Data Mining with Calendar Attributes

H.J. Hamilton and D.J. Randall
Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
{hamilton,randal}@cs.uregina.ca

ABSTRACT

This paper addresses the problem of data mining from temporal data based on calendar (date and time) attributes. The proposed methods uses a probabilistic domain generalization graph, i.e., a graph defining a partial order that represents a set of generalization relations for an attribute, with an associated probability distribution for the values in the domain represented by each of its nodes. We specify the components of a domain generalization graph suited to calendar attributes and define granularity, subset, lookup, and algorithmic methods for specifying generalizations between calendar domains. We provide a means of specifying distributions. We show how the calendar DGG can be applied to a data mining problem to produce a list of summaries ranked according to an interest measure given assumed probability distributions.

1. Introduction

Temporal and spatial attributes are common in relational and object-oriented databases. A ***temporal attribute*** might be used to specify the beginning of an event, such as a *birth date* to mark the start of a new life, or the ending of an event, such as the *check out time* from a hotel, or the duration of an event, such as the *elapsed time* of a race. A ***spatial attribute*** might be used to specify a geographical location or a coordinate of an object in an arbitrary space.

Probabilistic domain generalization graphs can be used to guide the data mining process in the presence of domain knowledge about temporal and spatial attributes. Informally, a ***domain generalization graph*** can be thought of as a graph showing possible generalizations as paths through a graph. A formal definition is given in [9]. Each node in the graph corresponds to a domain of values. Each link in the graph corresponds to a generalization relation, which is an mapping from the values in the domain of the initial node to the final node of the link. Every domain generalization graph has a single source node, called the ***bottom***, corresponding to the most specific domain of values, which are typically the base data values present in the database. Every domain generalization graph also has a single sink node, called the ***top***, corresponding to a domain called *Any* with only one value. The bottom node is the ancestor of all nodes in the domain generalization graph, so unlike in trees, the child nodes are above the parent nodes.

Domain generalization graphs are appropriate for summarization data mining, in which interesting summaries of all or part of the data are sought [9]. The act of creating a summary is a form of generalization. Multiple possible paths of generalization are conveniently represented and manipulated using a domain

generalization graph, because the set of paths through the graph corresponds to the set of possible generalizations that can be defined using the generalization relations. Wherever values at different levels of granularity are related by generalization relationships, domain generalization graphs should be considered as a possible representation.

In a probabilistic domain generalization graph, a description of a probability distribution is associated with each node in the graph to describe the expected distribution of the values in the domain. For example, if the domain of values is the names of the countries of the world and expectations are based on population, the distribution could be specified by giving each country's name associated with the ratio of that country's population to the world population.

A ***calendar attribute*** is one whose domain contains date and time values, such as birth dates and check out times, while a ***duration attribute*** is one containing durations [14]. A noteworthy problem in developing techniques for generalizing and presenting temporal data is that time can be represented in many ways depending on the context. Temporal values can be generalized in different ways and to different levels of granularity [11]. For example, if the calendar attribute in the data specifies login times, its values can be generalized to give the number of logins in (1) each hour (of each day) or (2) each of the seven possible days of the week. Choosing different levels of granularity for the first way can give values generalized to the day, the month, or the year. We present a revised domain generalization graph for calendar attributes by explicitly identifying the domains appropriate to the relevant levels of temporal granularity and the mappings between the values in these domains; an earlier version of this graph was presented in [15].

Here, we consider the problem of data mining in the presence of calendar attributes in relational databases using probabilistic domain generalization graphs. We use domain generalization graphs because they provide a graphical structure that can be used as both a navigational aid by the user and as a guide to heuristic data mining procedures. The problem addressed in this paper is as follows: given a probabilistic DGG for a calendar attribute and a relation containing values for that attribute, create a list of generalized relations (summaries) at various levels of generality ranked according to a specified numerical interest measure. Although only the highest ranked summaries are displayed, our method creates all distinct, nontrivial summaries that can be prepared. To create a summary, we perform generalization by transforming values in one domain to another, according to directed arcs in the domain generalization graph.

The remainder of this paper is organized as follows. In the following section, we review domain generalization graphs and present a particular graph for calendar attributes. We also describe four methods for generalizing temporal data for a calendar attribute. In Section 3, we relate our work to other recent research. In Section 4, we describe probabilistic domain generalization graphs and explain how probability distributions are attached to the nodes in such graphs. In Section 5, we describe an application of our methodology to a sample data set. Finally, in Section 6, we present our conclusions.

2. Generalizing Calendar Values

A *concept hierarchy* associated with an attribute in a database is represented as a tree, where leaf nodes correspond to the actual data values in the database, intermediate nodes correspond to more general representations of the data values, and the root node corresponds to the most general representation of the data values. Knowledge about higher-level concepts can be discovered through a series of transformations of specific values to more general values. search beginning at the leaf nodes using a process called *attribute-oriented generalization* [3,8]. If several concept hierarchies are associated with the same attribute, meaningful knowledge about the attribute can be expressed in different ways. Common attribute-oriented generalization methods require the user to select one concept hierarchy, ignoring the relative merits of other possible generalizations that could produce interesting results. To facilitate other possible generalizations, *domain generalization graphs* (DGGs) have recently been proposed [7,9]. A concept hierarchy is represented in a domain generalization graph as a sequence of nodes, with one node for each level of the concept hierarchy.

