ISSN: 0828-3494
ISBN: 0-7731-0452-6

Expectation Propagation in ExGen Graphs for Summarization: Preliminary Report

Liqiang Geng & Howard J. Hamilton
Technical Report CS-2003-03
September 2003

Department of Computer Science
University of Regina
Regina, Saskatchewan
CANADA S4S 0A2

Expectation Propagation in ExGen Graphs for Summarization: Extended Report

Liqiang Geng & Howard J. Hamilton
Department of Computer Science
University of Regina
Regina, Saskatchewan
Canada S4S 0A2
{gengl, hamilton}@cs.uregina.ca

Abstract

Summarization based on expected distribution domain generalization (ExGen) graphs aggregates data into summaries in many ways and identifies summaries that are far from user expectations, i.e., interesting. In this paper, we tackle two problems. First, we propose how to consistently propagate an expected distribution given by the user for one node to the entire ExGen graph. Secondly, we propose three interestingness measures. Based on these measures, we propose heuristics to prune nodes from the ExGen while searching for interesting summaries. We also demonstrate the interactive experimental process of our method and show the results we obtained by applying it to the Saskatchewan weather data.

1. Introduction

Summarization, which is the formation of interesting, compact descriptions of data, was identified by Fayyad et al. as one of the six chief data mining tasks [3]. Early work on attribute-oriented induction [2] and function finding [9] provided diverse approaches to summarization. Summarization is the crucial task addressed by online analytical processing (OLAP), data cubes with rollup and drilldown operators, and the rollup and cube by operators in SQL. Nonetheless, the common approaches to summarization have three weaknesses:

1. Many valid summaries of the same data or its subsets can be produced, and during exploration, the data analyst must examine summaries one by one to assess them.

2. The capacity for incorporating existing knowledge is limited. Although attribute-oriented induction allows the incorporation of domain knowledge as a concept hierarchy for each attribute, multiple ways of aggregating the values for an attribute during a single data mining task are not permitted. Similarly, hierarchical statistical models allow expectations, in the form of a priori distributions, to be specified, but only for a single hierarchy of distributions.

3. The ability to respond to changes in the user's knowledge during the knowledge discovery process is limited. For example, if it is discovered that more pay television shows are watched during the evening than the afternoon, it will subsequently be less interesting to learn that more shows are watched starting at 8:00PM than 4:00PM.

To overcome these limitations, we propose a summarization method based on expected distribution domain generalization (ExGen) graphs. A domain generalization graph (DGG) [4,6,7] is a graphical structure that can be used both as a navigational aid by the user and as a guide to heuristic data mining procedures. The nodes in a DGG are domains of values, and the arcs tell how to generalize values from one domain to values in another. Each path in the graph corresponds to a generalization consistent with the specified generalization relations. An expected distribution domain generalization graph, or ExGen graph, is a DGG where each node has been augmented with a description of the expected distribution (or simply expectations) of the values in the corresponding domain. Initial and changing domain knowledge relevant to summarization is readily described by ExGen graphs. During the knowledge discovery process, the expectations at a particular node,

which reflect a user's knowledge about the corresponding domain, can be updated. As well, expectations can be allowed to propagate to other nodes in the graph; for example, if the user revises the expectation about the number of shows watched in the evening, then the expectation about the number of shows watched from 8:00PM to 9:00PM can be automatically adjusted.

The ExGen mining process has five steps. First, a domain generalization graph for an attribute is created by explicitly identifying the domains appropriate to the relevant levels of granularity and the mappings between the values in these domains. Second, the expectations (expected probability distributions) are associated with each node to give an ExGen graph. Third, the data are aggregated in all possible ways consistent with this graph. Aggregation is performed by transforming values in one domain to another, according to the directed arcs in the domain generalization graph. Each aggregation is called a summary. Fourth, the summaries are ranked according to their distance from expectations for the appropriate domain, using a diversity-based interestingness measure [5,6], such as variance. Fifth, the highest ranked summaries are displayed. Expectations are then adjusted and steps repeated as necessary.

Using existing knowledge to prune mining results and highlight surprising findings has been studied by many researchers. Liu et al. proposed a specification for representing users' ambiguous and precise knowledge [8]. Mined association rules are compared with the knowledge. The system then ranks the association rules according to matching degree. Bay and Pazzani proposed an approach to mine contrast sets for categorical data [1]. Once the contrast sets are found, post-processing selects a subset that may be surprising to the user, based on the contrast sets that have already been displayed. Their approach does not allow incorporation of known generalization relations among attributes.

The remainder of this paper is organized as follows. In the following section, we give the theoretical basis for ExGen graph. In Section 3, we define generalized ExGen graph. In Section 4, we propose interestingness measures and a pruning strategy to reduce the number of the summaries. In Section 5, we demonstrate the experimental process and results on Saskatchewan weather data. In Section 6, we present our conclusions.

2. ExGen Graphs and Propagation of Expectations

An ExGen graph is used to represent the user's knowledge relevant to generalization. For a large ExGen graph, it is not practical to require the user to specify the expectations for all nodes. In exploratory data mining, the user may begin with very little knowledge about the domain, perhaps only vague assumptions about the (a priori) probabilities of the possible values at some level of granularity. After the user specifies the probabilities, the system should be able to create the preliminary distributions for the other nodes. When doing so, two principles are followed: (1) the distribution at all nodes should be consistent with each other, and (2) the distribution should be as even as possible. First we give some formal definitions.

Definition 1. Given a set $X = \{x_1, x_2, \ldots, x_n\}$ representing the domain of some attribute and a set $P = \{P_1, P_2, \ldots, P_m\}$ of partitions of the set X. we define a nonempty binary relation $\preceq$ (called a ***generalization relation***) on P, where we say $P_i \preceq P_j$ if for every section $S_a \in P_i$, there exists a section $S_b \in P_j$, such that $S_a \subseteq S_b$. For convenience, we often refer to the sections by labels.
If $P_i \preceq P_j$, for each section $S_b \in P_j$, there exists a set of sections $\{ S_{a_1}, \ldots, S_{a_k} \} \subseteq P_i$, denoted ***Spec***$(S_b, P_i)$, such that $S_b = \bigcup_{i=1}^{k} S_{a_i}$. The generalization relation $\preceq$ is a partial order relation.

Example 1. Let X be a domain of cities {Munich, Zurich, Nanjing, Shanghai, Regina, Vancouver, NYC} and let P be a set of partitions $\{P_1, P_2, P_3\}$, where P_1 is based on country, P_2 is based on continent, and P_3 is based on the size of population, defined as follows.

P_1 = {German cities, Swiss cities, Chinese cities, Canadian cities, US cities}, where the labeled sections are defined as follows: German cities = {Munich}, Swiss cities = {Zurich}, Chinese cities = {Nanjing, Shanghai}, Canadian cities = {Regina, Vancouver}, and US cities = {NYC};

P_2 = {European cities, Asian cities, North American cities}, where European cities = {Munich, Zurich}, Asian cities = {Nanjing, Shanghai}, and North American cities = {Regina, Vancouver, NYC};

P_3 = {Large cities, Medium cities, Small cities}, where Large cities ={Shanghai, NYC}, Medium cites = {Munich, Nanjing, Vancouver}, and Small cities = {Zurich, Regina}.

From these partitions, we can obtain $P_1 \preceq P_2$.

Definition 2. A ***domain generalization graph*** (*DGG*) $G = \langle P, E \rangle$ is constructed based on a generalization relation $\langle P, \preceq \rangle$ as follows. The nodes of the graph are the elements of P. There is a directed arc from P_i to P_j iff $P_i \neq P_j$, $P_i \preceq P_j$, and there is no $P_k \in P$ such that $P_i \preceq P_k$ and $P_k \preceq P_j$. Each node corresponds to a domain of values called ***sections***. Each arc corresponds to a generalization relation, which is a mapping from the values in the domain of the initial (or *parent*) node to that of the final node (or *child*) of the arc. The ***bottom*** (or source) node of the graph corresponds to the original domain of values X and the ***top*** (or sink) node T corresponds to the most general domain of values, which contains only the value ANY.

From Example 1, we obtain the DGG shown in Figure 1. The generalization relations represented are $X \preceq P_1$, $X \preceq P_3$, $P_1 \preceq P_2$, $P_2 \preceq T$, and $P_3 \preceq T$. The $X \preceq P_2$ relation is not represented in the graph, because $X \preceq P_1$ and $P_1 \preceq P_2$ are represented.

