Substructure Discovery in the SUBDUE System

Lawrence B. Holder, Diane J. Cook, and Surnjani Djoko
Learning and Planning Laboratory
Department of Computer Science and Engineering
University of Texas at Arlington
Box 19015, Arlington, TX 76019
Email: holder@cse.uta.edu, cook@cse.uta.edu, djoko@cse.uta.edu

Abstract

Because many databases contain or can be embellished with structural information, a method for identifying interesting and repetitive substructures is an essential component to discovering knowledge in such databases. This paper describes the SUBDUE system, which uses the minimum description length (MDL) principle to discover substructures that compress the database and represent structural concepts in the data. By replacing previously-discovered substructures in the data, multiple passes of SUBDUE produce a hierarchical description of the structural regularities in the data. Inclusion of background knowledge guides SUBDUE toward appropriate substructures for a particular domain or discovery goal, and the use of an inexact graph match allows a controlled amount of deviations in the instance of a substructure concept. We describe the application of SUBDUE to a variety of domains. We also discuss approaches to combining SUBDUE with non-structural discovery systems.

1 Introduction

A number of researchers have developed techniques for discovering concepts in data expressed in a non-structural, attribute-value (flat) representation. However, many databases also include structural data describing the relationships among the data objects. In addition, current flat databases can be embellished with structural information. For example, flat satellite measurement data can be augmented with information about the temporal and proximate ordering of the measurements.

In response to these needs, we have been investigating techniques for structural data discovery. One method for discovering knowledge in structural data is the identification of common substructures within the data. The motivation for this process is to find substructures capable of compressing the data and to identify conceptually interesting substructures that enhance the interpretation of the data. Once discovered, the substructure concept can be used to simplify the data by replacing instances of the substructure with a pointer to the newly discovered concept. The discovered substructure concepts allow abstraction over detailed structure in the original data and provide new, relevant attributes for interpreting the data.

We describe a new version of our SUBDUE system [7, 8, 3] that discovers interesting substructures in structural data based on the minimum description length principle and optional background knowledge. The next section describes work related to substructure discovery. Sections 3 and 4 describe the graph encoding and inexact graph match used by SUBDUE, and Section 5 describes SUBDUE's discovery algorithm. Section 6 presents some results obtained with SUBDUE. Section 7 describes the background knowledge component of SUBDUE, and Section 8 shows how multiple iterations of SUBDUE can generate a hierarchy of discovered concepts.

Eventually, the SUBDUE system must be integrated with methods for discovery in flat data. Although SUBDUE can accept flat data as input, the discovery process has no model for the distribution of attribute values. Therefore, every value is distinct and unrelated to other values. Section 9 discusses approaches to integrating SUBDUE with non-structural discovery systems such as AUTOCLASS [1] which provide for the inclusion of distribution models. Section 10 summarizes our results with SUBDUE and discusses directions for future work.

2 Related Work

Although specific to the blocks-world domain, Winston's ARCH program [24] discovers substructure in order to deepen the hierarchical description of a scene and to group objects into more general concepts. Levinson [11] developed a system for storing labeled graphs in which individual graphs are represented by the set of vertices in a universal graph. Subgraphs of the universal graph used by several individual graphs suggest common substructure in the individual graphs. Segen [20] describes a system for storing graphs using a probabilistic graph model to represent classes of graphs. High-probability components of the model represent substructure common to the model's instances.

The LABYRINTH system [21] extends the COBWEB incremental conceptual clustering system [5] to form hierarchical concepts of the structured objects. The upper-level components of the structured-object hierarchy represent substructures common to the input objects. Conklin and Glasgow [2] have developed the I-MEM system for constructing an image hierarchy, similar to that of LABYRINTH, used for discovering common substructure in a set of images expressed in terms of a set of predefined relations. The CLIP system [25] for graph-based induction iteratively discovers patterns (substructures) in graphs by expanding and combining patterns discovered in previous iterations. CLIP addresses the computational complexity by estimating compression afforded by the substructures and using a linear-time approximation to graph isomorphism.

