3

Graph-based Mining of Complex Data

Diane J. Cook, Lawrence B. Holder, Jeff Coble and Joseph Potts

Summary. We describe an approach to learning patterns in relational data represented as a graph. The approach, implemented in the Subdue system, searches for patterns that maximally compress the input graph. Subdue can be used for supervised learning, as well as unsupervised pattern discovery and clustering.

Mining graph-based data raises challenges not found in linear attribute–value data. However, additional requirements can further complicate the problem. In particular, we describe how Subdue can incrementally process structured data that arrives as streaming data. We also employ these techniques to learn structural concepts from examples embedded in a single large connected graph.

3.1 Introduction

Much of current data-mining research focuses on algorithms to discover sets of attributes that can discriminate data entities into classes, such as shopping or banking trends for a particular demographic group. In contrast, we are developing data-mining techniques to discover patterns consisting of complex relationships between entities. The field of relational data mining, of which graph-based relational learning is a part, is a new area investigating approaches to mining relational information by finding associations involving multiple tables in a relational database.

Two main approaches have been developed for mining relational information: logic-based approaches and graph-based approaches. Logic-based approaches fall under the area of inductive logic programming (ILP) [16]. ILP embodies a number of techniques for inducing a logical theory to describe the data, and many techniques have been adapted to relational data mining [6]. Graph-based approaches differ from logic-based approaches to relational mining in several ways, the most obvious of which is the underlying representation. Furthermore, logic-based approaches rely on the prior identification of the predicate or predicates to be mined, while graph-based approaches are more data-driven, identifying any portion of the graph that has high support. However, logic-based approaches allow the expression of more complicated patterns involving, e.g., recursion, variables, and constraints among variables. These representational limitations of graphs can be overcome, but at a computational cost.

Our research is particularly applicable to domains in which the data is event-driven, such as counter-terrorism intelligence analysis, and domains where distinguishing characteristics can be object attributes or relational attributes. This ability has also become a crucial challenge in many security-related domains. For example, the US House and Senate Intelligence Committees' report on their inquiry into the activities of the intelligence community before and after the September 11, 2001 terrorist attacks revealed the necessity for "connecting the dots" [18], that is, focusing on the relationships between entities in the data, rather than merely on an entity's attributes. A natural representation for this information is a graph, and the ability to discover previously-unknown patterns in such information could lead to significant improvement in our ability to identify potential threats. Similarly, identifying characteristic patterns in spatial or temporal data can be a critical component in acquiring a foundational understanding of important research in many of the basic sciences.

Problems of such complexity often present additional challenges, such as the need to assimilate incremental data updates and the need to learn models from data embedded in a single input graph. In this article we review techniques for graph-based data mining and focus on a method for graph-based relational learning implemented in the Subdue system. We describe methods of enhancing the algorithm to handle challenges associated with complex data, such as incremental discovery of streaming structural data and learning models from embedded instances in supervised graphs.

3.2 Related Work

Graph-based data mining (GDM) is the task of finding novel, useful, and understandable graph-theoretic patterns in a graph representation of data. Several approaches to GDM exist, based on the task of identifying frequently occurring subgraphs in graph transactions, i.e., those subgraphs meeting a minimum level of support. Kuramochi and Karypis [15] developed the FSG system for finding all frequent subgraphs in large graph databases. FSG starts by finding all frequent single and double edge subgraphs. Then, in each iteration, it generates candidate subgraphs by expanding the subgraphs found in the previous iteration by one edge. In each iteration the algorithm checks how many times the candidate subgraph occurs within an entire graph. The candidates whose frequency is below a user-defined level are pruned. The algorithm returns all subgraphs occurring more frequently than the given level.

Yan and Han [19] introduced gSpan, which combines depth-first search and lexicographic ordering to find frequent subgraphs. Their algorithm starts from all frequent one-edge graphs. The labels on these edges, together with labels on incident vertices, define a code for every such graph. Expansion of these one-edge graphs maps them to longer codes. The codes are stored in a tree structure such that if $\alpha = (a_0, a_1, ..., a_m)$ and $\beta = (a_0, a_1, ..., a_m, b)$, the β code is a child of the α code. Since every graph can map to many codes, the codes in the tree structure are not unique. If there are two codes in the code tree that map to the same graph and one is smaller than the other, the branch with the smaller code is pruned during the depth-first search traversal of the code tree. Only the minimum code uniquely defines the graph. Code ordering and pruning reduces the cost of matching frequent subgraphs in gSpan.

Inokuchi *et al.* [12] developed the *Apriori*-based Graph Mining (AGM) system, which uses an approach similar to Agrawal and Srikant's [2] *Apriori* algorithm for discovering frequent itemsets. AGM searches the space of frequent subgraphs in a bottom-up fashion, beginning with a single vertex, and then continually expanding by a single vertex and one or more edges. AGM also employs a canonical coding of graphs in order to support fast subgraph matching. AGM returns association rules satisfying user-specified levels of support and confidence.

We distinguish graph-based relational learning (GBRL) from graph-based data mining in that GBRL focuses on identifying novel, but not necessarily the most frequent, patterns in a graph representation of data [10]. Only a few GBRL approaches have been developed to date. Subdue [4] and GBI [20] take a greedy approach to finding subgraphs, maximizing an information theoretic measure. Subdue searches the space of subgraphs by extending candidate subgraphs by one edge. Each candidate is evaluated using a minimum description length metric [17], which measures how well the subgraph compresses the input graph if each instance of the subgraph were replaced by a single vertex. GBI continually compresses the input graph by identifying frequent triples of vertices, some of which may represent previously-compressed portions of the input graph. Candidate triples are evaluated using a measure similar to information gain. Kernel-based methods have also been used for supervised GBRL [14].

3.3 Graph-based Relational Learning in Subdue

The Subdue graph-based relational learning system[1] [4, 5] encompasses several approaches to graph-based learning, including discovery, clustering and supervised learning, which will be described in this section. Subdue uses a labeled graph $G = (V, E, L)$ as both input and output, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices, $E = \{(v_i, v_j) | v_i, v_j \in V\}$ is a set of edges, and L is a set of labels that can appear on vertices and edges. The graph G can contain directed edges, undirected edges, self-edges (i.e., $(v_i, v_i) \in E$), and multi-edges (i.e.,

[1]Subdue source code, sample data sets and publications are available at ailab.uta.edu/subdue.

more than one edge between vertices v_i and v_j). The input graph need not be connected, but the learned patterns must be connected subgraphs (called substructures) of the input graph. The input to Subdue can consist of one large graph or several individual graph transactions and, in the case of supervised learning, the individual graphs are classified as positive or negative examples.

