Approximate Query Evaluation Using Linear Constraint Databases*

Peter Revesz Rui Chen Min Ouyang
Computer Science and Engineering Department
University of Nebraska-Lincoln
Lincoln, NE68588, USA

Abstract

This paper shows that constraint databases *can be used for the approximation of several types of discretely recorded continuous data, for example* time series *data and some* spatio-temporal geographic *data. We show that time series data can be approximated by a piecewise linear approximation that runs in linear time in the number of data points, and the piecewise linear approximation can be represented in a linear constraint database. Similarly, the spatio-temporal geographic data that is composed of a set of spatial locations, where each location is associated with a time series, can be also approximated and represented in a linear constraint database. The approximations provide* data compression, faster query evaluation — *that preserve high* precision *and* recall — *and* interpolation *enabling the evaluation of queries that could not be evaluated before.*

1. Introduction

Databases often record continuous data at discrete time instances, for example, *time series* and *geographic time series* data.

A *time series* is a sequence of data points $(t_1, y_1), \cdots, (t_n, y_n)$ where each t_i is a time instance and y_i is the value of a measured attribute at time t_i, and the ts are monotone increasing. A *geographic time series* is a set of location and time series pairs. It can be represented as a set of ordered triples (l_j, t_i, y_i) where l_j is a location and (t_i, y_i) is a time series data point associated with location l_j.

Suppose that we record the daily high temperature at location 1 on Mondays and Fridays, at location 2 on Tuesdays and Thursdays, at location 3 on Saturdays, and at location 4 on Sundays. This may yield a geographic time series data that can be represented for a month, that starts on a Monday, as shown in Table 1.

*This research was supported in part by NSF grants IRI-9625055 and IRI-9632871 and by a Gallup Research Professorship.

The geographic time series data is not convenient for querying for several reasons. Suppose we would like to get the daily high temperature for each station on day 10 using the following straightforward SQL query:

$$\begin{array}{ll} \textbf{select} & Temp \\ \textbf{from} & Temperature \\ \textbf{where} & t = 10 \end{array} \quad (1)$$

This query would return an empty set because at none of the four weather stations was the daily high temperature recorded on day 10.

Recognizing such problems, we suggested in [3] to represent time series data by a *piecewise linear approximation* which in turn can be easily stored in a *linear constraint database* [8]. For example, the above geographic time series can be represented by the linear constraint database in Table 2.

The approximation has the following advantages:

Interpolation: Queries such as (1) can be evaluated using the linear constraint representation, which provides an interpolation for the time series data for each day of the month.

Data Compression: The number of pieces in an approximation is usually much fewer than the number of points in the time series. For example, Table 1 has 26 tuples, while Table 2 has only 8 tuples.

Query Evaluation Efficiency: With fewer number of tuples, some queries can be evaluated faster.

Piecewise linear approximation is studied in computational geometry and image processing. Some "optimal" algorithms have been proposed that return a solution with the minimum number of possible pieces. In 1991, Hakimi and Schmeichel [7] gave an $O(n^2)$ optimal piecewise linear approximation algorithm, where n is the number of time series data points. Recently, Agarwal and Varadarajan [2] gave an improved algorithm that requires $O(n^{\frac{4}{3}+\delta})$ time, where δ is any arbitrarily small positive constant. Both of these algorithms run slowly for large data sets. In addition, they

SN	t	Temp
1	1	68
1	5	71
1	8	76
1	12	73
1	15	71
1	19	68
1	22	70
1	26	74
1	29	80
2	2	71
2	4	69
2	9	68
2	11	66
2	16	70
2	18	68
2	23	71
2	25	75
2	30	77
3	6	75
3	13	78
3	20	72
3	27	68
4	7	72
4	14	75
4	21	72
4	28	74

Table 1. The Temperature Relation

generate solutions that are not easy to update. These disadvantages make the algorithms very inefficient for large databases when the data changes frequently.

We develop a new piecewise linear approximation algorithm that is not optimal in the number of pieces (usually returns about 10-20 percent more pieces) but runs in $O(n)$ time. This improves our earlier algorithm in [3] that required $O(n^2)$ time in the worst case. We also show that under some reasonable assumptions, the data compression of the new algorithm is proportional to Ψ^2, where Ψ is the error tolerance value. We also extend the piecewise linear approximation to geographic time series data, that is, we use spatio-temporal approximation.

