AFRL-IF-RS-TR-2006-249
Final Technical Report
July 2006

GRAPH-BASED STRUCTURAL PATTERN LEARNING

The University of Texas at Arlington

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2006-249 has been reviewed and is approved for publication.

APPROVED: /s/

DEBORAH A. CERINO
Project Engineer

FOR THE DIRECTOR: /s/

JOSEPH CAMERA
Chief, Information & Intelligence Exploitation Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUL 06	Final	Sep 01 – Mar 06

4. TITLE AND SUBTITLE
GRAPH-BASED STRUCTURAL PATTERN LEARNING

5a. CONTRACT NUMBER
F30602-01-2-0570

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
31011G

6. AUTHOR(S)
Lawrence B. Holder, Diane J. Cook

5d. PROJECT NUMBER
EELD

5e. TASK NUMBER
01

5f. WORK UNIT NUMBER
08

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The University of Texas at Arlington
Box 19015
Arlington Texas 76019-0015

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFED
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2006-249

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA #06-505

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The main objective of this project was to design, implement and evaluate new methods for performing pattern learning on structured data represented as graphs and evaluate their application to structural, relational databases relevant to the Evidence Assessment, Grouping, Linking and Evaluation (EAGLE) program. This work builds on existing methods for graph-based knowledge discovery and concept learning implemented in the SUBDUE structural pattern learning system. The graph-based structural pattern learning algorithm was extended to perform structural concept learning and structural, hierarchical conceptual clustering. The resulting system was evaluated using several structural databases, including those with known structural patterns, those of relevance to the target domains of the EAGLE program, and those developed as challenge problems within the EAGLE program.

15. SUBJECT TERMS
relational pattern-learning, graphical patterns, structural databases

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Deborah A. Cerino
U	U	U	UL	21	19b. TELEPONE NUMBER (*Include area code)*

Table of Contents

List of Figures

List of Tables

1. Introduction

The ability to learn patterns in relational data has become a crucial challenge in many security-related domains. For example, the U.S. 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" [1], 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.

The main objective of this project was to design, implement and evaluate new methods for performing pattern learning on structured data represented as graphs and evaluate their application to structural, relational databases relevant to the *Evidence Assessment, Grouping, Linking and Evaluation (*EAGLE) program. This work builds on existing methods for graph-based knowledge discovery and concept learning implemented in the SUBDUE structural pattern learning system. The graph-based structural pattern learning algorithm was extended to perform structural concept learning and structural, hierarchical conceptual clustering. The resulting system was evaluated using several structural databases, including those with known structural patterns, those of relevance to the target domains of the EAGLE program, and those developed as challenge problems within the EAGLE program.

The specific objectives of the project were to improve the performance of graph-based structural pattern learning both in terms of running time on large graphs and accuracy in predicting threat versus non-threat groups and events. Several performance improvements were achieved by focusing on the algorithm, especially on the primitive graph operations like isomorphism tests. Additional speedups were obtained by processing incremental graph data arriving in a stream. We also desired to build upon the SUBDUE system to identify anomalies in graph data, and to learn concepts from training examples embedded in a single graph.

An additional objective of this project was to transition the SUBDUE pattern learning technology developed by UTA to the intelligence community. One route toward this end was the transition of SUBDUE to the Research and Development Experimental Collaboration (RDEC) Facility. We also pursued an opportunity in collaboration with Cycorp and their Structured Knowledge Source Integration (SKSI) technology, which allowed us to build graphs from multiple databases via a common Cyc-based interface. We supported the development of the hypothesis specifications effort (HARM) by providing a specification of the pattern language used to express the graphical patterns learned by SUBDUE. We laid out a similar goal to support the TANGRAM effort to develop a model of an automated end-to-end information discovery system. Finally, to avoid the preliminary step of converting relational databases to graphs, we investigated the tight integration of SUBDUE within relational database management system (RDBMS) technologies.

2. Background

Numerous approaches have been developed for discovering concepts in databases using a linear, attribute-value representation. Although much of the data collected today has an explicit or implicit structural component (e.g., spatial or temporal), few discovery systems are designed to handle this type of data. Those systems that deal with structural databases typically do not scale for large databases due to the increased combinatorics inherent in the richer representations.

We have introduced a method for discovering substructures in structural databases using the minimum description length (MDL) principle [3] implemented in the SUBDUE system [2]. Unlike many existing methods, SUBDUE is designed to discover knowledge in a supervised or unsupervised fashion. In contrast to approaches that are designed for a single application domain, SUBDUE is devised for general-purpose automated discovery with or without domain knowledge. Hence, the method can be applied to many structural domains.

SUBDUE discovers patterns in structural data represented as a labeled graph and performs various types of data mining on the 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.

SUBDUE operates in three main modes: discovery, clustering and supervised learning. In **discovery** mode SUBDUE uses heuristic search guided by MDL to find patterns minimizing the description length of the entire graph compressed with the pattern. Once a pattern is found, SUBDUE can compress the graph using this pattern and repeat the process on the compressed graph to look for more abstract patterns, possibly defined in terms of previously-discovered patterns. Figure 1 shows how SUBDUE finds four instances of a pattern S_1 in the input graph, the graph compressed with pattern S_1, the pattern S_2 found in the next iteration, and the graph compressed with pattern S_2.

The ability of SUBDUE to iteratively discover patterns and compress the graph can be used to generate a **clustering** of the input graph. Essentially, clustering mode forces SUBDUE to iterate until the input graph can be compressed no further. The resulting patterns form a cluster lattice, such that if a pattern S is defined in terms of one or more previously-discovered patterns, then these patterns are parents of S in the lattice. In the example in Figure 1, S_2 is defined in terms of S_1. Each cluster in the lattice is defined conceptually by the graphical pattern discovered by SUBDUE, producing a hierarchical, conceptual clustering of the input data.