We now describe our calendar DGG for calendar attributes, a part of which is shown in Figure 1. This DGG is a refined version of one presented in [14]. In Figure 1, the node labelled *YYYYMMDDhhmmss* represents the most specific domain considered, i.e., the finest granularity of our calendar domain is one second. Higher-level nodes represent generalizations of this domain. The arcs between nodes represent generalization relations. To handle data with calendar values specified to finer granularity, e.g., microseconds, more specific nodes could be added to the DGG.

The calendar DGG can be used to guide the generalization of calendar data into higher-level concepts. In knowledge discovery, each traversal of an arc in the DGG corresponds to generalizing a set of data from the domain indicated by the initial node of the arc to the domain indicated by the terminal node of the arc. For example, we generalize from *YYYYMMDDhhmmss* to *YYYYMMDDhhmm* by removing the second information from the calendar attribute. Among the more complicated are the contrast between calendar months and lunar months. The arcs leaving the node labelled *YYYYMM* indicate that data can be generalized to the higher-level concept of *MM* (calendar month). For the *MM* concept we generalize a set of 28, 29, 30, or 31 days into a single calendar month. A *lunar-month* generalizes a set of 29 or 30 days into one lunar month as required to closely follow the actual lunar cycle. While a calendar month always starts at the beginning of a month (that is, DD = 01), a lunar month does not. Thus a *lunar-month* must be generalized from the node labelled *YYYYMMDD*. When a new representation is required in the calendar domain, this DGG can be extended by adding new nodes and arcs and by defining new generalization relations associated with the arcs.

Four types of generalization relations are associated with the arcs in a calendar DGG: granularity, subset, lookup, and algorithmic. For *granularity generalization*, we assume that *YYYYMMDDhhmmss* can be represented as six subattributes (*YYYY*, *MM*, *DD*, *hh*, *mm*, *ss*). We generalize by suppressing subattributes of the *YYYYMMDDhhmmss* attribute from least significant (*ss*) to most (*YYYY*). For example, to generalize from *YYYYMMDDhhmmss* to *YYYYMMDD*, the *ss*, *mm*, and *hh* subattributes are discarded. All five domains this creates are shown in Figure 1.

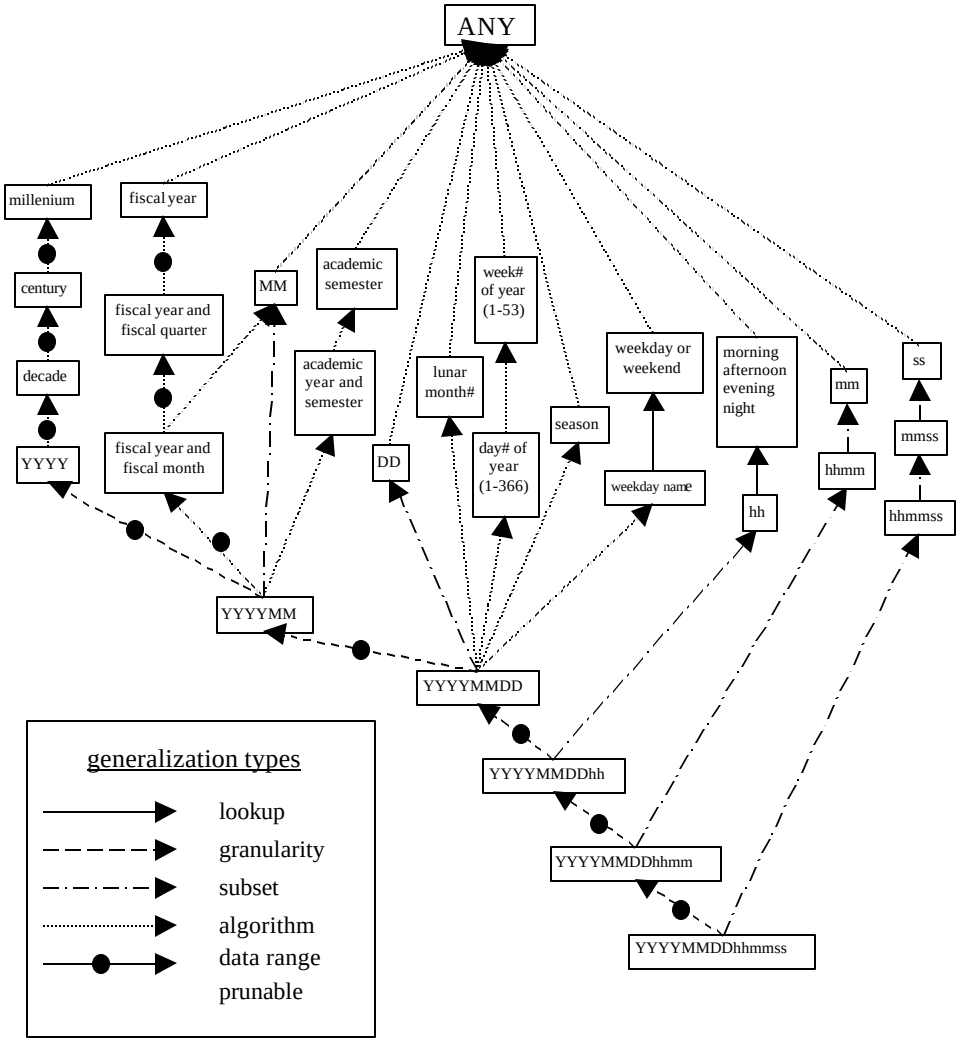

Figure 1. Domain Generalization Graph for a Calendar Attribute

In *subset generalization*, granularity generalization is extended so that any combination of the subattributes in *YYYYMMDDhhmmss* can be discarded. To generalize *YYYYMMDDhhmmss* to the periodic intervals of node *hh* (which could be used to discover that something happens every evening sometime in the hour starting at 9:00 pm), the *YYYY*, *MM*, *DD*, *mm*, and *ss* subattributes are discarded. The remaining subattributes do not need to be adjacent. This flexibility allows us to generalize to a node such as *MMhh*, which causes values to be categorized by both the month and hour. For simplicity, only a few of the domains that can be created by subset generalization are shown in Figure 1. Because subset generalization subsumes

granularity generalization, the latter is treated as a special case of the former in our implementation. Nonetheless, we find it a useful distinction at the conceptual level.