Many results in grammatical inference are applicable to constrained classes of graphs (e.g., trees) [6, 13], and discover graph-based production rules that can be interpreted as substructures common to the input graphs [9].

3 Encoding Graphs

The minimum description length (MDL) principle introduced by Rissanen [19] states that the best theory to describe a set of data is that theory which minimizes the description length of the entire data set. The MDL principle has been used for decision tree induction [17], image processing [10, 14, 15], concept learning from relational data [4], and learning models of non-homogeneous engineering domains [18].

We demonstrate how the minimum description length principle can be used to discover substructures in complex data. In particular, a substructure is evaluated based on how well it can compress the entire dataset using the minimum description length. We define the minimum description length of a graph to be the number of bits necessary to completely describe the graph.

According to the minimum description length (MDL) principle, the theory that best accounts for a collection of data is the one that minimizes $I(S)+I(G|S)$, where S is the discovered substructure, G is the input graph, $I(S)$ is the number of bits required to encode the discovered substructure, and $I(G|S)$ is the number of bits required to encode the input graph G with respect to S.

The encoding of the graph consists of the following steps. The graph connectivity can be represented by an adjacency matrix, where a value of 1 in entry ij represents at least one edge from

i to j. We assume that that the original graph G has n vertices and that the decoder has a table of the l_u unique labels in G.

1. Determine the number of bits *vbits* needed to encode the vertex labels of the graph. First, we need $(\lg v)$ bits to encode the number of vertices v in the graph. Then, encoding the labels of all v vertices requires $(v \lg l_u)$ bits. We assume the vertices are specified in the same order they appear in the adjacency matrix. The total number of bits to encode the vertex labels is $vbits \ = \ \lg v + v \lg l_u$.

2. Determine the number of bits *rbits* needed to encode the rows of the adjacency matrix A. Typically, in large graphs, a single vertex has edges to only a small percentage of the vertices in the entire graph. Therefore, a typical row in the adjacency matrix will have much fewer than v 1s, where v is the total number of vertices in the graph. We apply a variant of the coding scheme used by Quinlan and Rivest [17] to encode bit strings with length n consisting of k 1s and $(n-k)$ 0s, where $k \ll (n-k)$. In our case, row i $(1 \leq i \leq v)$ can be represented as a bit string of length v containing k_i 1s. If we let $b = \max_i k_i$, then the i^{th} row of the adjacency matrix can be encoded as follows.

   (a) Encoding the value of k_i requires $\lg(b+1)$ bits.

   (b) Given that only k_i 1s occur in the row bit string of length v, only $C(v, k_i)$ strings of 0s and 1s are possible. Since all of these strings have equal probability of occurrence, $\lg C(v, k_i)$ bits are needed to encode the positions of 1s in row i. The value of v is known from the vertex encoding.

   Finally, we need an additional $\lg(b+1)$ bits to encode the number of bits needed to specify the value of k_i for each row. The total encoding length in bits for the adjacency matrix is $rbits \ = \ \lg(b+1) + \sum_{i=1}^{v} \lg(b+1) + \lg C(v, k_i)$.

3. Determine the number of bits *ebits* needed to encode the edges represented by the entries $A[i,j] = 1$ of the adjacency matrix A. The number of bits needed to encode entry $A[i,j]$ is $(\lg m) + e(i,j)[1 + \lg l_u]$, where $e(i,j)$ is the actual number of edges between vertex i and j in the graph and $m = \max_{i,j} e(i,j)$. The $(\lg m)$ bits are needed to encode the number of edges between vertex i and j, and $[1 + \lg l_u]$ bits are needed per edge to encode the edge label and whether the edge is directed or undirected. In addition to encoding the edges, we need to encode the number of bits $(\lg m)$ needed to specify the number of edges per entry. The total encoding of the edges is $ebits \ = \ \lg m + \sum_{i=1}^{v} \sum_{j=1}^{v} \lg m + e(i,j)[1 + \lg l_u]$.

