Structure Discovery from Sequential Data

Jeffrey Coble, Diane J. Cook, Lawrence B. Holder and Runu Rathi

Department of Computer Science and Engineering
The University of Texas at Arlington
Box 19015, Arlington, TX 76019
{coble,cook,holder,rathi}@cse.uta.edu

Abstract

In this paper we describe I-Subdue, an extension to the Subdue graph-based data mining system. I-Subdue operates over sequentially received relational data to incrementally discover the most representative substructures. The ability to incrementally refine discoveries from serially acquired data is important for many applications, particularly as computer systems become more integrated into human lives as interactive assistants. This paper describes initial work to overcome the challenge of locally optimal substructures overshadowing those that are globally optimal. We conclude by providing an overview of additional challenges for sequential structure discovery.

Introduction

While much of data mining research is focused on algorithms that can identify sets of attributes that discriminate particular data entities, such as shopping or banking trends for a particular demographic group, our work is focused on data mining techniques to discover relationships between entities. Our work is particularly applicable to problems where the data is event driven, such as the types of intelligence analysis performed by counter-terrorism organizations or the agent-based MavHome smart home project (Cook et al. 2003). These types of problems require discovery of relational patterns between the events in the environment so that these patterns can be exploited for the purposes of prediction and action.

In this paper we present I-Subdue, which represents our work to extend structure discovery to serially acquired data. Systems such as MavHome require agents to operate in an environment over long periods of time, which means that the data is received sequentially and discriminating structures must be evolved by processing only the newest evidence. The same can be said for analytical tasks where data streams in over time. Domain examples in this paper are drawn from our work on the Defense Advanced Research Project Agency's Evidence Extraction and Link Discovery (EELD) program, which is a multi-faceted project designed to provide information-oriented tools to support counter-terrorism intelligence analysis.

Processing only new data with the intent of refining global knowledge is challenging from a variety of perspectives. The work presented in this paper describes our efforts to address the situation where discovery from a new data increment yields a locally optimal structure that is inconsistent with the globally optimal structure. We present a method by which metrics can be collected from the local discovery process, which can then be used to evaluate discovered structures for their global value.

We conclude by introducing additional challenges, such as identifying temporally displaced relationships and shifting concepts, both of which are complicated by the sequential discovery process.

Structure Discovery

The purpose of I-Subdue is to sequentially discover structural patterns in data received serially. The work we describe in this paper is built upon Subdue (Holder et al. 2002), which is a graph-based data mining system designed to discover common structures from relational data. Subdue represents data in graph form and can

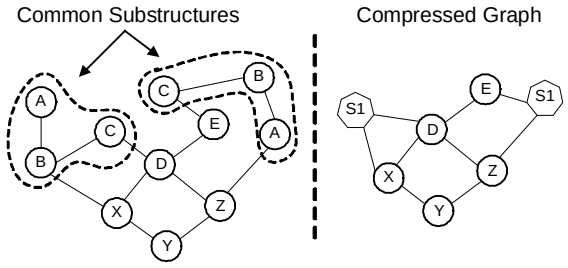

Figure 1. Subdue discovers common substructures within relational data by evaluating their ability to compress the graph.

support either directed or undirected edges. Subdue operates by evaluating potential substructures for their ability to compress the entire graph, as illustrated in Figure 1. In each iteration, Subdue finds the best substructure and compresses the graph by replacing the substructure in the graph with a placeholder vertex. Repeated iterations will discover additional substructures, potentially those that are hierarchical, containing previously compressed substructures.

Subdue uses the Minimum Description Length Principle (Rissanen 1989) as the metric by which graph compression is evaluated. Subdue is also capable of using an inexact graph match parameter to evaluate substructure matches,

so that slight deviations between two patterns can be considered as the same pattern.

Incremental Subdue

For our work on I-Subdue, we assume that data is streaming into a repository, which we then process

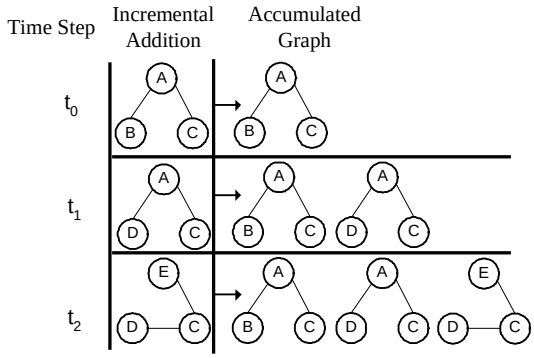

Figure 2. Data received incrementally can be viewed as a unique extension to the accumulated graph.

incrementally in blocks. We view each new data increment as a distinct graph structure. Figure 2 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.