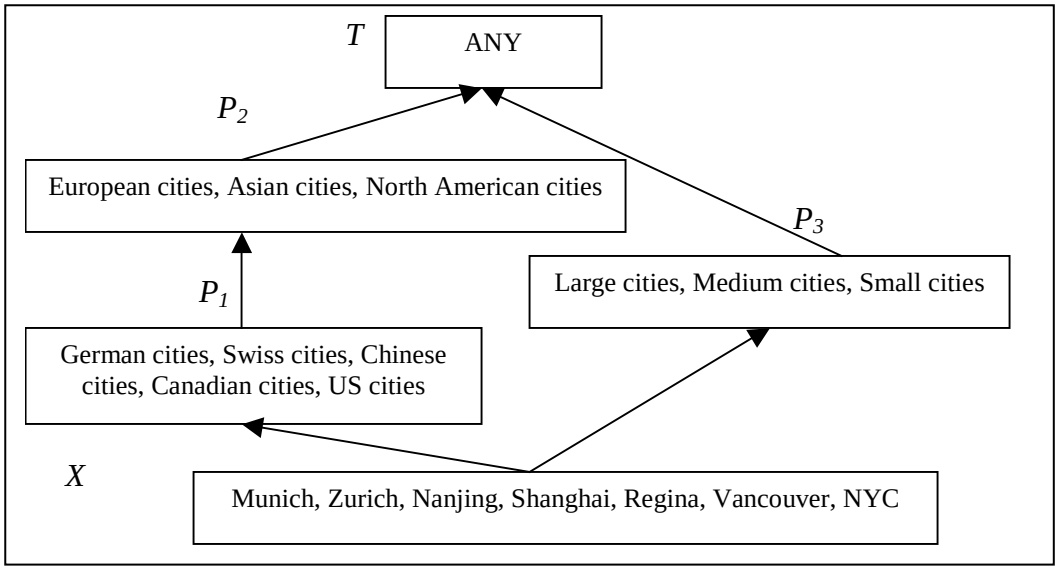


Figure 1. DGG for city domain

Definition 3. An ***expected distribution domain generalization*** (or ***ExGen***) ***graph*** is a DGG that has expectation associated with every node. Each expectation represents the expected probability distribution of

occurrence of the values in the domain corresponding to the node. For a node (i.e., partition) P_j = {S_1, …, S_k}, we have $\forall S_i \in P_j, 0 \le E(S_i) \le 1$ and $\sum_{i=1}^{k} E(S_i) = 1$, where $E(S_i)$ denotes the expectation of occurrence of section S_i.

Figure 2 is an ExGen graph extended from Example 1 and Figure 1. The probability value attached to each section denotes the expectation that an international activity will be held in any of the cities denoted by the section label.

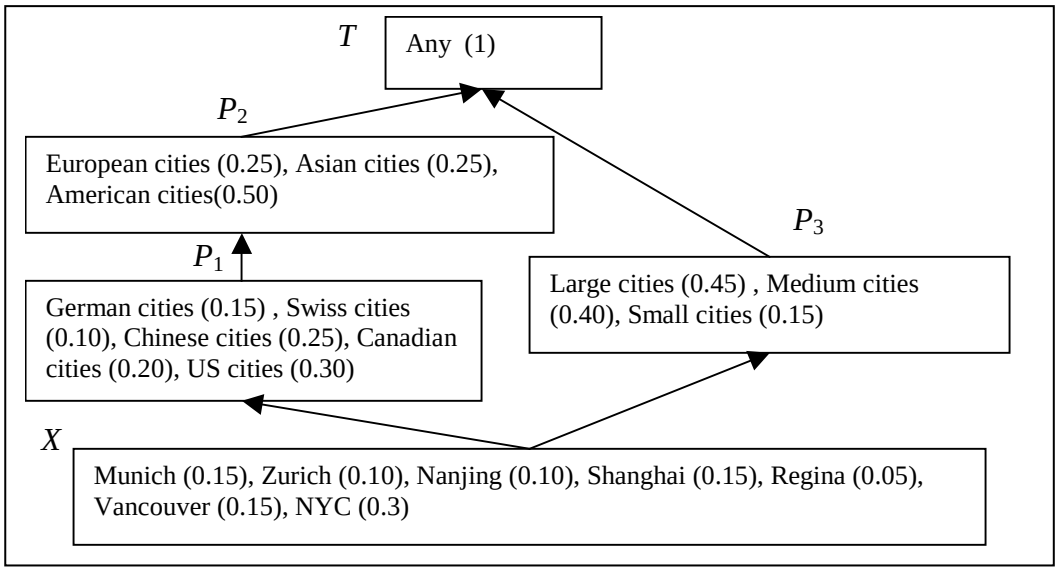


Figure 2. A sample ExGen

Definition 4. Assume node *Q* is a parent of node *R* in an ExGen graph, and therefore for each section $S_b \in R$, there exists a set of sections *Spec*(S_b, *Q*) = { S_{a_1}, ..., S_{a_k} } $\subseteq Q$, such that $S_b = \bigcup_{i=1}^{k} S_{a_i}$. If for all $S_b \in R$, $E(S_b) = \sum_{i=1}^{k} E(S_{a_i})$, we say that nodes *Q* and *R* are ***consistent***.

Definition 5. In a ExGen graph, we say that node *R* is ***bottom-consistent***, i.e., consistent with the bottom node *X*, if for all $S_i \in R$, $E(S_i) = \sum_{x \in S_i} E(x)$.

It is obvious that the bottom node *X* is itself bottom-consistent.

Definition 6. An ExGen graph *G* is ***consistent*** if all pairs of adjacent nodes in *G* are consistent.

Figure 2 is a consistent ExGen graph; for example, node *X* is a parent of node P_1. In node *X*, there are two Canadian cities, Regina and Vancouver, with *E*(Regina) = 0.05 and *E*(Vancouver) = 0.15, which are consistent with *E*(Canadian cities) = 0.20 in node P_1.

Lemma 1. An ExGen graph *G* is consistent iff every node in *G* is bottom-consistent.

Proof.

(1). Suppose *G* is consistent.

Let the ***distance*** between node *R* and node *Q* is the minimum number of arcs that must be traversed to get from *R* to *Q*. By definition 6, since *G* is consistent, the nodes at a distance of 1 from the bottom node (the child nodes of the bottom node) are bottom-consistent.

Now assume the nodes at a distance of *k* from the bottom nodes are bottom-consistent. Any node *R* at a distance of *k*+1 from the bottom node must be a child node of some node *Q* at a distance of *k* from the bottom node. Because G is consistent, *Q* and *R* are consistent. Therefore, for every $S_b \in R$, we have $Spec(S_b, Q) = \{S_{a_1}, ..., S_{a_k}\}$ and $E(S_b) = \sum_{i=1}^{k} E(S_{a_i})$.

We know $E(S_{a_i}) = \sum_{x \in S_{a_i}} E(x)$, since node *Q* is assumed to be bottom-consistent. Hence, $E(S_b) = \sum_{i=1}^{k} E(S_{a_i}) = \sum_{i=1}^{k} \sum_{x \in S_{a_i}} E(x) = \sum_{x \in S_b} E(x)$, i.e., *R* is bottom-consistent.

(2). Suppose every node in *G* is bottom-consistent.

Let *R* be an arbitrary non-bottom node with parent *Q*. For any $S_b \in R$, we have $Spec(S_b, Q) = \{S_{a_1}, ..., S_{a_k}\}$ and $S_b = \bigcup S_{a_i}$. Since *Q* and *R* are both bottom-consistent, we have $E(S_b) = \sum_{x \in S_b} E(x)$ and $E(S_{a_i}) = \sum_{x \in S_{a_i}} E(x)$. Therefore, $\sum_{i=1}^{k} E(S_{a_i}) = \sum_{i=1}^{k} \sum_{x \in S_{a_i}} E(x) = \sum_{x \in S_b} E(x) = E(S_b)$. *R* and *Q* are consistent. Since *Q* and *R* are an arbitrary parent-child pair, we say *G* is consistent.

For each node in an ExGen graph, the user can specify an expectation of explicit probability distribution, one of a parameterised standard distribution, or leave the expectation unconstrained. If a standard distribution is specified, it is discretized into an explicit probability distribution. If no expectation is specified for a node, one is obtained by propagation from a specified expectation. A specified expectation may be either a ***hard constraint***, which once specified, must be satisfied after all subsequent propagations, or a ***soft constraint***, which must be satisfied for the current propagation, but thereafter it need not be satisfied. We identify five situations for expectation propagations: given an expectation at the bottom node, no expectations for any node, given an expectation at a single non-bottom and non-top node, given expectations in multiple nodes, and updating expectation at one node. Let us consider each in turn.

(1) Given an expectation at the bottom node (Bottom-up propagation).

Assume expectation at the bottom node is $E = \{E(x_1), \ldots, E(x_n)\}$ and node *Q* is consistent through upward propagation. Let *R* is the child node of *Q*, S_b is a section of *R*, and $Spec(S_b, Q) = \{S_{a_1}, ..., S_{a_k}\}$ is the set of the sections in *Q* that forms S_b. When we propagate expectation from *Q* to *R*, we have $E(S_b) = \sum_{i=1}^{k} E(S_{a_i}) = \sum_{i=1}^{k} \sum_{x \in S_{a_i}} E(x) = \sum_{x \in S_b} E(x)$. Therefore, Bottom-up propagation in an ExGen graph preserves consistency.

