Data Visualization in the DB-Discover System

Robert J. Hilderman, Liangchun Li, and Howard J. Hamilton
Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada, S4S 0A2
{hilder,lil,hamilton}@cs.uregina.ca

Abstract

The DB-Discover system is a research software tool for knowledge discovery from databases. Utilizing an efficient attribute-oriented induction algorithm based upon the climbing tree and dropping condition methods, called multi-attribute generalization, it generates many possible summaries of data from a database. In this paper, we present the design of the data visualization capabilities of DB-Discover. We describe how our data visualization techniques help us manage the many possible summaries and how potentially interesting summaries are selected from them. A prototype system has been implemented which allows many alternative visualizations of the summaries. The results of our work enable a domain expert to quickly and efficiently analyze the contents of a database from many different perspectives.

1 Introduction

The design of the data visualization capabilities of the DB-Discover system is presented. *DB-Discover* is a research software tool for knowledge discovery from databases (KDD) developed at the University of Regina from 1992 to date [2, 4, 5]. Data visualization can be used in a KDD system at three stages: (1) to guide discovery, (2) to guide the presentation of the results, and (3) to present the results themselves. Here we focus on the second stage.

KDD algorithms can be broadly classified into two general areas, summarization and anomaly detection. *Summarization algorithms* find concise descriptions of data, such as high level summaries, sets of rules, or decision trees. *Anomaly detection algorithms* identify unusual features of data, such as combinations that occur with greater or lesser frequency than expected. DB-Discover produces many possible summaries of data retrieved from a database. We describe techniques for (1) managing the many possible summaries available for output, and (2) selecting the anomalous (and potentially interesting) summaries among them. As well, techniques for guiding the discovery process are described.

The remainder of this paper is organized as follows. In the following section, we describe DB-Discover. In Section 3, we show how data visualization can help a domain expert efficiently manage and analyze discovered information. In Section 4, we summarize our results and suggest future research.

2 The DB-Discover System

The DB-Discover system is based on *multi-attribute generalization*, an enhanced version of a well-known data mining technique called attribute-oriented induction [1]. *Attribute-oriented induction* summarizes data in a database according to user-defined concept hierarchies associated with relevant attributes. In this section, we describe, in general terms, how DB-Discover works to generate summaries from data in a database. First, in Section 2.1, we informally describe attribute-oriented induction and present an example showing how a concept hierarchy can be ascended to produce a summary. Then in Section 2.2, we describe domain generalization graphs, the primary data structure used in multi-attribute generalization, and show how they are used to extend the scope of the attribute-oriented induction methodology. Finally, in Section 2.3, we describe the process of multi-attribute generalization, whereby domain generalization graphs are used to guide the generalization of a set of attributes.

2.1 Attribute-Oriented Induction

Transforming a specific data description into a more general one is called *generalization*. Generalization techniques include the dropping condition and climbing tree methods [14]. The *climbing tree method* transforms the data in a database by repeatedly replacing specific attribute values with more general concepts according to user-defined concept hierarchies. A *concept hierarchy* (CH) associated with an attribute in a database

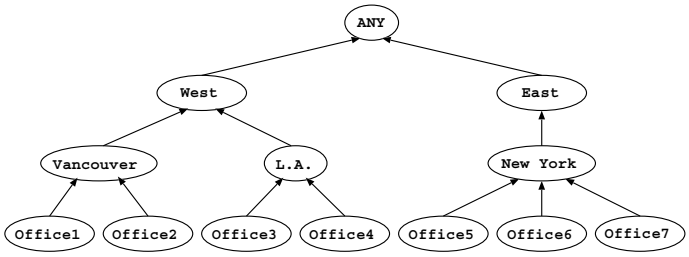

Figure 1. A CH for the *Office* attribute

is represented as a tree where leaf nodes correspond to actual domain values in the database, intermediate nodes correspond to a more general representation of the domain values, and the root node corresponds to the most general representation of the domain values. For example, a CH for the *Office* attribute in a sales database is shown in Figure 1. Knowledge about the higher level concepts (i.e., non-leaf nodes) can be discovered by generating the summaries corresponding to each level of the CH.

