# Structural Mining of Molecular Biology Data*

Diane J. Cook, Lawrence B. Holder, Shaobing Su, Ron Maglothin, and Istvan Jonyer
Department of Computer Science Engineering
University of Texas at Arlington
Email: {cook, holder, su, maglothi, jonyer}@cse.uta.edu
http://cygnus.uta.edu/subdue.html

**Abstract**

The increasing amount and complexity of molecular biology data evokes a need to focus on automated methods for mining this data. In addition, molecular biology data is frequently structural in nature, or is composed of parts and relations between the parts. Hence, there exists a need to develop tools to analyze and discover concepts in structural databases.

The goal of this research is to provide a system that mines databases represented as graphs. We demonstrate how the SUBDUE system can be used to perform three key techniques useful for mining molecular biology data: unsupervised pattern discovery, supervised concept learning from examples, and conceptual clustering. Applications of the SUBDUE system to protein databases demonstrate the effectiveness of the graph-based system to discover patterns in this data.

## 1 Introduction

In recent years, there has been an explosive amount of molecular biology information obtained and deposited in various databases. Identifying and interpreting interesting patterns from this massive amount of information has become an essential component in directing further molecular biology research. The sizes of these data sets are too large for effective analysis by humans; they require automated methods of sequence analysis and pattern discovery. Data mining algorithms that can find biologically important patterns in these large databases, and that can do so in polynomial running time, are in great demand.

*Supported by NSF grant IRI-9615272 and THECB grant 003656-045.

In response to this problem, a number of researchers have developed techniques for discovering concepts in databases. Although much of the molecular biology data collected today has an explicit or implicit structural component, few discovery systems handle this type of data [10].

One method for discovering knowledge in structural data is the identification of common substructures (concepts represented as graphs) within the data. Once identified, these substructures 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 the SUBDUE system that discovers interesting substructures in structural data. SUBDUE discovers substructures that compress the original database and represent interesting structural concepts in the data. By compressing previously-discovered substructures in the data, multiple passes of SUBDUE produce a hierarchical description of the structural regularities in the data. We demonstrate how the discovery capabilities of SUBDUE can be used to discover patterns in protein and DNA databases.

## 2 Related Work

A variety of approaches to unsupervised discovery using structural data have been proposed (e.g., [6, 20]). Many of these approaches use a knowledge base of concepts to classify the structural data. These systems perform concept learning over examples and categorization of observed data. While the above methods represent examples as distinct objects and process individual objects one at a time, our method stores the entire database (with embedded objects) as one graph and processes the graph as a whole.

There are several applications of pattern search in proteins on the secondary structure level. Mitchell et al. [14] use an algorithm that identifies subgraph isomorphism in protein structure by searching for an exact match of a specific pattern in a database. The approach of Grindley et al. [12] finds maximally common substructures between two proteins on the secondary structure

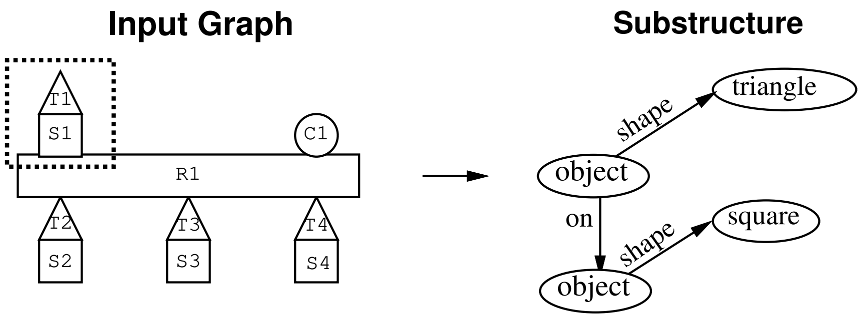


Figure 1: Example substructure in graph form.

level, and can therefore highlight areas of structural overlap between proteins. In Koch et al. [13], a graph is used to represent protein helices and strands to search for proteins that are distantly related at a structural level. Most of these studies focus on identifying predefined patterns in a group of proteins.