In *lookup generalization*, the domain of the higher-level concepts is specified in a table. To generalize the calendar attribute with this method, the lower-level concept is looked up in the table to obtain the corresponding higher-level concept. For example, to define a generalization from *day of week* to *weekday or weekend*, we list all weekday values along with their generalizations in a table. To generalize a value we search the table for the day of the week and select the corresponding weekday or weekend value.

In *algorithmic generalization*, an algorithm is used to generalize a value from one level of generality to another. Although algorithm generalization subsumes the three previous types, we find the distinction useful and only refer to algorithm generalization when none of the others is applicable. For example, to determine whether a particular year is a leap year, or to determine the day of the week on which someone was born, a lookup table would be extremely large, while algorithm generalization allows convenient expression.

Additionally, it can be valuable to duplicate the calendar attribute in a relation so that it can be generalized to multiple nodes in the DGG simultaneously. An example of this is comparing the number of instances of an event per hour on weekdays versus weekends. The calendar attribute is duplicated and one of the copies is generalized to *weekday or weekend* while the other is generalized to *hh*. This technique reduces the need to define new complex nodes in the DGG. It is never valuable to duplicate an attribute and generalize one to the point of being an ancestor of the other.

3. Related Work

Most work related to time in Artificial Intelligence (AI) has concentrated on defining a time domain and facilitating reasoning about events. Time can be represented using timepoints, time intervals, or a combination of the two, and it can be linear, non-linear or branching. In relational databases, time is represented by database timestamps that include time and date and that correspond to global (or absolute) timepoints in many temporal representation schemes. Thus, for knowledge discovery in relational databases that include timestamps, we use timepoints to define our calendar DGG and the intervals that correspond to less granular time periods. For our purposes, time is defined as linear, discrete, and bounded.

Terenziani discusses how to compose and intersect temporal constraints, and in doing so combines temporal logic with user-defined calendars [16]. Euzenat applies the idea of spatial granularity to the time domain and uses temporal algebra to construct upward and downward granularity change operators for converting time relationships from one granularity to another [6]. These ideas form the basis for the granularity generalization technique given in Section 2.

Cukierman and Delgrande [5] construct a calendar structure and calendar expressions that form the basis for our definition of a calendar DGG. They describe a decomposition technique that is the inverse of the generalization used in a calendar DGG. In addition, their directed acyclic graph of one calendar structure corresponds to the inverse of one concept hierarchy (and thus one path in the calendar DGG).

However, a DGG allows us to combine multiple calendar structures in the same representation, enabling us to take advantage of the duplication that exists between similar calendars. For example, a year is composed of 12 months, regardless of whether we are looking at a standard (Gregorian) year, an academic year, or a fiscal year.

Cukierman and Delgrande also define two orthogonal properties, *alignment* and *constancy*. Four types of decomposition for reasoning about schedulable, repeated activities follow from these two terms. *Aligned decomposition* means that the union of the submembers completely covers the original member (e.g., days from months). *Non-aligned decomposition* means that the union of the submembers does not completely cover the member leaving gaps at the beginning and/or end (e.g., weeks from months). *Constant decomposition* means that every time the member is broken into submembers, a constant ratio is maintained (e.g., days from weeks). A *non-constant decomposition* has a variable number of submembers (e.g., days from months).

For generalization, we define *compositions*, which are the inverses of Cukierman and Delgrande's decompositions. Four types of compositions can be defined analogously as *aligned composition*, *non-aligned composition*, *constant composition*, and *non-constant composition*. We intentionally do not include any non-aligned compositions in our calendar DGG. Instead, alternate paths of generalization from a common ancestor are explicitly shown. For example, months are not composed directly from weeks, but both months and weeks are composed from days in the calendar DGG. Constant composition is most appropriately implemented using a short algorithm, while non-constant composition is most appropriately implemented using a lookup table.

Other recent work has focussed on the connection between multiple temporal granularities and data mining. Granularity factors that affect data mining were described by Andrusiewicz and Orlowska [1]. Bettini et al. provide conventions similar to those given here for naming the multiple temporal granularities, although they do not provide a data structure similar to domain generalization graphs for representing the relationships between these granularities [2]. They also give preliminary ideas for applying their system in the context of data mining. Merlo et al. build on the work of Bettini et al. to specify the syntax and semantics of expressions involving data with multiple temporal granularities [12]. Combi et al. use a much simpler structure than our calendar DGG with an ordered granularity from SUP (top) to year, month, day, hour, minute, second, INF (bottom) [4]. However, they cannot handle phenomenon such as weeks.. Rainsford and Roddick provide a method for adding temporal semantics to association rules, based on a structured relationships among temporal relations, rather than our structure among domains [13]. Another approach to intensional query answering at multiple levels of abstraction is given by Suk et al. [17].

