the paper is organized as follows. In Section 2, we simply introduce SVM, in Section 3, we propose an encoding schema that transforms amino acid segments into vectors for further training, and in Section 4, we present a two dimensional prediction framework, In Section 5, we discuss our experimental results, and finally we give our conclusions.

2. REVIEW OF SUPPORT VECTOR MACHINE

SVM [18] has been successfully applied in solving a wide range of pattern recognition problems, including face detection [12], text classification [4], and etc. It is popular due to effective avoidance of overfitting, the ability to handle large feature spaces, information condensing of data sets, and promising empirical performance.

The basic idea of SVM is that for a training set which contains l samples, where each sample is represented by an n-dimensional input vector $\vec{\mathbf{x}}_i$ $(1 \le i \le l)$ and the training set is regarded as an n-dimensional input space $\Re^n$, SVM tries to estimate a *decision function* f: $\Re^n \rightarrow \{\pm 1\}$, i.e. for each n-dimensional input vectors $\vec{\mathbf{x}}_i$ $(i = 1, 2, \ldots, l) \in \Re^n$, there is a classification identifier $y_i \in \{+1, -1\}$ that indicates to which class, $+1$ or -1, $\vec{\mathbf{x}}_i$ belongs. The decision function is shown as follows,

$$f(\vec{\mathbf{x}}) = sign(\sum_{i=1}^{l} y_i \alpha_i \cdot \vec{\mathbf{x}} \cdot \vec{\mathbf{x}}_i + b) \qquad (1)$$

where l is the number of vectors $\vec{\mathbf{x}}_i$ $(1 \le i \le l)$ in the training set, vector $\vec{\mathbf{x}}$ is a variant that represents a testing vector in a test set, y_i $(\in \{+1, -1\})$ is a classification identifier, and coefficients α_i are obtained by solving the following convex Quadratic Programming (QP) problem:
Maximize

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{i=1}^{l} \alpha_i \alpha_j \cdot y_i y_j \cdot (\vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j) \qquad (2)$$

with constraints

$$\sum_{i=1}^{l} \alpha_i y_i = 0, \;\; 0 \le \alpha_i, \;\; i = 1, 2, \ldots, l. \qquad (3)$$

Vector $\vec{\mathbf{x}}$ can be classified by the decision function in the testing phase by calculating distances (dot products) between $\vec{\mathbf{x}}$ and $\vec{\mathbf{x}}_i$. (Note that the decision function depends only on the dot products of vectors in $\Re^n$.)

In the case where a linear boundary is inappropriate for classifying samples in the training set, a straightforward method can be used to map vectors in $\Re^n$ into a high dimensional *feature space* $\mathcal{H}$ via a nonlinear map Φ: $\Re^n \mapsto \mathcal{H}$. Therefore, SVM maps the input vector, $\vec{\mathbf{x}}_i \in \Re^n$, into a *feature vector*, $\Phi(\vec{\mathbf{x}}_i) \in \mathcal{H}$. Then the training problem only depends on the dot products of vectors, $\Phi(\vec{\mathbf{x}}) \cdot \Phi(\vec{\mathbf{x}}_i)$, in $\mathcal{H}$. However, if $\mathcal{H}$ is a high-dimensional space, $\Phi(\vec{\mathbf{x}}) \cdot \Phi(\vec{\mathbf{x}}_i)$ will be very expensive to compute [12]. Fortunately, the dot product in $\mathcal{H}$ has an equivalent expression $K(\vec{\mathbf{x}}, \vec{\mathbf{x}}_i)$ in $\Re^n$ such that $K(\vec{\mathbf{x}}, \vec{\mathbf{x}}_i) = \Phi(\vec{\mathbf{x}}) \cdot \Phi(\vec{\mathbf{x}}_i)$, where $K(\vec{\mathbf{x}}, \vec{\mathbf{x}}_i)$ is a kernel function. Therefore, the kernel function is used to perform $\Phi(\vec{\mathbf{x}})\cdot\ \Phi(\vec{\mathbf{x}}_i)$ in the input space $\Re^n$ rather than the potentially high dimensional feature space $\mathcal{H}$, i.e. using kernel function in $\Re^n$ without need to explicitly know what Φ is. If the kernel function contains a bias term[1], b can be accommodated within the kernel function, and hence the decision function in Equation 1 can be rewritten into Equation 4.

$$f(\vec{\mathbf{x}}) = sign(\sum_{i=1}^{l} y_i \alpha_i \cdot K(\vec{\mathbf{x}}, \vec{\mathbf{x}}_i)) \qquad (4)$$

where coefficients α_i satisfy the Equation 5 with the constraints in Equation 6.
Maximize

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{i=1}^{l} \alpha_i \alpha_j \cdot y_i y_j \cdot K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) \qquad (5)$$

with constraints

$$\sum_{i=1}^{l} \alpha_i y_i = 0,\ 0 \le \alpha_i \le C,\ i = 1, 2, \ldots, l. \qquad (6)$$

where, C is a unper bound value. A large C assigns a high penalty to the classification errors[12].
The most commonly accepted employed kernel functions are shown in Table 2. For a given training set, only the kernel function and the parameter C are needed to be selected to specify one SVM.

3. ENCODING SCHEMA

Each known protein sequence is composed of a number of structured segments. We first extract n-gram segments from known protein sequences with consideration of structured segments and the contexts of them. Then we represent each extracted segments as an input vector for the further training and prediction.

3.1 N-gram Extraction

It has been well accepted that the order in an amino acid sequence plays the most important role in predicting its secondary structure. Having looked at a lot of sequences, we know which segment tends to construct a secondary structure and which does not. For this task, we cannot possibly consider each residue separately. Therefore, we use n-gram model (any segments with n amino acid residues, where $n > 0$) to extract segments from amino acid sequences and transform them into input vectors. The total number of possible segments of an amino acid sequence using n-gram extraction is $m-n+1$, where m (>0) is the number of residues in the sequence. The choice of n is determined by the average length of different structure segments (helix or strand). Based on our study, for most helices the amino acid length is between five and 17, and for most strands the length is between five and 11. Thus, we will choose varied n between five and 17 and begin with a small size and gradually increase the size to determine the "best" size for future helix/non-helix (or strand/non-strand) prediction. (It is anticipated that the best sizes, n, of different classifications should achieve the best prediction result for their classification.)

[1]Many employed Kernels have a bias term and any finite kernel function can be made to have one[12].

3.1.1 Subsequences without context

In the case that extracting segments from a training set without considering context, we only consider structure segments (i.e., helix, strand, or coil) in known protein sequences, which includes identifying structure segments in protein sequences from the training set and extracting n-gram subsequences $m-n+1$ times from each structure segment sequentially, where m is the number of residues in the segment.

Example 1. Given a protein sequence "`MNIFEMLRID`" with a helix segment from position 2 to 7 (underlined), the four 3-gram subsequences are "`NIF`," "`IFE`," "`FEM`," and "`EML`."

3.1.2 Subsequences with context

Locally encoding structure segments has its inherent disadvantage, for example, the contexts of structure segments has been excluded. Consider the protein sequence "`MNIFEML RID`" again. Residue '`M`' and subsequence "`RID`" are contexts of helix segment "`NIFEML`" which would lead us to infer that the nearest neighbors (one or more residues) of "`NIFEML`" must have significant different properties from "`NIFEML`," otherwise these neighbors should be included in "`NIFEML`" as a partial helical structure.

Definition 1. N-gram context subsequences of structure segments. Given a protein sequence s and a secondary structure segment s_s of size m that begins at position i in s, the *left n-gram context* of s_s is a subsequence of size n that begins at position i - $\lfloor \frac{n}{2} \rfloor$ in s, and the *right n-gram context* of s_s is a subsequence of size n that begins at position $i + m$ - $\lceil \frac{n}{2} \rceil$ in s.

Example 2. Given a protein sequence "`DQAMRYKT`," the underlined residues are helical structures. The left 5-gram context subsequence of "`QAM`" is "$\emptyset$`DQAM`," where "$\emptyset$`D`" are the left two neighbors, where $\emptyset$ indicates that the followed "`D`" is the head of the sequence. The right 5-gram context string of it is "`QAMRY`," where "`RY`" are the right two neighbors.

3.2 Encoding Schema of Amino Acid Residues

The transformation from n-gram subsequences into input vectors is an *encoding* process. The classical encoding schema was adopted along with using neural network in predicting protein secondary structures [13], where in a segment, each residue is encoded as an orthogonal binary vector $[1, 0, \ldots, 0]$, $\ldots$, or $[0, 0, \ldots, 1]$, in which '1' indicates the relative position of the corresponding residue. This type of vectors is 21-dimensional, and each one of the first twenty coefficients of these vectors denotes a distinct amino acid in the segment and the last unit is used for specifying whether the corresponding residue is the "head" or the "tail" of the sequence [13]. However, even if chemical and biological methods, such as crystallography and NMR, are used, the "head" and the "tail" of a segment can not be accurately determined [14]. Therefore, we use the first 20 units of each 21-dimensional vector to represent a residue, which we call 20-dimensional *residue vector*. In this paper, we adopt the following ordered list of residues in [11] to encode a residue at the relative position in its corresponding sequence.

Table 2: The commonly accepted kernel functions

Typical Kernel function	Explanation
$K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = ((\vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j) + \theta)^d$	Polynomial function
$K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = \exp(-\gamma\lvert\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j\rvert^2)$	Gaussian radial basis function
$K(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j) = \tanh(\vec{\mathbf{x}}_i \cdot \vec{\mathbf{x}}_j - \theta)$	Multi layer perceptron function

Table 3: Five groups according to chemical properties of amino acid residues.

R-group classification	5D cardinal vectors	Amino acid residues
Nonpolar, aliphatic (c_1)	[1, 0, 0, 0, 0]	`A,V,L,I,M`
Aromatic (c_2)	[0, 1, 0, 0, 0]	`F,Y,W`
Polar, uncharged (c_3)	[0, 0, 1, 0, 0]	`G,S,P,T,C,N,Q`
Positively charged (c_4)	[0, 0, 0, 1, 0]	`K,H,R`
Negatively charged (c_5)	[0, 0, 0, 0, 1]	`D,E`

A, V, L, I, M, F, Y, W, G, S, P, T, C, N, Q, K, H, R, D, E

Example 3. Given a training set of protein sequences, the 20-dimensional residue vector of 'V'' is $[0,1,0,\ldots,0]_{1\times 20}$, where the second coefficient of the vector is 1, since 'V' is the 2nd residue in the ordered list given above.

In our prediction approach, instead of using 20-dimensional residue vector to represent a residue, we incorporate statistical information and the physio-chemical features of each residue into our encoding schema to enhance the classical approach by considering the probability of each residue and different biological groups [11] to which a residue belongs. In Table 3, twenty different amino acid residues are classified into five different *R-groups*, which combines Hydropathy index, molecular weights M_r, and pI value together [11]. Besides this R-group classification, amino acid residues of secondary structure sequences can be classified into many different groups according to their biological meanings, such as exchange group[5], hydrophobicity group[11], and etc (see appendix). [11] indicates that if two amino acids have similar biological properties, they should form similar structures. Therefore, we define *cardinal vector* to incorporate the different biological groups information of amino acid residues into a secondary structure sequence.

Definition 2. Cardinal vector. Given an m-class biological group of amino acid residues, its m-dimensional (mD) cardinal vector is a binary vector $\hat{vc}_m = [a_{c_1}, a_{c_2}, \ldots, a_{c_m}]$, where a_{c_i} $(1 \le i \le m) \in [0,1]$ and $\sum_{i=1}^{m} a_{c_i}^2 = 1$.

