An Augmented Visual Query Mechanism for Finding Patterns in Time Series Data

Eamonn Keogh[1], Harry Hochheiser[2], and Ben Shneiderman[2,3]

[1]Computer Science & Engineering Department
University of California - Riverside
Riverside, CA 92521
eamonn@cs.ucr.edu

[2]Department of Computer Science
Human-Computer Interaction Lab
[3]Institute for Advanced Computer Studies, and Institute for Systems Research
University of Maryland
College Park, MD 20742 USA
{hsh,ben@cs.umd.edu}

Abstract. Relatively few query tools exist for data exploration and pattern identification in time series data sets. In previous work we introduced *Timeboxes*. Timeboxes are rectangular, direct-manipulation queries for studying time-series datasets. We demonstrated how Timeboxes can be used to support interactive exploration via dynamic queries, along with overviews of query results and drag-and-drop support for query-by-example. In this paper, we extend our work by introducing Variable Time Timeboxes (VTT). VTTs are a natural generalization of Timeboxes, which permit the specification of queries that allow a degree of uncertainty in the time axis. We carefully motivate the need for these more expressive queries, and demonstrate the utility of our approach on several data sets.

1 Introduction

Time series data sets are ubiquitous, appearing in many domains including finance, meteorology, physiology and genetics. To date, most information visualization work on these data sets has focused on display and interactive exploration, often emphasizing the periodic nature of some calendar-based data sets [6]. Work in data mining has addressed the need for additional tools to identify patterns of trends of interest in these data sets. Algorithmic and statistical methods for identifying patterns [1,2,3,5,8,11] have provided substantial functionality in a wide variety of situations. In domains such as stock price analysis, familiar patterns have been named and identified as shorthand approaches to identifying trends of interest [12]. Tools for specifying dynamic queries over these data sets have recently been developed: QuerySketch supports query-by-example based on a sketch of a desired profile [20], and Spotfire's Array Explorer 3 supports graphical queries for temporal patterns [18].

In previous work we introduced timeboxes: an interactive mechanism for specifying queries on temporal data sets. Timeboxes are rectangular regions that are placed and directly manipulated on a timeline, with the boundaries of the region providing the relevant query parameters. In this paper we introduce an extension to timeboxes, which allow queries that have some flexibility in the time axis. Researchers in speech processing and other fields have long known the utility of such "time warped" queries. We call our new approach Variable Time Timeboxes (VTT). We carefully motivate the need for such queries, and demonstrate the utility of our approach on several data sets.

The rest of this paper is organized as follows, in Section 2 we provide an extensive review of Timeboxes, and introduce TimeSearcher, an application that uses timeboxes to provide an interactive environment for visualizing and querying time series. In Section 3 we motivate, introduce and test our new VTT approach. Section 4 considers related work. Finally, in Section 5 we offer some conclusions and directions for future work.

2 Timeboxes: Interactive Temporal Queries

Timeboxes are rectangular query regions drawn directly on a two-dimensional display of temporal data. The extent of the Timebox on the time (x) axis specifies the time period of interest, while the extent on the value (y) axis specifies a constraint on the range of values of interest in the given time period. More concretely, we define a timebox as follows.

Definition 1: A *timebox,* defined by two points (x_1, y_1) and (x_2, y_2), is a constraint on a time series indicating that for the time range $x_1 \le x \le x_2$, the dynamic variable must have a value in the range $y_1 \le y \le y_2$, (assuming $y_2 \ge y_1$).

Multiple timeboxes can be drawn to specify conjunctive queries. Data sets must match all of the constraints implied by the timeboxes in order to be included in the result set.

Creation of timeboxes is straightforward: the user simply clicks on the desired starting point of the timebox and drags the pointer to the desired location of the opposite corner. As this is identical to the mechanism used for creating rectangles in widely used drawing programs, this operation should be familiar to most users. Once the timebox is created, it may be dragged to a new location or resized via appropriate resize handles on the corners, using similarly familiar interactions.

