Computational Method for Temporal Pattern Discovery in Biomedical Genomic Databases

Mohammed I. Rafiq, Martin J. O'Connor, and Amar K. Das
Stanford Medical Informatics, MSOB X233, Stanford, California 94305
{mirafiq, das}@stanford.edu

Abstract

With the rapid growth of biomedical research databases, opportunities for scientific inquiry have expanded quickly and led to a demand for computational methods that can extract biologically relevant patterns among vast amounts of data. A significant challenge is identifying temporal relationships among genotypic and clinical (phenotypic) data. Few software tools are available for such pattern matching, and they are not interoperable with existing databases. We are developing and validating a novel software method for temporal pattern discovery in biomedical genomics. In this paper, we present an efficient and flexible query algorithm (called TEMF) to extract statistical patterns from time-oriented relational databases. We show that TEMF—as an extension to our modular temporal querying application (Chronus II)—can express a wide range of complex temporal aggregations without the need for data processing in a statistical software package. We show the expressivity of TEMF using example queries from the Stanford HIV Database.

Keywords: *temporal pattern recognition, statistical aggregation, biomedical genomics, HIV genotype testing*

1. Time in Biomedical Databases

In the past decade, there has been unparalleled expansion in the volume, scope and complexity of research data accumulating through microarray studies and other high-throughput biological technologies. Databases are a prerequisite component of any biomedical research application that needs to maintain, integrate and share data. Relational databases, in particular, are employed in many large-scale bioinformatics projects. By archiving the results of completed experiments and observational studies, these relational databases serve as vital resources for ongoing data analysis by the scientific community. Although the variety of projects that use relational databases shows how flexible its simple tabular format is for data storage, users of relational databases are frequently limited in their ability to manipulate such two-dimensional representations for complex types of data analyses. Researchers must develop additional computational programs and user interfaces to support the semantics of relationships among study results that the relational database model cannot capture.

In this paper, we focus on the challenges of verifying temporal patterns among results stored in relational databases. Time stamps are prerequisite measurements in studies that involve repeated measurements and longitudinal observations; such values can be difficult to manage and query in large databases that store time-course data or patient histories. The ability to examine the temporal components of such data is central to the investigation of causal relationships and dynamic processes in biological research [1]. Global clustering methods of expression data are commonly used by investigators to identify gene or protein co-expression at a given time point [2]. Other methods [3-6] have been introduced to find time-delayed and periodic correlations that may be the result of one gene activating or controlling another gene downstream in a regulatory network. Although statistical methods to support time-course studies continue to be proposed and validated, database methods for such studies have not been concomitantly advanced to support the temporal dimensions of data resulting from microarray experiments.

Methods for the maintenance and querying over time-course relationships among biomedical data were introduced nearly thirty years ago, and there has been tremendous research since on the management and use of time-oriented computational methods in clinical medicine [7]. However, these methods have not been incorporated into the data-management and data-analytic tools that investigators use in genomic and other research databases. We propose that database methods that can support temporal data analysis will

provide researchers the essential ability to explore, visualize, and infer biologically relevant associations of a contextual, time-delayed, or periodic nature.

2. Genomic Database Example

We have collaborated closely with developers of the Stanford HIV Drug Resistance Database to advance the data-analytic needs of investigators studying HIV drug resistance, which remains a major obstacle to the successful treatment of HIV type 1 (HIV-1) infections. Genotypic resistance testing, which involves sequencing the molecular targets of anti-HIV therapy—the HIV reverse transcriptase (RT) and protease enzymes—is now routinely used by physicians before selecting antiretroviral drug regimen. However, the interpretation of these test results is difficult given uncertainties about how HIV-1 genotype relates to drug susceptibility and treatment history. To address this data-analytic problem, the HIV Drug Resistance Database group has developed an on-line relational database (http://hivdb.stanford.edu) to archive HIV RT and protease sequences [8]. The database (abbreviated HIVDB) currently contains over 2000 subjects with time-stamped data on drug regimens, HIV genetic sequence, and HIV viral load. Its schema is based on a temporal linkage of sequence changes in HIV RT and protease enzymes to antiretroviral drug histories of the individual from whom the isolate was obtained; drug susceptibility data on sequenced isolates when available; and clinical outcome data—plasma HIV RNA levels (or, viral load) and CD4+ cell counts. Figure 1 provides an integrated view of data for a subject in the database, and indicates the complexity of temporal data obtained from research and clinical cases.