Geographic and spatio-temporal objects are considered within the European Chorochronos project [5], the Moving Objects Spatio-Temporal model (MOST) [11, 13], and object-relational databases such as Oracle8i [10] and PostgreSQL [9]. Grumbach et. al. [6] also propose an alternative data model to interpolate spatio-temporal data.

The rest of the paper is structured as follows. Section 2 describes our linear time piecewise linear approximation algorithm for time series data. We also give in this section an

SN	t	Temp	
1	t	f	$f = 68 + \frac{5}{11}(t-1),\ 1 \le t \le 12.$
1	t	f	$f = 73 - \frac{3}{10}(t-12),\ 12 \le t \le 22.$
1	t	f	$f = 70 + \frac{10}{7}(t-22),\ 22 \le t \le 30.$
2	t	f	$f = 71 - \frac{3}{16}(t-2),\ 1 \le t \le 18.$
2	t	f	$f = 68 + \frac{3}{4}(t-18),\ 18 \le t \le 30.$
3	t	f	$f = 75 - \frac{3}{14}(t-6),\ 1 \le t \le 20.$
3	t	f	$f = 72 - \frac{4}{7}(t-20),\ 20 \le t \le 30.$
4	t	f	$f = 72 + \frac{2}{21}(t-7),\ 1 \le t \le 30.$

Table 2. The Constraint Representation of the Temperature Relation

algorithm that efficiently updates the piecewise linear approximation in the case of inserting new time series data points. We also analyse, in terms of the error tolerance value, the data compression that can be achieved using our piecewise linear approximation. Section 3 describes our approximation algorithm for geographic time series data. Section 4 discusses the approximate query evaluation based on the piecewise linear approximation and gives some experimental results on precision and recall. Finally, Section 5 discusses future work.

2. Approximation of Time Series Data

2.1. A Piecewise Linear Approximation Algorithm

The piecewise linear approximation problem of a time series S given some error tolerance value Ψ is the problem of finding a piecewise linear function f such that the endpoints of each piece of f occur in S and the following holds:

$$|f(t_i) - y_i| \le \Psi \text{ for each } (t_i, y_i) \in S. \tag{2}$$

For example, Table 2 is a piecewise linear approximation of Table 1 with $\Psi = 5$.

Definition 2.1 For any two pairs of points (t_b, y_b) and (t_e, y_e) where $(b < e)$, we denote by $Y_{b,e}$ the line segment connecting them. For any given error tolerance value Ψ we also define a *lower line* $L_{b,e}$ and an *upper line* $U_{b,e}$ between the two data points as shown in Figure 1.

Note that $Y_{b,e}$ could be used as a linear approximation for the sequence of data points between (t_b, y_b) and (t_e, y_e). That would introduce at each time instance t_i for $b < i < e$ some interpolation error $\psi_{b,e}(t_i)$, where

$$\psi_{b,e}(t_i) = |Y_{b,e}(t_i) - y_i|\,. \tag{3}$$

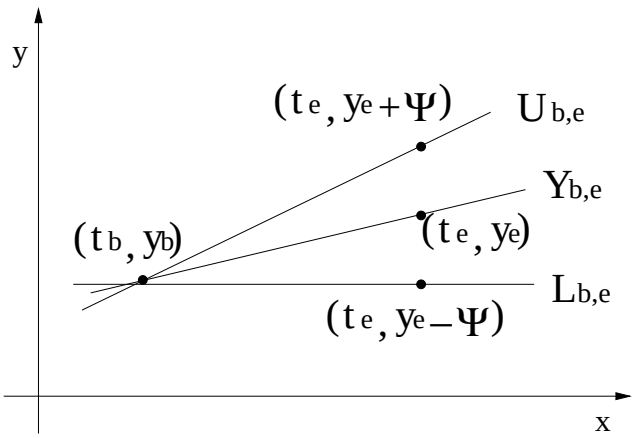

Figure 1. Lines $Y_{b,e}$, $L_{b,e}$ and $U_{b,e}$.