Definition 3. Probability of an m-class group with helix (strand or coil) structure in a training set. Given a training set s and an m-class group C=$\{c_1, \ldots, c_m\}$ of amino acid residues, the probability of a residue i in c_j (1$\le i \le m$) has helical structure is:

$$p^H_{i,c_j} = \frac{1}{N_H} \sum_{i \in c_j} N_{H_i} \tag{7}$$

where p^H_{i,c_j} $(1 \le i \le 20)$ is the probability that residue i belongs to group c_j and is a helix in s, N_H is the total number of residues that have the conformations of helical structures in the training set, and N_{H_i} is the number of residues in which residue i belongs to c_j and residue i is a helix in s. (p^E_{i,c_j} for strand and p^C_{i,c_j} for coil structures are computed accordingly.)

We use Kronecker product[20] of residue vectors, cardinal vectors, and frequency of residues in the training set to encode a residue, and we use concatenation of vectors of n consecutive residues to encode an amino acid subsequence with n residues. Residue vectors distinguish a residue from others, cardinal vectors catch the similar physio-chemical properties of residues, the frequency of a residue captures the tendency of the residue that locates in which structure, and concatenation of vectors exhibits the order of residues in a protein subsequence.

Definition 4. Encoding schema of protein segments. Given a training set s and an m-class biological group C=$\{c_1, \ldots, c_m\}$, the encoded vector of a residue i w.r.t. a c_j (1$\le i \le m$) is

$$\begin{aligned} v_{i,c_j} &= p^H_{i,c_j} \times v_{r_i} \otimes \hat{vc}_{m_{c_j}} \\ &= p^H_{i,c_j} \times [a_1 \times \hat{vc}_{m_{c_j}}, \ldots, \ a_{20} \times \hat{vc}_{m_{c_j}}] \\ &= [p^H_{i,c_j} \times a_1 \times a_{c_1}, \ldots, p^H_{i,c_j} \times a_{20} \times a_{c_1}, \ldots, \\ &\quad\; p^H_{i,c_j} \times a_1 \times a_{c_m}, \ldots, p^H_{i,c_j} \times a_{20} \times a_{c_m}] \end{aligned} \tag{8}$$

where v_{r_i} (= $[a_1, a_2, \ldots, a_{20}]$) is the residue vector of residue i, $\hat{vc}_{m_{c_j}}$ (= $[a_{c_1}, a_{c_2}, \ldots, a_{c_m}]$) is the cardinal vector of the m-class group to which i belongs, and p^H_{i,c_j} is the probability that residue i is a helix in s and is belonged to group c_j. For each subsequence s_s in s, the encoded schema of s_s is the concatenation of vectors of the consecutive residues in s_s. Hence, the size of an encoded vector of a residue is 20 $\times$ m, and the size of an encoded vector of subsequence with n residues is 20 $\times$ m $\times$ n.

Example 4. Consider a training set that contains the following two known protein sequences and their DSSP structures,[2]

$$\begin{aligned} s_1 &: \underbrace{\texttt{LWQFNGMIKCK}}_{H}\ \underbrace{\texttt{PS}}_{T(Coil)}\ \underbrace{\texttt{PLLD}}_{H}\ \underbrace{\texttt{FNN}}_{S(Coil)}\ \underbrace{\texttt{GCY}}_{T(Coil)} \\ s_2 &: \underbrace{\texttt{NLLLSDW}}_{E}\ \underbrace{\texttt{W}}_{\_(Coil)}\ \underbrace{\texttt{HQ}}_{S(Coil)}\ \underbrace{\texttt{S}}_{\_(Coil)}\ \underbrace{\texttt{IHKQEVGLS}}_{H} \end{aligned}$$

where H is α-helix, E is extended β-strand, T is hydrogen bonded turn, S is bend, and _ is random coil. Hence,

Helix (H)	: `LWQFNGMIKCK`, `PLLD` and `IHKQEVGLS`
Strand (E)	: `NLLLSDW`
Coil (S, T, _)	: `PS`, `FNNGCY`, and `WHQS`

[2]DSSP (Dictionary of Secondary Structure Protein) is available at http://www.sander.ebi.ac.uk/dssp.

Let's consider the first helix segment, i.e. "LWQFNGMIKCK," extracted from s_1 and the first coil segment, i.e., "PS," also extracted from s_1. If we use 3-gram and R-group, i.e., 5-group (see Table 3) to train (determine) a helix/non-helix classifier, "LWQ" is represented as a 300-dimensional (20 $\times$ 5 $\times$ 3) vector V_1, in which the first 100 units of V_1 (see below) encodes the amino acid residue 'L.' Since 'L' belongs to the *Nonpolar, aliphatic R group* (c_1), the probability that 'L' is a helix and 'L' belongs to c_1 (in the training set) is 33% since $p^H_{L,c_1} = \frac{8}{24} = 0.33$, where 24 is the number of helix, and 8 is the sum of all the residues that belong to c_1 that have the conformation of helical structure in the training set.

$$V_1 = [\, \underbrace{\underbrace{0,\ldots}_{10},\underbrace{0.33}_{p^H_{L,c_1}},\underbrace{0,\ldots}_{89}}_{L},\underbrace{\underbrace{0,\ldots}_{35},\underbrace{0.08}_{p^H_{W,c_2}},\underbrace{0,\ldots}_{62}}_{W},\underbrace{\underbrace{0,\ldots}_{70},\underbrace{0.33}_{p^H_{Q,c_3}},\underbrace{0,\ldots}_{29}}_{Q} \,]$$

Both strand and coil structures can be regarded as non-helix in the helix/non-helix classification. Thus, "PS_" is a non-helix subsequence with size 3, and '_' in "PS_" denotes a space needed for making up the required length. The encoding vector of "PS_" is V_2, which is shown below.

$$V_2 = [\, \underbrace{\underbrace{0,\ldots}_{52},\underbrace{0.33}_{p^H_{P,c_3}},\underbrace{0,\ldots}_{47}}_{P},\underbrace{\underbrace{0,\ldots}_{45},\underbrace{0.33}_{p^H_{S,c_3}},\underbrace{0,\ldots}_{54}}_{S}\ \underbrace{\underbrace{0,\ldots}_{100}}_{\_} \,]$$

4. PREDICTION FRAMEWORK

4.1 Training of Encoded Vectors Using SVM

Protein secondary structure prediction is a typical three-class classification problem. A simple strategy to handle the three-classification problem is to cast it to a set of binary categorizations. For an n-class classification, $n-1$ classes are used to construct training sets. The ith class will be trained with all of the samples in the ith class with "y = +1" labels and all other example data with "y = -1" labels. If input vectors are transformed from helix (strand or coil) segments, then their learned decision function is called *vertical decision function*, denoted f_v. Otherwise, if input vectors are transformed from context subsequences, then their learned decision function is called *horizontal decision function*, which can be further classified into *left horizontal decision function* derived from the left contexts subsequences, denoted f_{hl}, and *right horizontal decision function* derived from the right contexts subsequences, denoted f_{hr}. The final prediction synthesizes f_v and f_{hl} (f_{hr}) that will be discussed in detail in Section 4.3.

4.2 Sliding Window

The sliding window method is adopted to move the window over the test protein sequence. A window, which is a one-dimensional frame of slots, can be adopted for overlaiding on and moving over a test protein sequence to examine different segments of its amino acids. It extracts an protein segment of length L ($L \geq 1$) to match a potential structure class using the derived decision function generated after the training phase. Note that the size of the sliding window should be the same size of the n-gram so that the test vectors can be fed into the same input space with the training vectors.

4.3 Synthesis of Deduced Decision Functions

We develop a heuristic algorithm to synthesize the deduced decision function for the final prediction.

Algorithm 1. Prediction of a protein sequence s with a sliding window of size n, and return a labeled sequence.

```
string predict (string s, int n, SVMmodel model,
          SVMmodel l_model, SVMmodel r_model)
{
  bool mark = true;
  int pre_p = -1; //prediction of the previous segment
  string s_p = new(strlen(s)); // predict sequence
  int begin = 0; //begin position of sliding window
  while (begin < strlen(s)) {
     int current = svm_predict(segment(s,i,n), model);
     int left = svm_l_predict(l_seg(s, i, n), l_model);
     int right = svm_r_predict(r_seg(s, i, n), r_model);
     if (mark)
        begin += l_synthesis(s_p, begin, current,
                             left, n, &pre_p, &mark);
     else
        begin += r_synthesis(s_p, begin, current,
                             right, n, &pre_p, &mark);
  }
  return s_p;
}
int l_predict(int begin, int current, int left,
          int size, bool *pre_p, bool *mark)
{
  int len = size;
  if((current==-1) && (left==-1))
     { mark_seq(s_p, begin, size, -1); *pre_p = -1; }
  else if ((current==1) && (left==-1)) {
     if (begin_position == 0)
        { mark_seq(s_p, begin, len, 1); *pre_p = 1; }
     else {  mark_seq(s_p, begin, len, *pre_p);
             if (*mark)  *mark = false;
             else *mark = true;
     }
  }
  else if ((current==-1) && (left==1)) {
     len = size/2; mark_seq(s_p, begin, len, -1);
     *mark = false; *pre_p = -1;
  }
  else if ((current==1) && (left==1)) {
     len = 1; mark_seq(s_p, begin, len, 1);
     *pre_p = 1;
  }
  return len;
}
int r_predict(int begin, int current, int right,
            int size, bool *pre_p, bool *mark)
{
  int len = size;
  if((current==-1) && (right==-1))
     { mark_seq(s_p, begin, len, -1); *pre_p = -1; }
  else if ((current==-1) && (right==1)) {
     len = 1; mark_seq(s_p, begin, len, 1);
     *pre_p = 1;
  }
  else if ((current==1) && (right==-1)) {
     mark_seq(s_p, begin, len, *pre_p);
     if ((*pre_p)==1) *mark = true;
     else *mark = false;
  }
  else if ((current==1) && (right==1)) {
     len = size/2; mark_seq(s_p, begin, len, 1);
     *pre_p = 1;
  }
  return len;
}
```

For a helix/non-helix (strand/non-strand, coil/non-coil) classifier, we select the best accuracy value (will be discussed in Section 5.1), and the corresponding decision function $f(\vec{\mathbf{x}})$ (including the kernel function and the upper bound C), and n-gram for the prediction phase. Thus, given an unknown sequence s, we first use sliding window with the window size n and different biological groups (cardinal vectors) to transform subsequences of s into different sets of input vectors. Hereafter, for each set of vectors, we apply the derived decision functions to classify each subsequence into either the helix or non-helix (strand or non-strand, coil or non-coil) class. If different biological groups cause conflict predictions, a vote strategy will be used to decide the final prediction.

5. EXPERIMENTAL DATA

The experiments are carried out on the CB513 set and RS126 set with 513 and 126 non-redundant protein amino acid sequences, respectively, whose secondary structures are known. In these two data sets, the assignment of secondary structure is usually performed by DSSP[9], STRIDE[6], or DEFINE[15].

Different assignment methods have different explanations on partial of residues in protein sequences, therefore they will influence the prediction accuracy as discussed by the constructor of CB513. In this paper, we select DSSP assignment in order to compare with other prediction approaches. DSSP assignment classify eight states secondary structure, `H`, `G`, `I`, `B`, `E`, `T`, `S`, and `-` [1], into three classes *helix*(H), *strand*(E), and *coil*(C). Here, we use methods in [7; 17] that call `H`, `G`, and `I` as *helix*(H), `E` as *strand*(E), and other states as *coil*(C). This assignment is slightly different from other assignments reported in different literatures. For example, in the CASP competition[10], H contains DSSP classes `H` and `G`, while E contains DSSP classes `E` and `B`.

We use multiple cross-validation trials to minimize variation in the results caused by a particular choice of training or test sets. An sevenfold cross validation is used on CB513 (RS126), where CB513 (RS126) is divided into seven subsets randomly, and each subset has similar size. Among these seven subsets, data in six subsets are selected as training data and data in the remaining subset is test data.

5.1 Evaluation Metrics

For a given sequence, we evaluate our prediction approach and compare it with some prediction methods that have been well accepted using evaluation metrics. First, we evaluate the effectiveness of SVM in our prediction application domain. Hereafter, we select the best parameters (e.g. kernel function and upper bound C) in SVM according to the highest accuracy value in Equation 9, and use the selected paraments for further prediction and comparison. For the evaluation of the effectiveness of our prediction approach, we give the following metrics.