Our work with the HIVDB users has found a common need for statistical data aggregation and temporal pattern extraction. An example of a typical query on the HIV database is to

> *Retrieve data to analyze statistical correlations between genotype and the virologic response to a regimen, using a 'baseline' viral load and viral loads at other time points of interest after the start of a new treatment regimen.*

Since data in the HIVDB comes from treatment settings, the timing of viral load measurements varies after during a treatment. As a result, HIVDB investigators may try to identify a measurement at particular time points after the start of a regimen (such as 8 weeks later) by retrieving the measurement which is temporally closest to that time point and that falls within a window (such as 4 to 12 weeks) around that time point. Alternatively, investigators may want the average value in the time window. This data processing requires *pattern matching* (finding all values within a time window) and *statistical aggregation* (finding the closest value to a midpoint or the average value over the time window). Expressing these temporal patterns in a database query language like SQL requires significant user effort and may lead to code that is at least a few hundred lines long. Statistical aggregation is not a feature of standard database technology, and is done typically through data processing with statistical software tools. Thus, current approaches to temporal querying frequently lead to code that is hard to reuse in data analysis. We have addressed these problems by building a temporal querying method (called Chronus II), which is efficient and expressive for defining temporal patterns found in biomedical database applications.

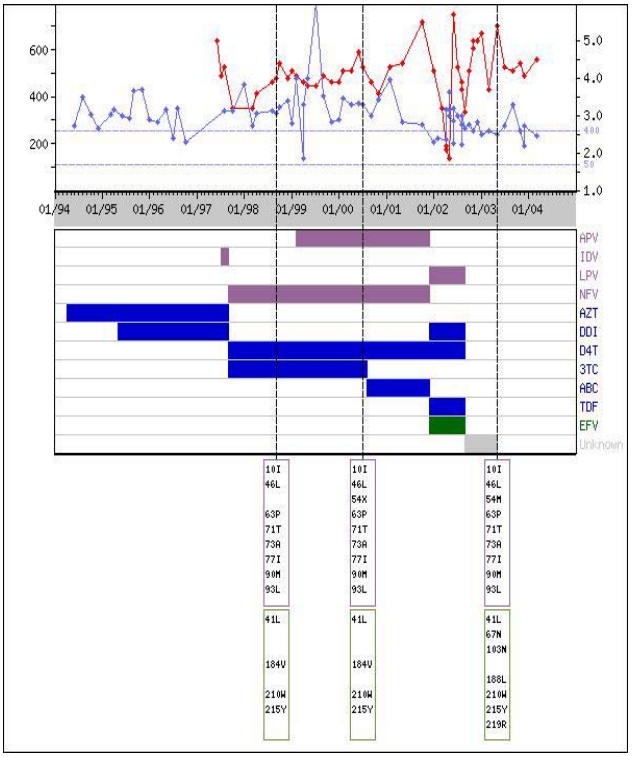

Figure 1. Longitudinal view of patient data from a single subject in the Stanford HIV Drug Resistance Database. The time graph indicates the complexity of inferring the relationship of clinical response—viral load (top line) and CD4 count (bottom line)—to mutations in HIV gene sequence (listed in bottom boxes) and treatment history (bars with drug name abbreviations).

3. Chronus II Querying Method

Chronus II [9] is a temporal querying method that has been designed to express complex queries over time-oriented values. It uses a querying language,

called Chronus Querying Language (CQL), and is based on the proposed temporal standard TSQL2 for the relational query language SQL, which has few temporal features. Chronus II is implemented in Java and is designed to operate as a modular program above existing relational databases using JDBC as its access layer. The Chronus II interface takes a CQL command, generates standard SQL statements for the non-temporal part of the command, and completes the processing of temporal operations in memory. An example temporal query in CQL might look like:

```
TEMPORAL SELECT P.Patient, P.Problem, D.Drug
FROM   PROBLEMS AS P, DRUGS AS
       D(Patient,Drug)
WHERE P.Patient=D.Patient
WHEN   DURATION(D, 'weeks') > 2 AND
       DURATION(START(P), 'now', 'years') < 1
       AND BEFORE(START(D), START(P))
```

As the example query shows, CQL syntax resembles standard SQL with additional clauses (in bold).The most significant addition is the WHEN clause. This clause is the temporal equivalent of the WHERE clause and supports specifications of temporal conditions, such as the DURATION and BEFORE operators. An equivalent SQL query would be far less concise, would need to be written as multiple queries, and would require temporary tables to hold intermediate results. Although Chronus II is well equipped to deal with queries relating to temporal patterns, it does not support statistical aggregate functions, like CLOSEST, INCREASING and DECREASING, over time windows that are needed by users of HIVDB. We have thus developed a computational method that adds such features to Chronus II.