Query processing occurs on mouse-up. When the user releases the mouse, the current position of the timebox is stored, the query is updated, and the new result set is displayed.

Construction of timeboxes is aided by drawing all of the items in the data set directly on the query area. This "graph envelope" display provides additional insight into the density, distributions, and patterns of change found among items in the data set, in a display that is reminiscent to a parallel coordinates visualization [10] (Figure 1).

The example data set shown in Figure 1 contains weekly stock prices for 1430 stocks and will be used in a brief scenario to illustrate the use of timeboxes. An analyst interested

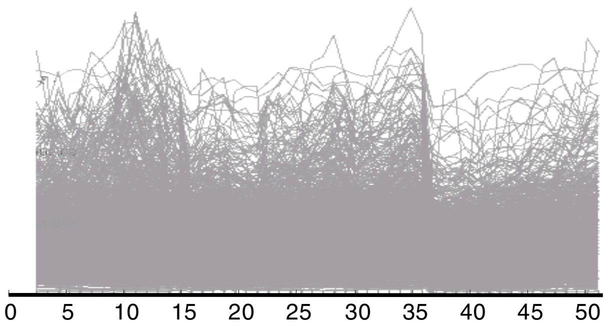


Fig 1. A "graph envelope" overview, formed by superimposing the time series for all of the items in the data set

in finding stocks that rose and then fell within a four-month period might start by drawing a timebox specifying stocks that traded between $28 and $64 during the first few weeks. When this query is executed, the graph envelope is updated to show only those records that match these constraints. We can quickly see that this query substantially limits the number of items under consideration, but many still remain (Figure 2.A).

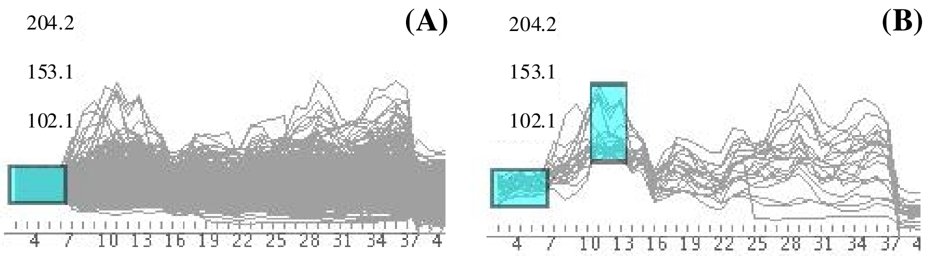


Fig. 2. (**A**) A single timebox query, for items between $28 and $64 during weeks 1-5. (**B**) A refinement of the query in (**A**) reduces the number of matching time series

To find stocks in this set that rose in subsequent weeks, the user draws a second box, specifying items that traded between $73 and $147 in weeks 10-12 (Figure 2.B).

As timeboxes are added to the query, the graph envelope provides an ongoing display of the effects of each action and an overview of the result set. Once created, the timeboxes can be scaled or moved singly or together to modify the query constraints.

The use of simple, familiar idioms for creation and modification of timeboxes supports interactive use with minimal cognitive overhead. Rapid automatic query processing (<100 ms) on mouse-up events provides the fast response necessary for dynamic queries [16], thus supporting interactive data exploration. Users can easily and quickly try a wide range of queries, and modify them to quickly see the effects of changes in query parameters. This ability to easily explore the data is helpful in identifying specific patterns of interest, as well as in gaining understanding of the data set as a whole.

Timeboxes also differ from traditional dynamic query widgets [16] in their construction and manipulation directly on the data space. As timeboxes are drawn directly on a graph space suitable for plotting a time series, the queries are easily interpreted at a glance.