The prediction accuracy using SVM is calculated as:

$$AC_{SVM} = \frac{N_{c_i}}{N_i} \times 100\% \qquad (9)$$

where i is a test set, N_{c_i} is the number of residues correctly predicted in i, and N_i is the number of residues in i.

We also employed several standard performance measures to assess prediction accuracy. Three-state overall per-residue accuracy (Q_3) and segment overlap calculation SOV [17] were calculated to evaluate accuracy.

Q_3, The overall per residue three-state accuracy method [17], is the most widely-used method for assessing the quality of a predicting approach, which is defined as

$$Q_3 = \frac{N_H + N_S + N_C}{N} \times 100\% \qquad (10)$$

where N_H, N_S, and N_C are the numbers of residues correctly predicted in the helix, strand, and coil, respectively, and N is the number of residues in the predicted sequences. Q_3, however, does not consider the problem of over-predictions, e.g. non-helix residues are treated as helix, or under-predictions, e.g. helix residues are treated as non-helix.

Another popular accuracy measurement, the *Segment overlap calculation* (SOV) has been proposed to assess the quality of a prediction in a more realistic manner. This is done by taking into account the type and position of secondary structure segment, the natural variation of segment boundaries among families of homologous proteins and the ambiguity at the end of each segment. SOV is calculated as

$$Sov = \frac{1}{N}\sum_{s} \frac{minov(s_{obs}; s_{pred}) + \delta}{maxov(s_{obs}; s_{pred})} \times len(s_{obs}) \qquad (11)$$

where N is the number of residues in the predicted sequences, s is a segment in the predicted sequences, s_{obs} and s_{pred} are observed and predicted segments, respectively, $minov$ is the actual overlap between s_{obs} and s_{pred}, $maxov$ is the extent of the either segment, and $len(s_{obs})$ is the length of the observed segment. δ is an integer chosen to be smaller than $minov$ and smaller than one-half the length of s_{obs}, δ = 1, 2, or 3 for short, intermediate, and long segments.

The per-residue accuracy for each type of secondary structure ($Q_H, Q_E, Q_C, Q_H^{pre}, Q_E^{pre}, Q_C^{pre}$) was also calculated. We distinguish Q_I and Q_I^{pre} (here, I=H, E, and C) as:

$$Q_I = \frac{N_{I_{cp}}}{N_{I_o}} \times 100\%, \qquad Q_I^{pre} = \frac{N_{I_{cp}}}{N_{I_p}} \times 100\% \qquad (12)$$

where, $N_{I_{cp}}$ is number of residues correctly predicted in state I, N_{I_o} is number of residues observed in state I, and N_{I_p} is number of residues predicted in state I.

In the following sub-sections, we use LIBSVM [2], an SVM software, to evaluate our prediction approach from different point of views, which include (i) determination of kernel function and parameter C, (ii) selection of window size, and (iii) comparison with other prediction methods.

5.2 Determination of Kernel Functions and Parameters in SVM

Given a data set, a proper kernel function and its parameters must be chosen to construct the SVM classifier. Different kernel functions and paraments will construct different SVM classifier. We hope to select an optimal kernel function and its parameters which can achieve high AC_{SVM}. However, there are no successful theoretical methods for determining the optimal kernel function and its parameters. In [7], the author has proved that the gaussian radial basis function $\exp(-\gamma|\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j|^2)$ can provide superior performance in generalization ability and converge speed. Therefore, we select different parameters and upper bound values C on $\exp(-\gamma|\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j|^2)$ to compare the testing accuracies on helix/non-helix (H/¬H) test. Since the average of helix in

CB513 is eight, we use 8-gram to select the kernel function and parameter C for $\mathsf{H}/\neg\mathsf{H}$.

As shown in Table 4, for each C, we select γ from 0.001 to 50, where 0.001 is the ratio of 1 to the number of dimensions 800 ($20 \times 5 \times 8$) in the training set. We found that the best accuracy (84.72%) is achieved by $\exp(-\gamma|\vec{\mathbf{x}}_i - \vec{\mathbf{x}}_j|^2)$ with γ being equal to 0.2 and C being equal to 1.5. It is well known that a large C value imposes a high penalty to the classification errors [12]. However, Table 4 shows that for each γ between 0.001 and 50, when C increases, the accuracy on the test set first increases to a peak value, and then decreases. This is because too small a value of C gives a poor accuracy while too large a value of C leads to overfitting. A proper value of C balances these problems. We found that C=1.5 is appropriate for not only $\mathsf{H}/\neg\mathsf{H}$ but also other classifiers. In the following tests we set $C = 1.5$ and $\gamma = 0.2$ to construct different SVM classifiers.

5.3 Dependency of Window Size

There is a trade off between the number of residues used and the level of "noise" in deciding the size of the window. If the window size is too large, the prediction process may be misled by the noise, while if the window size is too small, then the number of residues for predicting the protein structure may be insufficient. Therefore, A proper window size could lead to a good performance.

We construct three different binary classifiers. The optimal window size, n_{op}, for each binary classifier was determined on the CB513. The results in Table 5 indicate that the optimal window size is related to the average length of the secondary structure segments. In general, longer n_{op} values mean segments require larger optimal window sizes. (The average length of helix, strand, and coil in CB513 are 8, 5, and 5, respectively.) It is interesting to find that the optimal window size based on our prediction approach is the similar size with the one based on previous NN approaches [13; 17], e.g. the optimal window sizes for $\mathsf{H}/\neg\mathsf{H}$ are both 13.

5.4 Effectiveness of Our Prediction Approach

We now evaluate the effectiveness of our prediction approach. We first compare our prediction approach with PHD [17] and an SVM-based method [7]. PHD is one of the most accurate and reliable secondary structure prediction methods based on NNs, and the SVM-based method is a new proposed method that can outperform PHD. Our comparison is based on the same data sets (CB513 and RS126), the same secondary structure definitions, and the same accuracy assessments. For easy to illustrate, we call the SVM-based method SVM_{01} and our prediction method SVM_{SSP}. The comparison results among PHD, SVM_{01}, and SVM_{SSP} are shown in Table 6, where '-' means that the results cannot be obtained from the papers.

On the CB513 set, the Q_3 is 75.2%, which is 1.7% higher than SVM_{01}, and the SOV with SVM_{SSP} achieves 77.8%, which is 1.6% higher than SVM_{01}. Q_3 and SOV scores for the RS126 set are improved by 3.0% and 2.9%, respectively, compared with PHD results.

The comparison results between PHD and the two SVM-based methods indicate that the methods based on SVM outperformed PHD, because SVM-based method can successfully avoid many problems which other machine learning approaches often encounter. For example, structures of NNs (especially the size of the hidden layer) are hard to be determined, gradient based training algorithms only guarantee finding local minima, too many model parameters have to be optimized, overfitting problems are hard to be avoided, etc. In addition, the comparison results between SVM_{01} and SVM_{SSP} prove that the encoding schema of SVM_{SSP} and two dimensional prediction framework play significant roles on predicting protein secondary structures.

Table 6: Comparison with results of other systems.

	CB513		RS126		
	SVM_{01}	SVM_{SSP}	PHD	SVM_{01}	SVM_{SSP}
Q_3	73.5%	75.2%	70.8%	71.2%	73.8%
Q_H	75.0%	77.5%	72.0%	73.0%	75.1%
Q_E	60.0%	64.8%	66.0%	58.0%	61.2%
Q_C	79.0%	79.9%	72.0%	75.0%	81.2%
Q_H^{pre}	79.0%	82.1%	73.0%	77.0%	80.9%
Q_E^{pre}	67.0%	70.5%	60.0%	66.0%	69.7%
Q_C^{pre}	70.0%	70.6%	-	69.0%	70.1%
Sov	76.2%	77.8%	73.5%	74.6%	76.4%

6. CONCLUSIONS

In this paper, we propose an approach in predicting protein secondary structures using SVM. Our prediction approach takes a segment extracted from a given sequence using a sliding window and matches it against the structural classification results computed by SVM. We incorporate the biological classifications of amino acid residues and statistical information into an encoding schema, consider not only structured segments but also their contexts, and choose the most ideal kernel function, parameters, and size of the window for the prediction of different protein structure classes. The experimental results show that our two dimensional prediction framework outperforms the existing prediction methods.

7. ACKNOWLEDGEMENTS

The authors would like to thank Jianyin Shao of the Chemistry and Biochemistry department at Brigham Young University for his helpful discussions and suggestions. Special thanks are also given to Chih-Chung Chang and Chih-Jen Lin for providing LIBSVM.

8. REFERENCES

[1] P. D. Bank. http://www.rcsb.org/pdb/, 2002.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: a Library for Support Vector Machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.

[3] J. A. Cuff and G. J. Barton. Evaluation and Improvement of Multiple Sequence Methods for Protein Secondary Structure Prediction. *Proteins: Struct. Funct. Genet.*, 34:508–519, 1999.

[4] H. Drucker, D. Wu, and V. Vapnik. Support Vector Machines for Span Categorization. *IEEE Trans. on Neural Networks*, 10:1048–1054, 1999.

[5] M. O. D. (ed). Atlas of Protein Sequence and Structure. *National Biomedical Research Foundation (Washington, D. C.)*, 5, 1972.

Table 4: Accuracies AC_{SVM} of different parameters γ and upper bounds C for H/¬H under 8-gram on CB513 set, where * marks the best accuracy for each C.

	γ							
C	0.001	0.01	0.2	0.3	0.5	1	5	50
0.01	50.00%	50.00%	51.39%	65.83%	80.83%	*83.89%	50.00%	50.00%
0.1	50.00%	50.00%	83.89%	83.89%	83.61%	*84.44%	83.61%	51.11%
1.5	51.39%	82.50%	*84.74%	81.94%	82.50%	80.28%	53.89%	53.89%
5	79.72%	79.72%	80.83%	*81.94%	81.39%	81.67%	53.89%	53.89%
50	78.61%	81.28%	*81.67%	81.11%	81.67%	82.78%	78.06%	53.89%

Table 5: Accuracies AC_{SVM} of different window sizes for each binary classifier on CB513.

Binary classifier	Window size (n-gram)							Best window size
	5	7	9	11	13	15	17	n_{op}
H/¬H	79.38%	83.91%	84.82%	85.99%	86.72%	86.13%	84.54%	13
E/¬E	79.26%	81.65%	83.89%	84.06%	83.38%	82.45%	81.61%	11
C/¬C	75.12%	78.18%	79.80%	81.06%	80.58%	80.47%	79.01%	11

[6] D. Frishman and P. Argos. Knowledge-Based Protein Secondary Structure Assignment. *Proteins*, 23:566–579, 1995.

[7] S. Hua and Z. Sun. A Novel Method of Protein Secondary Structure Prediction with High Segment Overlap Measure: Support Vector Machine Approach. *Bioinformatics*, 308:397–407, 2001.

[8] J.Garnier, D.J.Osguthorpe, and B.Robson. Analysis of the Accuracy and Implications of Simple Methods for Predicting the Secondary Structure of Globular Proteins. *J. Mol Biol*, 120:97–120, 1978.

[9] W. Kabsch and C. Sander. A Dictionary of Protein Secondary Structure. *Biopolymers*, 22:2577–2637, 1983.

[10] J. Moult and et al. Critical Assessment of Methods of Protein Structure Prediction (CASP): Round II. *Proteins. supplement 1.*, 29(S1):2–6, 1997.

[11] D. Nelson and M. Cox. *Lehninger Principles of Biochemistry Amino*. Worth Publishers, 2000.

[12] E. E. Osuna, R. Freund, and F. Girosi. Support Vector Machines: Training and Applications (A. I. Memo1602). MIT A.I.Lab, 1997.

[13] N. Qian and T. J. Sejnowski. Predicting the Secondary Structure of Globular Proteins Using Neural Network Models. *J. Mol. Biol*, 202:865–884, 1988.