Molecular biologists also have several computational tools available for the analysis of DNA sequences. Pattern matching approaches scan sequences for instances of a given pattern [1, 15, 18]. Gene recognition approaches identify potential genes in the sequence data [3, 17]. However, the goal of pattern discovery is quite different from the previous goals of pattern matching. Our goal is to find patterns that occur often in the input sequences, with no bias toward already-known patterns. As such, pattern discovery methods offer the potential to find unexpected patterns that may be biologically important.

## 3 Unsupervised Concept Discovery Using Subdue

SUBDUE discovers substructures that compress the original data and represent structural concepts in the data. The substructure discovery system represents structural data as a labeled graph. Objects in the data map to vertices or small subgraphs in the graph, and relationships between objects map to directed or undirected edges in the graph. 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 that match, graph theoretically, to the graphical representation of the substructure. This graphical representation serves as input to the substructure discovery system. Figure 1 shows a geometric example of a database. The graph representation of the discovered substructure is also shown, and one of the four instances of the substructure is highlighted in the picture.

The substructure discovery algorithm used by SUBDUE is a beam search. SUBDUE's discovery algorithm is shown in Fig. 2. The first step of the algorithm is to initialize ParentList (containing substructures to be expanded), ChildList (containing substructures that have been expanded), and BestList (containing the highest-valued substructures SUBDUE has found so far) to be empty, and to set ProcessedSubs (the number of substructures that have been expanded so far) to 0. Each of the lists is a linked list of substructures, sorted in nonincreasing order by substructure value. For each unique vertex label, a substructure is assembled whose definition is a vertex with that label, and whose instances are all of the vertices in $G$ with that label. Each of these substructures is inserted into ParentList.

The inner *while* loop is the core of the algorithm. Each substructure in turn is removed from the head of ParentList, and each of its instances is extended in all possible ways. This is done by adding a new edge and vertex in $G$ to the instance, or just adding a new edge between two vertices, if both of the vertices are already part of the instance. The first instance of each unique expansion becomes a definition for a new child substructure, and all of the child instances that were expanded in the same way (i.e., by adding the same new edge or new edge with new vertex to the same old vertex) become instances of that child substructure. In addition, child instances that were generated by different expansions, and that match the child substructure definition within the matchcost threshold, also become instances of the child substructure.

Each child is then evaluated using the Minimum Description Length heuristic, and inserted in ChildList in order by the heuristic value. The beam width of the search is enforced by controlling the length of ChildList: after inserting a new child into ChildList, if the length of ChildList exceeds the BeamWidth, the substructure at the end of the list is destroyed. The parent substructure is

```
SUBDUE( Graph, BeamWidth, MaxBest, MaxSubSize, Limit )
    ParentList = {}
    ChildList = {}
    BestList = {}
    ProcessedSubs = 0
    Create a substructure from each unique vertex label and its single-vertex
        instances; insert the resulting substructures in ParentList
    while ProcessedSubs <= Limit and ParentList is not empty do
        while ParentList is not empty do
            Parent = RemoveHead( ParentList)
            Extend each instance of Parent in all possible ways
            Group the extended instances into Child substructures
            foreach Child do
                if SizeOf( Child ) <= MaxSubSize then
                    Evaluate the Child
                    Insert Child in ChildList in order by value
                    if Length( ChildList ) > BeamWidth then
                        Destroy the substructure at the end of ChildList
            ProcessedSubs = ProcessedSubs + 1
            Insert Parent in BestList in order by value
            if Length( BestList ) > MaxBest then
                Destroy the substructure at the end of BestList
        Switch ParentList and ChildList
    return BestList
```

Figure 2: Subdue's discovery algorithm.

inserted in BestList; the same pruning mechanism is used to limit the length of BestList to be no greater than MaxBest. When ParentList has been emptied, ParentList and ChildList are switched, so that ParentList now holds the next generation of substructures to be expanded. SUBDUE's running time is constrained to be polynomial by the BeamWidth and Limit (a user-defined limit on the number of substructures to process) parameters, as well as by computational constraints on the inexact graph match algorithm.