The *dropping condition method* transforms the data in a database by removing a condition from a conjunction of conditions, so that the remaining conjunction of conditions is more general. For example, assume the conjunction of conditions ($shape$ ="round"$\wedge size$ ="large"$\wedge colour$ ="white") describes the concept *ball*. Removing the condition $colour$ ="white", which is equivalent to generalizing the *Colour* attribute to *ANY*, yields the conjunction of conditions ($shape$ ="round"$\wedge size$ ="large"). The concept ball is now more general because it encompasses large round objects of any colour.

Attribute-oriented induction (AOI) [9, 10, 11] is a summarization algorithm that integrates the climbing tree and dropping condition methods for generalizing data in a database. An efficient variant of the AOI algorithm has been incorporated into DB-Discover. By associating a CH with an attribute, DB-Discover can generalize and summarize the domain values for the attribute in many different ways. An attractive feature of DB-Discover is that it is possible to obtain many different summaries for an attribute by simply changing the structure of the associated CH, and these new summaries can be obtained without modifying the underlying data.

For example, consider the database shown in Table 1. Using the CH from Figure 1 to guide the generalization, one of the many possible summaries which can be generated is shown in Table 2, where the *Office*, *Quantity*, and *Amount* attributes have been selected for generalization, and the actual values for the *Office* attribute in each tuple have been generalized to the level of *West* and *East*. The values in the *Quantity* and *Amount* attributes have been accumulated accordingly and a new attribute, called *Count*, shows the number of specific tuples accumulated in each generalized tuple.

Table 1. The sales database

Office	*Shape*	*Size*	*Colour*	*Quantity*	*Amount*
2	round	small	white	2	\$50.00
5	square	medium	black	3	\$75.00
3	round	large	white	1	\$25.00
7	round	x-large	black	4	\$100.00
1	square	x-large	white	3	\$75.00
6	round	medium	black	4	\$100.00
4	square	small	white	2	\$50.00

Table 2. An example sales summary

Office	*Quantity*	*Amount*	*Count*
West	8	\$200.00	4
East	11	\$275.00	3

As a result of recent research, the AOI variant in DB-Discover is among the most efficient of KDD methods [2, 3, 4, 5, 9, 13]. In particular, algorithms for generalizing relational databases are presented in [2] that run in $O(n)$ time, where n is the number of tuples in the input relation, and require $O(p)$ space, where p is the number of tuples in the generalized relation (typically $p << n$), while producing exactly the same output as created by previous algorithms. In [2], it is also proven that an AOI algorithm which runs in $O(n)$ time is optimal for the defined generalization problem.

2.2 Domain Generalization Graphs

A fundamental problem with AOI methods, in general, is that they are limited in scope. That is, where multiple CHs are associated with an attribute, the user is required to select just one during a discovery task. Consequently, only those summaries which can be generated from the chosen CH are presented to the user, ignoring the relative merits of other possible summaries that could be generated from the other CHs.

To facilitate the generation of other possible summaries, *domain generalization graphs* (DGGs) are used to augment CHs and expand the scope of our AOI methods [12, 8, 16]. Informally, a DGG associated with a CH defines a partial order which represents a set of generalization relations for the attribute. A DGG always includes a single source (i.e., the node at the lowest level corresponding to the domain of the attribute) and a single sink (i.e., the node at the highest level corresponding to the most general representation of the domain and which contains the value ANY). The node at each level in a DGG is a general description of the

nodes at the same level in the corresponding CH. For example, the nodes at each level of the CH in Figure 1 correspond to the nodes at the same level in the more general representation of the DGG in Figure 2. That is, *West* and *East* correspond to *Division*, *Vancouver*, *L.A.* and *New York* correspond to *City*, and *Office1* to *Office7* correspond to *Office*. Note that a CH always corresponds to a single-path DGG.