The total encoding of the graph takes $(vbits + rbits + ebits)$ bits. Both the input graph and discovered substructure can be encoded using the above scheme. After a substructure is discovered, each instance of the substructure in the input graph is replaced by a single vertex representing the entire substructure. The labels of edges incident on the vertices of a substructure instance can be changed to indicate the specific vertex within the substructure involved in the edge. The discovered substructure is represented in $I(S)$ bits, and the graph after the substructure replacement is represented in $I(G|S)$ bits. SUBDUE searches the space of possible substructures, preferring substructures S needing less information $I(S) + I(G|S)$ to describe the graph G.

4 Inexact Graph Match

Although exact structure match can be used to find many interesting substructures, many of the most interesting substructures show up in a slightly different form throughout the data. These

SUBDUE(G,*limit*,*beam*)
    $D = \{\}$
    $S = \text{vertices}(G)$
    while (computation $<$ *limit*) and ($S \neq \{\}$)
        order S from best to worst using MDL and background knowledge rules
        $S =$ first *beam* substructures of S
        $b = \text{first}(S)$
        $D = D \cap \{b\}$
        $E = \{b$ extended by one edge in all possible ways $\}$
        $S = S \cap E$
    return D

Figure 1: SUBDUE's discovery algorithm.

differences may be due to noise and distortion, or may just illustrate slight differences between instances of the same general class of structures. Consider the image shown in Figure 3. The pencil and the cube would make ideal substructures in the picture, but an exact match algorithm may not consider these as strong substructures, because they rarely occur in the same form throughout the picture.

To allow a controlled amount of variation between instances of a substructure, we use an inexact graph match. In this algorithm, each distortion of a graph is assigned a cost. A distortion is described in terms of basic transformations such as deletion, insertion, and substitution of vertices and edges. The distortion costs can be determined by the user to bias the match for or against particular types of distortions.

An inexact graph match between two graphs g_1 and g_2 maps g_1 to g_2 such that g_2 is interpreted as a distorted version of g_1. Given a set of particular distortion costs, we define $matchcost(g_1, g_2)$ as the value of the least-cost function that maps graph g_1 onto graph g_2. Although there are an exponential number of mappings to consider, we reduce the complexity of the algorithm by performing a branch-and-bound search through the space of partial mappings. Whenever the number of substructure instances affects the substructure value, instances of a substructure can be weighted by their similarity to the substructure definition.

5 SUBDUE's Discovery Algorithm

Input to SUBDUE is in the form of a labeled, directed multigraph (two vertices may have more than one edge between them). A *substructure* is a connected subgraph within the graphical representation. An *instance* of a substructure in an input graph is a set of vertices and edges from the input graph whose match cost (according to the inexact graph match) to the graphical representation of the substructure is no more than a given threshold.

The substructure discovery algorithm used by SUBDUE is a computationally-constrained beam search (see Figure 1). The algorithm begins with the substructure matching a single vertex in the graph. Each iteration through the algorithm selects the best substructures and expands the instances of these substructures by one neighboring edge in all possible ways. The algorithm retains the best substructures in a list, which is returned when either all possible substructures have been considered or the total amount of computation exceeds a given limit. The evaluation of each