4. Temporal EMF Algorithm

Large scale data analysis in relational databases typically requires specification of statistical aggregation functions, but the standard SQL query language only supports a small number of such functions, such as sum, count, minimum, and maximum. To develop a general computational method to summarize data over groups of time-stamped data and to formulate an arbitrarily complex aggregation directly in a database query, we use a proposed modification to SQL called extended multi-feature (EMF) syntax [10]. In our work, we have extended the EMF syntax to express temporal aggregation in a generalized and succinct manner. We have incorporated this extended EMF syntax into CQL, taking advantage of Chronus II's rich ability to handle time values. Our extension, called Temporal EMF (TEMF) required the following functionalities to be added to previous work on EMF:

i) Supporting multiple database tables in the FROM clause (the original EMF syntax only allows for a single table to be queried)
ii) Implementing the temporal equivalent of the basic aggregate operators available in SQL to specify arbitrarily complex temporal aggregate queries.
iii) Defining grouping variables over rows of data based on temporal conditions, such as duration operators, over stored time values.
iv) Allowing nested aggregation to be used in defining temporal aggregate queries. The nested aggregates can support combinations of both duration operators and basic temporal aggregate operators defined in (ii).

Our implementation of TEMF involves additional data processing in the Chronus II program. A table structure is created in memory for which the columns are and aggregates of each grouping variable. Chronus II parsing of a TEMF specification in a CQL query separates the two constituents, the CQL syntax and the EMF syntax. The CQL syntax is processed by Chronus II as before. The output of this query fills the rows of a memory-resident table structure (called mf-table) the rows of which correspond to the user-defined grouping attributes. Then, for each grouping variable defined in the query, we compare each row in the source table to each entry in the mf-table, and, if the defining condition of the grouping variable is satisfied, we update the aggregates of that grouping variable in the mf-table for that row. At the end of our computational algorithm, we make one scan through the mf-table and for each row which satisfies the HAVING condition, we output the corresponding data required in the SELECT clause.

Since the defining condition of a grouping variable may contain aggregates of previously defined variables, the order in which the variables are scanned is essential for the correctness of the algorithm. Hence, we undertake a dependency analysis [10] over the grouping variables as a first step. This involves running a topological sort algorithm, implemented using depth first search, on the grouping variables. Our algorithm requires one pass for each grouping variable. The dependency analysis can be used to reduce this number of passes by computing multiple variables in a single pass and by selecting those that do not depend on each other.

To illustrate the expressivity of our method, we provide the schemas of three tables of time-stamped data from HIVDB and an example TEMF extension to CQL to express the query example in Section 2.

```
TEMPORAL TABLE Treatment (ID, Regimen) AS VALID
STATE 'days';
TEMPORAL TABLE Sequence (ID, SeqName);
TEMPORAL TABLE RNA (ID, VLoad) AS VALID EVENT
'days';
```

```
TEMPORAL SELECT   T1.ID, T2.Regimen,
LAST(START(T2)) AS newRegimenStart,  LAST(FINISH(S))
AS baselineSequenceDate, LAST(VALID(V1)) AS
baselineVLoad, V2 .Vload, V3 .Vload
FROM     Treatment AS T1,  // pre baseline non-DDI regimen
         Treatment AS T2, // new DDI regimen
         Sequen AS S,     //  sequence data
         RNA AS V1,       //  viral loads  within 12 wks
         RNA AS V2,       // viral loads between 4 and 12 wks
         RNA AS V3     //  viral loads between 16 and 32 wks
WHERE    S.ID = T1.ID AND S.ID = T2.ID AND S.ID  = V1.ID
         AND T1.Regimen NOT LIKE '%DDI%' AND
         T2.Regimen = 'DDI' AND  V1.VLoad >= 500
WHEN      DURATION(T2, 'weeks') >= 12 AND
         BEFORE(S1,T2) AND DURATION(FINISH(S1),
         START(T2),'weeks') < 12 AND BEFORE(V1, T2)
         AND DURATION(V1, START(T2), 'weeks') < 12
         AND CONTAINS(PERIOD(START(T2) +
         'weeks(4)', START(T2)+'weeks(12)'), V2) AND
         CONTAINS(PERIOD(START(T2) + 'weeks(16)',
         START(T2)+'weeks(32)'), V3)
GROUP BY ID; X, Y
SUCH THAT X .ID = ID, Y .ID = ID
HAVING   MIN(DURATION (X .VALID (V2) ,
         X.FIRST(START(T2)) + 'weeks(8)'))  OR
         MIN(DURATION (Y .VALID (V3) ,
         Y.FIRST(START(T2)) + 'weeks(24)'))
```