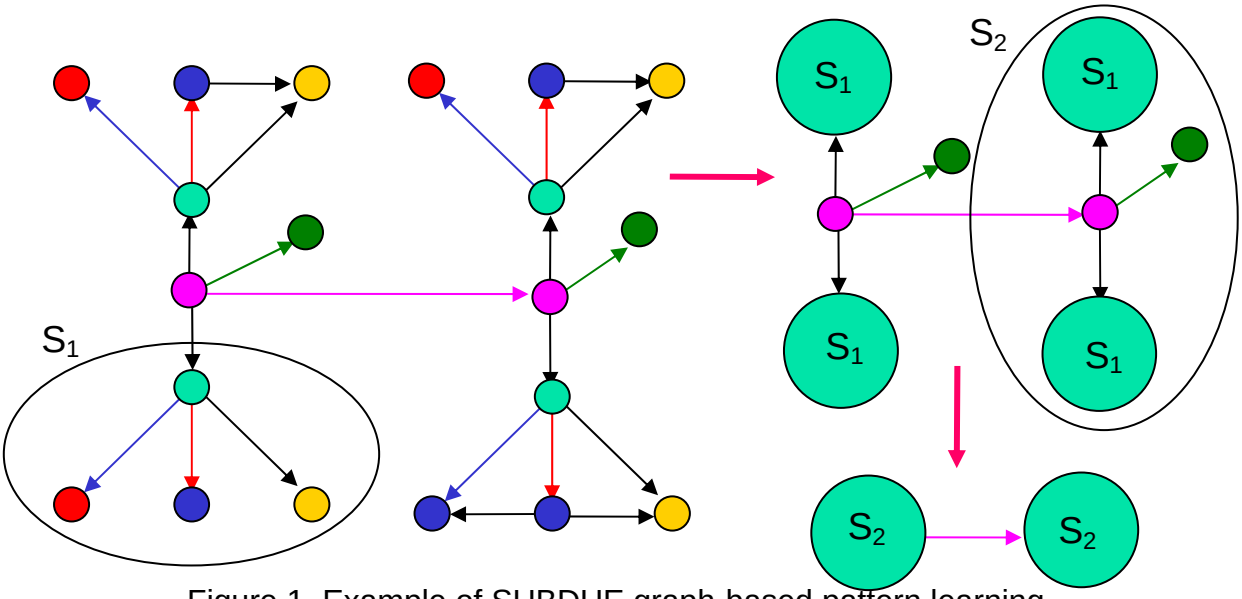

Figure 1. Example of SUBDUE graph-based pattern learning.

If graphs depicting both positive and negative examples of a phenomenon are available for input, then SUBDUE enters **supervised-learning** mode, searching for a pattern that compresses the positive graphs, but not the negative graphs. For example, given positive graphs describing criminal networks and negative graphs describing normal social networks, SUBDUE can learn patterns distinguishing the two, and these patterns can be used as a predictive model to identify emerging criminal networks.

SUBDUE offers a number of additional features. Users can input background knowledge in the form of predefined patterns. SUBDUE first compresses the input graph with these patterns before beginning discovery. SUBDUE uses an inexact graph match to find instances of a pattern, so the instances do not all have to exactly match the pattern. SUBDUE can be run on a distributed network by partitioning the input graph across multiple processors.

2.1. Algorithm

SUBDUE uses a polynomial-time beam search for its discovery algorithm, summarized in Figure 2. The goal of SUBDUE's search is to find the subgraph which yields the best compression of the input graph. The initial state of the search is the set of subgraphs consisting of all uniquely labeled vertices. The only search operator is the Extend Subgraph operator, which extends a subgraph in all possible ways by one edge and a vertex or by a single edge if both vertices are already in the subgraph. Substructures are kept on an ordered queue of length determined by the beam width.

To evaluate subgraphs the Minimum Description Length (MDL) principle is used, which states that the best theory is one that minimizes the description length of a database. Using this principle, a substructure is evaluated by its ability to minimize the value of the description length of the graph when compressed using the substructure.

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

```
SUBDUE(graph G, int Beam, int Limit )
        queue Q = {v | v has a unique label in G}
        bestSub = first substructure in Q
        repeat
            newQ = {}
            for each S in Q
                newSubs = S extended in all possible ways
                newQ = best Beam substructures in (newQ U newSubs)
                Limit = Limit - 1
                evaluate substructures in newQ by compression of G
                if best substructure in Q better than bestSub
                then bestSub = best substructure in Q
        until Q is empty or Limit <= 0
        return bestSub
```

Figure 2. Main SUBDUE discovery algorithm.

terminates and returns the list of best subgraphs, the graph can be compressed using the best subgraph. To compress a graph, all instances of the subgraph are replaced by a single representative vertex. Incoming and outgoing edges to and from the replaced subgraph connect to the new vertex that represents the subgraph. 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 discovered patterns. As a result, this hierarchy provides varying levels of interpretation that can be accessed based on the specific goals of the data analysis.

SUBDUE has been used as an unsupervised algorithm to find patterns in a number of structural databases. As examples, SUBDUE has found patterns in program source code that, upon compression, improve the modularity of the program. We have discovered patterns in CAD circuit data that represent functional concepts and aid experts in understanding of the data. In the area of computational biology, SUBDUE identified structural patterns in primary, secondary, and tertiary structures of three categories of proteins obtained from the Protein Data Bank. Additional discoveries have been made and validated by experts in the areas of geology, image analysis, Chinese character databases, and chemical reaction chains.

2.2. Interface

SUBDUE uses a simple text-based format for the input graphs and the graphical output patterns that can be easily converted to other formats for visualization. For example, the representation of the S_1 structure in the above figure would be as follows.

```
v 1 light_green
v 2 red
v 3 blue
```

```
v 4 yellow
d 1 2 blue
d 1 3 red
d 1 4 black
```

The general format consists of vertices and edges. The format for a vertex is "v <#> <label>", where <#> are consecutive integers starting at 1 and <label> is either a string or real number. Edges are either undirected "u <v1> <v2> <label>" or directed "d <v1> <v2> <label>", where <v1> and <v2> are the integer IDs for the connected vertices, and <label> is either a string or real number. Subdue can also accept multiple graphs preceded by an "XP" for a positive graph or an "XN" for a negative graph. If negative graphs are input, Subdue automatically enters supervised learning mode to find a pattern distinguishing the positive graphs from the negative graphs. A more detailed description of SUBDUE's interface can be found in the manual accompanying the download.