One of Subdue's user-specified parameters controls whether the substructure discovery search space is pruned by discarding a child substructure whose heuristic value is not greater than its parent's heuristic value. Early in the discovery process, the number of instances of a given substructure is very large, and this value dominates the value of any heuristic which uses the number of instances as a parameter. As the substructure grows, its number of instances decreases quickly, since the substructure is becoming more specific. This means that the heuristic value usually also decreases early in the discovery process. If the value decreases long enough, all the child substructures will be discarded, the list of substructures waiting for expansion will empty, and the discovery will halt. Disabling pruning during discovery keeps the child list full, even for substructure values that do not increase monotonically as the substructure definition grows in size. Since the list of substructures waiting for expansion never empties, another method for halting the program is needed. If the user specifies a maximum substructure size, child substructures that are larger than MaxSubSize will not be inserted in ChildList. This will cause ParentList and ChildList to eventually empty, and the discovery algorithm will halt.

Once a substructure is discovered, the substructure is used to simplify the data by replacing instances of the substructure with a pointer to the newly discovered substructure. The discovered substructures allow abstraction over detailed structures in the original data. Iteration of the substructure discovery and replacement process constructs a hierarchical description of the structural data in terms of the discovered substructures. This hierarchy provides varying levels of interpretation that can be accessed based on the specific goals of the data analysis.

### 3.1 The MDL Heuristic

Subdue's heuristic for evaluating substructures is based on the Minimum Description Length (MDL) principle, which states that the best theory to describe a set of data is that theory which minimizes the description length of the entire data set [16]. Description length calculation is based on the model of a local computer, the encoder, sending a description of a concept to a remote computer, the decoder. The local computer must encode the concept as a string of bits that can be sent to the remote computer, which decodes the bit string to restore the original concept. The concept's description length is the number of bits in the bit string. The MDL principle has been used for decision tree induction, pattern discovery in biosequences, image processing, concept learning from relational data, and learning models of non-homogeneous engineering domains.

SUBDUE's implementation of the MDL principle is in the context of graph compression using a substructure. Here, the best substructure in a graph is one that minimizes $DL(S) + DL(G|S)$, where $S$ is the discovered substructure, $G$ is the input graph, $DL(S)$ is the number of bits required to encode the discovered substructure, and $DL(G|S)$ is the number of bits required to encode the input graph $G$ after it has been compressed using substructure $S$. Cook and Holder [7] describe the exact computation of graph description length used in SUBDUE.

### 3.2 Inexact graph match

Because instances of a substructure can appear in different forms throughout the database, an inexact graph match is used to identify substructure instances [8]. In this inexact match approach, 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.

Given graphs $g_1$ with $n$ vertices and $g_2$ with $m$ vertices, $m \geq n$, the complexity of the full inexact graph match is $O(n^{m+1})$. Because this routine is used heavily throughout the discovery process, the complexity of the algorithm can significantly degrade the performance of the system.

To improve the performance of the inexact graph match algorithm, we search through the spac of possible partial mappings using a uniform cost search. The cost from the root of the tree to a given node is calculated as the cost of all of the distortions corresponding to the partial mapping for that node. Vertices from the matched graphs are considered in order from the most heavily connected vertex to the least connected. Because uniform cost search guarantees an optimal solution, the search ends as soon as the first complete mapping is found.

In addition, the user can limit the number of search nodes considered by the branch-and-bound procedure (defined as a function of the input graph sizes). Once the number of nodes expanded in the search tree reaches the defined limit, the search resorts to hill climbing using the cost of the mapping so far as the measure for choosing the best node at a given level. A complete description of the polynomial inexact graph match used by SUBDUE is provided by Cook and Holder [7].

Employing computational constraints such as a bound on the number of substructures considered ($L$) and the number of partial mappings considered during an inexact graph match ($g$), SUBDUE is constrained to run in polynomial time. The worst-case run time of the system is the product of the number of generated substructures, the number of instances of each substructure, and the number of partial mappings considered during graph match. This expression is equal to $(\sum_{i=1}^{L} i * ((v-1)-(i-1))) * (v(L-1)) * g$, where $v$ represents the number of vertices in the input graph. The derivation of this expression is provided in the literature [8].

## 4 Application of Unsupervised Subdue to Molecular Biology

The SUBDUE discovery system has been applied to databases in a number of domains. With each new application area, capabilities are identified that need to be added to our system. We have successfully applied SUBDUE with and without domain knowledge to databases in domains including image analysis, CAD circuit analysis, Chinese character databases, program source code, and chemical reaction chains [8, 9].