2.1 TimeSearcher

TimeSearcher [9] uses timeboxes to pose queries over a set of entities with one or more time-varying attributes. Entities have one or more static attributes, and one or more time-varying attributes, with the number of time points and the definition of those points being the same for every entity in a given data set. If there are multiple time-varying attributes, any one of them can be selected for querying, through a drop-down menu that specifies the dynamic attribute being queried. All active queries refer to the same attribute.

When a data set is loaded, entities in the data set are displayed in a window in the upper left-hand corner of the application. Each entity is labeled with its name, and the values of the active dynamic attribute are plotted in a line graph. Complete details about the entity (details-on-demand) can be retrieved by simply clicking on the graph for the desired entity: this will cause the relevant information to be displayed in the lower right-hand window (Figure 3).

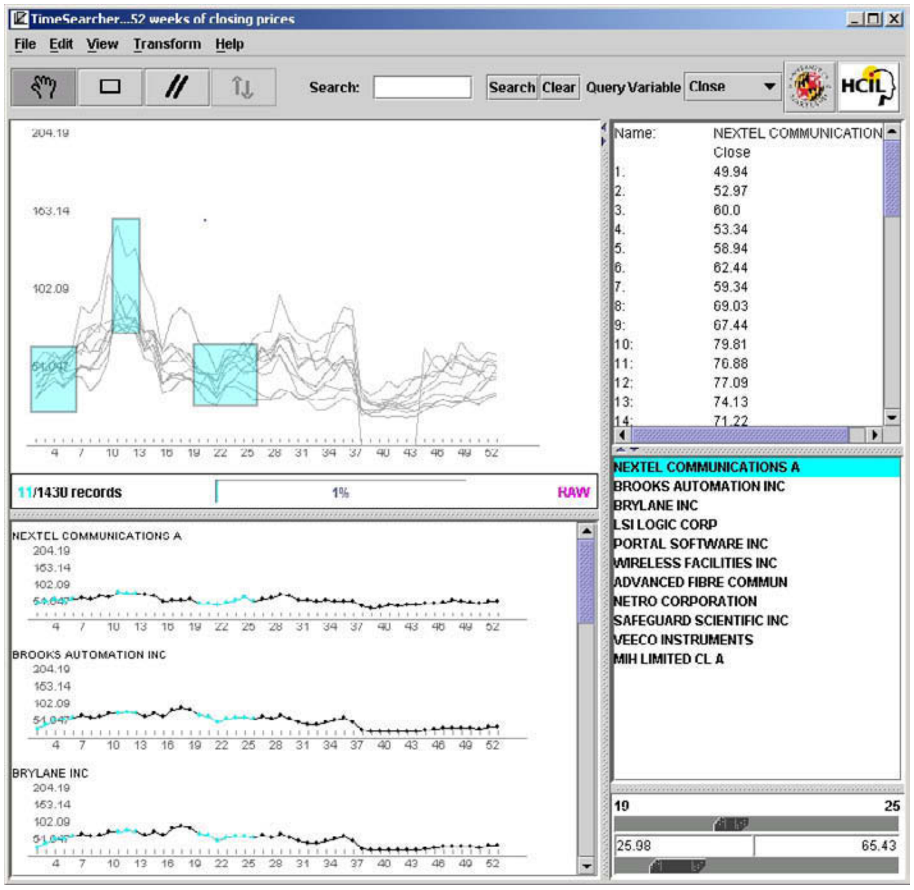
The upper-left corner of the TimeSearcher window is the query input space. This space initially contains an empty grid. To specify a query, users simply draw a timebox in the desired location. Query processing begins as soon as users release the mouse, signifying the completion of the box. Thus, users do not need to press a button to explicitly start a search. When query processing completes, the display in the top half of the application window is updated to show those entities that match the query constraints. For all of these entities, the time points that correspond to the queries are highlighted, in order to simplify interpretation of the display.

Once the initial query is created, the timeboxes can be moved and resized. The hand and box icons on the lower toolbar are used to switch between creating timeboxes and moving/resizing them. As is the case with initial timebox creation, query processing begins immediately upon completion of the movement/resizing of the timebox.