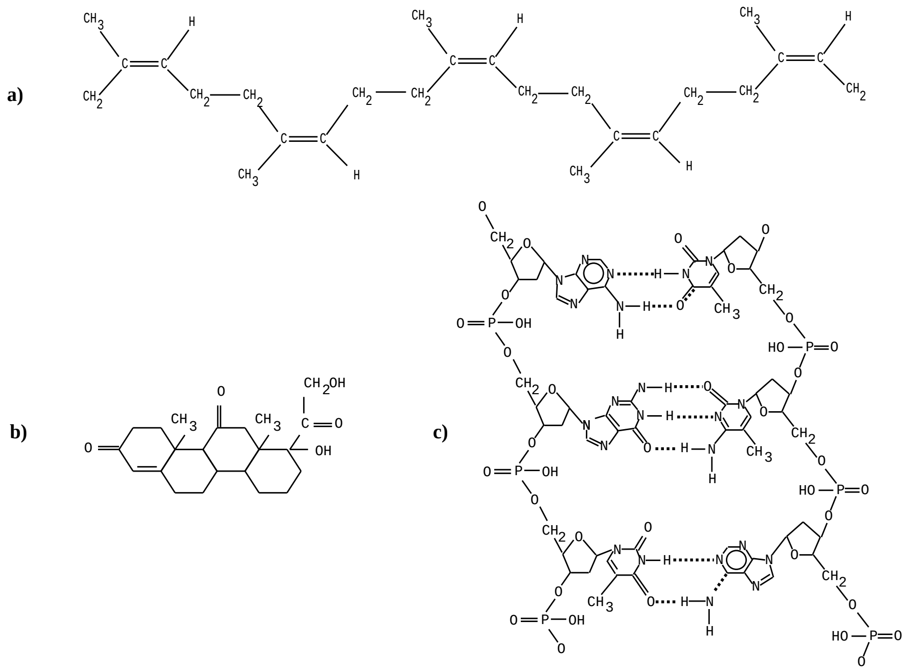

Figure 2: a) Natural rubber, b) Cortisone, c) Portion of a DNA molecule

substructure is guided by the minimum description length principle and background knowledge rules provided by the user (see Section 7).

6 Domains

The databases used in our experiments are taken from the domains of chemical compound analysis, scene analysis, CAD circuit analysis, and analysis of artificially-generated graphs. In the chemical compound domain, individual atoms are represented by vertices in the graph and bonds are represented by labeled edges in the graph. Figures 2 shows the graphs that were used for our experiments.

To apply SUBDUE to image data, we extract edge information from the image and construct a graph representing the scene. The vertex labels follow the Waltz labelings [22] of junctions of edges. An *edge* arc represents the edge of an object in the image, and a *space* arc links non-connecting objects together. Figure 3 shows the scene used for our experiments.

The data for the CAD circuit domain was obtained from National Semiconductor, and consists of a set of components making up a circuit as output by the Cadence Design System. The particular circuit used for this experiment is a portion of an analog-to-digital converter. In this domain, each component and interconnection is represented by a vertex labeled with the component name or T (interconnection). Each connection between components and between interconnections is represented by an arc.

In the final domain, we artificially generate graphs to evaluate SUBDUE's ability to discover substructures capable of compressing the graph. Four substructures are created of varying sizes and embedded in larger graphs whose size is 15 times the size of the substructure. The graphs vary across four parameters: number of possible vertex and edge labels, connectivity of the substructure, coverage of the instances, and the amount of distortion in the instances, totaling 96 graphs.

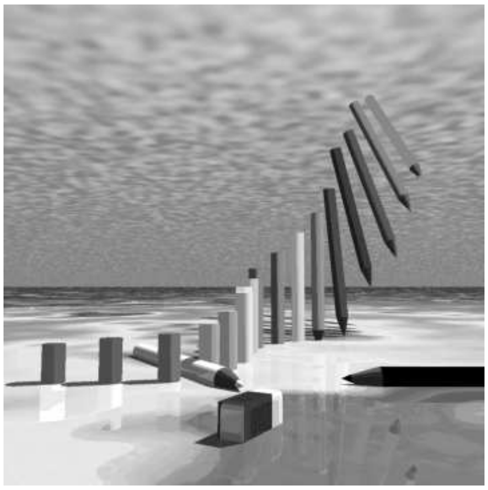

Figure 3: Scene analysis example.

Database	DL_{orig}	Threshold	DL_{comp}	Compression
Rubber	371.78	0.1	95.20	0.26
Cortisone	355.03	0.3	173.25	0.49
DNA	2427.93	1.0	2211.87	0.91
Pencils	1592.33	1.0	769.18	0.48
CAD – M1	4095.73	0.7	2148.8	0.52
CAD – S1SegDec	1860.14	0.7	1149.29	0.62
CAD – S1DrvBlk	12715.12	0.7	9070.21	0.71
CAD – BlankSub	8606.69	0.7	6204.74	0.72
CAD – And2	427.73	0.1	324.52	0.76
Artificial (avg of 96 graphs)	1636.25	0.0...1.0	1164.02	0.71

Table 1: Graph compression results.