More recently, we have applied SUBDUE to several large databases that contain data whose interpretation will be beneficial to scientists, and that reveal capabilities that need to be added to the discovery system. First, we have applied SUBDUE to the July 1997 release of the Brookhaven Protein Data Bank (PDB). The goal was to identify structural patterns in primary, secondary, and tertiary structure of three categories of proteins: hemoglobin, myoglobin, and ribonuclease A. These patterns would act as signatures for the protein category, distinguishing proteins in the category from other types of proteins and providing a mechanism for classifying unknown proteins.

We convert primary structure information from the SEQRES records of each PDB file by representing each amino acid in the sequence as a graph vertex. The vertex numbers increase in the sequence order from N-terminus to C-terminus, and the vertex label is the name of the amino acid. An edge labeled "bond" is added between adjacent amino acids in a sequence.

Secondary structure is extracted by listing occurrences of helices and strands along the primary sequence. Each helix is represented by a graph vertex labeled "h" followed by the helix type and length. Each strand is represented by a graph vertex labeled "s" followed by the orientation of the strand and the length of the strand. Edges between two consecutive vertices is labeled "sh" if they belong to the same PDB file. The 3D features of the protein are represented using the X, Y, and Z coordinates of each atom in the protein. Each amino acid $\alpha$-carbon is represented as a graph vertex. If the distance between two $\alpha$-carbons is greater than 6 $\AA$, the information is discarded. Otherwise, edges between two $\alpha$-carbons are created and labeled as "vs" (very short, distance $\leq 4$ $\AA$), or "s" (short).

SUBDUE indeed found such a pattern for each protein category. Using primary structure information, patterns were identified that were unique to each class of protein but occurred in 63 of the 65 hemoglobin cases, 67 of the 103 myoglobin cases, and 59 of the 68 ribonuclease A cases.

Figure 3 summarizes one of the findings for hemoglobin secondary structure, presenting an overall view of the protein, the portion of the protein where the SUBDUE-discovered pattern exists, and the schematic views of the best pattern. The patterns discovered for each sample category covered a majority of the proteins in that category (33 of the 50 analyzed hemoglobin proteins, 67 of the 89 myoglobin proteins, and 35 of the 52 ribonuclease A proteins contained the discovered patterns). Detailed analysis of those that do not have the pattern indicates that there are many possible reasons. The structure of a protein is affected by many factors. The accuracy of the structure is affected by the quality of the protein sample, experimental conditions, and human error. Discrepancies may also be due to physiological and biochemical reasons. Structure of the same protein molecule may differ from one species to another. The protein may also be defective. For example, sickle-cell anemia is the classic example of a genetic hemoglobin disease. The defective protein does not have the right structure to perform its normal function.

The secondary structural patterns for the hemoglobin, myoglobin, and ribonuclease A proteins were mapped back into the PDB files. When mapped back, one discovered hemoglobin pattern was found to belong to the $\beta$ chains and the other belonged to the $\alpha$ chains of a hemoglobin molecule (a hemoglobin molecular contains two $\alpha$ and two $\beta$ chains). The discovered myoglobin pattern appears in a majority of the myoglobin proteins in the data set. Finally, upon mapping back discovered ribonuclease A patterns, we observed that several ribonuclease S proteins have the same patterns as those in ribonuclease A proteins. This is consistent with the fact that ribonuclease S is a complex consisting of two fragments (S-peptide and S-protein) of the ribonuclease A proteins. The pattern in the ribonuclease S comes from the S-protein fragment.

Dr. Steve Sprang, a molecular biologist at the University of Texas Southwestern Medical Center, evaluated the patterns discovered by the SUBDUE system. This scientist was asked to review the original database and the discovered substructures, and determine if the discovered concepts were indicative of the data and interesting discoveries. Dr. Sprang indicated that SUBDUE did find an interesting pattern in the data that was previously unknown and suggests new information about the micro-evolution of such proteins in mammals [19].

The secondary structure patterns discovered are also distinct to each protein category. The global data set is searched to identify the possible existence of the discovered pattern from each category. Results indicate that there is no exact match of the best patterns of one category in another category of proteins.