We propose below an $O(n)$ time algorithm that finds a piecewise linear function that approximates any time series data S.

PIECEWISE LINEAR APPROXIMATION ALGORITHM:

Input: A time series $(t_1, y_1), \ldots, (t_n, y_n)$.
Ψ the error tolerance value.
Output: A piecewise linear function.
Local Vars: The b and e are integer variables such that $Y_{b,e-1}$ satisfies Condition (2).
L and U stand for two lines.

Let $L := L_{1,2}$ and $U := U_{1,2}$.
Let $b := 1,\ e := 3$

while $e \leq n$ **do**
  Construct the lines $L_{b,e}$ and $U_{b,e}$
  if $L(t_e) < L_{b,e}(t_e)$ **then** $L := L_{b,e}$ **end-if**
  if $U(t_e) > U_{b,e}(t_e)$ **then** $U := U_{b,e}$ **end-if**
  if $L(t_e) > U(t_e)$ **then**
    Create a piece $Y_{b,e-1}$
    Let $b := e - 1$
    Let $L := L_{b,e}$ and $U := U_{b,e}$
  end-if
  $e := e + 1$
end-while
Create a piece $Y_{b,e-1}$.

First, we prove the correctness of our approximation algorithm.

Theorem 2.1 The piecewise linear approximation algorithm is correct for any error tolerance value Ψ and time series S. ■

Next, we analyze the computational complexity.

Theorem 2.2 The computational complexity of the piecewise linear approximation algorithm is $O(n)$ time, where n is the number of points in the time series. ■

Now we analyze the expected number of points in a piece, assuming that the time series $(t_1, y_1), \cdots, (t_n, y_n)$ satisfies the following property, for some constant M and for each $1 < i \leq n$

$$\begin{cases} y_1 = 0 \\ Prob(y_i - y_{i-1} = M) = 0.5 \\ Prob(y_i - y_{i-1} = -M) = 0.5 \end{cases} \quad (4)$$

For example, consider the time series which starts with $(0, 0)$ and records, at each later time when a coin is flipped, the number of heads minus the number of tails seen since the beginning. This time series satisfies Property (4) with $M = 1$ if heads and tails have the same probability.

As another example, the daily temperature could be described by a time series that satisfies Property (4), if we use a thermometer in which the adjacent scales are M Fahrenheit degrees apart instead of the usual single Fahrenheit degrees, where M is the largest daily change, and if we record only on those days when there is a change in temperature according to the rougher thermometer.

Let $E(\Psi, M)$ be the expected number of original points spanned by a single piece of the piecewise linear approximation, including the two endpoints, when the approximation uses the tolerance Ψ and the time series satisfies Property (4). We can prove the following.

Theorem 2.3 If a time series satisfies Property (4), then

$$E(\Psi, M) \geq \left(\left\lfloor \frac{\Psi}{2M} \right\rfloor + 1 \right)^2.$$

■

For example, for the coin flipping time series when $\Psi = 6$ each piece of the piecewise linear approximation function is expected to span at least 16 original time series points.

2.2. Updating Piecewise Linear Functions

In this section we consider what happens if we approximated by a constraint relation some time series and the user requests an insertion or deletion of a point in the time series. Note that the user can request insertions and deletions of time series points (i.e., tuples of the relational database) and not the constraint database, because the constraint database representation is hidden from the user.

If the user requests a deletion of a time series data point, then the request can be ignored because the approximation

function still satisfies the error tolerance for the remaining points.

The insertion of points is much more complex. In this case, we have to update the piecewise linear function. Consider Figure 2. In (1) the original piecewise linear function is shown by a solid line and the edges of the error tolerance range are shown by dashed lines. In (2) the point P_1 is to be inserted, but the piecewise linear function is not changed since P_1 is within the error tolerance range. In (3) the point P_2 is to be inserted, and the piecewise linear function is updated by splitting the middle piece into two pieces. In (4) the point P_3 is to be inserted, and the piecewise linear function is updated by splitting the third piece into two pieces.