In our first experiment, we test SUBDUE's ability to compress a structural database. We applied the discovery algorithm to each of the databases mentioned above. All subgraphs with a matchcost less than or equal to a specified threshold are included as instances of a substructure. We repeat the experiment with match thresholds ranging from 0.0 to 1.0 in increments of 0.1. Table 1 lists the results of this experiment. Compression is defined as $\frac{\text{DescriptionLength of compressed graph}}{\text{DescriptionLength of original graph}}$. These results demonstrate that even a single substructure discovered by SUBDUE can significantly reduce the amount of data needed to represent an input graph.

7 Background Knowledge

Although the principle of minimum description length is useful for discovering substructures that maximize compression of the data, scientists may realize more benefit from the discovery of substructures that exhibit other domain-specific and domain-independent characteristics.

To make SUBDUE more powerful across a wide variety of domains, we have added the ability to guide the discovery process with background knowledge. Although the minimum description length principle still drives the discovery process, the background knowledge can be used to input a bias toward certain types of substructures. This background knowledge is encoded in the form of rules for evaluating substructures, and can represent domain-independent or domain-dependent rules. Each time a substructure is evaluated, these input rules are used to determine the value of the substructure under consideration. Because only the most-favored substructures are kept and expanded, these rules bias the discovery process of the system.

Each background rule can be assigned a positive, zero, or negative weight, that biases the procedure toward a type of substructure, eliminates the use of the rule, or biases the procedure away from a type of substructure, respectively. The value of a substructure is defined as the description length of the input graph using the substructure multiplied by the weighted value of each background rule from a set of rules R applied to the substructure.

$$value(s) = DL(G, s) \times \prod_{r=1}^{|R|} \text{rule}_r(s)^{w_r}$$

Two of the domain-independent heuristics that have been incorporated as rules into the SUBDUE system are compactness and coverage. The first rule, *compactness*, is a generalization of Wertheimer's *Factor of Closure*, which states that human attention is drawn to closed structures [23]. A closed substructure has at least as many edges as vertices, whereas a non-closed substructure has fewer edges than vertices [16]. Compactness is thus defined as the ratio of the number of edges in the substructure to the number of vertices in the substructure.

The second rule, *coverage*, measures the fraction of structure in the input graph described by the substructure. The coverage rule is motivated from research in inductive learning and provides that concept descriptions describing more input examples are considered better [12]. Although the MDL principle measures the amount of structure, the coverage rule includes the relevance of this savings with respect to the size of the entire input graph. Coverage is defined as the number of unique vertices and edges in the instances of the substructure divided by the total number of vertices and edges in the input graph.

Domain-dependent rules can also be used to guide the discovery process in a domain where scientists can contribute their expertise. For example, circuit components can be classified according to their passivity. A component is said to be passive if it never delivers a net amount of energy to the outside world. A component which is not passive is said to be active. The *active component* rule favors substructures containing an active component. Once the active components are selected by SUBDUE, they can be compressed and attention can be focused on the passive components. This rule could also be weighted negatively to exclude substructures with active components. Similarly, the *loop* analysis rule favors subcircuits containing loops. A loop is defined here as a closed path whose starting vertex is the same as its ending vertex.

The substructure affording the most compression will not always be the most interesting or important substructure in the database. However, the additional background rules can be used to increase the chance of finding interesting substructures in these domains. In the case of the cortisone compound, we might be interested in finding common closed structures such as benzene rings. Therefore, we give a strong weight (8.0) to the *compactness* background rule and use a match threshold of 0.2 to allow for deviations in the benzene ring instances. In the resulting output, SUBDUE finds the benzene ring shown in Figure 4a.