4. Probabilistic Domain Generalization Graphs

Given a set of generalized relations, an interest measure can assign a numeric score to each. These scores can be used to rank the results. These scores can also be used to

determine which generalized relations are consistent with an expectation and which ones conflict with it. Interest measures are computed by comparing an observed distribution to an expected distribution. The expectation of a tuple is the product of the expectations of the value in each of its component attributes.

The simplest approach is to assume a uniform distribution, but this approach requires specific assumptions about where the distribution is uniform. For example, at the *WeekdayOrWeekend* node (hereafter called *WDWE*), the domain consists of two elements and a uniform distribution gives an expected frequency of 0.50 for each of Weekend and Weekday, the two elements of the domain. Similarly, a uniform distribution among the seven elements of the *WeekdayName* node's domain, gives 0.14 for each of Sunday, Monday, etc. Unfortunately, these two distributions are inconsistent because only two days (Saturday and Sunday), with a total expectation of 0.29, are generalized to Weekend with an expectation of 0.50, while the remaining five days, with a total expectation of 0.71, are generalized to Weekday. To avoid such inconsistencies and generally simplify the process of specifying expectations, knowledge about expectations can be propagated among the nodes of the DGG. For example, more reasonable expectations of 0.29 for Weekend and 0.71 for Weekday result from propagating the distribution upward from *WeekdayName* to *WDWE*.

We define ***upward propagation*** as the process of translating a distribution from a node in the DGG up to at least one its child node(s) and possibly higher as well. The distribution at the higher level is the original distribution proportionately weighted according to the relevant generalization relation. Either a uniform or a non-uniform distribution can be upward propagated, but the distribution is assumed to be uniform unless otherwise stated. ***Bottom-up propagation*** propagates a distribution from the bottom node of the DGG to all other nodes of the DGG. ***Downward propagation*** is the process of propagating a distribution from a node to at least one of its parent nodes and possibly lower as well. ***Top-down propagation*** propagates a distribution from the top node of the DGG to other nodes of the DGG. Top-down propagation is based on the assumptions of uniformity at the most general level.

The concept hierarchies for the (WeekdayName, WeekdayOrWeekend, Any) path of the calendar DGG, as shown in Figure 2 illustrate upward and downward propagation. These concept hierarchies have the same nodes, but a different probability distribution is created based on the direction of propagation. For top-down propagation, consider the node *Any*, which is the top of the hierarchy of values shonw in Figure 2(a). The *Any* node has a domain of only one value, but this is assumed to represent a distribution, i.e., an expectation of 1 for this value. Since the single value at the top is created by generalizing from the two values in the domain of the *WDWE* node, we assume the distribution is 0.50 each. The distribution at the WeekdayName level follows from this assumption and further downward propagation.

As an example of upward propagation, we might assume that logins are uniformly distributed among the seven members of the *WeekdayName*, and propagate this distribution upward to obtain a 0.29 / 0.71 distribution for *WDWE*. In bottom-up propagation, *WeekdayName*'s distribution would itself be propagated upward from its parent nodes.

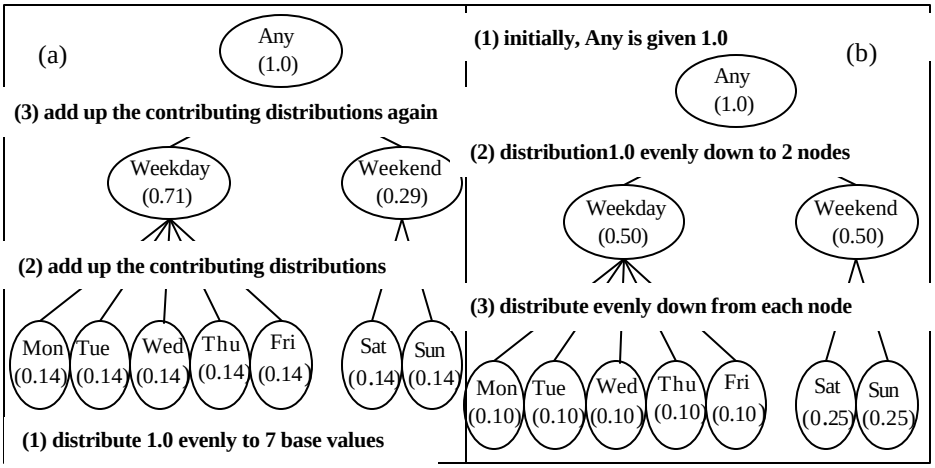

Figure 2. Propagation of Expectation Values: (a) upward, (b) downward.

The bottom-up approach always creates consistent distributions among all nodes in the DGG because the distribution at each node is consistent with the distribution of the base values at the bottom node. However, the top-down approach can suggest inconsistent distributions for a particular node in the DGG based on multiple paths down to it. Propagating a distribution from a single child down to a single parent gives an unambiguous result. Similarly, propagating from a single child down to multiple parents is equally straightforward. But when values are propagated down from multiple children to a single parent, it can be impossible to calculate consistent values without recalculating as much as the whole graph.