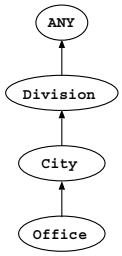

Figure 2. A DGG for the *Office* attribute

When there are multiple CHs associated with a single attribute, meaning knowledge about an attribute can be expressed in many different ways, a multi-path DGG can be constructed from the single-path DGGs. Figure 3 shows a multi-path DGG on the right that has been constructed from the two single-path DGGs on the left. Here we assume that a common name used in the single-path DGGs associated with the attribute represents the same partition of the domain in the associated CHs. For example, in Figure 3, the *ANY*, *City*, and *Office* nodes in the single-path DGGs on the left represent the same partition of the domain in the associated CHs. Consequently, the like-named nodes can be combined in the multi-path DGG.

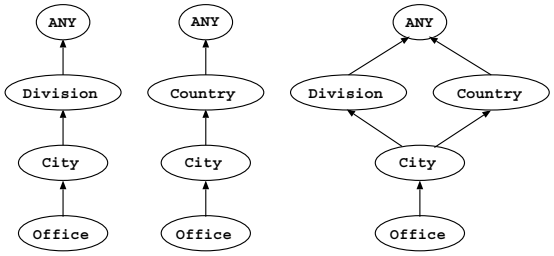

Figure 3. A multi-path DGG for the *Office* attribute

DB-Discover traverses a multi-path DGG to generate the minimum number of summaries for an attribute in a space efficient manner. For example, assume that we want to generate a summary at the *City* level of the multi-path DGG in Figure 3. The *Office* node represents the domain for the *Office* attribute. To generate this summary, either of the CHs associated with the original single-path DGGs could be selected to guide the generalization. Since the *City* node represents the same partition of the domain in the associated CHs, it does not matter to DB-Discover which CH is chosen to obtain this summary. If DB-Discover generated summaries by following the two single-path DGGs independently, then six summaries would be generated, of which two would be duplicate summaries corresponding to the *City* and *ANY* nodes. The other two summaries would correspond to the *Division* and *Country* nodes. However, by following the multi-path DGG, it only generates four summaries corresponding to the *City*, *Division*, *Country*, and *ANY* nodes.

2.3 Multi-Attribute Generalization

A typical discovery task usually involves generalization of a set of attributes, where each attribute is associated with a multi-path DGG. When generalizing a set of attributes, the set can be considered a single attribute whose domain is the cross product of the individual attribute domains. For example, given the domains for the attributes *Shape*, *Size*, and *Colour* shown in Table 3, the domain for the compound attribute *Shape-Size-Colour* is as shown in Table 4.

Table 3. Domains for the *Shape*, *Size*, and *Colour* attributes

Shape	*Size*	*Colour*
round	small	black
square	medium	white
	large	
	x-large	

Table 4. Domain for the compound attribute *Shape-Size-Colour*

Shape-Size-Colour
round-small-black
round-small-white
round-medium-black
round-medium-white
round-large-black
round-large-white
round-x-large-black
round-x-large-white
square-small-black
square-small-white
square-medium-black
square-medium-white
square-large-black
square-large-white
square-x-large-black
square-x-large-white

A generalization from the domain for the *Shape-Size-Colour* attribute is a combination of nodes from

the DGGs associated with the individual attributes, taking one node from the DGG for each attribute. For example, consider again the database shown in Table 1. The *Shape*, *Size*, *Colour*, and *Quantity* attributes have been selected for generalization. Using the CHs from Figure 4 for the *Shape*, *Size*, and *Colour* attributes, where two CHs are shown for the *Size* attribute, and the associated DGGs shown in Figure 5, one of the many possible summaries which can be obtained by generalizing to the DGG node combination *ANY-Package-Colour* is shown in Table 5. We use the climbing tree method to obtain the generalized values for the *Shape* and *Size* attributes. Since the value for the *Shape* attribute now covers all possible values, we could use the dropping condition method to remove the *Shape* column from our summary without losing any meaningful information.