In the same way, we can use the background rules to find the pencil substructure in the image data. We know that the pencils have a high degree of closure and of coverage, so the weights for these rules are set to 1.0. With these weights, SUBDUE is able to find the pencil substructure shown in Figure 4b for all tested match thresholds between 0.0 and 1.0.

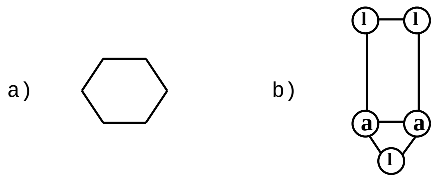

Figure 4: Results of Subdue guided by background knowledge

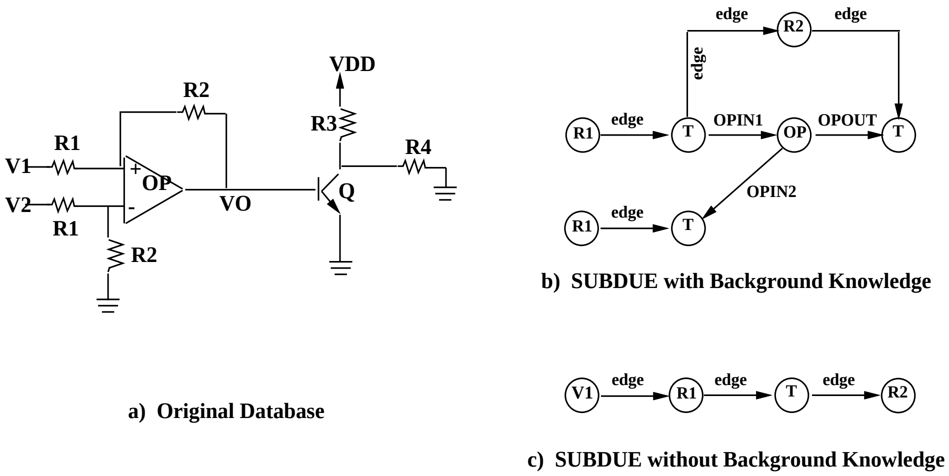

Figure 5: CAD circuit and discovered substructures

As a final example, the circuit shown in Figure 5a is passed to SUBDUE with a weight of 1.0 for the *active component* rule and a weight of 3.0 for the *loop* rule. The substructure discovered by SUBDUE is identified as a difference amplifier, whereas the substructure discovered by SUBDUE using no background knowledge exhibits now known functionality.

8 Hierarchical Concept Discovery

After a substructure is discovered, each instance of the substructure in the input graph can be replaced by a single vertex representing the entire substructure. The discovery procedure can then be repeated on the compressed data set, resulting in new interesting substructures. If the newly-discovered substructures are defined in terms of existing substructure concepts, the substructure definitions form a hierarchy of substructure concepts.

Hierarchical concept discovery also adds the capability to improve SUBDUE's performance. When SUBDUE is applied to a large input graph, the complexity of the algorithm prevents consideration of larger substructures. Using hierarchical concept discovery, SUBDUE can first discover those smaller substructures which best compress the data. Applying the compression reduces the graph to a more manageable size, increasing the chance that SUBDUE will find the larger substructures on the subsequent passes through the database.

Once SUBDUE selects a substructure, all vertices that comprise the exact instances of the substructure are replaced in the graph by a single vertex representing the discovered substructure. Edges connecting vertices outside the instance to vertices inside the instance now connect to the new vertex. Edges internal to the instance are removed. The discovery process is then applied to the compressed data.

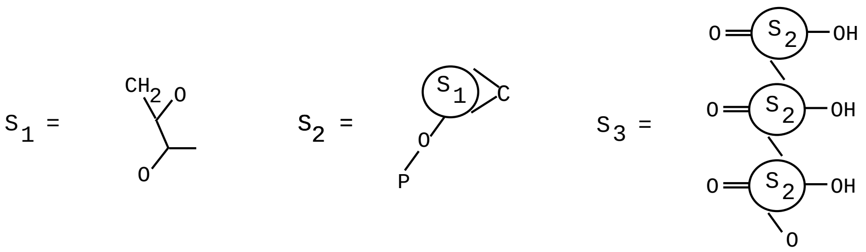

Figure 6: Hierarchical discovery in DNA data.