Four methods are provided for a user to specify expected distributions. The generalized relations are compared to these distributions. The methods are as follows.
(1) Data driven: As the generalization space is generated, the algorithm records how the values for each attribute are generalized. When the generalization space is complete, the distribution for every node in every DGG is available. This method is based on bottom-up propagation of a *non-uniform* distribution. Its two principal weaknesses are: (1) if any base value does not occur in the data, it is assigned 0 in the distribution, and (2) since the expectation is derived purely from the data, there will be no differences between expectation and observation.
(2) Data dictionary: If the expected distribution of values for an attribute is known at for any node in the DGG, the user can specify this distribution in a table in the database. The table contains one column with all unique values for the domain and a second column with the corresponding expected frequency (in the range 0-1). An entry is added to the *DGG* file to specify the database table and column names. The expected frequencies in the data dictionary are propagated upward to all DGG nodes reachable from the specified node. They can also be propagated downward to parent, and then grandparent, etc. as long as the ancestor node has only one child node.

```
username date        in   -  out   duration
user328  Sun Jan 18 00:26 - 00:26  (00:00)
user328  Sun Jan 18 00:55 - 00:59  (00:03)
user645  Sun Jan 18 01:21 - 01:42  (00:20)
...
user602  Sat Jan 24 23:48 - 23:48  (00:00)
```

Figure 3. Sample 'last' output

(3) Explicit histogram: The finest degree of control is available with an explicit histogram specification. It can optionally be specified for each DGG node. An example specification is as follows.

```
BeginExplicitHistogram
DGGNode=WeekdayOrWeekend
Weekday   0.714285714
Weekend   0.285714286
EndExplicitHistogram
```

The DGGNode line identifies the node. Each subsequent line contains a value occurring in that node's domain and the corresponding expected frequency. A warning is displayed if the sum of the expected frequencies exceeds one. In the current implementation, no propagation occurs.

(4) Explicit uniform distribution: The distribution for all elements at some node in the DGG is specified as uniform with an optional `Explicit Uniform Distribution` specification. An example specification is as follows.

```
BeginExplicitUniformDistributions
WeekdayName     0.142857142
DayNumberOfYear 0.002739726
EndExplicitUniformDistributions
```

Each content line specifies the name of a DGG node and the expected frequency to be distributed to each value in the domain. This type of specification is provided as a convenience for cases where a data dictionary or explicit histogram would be cumbersome. As with explicit histograms, a warning message is displayed if the sum of the expected frequencies exceeds one, and no propagation is done in the current implementation.

Our implementation prioritizes these alternatives in the following order: explicit histogram, explicit uniform distribution, data dictionary, and data driven. Thus, a value from an explicit histogram overrides a value from an explicit uniform distribution, and so forth. An output summary is produced as comma-separated values, which are readily displayed and processed with Microsoft Excel and other standard tools.

6. Example Application

We conducted a series of experiments using data from login statistics from 1998-2000. We studied the usage of two computers, Hercules and Mercury, but since they were similar, here we describe only the Hercules results.

The input data are 523253 lines of output from the Unix "`last`" program, which produces a single line for each user session that shows login and logout times and the duration of the session. The input data were collected over a period of somewhat more than two years, from June 1998 to June 2000. Sample output from `last` is shown in Figure 3. We focus on the login times, which can be easily converted to a format that can be mapped to the *YYYYMMDDhhmm* node in the DGG. Times are not recorded to seconds. The problem is to find interesting summaries of the data at various levels of granularity. We considered a simpler version of the same problem with far less data in [15].

We used variance as the measure of interest. We computed variance of the fraction of all logins for the following distributions:

USER	LOGINTIME	#Tuples	Coverage	Variance
Any	WeekdayOrWeekend	2	1	0.1889
Any	FiscalYear	3	1	0.0505
Group	Any	12	1	0.0341
Group	WeekdayOrWeekend	22	0.917	0.0129
Any	AcademicYear	3	1	0.0106
Any	YYYY	3	1	0.0106
Group	FiscalYear	29	0.806	0.0066
Group	AcademicYear	29	0.806	0.0052
Group	YYYY	29	0.806	0.0052
Any	AcademicYearAndSem	7	1	0.0045

Table 1. Top Ranked Nodes for Hercules data and (C, U)

Distribution 1. (C, U): We assumed a uniform distribution at all nodes in the username and calendar DGGs, based on the set of all occurring values. Thus, for 700 users, we expected 1/700 of the logins for each user, but for 12 user groups, we expected 1/12 of all logins to each group, regardless of the number of users in each group. A uniform distribution was assumed for years (among 1998, 1999, and 2000 for YYYY), for months in general (MM), for the 36 months (YYYYMM), for the 31 possible days of the month (DD), for the 7 days of the week (DayOfWeek), for the 2 parts of the week (WeekdayOrWeekend), etc.

Distribution 2. (C',U): We used the uniform distribution U for the username attribute and an adjusted distribution C' for the calendar DGG with two types of adjustments. First, the distribution for MM was adjusted for the varying number of days in the months (28 to 31) and the distribution for WeekdayOrWeekend was adjusted for the varying number of weekdays (5) and weekend days (2) in a week. The adjusted distributions were propagated downward. Secondly, because we knew that only parts of 1998 and 2000 were present, a further adjustment was made starting at the YYYYMMDD node. The distribution was propagated upward to all reachable nodes.