2.3. Availability

SUBDUE is written in C and runs on both UNIX and Windows platforms. Source code is available for download from http://ailab.uta.edu/subdue.

3. Performance Evaluation on EAGLE Data

The primary means of evaluating SUBDUE on EAGLE data was the use of IET's data generator, which simulates the evidence available about terrorist groups and their plans prior to their execution. This domain is motivated from an understanding of the real problem of intelligence data analysis. The EAGLE domain consists of a number of concepts, including threat and non-threat actors, threat and non-threat groups, targets, exploitation modes (vulnerability modes are exploited by threat groups, productivity modes are exploited by threat and non-threat groups), capabilities, resources, communications, visits to targets, and transfer of resources between actors, groups and targets. The domain follows a general plan of starting a group, recruiting members with needed capabilities, acquiring needed resources, visiting a target, and then exploiting the target. These events involve various forms of communication and transfer of assets.

The EAGLE simulator generates various threat and non-threat groups, and then executes various vulnerability and productivity exploitations. The simulator generates evidence related to all these events, and this evidence is passed through filters, e.g., varying the degree of observability and noise in the final evidence. This final evidence is stored in an Evidence Data Base (EDB) and is the data from which we are to learn. We address two different relational learning problems in this domain. First, we attempt to learn patterns distinguishing vulnerability exploitation cases (terrorist attacks) from productivity exploitation cases (legitimate uses). Second, we attempt to learn patterns distinguishing threat groups from non-threat groups.

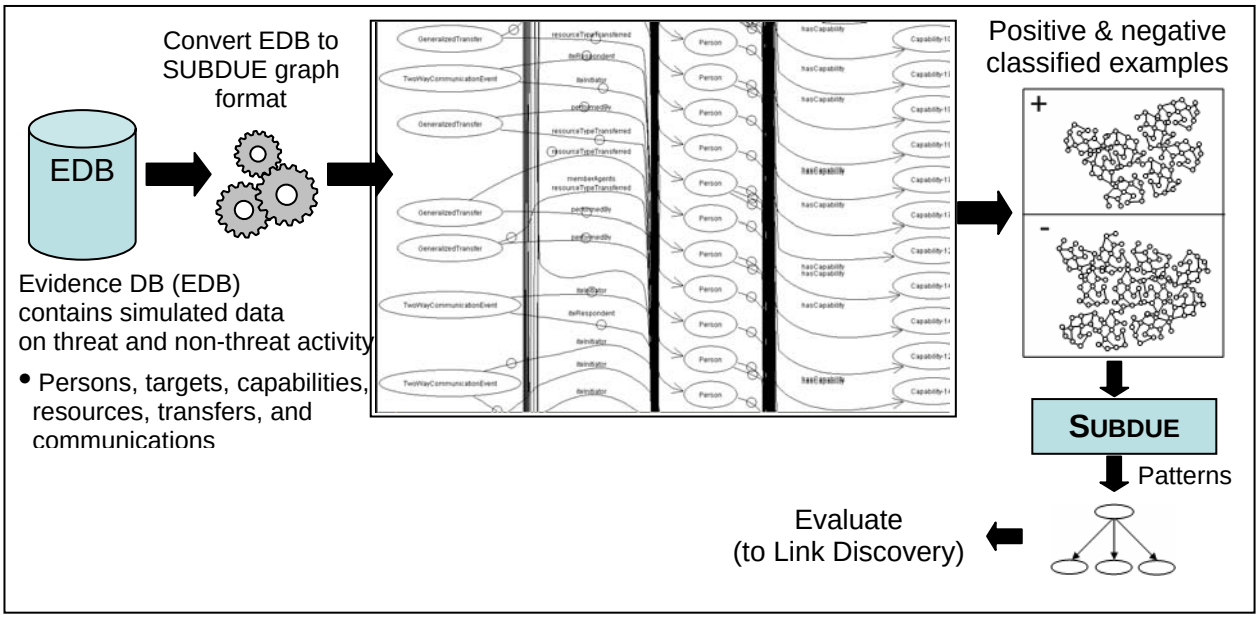

Figure 3. Subdue pattern-learning process on EAGLE data.

Figure 3 illustrates the process of running SUBDUE on the EAGLE data. First, known events and groups in the EDB are extracted and converted to a graph representation. Threat events or groups are made the positive examples and non-threat events or groups are made the negative examples. SUBDUE is passed in the events or groups and then learns a pattern that can distinguish the threat from non-threat events or groups. Ultimately, these patterns are passed on to a link discovery system to help identify potential threat events and groups in new data. However, we perform our own evaluation of the results by measuring the learning time and accuracy of the result.

Table 1 shows the performance of SUBDUE on EAGLE data throughout the program. In 2003, the data was small in size, which allowed SUBDUE to run fast, but accuracy results were poor due to lack of variability in the data. In 2004, the data increased in size by an order of magnitude. SUBDUE was able to improve the accuracy of its learned patterns due to the increased amount of data, but maintained a linear increase in running time based on the graph size. Numerous performance enhancements were made to SUBDUE in 2005, so we re-ran SUBDUE on the same data as in 2004 and found that we were able to reduce the running time by almost 50% with no loss in the accuracy of the learned pattern.

Overall, we see that SUBDUE is able to achieve good performance on the EAGLE domain.

Table 1. Performance evaluation of Subdue on EAGLE data throughout program.