(2) No information about expectation in any node.

In this situation, we assume the distribution for each element of the domain is uniform. In other words, we assume a uniform distribution at the bottom node. Then we apply bottom-up propagation.

(3) Given an expectation at a single non-bottom and non-top node.

Given an expectation $E = \{E_1, \ldots, E_k\}$ for a node $R = \{S_1, \ldots, S_k\}$, we distribute the expectation uniformly among $Sepc(S_i, X)$ for each section S_i, and therefore get the expectation for the jth element of the ith section $E_{ij} = E_i / |S_i|$. After obtaining the expectation at the bottom node, we use bottom-up propagation to distribute expectations.

(4) Given expectations in multiple nodes.

We find the greatest lower bound of all nodes whose distribution is known. We calculate the expectation of the greatest lower bound and then use the same method as with case 3 to determine the bottom node's distribution.
Assume that node i has j_i sections S_{ik}, where $1 \le k \le j_i$, E_{ik} is the given expectation for S_{ik}, and x is a basic element. We have $E(S_{ik}) = E_{ik}$, i.e., $\sum_{x \in S_{ik}} E(x) = E_{ik}$. There are three possible cases for the solutions for the linear equation group.

(a) One non-negative solution. We do not need to do anything, except assign the solution as the initial expectation to the bottom node and do upward propagation.

(b) More than one non-negative solution.
This is the most usual case for our problem. It can be converted to an optimisation problem. We want to satisfy the equation group and at the same time to make the distribution as even as possible. That is we need to minimize the following expression
$\sum_{i=1}^{n} (E(x) - 1/n)^2$, where n is the number of the elements in the domain.
Or maximize the entropy measure
$-\sum_{i=1}^{n} E(x) \log E(x)$.
After obtaining the expectation at bottom node, we use bottom-up propagation to distribute the expectations.

(c). No non-negative solutions. We remind the user to change the input knowledge to remove the conflict.

(5) Updating a distribution in one node.

We need to satisfy the new conditions as well as all the hard constraints, and at the same time minimize the following expression to make the changes to the nodes as little as possible.
$\sum_{i=1}^{n} (E\_new(x) - E\_old(x))^2$

or $-\sum_{i=1}^{n} E\_new(x)\log E\_new(x)$.

For large scale ExGen graphs, the above mentioned optimisation process has a prohibitive time cost. We propose an alternative method to decide the distribution for bottom nodes. We convert $E(S_{ik}) = E_{ik}$ to $E(x) = \frac{E\_old(x)}{E\_old(S_{ik})} E_{ik}, x \in S$, which means that we accept the probability ratio among the groups and also retain the ratio among the elements among each group. The advantage of this approach is computational efficiency, because it only involves linear computation; the disadvantage is that all the distributions specified are soft constraints, i.e., after specifying and propagating a new constraint, any old constraint may be violated. In the following we will identify some theoretical properties that can ensure that the old constraints remain valid.

Definition 7. For a pair of nodes, $A = \{S_{A1}, \ldots, S_{Am}\}$ and $B = \{S_{B1}, \ldots, S_{Bn}\}$ in an ExGen graph, if $E(S_{Ai})E(S_{Bj}) = E(S_{Ai} \cap S_{Bj})$, for all i and j, we say that partitions A and B are independent.

Lemma 2. For a pair of nodes $A = \{S_{A1}, \ldots, S_{Am}\}$ and $B = \{S_{B1}, \ldots, S_{Bn}\}$ in a ExGen graph, if they are independent, and we use the linear propagation method, then changing distribution in one node does not change the distribution in the other.

Proof

Assume we change the distribution of node B from $E(S_{B1}), \ldots, E(S_{Bn})$ to $E'(S_{B1}), \ldots, E'(S_{Bn})$,

$$E'(S_{Ai}) = \sum_{j=1}^{n} E'(S_{Bj} \cap S_{Ai}),$$

Because we use the linear propagation method, we have $E'(S_{Bj} \cap S_{Ai}) = \frac{E'(S_{Bj})}{E(S_{Bj})} E(S_{Bj} \cap S_{Ai})$,

$$E'(S_{Ai}) = \sum_{j=1}^{n} E'(S_{Bj} \cap S_{Ai}) = \sum_{j=1}^{n} \frac{E'(S_{Bj})}{E(S_{Bj})} E(S_{Bj} \cap S_{Ai}).$$

Since A and B are currently independent, we have $E(S_{Ai}) = \frac{E(S_{Bj} \cap S_{Ai})}{E(S_{Bj})}$, therefore,

$$E'(S_{Ai}) = \sum_{j=1}^{n} E(S_{Ai})E(S_{Bj} \cap S_{Ai}) = E(S_{Ai}).$$

Therefore, distribution for node A does not change.

Lemma 3. If A and B are independent, then A is independent of B's descendents, and B is independent of A's descendents.

Proof.

We have $E(S_{Ai}) * E(S_{Bj}) = E(S_{Ai} \cap S_{Bj})$ for all i and j, $i = 1$ to m, and $j = 1$ to n.

Assume C is a descendent node of A with k sections S_{Cm}, $1 \le m \le k$. We have

$$E(S_{cm}) * E(S_{Bj}) = \sum_{S_{Ai} \in Spec(S_{Pj}, A)} E(S_{Ai}) * E(S_{Bj})$$

$$= \sum_{S_{Ai} \in Spec(S_{Pj}, A)} E(S_{Ai} \cap S_{Bj}) = E(\bigcup_{S_{Ai} \in Spec(S_{Pj}, A)} S_{Ai} \cap S_{Bj}) = E(C_m \cap S_{Bj})$$

Theorem. If all paths between the bottom node and top node do not cross over and all pairs of the children of the bottom node are independent, then updating distribution in one node (except the bottom node) only need to propagate among the path where the node is.

Both constraint-based propagation method and linear propagation method are based on three principles: (1) the propagation should be consistent, (2) if new information is being added, the new information should be preserved, while the old condition should be changed as little as possible, (3) if available information is does not fully constrain the distribution, at a node, the distribution should be made uniform as possible.

The following is an example to illustrate the constraint-based propagation.

Suppose we are given a DGG as shown in Figure 3. Node MMDDMAN has the domain of morning, afternoon, and night of a specific non-leap year {Morning of January 1, Afternoon of January 1, night of January 1, …, night of December 31}, MMDD has the domain of specific days {January, 1, January 2, …, December 31}, MAN has the domain {Morning, Afternoon, Night}, MM has the domain {January, February, …, December}, and Week has the domain {Sunday, Monday, …, Saturday}. We specify expected probability of accessing the computer system in the University of Regina for some nodes as follows: the expectation [0.09, 0.12, 0.12, 0.09, 0.04, 0.04, 0.04, 0.04, 0.09, 0.12, 0.12, 0.09] for node MM, the distribution [0.1, 0.2, 0.2, 0.2, 0.1, 0.1, 0.1] for node Week, and no information about the distribution of other nodes. Since the node *MMDD* is the greatest lower bound of *MM* and *Week*, we first calculate the distribution in *MMDD* for a non-leap year as [e_1, e_2, …, e_{365}]. Assuming January 1st is Sunday, we have the following linear constraints,

$$\sum_{i=1}^{31} e_i = 0.09,\ \sum_{i=32}^{59} e_i = 0.12,\ \ldots,\ \sum_{i=335}^{365} e_i = 0.09,$$

$$\sum_{i\%7=1} e_i = 0.1,\ \sum_{i\%7=2} e_i = 0.2,\ \ldots,\ \sum_{i\%7=0} e_i = 0.1$$

And we minimize the following formula,

$$\sum_{i=1}^{365} (e_i - 1/365)^2$$

Then we have *MMDD* = [0.0020, 0.0040, 0.0040, …, 0.0022, 0.0021]. Next we calculate the distribution for bottom node by dividing the distribution of *MMDD* by 3 (divide evenly among morning, afternoon and night). We get *MMDDMAN* = [0.00067, 0.00133, …, 0.0007]. Finally we use upward propagation to get the distribution for *MAN* = [0.333, 0.333, 0.333]. *MAN* is given a uniform distribution because no prior distribution was available.