When multiple timeboxes are present, they can be modified individually or simultaneously in groups of two or more. This functionality is particularly useful for searches for complex patterns (Figure 3). In these cases, users can select some or all of the timeboxes (using standard lasso and shift-click interactions) and simultaneously apply the same translation and/or scale along either or both axes to all selected timeboxes. This is useful for searching for instances of a pattern that vary slightly in scale or magnitudes, or for modifying queries based on example items.

TimeSearcher uses "Graph Envelope" displays to provide overviews of the entire data set [9], and a simple drag-and-drop "query-by-example" mechanism supports the similarity queries often discussed in research in the mining of time series data [1,3,5,8,11].

TimeSearcher is implemented in Java, using the Swing toolkit for user-interface components. Drawing and scenegraph control in the data and query displays is provided by Jazz, a zooming toolkit written in Java [4].



Name:	NEXTEL COMMUNICATION
	Close
1.	49.94
2.	52.97
3.	60.0
4.	53.34
5.	58.94
6.	62.44
7.	59.34
8:	69.03
9:	67.44
10:	79.81
11:	76.88
12:	77.09
13:	74.13
14:	71.22

NEXTEL COMMUNICATIONS A
BROOKS AUTOMATION INC
BRYLANE INC
LSI LOGIC CORP
PORTAL SOFTWARE INC
WIRELESS FACILITIES INC
ADVANCED FIBRE COMMUN
NETRO CORPORATION
SAFEGUARD SCIENTIFIC INC
VEECO INSTRUMENTS
MIH LIMITED CL A

Fig. 3. The TimeSearcher application window. Clockwise from upper-left: query space, details-on-demand, item list, range sliders for query adjustment, and data items

3 An Augmented Query Mechanism

Timeboxes offer a very flexible query language, but it is not complete. To see why, consider the following motivating example. One of the classic symptoms of Hodgkin's disease is the appearance of two dramatic elevations of the patient's temperature in a 24-hour period. Figure 4 shows two examples.

One notable feature of such patterns is that the two peaks may be close together (almost to the point of merging) or as far apart as 18 hours. This uncertainly in the time axis is impossible to represent with Timeboxes. If we place two Timeboxes six hours apart (the mean value reported in the literature), we run the risk of missing positive examples which where the peaks are further apart or closer together.

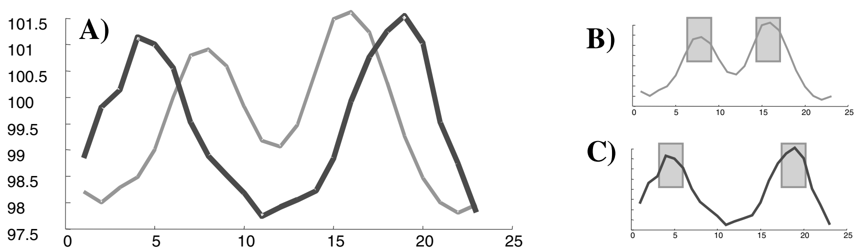


Fig. 4. Two peaks in a patients temperature over a single day is a classic symptom of Hodgkin's disease, however the peaks may be an arbitrary distance apart as shown in (A). Our current definition of TimeBoxes cannot detect peaks at arbitrary locations. Although we could create queries to find certain examples of double peak patterns (i.e B *or* C), we cannot use TimeBoxes to create a single query to which will discover all double peak patterns

Two overcome this limitation we can expand the definition of Timebox, to allow constraints of the following form. The time series of interest must be within a specified range for some specified duration *anytime* within a specified time window. We call such constraints, a Variable Time Timeboxes (VTT). We can define VTTs more concretely as follows.