3.3.1 Substructure Discovery

Subdue searches for a substructure that best compresses the input graph. Subdue uses a variant of beam search for its main search algorithm. A substructure in Subdue consists of a subgraph definition and all its occurrences throughout the graph. The initial state of the search is the set of substructures consisting of all uniquely labeled vertices. The only operator of the search is the *ExtendSubstructure* operator. As its name suggests, it extends a substructure in all possible ways by a single edge and a vertex, or by only a single edge if both vertices are already in the subgraph.

The search progresses by applying the *ExtendSubstructure* operator to each substructure in the current state. The resulting state, however, does not contain all the substructures generated by the *ExtendSubstructure* operator. The substructures are kept on a queue and are ordered based on their description length (sometimes referred to as value) as calculated using the MDL principle described below.

The search terminates upon reaching a user-specified limit on the number of substructures extended, or upon exhaustion of the search space. Once the search terminates and Subdue returns the list of best substructures found, the graph can be compressed using the best substructure. The compression procedure replaces all instances of the substructure in the input graph by single vertices, which represent the substructure definition. Incoming and outgoing edges to and from the replaced instances will point to, or originate in the new vertex that represents the instance. The Subdue algorithm can be invoked again on this compressed graph. This procedure can be repeated a user-specified number of times, and is referred to as an iteration.

Subdue's search is guided by the minimum description length (MDL) [17] principle, which seeks to minimize the description length of the entire data set. The evaluation heuristic based on the MDL principle assumes that the best substructure is the one that minimizes the description length of the input graph when compressed by the substructure [4]. The description length of the substructure S given the input graph G is calculated as $DL(S) + DL(G|S)$, where $DL(S)$ is the description length of the substructure, and $DL(G|S)$ is the description length of the input graph compressed by the substructure. Description length $DL()$ is calculated as the number of bits in a minimal encoding of the graph. Subdue seeks a substructure S that maximizes compression as calculated in Equation (3.1).

$$Compression = \frac{DL(S) + DL(G|S)}{DL(G)} \tag{3.1}$$

As an example, Figure 3.1a shows a collection of geometric objects described by their shapes and their "ontop" relationship to one another. Figure 3.1b shows the graph representation of a portion ("triangle on square") of the input graph for this example and also represents the substructure minimizing the description length of the graph. Figure 3.1c shows the input example after being compressed by the substructure.

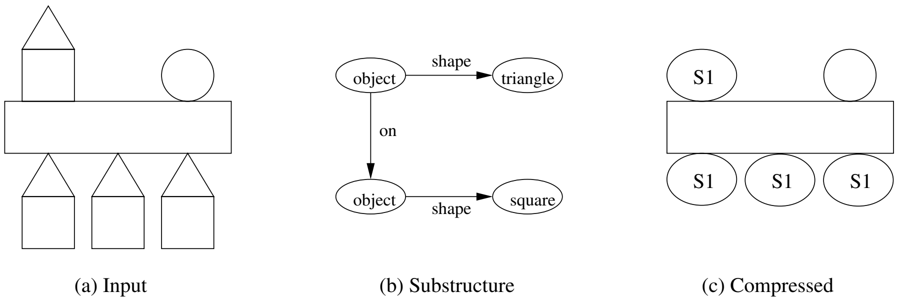

Fig. 3.1. Example of Subdue's substructure discovery capability.

3.3.2 Graph-Based Clustering

Given the ability to find a prevalent subgraph pattern in a larger graph and then compress the graph with this pattern, iterating over this process until the graph can no longer be compressed will produce a hierarchical, conceptual clustering of the input data. On the *ith* iteration, the best subgraph S_i is used to compress the input graph, introducing new vertices labeled Si in the graph input to the next iteration. Therefore, any subsequently-discovered subgraph S_j can be defined in terms of one or more S_i, where $i < j$. The result is a lattice, where each cluster can be defined in terms of more than one parent subgraph. For example, Figure 3.2 shows such a clustering done on a portion of DNA. See [13] for more information on graph-based clustering.

3.4 Supervised Learning from Graphs

Extending a graph-based discovery approach to perform supervised learning involves, of course, the need to handle negative examples (focusing on the two-class scenario). In the case of a graph the negative information can come in two forms. First, the data may be in the form of numerous small graphs, or graph transactions, each labelled either positive or negative. Second, data may be composed of two large graphs: one positive and one negative.