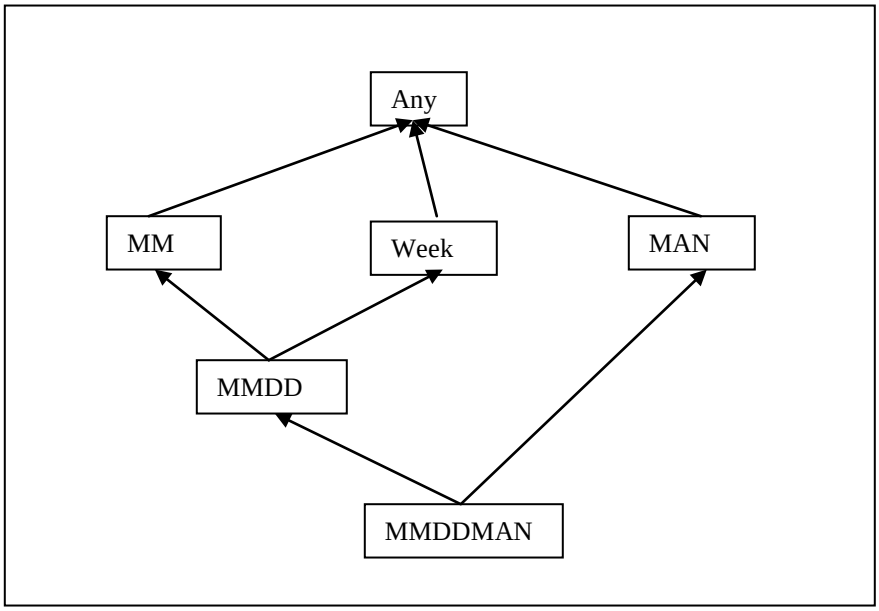


Figure 3. A DGG for Time

Now if new information is obtained that the distribution among morning, afternoon and night is *MAN* = [0.3, 0.5, 0.2], we have the following constraints in the bottom node.

$$\sum_{i \in Morning} e_i = 0.3, \quad \sum_{i \in Afternoon} e_i = 0.5, \quad \sum_{i \in night} e_i = 0.2.$$

We minimize the following formula,

$$\sum_{di \in MMDDMAN} (e_i - e_i\_old)^2$$

and get the new distribution for bottom node. Then after upward propagation, we get *MMDDMAN* = [0.00057, 0.0011, …, 0.00029], *MM* = [0.0884, 0.1186, 0.1184, 0.0885, 0.0433, 0.0427, 0.0431, 0.0432, 0.0885, 0.1184, 0.1185, 0.0884] and *Weekday* = [0.1021, 0.1974, 0.1974, 0.1974, 0.1021, 0.1019, 0.1018]. It can be seen that after adopting the new distribution in node *MAN*, the other nodes are changed a little, which means the original information is significantly preserved.

If we specify all the constraints at the same time, similarly we get *MMDDMAN* = [0.00057, 0.0011, …, 0.00028] and *MMDD* = [0.0019, 0.0041, 0.0041, 0.0040, …, 0.0021, 0.0021]. The distribution for *MM* and *Weekday* and *MAN* are identical to the distributions specified for them.

If we use the linear-based process, we get *MMDDMAN* = [0.0006, 0.0010, …, 0.00041], *MM* = [0.0938, 0.1111, 0.1222, 0.0875, 0.0426, 0.0389, 0.0407, 0.0417, 0.0875, 0.1250, 0.1195, 0.0895] and *Weekday* = [0.1000, 0.2000, 0.2000, 0.2000, 0.1000, 0.1000, 0.1000].

3. Propagation in Generalized ExGen (GenSpace) graphs

An ExGen graph is based on one attribute. If the generalization space consists of more than one attribute, we need to construct a GenSpace graph. Each node in a GenSpace is a Cartesian product of nodes in all ExGen graphs. The number of the nodes in the generalization state space is $l_1 * \ldots * l_n$, where n is the number of the

attributes and l_i is the number of nodes in ExGen graph for attribute i. We give the definition of GenSpace graph as follows.

Definition 7. Given a set of attributes $\{A_1, A_2, \ldots A_n\}$, and $<P_i, E_i>$ representing ExGen graph for attribute A_i. A general space is defined as a pair $<P, E>$, where $P = P_1 * P_2 * \ldots * P_n$. For two nodes $Q = [P_{Q1}, P_{Q2}, \ldots, P_{Qn}]$ and $R = [P_{R1}, P_{R2}, \ldots, P_{Rn}] \in P$, where $P_{Qi} \in P_i$ and $P_{Ri} \in P_i$ denotes the partition of attribute A_i in nodes Q and R respectively, if $P_{Qi} \preceq P_{Ri}$ for $1 \le i \le n$, we say $Q \preceq R$. The generalization relationship $\preceq$ is a partial order relation and $\langle P, \preceq \rangle$ defines a partially ordered set from which we can construct a graph called a ***generalized expected distribution domain generalization graph***, or **GenSpace graph**, $\langle P, E \rangle$.

The following is the algorithm to constructing a GenSpace graph.

Input: a set of ExGen graphs for a set of attributes.

1. Select all the bottom nodes of ExGen graphs and create the bottom node of GenSpace graph by their Cartesian product. Then put the node in a queue.
2. Select the head of the queue and for each element of the tuple find the children in its corresponding ExGen graph and replace the element of the tuple with the child nodes and form a set of new nodes. For the nodes that are not yet in the GenSpace graph, insert these nodes as the current nodes' children in GenSpace graph and also insert these nodes in the queue, otherwise, connect parent-child nodes in the GenSpace graph. Delete the current node in the queue.
3. Repeat step 2 until the queue is empty.

It can be seen that the GenSpace graph constructed from the algorithm satisfies the following conditions. There is a directed arc from node Q to R iff $Q \neq R$, $Q \preceq R$, and there is no node S such that $Q \preceq S$ and $S \preceq R$. The top node (or sink node) corresponds to the most general domain of values, which consists of only the value tuple [ANY, ANY, …, ANY].

The following is an example to illustrate the construction of a GenSpace graph.

Given two ExGen graphs in Figure 4,

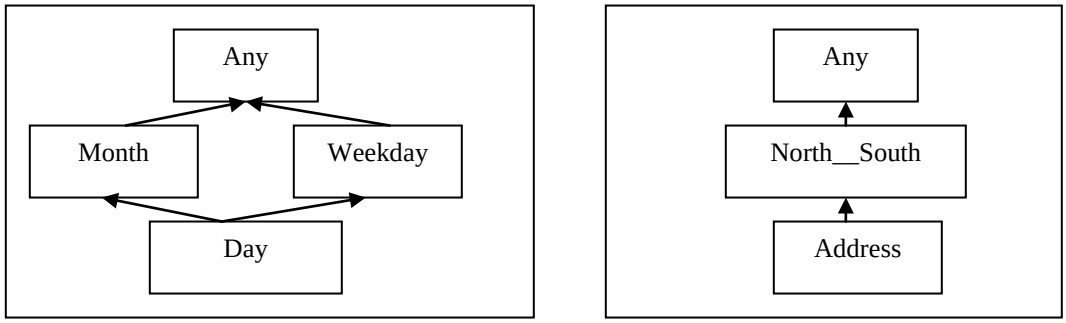


Figure 4. ExGens for single attributes

We generate the GenSpace graph in Figure 5.

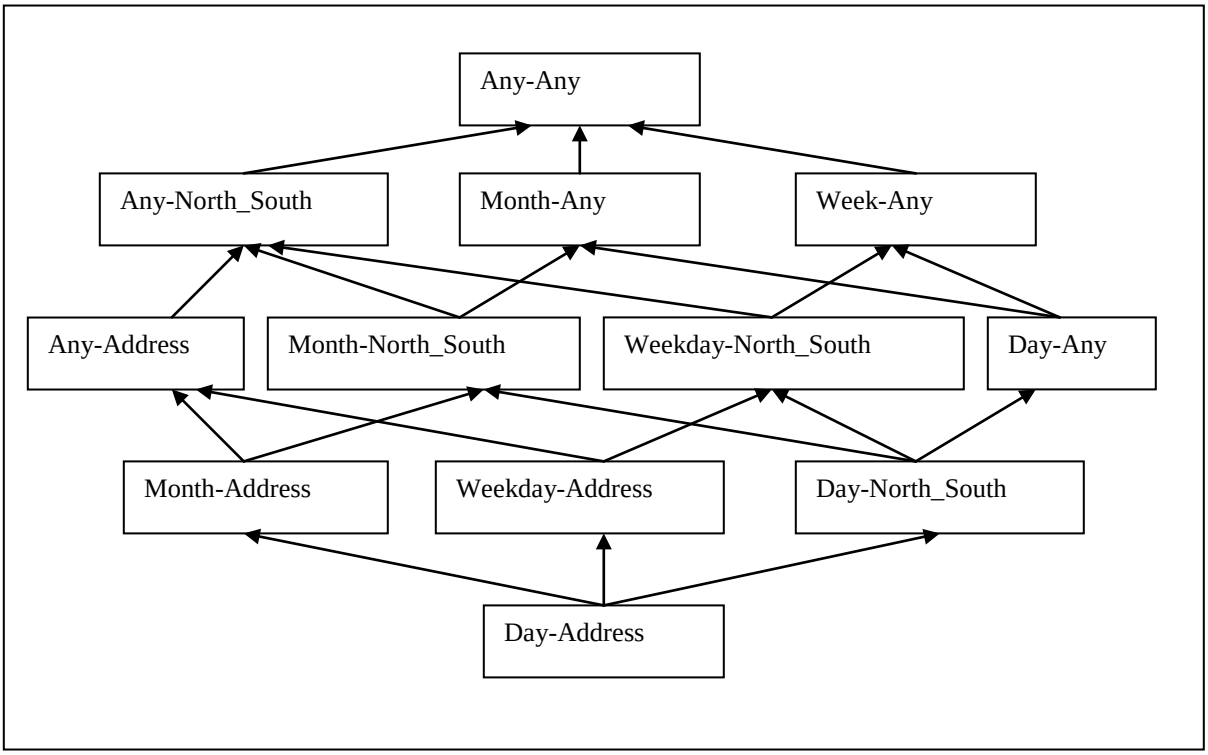


Figure 5. An GenSpace graph.