Definition 2: A *Variable Time Timebox* (VTT), defined by two points (x_1, y_1) and (x_2, y_2) and a single integer R, is a constraint on a time series indicating that for the time range $x_1 - R \le x \le x_2 + R$, the dynamic variable must have a value in the range $y_1 \le y \le y_2$, for at least $(x_2 - x_1)$ consecutive time units, (assuming $y_2 \ge y_1$ and $x_2 \ge x_1$).

Figure 5 illustrates the difference between Timeboxes and VTTs. Note that Timeboxes can be considered a special case of VTTs, where the parameter R is set to zero. As such we can claim that VTTs are more flexible and expressive than Timeboxes.

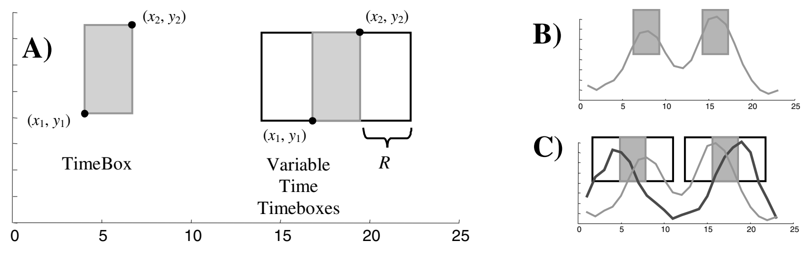


Fig. 5. A Timebox may be visualized as a shaded rectangle through which a qualifying sequence must pass. In contrast, an Variable Time Timeboxes (VTT) requires a qualifying sequence to have its value within the shaded region, allowing the shaded region to move anywhere within the larger constraining rectangle (**A**). Unlike Timeboxes (**B**), VTTS are able to detect patterns with a degree of uncertainty in the time axis

VTTs can be placed on the GUI the same way Timeboxes are. By default the *R*-value is set to zero, thus, on mouse-up they act as classic Timeboxes. However the user can left click the edges of the VTT to (symmetrically) resize the rectangle representing the R-value.

3.1 Efficiently Supporting VTTs

Although VTTS are more expressive than classic TimeBoxes, they also require more effort to support efficiently. As explained in previous work, Timebox queries can be processed via a modified orthogonal range tree query algorithm [7]. We can support VTTs efficiently by leveraging off previous work on indexing inverse time series queries [13]. An inverse query computes all time points at which a sequence contains values equal to the query value. A time series can be indexed by grouping sections with little variability in the Y-axis, and bounding the sections with a Minimum Bounding Rectangle (MBR). Figure 6 illustrates the idea.

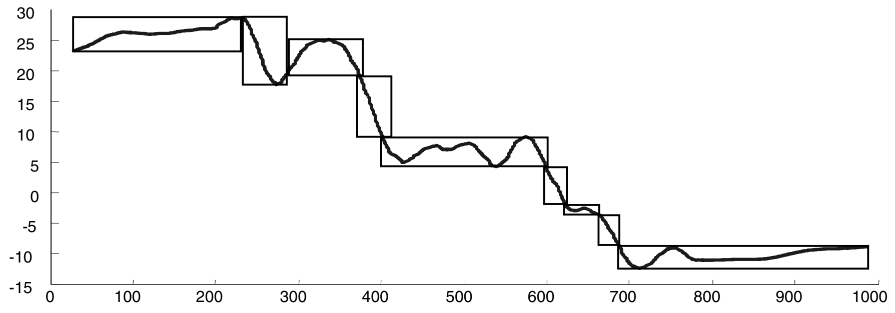


Fig. 6. A time series divided into 9 MBRs. Although the MBRs are actually one dimensional, they are shown as two dimensional for clarity

The MBRs are indexed using a R-Tree [8], a data structure that can efficiently support a variety of queries.