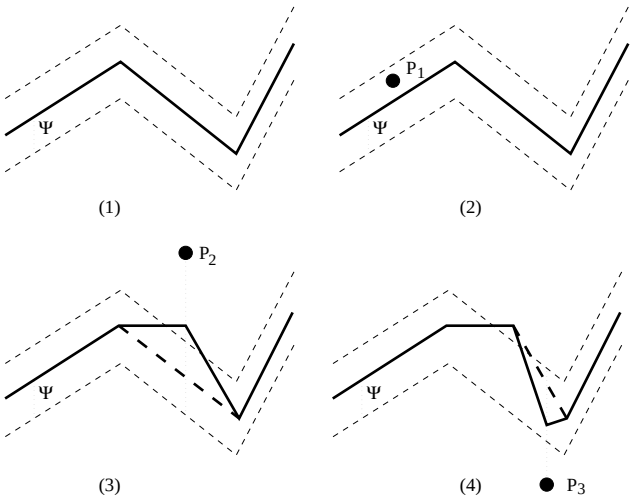

Figure 2. Inserting Points into a Piecewise Linear Function

Note that a piecewise linear approximation can be represented as a set of endpoints of the pieces. We also add a Boolean tag o to each point. The tag will be *true* if it is a point that is an original point, otherwise it is a point which was inserted and the tag will be *false*. This allows us to reconstruct from any updated piecewise linear function f, the original piecewise linear function, denoted f_o, as the sequence of points with the *true* tags. Finally, we assume that for no point (t_α, y_α) to be inserted is there already a point with time t_α.

Theorem 2.4 Suppose that a time series S is approximated by a piecewise linear function f_o with n "pieces" and Ψ error tolerance. Then any set I of m insertions such that each insertion point is at most some constant $\delta \geq \Psi$ distance from f_o can be done by the insertion algorithm in $O(m \log n)$ time such that the updated piecewise linear function f has at most $n + m$ "pieces" and the following

INSERTION ALGORITHM:

Input: A piecewise linear function f represented as a sequence of points $(t_1, y_1, o_1), \ldots, (t_n, y_n, o_n)$.
Ψ the error tolerance value.
(t_α, y_α) the point to be inserted.
Output: An updated piecewise linear function.

if $t_\alpha < t_1$ **then**
  Add $(t_\alpha, y_\alpha, false)$ as the first point in f.
else if $t_\alpha > t_n$ **then**
  Add $(t_\alpha, y_\alpha, false)$ as the last point in f.
else if $f_o(t_\alpha) - \Psi > y_\alpha$ **then**
  Add $(t_\alpha, \frac{1}{2}((f_o(t_\alpha) + \Psi) + y_\alpha), false)$
  between points with times $t_i < t_\alpha$ and $t_{i+1} > t_\alpha$.
else if $f_o(t_\alpha) + \Psi < y_\alpha$ **then**
  Add $(t_\alpha, \frac{1}{2}((f_o(t_\alpha) - \Psi) + y_\alpha), false)$
  between points with times $t_i < t_\alpha$ and $t_{i+1} > t_\alpha$.
end-if

holds.

$$|f(t_i) - y_i| \leq \frac{\Psi + \delta}{2} \text{ for each } (t_i, y_i) \in S \cup I.$$

■

For example, if $\delta = 3\Psi$, then the error tolerance for the updated piecewise linear approximation will be 2Ψ for all original and newly inserted data points.

3. Approximation of Geographic Time Series Data

Geographic data is usually represented using either the raster or the vector data models. *Triangulated Irregular Networks* or (TINs) and *Voronoi diagrams* are example data structures that are used in the vector data model [1, 4, 12].

A TIN can be created for any given set of locations in the plane, and ARC/INFO, the most popular GIS software, supplies several functions to generate TINs. Given a geographic time series data set $S = \{(P_1, S_1), (P_2, S_2), \cdots, (P_n, S_n)\}$, where the P_is are the 2D locations of the form (x_i, y_i) and each S_i is a time series corresponding to P_i, we can use the TIN generation algorithm to transform the set of spatial points $\{(x_1, y_1, 0), (x_2, y_2, 0), \cdots, (x_n, y_n, 0)\}$ into a TIN structure. Then we convert each TIN triangle to a linear constraint tuple. For each triangle with vertices P_i, P_j and P_k, we approximate the temporal value of this triangle for each time instance by Equation (5) below.