Application of the unsupervised and supervised learning algorithms in SUBDUE is continuing

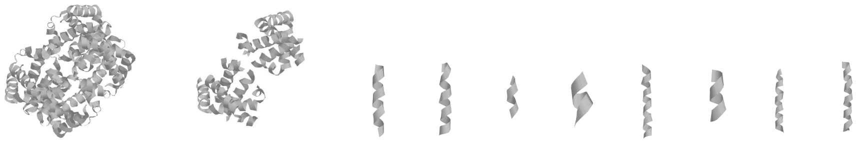

Figure 3: Hemoglobin structure, discovered pattern, and schematic view.

in the domains of biochemistry, geology, program source code, and aviation data. SUBDUE thus brings unique capabilities to the unsupervised analysis of data that is structural in nature.

# 5 Supervised Concept Learning Using Subdue

We have extended SUBDUE to act not only as an unsupervised discovery system, but also to perform supervised graph-based relational concept learning. Few general-purpose learning methods use a formal graph representation for knowledge, perhaps because of the arbitrary expressiveness of a graph and the inherent NP-hardness of typical learning routines. Structural learning methods customized for biological and chemical domains have been successful due to the natural graphical representation of this data, but no domain-independent concept learning systems use a graph representation. Therefore, we have added a concept learning capability to the SUBDUE graph-based discovery system.

The key to adding concept-learning capabilities to SUBDUE is the inclusion of a "negative" graph into the process. Substructures that occur often in the positive graph, but not often in the negative graph, are likely to represent the target concept. Therefore, the SUBDUE concept learner (which we will refer to as SUBDUECL) accepts both a positive and a negative graph, and evaluates substructures based on their compression of the positive graph and lack of compression of the negative graph. SUBDUECL searches for the substructure $S$ minimizing the cost defined as

$$value(G_p, G_n, S) = DL(G_p, S) + DL(S) + DL(G_n) - DL(G_n, S)$$

where $DL(G, S)$ is the description length, according to the MDL encoding, of a graph $G$ after being compressed using substructure $S$, and $DL(G)$ is the description length of a graph $G$. This cost represents the information needed to represent the positive graph $G_p$ using the substructure $S$ plus the information needed to represent the portion of the negative graph $G_n$ that was compressed using substructure $S$. Therefore, SUBDUECL prefers substructures that compress the positive graph, but not the negative graph.

A challenge in adding concept learning capabilities to SUBDUE is the discovery system's bias toward finding only one good substructure in the entire input graph. Inductive logic programming (ILP) systems may offer an advantage in structural learning over SUBDUE because they typically find theories composed of many rules [4]; whereas, SUBDUE finds essentially one rule. The iterative, hierarchical capabilities of SUBDUE somewhat address this problem, but the substructures found in later iterations are typically defined in terms of previously-discovered substructures, and are therefore only specializations of the earlier, more general rule. To avoid this tendency, SUBDUECL discards any substructure that contains substructures discovered during previous iterations. SUBDUECL iterates until no substructure can be found that compresses the positive graph more than the negative graph.

## 6 Application of Supervised Subdue to Molecular Biology

For over a decade, the molecular biology research community has been involved in a worldwide effort to obtain the entire genomic DNA sequences of several organisms. The most well-known of these projects is the Human Genome Project. As part of this project, genomes for a number of organisms have been completely sequenced. We demonstrate how SUBDUECL can be used to find biologically important patterns in the DNA sequence of baker's yeast, *Saccharomyces cerevisiae.*

DNA sequence patterns can be classified by the regular languages they describe. Although it is possible to discover simple patterns in linear time, allowing don't-care and arbitrarily repeating characters increases the complexity of the algorithm. However, the general pattern classes are more biologically common and are of greater interest to researchers. Our goal is to determine if SUBDUE

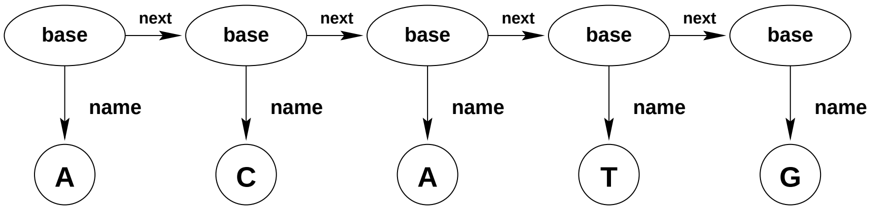


Figure 4: Backbone graph representation.

can automatically find such patterns in DNA data.