For a VTT query that asks for all time points that correspond to a time series that takes on a value equal to *y'*, where $y_1 \leq y' \leq y_2$, we perform a range query with the R-Tree to retrieve all the leaf queues that enclose value *y'*. The returned time series are guaranteed to contain the full answer set to the query, plus possibly some "false alarms", i.e. subsequences that on the value y', but for less than (x_2- x_1) consecutive time units. These false alarms are removed by a post-processing step. This technique is similar in spirit to the indexing technique introduced in [8]. Here the authors prove that any lower bounding technique can be used to index data, such that all qualifying sequences are retrieved. The only issue is the tightness of the lower bound. If the lower bound is very weak, many additional non-qualifying sequences will be retrieved (the so-called "false alarms"), although these can be removed in a post processing stage, this would cause the system to be degrade greatly in terms of speed. The other extreme, an arbitrarily tight lower bound, would allow a constant time access method. In general, this method works very well for most time series, since most real world time series have strong autocorrelation, and are therefore well approximated in by the low dimensionality MBRs, giving relatively tight bounds.

3.2 An Experiment to Demonstrate the Utility of VTTS

Our subjective evaluation of VTTs suggests that it is a useful tool for exploring large time series databases; in this section we provide an objective experimental evaluation of their utility. In particular, we will support our claim that VTTs allow more expressive queries, by showing experiments where our proposed approach outperforms previous work in the task of separating two different classes of time series.

In order to allow replication of our results, and to permit comparison to existing work, we have used the two most referenced times series datasets in the literature.

- **Cylinder-Bell-Funnel**: This synthetic dataset has been in the literature for 8 years, and has been cited at least a dozen times [21]. The dataset consists of 3 different basic shapes, which are produced by a function that has a stochastic element. For our experiments we consider only the "Funnel" and "Bell" classes.
- **Control-Chart**: This synthetic dataset has been freely available for the UCI Data Archive since June 1998 [21]. There are 6 different classes of time series. For our experiments we consider only the "Increasing Trend" and "Upward Shift" classes.

Our experiment consists of first showing the subject labeled examples of each class. We show the user as many as they wish to see, until they can identify unlabeled examples with 100% precision. The user is then shown a graph envelope view containing 10 examples of each of the two classes. The users task is to create a single query that can separate the two classes. The user attempts this with both Timeboxes and VTTS (with their choice of order). The experiment is repeated 10 times for each subject, and for each of the two datasets. Figure 7 shows an example of an experiment with the Control Chart dataset.

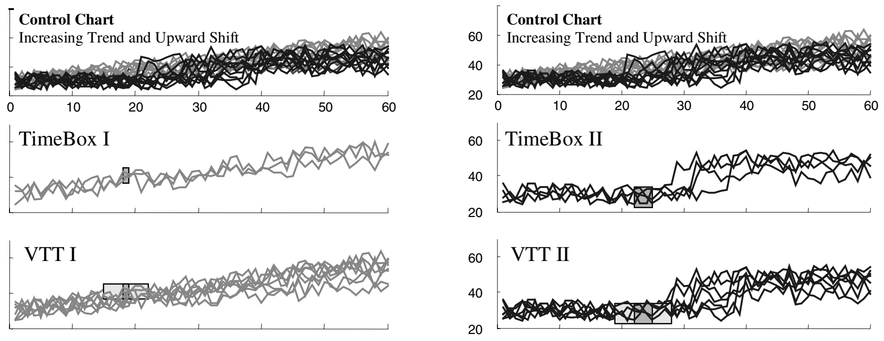


Fig. 7. The basic experiment setup on the Control Chart dataset. The user is shown a graph envelope containing 20 sequences, 10 each of "Increasing Trend" and "Upward Shift". The user is asked to create a single query to separate the two classes, using timeboxes and VVTs. The two queries may be placed at different locations, however for comparison purposes they are placed in exactly the same locations above. Note that for both cases VTTs are able to do a better job of separating the classes

In the example shown timeboxes separated out 3 of the 10 examples of "increasing trend", whereas VVTs were able to separate out 8 of the examples. For the second experiment we attempted to separate out just the "upward shift" sequences. Here timeboxes separated out 4 of the 10 examples of whereas VVTs were able to separate out 7 of the examples. Figure 8 depicts an example of an experiment with the Cylinder-Bell-Funnel dataset, where similar results can be observed.