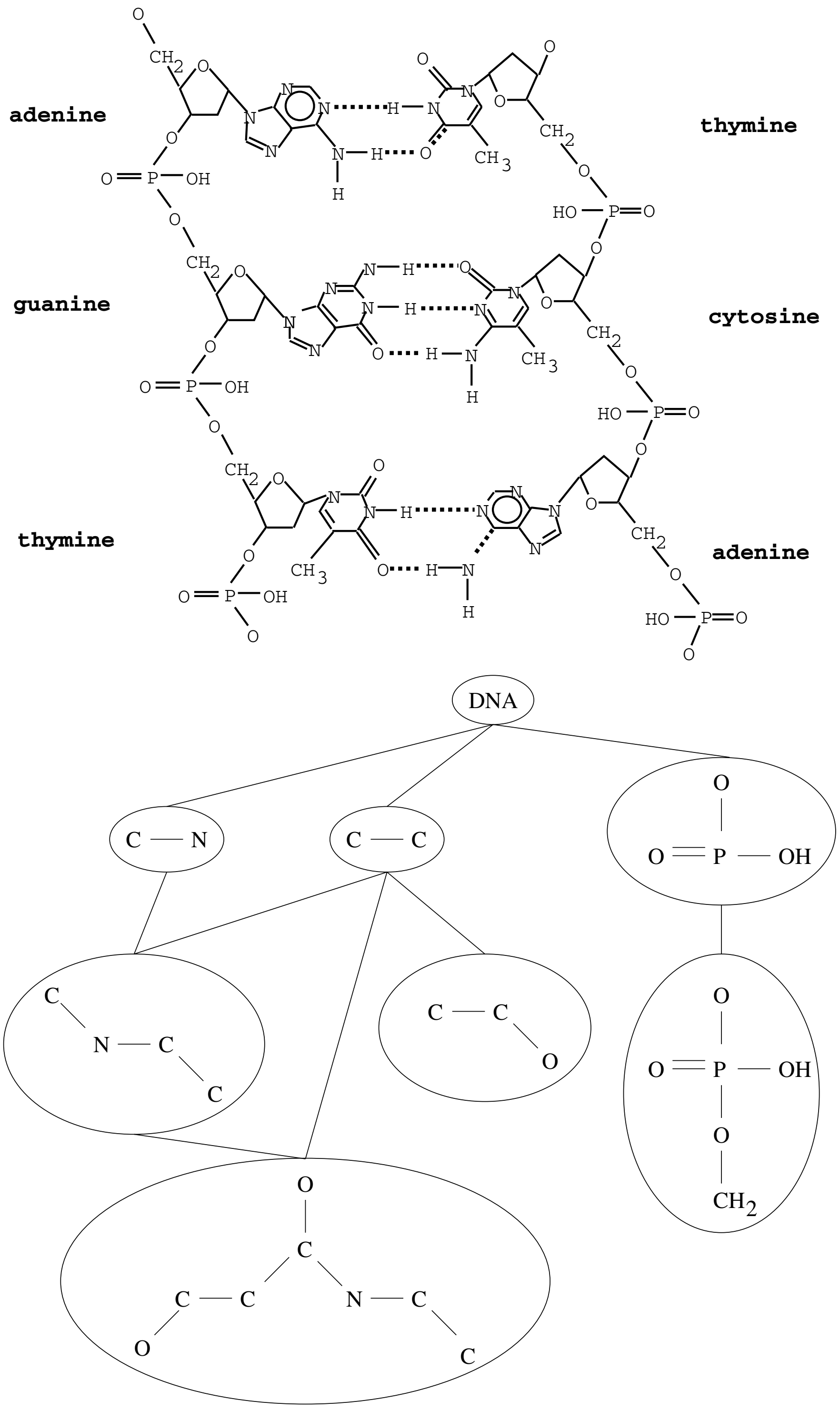

Fig. 3.2. Example of Subdue's clustering (bottom) on a portion of DNA (top).

The first scenario is closest to the standard supervised learning problem in that we have a set of clearly defined examples. Figure 3.3a depicts a simple set of positive and negative examples. Let G^+ represent the set of positive graphs, and G^- represent the set of negative graphs. Then, one approach to supervised learning is to find a subgraph that appears often in the positive graphs, but not in the negative graphs. This amounts to replacing the information-theoretic measure with an error-based measure. For example, we would find a subgraph S that minimizes

$$\frac{|\{g \in G^+ | S \not\subseteq g\}| + |g \in G^- | S \subseteq g\}|}{|G^+| + |G^-|},$$

where $S \subseteq g$ means S is isomorphic to a subgraph of g. The first term of the numerator is the number of false negatives and the second term is the number of false positives.

This approach will lead the search toward a small subgraph that discriminates well, e.g., the subgraph in Figure 3.3b. However, such a subgraph does not necessarily compress well, nor represent a characteristic description of the target concept. We can bias the search toward a more characteristic description by using the information-theoretic measure to look for a subgraph that compresses the positive examples, but not the negative examples. If $I(G)$ represents the description length (in bits) of the graph G, and $I(G|S)$ represents the description length of graph G compressed by subgraph S, then we can look for an S that minimizes $I(G^+|S) + I(S) + I(G^-) - I(G^-|S)$, where the last two terms represent the portion of the negative graph incorrectly compressed by the subgraph. This approach will lead the search toward a larger subgraph that characterizes the positive examples, but not the negative examples, e.g., the subgraph in Figure 3.3c.

Finally, this process can be iterated in a set-covering approach to learn a disjunctive hypothesis. If using the error measure, then any positive example containing the learned subgraph would be removed from subsequent iterations. If using the information-theoretic measure, then instances of the learned subgraph in both the positive and negative examples (even multiple instances per example) are compressed to a single vertex. See [9] for more information on graph-based supervised learning.

3.5 Incremental Discovery from Streaming Data

Many challenging problems require processing and assimilation of periodic increments of new data, which provides new information in addition to that which was previously processed. We introduce our first enhancement of Subdue, called Incremental-Subdue (I-Subdue), which summarizes discoveries from previous data increments so that the globally-best patterns can be computed by examining only the new data increment.

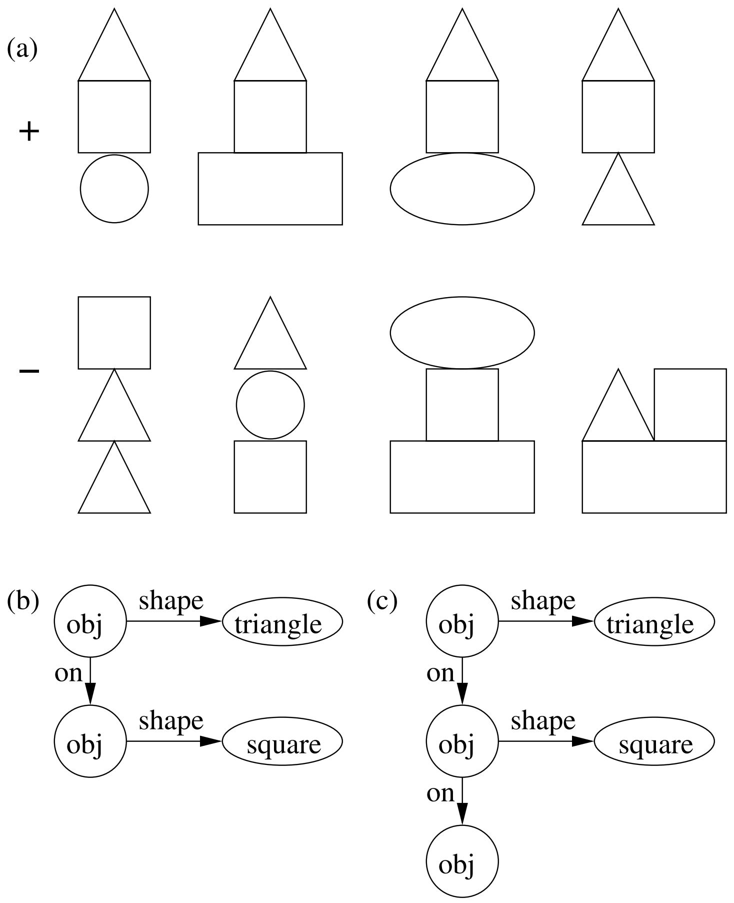

Fig. 3.3. Graph-based supervised learning example with (a) four positive and four negative examples, (b) one possible graph concept and (c) another possible graph concept.