Definition 3.1 Given a triangle with points P_i, P_j and P_k, and their corresponding series of temporal data S_i, S_j and S_k. The series of the temporal data of the triangle S_{ijk} is defined as follows:

$$S_{ijk}(t) = \frac{1}{3}(S_i(t) + S_j(t) + S_k(t)) \tag{5}$$

This definition is motivated by the following. In each TIN triangle, we have a plane within 3D that passes through the vertices P_i, P_j and P_k, that are associated with the time series S_i, S_j and S_k. The time series give the z coordinates of the three corner vertices. Hence the centroid of the triangle is a time series as defined in Equation (5). Further, the time series of the centroid of each triangle can be transformed into a piecewise linear function and a linear constraint tuple.

Alternatively, instead of a TIN, we can construct a Voronoi diagram for the P_i locations. For each cell of the Voronoi diagram containing point P_i we use S_i to approximate the attribute of interest for each point within the whole cell. This can be also represented in a linear constraint database.

4. Approximate Query Evaluation Methods

Instead of querying the original (geographic) time series data, we can query their approximations. Since the approximations are linear constraint relations, we can use several recent linear constraint database systems.

In this section, we give some sample SQL queries and compare their regular and approximate query evaluations. We compare the number of tuples, the precision, and the recall of the output relations. *Precision* is the percent of the tuples that are relevant out of those retrieved, and *recall* is the percent of the relevant tuples retrieved. Because the approximation is equivalent to an infinite number of regular tuples, in the calculation of precision and recall, we only counted tuples with integer time values.

4.1. Approximate Evaluation for Time Series Data

We did some experiments using a temporal dataset that contained the daily high temperature and the daily low temperature between January 1, 1987 and December 31, 1996 for weather station number 252820 in Nebraska. We used different Ψ values to restrict the piecewise linear transformation. We obtained the weather data from the website of the National Climatic Data Center at `http://www.ncdc.noaa.gov` and smoothened the data using the running window method with window size 7. Table 4.1 gives some data compression characteristics of the modified data set using different values of Ψ.

Ψ	#Pieces in High Temperature Dataset	# Pieces in Low Temperature Dataset
—	3,653	3,653
1.0	1,426	1,084
2.0	790	594
4.0	428	335
8.0	197	140

Table 3. Number of Pieces in Datasets

Let $R_1(day,\ high\_temp)$ be the daily high temperature relation, and $R_2(day,\ low\_temp)$ be the daily low temperature relation, and let $R(day,\ high\_temp,\ low\_temp)$ be the join of R_1 and R_2.

The following are some query evaluation tests for SQL queries using relation R. In these tests, we evaluated the relational queries in ORACLE and the approximate queries (first translated to a Datalog) in the MLPQ constraint database system. After the MLPQ evaluation, we converted the output constraint relation to pairs of days, where the values of the days were restricted to be integers.

Example 4.1 Find all pairs of days such that for each the high temperature in one day is greater than or equal to that in the other. The SQL query is as follows and the test results are shown in Table 4.

$$\begin{array}{ll} \textbf{select} & R1.day,\ R2.day \\ \textbf{from} & R \textbf{ as } R1,\ R \textbf{ as } R2 \\ \textbf{where} & R1.high\_temp > R2.high\_temp \end{array} \tag{6}$$

Ψ	MLPQ Constraints	Precision	Recall
—	6,645,646	100.00%	100.00%
1.0	2,006,001	99.39%	99.49%
2.0	1,208,010	98.54%	98.67%
4.0	235,639	96.83%	96.97%
8.0	51,681	93.39%	93.53%

Table 4. $R_1.high\_temp > R2.high\_temp$**.**

Example 4.2 Find all pairs of days such that for each the high temperature in one day is greater than or equal to that in the other and the low temperature in one day is also greater than or equal to that in the other. The SQL query is as follows and the test results are shown in Table 5.