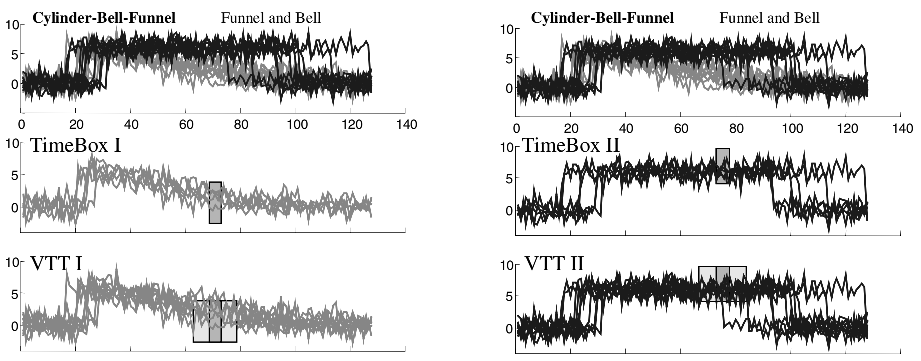


Fig. 8. The basic experiment setup on the Cylinder-Bell-Funnel dataset. The user is shown a graph envelope containing 20 sequences, 10 each of "Funnel" and "Bell". The user is asked to create a single query to separate the two classes, using timeboxes and VVTs. The two queries may be placed at different locations, however for comparison purposes they are placed in exactly the same locations above. Note that for both cases VTTs are able to do a better job of separating the classes

Our experimental subjects were 10 undergraduate students at the University of California Riverside. They were given a five-minute introduction to both timeboxes and VTTS, and allowed to "play" with both for ten minutes before the experiment began. We measured the quality of the separation Q, achieved by both tools as:

$$Q = 2 * (\textit{Correctly Separated} - \textit{False Positives}) / \textit{Size of Dataset}$$

Because our datasets has equal numbers of each of the two classes 10, this measure is in the range [-1, 1], with 1 indicating perfect separation of the classes. Table 1 summarizes the results of our experiments.

Table 1. The quality of the separation (Q) the two query mechanisms under consideration, on the Cylinder-Bell-Funnel and Control Chart datasets

	Timeboxes	Variable Time Timeboxes
Cylinder-Bell-Funnel	0.27	0.82
Control Chart	0.38	0.78

It is clear from these results that VTTs are a more powerful and intuitive tool for querying time series databases.

4 Related Work

Time series data accounts for an increasingly large fraction of the world's supply of data. A random sample of 4,000 graphics from 15 of the world's newspapers published from 1974 to 1989 found that more than 75% of all graphics were time series [19]. Visualizations of time-series data attempt to improve the utility of these common graphs, through the use of techniques such as increased data density or polar-coordinate displays that emphasize the serial periodic nature of the data set [7], or by distorting the time axis to realize denser information displays [14]. A recent survey of linear temporal visualizations is found in [17]. Generally, these tools focus on visualization and navigation, with relatively little emphasis on querying data sets.

A few tools have been developed for querying time-series data. MIMSY [15] provided an early example of searches for temporal patterns in stock market data, using text entry fields, pull-down menus, and other traditional widgets to specify temporal constraints. QuerySketch is an innovative query-by-example tool that uses an easily drawn sketch of a time-series profile to retrieve similar profiles, with similarity defined by Euclidean distance [20]. Although the simplicity of the sketch interface is appealing, the use of Euclidean distance as a metric can lead to non-intuitive results [11].

Spotfire's Array Explorer [18] supports graphically editable queries of temporal patterns, but the result set is generated by complex metrics in a multidimensional space. This potent approach produces useful results, but users may wish to constrain result sets more precisely.