Definition 8. Assume node *Q* is a parent of node *R* in an GenSpace graph, and therefore for each section $S_b \in R$, there exists a set of sections $Spec(S_b, Q) = \{ S_{a_1}, ..., S_{a_k} \} \subseteq Q$, such that $S_b = \bigcup_{i=1}^{k} S_{a_i}$. If for all $S_b \in R$, $E(S_b) = \sum_{i=1}^{k} E(S_{a_i})$, we say that *Q* and *R* are ***consistent***.

Definition 9. If all pairs of directly linked nodes in a GenSpace graph are consistent, the GenSpace graph is consistent.

3.1 Procedure to obtain GenSpace graph with initial expectations

1. User provides the ExGen graphs for all attributes and the expectations on some specific nodes.

2. The system calculates the distributions for the bottom nodes of all ExGen graphs.

3. The system generates the GenSpace graph.

4. Assuming the independence between all attributes initially, we obtain the distribution for the bottom node of GenSpace graph:

$$E(e_{1l_1}, ..., e_{kl_k}, ..., e_{nl_n}) = \prod_{k=1}^{n} E(e_{kl_k})$$

5. The system propagates the distribution bottom up to generate the distributions for all the nodes in the GenSpace graph.

3.2 Propagation in GenSpace graph

1. Using optimisation

If the expectation for node i (i = 1, 2, …,m) [S_{i1}, S_{i2}, …, S_{in}] are given, we have $E(x_{i1j_{i1}}, x_{i2j_{i2}},..., x_{injin})$, where $S_{ikj_{ik}} \in d_{ik}$ and $k \in \{1, 2,…, n\}$.We find the greatest lower bound of all the nodes whose distribution is known. We call the elements in the greatest lower bound the basic elements. We calculate the expectation of the greatest lower bound.

Assume E_{kl} (k = 1, 2, …, n) is the distribution for basic element. We have constraints $E(S_{i1j_{i1}}, S_{i2j_{i2}},..., S_{injin}) = E_{ik}$, i.e., $\sum_{ekl_k \subseteq vikj_{ik}} E(x_{1l_n},..., x_{kl_k},..., x_{nl_n}) = E_{ik}$. The problem is to optimise $\sum_{ekl_k \subseteq vikj_{ik}} (E\_new(x_{1l_n},..., x_{kl_k},..., x_{nl_n}) - E\_old(x_{1l_n},..., x_{kl_k},..., x_{nl_n}))^2$.

2. Linear method

For large scale GenSpace graphs, the cost of the above mentioned optimisation process is prohibitive. We use the following method to decide the distribution for bottom nodes. We convert the distribution for node i $E(S_{i1j_{i1}}, S_{i2j_{i2}},..., S_{injin}) = E_{j_1 j_2 ... j_n}$ to

$$E(x_{1l_n},..., x_{kl_k},..., x_{nl_n}) = \frac{E\_old(S_{1l_n},..., S_{kl_k},..., S_{nl_n})}{E\_old(S_{i1j_{i1}}, S_{i2j_{i2}},..., S_{injin})} E_{j_1 j_2 ... j_n}, x_{kl_k} \in S_{ikj_{i2}}$$

4. Pruning the nodes in searching the interesting summaries

The number of the nodes of GenSpace graph is exponential to the number of the attributes. We use breadth first search starting from the bottom node to search for interesting summaries. In each layer we choose n nodes with the greatest distance and prune the other nodes and all their descendent nodes. The intuition is that if at one level the expected distribution approximates to the real distribution, the summarization based on this distribution is also less interesting to the user, hence pruned. This heuristics tends to prune too many nodes, sometimes the upper layer will never be calculated. An alternative is we can only prune the less interesting nodes in current level and the nodes that only have the pruned nodes as the parents.

These heuristics cannot guarantee to find the most interesting summaries. In the next section we will propose three interestingness measures. Based on this measures, we propose a heuristics, which can guarantee to find the most interesting summaries.

Suppose in a GenSpace graph, a node P consists of partition {S_1, S_2, …, S_n}, we propose

Interestingness measure 1: $Intr(P) = \frac{1}{n}\sum_{i=1}^{n}\frac{|e(S_i)-o(S_i)|}{o(S_i)}$,

Interestingness measure 2: $Intr(P) = \frac{1}{n}\sum_{i=1}^{n}\frac{|e(S_i)-o(S_i)|}{\min(e(S_i),o(S_i))}$,

Interestingness measure 3: $Intr(P) = \frac{1}{n}\sum_{i=1}^{n}\frac{|e(S_i)-o(S_i)|}{\max(e(S_i),o(S_i))}$,

where $o(S_i)$ and $e(S_i)$ are observed and expected probabilities for S_i, respectively.

Measure 1 is asymmetric for observed probability and the estimation. Measure 2 and 3 are symmetric. Measure 1 and 2 can take any non-negative values. Measure 3 can take values in [0,1].

Lemma 3. In a GenSpace graph, a node P consists of partition $\{S_1, S_2, \ldots, S_n\}$, and we use the interestingness measure1, 2 or 3. If for all partition a_i we have $\frac{|e(S_i)-o(S_i)|}{o(S_i)} \le t, t \ge 0,$ we can have interestingness measure of all the descendents of P less than t.

Proof for measure1.

Because $\frac{|e(a_i)-o(a_i)|}{o(a_i)} \le t,$ for any i, we have interestingness measure for node P $Intr(P) = \frac{1}{n}\sum_{i=1}^{n}\frac{|e(a_i)-o(a_i)|}{o(a_i)} \le t$.

Suppose node Q is a descendent of node P. We have the partition for $Q = \{b_1, b_2, \ldots, b_m\}$, where $b_i = \bigcup_{a_j \in b_i} a_j$, $\frac{|e(b_i)-o(b_i)|}{o(b_i)} = \frac{|\sum_{a_j\in b} e(a_j) - \sum_{a_j\in b} o(a_j)|}{\sum_{a_j\in b} o(a_j)} = \frac{|\sum_{a_j\in b}(e(a_j)-o(a_j))|}{\sum_{a_j\in b} o(a_j)} \le \frac{\sum_{a_j\in b}|e(a_j)-o(a_j)|}{\sum_{a_j\in b} o(a_j)}$

Because for any nonnegative value x_i and y_i, we have $\frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} y_i} \le \max(\frac{x_i}{y_i})$,

We get $\frac{|e(b_i)-o(b_i)|}{o(b_i)} \le \max_{a_j\in b}(\frac{|e(a_j)-o(a_j)|}{o(a_j)}) \le t$,

Therefore, $Intr(Q) = \frac{1}{m}\sum_{i=1}^{n}\frac{|e(b_i)-o(b_i)|}{o(b_i)} \le t$.

From this lemma we can see that if all the variance of the partitions in a node has an upper bound, then all the interest measures of its descendents can not exceed this upper bound.

If the domain of the attributes is determined by the value appearing in the data set, $o(b_i)$ will never be zero. However if the domain is specified by the user, it could be zero. Here we define $\frac{0}{0} = 1$ *and* $\frac{t}{0} =$ *the greatest value in the system* $(t \ne 0)$.

Proof for measure 2.

Because $\frac{|e(a_i)-o(a_i)|}{\min(e(a_i),o(a_i))} \le t$, for any i, we have interest measure for node P $Intr(P) = \frac{1}{n}\sum_{i=1}^{n}\frac{|e(a_i)-o(a_i)|}{\min(e(a_i),o(a_i))} \le t$.