$$\begin{array}{ll} \textbf{select} & R1.day,\ R2.day \\ \textbf{from} & R \textbf{ as } R1,\ R \textbf{ as } R2 \\ \textbf{where} & R1.high\_temp > R2.high\_temp \\ \textbf{and} & R1.low\_temp > R2.low\_temp \end{array} \tag{7}$$

Ψ	MLPQ Constraints	Precision	Recall
—	6,091,441	100.00%	100.00%
1.0	1,836,631	99.30%	99.44%
2.0	639,797	98.40%	98.66%
4.0	215,649	96.45%	96.67%
8.0	46,449	92.72%	92.68%

Table 5. $R1.high\_temp > R2.high\_temp$ **and** $R1.low\_temp > R2.low\_temp$**.**

4.2. Approximate Evaluation of Geographic Time Series Data

Approximation can be used in many spatio-temporal applications. Suppose there are several weather stations in some region, and each weather station records a series of precipitation values for each month. For each station, we can approximate its precipitation by a piecewise linear function. We construct the Voronoi diagram for these stations, and for each cell of the diagram we use the precipitation for the station within the cell to approximate the precipitation value of the whole cell.

Example 4.3 We define the relations $Precipitation(id, prec, month)$ and $Voronoi(id, cell)$, where id is the station number, $prec$ is the precipitation at the month $month$, and $cell$ is the Voronoi cell for station id. Suppose that we would like to find for month 47 the drought areas, i.e., the areas where the precipitation is lower than 0.5 inches. This can be expressed by the following SQL query:

$$\begin{array}{ll} \textbf{select} & cell \\ \textbf{from} & Precipitation,\ Voronoi \\ \textbf{where} & Precipitation.id = Voronoi.id \textbf{ and} \\ & month = 47 \textbf{ and } prec < 0.5 \end{array} \quad (8)$$

5. Conclusion and Future Works

This work raises the possibility of using approximate query evaluation for large (geographic) time series data. Remaining open problems include finding an optimal $O(n)$ time approximation algorithm, and generalizing the query evaluation method to queries with negation.

References

[1] N. Adam and A. Gangopadhyay. *Database Issues in Geographic Information Systems*. Kluwer Academic Publishers, 1997.

[2] P. K. Agarwal and K. R. Varadarajan. Efficient Algorithms for Approximating Polygonal Chains. *Journal of Discrete & Computational Geometry*, 23:273–291, 2000.

[3] R. Chen, M. Ouyang, and P. Revesz. Approximating Data in Constraint Databases. In *Proc. the 4th International Symposium on Abstraction, Reformulation and Approximation, HorseshoeBay, Texas*, volume 1864 of *Lecture Notes in Artificial Intelligence*, pages 124–143. Springer-Verlag, July 2000.

[4] B. Dent. *Cartography Thematic Map Design*. McGraw-Hill, 1999.

[5] A. Frank, S. Grumbach, R. Guting, and et.al. Chorochronos: A Research Network for Spatiotemporal Database Systems. *SIGMOD Record*, 28(3):12–21, 1999.

[6] S. Grumbach, P. Rigaux, and L. Segoufin. Manipulating Interpolated Data is Easier than You Thought. In *Proc. International Conference on Very Large Databases*, 2000.

[7] S. L. Hakimi and E. F. Schmeichel. Fitting Polygonal Functions to A Set of Points in the Plane. In *CVGIP: Graph. Mod. Image Proc.*, pages 132–136, 1991.

[8] P. Kanellakis, G. Kuper, and P. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51:26–52, 1995.

[9] B. Momjian. PostgreSQL, Introduction and Concepts. Addison Wesley, 2000.

[10] J. Sharma. Oracle8i Spatial: Experiences with Extensible Databases. In *An Oracle Technical White Paper*, May 1999.

[11] A. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and Querying Moving Objects. In *Proc. International Conference on Data Engineering*, pages 422–433, 1997.

[12] J. Star and J. Estes. *Geographic Information Systems: An Introduction*. Prentice-Hall, Inc., 1989.

[13] O. Wolfson, A. Sistla, B. Xu, J. Zhou, and S. Chamberlain. DOMINO: Databases fOr MovINg Objects tracking. In *Proc. ACM SIGMOD Symp. on the Management of Data*, pages 547–549, 1999.