The data for our experiments come from a recent study by Brazma et al. [2]. One part of their study used windows of sequences that occur immediately upstream of yeast gene start sites. These yeast genes were clustered into groups based on similar gene expression profiles, then discretized into seven time intervals and either three or five intervals based on gene expression rate. Each cluster contains genes that are likely to be involved in related cellular processes, and perhaps even regulated by similar or identical mechanisms.

To allow don't care positions in the data representation and thus the discovered patterns, we use a "backbone" graph representation shown in Fig. 4. This representation separates the base names from the vertices representing the bases. The intention is to allow SUBDUE to represent don't-care bases by leaving out the "name" edges and neighboring vertices. Interestingly, the backbone representation mimics the actual chemical structure of a DNA molecule, in which the DNA bases are connected by deoxyribose sugars to a linear phosphate backbone.

For the results shown in Table 1, SUBDUE was run using an exact match and five iterations. The best pattern for each iteration is shown in the table. For this experiment, cluster cr4.111101.77, with 77 examples, was used as the positive training set. The negative examples consist of 100 sequences randomly selected from the set of upstream sequences not belonging to the targeted cluster. The last column in each table contains a list of TRANSFAC transcription factor binding sites containing substrings that exactly match the discovered pattern. TRANSFAC matches are the primary criterion used to evaluate the discovered patterns. Patterns which match TRANSFAC

| Pattern | $N_+$ | $N_-$ | TRANSFAC exact matches |
|---|---|---|---|
| ATCCAT | 16 | 12 | MOUSE$AP2_02, HS$GRH_05, RAT$GRH_16, HCMV$IE1_22, RICE$GL51_02, HS$BCR_04, HS$BCR_04, RAT$SPI_01, HS$GBP_02, HS$EG_05, MOUSE$GSR_01, MOUSE$GSR_03 |
| GGGA.G.A | 19 | 16 | MOUSE$HOX23_01, HS$CREBP1_01, RABBIT$UG_18, HS$ALDA_02, HS$APOB_14, MOUSE$PCP2_02 |
| TCCCT | 65 | 35 | Y$G3PDH_01 |
| AAGGG | 95 | 37 | CAMV$35SR_01, RAT$ALBU_18, AD$E2AE1_11, AD$E2AE1_15, RAT$EAI_11, RAT$BF_03, DROME$FTZ_03, MOUSE$BMG_06, RAT$GLU_01, HS$GHA_01, HS$H3_01, HS$H3_02, DROME$HSP27_04, HS$HSP70_07, HS$HSP70_08, HS$IFNB_03, HS$IFNB_04, HS$ISG15_01, HS$ISG15_02, HS$ISG15_03, HS$ISG15_04, HS$ISG15_05, MOUSE$M2EB_01, MOUSE$CMYC_01, RAT$AMHC_01, HS$TF_02, MOUSE$M2AAK_01, MOUSE$IGKL_22, CAMV$35SR_04, DROOR$ADH_12, PCF$CONS, HS$CREBP1_04, HS$CREBP1_16, HS$CREBP1_17, HS$TPO_03, DROME$EVE_26, DROME$EVE_27, MOUSE$ADIP_02, KR$CONS_02, DROME$HB_03, DROME$KNI_01, DROME$KNI_02, HS$GHA_08, MOUSE$IGH_47, HS$GAPDH_01, MOUSE$S16H_02, HPV$HPV16_12, CHICK$APOVLDL_07, HS$ADH3_01, MOUSE$TTPA_06, HS$DPOLA_01, CHICK$GATA1_05, HS$EG_02, HS$TF_10, HS$PGK_05, RABBIT$UG_19, EBV$BZLF1_06, MOUSE$BMG_10, MOUSE$CD4_02, HS$GFER_01, RAT$OMP_06, RAT$OCNC_01, HS$APOB_12, HS$APOB_14, MOUSE$MB1_02, MOUSE$M2AAD_01, AD$E2AE1_24, RAT$BMHC_06, MOUSE$LB1_01, RAT$GRH_23, HS$PR264_04, DROME$SNA_01, HS$AG_15, MOUSE$RARB_01, HS$PDGF2_02 |
| CCCT | 128 | 76 | Y$BCY_01, Y$GAL1_04, Y$X40_01, Y$CYC1_12, Y$GAL1_14, Y$G3PDH_01, Y$POX1_01, Y$DDR2_01, Y$DDR2_02, Y$TPI_02 |

Table 1: Best patterns found by SUBDUE in cluster cr4.111101.77.

site substrings are more likely to be part of binding sites themselves, and thus are more likely to be involved in gene regulation.