Suppose node $Q = \{b_1, b_2, \ldots, b_m\}$ is a descendent of node P.

$$\frac{|e(b_i)-o(b_i)|}{\min(e(b_i),o(b_i))} = \frac{|\sum_{a_j\subseteq b_i} e(a_j) - \sum_{a_j\subseteq b_i} o(a_j)|}{\min(\sum_{a_j\subseteq b_i} e(a_j), \sum_{a_j\subseteq b_i} o(a_j))} = \frac{|\sum_{a_j\subseteq b_i} (e(a_j)-o(a_j))|}{\min(\sum_{a_j\subseteq b_i} e(a_j), \sum_{a_j\subseteq b_i} o(a_j))} \le \frac{\sum_{a_j\subseteq b_i}|e(a_j)-o(a_j)|}{\sum_{a_j\subseteq b_i}\min(e(a_j),o(a_j))}$$

Because for any nonnegative value x_i and y_i, we have $\frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} y_i} \le \max(\frac{x_i}{y_i})$,

We get $\frac{|e(b_i)-o(b_i)|}{\min(e(b_i),o(b_i))} \le \max_{a_j\in b}(\frac{|e(a_j)-o(a_j)|}{\min(e(a_j),o(a_j))}) \le t$,

Therefore, $Intr(Q) = \frac{1}{m}\sum_{i=1}^{n}\frac{|e(b_i)-o(b_i)|}{\min(e(b_i),o(b_i))} \le t$.

Proof for measure 3.

Use induction.

(1) If $i = 2$, we assume $b = a_1 \cup a_2$,

$$\frac{|e(b)-o(b)|}{\max(e(b),o(b))} = \frac{|e(a_1)+e(a_2)-o(a_1)-o(a_2)|}{\max(e(a_1)+e(a_2),o(a_1)+o(a_2))}$$

(1.1) If $e(a_1) \ge o(a_1)$ *and* $e(a_2) \ge o(a_2)$, $\frac{|e(b)-o(b)|}{\max(e(b),o(b))} = \frac{e(a_1)-o(a_1)+e(a_2)-o(a_2)}{e(a_1)+e(a_2)}$

$\frac{|e(a_1)-o(a_1)|}{\max(e(a_1),o(a_1))} = \frac{e(a_1)-o(a_1)}{e(a_1)}$, $\frac{|e(a_2)-o(a_2)|}{\max(e(a_2),o(a_2))} = \frac{e(a_2)-o(a_2)}{e(a_2)}$,

Therefore, we have $\frac{|e(b)-o(b)|}{\max(e(b),o(b))} \le \max(\frac{|e(a_1)-o(a_1)|}{\max(e(a_1),o(a_1))}, \frac{|e(a_2)-o(a_2)|}{\max(e(a_2),o(a_2))})$

(1.2) If $e(a_1) \le o(a_1)$ *and* $e(a_2) \le o(a_2)$, the proof is similar.

(1.3) If $e(a_1) \ge o(a_1)$ *and* $e(a_2) \le o(a_2)$,

(1.3.1) If $e(a_1)+e(a_2) \ge o(a_1)+o(a_2)$,

we have $\frac{|e(b)-o(b)|}{\max(e(b),o(b))} = \frac{e(a_1)+e(a_2)-o(a_1)-o(a_2)}{e(a_1)+e(a_2)}$,

$\frac{|e(a_1)-o(a_1)|}{\max(e(a_1),o(a_1))} = \frac{e(a_1)-o(a_1)}{e(a_1)}$

$$\frac{|e(b)-o(b)|}{\max(e(b),o(b))}-\frac{|e(a_1)-o(a_1)|}{\max(e(a_1),o(a_1))}=\frac{e(a_1)+e(a_2)-o(a_1)-o(a_2)}{e(a_1)(e(a_1)+e(a_2))}=\frac{e(a_2)o(a_1)-e(a_1)o(a_2)}{e(a_1)(e(a_1)+e(a_2))}\le 0$$

$$\frac{|e(b)-o(b)|}{\max(e(b),o(b))}\le\frac{|e(a_1)-o(a_1)|}{\max(e(a_1),o(a_1))}\le\max(\frac{|e(a_1)-o(a_1)|}{\max(e(a_1),o(a_1))},\frac{|e(a_2)-o(a_2)|}{\max(e(a_2),o(a_2))})$$

(1.3.2) If $e(a_1)+e(a_2)\le o(a_1)+o(a_2)$, similarly we can get

$$\frac{|e(b)-o(b)|}{\max(e(b),o(b))}-\frac{|e(a_2)-o(a_2)|}{\max(e(a_2),o(a_2))}=\frac{e(a_2)o(a_1)-e(a_1)o(a_2)}{o(a_1)(o(a_1)+o(a_2))}\le 0,$$

$$\frac{|e(b)-o(b)|}{\max(e(b),o(b))}\le\frac{|e(a_2)-o(a_2)|}{\max(e(a_2),o(a_2))}\le\max(\frac{|e(a_1)-o(a_1)|}{\max(e(a_1),o(a_1))},\frac{|e(a_2)-o(a_2)|}{\max(e(a_2),o(a_2))})$$

(1.4) If $e(a_1)\le o(a_1)$ *and* $e(a_2)\ge o(a_2)$, the proof is similar to situation (1.3).

Therefore, for *i* = 2, we have $\frac{|e(b)-o(b)|}{\max(e(b),o(b))}\le\max(\frac{|e(a_1)-o(a_1)|}{\max(e(a_1),o(a_1))},\frac{|e(a_2)-o(a_2)|}{\max(e(a_2),o(a_2))})$

(2). Assuming for *i* = *k*,

We have $\frac{|e(c)-o(c)|}{\max(e(c),o(c))}\le\max_{i\in[1,k]}(\frac{|e(a_i)-o(a_i)|}{\max(e(a_i),o(a_i))})$.

When *i* = *k* + 1,

$$\frac{|e(b)-o(b)|}{\max(e(b),o(b))}=\frac{|e(c)+e(a_{k+1})-o(c)-o(a_{k+1})|}{\max(e(c)+e(a_{k+1}),o(c)+o(a_{k+1}))}$$

$$\le\max(\frac{|e(c)-o(c)|}{\max(e(c),o(c))},\frac{|e(a_{k+1})-o(a_{k+1})|}{\max(e(a_{k+1}),o(a_{k+1}))})$$

$$\le\max(\max_{i\in[1,k]}(\frac{|e(a_i)-o(a_i)|}{\max(e(a_i),o(a_i))}),\frac{|e(a_{k+1})-o(a_{k+1})|}{\max(e(a_{k+1}),o(a_{k+1}))})$$

$$=\max_{i\in[1,k+1]}(\frac{|e(a_i)-o(a_i)|}{\max(e(a_i),o(a_i))})$$

This lemma indicates that when we use the proposed interest measure, the biggest child's variance is less than the biggest parent's variance (or the child's variance is less than the greatest variance of the child nodes). But it does not mean that the average variance of the child is less than that of parent. Figure 6 gives an example to illustrate this lemma.

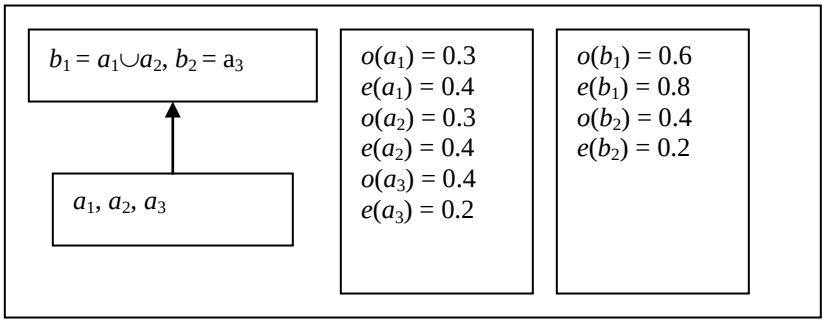


Figure 6. An Example to illustrate Lemma 3.

In this case, $v(a_1) = \frac{|e(a_1) - o(a_1)|}{\max(e(a_1), o(a_1))} = 0.25$, similarly $v(a_2) = 0.25$, $v(a_3) = 0.5$, $v(b_1) = 0.25$, $v(b_2) = 0.5$. We can see that the variances of child node is not greater than the greatest variance of parent node. However, the interest measure for child node (0.375) is greater than that for parent node (0.333).

These properties does not apply to the measure $\frac{1}{n}\sum_{i=1}^{n}(e(a_i) - o(a_i))^2$ or $\frac{1}{n}\sum_{i=1}^{n}|e(a_i) - o(a_i)|$. The following is a counter example.

Assume $b = a_1 \cup a_2$, $o(a_1) = 0.1, e(a_1) = 0.2, o(a_2) = 0.2, e(a_2) = 0.3$, we have $o(b) = 0.3, e(b) = 0.5$. We denote the variance as *v*. We have $v(a_1) = (e(a_1) - o(a_1))^2 = 0.01$,
$v(a_2) = (e(a_2) - o(a_2))^2 = 0.01$, and $v(b) = (e(b) - o(b))^2 = 0.04$. $v(b) \geq \max(v(a_1), v(a_2))$.

We use this lemma to prune nodes in the search process.