EAGLE Data **2003**	Threat Examples	Non-threat Examples	Vertices	Edges	Accuracy	Time (min)
Mean Event	93	126	25,127	87,371	63%	23
Mean Group	14	59	80,146	142,458	84%	103
Mean	53	92	52,637	114,915	74%	63
EAGLE Data **2004**						
Mean Event	28	280	533,196	630,733	80%	211
Mean Group	22	62	457,209	597,163	85%	1453
Mean	25	185	499,952	616,046	82%	754
EAGLE Data **2005**						
Mean Event	28	280	533,196	630,733	80%	86
Mean Group	22	62	457,209	597,163	85%	813
Mean	25	185	499,952	616,046	82%	404

4. Incremental Processing of Graph Data

As part of the EAGLE project, we specified a goal to develop, implement and evaluate new techniques for incremental graph-based pattern learning. New data will augment the graph-based representation and update candidate patterns, rather than running from scratch on the new graph.

We identified the following specific challenges in performing incremental graph-based pattern learning on streaming data.

- Need to exploit previous discoveries to continuously update globally “best” patterns by only examining the new data increment.
- Instances of structures may extend across temporal boundaries of data increments.
- The patterns being learned may change over time.
- New data may contradict (i.e., delete) old data resulting in modifications/deletions of previously-learned patterns.

In the case when patterns do not extend across the boundaries of data increments, we developed an approach based on maintaining summary statistics of the compression values for the best patterns in each increment. Rather than performing a global re-computation of each pattern’s compression value, we sum the compression values over all the increments for each pattern. Those patterns with highest compression, summed over all increments, are identified as the best patterns for all the

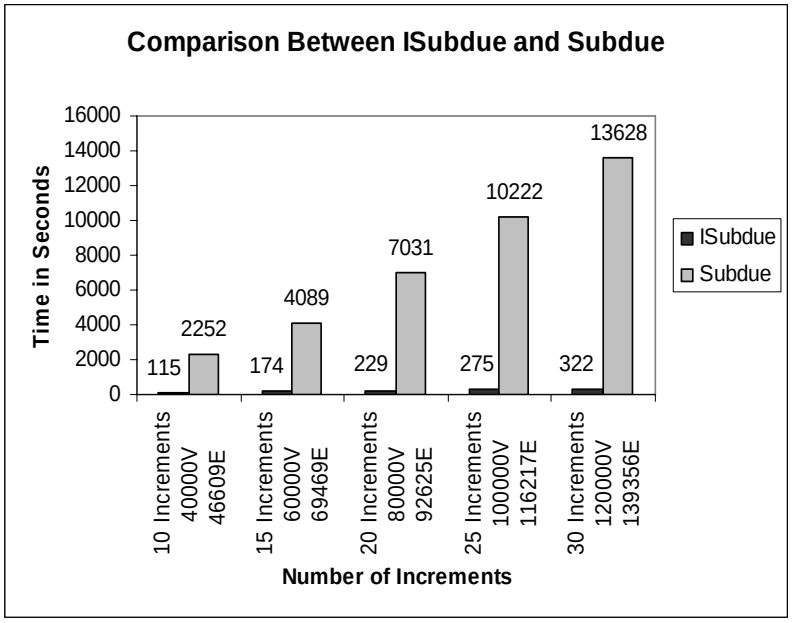

Figure 4. Comparison of run-times for ISubdue and Subdue on increasing numbers of increments.

data. Results have shown that this procedure correctly identifies the globally-best pattern in a majority of the cases.

When it is possible that pattern instances extend across incremental boundaries (e.g., a new actor is introduced into the data, but who has a relationship to another actor introduced earlier in the data stream), then additional steps are required to update the compression statistics based on these additional instances. Specifically, we first identify all edges that extend from a vertex in the current data increment to a vertex in a previous increment. These minimal patterns are then grown one edge at a time, as long as they remain a subgraph of one of the best patterns identified in step one, otherwise they are discarded. Any new instances are added to the support of existing patterns, possibly reordering them and identifying a new best pattern.

Using the above algorithm we have found that incremental Subdue (I-Subdue) can identify the same globally-best patterns, and all their instances (including those that cross increment boundaries), as original SUBDUE, and in less time. The plot below show a comparison of the two systems as the number if data increments increases. Each system found the same best patterns, but I-Subdue processed the data in less time.

Figure 4 illustrates the comparative run-time performance of I-Subdue and SUBDUE on the same EAGLE data. The x-axis indicates the number of increments that were processed and the respective size in terms of vertices and edges. For example, for the 15-increment run, I-Subdue processes fifteen successive increments with edges spanning from vertices in the current increment back to vertices in previous increments. We then aggregate all fifteen increments and batch process them with SUBDUE for the comparative result. The results demonstrate that I-Subdue not only accurately handles incremental data, but actually improves run time over the original SUBDUE algorithm.

5. Anomaly Detection

Existing graph-based pattern learning seeks to find prevalent patterns. However, detection of anomalous patterns is also of interest, especially in identifying asymmetric threats. As an outgrowth of the concept drift detection work using I-Subdue, we have discovered a new method of identifying anomalous patterns. Our work to develop a systematic process for measuring change in graph-based relational data is derived from the idea of computing a median graph. If a distance measure d(g1,g2) can be efficiently computed, this can be used to determine the median of a set of graphs (in our case, the top n reported substructures from each data increment).

To compute such a distance function in polynomial time, we rely on the error-correcting graph isomorphism (ecgi). An ecgi is a bijective function, f, that maps vertices from graph g1 to vertices in graph g2 such that the graph edit distance, or total cost of the operations needed to make the graphs isomorphic, is minimized. We use a genetic algorithm to approximate a median graph that minimizes the total distance to the set of represented graphs.

In domains with continuous data streams, one of the major goals is to guide an analyst to salient patterns. In the counter-terrorism example this is particularly essential given the voluminous amount of data that analysts receive each day. It is unrealistic to assume that one can simple identify interesting patterns a priori and then alert the analyst when they are found in the data.

Outlier detection is a potentially useful tool to deal with this problem. Consider the example depicted in Figure 5, with five input graphs and the median graph. The input graphs are the best patterns reported by I-Subdue for five successive data increments. The graphs represent the star and chain communication patterns, which are common to the counter-terrorism domain.