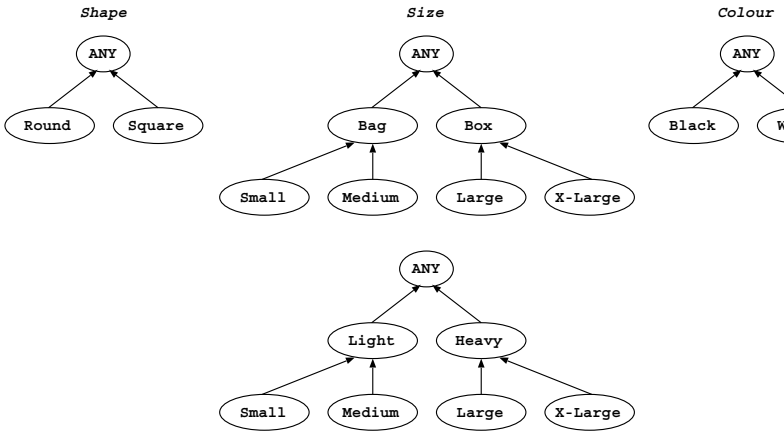

Figure 4. CHs for the *Shape*, *Size*, and *Colour* attributes

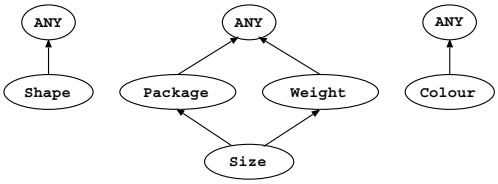

Figure 5. DGGs for the *Shape*, *Size*, and *Colour* attributes

The *generalization state space* consists of all possible summaries which can be generated from the DGGs associated with the set of attributes. This space is more general than a version space ([15]) because we allow more complex generalizations than value-to-ANY and our summaries contain a relation rather than a single conjunctive description. On the other hand, it is more constricted than a Galois lattice ([7]) because not every possible subset is included in the lattice. The generalization state space is obtained by determining all possible combinations of nodes from the DGGs, and then generating the summary which corresponds to each node combination. For example, the generalization state space for the compound attribute *Shape-Size-Colour* example consists of $2 \times 4 \times 2 = 16$ summaries, where 2, 4, and 2 are the number of nodes in the DGG associated with the *Shape*, *Size*, and *Colour* attributes, respectively. The two summaries corresponding to the node combinations *Shape-Size-Colour* and *ANY-ANY-ANY* are considered trivial. In the *Shape-Size-Colour* summary, all of the attributes are ungeneralized (i.e., as in the original database), and in the *ANY-ANY-ANY* summary, all of the attribute values are generalized to the value ANY. The size of the generalization state space depends only on the number of nodes in the associated DGGs; it is not dependent upon the number of tuples in the input relation. For m attributes, a database of n tuples, and an $O(n)$ generalization algorithm, creating all possible summaries requires $O(n \prod_{i=1}^{m} |D^i|)$ time, where $|D^i|$ is the number of nodes in DGG D^i. We have developed and implemented practical serial and parallel algorithms for traversing the generalization state space where m is small (≤ 5) and n is large ($>$ 1,000,000) [8, 12].

Table 5. Summary for the DGG node combination *ANY-Package-Colour*

Shape	*Size*	*Colour*	*Quantity*	*Count*
ANY	bag	white	4	2
ANY	bag	black	7	2
ANY	box	white	4	2
ANY	box	black	4	1

3 Data Visualization in DB-Discover

In the example in the previous section, the DGGs used were of low complexity, and few summaries were generated. In practical cases, multi-attribute generalization has the potential to generate many summaries. This is because the number of attributes to be generalized could be large or the DGGs associated with the attributes could be complex. Our experience in applying data mining techniques to the databases of our commercial partners has shown that domain experts typically perform AOI on a few attributes for which considerable domain knowledge has been accumulated. This domain knowledge can be translated into a complex set of multi-path DGGs.

To assist the domain expert, we propose a tool, with a graphical user interface, called DBD-View. *DBD-View* ranks and arranges summaries, according to various user-defined criteria, so that they can be quickly and efficiently analyzed. Our approach is based upon a simple, yet powerful architecture which uses different views for visualizing the summaries generated by a discovery task. In this section, we describe DBD-View and show how it works to manage summaries. First, in Section 3.1, we describe the window upon which views are placed, known as the *viewspace*. Second, in Section 3.2, we describe the method used for ranking summaries according to their size and interestingness. Third, in Section 3.3, we describe the global view. The *global view* is a visualization which shows a representation of the complete generalization state space. Next, in Section 3.4, we describe the local view. The *local view* is a visualization which shows a representation of the generalization state space which allows it to be broken down into smaller components. Then, in Section 3.5, we describe the summary view. The *summary view* is a visualization which shows a representation of a single summary. Finally, in Section 3.6, we describe other functionality provided by DBD-View which allows a domain expert to change the default settings.