In our work, we assume that data is received in incremental blocks, as is the case for many long-term analytical tasks. Continuously reprocessing the accumulated graph after each increment would be intractable, so instead we wish to develop methods to iteratively refine the substructure discoveries with a minimal amount of reexamination of old data so that the globally-best patterns can be identified based on previous local discoveries.

This work is related to the problem of online sequential learning in which training data is received sequentially [3, 8]. Because learning must start again with each increment, a summary must be generated of prior data to lighten the computational load in building a new model. Online approaches also deal with this incremental mining problem, but restrict the problem to itemset data and assume the data arrives in complete and independent units [1, 7, 11].

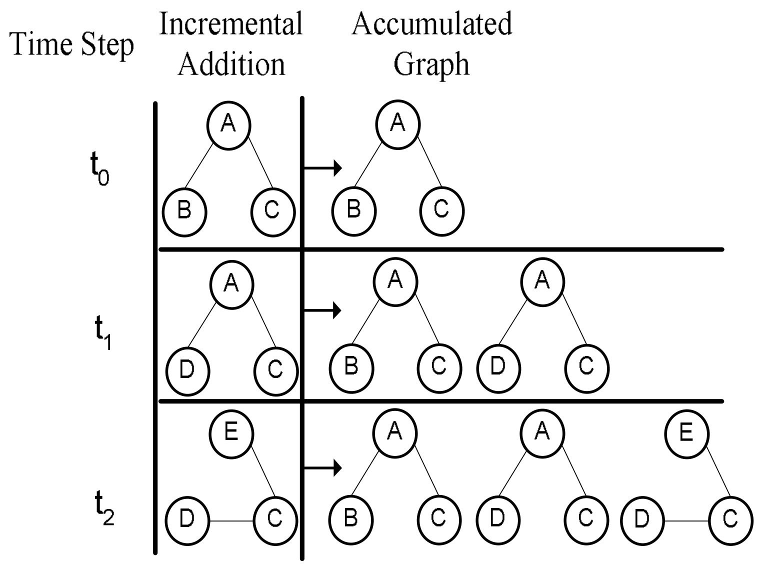

Fig. 3.4. Incremental data can be viewed as a unique extension to the accumulated graph.

In our approach, we view each new data increment as a distinct data structure. Figure 3.4 illustrates one conceptual approach to mining sequential data, where each new increment received at time step t_i is considered independently of earlier data increments so that the accumulation of these structures is viewed as one large, but disconnected, graph. The original Subdue algorithm would still work equally well if we applied it to the accumulated graph after each new data increment is received. The obstacle is the computational burden required for repeated full batch processing.

The concept depicted in Figure 3.4 can be intuitively applied to real problems. For example, a software agent deployed to assist an intelligence analyst would gradually build up a body of data as new information streams in over time. This streaming data could be viewed as independent increments from which common structures are to be derived. Although the data itself may be generated in very small increments, we would expect to accumulate some minimum amount before we mine it. Duplicating nodes and edges in the accumulated graph serves the purpose of giving more weight to frequently-repeated patterns.

3.5.1 Sequential Discovery

Storing all accumulated data and continuing to periodically repeat the entire structure discovery process is intractable both from a computational perspective and for data storage purposes. Instead, we wish to devise a method by which we can discover structures from the most recent data increment and simultaneously refine our knowledge of the globally-best substructures discovered so far. However, we can often encounter a situation where sequential applications of Subdue to individual data increments will yield a series of locally-best substructures that are not the globally-best substructures, that would be found if the data were evaluated as one aggregate block.

Figure 3.5 illustrates an example where Subdue is applied sequentially to each data increment as it is received. At each increment, Subdue discovers the best substructure for the respective data increment, which turns out to be only a local best. However, if we aggregate the same data, as depicted in Figure 3.6, and then apply the baseline Subdue algorithm we get a different best substructure, which in fact is globally best. This is illustrated in Figure 3.7. Although our simple example could easily be aggregated at each time step, realistically large data sets would be too unwieldy for this approach.

In general, sequential discovery and action brings with it a set of unique challenges, which are generally driven by the underlying system that is generating the data. One problem that is almost always a concern is how to re-evaluate the accumulated data at each time step in the light of newly-added data. There is a tradeoff between the amount of data that can be stored and re-evaluated, and the quality of the result. A summarization technique is often employed to capture salient metrics about the data. The richness of this summarization is a tradeoff between the speed of the incremental evaluation and the range of new substructures that can be considered.

3.5.2 Summarization Metrics

We need to develop a summarization metric that can be maintained from each incremental application of Subdue and will allow us to derive the globally-best substructure without reapplying Subdue when new data arrives. To accomplish this goal, we rely on a few artifacts of Subdue's discovery algorithm. First, Subdue maintains a list of the n best substructures discovered from any data set, where n is configurable by the user.