Upon examination of the patterns, the input graph g5 is clearly an outlier. To detect outliers automatically, we can compute a pairwise graph distance measurement between the median graph and each of the input graphs. For the graphs in Figure 5, the distances are as follows: $d(\bar{g}, g_1) = 1; d(\bar{g}, g_2) = 1; d(\bar{g}, g_3) = 2; d(\bar{g}, g_4) = 1; d(\bar{g}, g_5) = 7$

Clearly graph g5 stands out in its distance from the median. We can employ statistical methods to specify the outlier threshold. The resulting anomalies can be reported to the user for further investigation.

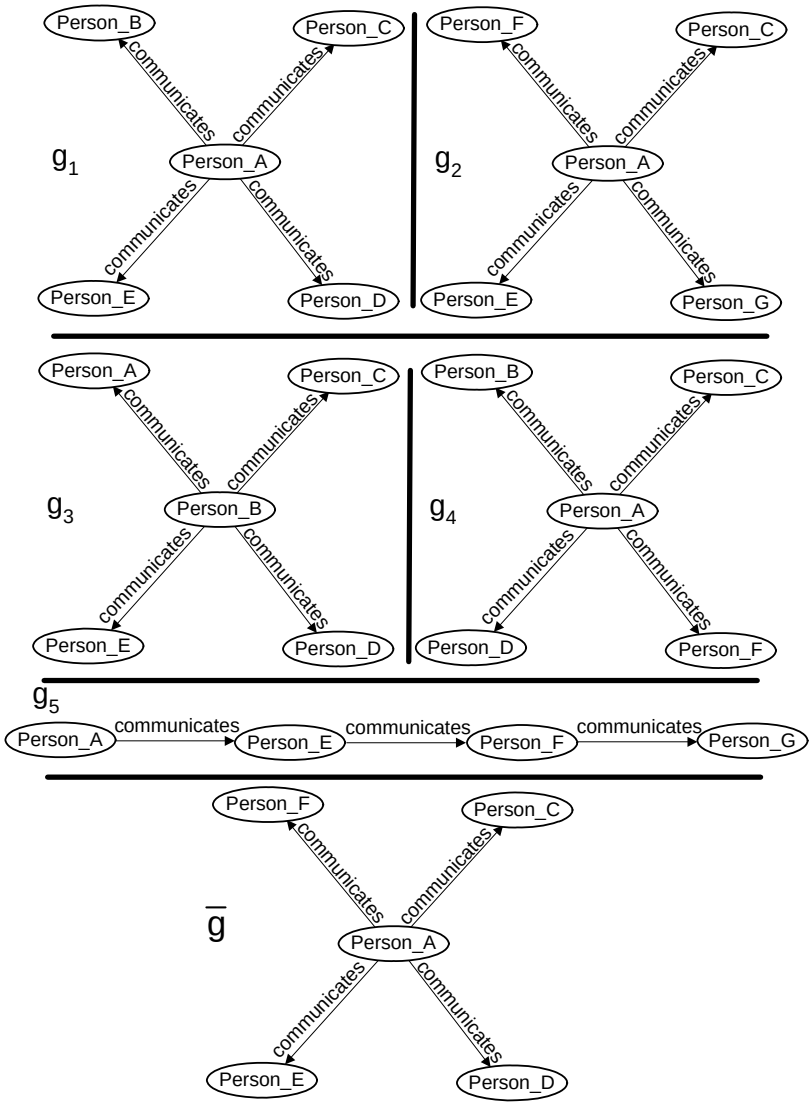

Figure 5. Input graphs g1 through g5, with the computed median graph.

6. Learning from Supervised Graphs

Instead of isolating well-defined, disconnected examples of cases and groups, a supervised graph keeps all the data in one large graph and annotates each graph component with its degree of participation in the various categories of interest (e.g., membership in threat group). Figure 6 depicts the scenario in which threat and non-threat examples all exist in the same graph and possibly overlap. As part of this project, we developed, implemented and evaluated new techniques for learning patterns in supervised graphs.

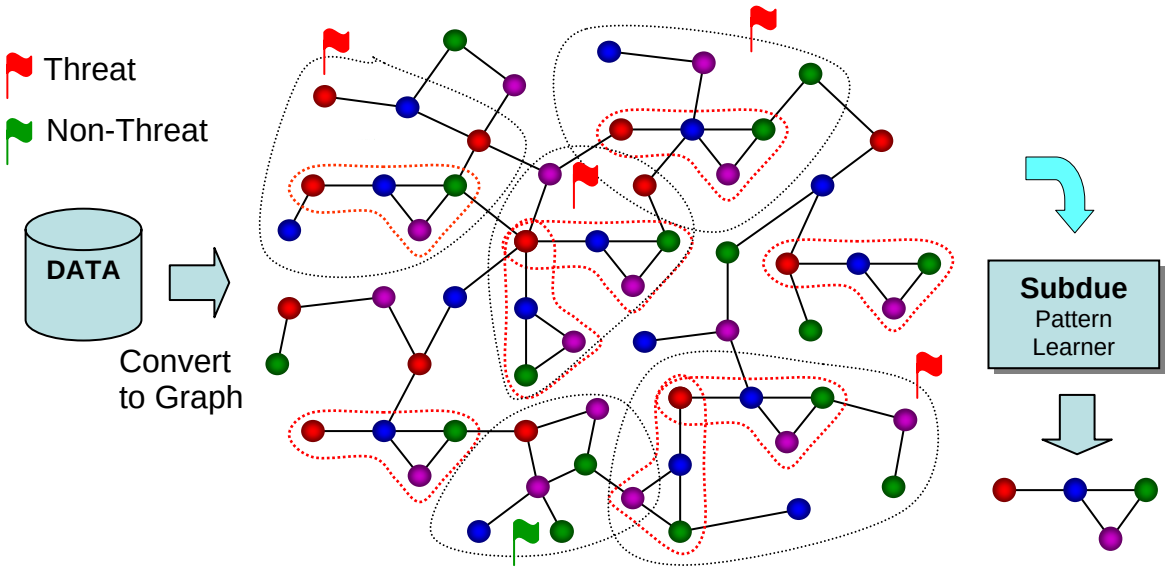

Figure 6. Learning from supervised graphs.