3.1 The Viewspace

The viewspace is a window upon which up to three different views may be placed. The screen snapshot in Figure 6 shows a viewspace that has been filled with a global view (top left), a local view (bottom left), and a summary view (right). We describe the options available for displaying these views in the sections that follow.

Figure 6. The viewspace showing global, local, and summary views

3.2 Ranking Summaries

We rank the interestingness of the summaries generated by DB-Discover using measures based upon relative entropy and variance. *Relative entropy*, also known as the *Kullback-Leibler (KL) distance*, has been suggested as an appropriate measure for comparing data distributions in unstructured textual databases [6]. Here we use KL-distance to compare the distribution defined by the structured data in a summary to that of a model distribution of the data. *Variance*, the most common measure of variability used in statistics, is used to compare the distribution defined by the data in a summary to that of a uniform or expected distribution of the data. Ranking summaries using these interestingness heuristics allows us to reduce, or prune, the number of summaries which must be considered by the domain expert.

As a result of previous work ([8]), we believe that an important property of interestingness measures should be to rank summaries of low complexity (i.e., those with few attributes and/or few tuples) as more interesting. The variance heuristic has this property. Low complexity summaries are attractive because they are more concise and intuitively easier to analyze and understand.

3.3 The Global View

There are two types of global views which are used to represent the complete generalization state space: a scatterplot global view and a spatial global view. The *scatterplot global view* represents the generalization state space as a two-dimensional graph, where the number of tuples in a summary is plotted against the interestingness of the summary. For example, the screen snapshot in Figure 7 shows the scatterplot global view of the complete generalization state space for a small discovery task. Each dot on the graph corresponds to one of the summaries in the generalization state space. A summary corresponding to a dot to the

left of the graph is considered less interesting than one to the right of the graph. A summary corresponding to a dot near the top of the graph contains more tuples than one near the bottom of the graph.

Figure 7. The scatterplot global view

The *spatial global view* represents the generalization state space as a directed graph consisting of nodes, of various sizes and colours, connected by edges. The relative size of a node is representative of the number of tuples in the corresponding summary. That is, the larger the node, the larger the corresponding summary. Similarly, the colour of a node is representative of the interestingness of the corresponding summary. For example, the screen snapshot in Figure 8 shows the spatial global view of the complete generalization state space for a small discovery task. Since node *D2* is larger than node *D3*, then node *D2* corresponds to a summary containing more tuples. Also, since node *D3* is lighter than node *D2*, then node *D3* corresponds to a summary of higher interest.

The domain expert can utilize the scatterplot and spatial global views to obtain an easy-to-understand, initial overview analysis of the summaries in the generalization state space. If a particular summary seems interesting, he can left click on a dot (in scatterplot global view) or node (in spatial global view) to make the corresponding summary the active summary. When

Figure 8. The spatial global view

a summary becomes the *active summary*, the local and summary views are also updated to coincide with the new global view. If the domain expert would like to save a particular summary, he can left double-click on a dot or node to display the corresponding summary in a new summary view window that is separate from the viewspace. The summary view shown in this window is displayed unchanged until the window is closed by the domain expert (i.e., it does not change when the active summary changes).

3.4 The Local View

There are two types of local views which are used to dissect the generalization state space: a separate graphs local view and a spatial local view. The *separate graphs local view* represents a summary in the generalization state space as a combination of nodes from the DGGs associated with the attributes being generalized. For example, the screen snapshot in Figure 9 shows the three DGGs from which the summaries in the generalization state space were generated. There is one solid (highlighted) node in each of these DGGs where the DGG node combination coincides with the active summary in the global and summary views.