It is easy to see how the concept depicted in Figure 1 can be applied to real problems. For instance, a software agent

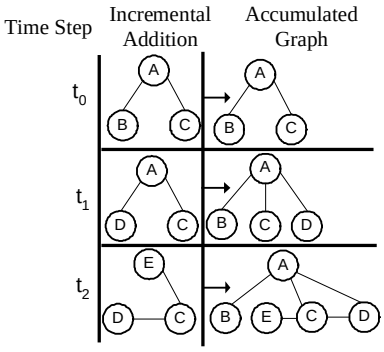

Figure 3. Data received incrementally can be viewed as augmentation of the accumulated graph, with duplicate nodes serving as anchor points for new nodes and vertices.

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 collect 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. This incremental mechanism would also be suited for applications such as long-term behavioral monitoring of the inhabitants of a smart home, with the intent of continuously evaluating and refining observed patterns.

Figure 3 represents another option for dealing with serially acquired data in which new increments are a mechanism for introducing new vertices and edges. Envisioning circumstances under which such a scheme would be desirable is more difficult, but we may want to model a situation in which new associations are made between variables, without necessarily weighting the existing variables more heavily by repeating their vertices.

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 easily 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, which would be found assuming the data could be evaluated as one aggregate block.

Figure 4 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 only be locally best. However, as illustrated in Figure 5, applying Subdue to the aggregate data will yield a different best substructure, which in fact is globally best. Although our simple example could easily be aggregated at each time step, realistically large data sets would be too unwieldy to do so.

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 from which structures are discovered. One problem that is almost always a concern is how to reevaluate the accumulated data at each time step in light of newly added data. There is generally a tradeoff between the amount of data that can be stored and reevaluated and the quality of the result. A summarization technique is usually 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.

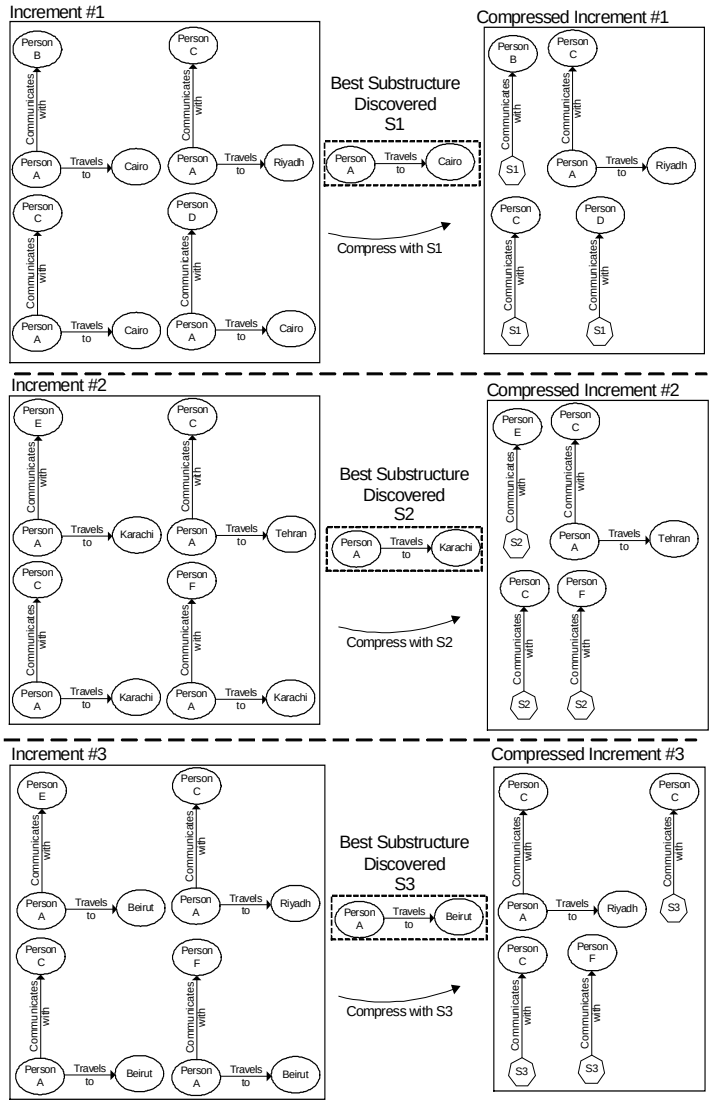

Figure 4. Three data increments received serially and processed individually by Subdue. The best substructure is shown for each local increment.