As an outcome of this project, we implemented a new version of SUBDUE, called Subdue-EC for Embedded Classification, that learns to classify examples embedded in one large graph by building a set of substructures that can be used to classify designated areas of a new graph. To determine the class membership of each of the areas of interest in a new graph, one searches for instances of the substructures in the new graph, and, if an instance contains one or more appropriately labeled vertices, then that area is a member of the class that the substructure "classifies". We call these substructures classifying substructures, and we note that the order in which they are discovered by Subdue-EC is the order in which they should be applied to the new graph; therefore, we are learning an ordered sequence of classifying substructures.

We evaluated the completed implementation on data from the NASA Sea Surface Temperature (SST) dataset as well as the EAGLE counterterrorism data using the graph representation described earlier. The NASA data uses a vertex to represent each grid location, with connecting vertices giving the hemisphere of the location and current temperature. The vertex is connected by a directed edge to its western and northern neighbors, and multiple graphs are created for each month. Our goal is to classify location vertices by whether they will increase, decrease, or stay at the same temperature the next month. Using 6,480 grid locations, Subdue-EC classified the data points at 85.21% accuracy using ten-fold cross-validation. In addition, using training data from one year to predict monthly differences the next year resulted in 81.98% classification accuracy.

For the EAGLE data, we created a graph in which vertices represent member agents from Threat and non-Threat groups. Anyone with whom these agents communicate is also added to the graph and connected to the agent with an "association" edge. Communication events between associates are similarly represented with "association" edges. Finally, individuals may be described using attribute and capability vertices.

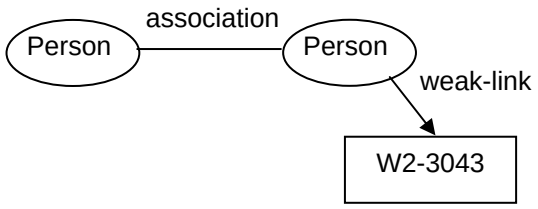

Figure 7. A discovered substructure showing an association between two individuals, each with certain capabilities. The individual on the left is a known threat.

Our experiments were conducted on a large graph consisting of 435,429 vertices and 763,504 edges representing 61,105 people as well as a smaller graph consisting of 217,901 vertices and 314,793 edges representing 30,715 people. Class vertices labeled THREAT were attached to members of known threat groups, and NON-THREAT vertices were attached to members of non-threat groups.

Our goal of the experiments was to see how well Subdue-EC could classify threat and non-threat individuals, given training examples embedded in a single connected graph. In the original graphs there is a large predominance of non-threat individuals (59,373, in contrast to the 1,732 threat individuals). To provide a stronger sample to the learning algorithm, we randomly sampled non-threat individuals to create a training set size equal to that of threat individuals.

For the individuals that included one or more of the classifying substructures, Subdue-EC's classification accuracy was 71.98%. However, the computational limitations of the discovery algorithm prevented further substructures from being discovered in a reasonable amount of time, so 2,304 individuals remained unclassified. The greatest number of misclassifications were false positives (classified as threats when the true classification is non-threat), which is a preferred type of mistake for this problem.

Of the substructures that were discovered, many consisted of an individual exhibiting a particular capability. However, a few of the substructures, such as the one shown in Figure 7, highlight an association between two individuals in addition to attributes and capabilities of the individuals.

The fact that Subdue-EC discovered useful substructures that highlight relationships between the individuals to be classified highlights the strength of Subdue-EC. If the individuals had been separated into disjoint examples, this relationship could not have been found. If we tried to extract individuals with a large enough neighborhood of information around them to find these discoveries, several difficulties would arise. First, how much information do we retain? The user cannot always know a priori how much of a neighborhood must be extracted in order to retain all potentially useful information. Second, when the neighborhood of information is extracted, it is in essence reproduced for each example object that requires the information. This results in substantial cost increase both in memory and in processing time for the discovery algorithm.

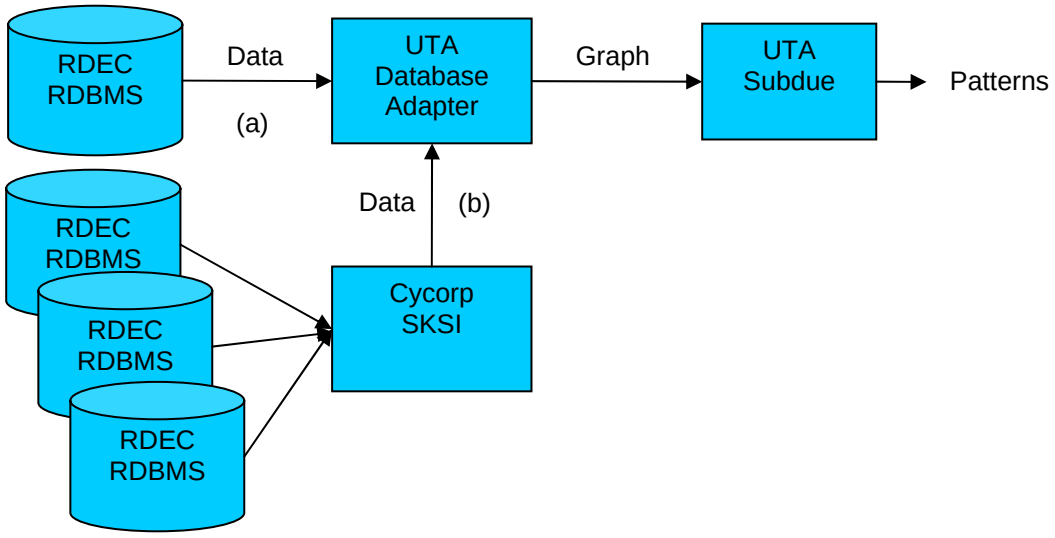

Figure 8. Alternative approaches to RDEC RDBMS-Subdue integration.