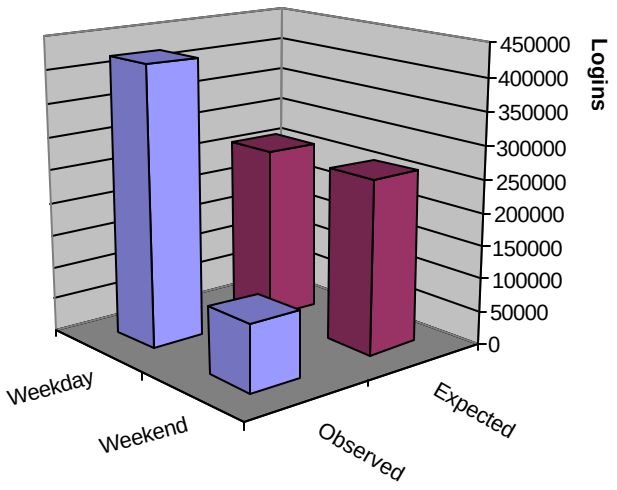

Figure 4. Observed and Expected Logins for (Any, WeekendOrWeekday)

After adjustment, the expected frequencies for the years were 0.25 for 1998, 0.60 for 1999, and 0.15 for 2000. The distribution at some nodes, such as YYYYMM, was affected by both upward and downward propagations. In these cases, we manually combined the distributions. For example, the information that in a typical year, 7 months have 31 days, 4 have only 30 days, approximately 0.25 have 29 days, and approximately 0.25 have 28 days was propagated downward. Suppose that the information that the data covered only the period 1 April 1998 to 30 April 2000 was propagated upward. Then the combined distribution would be based on the proportions: 14 with 31, 9 with 30, 1 with 29, and 1 with 28.

Distribtuion 3. (C', U'): We used the adjusted distribution C' for the calendar attribute and an adjusted distribution U' for the user attribute with somewhat arbitrary expectations based on our naïve beliefs about classes of users.

Based on these distributions, let us now examine a few of the highest ranked nodes in the generalization space and their corresponding graphs. As shown, in Table 1, for (C, U), the combination of the (any, WDWE) domains has the highest interest value. In other words, if the week was divided into two parts, weekdays and weekend, then the weekdays part had significantly more logins than the weekend part, as shown in Figure 4. This type of extremely obvious result is frequently obtained in data mining. With probabilistic DGGs, we have a convenient means of adjusting our expectations to avoid further repetition of both this obvious result and all its logical consequences.

As previously mentioned, the (C',U) distribution adjusts expectations to handle the inequalities in the number of days classified as weekdays (5) and weekend days (2), and all similar inequalities the calendar attribute. The interest ratings of nodes related to these concepts were lower with (C',U) than with (C, U). In the results shown in Table 2, (Group, Any) is ranked first instead of third as with (C, U). Also, (Any, WDWE) is second instead of first, and (Group, WDWE) is third instead of fourth.

USER	LOGINTIME	#Tuples	Coverage	Variance
Group	Any	12	1	0.0341
Any	WeekdayOrWeekend	2	1	0.0177
Group	WeekdayOrWeekend	22	0.917	0.0124
Group	FiscalYear	29	0.806	0.0063
Group	AcademicYear	29	0.806	0.0052
Group	YYYY	29	0.806	0.0052
Any	AcademicYearAndSem	7	1	0.0020
Any	AcademicYear	3	1	0.0015
Any	YYYY	3	1	0.0015
Any	FiscalYearAndQuarter	9	1	0.0013

Table 2. Top Ranked Nodes for Hercules data and (C', U)

The three nodes ranked highest in Table 2 are related. Investigation revealed that the (Group, Any) relation is unusual because `csugrd` and `unknown` have far more logins than expected while every other group has far fewer. This relationship is shown in Figure 5 for (Group, Any), where the X axis (across) gives the group, the Y axis (upward) gives the number of logins, the Z axis has observed values in front and expected values behind.

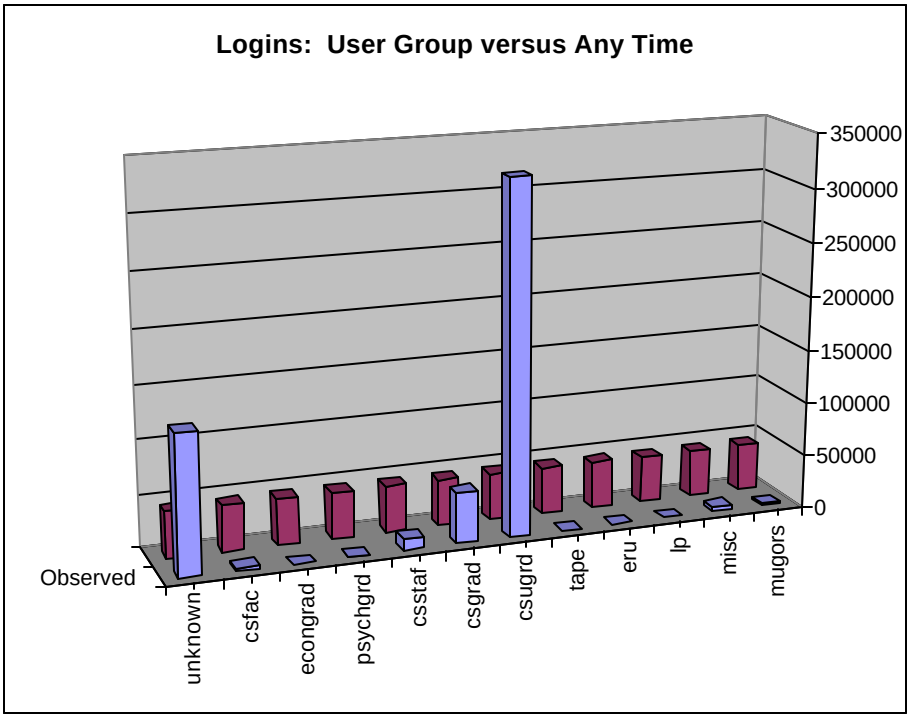

Figure 5. Graph for the (Group, Any) node for (C', U)