[14] H. H. Rashidi and K. L. Buehler. *Bioinformatics Basics Applications in Biological Science and Medicine*. CRC Press, 2000.

[15] F. M. Richards and C. E. Kundrot. Identification of Structural Motifs from Protein Coordinate Data: Secondary Structure and First-Level Supersecondary Structure. *Proteins*, 3:71–84, 1988.

[16] B. Rost and C. Sander. Prediction of Protein Secondary Structure at Better Than 70% Accuracy. *J. Mol Biol*, 232:584–599, 1993.

[17] B. Rost, C. Sander, and R. Schneider. Redefining the Goals of Protein Secondary Structure Prediction. *J. Mol Biol*, 235:13–26, 1994.

[18] V. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, New York, 1995.

[19] M. J. Zvelebil, G. J. Barton, W. R. Taylor, and et al. Prediction of Protein Secondary Structure and Active Sites Using the Alignment of Homologous Sequences. *J. Mol Biol*, 195:957–961, 1987.

[20] D. Zwillinger, S. G. Krantz, and K. H. Rosen, editors. *Standard Mathematical Tables and Formulae (30th edition)*. CRC Press, 1996.

APPENDIX

A. CARDINAL VECTORS WITH PHYSIOCHEMICAL PROPERTIES

The exchange group and hydrophobicity are shown in Table 7 and 8, respectively.

Table 7: Exchange groups of amino acid residues.

Exchange groups	6D cardinal vectors	Amino acid residues
(A)	[1, 0, 0, 0, 0, 0]	`C`
(C)	[0, 1, 0, 0, 0, 0]	`A,G,P,S,T`
(D)	[0, 0, 1, 0, 0, 0]	`D,E,N,Q`
(E)	[0, 0, 0, 1, 0, 0]	`H,K,R`
(F)	[0, 0, 0, 0, 1, 0]	`I,L,M,V`
(G)	[0, 0, 0, 0, 0, 1]	`F,W,Y`

Table 8: Hydrophobicity groups of amino acid residues.

Hydrophobicity groups	2D cardinal vectors	Amino acid residues
hydrophobic(O)	[1, 0]	`C,D,E,G,H,K,N,Q,R,S,T,Y`
hydrophilic(I)	[0, 1]	`A,F,I,L,M,P,V,W`

Assuring Privacy when Big Brother is Watching

Murat Kantarcıoĝlu   Chris Clifton
Department of Computer Sciences
Purdue University
250 N University St
West Lafayette, IN 47907-2066
{kanmurat, clifton}@cs.purdue.edu

ABSTRACT

Homeland security measures are increasing the amount of data collected, processed and mined. At the same time, owners of the data raised legitimate concern about their privacy and potential abuses of the data. Privacy-preserving data mining techniques enable learning models without violating privacy. This paper addresses a complementary problem: What if we want to *apply* a model without revealing it? This paper presents a method to apply classification rules without revealing either the data or the rules. In addition, the rules can be verified not to use "forbidden" criteria.

Keywords

Privacy, Security, Profiling

1. INTRODUCTION

The new U.S. government CAPPS II initiative plans to classify each airline passenger with a green, yellow or red risk level. Passengers classified as green will be subject to only normal checks, while yellow will get extra screening and red won't fly. [2] Although government agencies promise that no discriminatory rules will be used in the classification and that privacy of the data will be maintained, this does not satisfy privacy advocates.

One solution is to have the government send the classification rules to the owners of the data (e.g., credit reporting agencies). The data owners would then verify that the rules are not discriminatory, and return the classification for each passenger. The problem with this approach is that revealing the classification rules gives an advantage to terrorists. For example, knowing that there is a rule "A one-way ticket paid in cash implies yellow risk", no real terrorist will buy one way tickets with cash.

This appears to give three conflicting privacy/security requirements. The data must not be revealed, the classifier must not be revealed, and the classifier must be checked for validity. Although these seem contradictory, we show that if such a system *must* exist, it can be done while achieving significant levels of privacy. We prove that under reasonable assumptions (i.e., the existence of one way functions, non-colluding parties) it is possible to do classification similar to the above example that provably satisfies the following conditions:

- No one learns the classification result other than designated party.
- No information other than the classification result will be revealed to designated party.
- Rules used for classification can be checked for the presence of certain conditions without revealing the rules.

While such a system still raises privacy issues (particularly for anyone classified as red), the security risks are significantly reduced relative to a "give all information to the government" (or anyone else) model.

2. RELATED WORK

This work has many similarities with privacy-preserving distributed data mining. This was first addressed for the construction of decision trees[11]. This was based on the concepts of secure multiparty computation (discussed below.) There has since been work to address association rules in horizontally partitioned data[6; 7], EM Clustering in Horizontally Partitioned Data[10], association rules in vertically partitioned data[13], and generalized approaches to reducing the number of "on-line" parties required for computation[8]. However, the goal in this paper is not to *learn* a data mining model, but to privately *use* a model. (The other approach to privacy-preserving data mining, adding noise to data before the mining process(i.e [1]), doesn't make sense when the goal is to privately evaluate the results of the model on specific data items.) A good survey of privacy and data mining can be found in [9].

2.1 Secure Multiparty Computation

In the late 1980's, work in secure multiparty computation demonstrated that a wide class of functions can be computed securely under reasonable assumptions (see Theorem 2.1.) We give a brief overview of this work, concentrating on material that is used later in the paper. For simplicity, we concentrate on the two party case. Extending the definitions to the multiparty case is straightforward.

Secure multiparty computation has generally concentrated on two models of security. The semi-honest model assumes each party follows the rules of the protocol, but is free to later use what it sees during execution of the protocol to compromise security. The malicious model assumes parties can arbitrarily "cheat", and such cheating will not compromise either security or the results (i.e., the results from the nonmalicious parties will be correct, or the malicious party will be detected.)

We assume an intermediate model: preserving privacy with non-colluding parties. A malicious party may corrupt the results, but will not be able to learn the private data of other parties without colluding with another party. This is a realistic assumption for our problem: Both parties want to get correct results (class of a passenger), and are unwilling to compromise this goal through malicious behavior. The real security issue is the potential for misuse of the data or rules if accidentally released. The risk that a terrorist would discover rules stored at an airline or credit agency is much greater than the risk that they would alter the complex software to defeat the system, particularly as altering the protocol to maliciously learn the rules could be detected by inserting passengers with known security classes as a continuous test of the system.

The formal definition of secure multiparty computation[5] provides a methodology for proving the security of our approach. Informally, this definition states that a computation is secure if the view of each party during the execution of the protocol can be effectively simulated by the input and the output of the party. This is not quite the same as saying that private information is protected. For example, assume two parties use a secure protocol to compare two positive integers. If A has 2 as its integer and the comparison result indicates that 2 is bigger than equal other site's integer, A can conclude that B has 1 as its input. Site A can deduce this information by solely looking at its local input and the final result - the disclosure is a fault of the problem being solved and not the protocol used to solve it. To prove a protocol is secure, we have to show that anything seen is not giving more information then seeing the final result.

Yao introduced secure multiparty computation with his millionaire's problem (and solution). Two millionaires want to know who is richer, without either disclosing their net worth[14]. Goldreich extended this to show there is a secure solution for *any* functionality[5]. The general secure two party evaluation is based on expressing the function $f(x, y)$ as a circuit and encrypting the gates for secure evaluation. This leads to the following theorem:

THEOREM 2.1. *[4] Suppose that trapdoor permutation exist. Then any circuit is privately computable (in the semi-honest model).*

Trapdoor permutations can be constructed under the intractability of factoring assumption.

This works as follows. The function f to be computed is first represented as a combinatorial circuit. Each party sends a random bit to the other party for each input, and replaces their input with the exclusive or of their input and the random bit. This gives each party a share of each input, without revealing anything about the input. The parties then run a cryptographic protocol to learn their share of the result of each gate in the circuit. At the end, the parties combine their shares to obtain the final result. This protocol has been proven to produce the desired result without disclosing anything except that result.

3. PRIVATE CONTROLLED CLASSIFICATION

We first formally define the problem of classifying items using decision rules, when no party is allowed to know both the rules and the data, and the rules must be checked for "forbidden" tests. This problem could be solved using the generic method described above. While we have a construction and proof of such a circuit, it is omitted due to space constraints. Instead, we present a comparatively efficient approach based on the notion of an untrusted third party that processes streams of data from the data and rule sources.

3.1 Problem Statement

Given an instance x from site Data with v attributes, we want to classify x according to a rule set R provided by site Government. Let us assume that each attribute of x has n bits, and let x_i denotes the i^{th} attribute of x. We assume that each given classification rule $r \in R$ is of the form $(L_1 \wedge L_2 \wedge \cdots \wedge L_v) \rightarrow C$ where C is the predicted class if $(L_1 \wedge L_2 \wedge \cdots \wedge L_v)$ evaluates to true. Each L_i is either $x_i = a$, or a don't care (always true). (While the don't care clauses are redundant in the problem definition, they will need to be included explicitly in the protocol to mask the number of clauses in each rule. By using don't cares, G can define rules with an arbitrary number of clauses; the other parties gain no information about the number of "real" clauses in the rule.) We assume that for any given x only one rule will be satisfied.

In addition, D has a set F of rules that are not allowed to be used for classification. In other words, D requires $F \cap R = \emptyset$. The goal is to find the class value of x according to R while satisfying the following conditions:

- D will not be able to learn any rules in R,
- D will be convinced that $F \cap R = \emptyset$ holds, and
- G will only learn the class value of x and what is implied by the class value.

3.2 Untrusted Non-colluding Site

To achieve a solution that is both secure and efficient, we make use of an untrusted, non-colluding site. Use of such a site was first suggested in [3]. Application to privacy-preserving data mining was discussed in [8]. The key to such a site is that without active help from one of the other sites it learns nothing, although by colluding with other sites it may be able to obtain information that should not be revealed. Thus, the only trust placed in the site is that it will not collude with any of the other sites to violate privacy. Although this seems like a strong assumption, it occurs often in real life. For example, bidders or sellers on e-bay assume that e-bay is not colluding with other bidders or sellers against them. In our approach, the untrusted site learns only the number of literals v, the number of rules $|R|$, and how many literals of a given rule are satisfied. It does not learn what those literals are, what the class is, how they relate to literals satisfied by other rules or other data items, or anything else except what is explicitly stated above.

3.3 Commutative Encryption

A key tool used in this protocol is commutative encryption. An encryption algorithm is commutative if the following two equations hold for any given feasible encryption keys $K_1, \ldots, K_n \in K$, any item to be encrypted $m \in M$, and any permutations of i, j: $\forall m_1, m_2 \in M$ such that $m_1 \neq m_2$

$$E_{K_{i_1}}(\dots E_{K_{i_n}}(m)\dots) = E_{K_{j_1}}(\dots E_{K_{j_n}}(m)\dots) \quad (1)$$

and for any given $k, \epsilon < \frac{1}{2^k}$

$$Pr(E_{K_{i_1}}(\dots E_{K_{i_n}}(m_1)\dots) = E_{K_{j_1}}(\dots E_{K_{j_n}}(m_2)\dots)) < \epsilon \quad (2)$$

Commutative encryption is used to check if two items are equal without revealing them. For example, assume that party A has item i_A and party B has item i_B. To check if the items are equal, each party encrypts its item and sends it to the other party: Party A sends $E_{K_A}(i_A)$ to B and party B sends $E_{K_B}(i_B)$ to A. Each party encrypts the received item with its own key, giving party A $E_{K_A}(E_{K_B}(i_B))$ and party B $E_{K_B}(E_{K_A}(i_A))$. At this point, they can compare the encrypted data. If the original items are the same, equation 1 ensures that they have the same encrypted value. If they are different, equation 2 ensures that with high probability the encrypted values are different. During this comparison, each site sees only the other site's encrypted value. Assuming the security of the encryption, this simple scheme is a secure way to check equality.

One example of a commutative encryption scheme is Pohlig-Hellman [12], based on the assumption of the intractability of the discrete logarithm problem. This or any other commutative encryption scheme will work for our purposes.