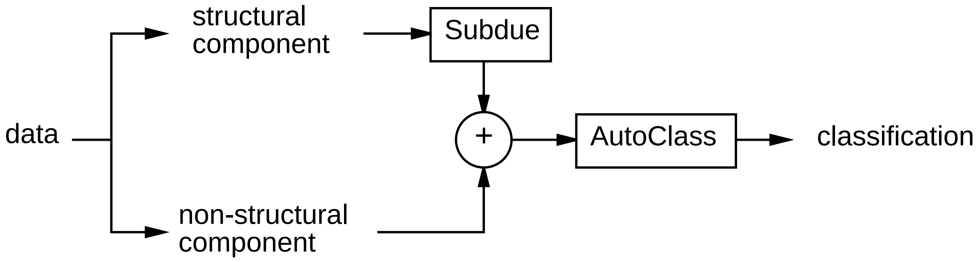

Figure 7: Integration of Subdue and AutoClass.

To demonstrate the ability of SUBDUE to find a hierarchy of substructures, we let the system make multiple passes through a database that represents a portion of a DNA molecule. Figure 2 shows a portion of two chains of a double helix, using three pairs of bases which are held together by hydrogen bonds. Figure 6 shows the substructures found by SUBDUE after each of three passes through the data. Substructure s_3 represents each of the two chains in the DNA molecule. As opposed to the 0.91 factor of compression for the DNA database shown in Table 1, the factor of compression after running three passes of SUBDUE through this database is 0.56.

9 Integration with Non-Structural Discovery

One weakness of the SUBDUE system is the inability to incorporate models of non-structural attribute values. For example, if objects in the domain have a temperature that ranges from 0 to 100, SUBDUE considers the values 50.0001 and 50.0002 as different as 0 and 100. Knowledge about the distribution of attribute values is necessary to appropriately match this type of data in the substructures. One possibility is to infer the parameters of a normal model for each attribute from the data and use this information to affect the match cost of two graphs. Another possibility is to use SUBDUE as a pre-processor of the structural component of the data in order to construct new non-structural attributes for addition to the set of existing, non-structural attributes. The new set of attributes can then be processed by a nonstructural discovery method, in which the discovered concepts will be biased by the inclusion of structural information.

We are pursuing the second possibility by investigating the integration of SUBDUE with the AUTOCLASS system [1]. AUTOCLASS is an unsupervised, non-structural discovery system that identifies a set of classes describing the data. Each attribute is described by a given model (e.g., normal), and each class is described by particular instantiations of the models for each attribute. AUTOCLASS searches for the classification maximizing the conditional probability of the data given the classification.

Figure 7 diagrams the proposed integration of SUBDUE and AUTOCLASS. First, SUBDUE is run on the structural component of the data to produce prevalent substructures in the data. For each discovered substructure, the vertices of the substructure (which correspond to individual objects or examples in the non-structural data component) become new attributes whose values reflect the degree with which that particular object or example participates as the vertex in the substructure. For example, if SUBDUE discovered a substructure consisting of two vertices and one edge, then two new, non-structural attributes would be added to the original attributes of each data object. The value of the first new attribute, corresponding to the first vertex of the substructure, measures the degree to which the object occurs as this vertex in an instance of the substructure. Likewise, the second new attribute measures a similar value corresponding to the second vertex of the substructure. These values can be determined by the match cost between the substructure and the instance containing the object.

The new data, augmented with this non-structural information about the structural regularities in the data, can now be passed to the AUTOCLASS system. The structural information will bias the classifications preferred by AUTOCLASS towards those consistent with the structural regularities in the data. Implementation of the integrated discovery system is underway. We plan to evaluate the integration of SUBDUE and AUTOCLASS by comparing the results of AUTOCLASS alone on data with a weak, non-structural classification and a strong, structural classification. We also plan to compare previous AUTOCLASS results with the classifications obtained with the integrated discovery system using the same data augmented with natural structural information such as temporal and proximate ordering.