7. Transition Subdue to RDEC

A major objective of this project was to transition the SUBDUE pattern learning technology developed by UTA to RDEC. The transition includes a database adapter capable of converting RDEC data (assuming the Y3 EDB schema) into graphical examples of threat and non-threat individuals, groups and/or exploitation cases. These examples are input to SUBDUE, which learns patterns that can discriminate threat examples from non-threat examples. These patterns can be converted back into a form useable by RDEC. This process is illustrated in Figure 8.

We also pursued a second RDEC transition opportunity in collaboration with Cycorp and their Structured Knowledge Source Integration (SKSI) technology. The SKSI integrates multiple databases under one common interface (Cyc). Instead of our SUBDUE database adapter being fitted to each RDEC database, we interfaced the adapter to SKSI in order to access multiple databases as one.

To accomplish this goal, we designed a tool to convert Y2.5 EDBs into SUBDUE format. The EDB Y2.5 Data Generator automatically generates Subdue-formatted input files from a MySQL Y2.5-schema database. Data can be generated in the form of individual graphs for each instance of a particular entity type (e.g., ThreatGroup or NonThreatGroup in EAGLE databases). Once graphs have been generated, Subdue can be run on all the graphs to find common patterns, or can be run in supervised learning mode, where the graphs from a set (e.g., ThreatGroup) are designated as positive examples, and the graphs from another set (e.g., NonThreatGroup) are designated as negative examples. In this mode SUBDUE would find structural patterns that can distinguish the positive examples from the negative examples.

The software is written in Java and has been developed and tested on a Linux system, but should run on any standard distribution. The software is controlled by a number of parameters. The entityType parameter can be set to any entity type (using the CYCCOLLECTION format) used in the database. The generator will generate a graph for each entity of this type. This is done by traversing the links from this entity to other entities in the database. The depthLimit controls how many links away from the initial entity the generator traverses to expand the graph. A value of 0 implies no depth limit. Note that values as small as 3 or 4 may cause each graph to encompass the entire database.

In addition, as requested, we have developed a tool to convert CYC knowledge into a SUBDUE graph format. Cyc knowledge can be queried by a user and the results are returned to the web browser. Since the goal of this project is to take the results and create a SUBDUE input graph file, we created a Perl program that "dumps" the results to a file. A graph is then created expressing the knowledge returned from the Cyc query.

The implementation of this tool was successfully tested on sample Cyc databases. Additional enhancements, such as a graphical interface, could be considered for the future.

8. Integration of SUBDUE with Relational Database Management Systems

Since most EAGLE-related data are stored in RDBMSs, a tighter integration of SUBDUE with RDBMS technologies allows improved scalability by reducing the computational and cognitive overhead of converting the database into the graph representation required by SUBDUE. We investigated a more incremental method whereby the database adapter requests data from the database only when prompted by SUBDUE during the pattern learning process. Therefore, data that SUBDUE deems uninteresting will not need to be converted to graph form. Once this capability was implemented, we then investigated an even tighter integration in which the data remains within the RDBMS and is not converted into graph form. This involves an integration of SUBDUE with the RDBMS at the algorithm level.

We collaborated with Dr. Sharma Chakravarthy, a database professor here at UTA, to achieve this project objective. The approach developed in this task infers the entity-relationship model for the database using the table descriptions along with primary and foreign key constraints and generates the instances of the graphs from the populated instances of the relations. As part of this task the following issues were addressed: i) representation of individual relations as a graph (template), ii) representation of multiple relations as a graph (based on foreign key constraints), iii) alternative representations for ii), and iv) sequences in which to process the relations to generate graph instances.

Graph representation is important as the size of the input for mining may increase significantly (e.g., can double) based on the representation chosen. This will have a critical impact on the processing time and main memory requirements. Also, choice of a representation that captures the semantics of the transactions is important. Otherwise, it may be difficult to interpret the results of mining. Finally, the sequence in

which relations are processed for generating the graph determines the amount of memory used for generating the graph and the time complexity of the algorithm.

The XRDB2Graph algorithm developed for this task is an efficient algorithm that converts any relational database into a graph form that can be used by SUBDUE. XRDB2Graph is loosely coupled with the database platform (using the JDBC bridge) in order to accommodate any relational database (MySql, Oracle, MS SQLServer, etc.). We have used only SQL92 so that compatibility and portability is guaranteed across widely used databases. XRDB2Graph also minimizes the maximum memory requirement while transforming relational database to Graph domain; hence larger databases can be transformed into Graphs without need for large amounts of main memory. Even with SQL92, as there are some differences between create tables of different databases, the system is modularized. Only a small portion of the system needs to be customized for a particular database.

9. Conclusions and Future Work

All of the specified goals for this project were accomplished. Moreover, we demonstrated the ability of a graph-based relational learner to discover patterns of importance in terrorist data. Using the SUBDUE algorithms, these patterns can represent interesting behaviors needing additional analysis, or may be used to categorize data as potential threats or non-threats.

Work on the EAGLE project resulted in new enhancements to the SUBDUE system that are a benefit to the AFRL program as well as to the scientific community. Through our integration efforts we greatly enhanced the scalability of SUBDUE. Incremental processing of data, conversion of Cyc knowledge to SUBDUE graphs, and integration of SUBDUE with RDBMSs greatly broadens the applications of the algorithms to a wide variety of data types and sizes. In addition, we have increased the representational power of SUBDUE by allowing it to learn concepts from a single supervised graph and handle streaming graph data. We also demonstrate that a graph-based relational learner can be used for additional tasks such as detecting anomalies, all of which is useful when analyzing security data.

Work on the EAGLE project has also highlighted a number of areas for additional work. We intend to continue investigating methods of increasing the scalability of SUBDUE. We will also use the algorithms to detect trends in time-varying data and increasing the learning power of SUBDUE using graph grammars. We believe that this work will continue to provide valuable tools to the intelligence community.