3.4 Protocol for Private Controlled Classification

We now show how to solve the problem presented in Section 3.1 between sites D, G, and an untrusted, non-colluding site C, where C learns only the number of attributes v, the number of rules $|R|$, and the number of literals satisfied by each rule for a given instance x.

The basic idea behind the protcol is that sites D and G send synchronized streams of encrypted data and rule clauses to site C. The order of attributes are scrambled in a way known to D and G, but not C. Each attribute is given two values, one corresponding to don't care, the other to its true value. Each clause also has two values for each attribute. One is simply an "invalid" value (masking the real value). The other is the desired result, either the a (for a clause $x_j = a$), or the agreed upon "don't care" value. C compares to see if either the first or second values match, if so then either the attribute is a match or the clause is a don't care. If there is a match for every clause in a rule, then the rule is true.

The key is that the don't care, true, and invalid values are encrypted differently for each data/rule pair in the stream, in a way shared by D and G but unknown to C. The order (is the first attribute the value, or the don't care value) also changes, again in a way known only to D and G. Since all values are encrypted (again, with a key unknown to C), the non-colluding site C learns nothing except which rule matches. Since the rule identifier is also encrypted, this is useless to C.

The protocol operates in three phases: encryption, prediction, and verification. Three methods are used for encryption, a one-time pad based on a pseudo-random number generator shared by D and G, a standard encryption method E to encrypt data sent to C, and a commutative method Ec for checking for forbidden rules. To aid in understanding the discussion, a summary of the functions / symbols used is given in Table 1.

Table 1: Function and Symbol Descriptions

Symbol	Description
E_K	Encryption with key K
Ec_K	Commutative encryption with key K
R_g	Common pseudo-random generator shared by site D, G
x	Data item to be evaluated, a vector of attributes x_j
A	Rules × attributes matrix of encrypted instances created by site D
$R[i]$	Rule i, consisting of clauses corresponding to attributes of x
B	Rules × attributes matrix of encrypted rules created by site G
$n_j, (n_j + 1)$	Values outside the domain of j^{th} attribute
i	Index for rules
j	Index for attributes
σ	Index for "match" and "invalid" pairs

Protocol 1 Private classification: encryption phase

Require: D and G share a pseudo-random generator R_g, G has a private generator Rp_g. $R_g(k)$ ($Rp_g(k)$) represents the k^{th} number generated by the pseudo-random generators. Let n be a vector of elements where n_j and $n_j + 1$ are values out side the domain of x_j.

```
$cntd = cntg = cntgp = 0$ ;

*At site D:*
$K_r = R_g(cntd++)$;
**for** $i = 1; i \leq |R|; i++$ **do**
  **for** each attribute $x_j$ of $x$ **do**
    $\sigma = (R_g(cntd++) \bmod 2)$
    $A[i][j][\sigma] = E_{K_r}(x_j \oplus R_g(cntd++))$ ;
    $A[i][j][(1-\sigma) \bmod 2] = E_{K_r}(n_j \oplus R_g(cntd++))$ ;
  **end for**
**end for**
send $A$ to site $C$

*At site G:*
randomly permute rules in $R$;
$K_r = R_g(cntg++)$;
let $R[i][j]$ is $i^{th}$ rule's $j^{th}$ literal;
let $R[i][v+1]$ is the predicted class for $i^{th}$ rule;
**for** $i = 1; i \leq |R|; i++$ **do**
  **for** $j = 1; j \leq v; j++$ **do**
    $\sigma = (R_g(cntg++) \bmod 2)$
    **if** $R[i][j]$ is of the form $X_j = a_j$ **then**
      $B[i][j][\sigma] = E_{K_r}(a_j \oplus R_g(cntg++))$ ;
      $B[i][j][(1-\sigma) \bmod 2] = E_{K_r}((n_j+1) \oplus R_g(cntg++))$ ;
    **else**
      $B[i][j][\sigma] = E_{K_r}((n_j+1) \oplus R_g(cntg++))$ ;
      $B[i][j][(1-\sigma) \bmod 2] = E_{K_r}(n_j \oplus R_g(cntg++))$ ;
    **end if**
  **end for**
  $r_c = Rp_g(cntpg++)$;
  $\bar{B}[i] = (r_c, E_{k_r}(r_c) \oplus R[i][v+1])$;
**end for**
send $B, \bar{B}$ to site $C$;
```

In the encryption phase, site D first encrypts every attribute of x using xor with a random number. The xor with random is effectively a one-time pad to prevent revealing anything to site C. An additional attribute is added corresponding to the don't care condition using the "illegal" value n_j not in the domain of x_j. Site G creates encrypted values for each literal based on whether it must match or is a don't care. For the don't care $B[i][j][(1-\sigma) \bmod 2]$ will be the same as $A[i][j][(1-\sigma) \bmod 2]$ (n_j xored with the same random number). G applies a slightly different cryptographic scheme for class values (again padding with a newly generated random each time), ensuring C doesn't see that different rules may have the same class value. This is necessary to ensure C doesn't learn about class distribution of different instances over multiple runs of the protocol. This is repeated for every rule (giving multiple encryptions of the same x, but with different encryption each time.) The encryption phase is describe in detail in Protocol 1.

Protocol 2 Private classification: prediction phase

```
Require: $A, B, \bar{B}$ be generated and sent to site $C$.
  At site C:
  for $i = 1; i \le |R|; i++$ do
    if $\forall j, 1 \le j \le v, (A[i][j][0] == B[i][j][0] \lor A[i][j][1] == B[i][j][1])$ then
      $(rs, cs) = \bar{B}[i]$;
      break;
    end if
  end for
  randomly generate $r_f$
  send $(rs, cs \oplus r_f)$ to D and send $r_f$ to G;

  At site D:
  receive $(rd, cd)$ from site $C$;
  send $cd \oplus E_{k_r}(rd)$ to site $G$

  At site G:
  receive $r$ from site $C$ and $c$ from site $D$
  output $c \oplus r$ as the classification result
```

Site C compares the vectors to find which rule is satisfied in its entirety. However, it cannot directly send the prediction result to site G as this would reveal which rule is satisfied, since this is the encrypted (and distinct for each rule) value rather than the actual class. C instead sends the result xored with a random number to site D. D decrypts this to get the class, but the true class value is masked by the random generated by C. Finally G can combine the information it gets from site C and site D to learn the classification. This process is fully described in Protocol 2.

3.5 Checking for Forbidden Rules

Protocols 1 and 2 generate the correct classification without revealing rules or data. Protocol 3 shows how C and D can test if $F \cap R = \emptyset$ (presumably before D returns the masked result to G in Protocol 2.)

The main idea of Protocol 3 is that equality can be checked without revealing the items using commutative encryption. Since site D knows which random numbers are used by site G, it can mask F in the same way using those random numbers and encrypt with E_{K_r}. Site D also gets the masked R after encryption by site C, encrypts those, and returns them to C. C now has the double encrypted version of the

Protocol 3 Private classification: verification phase

```
Require: Let $\bar{B}$ be the rule set received from $G$ in Protocol 1, $Ec$ be a commutative encryption method.
  At site C:
  $K = \{$ create $2 * |F| * v$ random keys $\}$
  for $i = 1; i \le |F|; i++$ do
    for $j = 1; j \le v; j++$ do
      $En[i][j][0] = Ec_{K[i][j][0]}(B[r_f][j][0])$
      $En[i][j][1] = Ec_{K[i][j][1]}(B[r_f][j][1])$
    end for
  end for
  send $En$ to site $D$.

  At site D:
  receive $En$;
  $Kd = \{$ create $2 * |F| * v$ random keys $\}$
  for $i = 1; i \le |F|; i++$ do
    for $j = 1; j \le v; j++$ do
      $\bar{En}[i][j][0] = Ec_{Kd[i][j][0]}(En[i][j][0])$
      $\bar{En}[i][j][1] = Ec_{Kd[i][j][1]}(En[i][j][1])$
    end for
  end for
  Generate $F_e$, the matrix of encrypted forbidden rules, from $F$ using the method used by $G$ to generate $B$ from $R$ in Protocol 1.
  for $i = 1; i \le |F|; i{+}{+}$ do
    for $j = 1; j \le v; j++$ do
      $Ef[i][j][0] = Ec_{Kd[i][j][0]}(F_e[i][j][0])$
      $Ef[i][j][1] = Ec_{Kd[i][j][1]}(F_e[i][j][1])$
    end for
  end for
  send $Ef$,$\bar{En}$ to site $C$

  At site C:
  receive $Ef$, $\bar{En}$;
  for $i = 1; i \le |F|; i++$ do
    for $j = 1; j \le v; j++$ do
      $Ez[i][j][0] = Ec_{K[i][j][0]}(Ef[i][j][0])$
      $Ez[i][j][1] = Ec_{K[i][j][1]}(Ef[i][j][1])$
    end for
  end for
  for $i = 1; i \le |F|; i++$ do
    if $\forall j, 1 \le j \le v, (\bar{En}[i][j][0] == Ez[i][j][0] \lor \bar{En}[i][j][1] == Ez[i][j][1])$ then
      output that $F \cap R = \emptyset$ does not hold.
    end if
  end for
```

classification rule set R. It now encrypts F. The items can be compared since $Ec_{k_1}(Ec_{k_2}(A)) = Ec_{k_2}(Ec_{k_1}(A))$.

3.6 Security of the Protocol

To prove that this protocol reveals nothing but the number of matching literals, we use the secure multiparty computation paradigm of defining a simulator whose output is computationally indistinguishable from the view seen by each party during execution of the protocol. This reduces to showing that the received messages can be simulated; the algorithm itself generates the rest of the view. During the encryption phase, D and G receive no messages. Assuming encryption is secure, output of the encryption is computationally indistinguishable from a randomly chosen string over the domain of the encryption output. We can define the simulator as follows: The simulator randomly generates numbers and assigns them to A_s. For B_s, first a number of locations corresponding to the number of matching literals are chosen at random. For these locations, $A[i][j][\sigma]$ is used for $B[i][j][\sigma]$. For other locations, random values are generated.

Assume that the output of the simulator's A_s, B_s is **not** computationally indistinguishable from the A, B seen during the protocol. Let M be the distinguisher. Then M must be able to distinguish between some $A[i][j][\sigma]$ and $A_s[i][j][\sigma]$. This means M can distinguish between a random number and $E_{K_r}(X \oplus R)$, contradicting our secure encryption assumption. For $\bar{B}$ a similar argument applies.

For the classification phase site D only sees r, d. Since both of them are random numbers, a simple random number generator can be used for the simulator. The message G receives can be simulated using a random number generator (simulating the message from C) and the actual result ($result \oplus r$).

For the verification phase note that what each party sees is an item encrypted by a random key for each field. Assume that the places of the literals that are equal is the part of the final result (changing the order of literals at each execution prevents this from revealing any real information.) Site C can use the same simulator given for the encryption phase. Therefore we can conclude that the method is secure under the stated assumptions.

In practice the site C would operate in a stream mode, with D sending each new instance x and G re-encrypting the rules R. To C these would appear to be continuous streams – the repetition in R is masked by the random pad. This avoids C learning anything based on results over multiple instances (e.g., giving C a single encrypted rule set for all instances would enable C to learn what rule was most common, even if it didn't know what that rule was.)

One interesting advantage to this method over the generic method is that by colluding (e.g., under a court order), any two sites could reveal what the third site has. For example if G does use a forbidden rule, C and D can collaborate to expose what that rule is. This is not directly possible with generic circuit evaluation, since a malfeasant G could delete the keys used in the protocol to permanently hide the forbidden rule.

4. COST ANALYSIS

Privacy is not free. Keeping the necessary information private requires many encryptions for each classification. Given v literals in each rule both sites G and D perform $O(|R| \cdot v)$ encryptions in the encryption phase. The prediction phase requires a single encryption. Verification requires $O(|F| \cdot v)$ encryptions. The total number of encryptions is then $O((|R| + |F|) \cdot v)$.