Second, we modify the Compression measure used by Subdue, as shown in Equation (3.2).

$$Compress_m(S_i) = \frac{DL(S_i) + \sum_{j=1}^{m} DL(G_j|S_i)}{\sum_{j=1}^{m} DL(G_j)} \tag{3.2}$$

I-Subdue calculates compression achieved by a particular substructure, S_i, through the current data increment m. The $DL(S_i)$ term is the description

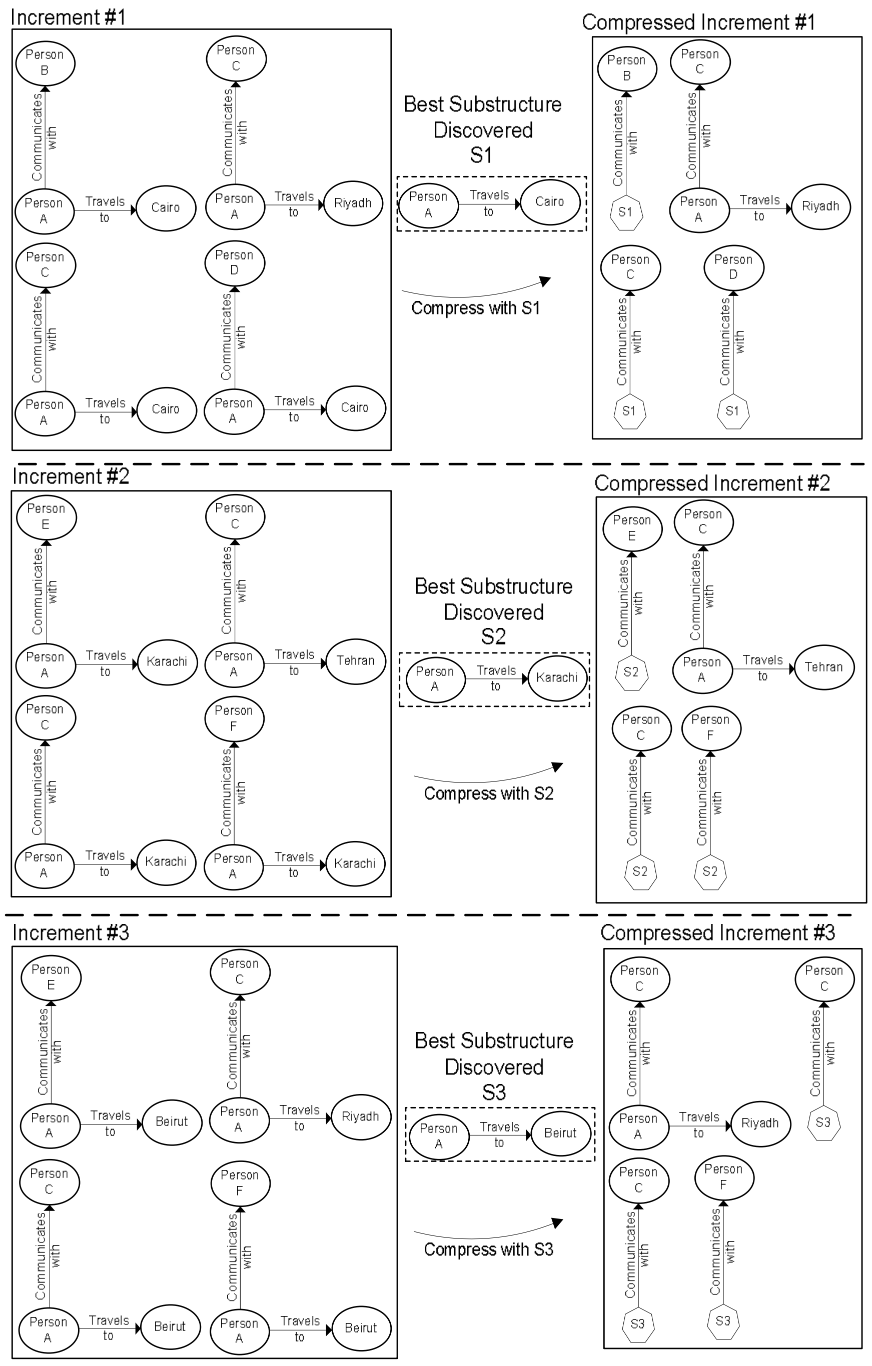

Fig. 3.5. Three data increments received serially and processed individually by Subdue. The best substructure is shown for each local increment.

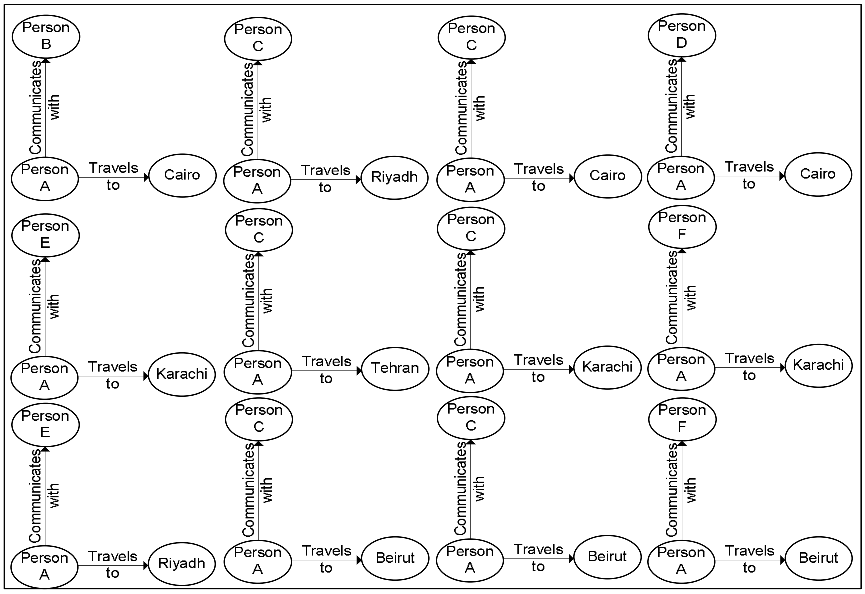

Fig. 3.6. Accumulated graph for Subdue batch processing.

length of the substructure, S_i, under consideration. The term $\sum_{j=1}^{m} DL(G_j|S_i)$ represents the description length of the accumulated graph after it is compressed by substructure S_i. Finally, the term $\sum_{j=1}^{m} DL(G_j)$ represents the full description length of the accumulated graph. I-Subdue then can re-evaluate substructures using Equation (3.3) (an inverse of Equation (3.2)), choosing the one with the lowest value as globally best.

$$argmax(i) \left[\frac{DL(S_i) + \sum_{j=1}^{m} DL(G_j|S_i)}{\sum_{j=1}^{m} DL(G_j)} \right] \tag{3.3}$$

The process of computing the global substructure value takes place in addition to the normal operation of Subdue on the isolated data increment. We only need to store the requisite description-length metrics after each iteration for use in our global computation.