10 Conclusions

Discovering knowledge in data containing structural information can be achieved by identifying repetitive substructures in the data. The substructures represent new concepts found in the data and a means of reducing the complexity of the data by abstracting over instances of the substructure. The SUBDUE system provides a method for discovering substructures in data expressed as a labeled, directed graph. We have shown how the minimum description length principle and background knowledge used by SUBDUE can guide substructure discovery in a variety of domains. Once a substructure is discovered, instances of the substructure can be replaced by the concept definition. This affords compression of the data description and provides a basis for discovering hierarchically-defined structures.

The next step is to extend SUBDUE by adding the ability to intelligently handle non-structural data. Current efforts are underway to integrate SUBDUE and AUTOCLASS, and methods for including attribute value models into the inexact graph match are also under investigation. The large amount of structural information that can be added to non-structural data collected on physical phenomena provides a large testbed for comparing an integrated discovery system based on SUBDUE to other non-structural discovery systems.

Acknowledgements

The authors would like to thank Pat Langley for numerous comments regarding the evaluation of our system. We would also like to thank Tom Lai for implementing an improved graph match procedure. This work is supported by NASA grant NAS5-32337.

References

[1] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64, 1988.

[2] D. Conklin, S. Fortier, J. Glasgow, and F. Allen. Discovery of spatial concepts in crystallographic databases. In *Proceedings of the ML92 Workshop on Machine Discovery*, pages 111–116, 1992.

[3] D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255, 1994.

[4] M. Derthick. A minimal encoding approach to feature discovery. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 565–571, 1991.

[5] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[6] K. S. Fu. *Syntactic Pattern Recognition and Applications.* Prentice-Hall, 1982.

[7] L. B. Holder and D. J. Cook. Discovery of inexact concepts from structural data. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):992–994, 1993.

[8] L. B. Holder, D. J. Cook, and H. Bunke. Fuzzy substructure discovery. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 218–223, 1992.

[9] E. Jeltsch and H. J. Kreowski. Grammatical inference based on hyperedge replacement. In *Fourth International Workshop on Graph Grammars and Their Application to Computer Science*, pages 461–474, 1991.

[10] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computational Vision*, 3(1):73–102, 1989.

[11] R. Levinson. A self-organizing retrieval system for graphs. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 203–206, 1984.

[12] R. S. Michalski and R. E. Stepp. Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 331–363. Tioga Publishing Company, 1983.

[13] L. Miclet. *Structural Methods in Pattern Recognition.* Chapman and Hall, 1986.

[14] E. P. D. Pednault. Some experiments in applying inductive inference principles to surface reconstruction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1603–1609, 1989.

[15] A. Pentland. Part segmentation for object recognition. *Neural Computation*, 1:82–91, 1989.

[16] R. Prather. *Discrete Mathemetical Structures for Computer Science.* Houghton Miffin Company, 1976.

[17] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[18] R. B. Rao and S. C. Lu. Learning engineering models with the minimum description length principle. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 717–722, 1992.

[19] J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, 1989.

[20] J. Segen. Graph clustering and model learning by data compression. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 93–101, 1990.

[21] K. Thompson and P. Langley. Concept formation in structured domains. In D. H. Fisher and M. Pazzani, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*, chapter 5. Morgan Kaufmann Publishers, 1991.

[22] D. Waltz. Understanding line drawings of scenes with shadows. In P. H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, 1975.

[23] M. Wertheimer. Laws of organization in perceptual forms. In W. D. Ellis, editor, *A Sourcebook of Gestalt Psychology*, pages 331–363. Harcourt, Brace and Company, 1939.

[24] P. H. Winston. Learning structural descriptions from examples. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 157–210. McGraw-Hill, 1975.

[25] K. Yoshida, H. Motoda, and N. Indurkhya. Unifying learning methods by colored digraphs. In *Proceedings of the Learning and Knowledge Acquisition Workshop at IJCAI-93*, 1993.