The communication cost of the protocol is similar. Assume the size of each encrypted value is t bits. The encryption phase sends $O(|R| \cdot v \cdot t)$ bits, and the prediction phase sends $3t$ bits. In the verification phase the set F is transmitted in encrypted form, $O(|F| \cdot v \cdot t)$ bits. The total communication cost is $O((|R| + |F|) \cdot v \cdot t)$ bits.

While the communication cost may seem high, particularly since this is the cost per instance to be evaluated, it is likely to work well in practice. Bandwidth is growing rapidly, it is generally latency that limits performance. This protocol adapts well to streaming - the small number of messages, and the fact that each site sends only one per phase, means a continuous stream of instances could be fed through the system. The throughput of the system could approach the bandwidth of the communication network.

Note that the generic circuit evaluation is likely to be significantly more expensive. We have a circuit construction that solves the problem with $O((|R| + |F|) \cdot v \cdot n))$ encryptions where n is the number of bits to represent a literal. The circuit size is $O(|R| \cdot |F| \cdot v \cdot n)$ gates, giving a bit transmission cost of $O(|R| \cdot |F| \cdot v \cdot n \cdot t)$. Due to the necessity of representing all possible input in the construction of the circuit, a significantly more efficient approach based on generic circuit evaluation is unlikely.

5. CONCLUSIONS

As the usage of data mining results for homeland security and other potentially intrusive purposes increases, privately *using* these results will become more important. We have shown that it is possible to ensure privacy without compromising the ultimate goal.

The protocol presented here is a first cut at addressing this problem. Developing a practical solution requires additional work. One example is negative clauses. We have a method to easily extend this protocol to handle clauses with negation, however it reveals at least an upper bound on the number clauses with negation. Developing a more secure approach, and proving that secure, is an open topic. Other issues include non-boolean rules (e.g., best match), allowing multiple rule matches, supporting less structured data formats (e.g., text), and demonstrating practical efficiency.

Continued research in privacy preserving data mining must take into account not just generating the results, but efficiently and privately using them. Another research direction is lower bounds on computation and the communication cost. We would like to explore the trades off between cost and privacy. We believe that research can develop methods to increase security in our lives while sacrificing less privacy than generally assumed.

6. REFERENCES

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, TX, May 14-19 2000. ACM.

[2] Computer assisted passenger pre-screening II program.

http://www.fcw.com/fcw/articles/2003/0310/news-tsa-03-10-03.asp.

[3] U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In *26th ACM Symposium on the Theory of Computing (STOC)*, pages 554–563, 1994.

[4] O. Goldreich. Secure multi-party computation, Sept. 1998. (working draft).

[5] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game - a completeness theorem for protocols with honest majority. In *19th ACM Symposium on the Theory of Computing*, pages 218–229, 1987.

[6] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, pages 24–31, Madison, Wisconsin, June 2 2002.

[7] M. Kantarcıoĝlu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE-TKDE*, submitted.

[8] M. Kantarcioglu and J. Vaidya. An architecture for privacy-preserving mining of client information. In C. Clifton and V. Estivill-Castro, editors, *IEEE International Conference on Data Mining Workshop on Privacy, Security, and Data Mining*, volume 14, pages 37–42, Maebashi City, Japan, Dec. 9 2002. Australian Computer Society.

[9] Special section on privacy and security. *SIGKDD Explorations*, 4(2):i–48, Jan. 2003.

[10] X. Lin and C. Clifton. Privacy preserving clustering with distributed EM mixture modeling. *Knowledge and Information Systems*, Submitted.

[11] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Advances in Cryptology – CRYPTO 2000*, pages 36–54. Springer-Verlag, Aug. 20-24 2000.

[12] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, IT-24:106–110, 1978.

[13] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, Edmonton, Alberta, Canada, July 23-26 2002.

[14] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE, 1986.

Dynamic Inference Control

Jessica Staddon
Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA 94304, USA
staddon@parc.com

ABSTRACT

An inference problem exists in a multilevel database if knowledge of some objects in the database allows information with a higher security level to be inferred. Many such inferences may be prevented prior to any query processing by raising the security level of some of the objects, however this inevitably impedes information access, as a user with low authorization who queries just one of the objects with raised security must seek clearance even when not in danger of making the inference. More flexible access control is possible when inferences are prevented during query processing, however this practice can result in slow query response times. We demonstrate that access control can be made sufficiently dynamic to ensure easy access to the information users are entitled to, while retaining fast query processing. Our inference control schemes provide collusion resistance and have a query processing time that depends only on the length of the inference channels (not on the length of user query histories). In addition, our schemes provide a property we call *crowd control* that goes beyond collusion resistance to ensure that if a large number of users have queried all but one of the objects in an inference channel, then no one will be able to query the remaining object regardless of the level of collusion resistance provided by the scheme.

1. INTRODUCTION

In a multilevel database an inference problem exists if users are able to infer sensitive information from a sequence of queries that each have a low security classification (i.e. are not sensitive). For example, any user may be able to query a database to retrieve a list of ship names and the ports at which they are docked. In addition, the knowledge of which ports are being used to load ships with weapons may have a low security classification. Yet, these two queries together constitute an *inference channel*, because if both are made then it's possible to infer exactly which ships are carrying weapons, and this may be sensitive.

When protecting against inferences, there is an inherent trade-off between the granularity of the access control and the query processing time. The approach to inference control proposed in [17; 21] requires essentially no query processing. In [17; 21], the security levels of specific objects in the database are raised in order to prevent a user with low security clearance from completing enough queries to be able to make an undesired inference. By ensuring that at least one object in each inference channel requires high clearance, low-clearance users are prevented from making inferences. However, a user who only wants to query one particular object whose security level has been raised will be unable to do so without receiving additional authorization, even though the information they seek may be completely innocuous on its own. Hence, because access controls are predetermined, this approach may impede access to information unnecessarily .

Another approach to inference control is to determine at query time whether a query can be safely answered. This can be done, for example, by maintaining user query histories. When a user makes a query it is checked against the user's query history and all known inference channels, before granting access to the results of the query. More sophisticated methods of query-time inference control use inference engines to determine access [9]. However, since query histories can be quite long, this approach can result in slow query processing.

We introduce a new approach to inference control that allows for fast query processing while enabling fine-grained access control, and thus, flexible information access. In our approach, access-enabling tokens are associated with objects in the database, and users are allocated keys that they use to generate the necessary tokens. Once a token is used to query an object the key it was derived from cannot be used to query any other object in the inference channel. This is implemented by deleting (or, revoking) the tokens generated with this key from other objects in the channel. Hence, query processing depends on the length of the channel rather than the ever-growing user query histories. In addition, because initially the same tokens are associated with each object, our approach allows for flexible information access. A user can access any objects in the inference channel provided doing so will not enable the user to make the undesired inference, even through collusion with other users.

Our approach to inference control is inspired by cryptographic revocation techniques for large groups of users. The motivating intuition behind the use of these techniques is that group dynamics play an essential role in ensuring inference control: it is not enough to only consider the user requesting the object when deciding whether or not to grant access, instead all of the users of the database and all of the queries they've made, should somehow be taken into account. The difficulty comes in finding a way to do this without relying on the time-consuming processing of user query histories. As an example, one might imagine solving the problem by associating counters with objects in the channel, and cutting off access to the channel when the counters get too high. However, the counters reflect x queries by 1 user and 1 query by each of x users in the same way, and this doesn't sufficiently capture the access dynamics. By leveraging ideas from group communications we are able to provide some separation between these cases and an automated mechanism for updating the access controls that is far more likely to be affected by large-scale querying by a few users, than scattered queries by many users. More precisely, our schemes provide collusion resistance and a desirable new property that we call *crowd control.* Crowd control ensures that if a large number of users have queried all but one of the objects in the channel then no one will be able to query the remainder of the channel even if they have never queried the database before.

Overview. In Section 1.1 we discuss related work and in Section 2 we provide the necessary background definitions and notation. In Section 3 we provide our two inference control schemes and in Section 4 we discuss their attributes and highlight areas for future work.

1.1 Related Work

Our approach to inference control relies on the identification of inference channels in an initial pre-query processing step. Of course, it is impossible to identify all inference channels [24] but previous work has demonstrated significant success in identifying inferences both from the database schema [6; 17; 2; 26] and the data itself [5; 8; 28]. Our inference control techniques work with either approach, however we note that if the latter approach is used then it should be re-run periodically to accommodate changes in the data and this *could* lead to a need to distribute additional keys to users (if, for example, an inference channel of a length exceeding all others is identified).

Inferences may also be detected at query processing time (see [9; 25; 12]). This approach may lead to long query processing since it most effectively operates on a user's entire query history (in conjunction with a logic-based inference engine).

Once the channels are identified we assign keys to users and tokens to the objects in the inference channels in such a way that users have flexibility in the objects they choose to access, provided the users are unable to complete the channel. Because the queries made by each user must affect the access capabilities of other users in order to ensure collusion resistance and the desirable property of crowd control, we draw inspiration for our schemes from secure group communication. In particular, the key allocation method employed in our schemes is similar to what's currently known as a *subset-cover* scheme [14] in the group communication setting. We believe that these cryptographic techniques are in many ways better suited to the inference control problem than the group communication setting. For example, if a large number of users have queried all but one object in an inference channel then it may be wise to consider this information to be in the public domain and to block all users (even those who have never queried the database) from querying the remaining object in the channel. The analogous situation in group communication is the revocation of an innocent user as the result of the revocation of several unauthorized users. This is a well-known problem in group communication that much work has gone into remedying (see, for example, [10]) yet it is a desirable property of an inference control scheme (what we call crowd control).

As mentioned in Section 1, an alternative approach to inference control that can be viewed as fitting in our general framework is to maintain a bit for each object indicating whether or not it has been released as the result of a query (see, for example, [9]). Our approach allows for more accurate access control because we can distinguish between n users each accessing $m-1$ objects in an inference channel of length m, and $n(m-1)$ users each accessing a single object in the channel. The method of [9] cannot, and this distinction is important since in the former case the remaining object in the channel should probably be blocked from all users if n is large, but in the latter case this may be unnecessary.

Finally, we note that if it is determined that a user's current query will enable them to make an undesired inference then this may be prevented by using query response modification (see, for example, [9; 23]). Our techniques can be used in conjunction with query response modification, that is, rather than requiring that the user receive higher authorization before completing the inference channel, it is also possible to simply modify the response, provided that still yields sufficiently useful information.

A survey of many of the existing approaches to inference control is in [4].

2. PRELIMINARIES

We denote the number of users of the database by n, and the users themselves by, $U_1, \ldots, U_n$. We view inference channels as being composed of objects. For our purposes, "object" is a generic term to describe a unit of information that is recoverable from the database, for example, a fact, attribute or relation. The example inference channel of Section 1 consists of objects that are each relations: the relation between ships and ports, and the relation between ports and weapons. We use m to denote the number of objects in an inference channel, and let $O_1, \ldots, O_m$ denote the objects in the channel. We sometimes refer to an inference channel of length m as an m-channel. The ship-port inference channel of Section 1 is a 2-channel.

In the initialization phase, users receive a set of encryption keys that they use to prove authorization (perhaps in addition to authenticating themselves to the database) to access the objects in an inference channel. Users prove this by encrypting some information specific to their object of interest. For example, the token might be an encryption of the attribute names needed to pose the query to the database. Referring back to the example of Section 1, we might require that the user form the token that is an encryption, under an approved key, of the attributes "ship name" and "port" concatenated together (i.e. the attributes are first represented as bit strings, then concatenated and then encrypted).[1]

We denote the set of keys allocated to U_i by K_i, $i = 1, \ldots, n$. Each encryption key may be known to several users. Before any queries have been made, each encryption key can potentially be used to access any object in the channel. However, users only have enough keys to generate tokens for a proper subset of the objects in the channel. We say a user has *max-*

[1]It is possible that with this approach a sequence of queries will be treated as though they form an inference channel when they do not. This is a common problem in inference control. We discuss ways to remedy this in Section 4.