As an illustration of our approach, consider the results from the example depicted in Figure 3.6. The top $n = 3$ substructures from each iteration are shown in Figure 3.8. Table 3.1 lists the values returned by Subdue from the local top n substructures discovered in each increment. The second best substructures in increments 2 and 3 (S_{22}, S_{32}) are the same as the second best substructure in increment 1 (S_{12}), which is why the column corresponding

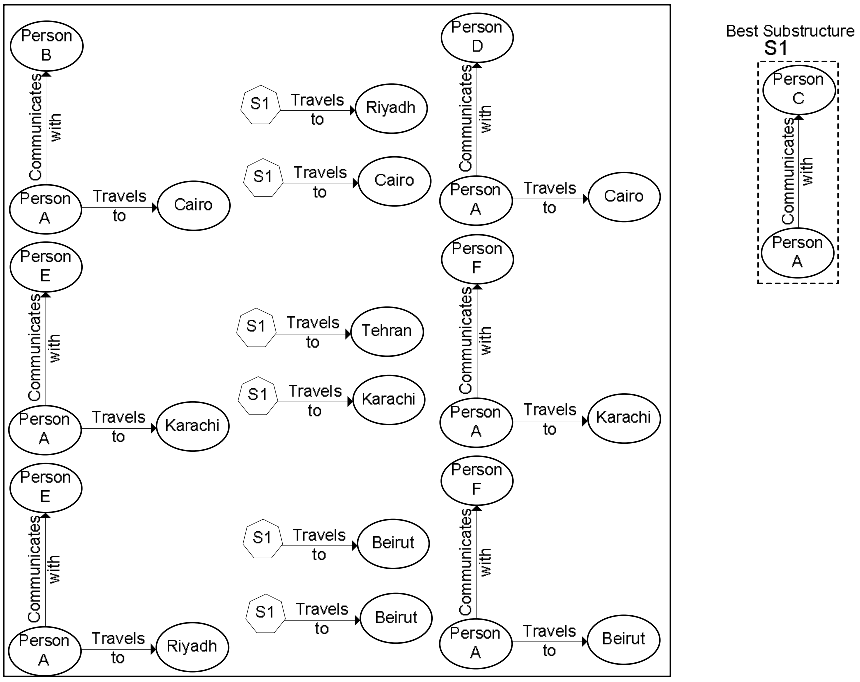

Fig. 3.7. Result from applying Subdue to the three aggregated data increments.

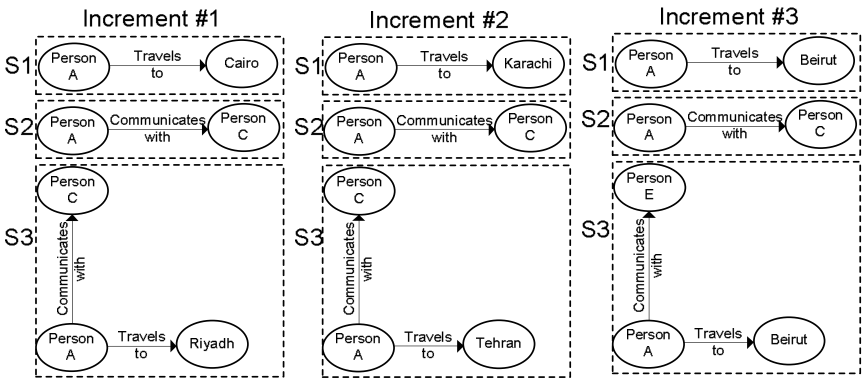

Fig. 3.8. The top n=3 substructures from each local increment.

Table 3.1. Substructure values computed independently for each iteration.

Increment	Substructures from Increment #1			Substructures from Increment #2		Substructures from Increment #3	
	S_{11}	S_{12}	S_{13}	S_{21}	S_{23}	S_{31}	S_{33}
1	**1.2182**	1.04808	0.9815				
2		1.04808		**1.21882**	0.981511		
3		1.03804				**1.15126**	0.966017

Table 3.2. Using I-Subdue to calculate the global value of each substructure.

Increment	Substructures from Increment #1			Substructures from Increment #2		Substructures from Increment #3		
	S_{11}	S_{12}	S_{13}	S_{21}	S_{23}	S_{31}	S_{33}	$DL(G_j)$
1	**1.2182**	1.04808	0.9815					117
2	1.0983	**1.1235**	0.9906	1.0986	0.9906			117
3	1.0636	**1.1474**	0.9937	1.0638	0.9937	1.0455	0.9884	116
$DL(S_i)$	15	15	25.7549	15	25.7549	15	26.5098	

to S_{12} has a value for each iteration. The values in Table 3.1 are the result of the compression evaluation metric from Equation (3.1). The locally-best substructures illustrated in Figure 3.5 have the highest values overall.

Table 3.2 depicts our application of I-Subdue to the increments from Figure 3.5. After each increment is received, we apply Equation (3.3) to select the globally-best substructure. The values in Table 3.2 are the inverse of the compression metric from Equation (3.2). As an example, the calculation of the compression metric for substructure S_{12} after iteration 3 would be $\frac{DL(S_{12})+DL(G_1|S_{12})+DL(G_2|S_{12})+DL(G_3|S_{12})}{DL(G_1)+DL(G_2)+DL(G_3)}$. Consequently the value of S_{12} would be (117 + 117 + 116) / (15 + 96.63 + 96.63 + 96.74) = 1.1474.

For this computation, we rely on the metrics computed by Subdue when it evaluates substructures in a graph, namely the description length of the discovered substructure, the description length of the graph compressed by the substructure, and the description length of the graph. By storing these values after each increment is processed, we can retrieve the globally-best substructure using Equation (3.3). In circumstances where a specific substructure is not present in a particular data increment, such as S_{31} in iteration 2, then $DL(G_2|S_{31}) = DL(G_2)$ and the substructure's value would be calculated as (117 + 117 + 116) / (15 + 117 + 117 + 85.76) = 1.0455.