Support for progressive refining of queries was addressed by Keogh and Pazzani, who suggested the use of relevance feedback for results of queries over time series data [11].

5 Conclusions

The additional expressive power provided by VTTs presents some additional challenges that merit further study. As VTTs require specification of an additional parameter, creation and manipulation will likely be more difficult than is the case with simple timeboxes. Identification of appropriate mechanisms for these tasks, perhaps including evaluation of alternative designs, will be needed to identify a preferred strategy. VTTs also raise questions of semantics: for example, what is the interpretation of overlapping VTTs? The interpretation of overlapping timeboxes is straightforward, but overlapping VTTs might confuse users. Other analogous extensions to the timebox model might also be possible. For example, variable value timeboxes (VVTs) might support variations in values similar to the valuations in time supported by VTTs. These queries would present further challenges in creation, interpretation, and efficient processing.

Acknowledgements. Harry Hochheiser is supported by a fellowship from America Online.

References

1. 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. The *VLDB Journal*, pages 490-501, 1995.
2. R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zait. Querying shapes of histories. *Proceedings of the 21st International Conference on Very Large Databases*, pages 502-514, 1995.
3. R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. L. P. Chen, editors, *Proceedings 11th International Conference on Data Engineering*, ICDE, pages 3-14, Taipei Tawian, March 1995. IEEE Press.

4. B. Bederson, J. Meyer, and L. Good. Jazz: An extensible zoomable user interface graphics toolkit in java, November 2000.
5. D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229-248. AAAI Press/MIT Press, 1996.
6. J. V. Carlis and J. A. Konstan. Interactive visualization of serial periodic data. *ACM Symposium on User Interface Software and Technology*, pages 29-38, San Francisco CA, November 1998. ACM Press.
7. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf Computational Geometry: Algorithms and Applications. Spring-Verlag, 2000.
8. C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 419-429, Minneapolis Minnesota, May 1994.
9. H. Hochheiser and B. Shneiderman. (2001). Interactive exploration of time-series data, In *Proc. Discovery Science 4th International Conference 2001*, Editors (Jantke, K. P. and Shinohara, A.), Springer-Verlag, Berlin. pp. 441-446.
10. A. Inselberg. Multidimensional detective. *IEEE Conference on Information Visualization*, Phoenix AZ, October 1997.
11. E. J. Keogh and M. J. Pazzani. Relevance feedback retrieval of time series data. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR '99*, pages 183-190, Berkeley CA, August 1999. ACM.
12. J. Little and L. Rhodes. Understanding Wall Street. Liberty Publishing, Inc., Cockeysville MD, 1978.
13. A. Nanopoulos and Y. Manolopoulos. Indexing Time-series Databases for Inverse Queries, *Proceedings 9th International Conference on Database and Expert Systems Applications* (DEXA 98), pages.551-560, Vienna, 1998
14. S. Powsner and E. Tufte. Graphical summary of patient status. The Lancet, 344:386-389, 1994.
15. G. Roth. MIMSY: A system for analyzing time series data in the stock market domain. Master's thesis, University of Wisconsin, Department of Computer Science, 1993.
16. B. Shneiderman. Dynamic queries for visual information seeking. IEEE Software, 11(6):70-77, November 1994.
17. Silva and T. Catarci. Visualization of linear time-oriented data: a survey. *Proceedings of the first International Conference on Web Information Systems Engineering*, Hong Kong, June 2000. IEEE Computer Society.
18. Spotfire. http://www.spotfire.com.
19. E. Tufte. The visual display of quantitative information. Graphics Press, Cheshire, Connecticut. (1983).
20. M. Wattenberg. Sketching a graph to query a time series database. *Proceedings of the 2001 Conference Human Factors in Computing Systems*, Extended Abstracts, pages 381-382, Seattle WA, March 31- April 5, 2001. ACM Press.
21. UCI Knowledge Discovery in Databases Archive: *http://kdd.ics.uci.edu/*