The domain expert can navigate throughout the

Figure 9. The separate graphs local view

generalization state space by selecting different combinations of nodes from the separate graphs local view. If a particular summary seems interesting, he can left click one node in each DGG to highlight it and change the active summary to that which corresponds to the selected DGG node combination. When a summary becomes the active summary, the global and summary views are also updated to coincide with the new local view.

The *spatial local view* represents a logically connected group of summaries in the generalization state space. For example, the screen snapshot in Figure 10, shows a neighborhood in the generalization state space immediately surrounding the node corresponding to the active summary. The node corresponding to the active summary is the single highlighted node at the centre of the view, and is called the *current node*. Below the current node are all nodes in the generalization state space from which it is possible to reach the current node by generalizing one attribute up one more level in the associated DGG. These nodes are a more specific representation of the current node. Above the current node, are all nodes in the generalization state space which can be reached from the current node by generalizing one attribute up one more level in the associated DGG. These nodes are a more general representation of the current node. The labels in the nodes identify the attribute that has been generalized (for those nodes below the current node) or will be generalized (for those nodes above the current node).

Figure 10. The spatial local view

The domain expert can navigate throughout the generalization state space by selecting a node from the spatial local view. If a particular summary seems interesting, he can left click on the corresponding node to highlight it and change the active summary to that which corresponds to the active node. The spatial local view is then updated to show the selected node as the current node (i.e., by moving it to the centre of the view). When a node becomes the active node, the global and summary views are also updated to coincide with the new local view.

3.5 The Summary View

The default output format for displaying the summary view is to display it as a table. A sample table is shown in Figure 11. Alternative output formats include bar charts, column charts, line charts, pie charts, comma-delimited files, and tab-delimited files. Comma- and tab-delimited files allow the output to be used as input to commonly used spreadsheet packages.

Figure 11. The tabular summary view

3.6 Changing the Default Settings

The default view settings for displaying the results of a discovery task are *scatterplot*, *separate graphs*, and *tabular* for the global, local, and summary views, respectively. The default interestingness heuristic is *variance from uniform*, with *pruning up and down* using the *less than or equal* comparator. The domain expert can change the default settings for DBD-View to change the way in which views are displayed and summaries are ranked. These settings can be accessed through the *View* and *Options* drop-down menus shown on the menu bar of Figure 6. The *View* drop-down menu is also available as a context-sensitive menu, by right-clicking anywhere in the viewspace, as shown on the scatterplot global view in Figure 7. The *Options* menu is not shown.

The *View* menu has settings for the three views corresponding to the summary, local, and global views. Currently selected settings are indicated by a check mark. To change a view setting, double-click the mouse pointer on the required setting. This removes the check mark from the previously selected setting and places a check mark at the newly selected setting. A check mark can be toggled off and on by clicking the mouse pointer on the currently selected setting. If no setting is selected for a particular view, that view will be removed from the viewspace. If no setting was previously selected for a particular view, then that view will be added to the viewspace.

The *Options* menu has settings for the interestingness heuristic, the pruning direction, and the pruning comparator. These settings can be changed in the same manner as the *View* settings described above.

Interestingness heuristics settings include *variance from uniform*, *relative entropy*, and *variance from expected*. When an alternative interestingness heuristic is selected, the summaries do not need to be regenerated. The views are simply redisplayed according to the chosen heuristic. That is, no additional recalculation is required because the ranking for each interestingness heuristic is calculated and stored at the time a summary is generated.

The pruning direction settings include *pruning up*, *pruning down*, and *pruning up and down*. The pruning comparator settings include *none*, *less than*, and *less than or equal*. Pruning reduces the number of summaries which must be considered by the domain expert in some circumstances. For example, if a node is a direct descendant of some other node in the generalization state space, but the corresponding summary has higher interest, then the ancestor can be eliminated from further consideration. For example, nodes *D2*, *D3*, and *D4* are all direct descendants of node *D1*. Since *D2*, *D3*, and *D4* also have higher interest according to the present measure, *D1* can be ignored. Similar pruning can be done in the situation where a node is a direct descendant of a node that has higher interest. For example, node *D5* is a direct descendant of node *D2*. Since *D2* has higher interest, *D5* can be ignored.