3.5.3 Experimental Evaluation

To illustrate the relative value of I-Subdue with respect to performance in processing incremental data, we have conducted experiments with a synthetic data generator. This data generator takes as input a library of data labels, configuration parameters governing the size of random graph patterns and one or more specific substructures to be embedded within the random data. Connectivity can also be controlled.

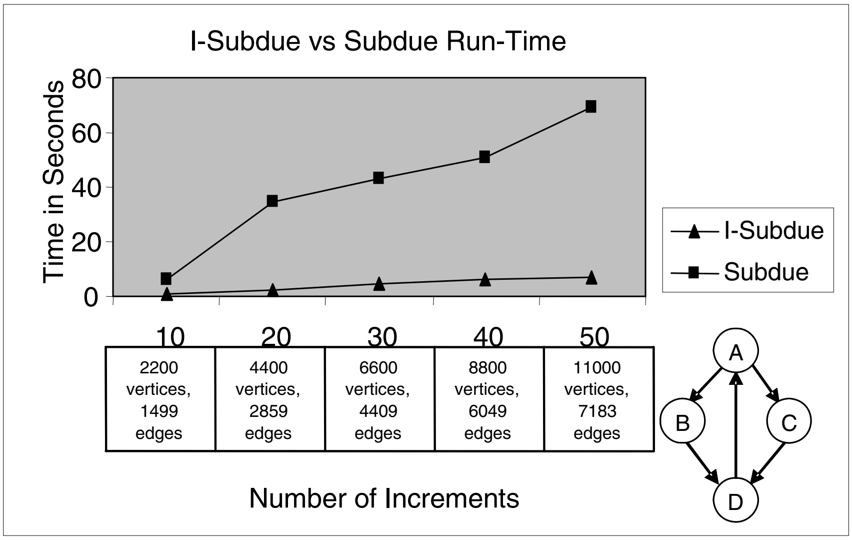

Fig. 3.9. Comparison of I-Subdue with Subdue on 10–50 increments, each with 220 new vertices and 0 or 1 outgoing edges.

For the first experiment, illustrated in Figure 3.9, we compare the performance of I-Subdue to Subdue at benchmarks ranging from 10 to 50 increments. Each increment introduced 220 new vertices, within which five instances of the four-vertex substructure pictured in Figure 3.9 were embedded. The quality of the result, in terms of the number of discovered instances, was the same.

The results from the second graph are depicted in Figure 3.10. For this experiment, we increased the increment size to 1020 vertices. Each degree value between 1 and 4 was shown with 25% probability, which means that on average there are about twice as many edges as vertices. This more densely connected graph begins to illustrate the significance of the run-time difference between I-Subdue and Subdue. Again, five instances of the four-vertex substructure shown in Figure 3.10 were embedded within each increment. The discovery results were the same for both I-Subdue and Subdue with the only qualitative difference being in the run time.

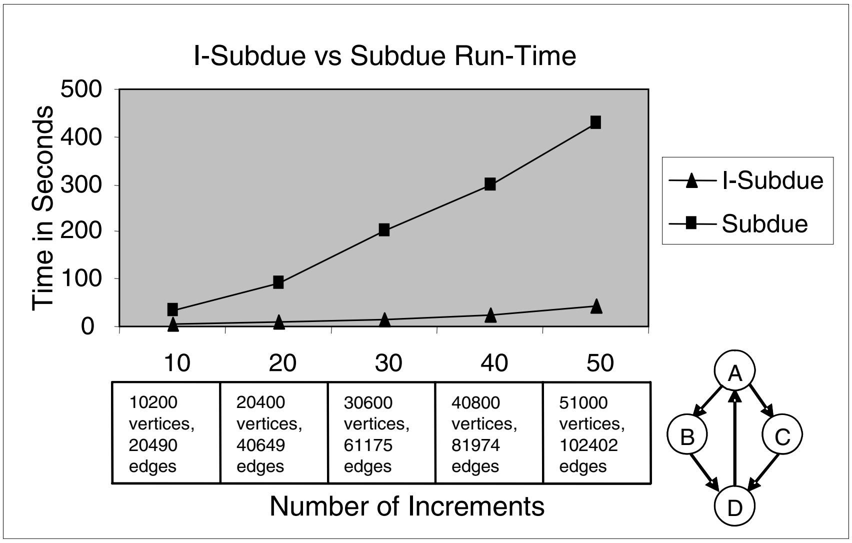

Fig. 3.10. Comparison of I-Subdue with Subdue on 10–50 increments, each with 1020 new vertices and 1 to 4 outgoing edges.

3.5.4 Learning from Supervised Graphs

In a highly relational domain, the positive and negative examples of a concept are not easily separated. We call such a graph a *supervised graph*, in that the graph as a whole contains embedded class information which may not easily be separated into individual labeled components. For example, consider a social network in which we seek to find relational patterns distinguishing various income levels. Individuals of a particular income level can appear anywhere in the graph and may be related to individuals at other income levels, so we cannot easily partition the graph into separate training cases without potentially severing the target relationships.

This scenario presents a challenge to any data mining system, but especially to a graph-based relational learning system, where clearly classified data (data labeled with a class value) may be tightly related to less clearly classified data. This is the second challenge discussed in this chapter. We are investigating two approaches to this task. We assume that the class values of certain vertices and edges are specified in the input data file. Not all vertices and edges will have such a value, as some may provide supplementary supporting information.

For the first approach, we rely upon a cost mechanism available in Subdue. A cost mechanism was added because expenses might be associated with the retrieval of portions of data. For example, adding personal details such as credit history to our social network can enhance the input data, but may be acquired at a price in terms of money, time, or other resources. To implement the cost feature, the cost of specific vertices and edges is specified in the input file. The cost for substructure S averaged over all of its instances, $Cost(S)$, is then combined with the MDL value of S using the equation $E(S) = (1 - Cost(S)) \times MDL(S)$. The evaluation measure, $E(S)$, determines the overall value of the substructure and is used to order candidate substructures.