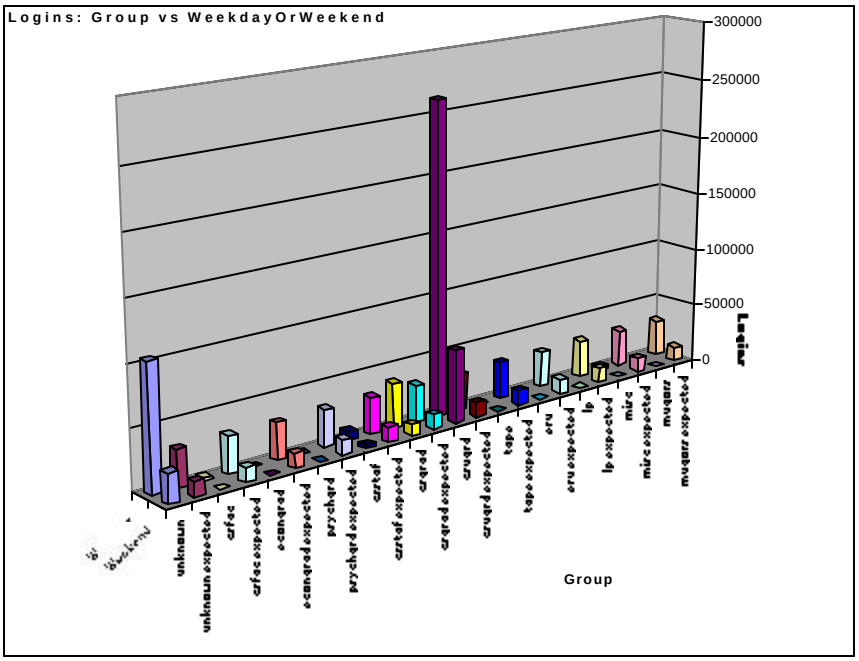

Figure 6. Graph for the (Group, WeekdayOrWeekend) node for (C', U)

The (Any, WDWE) relation is unusual because `weekday` logins are higher than expected based on the number of days. As well, these two factors combine to make (Group, WDWE) unusual. This relationship is shown in Figure 6 for (Group, WeekdayOrWeekend), where the X axis (across) gives the group, the Y axis (upward) gives the number of logins, and the Y axis gives the part of the week. On the X axis, the first value is the observed number of logins for the first group, the second value is the expected number of logins for this group, the third value is the observed number for the second group, the fourth value is the expected number for the second group, and so forth.

In Figure 6, the most salient feature is the great disparity between the number of logins among various groups. This disparity was addressed directly by the (C', U') distribution, where the U' distribution weighted the three groups CS undergraduates, unknown, and CS graduates much more highly than other groups.

USER	LOGINTIME	#Tuples	Coverage	Variance
Any	WeekdayOrWeekend	2	1	0.0177
Any	AcademicYearAndSem	7	1	0.0020
Any	AcademicYear	3	1	0.0015
Any	YYYY	3	1	0.0015
Any	FiscalYearAndQuarter	9	1	0.0013
Any	WeekdayName	7	1	0.0011
Any	FiscalYear	3	1	0.0011
Group	FiscalYear	29	0.806	0.0008
Any	MM	12	1	0.0007
Group	AcademicYear	29	0.806	0.0007

Table 3. Top Ranked Nodes for Hercules data and (C', U')

The ranking of the nodes for Hercules with the (C', U') distribution differs from that of (C', U). Some nodes, most notably (Group, Any) and (Group, WDWE), have lower rankings with (C', U') because of the added knowledge about the distribution in logins among the groups. As with (C, U), the top ranked node is (Any, WDWE), but this time it is because significant differences remain in the number of logins on weekdays versus weekends. As displayed in Figure 7, the graph for the (Any, AcademicYearAndSemester) node shows that fewer logins occur during the spring/summer semesters (e.g., 1999-2) than during the other semesters. In this graph, the 1998-2 and 2000-2 semesters have smaller expected values because the data cover only parts of these periods. As well, the expected values for the other semesters vary slightly because they are composed of differing numbers of days. The

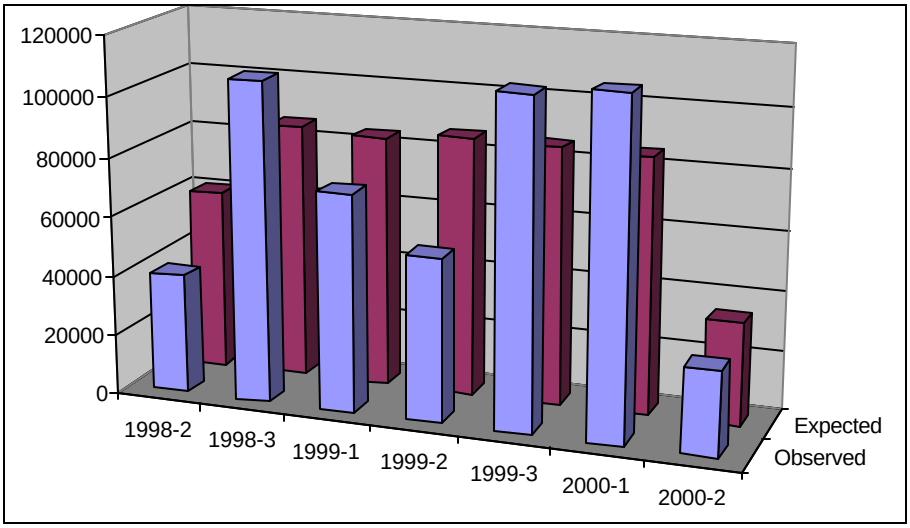

Figure 7. Graph for the (Any, AcademicYearAndSemester) Node for (C', U')

relationship identified as of interest, though, is that some semesters are considerably higher than expected and others lower.