1. Find the best *n* nodes.

From bottom to top in a GenSpace graph, we calculate the measures for first *n* nodes. They are the currently selected nodes.
Let *t* be the smallest interest measure in the *n* nodes.
For the next unvisited and unpruned node, calculate the variances for each partition. If they are all less than *t*, we prune it and all its descendent nodes, otherwise, we check if its interest measure is greater than the smallest interest measure of currently selected n nodes, if yes, replace this old one with the new one and set *t* be the least interest measure of the currently selected nodes.

This algorithm can find the *n* most interesting summaries, but in the worst case, no node will be pruned.

2. User gives the threshold *t*.

Oonce we find a node whose partitions are all less than *t*, we prune it and all its decedents. The greater *t* is, the more nodes are pruned.

In propagation process, we do not really need to calculate distribution for all nodes, but only need to calculate distribution for bottom node. We calculate distribution of a node in the process of calculating its interest measure. Because some nodes will be pruned, their distributions will not be needed.

5. Experimental Results

We now describe the data mining technique we implemented in our DGG-Discover 5.0 software. For clarity, the method is first briefly explained and then its application to the datasets concerning weather in the province of Saskatchewan, Canada.

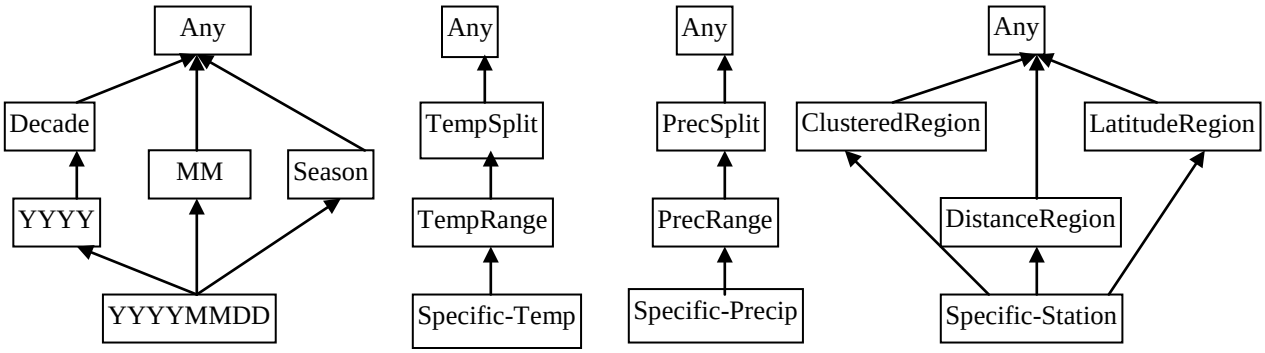
Our goal was to assess whether, with no previous experience with this dataset, this discovery methodology could guide exploration of the data. We used the daily high temperature (in 0.1 degree Celcius) and daily total precipitation (in mm, with snow converted to equivalent water) for all weather stations for all days from January 1, 1900 to December 31, 1949. Other fields concerning low temperatures and snowfall were not

used in our analysis. The number of daily weather observations (tuples) was 211,534. Example data is given in Table 1, where StationName and Latitude and Longitude depend on the Station attribute.

Station	StationName	Latitude	Longitude	Date	High Temperature	TotalPrecip
4012400	Estevan A	49.217	102.967	12/27/1944	-12.8	0
4063560	Island Falls	55.533	102.350	12/27/1944	-22.2	0
4016560	Regina A	50.433	104.667	12/27/1944	-14.4	0
4018160	Tugaske	50.883	106.300	12/27/1944	-20.6	0
4048520	Waseca	53.133	109.400	12/27/1944	-18.9	0
4019080	Yorkton	51.267	102.467	12/27/1944	-18.9	0

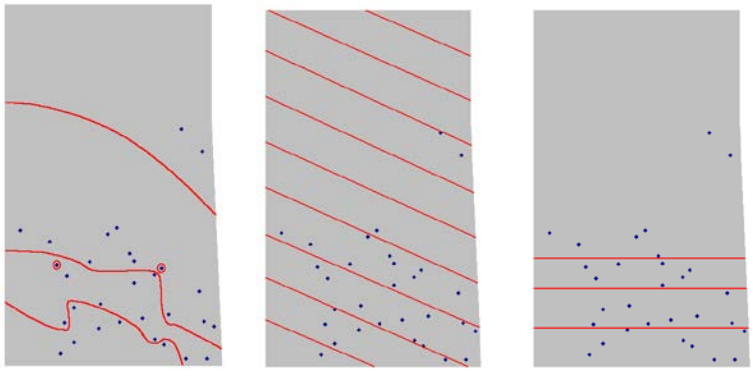
Table 1. Sample Weather Data.

The attributes we used in our experiment are *Station, Date, High Temperature*, and *TotalPrecip*. DGG in Figure 7(a) corresponds to attribute *Date* and indicates that a particular date (*YYYYMMDD*) can be generalized to a year, a month, or a season. As well, years can be generalized to decades. For the *High Temperature*, *TotalPrecip*, and *Station* attributes, we used the DGGs shown in Figure 7(b), 7(c), and 7(d), respectively. The three paths in the Station DGG correspond to the four maps shown in Figure 8.



(a) (b) (c) (d)

Figure 7. DGGs for Date, HighTemperature, TotalPrecip, and Station Attributes



(a) Clustered Regions (CRegion) (b) Distance Regions (DRegion) (c) Latitude Regions (LRegion)

Figure 8. Maps Corresponding to the Interior DGG nodes for the Station Attribute

Figure 7(a) shows the stations clustered using the *k*-means algorithm with $k = 6$ (other values of k gave less plausible maps). Figure 7(b) shows ten regions defined based on the adjusted distance d to any point (Lat, Long) from the southwest corner of the province, which is at (49°N, 110°E), using the formula
$d = (\text{Lat} - 49) + 0.35\ (110 - \text{Long})$

This assumes that similar weather occurs along a slanted line across the province. The 0.35 is an arbitrary constant defined based on a map of the province showing ecological regions [8]. Since the northeast corner of the province is at (60°N, 102°E), the values for d ranged from 0 to 13.8. To create the DistRange node, we divided this range into 10 equal-sized intervals. Figure 7(c) shows the stations grouped from south to north into four regions (South, LowMid, HighMid, and North) to create the Latitude node.

Associating expectations: We assumed initial distribution for some nodes in the DGGs For the *Date* attribute, we assumed a uniform distribution at the *Year* (*YYYY*) node, i.e., we assumed an equal number of observations from each year 1900-1949. For the *Station* attribute, we assumed a uniform distribution at each weather station (*Specific-Station* node). For the *HighTemperature* attribute, we assumed a uniform distribution at the *TempRange* node, which means that cold days, cool days, warm days, and hot days all have probability value of 0.25. Similarly, for *TotalPrecip*, we assumed a uniform distribution at the *PrecRange* node, which means that the numbers of days for no precipitation, low precipitation, medium precipitation and high precipitation are equal. A crucial property of our method is that it is self-correcting: if inappropriate initial distributions are assumed, the user is quickly informed of them and through re-specifying and propagating the new expectation to rectify them.

Generalize the Data: Next the weather data are aggregated in all possible ways consistent with the domain generalization graphs. Since the *Date*, *HighTemperature*, *TotalPrecip*, and *Station* DGGs have 6, 4, 4, and 5 nodes, respectively, the number of aggregations or summaries produced is 6(4)(4)(5) = 480. Each summary is evaluated with an interestingness measure and also optionally stored as a Comma-Separated-Value (CSV) file.

Rank the Summaries: The summaries are ranked according to their distance from the expected distribution using a selected interestingness measure.

A list of the k highest ranking summaries is displayed; Table 2 shows the top 10 summaries for this example. Also, the highest ranked summary, which corresponds to the (Any, Any, Any, PrecSplit) node combination, is automatically displayed as an Excel graph by our interface This two-line summary shows that, when Station, Date, and Temperature are ignored (set to Any), the percentage of daily observations in the data with rain is 21%, and the number without is 79%. Since the original expectation was 75%/25%, the actual distribution is far from the expected one, and according to the variance measure, farthest from the expectation of any node combination.

Adjust Expectations: At this point, the user has learned something about the domain, and the expectations can be adjusted according to the acquired knowledge. To continue this example, we assume that the distribution is simply accepted for the PrecipSplit node and propagated to the *PrecipRange* and *Specific-Precip* nodes in the TotalPrecip ExGen. This assumption follows in a straightforward fashion from the results, and can be readily automated. The effect on further data mining corresponds to saying: "I accept that only 21% of the days have precipitation; now, don't tell me about that again or about any logical consequence of that."