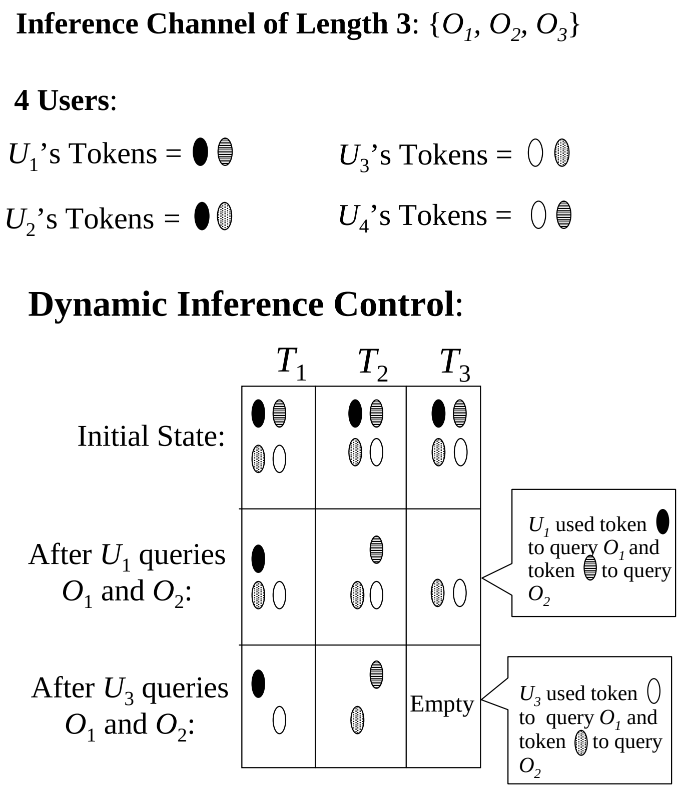
imally queried the channel if they have used all possible keys to query the channel. By limiting the number of keys per user, we guarantee collusion resistance–a coalition of users cannot together query all the objects in the inference channel. The following definition makes this more precise.

Definition 1. *Let c be an integer, $0 < c \leq n$. We say that an inference protection scheme is c-collusion resistant if c users acting in collusion are unable to query all the objects in any inference channel.*

Once an encryption key is used to gain access to object O_i, the same key cannot be used to gain access to any object, O_j, $j \neq i$, in the same inference channel. This is accomplished by deleting (or, revoking) the tokens generated by that key from the rest of the inference channel. In other words, part of the automated access control mechanism is to update the list of acceptable tokens for objects in the inference channel each time an object in the channel is accessed (this processing time depends on the length of the channel). Hence, because a key may be used by many users, the queries of each user potentially affect the access capabilities of many users. For example, if two of the keys used to query an object in the channel belong to the same user, then this user will be unable to maximally query the channel. These properties allow us to achieve a property that we call *crowd control*: if a lot of users have queried a maximal portion of an inference channel (e.g. $m-1$ out of m objects) then no user should be able to complete the inference channel by making the remaining query. The reasoning here is that if an object has been queried a lot it is more likely to have been leaked and so it may be prudent to consider the query results to be in the public domain. So, if this is the case for most of the channel, access to the remaining object should be explicitly prohibited.

Definition 2. *Let $0 < \epsilon < 1$, and let U denote a randomly selected user. Consider a c-collusion resistant inference protection scheme. If when more than x sets of c users have together queried some set of $m-1$ objects, $O_{i_1}, \ldots, O_{i_{m-1}}$, in inference channel $\{O_1, \ldots, O_m\}$, then with probability at least $1-\epsilon$, U cannot access the remaining object, $\{O_1, \ldots, O_m\} - \{O_{i_1}, \ldots, O_{i_{m-1}}\}$, we say the scheme has (x, c, ϵ)-crowd control.*

Figure 1 is a high level example of how token sets, and consequently access control, changes as a result of user queries. The figure simplifies our approach in two ways. First, we don't show the keys each user receives, but rather just the tokens they generate from them (depicted here as ovals). The second, and more important, simplification is that tokens corresponding to different objects appear identical. In reality, the tokens are particular to the object with which they are associated, while the encryption keys used to generate them may be the same (see the discussion in Section 4 for more on this).
An important part of our analysis is assessing how information access changes as more users query the channel. To do this we need to understand how one user's set of keys relates to another user's set of keys. This is important because with each query the set of acceptable tokens for every other query can change. More precisely, we often study how much the key sets of one group of users, say $U_1, \ldots, U_r$, *cover* the key set of user, U_{r+1}. The size of the cover is the number of keys



Inference Channel of Length 3: {O_1, O_2, O_3}
4 Users:
U_1's Tokens =
U_3's Tokens =
U_2's Tokens =
U_4's Tokens =
Dynamic Inference Control:
T_1
T_2
T_3
Initial State:
After U_1 queries O_1 and O_2:
After U_3 queries O_1 and O_2:
Empty
U_1 used token to query O_1 and token to query O_2
U_3 used token to query O_1 and token to query O_2

Figure 1: In this example there are four users each with two tokens (or, two keys that they use to generate tokens) and collusion resistance is $c = 1$. For $i = 1, 2, 3$, the ith column indicates which tokens can be used to access object O_i after the queries listed on the left hand side have been executed. After both U_1 and U_2 have queried objects O_1 and O_2, no one can query object O_3 (it has no more acceptable tokens) but everyone can still access both O_1 and O_2.

U_{r+1} has in common with at least one of $U_1, \ldots, U_r$, that is, the value: $|K_{r+1} \cap (\cup_{i=1}^{r} K_i)|$.
Finally, for simplicity of exposition, we often assume that a fractional quantity (i.e. $\frac{a}{b}$) is an integer. This should be clear from context.

3. DYNAMIC INFERENCE CONTROL

We assume that inference channels have been identified (for example, using a tool such as [17]) prior to the deployment of our inference control scheme. Our protocol consists of three phases:

- Key Allocation: Users are allocated $(m_{max}-1)/c$ keys, where m_{max} is the maximum length of an inference channel in the database, and c is the desired degree of collusion resistance.

- Initialization of the database: For each inference channel $\mathcal{Q} = \{O_1, \ldots, O_m\}$, a set of tokens, T_i, is associated with each object such that each user is capable of generating exactly $(m-1)/c$ tokens in T_i, for $i = 1, \ldots, m$. Initially, that is prior to any queries, the token sets are identical: $T_1 = T_2 = \ldots = T_m$.

- Query processing: If token $t \in T_i$ is used to gain access to O_i, then for every $s \neq i$, any token in T_s that was generated by the same key is deleted. Hence, the token sets change as queries are made.

We present two schemes that differ in how the first stage is completed; initialization of the database is essentially the same for both and query processing is as described above. The first is a simple randomized scheme that achieves probabilistic guarantees on crowd control (that is, $\epsilon > 0$). We analyze that scheme according to the crowd control and information access it permits as a function of the number of users querying the database.
We also present an algebraic scheme that offers deterministic guarantees on information access. Due to space constraints we do not analyze the scheme, although its analysis is similar to that of the randomized approach.

3.1 A Probabilistic Key Allocation Scheme

To allocate keys to the users we adopt a "bucket"-based approach that is often used in secure group communication (see, for example, [10]). Let there be $(m_{max} - 1)/c$ buckets, $B_1, \ldots, B_{(m_{max}-1)/c}$, each containing q encryption keys (that is, there are $q(m_{max} - 1)/c$ keys total). The keys themselves are randomly generated and are of a bit length suitable for the symmetric encryption scheme being used. For $i = 1, \ldots, n$, U_i receives a randomly selected key from each bucket for a total of $(m_{max} - 1)/c$ keys per user.
Token sets for an inference channel of length $m \leq m_{max}$ are formed by choosing a subset of $(m-1)/c$ buckets, $B_{i_1}, \ldots, B_{i_{\frac{m-1}{c}}}$, and using the keys in each bucket to generate the tokens for each object in the m-channel, as described in Section 2 (i.e. initially, every key in $B_{i_1} \cup \ldots \cup B_{i_{\frac{m-1}{c}}}$ can be used to access each object). Hence, before any queries are made, each user has the ability to query any $\frac{m-1}{c}$ objects, and so the scheme is c-collusion resistant. The following theorem shows how, for a given value of ϵ, the degree of crowd control afforded by the scheme depends on m and q. The idea behind the theorem is that a large set of users are likely to cover the key set of another user, so if these users have maximally queried the channel, the other user will be blocked.

Theorem 1. *Let $0 < \epsilon < 1$, and let c denote the collusion resistance of an instance of the probabilistic inference control scheme. Let $x = \frac{\ln(1-(1-\epsilon)^{c/(m-1)})}{\ln(1-c/q)}$ then the scheme has (x, c, ϵ)-crowd control.*

Proof. It suffices to show that if more than $x = \frac{\ln(1-(1-\epsilon)^{1/(m-1)})}{\ln(1-1/q)}$ sets of c users have queried $m-1$ objects, $O_{i_1}, \ldots, O_{i_{m-1}}$, in an inference channel of length m then the probability that a user U can query O_{i_m} is less than ϵ. U is unable to query O_{i_m} if the key sets of the users who have queried any of the other objects in the channel cover the relevant part of U's key set (i.e. those keys of U's that are useful for querying this channel). Of course, this happens with probability 1 if U has made $(m-1)/c$ other queries, and otherwise, the probability of this covering is at least $(1-(1-c/q)^x)^{(m-1)/c}$. Setting the latter quantity to be at least $1-\epsilon$ and solving for x gives the quantity in the theorem statement. $\square$

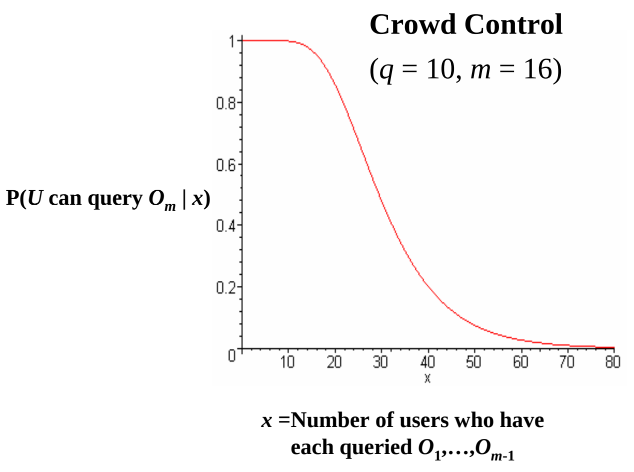


Figure 2: This figure shows how U's access to an object in an m-channel changes as the number of users who have each accessed the $m-1$ other objects in the channel grows. Here, $c = 1$, $q = 10$ and $m = 16$ (hence, this scheme can accommodate 10^{15} users). When the number of users is 70 or more, U can only access the object with probability at most .01.

Note that this theorem can be used to lower bound the probability that U can access object O_m when x sets of c users each have accessed *any* portion (i.e. not necessarily a maximal portion) of $O_1, \ldots, O_{m-1}$. However, it is a far weaker bound in this case. We claim that this is as it should be: if a large number of users have maximally queried the channel then for security purposes the remainder of the channel should be blocked, however if the users have only partially accessed the channel then the security risk to leaving it accessible is far less. To get a better idea of how access changes as a function of x, we include a concrete example of the access probabilities in Figure 2.
Theorem 1 shows that if a particular $(m-1)$-subset of the objects in a channel has been queried a lot, then users will be unable to query the *remaining* object in the channel whether or not they've already made some queries. It's likely that most users will still be able to access some $((m-1)/c)$-subset of the queried objects, however, as the following lemma shows.