When variance is used as the measure of interest, our technique provides an easy way for a user to construct a hierarchical statistical model based on his/her knowledge of the domain. By studying the summaries correspond to the highest ranked nodes, the user may gradually recognize the factors that contribute to the observed variance. As expectations are adjusted, the variance may be reduced closer and closer to zero. Unlike traditional hierarchical statistical models, our approach allows multiple paths through the hierarchy. Although the example shown used variance, any other appropriate interest measure, such as the other 15 measures included in the HMI set of heuristic interestingness measures [10], could be used in a similar fashion.

5. Conclusions

We have specified the components of a probabilistic domain generalization graph suitable for a calendar attribute. Four types of generalization were discussed. The calendar DGG is adaptable, allowing the user to easily add new arcs and nodes to the DGG when knowledge about a calendar attribute can be expressed in different ways. The proposed DGG does not depend on special features of a particular database or database platform; thus, it can be transported to other databases. Probabilistic DGGs are appropriate for data mining. Because of the complexity of the calendar DGG, it is useful to specify distributions of values at various nodes and explore the consequences of these distributions on the interestingness ratings of various summaries of the same data.

The use of other interestingness measures besides variance should be explored in the context of providing additional guidance to the user when selecting among many possible summaries.

Acknowledgement: The research reported in this paper was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada and the Institute for Robotics and Intelligent Systems. The referees contributed valuable suggestions.

References

1. A. Andrusiewicz and M. E. Orlowska, "On Granularity Factors That Affect Data Mining," *Eighth International Database Workshop, Data Mining, Data Warehousing and Client/Server Databases*, Hong Kong, 1997.
2. C. Bettini, S. Jajodia, and X.S. Wang. *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer-Verlag, Berlin, 2000.
3. C.L. Carter and H.J. Hamilton. Efficient attribute-oriented algorithms for knowledge discovery from large databases. *IEEE Transactions on Knowledge and Data Engineering*, **10**(2):193-208, March/April 1998.
4. C. Combi, F. Pinciroli, and G. Pozzi. Managing Time Granularity of Narrative Clinical Information: The Temporal Data Model TIME-NESIS. In *Proceedings of the Third International Workshop on Temporal Representation and Reasoning (TIME-96)*, pages 88-93, Key West, Florida, May 1996.

5. D. Cukierman and J. Delgrande. A language to express time intervals and repetition. In *Proceedings of the Second International Workshop on Temporal Representation and Reasoning (TIME-95)*, pages 41-48, Melbourne, Florida, April 1995.
6. J. Euzenat. An algebraic approach to granularity in time representation. In *Proceedings of the Second International Workshop on Temporal Representation and Reasoning (TIME-95)*, pages 147-154, Melbourne, Florida, April 1995.
7. H.J. Hamilton, R.J. Hilderman, and N. Cercone. Attribute-oriented induction using domain generalization graphs. In *Proceedings of the Eighth IEEE International Conference on Tools with Artificial Intelligence (ICTAI'96)*, pages 246-253, Toulouse, France, November 1996.
8. J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(1):29-40, February 1993.
9. R. J. Hilderman, H. J. Hamilton, and N. Cercone, Data Mining in Large Databases using Domain Generalization Graphs, *Journal of Intelligent Information Systems*, vol. 13, pp. 195-234, 1999.
10. R.J. Hilderman and H.J. Hamilton, Heuristic Measures of Interestingness. In *Proceedings of the Third European Conference on the Principles of Data mining and Knowledge Discovery (PKDD'99)*, pages 232-241, Prague, Czech Republic, September 1999.
11. J. Hobbs. Granularity. *Proc. International Joint Conference on Artificial Intelligence*, Los Angles, pages 432-435.
12. I. Merlo, E. Bertino, E. Ferrari, S. Gadia, G. Guerrini. Querying Multiple Temporal Granularity Data. In *Proceedings of the Seventh International Workshop on Temporal Representation and Reasoning (TIME-2000)*, pages 103-114, Cape Breton, Nova Scotia, Canada, July 2000.
13. C. P. Rainsford and J. F. Roddick. Adding Temporal Semantics to Association Rules, in *Third European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*. Prague: Springer, 1999, pp. 504-509.
14. D. J. Randall, H. J. Hamilton, and R. J. Hilderman. A Technique for Generalizing Temporal Durations in Relational Databases. In *Eleventh International FLAIRS Conference (FLAIRS-98)*, pages 193-197, Sanibel Island, FL, May 1998.
15. D. J. Randall, H. J. Hamilton, and R. J. Hilderman. Temporal Generalization with Domain Generalization Graphs, *International Journal of Pattern Recognition and Artificial Intelligence*. **13**(2):195-217, 1999.
16. P. Terenziani. Reasoning about Periodic Events. *In Proceedings of the Second International Workshop on Temporal Representation and Reasoning (TIME-95)*, pages 137-144, Melbourne, Florida, April 1995.
17. S.-C. Yoon and E. K. Park. An Approach to Intensional Query Answering at Multiple Abstraction Levels using Data Mining Approaches, *32nd Annual Hawaii International Conference on Systems.*