Station	Date	Temperature	Precipitation	Variance
Any	Any	Any	PrecSplit	0.688432
Any	Any	TempSplit	PrecSplit	0.116666
Any	Any	Any	PrecRange	0.110478
LRegion	Any	Any	PrecSplit	0.0263574
Any	Any	TempRange	PrecSplit	0.0255874
Any	Any	TempSplit	PrecRange	0.0251891
Any	Season	Any	PrecSplit	0.0247278
CRegion	Any	Any	PrecSplit	0.0207634
DRegion	Any	Any	PrecSplit	0.0202915
Any	Decade	Any	PrecSplit	0.0181746

Table 2. Top-Ranked Summaries After Run 1

Continue Data Mining: Using the revised expectations, the ten highest ranked summaries are shown in Table 3. Most summaries with *PrecipSplit* or *PrecipRange* have disappeared from the top ten list because of the change in expectations. The summaries (Any, Any, Any, PrecRange) and (Any, Decade, Any, PrecSplit) remain in the top ten in the table. However, the summary (Any, Any, Any, PrecRange) changed its rank from 3 to 7 and its interestingness measure decreased from 0.110478 to 0.00291085. Although summary (Any, Decade, Any, PrecSplit) has a higher ranking in the new table (from 10 to 8), its interestingness measure decreased from 0.0181746 to 0.00287561. This indicates that through propagation from node PrecSplit, the distribution for other nodes in this ExGen has been adjusted appropriately.

Now, the highest ranked summary tells us that the expectation for decade node is far away from the observed one. Among the five decades from 1900 to 1949, the percentage of observations from the decades in order are: 7%, 13%, 21%, 25%, and 34%. Again, the user can simply accept this, or explore deeper to understand why. The actual reason was that only a few weather stations existed in 1900 and others were gradually added. This relationship is best addressed by creating a joint distribution (discussed further below) between Station and Date (at the Year or YYYYMMDD node). For this example, we will assume that the user simply accepts the observed distribution as the expected for Decade and propagates it downward to Year.

Station	Date	Temperature	Precipitation	Variance
Any	Decade	Any	Any	0.0105093
Any	Any	TempSplit	Any	0.00842794
Any	Season	TempSplit	Any	0.00670198
LRegion	Any	Any	Any	0.00634375
CRegion	Any	Any	Any	0.00588974
Any	Any	TempRange	Any	0.00377398
Any	Any	Any	PrecRange	0.00291082
Any	Decade	Any	PrecSplit	0.00287561
Any	Decade	TempSplit	Any	0.00254169
DRegion	Any	Any	Any	0.00238893

Table 3. Top Ranked Summaries After Run 2.

Table 4 to 7 list the ten most interesting summaries for runs 3 to 6.

When the top-ranked summary has more than one domain value that is not “Any”, the summary corresponds

to a ***joint probability distribution*** (or ***joint expectation***). For example, after Run 4, the top-ranked summary is (Any, Season, TempSplit, Any), which means that the proportion of Low Temperature and High Temperature days varies with the season. To accept this distribution, a joint expectation between Season and TempSplit is created. All subsequent runs will consult this table whenever an expectation for the combination of Season and TempSplit needs to be calculated. After propagation of the joint expectation and calculation of the interestingness of the summary, we can see that not only has (Any, Season, TempSplit, Any) disappeared from the top ten summaries, but also closely related nodes (Any, Season, TempSplit, PrecSplit) and (Any, Season, TempRange, Any) have become less interesting and also disappeared.

The results that we obtain from the system are an ordered list of summaries that have high interestingness values. Within each summary is a list of expected and observed probability distributions. If the variance of a record in the summary is greater than a threshold, we will highlight this record and provide it to the user. If the observed probability is higher than the expectation, we say that this record is more likely than what the user expected and highlight it in red. Otherwise, we say it is less likely and highlight it in blue. For example, if we have surprising record "summer, hot" with observation and expectation probabilities with 0.2 and 0.1, we highlight it in red, which means that "hot, summer" records occurred more frequently than expected.

Station	Date	Temperature	Precipitation	Variance
Any	Any	TempSplit	Any	0.00842794
Any	Season	TempSplit	Any	0.00670198
LRegion	Any	Any	Any	0.00634375
CRegion	Any	Any	Any	0.00588974
Any	Any	TempRange	Any	0.00377398
Any	Any	Any	PrecRange	0.00291082
DRegion	Any	Any	Any	0.00238893
Any	Season	TempSplit	PrecSplit	0.00211296
Any	Season	TempRange	Any	0.00207967
Any	Any	TempSplit	PrecSplit	0.00192754

Table 4. Top Ranked Summaries After Run 3.

Station	Date	Temperature	Precipitation	Variance
Any	Season	TempSplit	Any	0.00640098
LRegion	Any	Any	Any	0.00634375
CRegion	Any	Any	Any	0.00588974
Any	Any	Any	PrecRange	0.00291082
CRegion	Any	Any	PrecSplit	0.00176968
DRegion	Any	Any	Any	0.00238893
Any	Any	TempRange	Any	0.00236933
Any	Season	TempSplit	PrecSplit	0.00201667
Any	Season	TempRange	Any	0.00200944
LRegion	Any	Any	PrecSplit	0.0017268

Table 5. Top Ranked Summaries After Run 4

Station	Date	Temperature	Precipitation	Variance
LRegion	Any	Any	Any	0.00634375
CRegion	Any	Any	Any	0.00588974
Any	Any	Any	PrecRange	0.00291082
DRegion	Any	Any	Any	0.00238893
Any	Any	TempRange	Any	0.00236933
CRegion	Any	Any	PrecSplit	0.00176968
LRegion	Any	Any	PrecSplit	0.0017268
LRegion	Any	TempSplit	Any	0.00150721
CRegion	Any	TempSplit	Any	0.00146328
CRegion	Any	Any	PrecRange	0.000773255

Table 6. Top Ranked Summaries After Run 5

Station	Date	Temperature	Precipitation	Variance
CRegion	Any	Any	Any	0.00399823
Any	Any	Any	PrecRange	0.00291082
Any	Any	TempRange	Any	0.00236933
DRegion	Any	Any	Any	0.00135919
CRegion	Any	Any	PrecSplit	0.00120389
CRegion	Any	TempSplit	Any	0.00101855
Any	Any	TempSplit	PrecRange	0.000669804
Any	Any	TempRange	PrecSplit	0.000587879
CRegion	Any	Any	PrecRange	0.000570345
Any	Season	TempRange	Any	0.000515874

Table 7. Top Ranked Summaries After Run 6.

6. Conclusion

We have outlined an approach to summarization, a type of data mining that aggregates data in a variety of ways. Our approach is based on ExGen graphs, and it is well suited for domains where attributes play a crucial role due to the complexity of background knowledge about these types of attributes. To reduce the tedious work of specifying expectations for all nodes and to keep consistency among all the nodes, we present a method for propagating expectations in ExGen Graphs. Due to the exponential nature of multi-attributes, we also proposed node-pruning methods to reduce the search space. Experiments with DGG-Discover 5.0 on Saskatchewan weather demonstrate the interactive process of our method and show that it provides a step by step means of finding the facets of the data that are interesting in light of previously known or discovered knowledge.

References

[1] Bay, S.D., and Pazzani, M.J., Detecting group differences: mining contrast sets. *Data Mining and Knowledge Discovery*, 5, 213-246, 2001.

[2] Cai, Y., Cercone, N., and Han J., Attribute-oriented Induction in Relational Databases. In Piatetsky-Shapiro, G. and Frawley, W.J. (eds), *Knowledge Discovery in Databases*, AAAI Press, 1991, 213-228.

[3] Fayyad, U.M., Piatetsky-Shapiro, G., and Smyth, P., From Data Mining to Knowledge Discovery: An Overview. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, R., and Uthurusamy, R. (eds), *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996, 1-34.

[4] Hamilton, H.J., Hilderman, R.J., and Cercone, N.. Attribute-oriented Induction Using Domain Generalization Graphs. In *Proc. Eighth IEEE International Conference on Tools with Artificial Intelligence (ICTAI'96)*, 246-253, Toulouse, France, November 1996.

[5] Hilderman, R.J. and Hamilton, H.J., Heuristic Measures of Interestingness. In *Proc. PKDD'99*, 232-241, Prague, September 1999.
[6] Hilderman, R.J. and Hamilton, H.J., *Knowledge Discovery and Measures of Interest*, Kluwer Acadamic, 2001.
[7] Hilderman, R. J., Hamilton, H. J., and Cercone, N., Data Mining in Large Databases using Domain Generalization Graphs, *Journal of Intelligent Information Systems*, 13:195-234, 1999.
[8] Liu, B., Hsu, W., Chen, S., and Ma, Y., Analyzing the Subjective Interestingness of Association Rules, *IEEE Intellgent Systems*, 15 (5) : 47-55, 2000.
[9] Zytkow, J., From Contingency Tables to Various Forms of Knowledge in Databases. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, R., and Uthurusamy, R., *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996, 329-349.