Pruning can also be done based upon the table threshold, attribute threshold, and interest threshold. When pruning using the table threshold, all nodes whose corresponding summaries contain more tuples than the table threshold will be pruned. When pruning using the attribute threshold, all nodes whose corresponding summaries contain an attribute where the number of distinct values remains greater than the attribute threshold will be pruned. And finally, when pruning using the interest threshold, all nodes whose corresponding summaries have interestingness less than the interest threshold will be pruned.

4 Conclusion and Future Work

We have presented the design of the data visualization capabilities of the DB-Discover system. Utilizing multi-attribute generalization, DB-Discover generates all possible summaries from the DGGs associated

with a set of attributes. We showed how data visualization techniques can be used in a KDD system to manage the summaries generated and to select potentially interesting summaries for further analysis. The primary benefit from our approach is that a domain expert can quickly and efficiently analyze the contents of a database from many different perspectives.

Our experience has shown that this is a promising approach toward greater understanding of discovered knowledge. However, future work will involve enhancing the data visualization capabilities of DB-Discover. For example, more work needs to be done to refine the interestingness and pruning heuristics. Also, techniques for displaying and navigating extremely large generalization state spaces are required. Finally, techniques for allowing the domain expert to interactively guide a discovery task will be investigated.

References

[1] Y. Cai, N. Cercone, and J. Han. Attribute-oriented induction in relational databases. In G. Piatetsky-Shapiro and W. Frawley, editors, *Knowledge Discovery in Databases*, pages 213–228, Cambridge, MA, 1991. AAAI/MIT Press.

[2] C.L. Carter and H.J. Hamilton. Efficient attribute-oriented algorithms for knowledge discovery from large databases. IEEE Trans. on Knowledge and Data Engineering. To appear.

[3] C.L. Carter and H.J. Hamilton. Fast, incremental generalization and regeneralization for knowledge discovery from databases. In *Proceedings of the 8th Florida Artificial Intelligence Symposium*, pages 319–323, Melbourne, Florida, April 1995.

[4] C.L. Carter and H.J. Hamilton. A fast, on-line generalization algorithm for knowledge discovery. *Applied Mathematics Letters*, 8(2):5–11, 1995.

[5] C.L. Carter and H.J. Hamilton. Performance evaluation of attribute-oriented algorithms for knowledge discovery from databases. In *Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence (ICTAI'95)*, pages 486–489, Washington, D.C., November 1995.

[6] R. Feldman and I. Dagan. Knowledge discovery in textual databases (KDT). In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 112–117, Montreal, August 1995.

[7] R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on galois (concept) lattices. *Computational Intelligence*, 11(2):246–267, 1995.

[8] 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.

[9] J. Han. Towards efficient induction mechanisms in database systems. *Theoretical Computer Science*, 133:361–385, October 1994.

[10] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: an attribute-oriented approach. In *Proceedings of the 18th International Conference on Very Large Data Bases*, pages 547–559, Vancouver, August 1992.

[11] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. on Knowledge and Data Engineering*, 5(1):29–40, February 1993.

[12] R.J. Hilderman, H.J. Hamilton, R.J. Kowalchuk, and N. Cercone. Parallel knowledge discovery using domain generalization graphs. In J. Komorowski and J. Zytkow, editors, *Proceedings of the 1st European Conference on the Principles of Data Mining and Knowledge Discovery*, pages 25–35, Trondheim, Norway, June 1997.

[13] H.-Y. Hwang and W.-C. Fu. Efficient algorithms for attribute-oriented induction. In *Proceedings of the 1st International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pages 168–173, Montreal, August 1995.

[14] R.S. Michalski. A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 83–134. Tioga Publishing Company, 1983.

[15] T.M. Mitchell. *Version Spaces: An Approach to Concept Learning*. PhD thesis, Stanford University, 1978.

[16] W. Pang, R.J. Hilderman, H.J. Hamilton, and S.D. Goodwin. Data mining with concept generalization graphs. In *Proceedings of the Ninth Annual Florida AI Research Symposium*, pages 390–394, Key West, Florida, May 1996.