All of the patterns shown in Table 1 have exact TRANSFAC matches and are non-trivial (the smallest has four bases). Pattern AAGGG is especially interesting because it has the greatest number of positive instances, no exact matches to yeast TRANSFAC entries, but 75 exact matches to transcription factor binding sites of other organisms. These features indicate a strong possibility that this may be part of an actual, as yet undiscovered, yeast transcription factor binding site.

The results of this study demonstrate that SubdueCL can be used to discover patterns in DNA sequences that may be involved in the regulation of gene expression. Using a graph representation, patterns involving don't care and arbitrarily repeated characters can be learned in addition to simple patterns, all within a polynomial run time.

# 7 Structural Hierarchical Clustering Using Subdue

Clustering techniques provide a useful means of gaining better understanding of data, in many cases through revealing hierarchical topologies. Clustering has been applied in diverse fields such as analytical chemistry, geology, biology, zoology and archeology, and is a key component in model fitting, hypothesis generation and testing, data exploration and data reduction. A simple example of hierarchical clustering is the classification of vehicles into groups such as cars, trucks, motorcycles, tricycles, and so on, which are then further subdivided into smaller and smaller groups based on individual traits.

Current clustering techniques have some intrinsic disadvantages. Statistical and syntactic approaches have trouble expressing structural information, and neural approaches are greatly limited in representing semantic information. Despite these limitations, a number of clustering algorithms have demonstrated success including Cobweb [11], Labyrinth [20], and AutoClass [5]. These approaches usually have the disadvantage of being applicable only to metric data, which excludes discrete-valued and structured databases.

Our graph-based hierarchical conceptual clustering algorithm uses SUBDUE to construct a hierarchical lattice of clusters. The substructure discovered after a single iteration comprises a cluster. This cluster is inserted into the classification lattice and used to compress the input graph. The compressed graph is passed again to Subdue to find another substructure. This iteration allows Subdue to find new substructures defined in terms of previously discovered substructures.

Previous work suggested the user of classification trees; however, in structured domains a strict tree is inadequate. In these domains a lattice-like structure may emerge instead of a tree. When substructures are added to the lattice, their parents may include other, non-root nodes in the lattice. If a substructure is composed of two of the same previously-discovered substructures, then there will be two links from the parent to the child in the lattice.

A small substructure is more general than a large one, and should represent a parent node in the classification lattice for any more specific clusters. To ensure that a small substructure is found, during the substructure search we modify SUBDUE to finish an iteration when it finds a local minima rather than the global minima (which may be a much larger substructure). In many cases the local minima is also the global minima, and in other cases the larger substructure will usually be found on a later iteration. In this process, SUBDUE effectively searches the space of all classification lattices.

## 8 Application of Structural Clustering to Molecular Biology

To illustrate the clusters that Subdue generates for structured data, we apply the algorithm to a portion of a DNA data shown in Fig. 5. In the input graph, vertices represent atoms and small molecules, and edges represent bonds. A portion of the classification lattice generated by Subdue is shown in Fig. 6. As the figure shows, the first level of the lattice represents small commonly-occurring substructures (covering 61% of the data). Subsequently identified clusters are based on these smaller clusters that are combined with each other or with other atoms or molecules to form a new cluster. The classification lattice can be used to understand portions of the data that behave

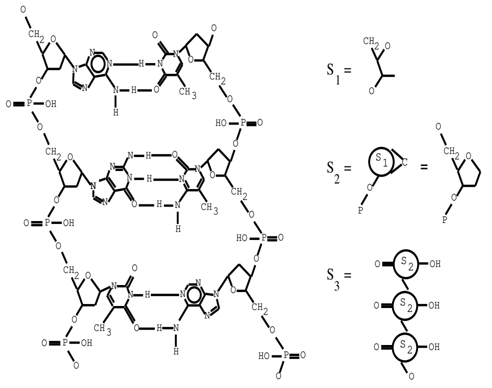


Figure 5: Portion of a DNA molecule.