10.Publications from this Work

Publications resulting from this project are listed in chronological order, beginning with 2003.

- I. Jonyer, L. Holder and D. Cook, “MDL-based Context-Free Graph Grammar Induction,” *Proceedings of the Sixteenth International Conference of the Florida AI Research Society (FLAIRS)*, May 2003.

- R. Mehta, D. Cook and L. Holder, “Identifying Inhabitants of an Intelligent Environment Using a Graph-Based Data Mining System,'' *Proceedings of the Sixteenth International Conference of the Florida AI Research Society (FLAIRS)*, May 2003.
- L. Holder and D. Cook, “Graph-based Relational Data Mining: Current and Future Directions,” *SIGKDD Explorations Special Issue on Multi-Relational Data Mining*, Volume 5, Issue 1, 2003.
- I. Jonyer, L. Holder and D. Cook, “MDL-Based Context-Free Graph Grammar Induction and Applications,” *International Journal on Artificial Intelligence Tools*, 13(1):65-80, March 2004.
- J. Coble, D. Cook, L. Holder and R. Rathi, “Structure Discovery from Sequential Data,” *Proceedings of the Seventeenth International Conference of the Florida AI Research Society (FLAIRS)*, May 2004.
- M. Mukherjee and L. Holder, “Graph-based Data Mining for Social Network Analysis,” *Proceedings of the ACM KDD Workshop on Link Analysis and Group Detection*, August 2004.
- I. Jonyer, L. Holder, and D. Cook, “Attribute-Value Selection Based on the Minimum Description Length”, *Proceedings of the International Conference on Artificial Intelligence*, 2004.
- J. Kukluk, L. Holder, and D. Cook, “Algorithm and Experiments in Testing Planar Graphs for Isomorphism”, *Journal of Graph Algorithms and Applications*, Volume 8, Number 3, 2004.
- J. Coble, R. Rathi, D. Cook and L. Holder, “Iterative Structure Discovery in Graph-Based Data,” *International Journal on Artificial Intelligence Tools*, 14(1-2), 2005.
- A. Rakhshan, L. Holder and D. Cook, “Structural Web Search Engine”, *International Journal on Artificial Intelligence Tools*, 13(1):27-44, 2005.
- L. Holder, D. Cook, J. Coble and M. Mukherjee, “Graph-based Relational Learning with Application to Security,” in *Fundamenta Informaticae Special Issue on Mining Graphs, Trees and Sequences*, 2005.
- L. Holder and D. Cook, “Graph-based Data Mining.” In J. Wang (ed.) *Encyclopedia of Data Warehousing and Mining*, Idea Group Publishing, 2005.
- L. Holder, D. Cook, J. Coble and M. Mukherjee, “Graph-based Relational Learning with Application to Security,” in L. De Raedt, T. Washio and J. Kok (eds.) *Mining Graphs, Trees and Sequences*, IOS Press, 2005.
- J. Potts, D. Cook, L. Holder and J. Coble, “Learning Concepts from Intelligence Data Embedded in a Supervised Graph,” *Proceedings of the International Conference on Intelligence Analysis*, April 2005.
- J. Coble and D. Cook, “Structure Discovery in Sequentially Connected Data”, *Proceedings of the Florida Artificial Intelligence Research Symposium*, May 2005 (Best Paper Award).
- R. Rathi and D. Cook, “A Serial Partitioning Approach to Scaling Graph-Based Knowledge Discovery”, *Proceedings of the Florida Artificial Intelligence Research Symposium*, May 2005.

- J. Potts, D. Cook, and L. Holder, “Learning from Examples in a Single Graph”, *Proceedings of the Florida Artificial Intelligence Research Symposium*, May 2005.
- D. Cook, L. Holder, J. Coble and J. Potts, “Graph-based Mining of Complex Data," in S. Bandyopadhyay, U. Maulik, L. Holder and D. Cook (Editors), *Advanced Methods for Knowledge Discovery from Complex Data*, Springer, September 2005.
- N. Ketkar, L. Holder and D. Cook, “Qualitative Comparison of Graph-based and Logic-based Multi-Relational Data Mining: A Case Study," *Proceedings of the ACM KDD Workshop on Multi-Relational Data Mining*, August 2005.
- N. Ketkar, L. Holder, D. Cook, R. Shah and J. Coble, “Subdue: Compression-based Frequent Pattern Discovery in Graph Data," *Proceedings of the ACM KDD Workshop on Open-Source Data Mining*, August 2005.
- N. Ketkar, L. Holder and D. Cook, “Comparison of Graph-based and Logic-based MRDM,” *ACM SIGKDD Explorations Special Issue on Link Mining*, Volume 7, Issue 2, December 2005.
- C. D. Corley, D. J. Cook, L. B. Holder, and K. P. Singh, Graph-based data mining in epidemia and terrorism data, to appear in *Proceedings of the Conference on Quantitative Methods and Statistical Applications in Defense and National Security*, 2006.
- J. Potts, D. J. Cook, and L. B. Holder, Learning from Supervised Graphs, to appear in *Applied Graph Theory* (M. Last, A. Kandel, and H. Bunke, editors), 2006.
- Mining Graph Data, (D. J. Cook and L. B. Holder, editors), John Wiley and Sons, to appear in September 2006.
- J. Coble, D. J. Cook, and L. B. Holder, Structure Discovery in Sequentially-Connected Data Streams, to appear in *International Journal on Artificial Intelligence Tools*, 2006.

Other References

1. U.S. Congress, “Joint Inquiry into Intelligence Community Activities Before and After the Terrorist Attacks of September 11, 2001,” December 2002.
2. D. J. Cook and L. B. Holder, “Graph-based data mining,” *IEEE Intelligent Systems*, 15(2):32-41, 2000.
3. J. Rissanen. Stochastic Complexity in Statistical Inquiry. World Scientific Publishing Company, 1989.