Lemma 1. *Let $0 < \epsilon < 1$, and let c denote the collusion resistance of an instance of the probabilistic inference control scheme. If $x > \frac{\ln(1-(1-\epsilon)^{c^2/m^2})}{\ln(1-1/q^2)}$ users have each maximally queried an m-channel, then with probability greater than $1-\epsilon$, a user U, who is not amongst the x users, can maximally query the same channel.*

Proof. A user U cannot maximally query the channel if two of U's keys have been used to query a single object. This is impossible if for every pair of U's keys, one of the users who has maximally queried the channel has both such keys. The probability of this is at *least* (note this is a coarse bound): $(1-(1-1/q^2)^x)^{(m/c)^2}$. Setting this quantity to be greater than $1-\epsilon$, we get the bound in the lemma statement. $\square$

The above results demonstrate how a threshold number of queries to a channel affect the access capabilities of other users, but they don't describe how information access changes before the threshold is reached. We prove a lower bound on information access as a function of the number of users querying the channel by using the fact that any of U's keys that have *not* been used to query an object in the channel, may be used to query *any* object in the channel.

THEOREM 2. *Let* $0 < \alpha < 1$. *Consider a* c*-collusion resistant instance of the probabilistic inference control scheme. If the number of users who have maximally queried the channel is* $x < \frac{q(1-\alpha)\epsilon^{c/((m-1)(1-\alpha))}}{e}$ *then with probability at least* $1-\epsilon$*, another user can access any subset of objects in the channel of size* $\frac{\alpha(m-1)}{c}$.

PROOF. Consider a set of $x+1$ users, $U \notin \{U_1, \ldots, U_x\}$, the expected number of U's keys that $U_1, \ldots, U_x$ cover, is $\frac{(m-1)x}{cq}$. We show that the probability that $U_1, \ldots, U_x$ cover more than $1-\alpha$ of the keys U has for querying the channel is less than ϵ. From Chernoff bounds [13], it follows that this is true when the following inequality holds: $\left(\frac{e^{(q(1-\alpha)/x)-1}}{\left(\frac{q(1-\alpha)}{x}\right)^{\frac{q(1-\alpha)}{x}}}\right)^{\frac{(m-1)x}{cq}} < \epsilon$. This inequality is satisfied by the x bound given in the theorem statement. □

To reduce user U's access to an inference channel, two users sharing distinct keys with U must use these keys to query the same object, O, in the channel. In such a situation, U is unable to use either key to query other objects in the channel, while U only needs one of the keys to query O, hence one of U's keys cannot be used. Hence, restricting a user's access to the channel requires more about the key allocation structure than we use in Theorem 2, and so we suspect that this lower bound is generally not tight (and Lemma 1 provides particular evidence that it's not tight).

3.2 A Variant with Deterministic Guarantees

The key allocation scheme of Section 3.1 guarantees crowd control and information access probabilistically as a function of the number of users querying an inference channel. In some settings deterministic guarantees are necessary. We can achieve some deterministic guarantees by using error-correcting codes to allocate keys to users. Specifically, we use Reed-Solomon codes (see, for example, [27]) to allocate keys to users. Reed-Solomon codes use a polynomial that's unique to each user to determine each user's codeword, or in our case, key set. Because two polynomials of the same degree intersect on at most as many points as their degree, two users in our inference control scheme will share at most as many keys as the degree of their polynomials. Using this fact, we construct a scheme with deterministic (i.e. $\epsilon = 0$) information access guarantees. The following makes the key allocation part of such a scheme more precise, the initialization of the database and the query processing are both just as before.

We consider all polynomials of degree t, $0 < t < (m_{min} - 1)/c$, over the finite field F_q of q elements, where m_{min} is the minimum length of an inference channel. To each user, U, we associate a unique such polynomial, $p_U(x) \in F_q[x]$. For each element $(\gamma, \beta) \in F_q \times F_q$ we generate a random key, $k_{\gamma,\beta}$ of the desired bit length. User U receives the set of keys $K_U = \{k_{\gamma,p_U(\gamma)} | \gamma \in A \subseteq F_q\}$, where A is a set of size $(m_{max} - 1)/c$. Note that this is very similar to the bucket-based construction of Section 3.1 except that using polynomials to allocate keys from the "buckets" gives more control over the overlap between users' key sets. The following lemma demonstrates one of the deterministic guarantees provided.

LEMMA 2. *Consider a* c*-collusion resistant inference protection scheme that uses the key allocation method of this section. If* $x < \frac{(m-1)(1-\alpha)}{ct}$ *users have maximally accessed an* m*-channel then another user can access any subset of objects in the channel of size* $\frac{\alpha(m-1)}{c}$ *with probability* 1.

PROOF. Consider a user U who is not among the x users who have maximally accessed the channel. U shares at most t keys with each of the x users, and so at most $tx < t(\frac{(m-1)(1-\alpha)}{ct}) = \frac{(m-1)(1-\alpha)}{c}$ keys total. Hence, U has more than $\frac{\alpha(m-1)}{c}$ keys that none of the x users have and can access a different object in the channel with each key. □

An analysis very similar (but a bit more involved due to the fact that keys aren't assigned independently) can be performed to prove crowd control and lower bounds on information access. The scheme performs comparably to the earlier one.

4. DISCUSSION AND OPEN PROBLEMS

We have concentrated on a single inference channel for simplicity of exposition. When using our methods to prevent inferences across multiple channels a potential problem arises when a single object appears in channels of different lengths. To ensure that no inferences can be made by exploiting the varying channel lengths it may be necessary to reduce the number of acceptable keys associated with objects in overlapping channels, thus reducing information access.

In addition to focusing on a single channel we have also assumed a fixed user base. Over time, however, it is likely that users will lose access to the database (i.e. be revoked) and new users will be added. The scheme of Section 3.1 has a fixed upper bound on the number of users it can accommodate: $q^{(m_{max}-1)/c}$ (the bound for the scheme of Section 3.2 may be less depending on t). To allow for the addition of new users we can choose q to be larger than is currently required. Of course, this will have consequences for crowd control: increasing q means users' key sets are more disjoint and so the queries of individual users tend to have less of an impact on the access capabilities of others.

When a user is revoked from accessing the database their keys must no longer be valid. Simply deleting the tokens generated from their keys might unfairly restrict the access of others, so rekeying of the valid users may be needed. There is a large body of work on revocation in group communication that we may leverage for this problem. For example, efficient techniques for key updating over a broadcast channel are provided in [16; 10]. In addition, we note that protection against key leaks can be built into the key allocation scheme by ensuring that if a group of users pool their keys a leak a subset of the resulting pooled set, we will be able to *trace* at least one of the culprit users. Such tracing capability can be incorporated into exactly the type of key allocation we use in this paper (see, for example, [19]) but it will tend to increase the crowd control threshold, x.

As mentioned earlier (and as Lemma 1 indicates) our lower bound on information access for the scheme given in Theorem 2 isn't tight. A proof that takes more of the combinatorics of the key allocation scheme into account should produce a better bound. In addition, there may be ways to reduce user storage while retaining inference control. Using similar techniques to those in [14], it may be possible to allocate to each user a smaller set of keys that they can then use to generate additional keys, and thus, tokens.

Our approach offers more flexible information access than existing approaches with fast query processing, but it still may not offer quite the flexibility afforded by schemes that rely on user query histories to determine access control. This is because, as mentioned in Section 1.1, we require users to form tokens that are derived from information that's specific, but not necessarily unique, to the object of the user's interest. We could easily base the tokens on unique information but we think that doing so will increase the total number of inference channels so much as to negatively impact performance.

Finally, we note that with our schemes it is possible to determine what information the user *could* have accessed just from the current access control state (i.e. the value of the token sets at various points in time). For example, any key that appears in a token set T_i and no others must have been used to query object O_i, whereas a key that appears in all the token sets could not have been used. Hence, if a user is known to be compromised, the authorities can use their knowledge of the user's keys to determine what information the user *could* have accessed. Of course, provided the users' keys sets are kept private unless such a security breach occurs, the fact that keys are shared amongst users also provides desirable privacy to the users of the database.

Acknowledgements

The author is grateful to Teresa Lunt for numerous helpful discussions and to David Woodruff for comments on an earlier draft of this paper.

5. REFERENCES

[1] A. Barani-Dastjerdi, J. Pieprzyk and R. Safavi-Naini. *Security in databases: A survey study.* Manuscript, 1996.

[2] L. Binns. *Implementation considerations for inference detection: Intended vs. actual classification.* In Proceedings of the IFIP WG 11.3 Seventh Annual Working Conference on Database Security, 1993.

[3] D. Denning and T. Lunt. *A multilevel relational data model.* In IEEE Symposium on Security and Privacy, 1987.

[4] S. Jajodia and C. Meadows. *Inference problems in multilevel secure database management systems.* In Information Security: An Integrated Collection of Essays, M. Abrams et al., eds., IEEE Computer Society Press (1995), pages 570-584.

[5] J. Hale and S. Shenoi. *Catalytic inference analysis: Detecting inference threats due to knowledge discovery.* In the IEEE Symposium on Security and Privacy, 1997, pp. 188-199.

[6] T. Hinke. *Database inference engine design approach.* In Database Security II: Status and Prospects, 1990.

[7] T. Hinke, H. Degulach and A. Chandrasekhar. *A fast algorithm for detecting second paths in database inference analysis.* Journal of Computer Security, 1995.

[8] T. Hinke, H. Degulach and A. Chandrasekhar. *Layered knowledge chunks for database inference.* In Database Security VII: Status and Prospects, 1994.

[9] T. Keefe, M. Thuraisingham, W. Tsai. *Secure query-processing strategies.* IEEE Computer, Vol. 22, No. 3, 1989, pp. 63-70.

[10] R. Kumar, S. Rajagopalan and A. Sahai. *Coding constructions for blacklisting problems without computational constructions.* In Advances in Cryptology – Crypto '99, pp. 609-623.

[11] T. Lunt. *Access control policies for database systems.* In Database Security II: Status and Prospects, pp.41-52.

[12] D. Marks, A. Motro, S. Jajodia. *Enhancing the controlled disclosure of sensitive information.* In Proc. European Symp. on Research in Computer Security, Rome, Italy, September, 1996.

[13] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 2000.

[14] D. Naor, M. Naor and J. Lotspiech. *Revocation and tracing schemes for stateless receivers.* In Advances in Cryptology – Crypto 2001.

[15] T. Lin. *Inference secure multilevel databases.* In Database Security VI, pp.317-333.

[16] M. Naor and B. Pinkas. *Efficient trace and revoke schemes.* In Financial Cryptography 2000.

[17] X. Qian, M. Stickel, P. Karp, T. Lunt and T. Garvey. *Detection and elimination of inference channels in multilevel relational database systems.* In IEEE Symposium on Security and Privacy, 1993.

[18] G. Smith. *Modeling security-relevant data semantics.* In IEEE Symposium on Security and Privacy, 1990.

[19] D. Stinson and R. Wei. *Key preassigned traceability schemes for broadcast encryption.* Lecture Notes in Computer Science 1556 (1999), 144-156 (SAC '98 Proceedings).

[20] B. Sowerbutts and S. Cordingley. *Database architectonics and inferential security.* In Database Security IV, pp. 309-324.

[21] M. Stickel. *Elimination of inference channels by optimal upgrading.* In IEEE Symposium on Security and Privacy, 1994.

[22] B. Thuraisingham. *Data mining, national security, privacy and civil liberties.* To appear in Knowledge Discovery and Data Mining Magazine (SIG KDD), 2003.

[23] B. Thuraisingham. *Mandatory security in object-oriented database systems.* In proceedings of OOPSLA '89.

[24] B. Thuraisingham. *Recursion theoretic properties of the inference problem.* In the IEEE Third Computer Security Foundations Workshop, 1990.

[25] B. Thuraisingham. *Towards the design of a secure data/knowledge base management system.* In Data Knowledge and Engineering, 1990.

[26] B. Thuraisingham. *The use of conceptual structures for handling the inference problem.* In Database Security V, pp.333-362.

[27] J. van Lint. *Introduction to Coding Theory.* Springer-Verlag, 1999.

[28] R. Yip and K. Levitt. *Data level inference detection in database systems.* In the IEEE Eleventh Computer Security Foundations Workshop, 1998.