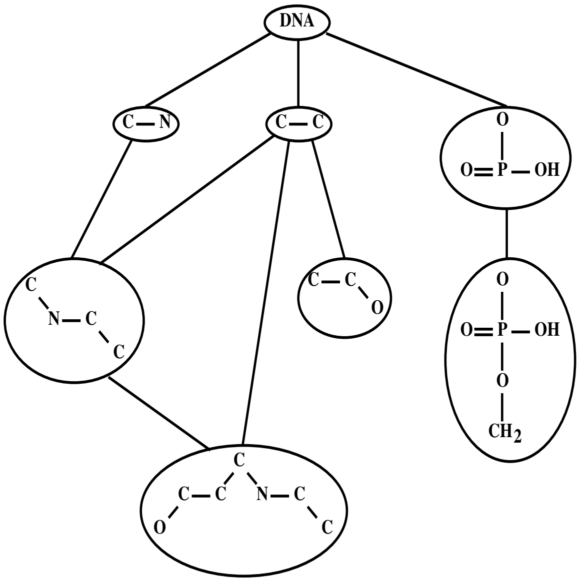


Figure 6: Hierarchical clustering of a DNA molecule.

in similar manners, to better understand the database as a whole, and to predict unknown features of data elements given features of elements in the same cluster.

# 9 Conclusions

The structural component of molecular biology databases requires data mining algorithms capable of handling structural information. The SUBDUE system is specifically designed to discover concepts in structural databases. In this paper, we have described how SUBDUE can be used to perform unsupervised pattern discovery, supervised concept learning and structural clustering to efficiently learn concepts in molecular biology databases.

This comparison has identified a number of avenues for enhancements to both types of learners. The graph-based learner SubdueCL would benefit from the ability to represent recursion, which plays a central part in many logic-based concepts. More research is needed to identify enhancements to the representation that can describe recursive structures, such as using graph grammars.

The results obtained in this study indicate that SUBDUE is suitable for knowledge discovery in molecular structural databases. Planned fugure work includes investigating additional graph representations of molecular biology data, searching for patterns in the 3D structure of proteins, and analyzing additional DNA sequences.

# References


[1] S. F. Altschul, W. Gish, W. Miller, E. W. Meyers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[2] A. Brazma, I. Jonassen, J. Vilo, and E. Ukkonen. Predicting gene regulatory elements in silico on a genomic scale. *Genome Research*, 8(11):1202–1215, 1998.

[3] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268:78–94, 1997.

[4] R. M. Cameron-Jones and J. R. Quinlan. Efficient top-down induction of logic programs. *SIGART Bulletin*, 5(1):33–42, 1994.

[5] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 6, pages 153–180. MIT Press, 1996.

[6] D. Conklin. Machine discovery of protein motifs. *Machine Learning*, 21:125–150, 1995.

[7] 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.

[8] D. J. Cook, L. B. Holder, and S. Djoko. Scalable discovery of informative structural concepts using domain knowledge. *IEEE Expert*, 11(5), 1996.

[9] S. Djoko, D. J. Cook, and L. B. Holder. An empirical study of domain knowledge and its benefits to substructure discovery. *IEEE Transactions on Knowledge and Data Engineering*, 9(4):575–586, 1997.

[10] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 1, pages 1–34. MIT Press, 1996.

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

[12] H. M. Grindley, P. J. Artymiuk, D. W. Rice, and P. Willett. Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm. *Journal of Molecular Biology*, 229:707–721, 1993.

[13] I. Koch, T. Lengauer, and E. Wanke. An algorithm for finding maximal common subtopologies in a set of protein structures. *Journal of Computational Biology*, 3(2):289–306, 1996.

[14] E. M. Mitchell, P. J. Artymiuk, D. W. Rice, and P. Willett. Use of techniques derived from graph theory to compare secondary structure motifs in proteins. *Journal of Molecular Biology*, 212:151–166, 1990.

[15] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:444–453, 1970.

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

[17] S. Salzburg, A. Delcher, S. Kasif, and O. White. Microbial gene identification using interpolated Markov models. *Nucleic Acids Research*, 26(2):544–548, 1998.

[18] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[19] S. Sprang. Personal Communication, 1998.

[20] 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.