Class membership in a supervised graph can now be treated as a cost, which varies from no cost for clearly positive members to +1 for clearly negative members. As an example, we consider the problem of learning which regions of the ocean surface can expect a temperature increase in the next time step. Our data set contains gridded sea surface temperatures (SST) derived from NASA's Pathfinder algorithm and a five-channel Advanced Very High Resolution Radiometer instrument. The data contains location, time of year, and temperature data for each region of the globe.

The portion of the data used for training is represented as a graph with vertices for each month, discretized latitude and longitude values, hemisphere, and change in temperature from one month to the next. Vertices labelled with "increase" thus represent the positive examples and "decrease" or "same" labels represent negative examples. A portion of the graph is shown in Figure 3.11. The primary substructure discovered by Subdue for this data set reports the rule that when there are two regions in the Southern hemisphere, one just north of the other, an increase in temperature can be expected for the next month in the southernmost of the two regions. Using three-fold cross validation experimentation, Subdue classified this data set with 71% accuracy.

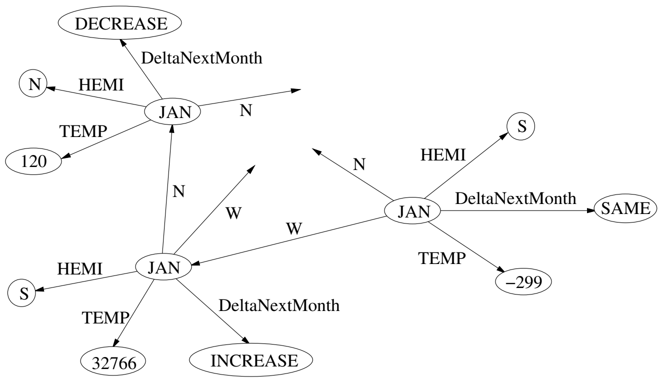

Fig. 3.11. Graph representation of a portion of NASA's SST data.

The second approach we intend to explore involves modifying the MDL encoding to take into account the amount of information necessary to describe the class membership of the compressed graph. Substructures would now be discovered that not only compress the raw data of the graph but also express class membership for vertices and edges within the graph.

3.6 Conclusions

There are several future directions for our graph-based relational learning research that will improve our ability to handle such challenging data as described in this chapter. The incremental discovery technique described in this chapter did not address data that is connected across increment boundaries. However, many domains will include event correlations that transcend multiple data iterations. For example, a terrorist suspect introduced in one data increment may be correlated to events that are introduced in later increments. As each data increment is received it may contain new edges that extend from vertices in the new data increment to vertices received in previous increments. We are investigating techniques of growing substructures across increment boundaries. We are also considering methods of detecting changes in the strengths of substructures across increment boundaries, that could represent concept shift or drift.

The handling of supervised graphs is an important direction for mining structural data. To extend our current work, we would like to handle embedded instances without a single representative instance node (the "increase" and "decrease" nodes in our NASA example) and instances that may possibly overlap.

Finally, improved scalability of graph operations is necessary to learn patterns, evaluate their accuracy on test cases and, ultimately, to use the patterns to find matches in future intelligence data. The graph and subgraph isomorphism operations are a significant bottleneck to these capabilities. We need to develop faster and approximate versions of these operations to improve the scalability of graph-based relational learning.

Acknowledgments: This research is sponsored by the Air Force Research Laboratory (AFRL) under contract F30602-01-2-0570. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of AFRL or the United States Government.

References

[1] Agrawal, R. and G. Psaila, 1995: Active data mining. *Proceedings of the Conference on Knowledge Discovery in Databases and Data Mining.*

[2] Agrawal, R. and R. Srikant, 1994: Fast algorithms for mining association rules. *Proceedings of the Twentieth Conference on Very Large Databases*, 487–99.

[3] Blum, A., 1996: On-line algorithms in machine learning. *Proceedings of the workshop on on-line algorithms.*

[4] Cook, D. J. and L. B. Holder, 1994: Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, **1**, 231–55.

[5] — 2000: Graph-based data mining. *IEEE Intelligent Systems*, **15**, 32–41.

[6] Dzeroski, S. and N. Lavrac, eds., 2001: *Relational Data Mining.* Springer.

[7] Fang, H., W. Fan, P. Yu and J. Han, 2003: Mining concept-drifting data streams using ensemble classifiers. *Proceedings of the Conference on Knowledge Discovery and Data Mining.*

[8] Friedman, N. and M. Goldszmidt, 1997: Sequential update of Bayesian network structure. *Proceedings of the Conference on Uncertainty in Artificial Intelligence.*

[9] Gonzalez, J., L. Holder and D. Cook, 2002: Graph-based relational concept learning. *Proceedings of the Nineteenth International Conference on Machine Learning.*

[10] Holder, L. B. and D. J. Cook, 2003: Graph-based relational learning: Current and future directions. *ACM SIGKDD Explorations*, **5**, 90–93.

[11] Hulten, G., L. Spencer and P. Domingos, 2001: Mining time-changing data streams. *Proceedings of the Conference on Knowledge Discovery and Data Mining.*

[12] Inokuchi, A., T. Washio and H. Motoda, 2003: Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, **50**, 321–54.

[13] Jonyer, I., D. Cook and L. Holder, 2001: Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research*, **2**, 19–43.

[14] Kashima, H. and A. Inokuchi, 2002: Kernels for graph classification. *Proceedings of the International Workshop on Active Mining.*

[15] Kuramochi, M. and G. Karypis, 2001: Frequent subgraph discovery. *Proceedings of the First IEEE Conference on Data Mining.*

[16] Muggleton, S., ed., 1992: *Inductive Logic Programming.* Academic Press, San Diego, CA, USA.

[17] Rissanen, J., 1989: *Stochastic Complexity in Statistical Inquiry.* World Scientific, Singapore.

[18] US Senate, 2002: Joint inquiry into intelligence community activities before and after the terrorist attacks of September 11, 2001. S. Rept.107-351.

[19] Yan, X. and J. Han, 2002: gSpan: Graph-based substructure pattern mining. *Proceedings of the International Conference on Data Mining.*

[20] Yoshida, K., H. Motoda and N. Indurkhya, 1994: Graph-based induction as a unified learning framework. *Journal of Applied Intelligence*, **4**, 297–328.