This query finds patients who satisfy the following criteria: the patient was changed to a new regimen containing DDI, continued for at least 12 weeks, and was not previously on a regimen containing DDI; a viral load was measured within 12 weeks of the start of the new regimen and the value was higher then 500; an HIV genotype test was done less than 12 weeks prior to the new regimen. The TEMF specification (in bold) returns the viral load measurements, if available, that are closest to the midpoints of a 4-to-12 and a 16-to-32 week time window after start of the new regimen. The TEMF clauses could easily be changed to find the FURTHEST or the AVERAGE value within that window by changing the MIN operator to MAX or AVG, respectively. Users thus can select the appropriate type of temporally aggregated results to determine patterns of responses to treatment and how such outcomes correlate with mutations in genotype-test results present at the start of the treatment. The flexibility of the TEMF temporal aggregation method provides an essential querying feature that is lacking in the SQL language for relational databases.

5. Discussion

As the number of research data grows rapidly, investigators urgently need computational methods that can enable them to manage the complexity of such data, to learn from those data, and to advance scientific knowledge as a result. Since biological phenomena are inherently dynamic in nature, many researchers need pattern recognition methods that can identify important temporal relationships among large-scale data sets. In this paper, we have provided a novel method for temporal aggregation and shown its applicability for a particular biomedical genomic database. Our method gives users an advantage over using separate software for temporal data analysis by allowing them formulate and modify statistical aggregate functions in a few lines of query syntax. Investigators can thus undertake data exploration by specifying aggregations over time-stamped values at the same time they are querying the primary data results. Such flexibility may hasten advancements in research through more rapid discoveries of previously hidden temporal relationships in genomic and other biomedical research databases.

6. References

[1] Nicholson JK, Holmes E, Lindon JC, Wilson ID. The challenges of modeling mammalian biocomplexity. Nat Biotechnol 2004; 22:1268-74.

[2] Svrakic NM, Nesic O, Dasu MR, Herndon D, Perez-Polo JR. Statistical approach to DNA chip analysis. Recent Prog Horm Res 2003; 58:75-93.

[3] Qian J, Dolled-Filhart M, Lin J, Yu H, Gerstein M. Beyond synexpression relationships: local clustering of time-shifted and inverted gene expression profiles identifies new, biologically relevant interactions. Mol Biol 2001; 314:1053-1066.

[4] Park T, Yi S-G, Lee S, Lee SY, Yoo D-H, Ahn J-K, Lee Y-S. Statistical tests for identifying differentially expressed genes in time-course microarray experiments. Bioinformatics 2003; 19: 694-703.

[5] Guo X, Qi H, Verfaille CM, Pan W. Statistical significance analysis of longitudinal gene expression data. Bioinformatics 2003; 10: 1628-1635.

[6] Wichert S, Fokianos K, Strimmer K, Identifying periodically expressed transcripts in microarray time series data. Bioinformatics 2004; 20: 5-20.

[7] Combi C, Shahar Y. Temporal reasoning and temporal data maintenance in medicine: issues and challenges. Comput Biol Med 1997; 27: 353-368.

[8] Kuiken, C., Korber, B., and Shafer R.W. HIV sequence databases. AIDS Rev 2003; 5: 56-65.

[9] O'Connor MJ, Tu SW, Musen MA. The Chronus II temporal database mediator. Proc AMIA Annual Symp 2002, pp. 567-571.

[10] Chatziantoniou, D. The PanQ tool and EMF SQL for complex data management. International Conference on Knowledge Discovery and Data Mining (KDD-99), pp. 420-42.