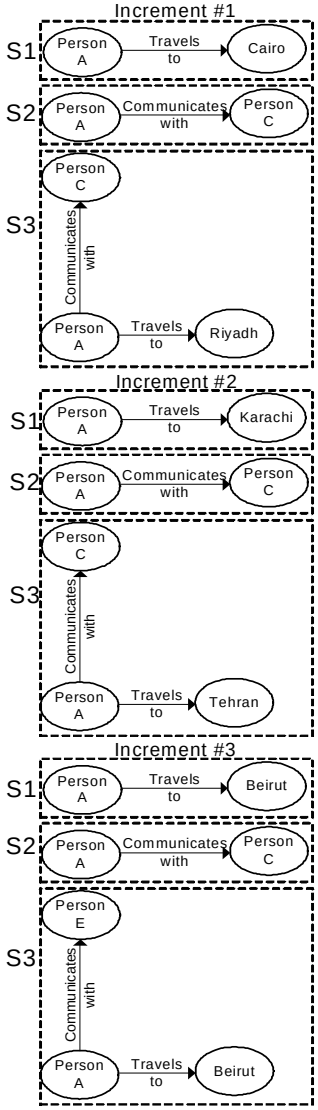

Figure 6. The top n=3 substructures found independently in each iteration.

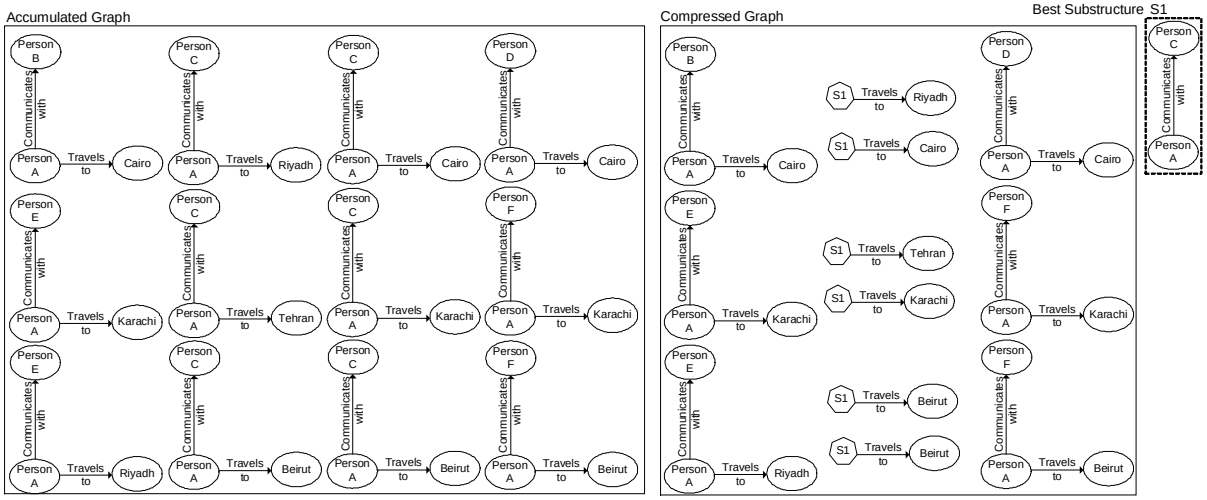

Figure 5. Result from applying Subdue to the three aggregated data increments in one batch.

Summarization Metrics. Our goal for this research is to develop a summarization metric that can be maintained from each incremental application of Subdue that will allow us to derive the globally best substructure without reapplying Subdue to the accumulated data.

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 dataset, where *n* is configurable by the user. The default value for *n* is 3, but any number of ranked substructures can be maintained, limited only by constraints on the beam search that Subdue uses to prune its search space.

Second, we use the value metric Subdue maintains for each substructure. Subdue measures graph compression with the minimum description length principle as illustrated in Equation 1, where *DL(S)* is the description length of the substructure being evaluated, *DL(G|S)* is the description length of the graph as compressed by the substructure, and *DL(G)* is the description length of the original graph. The better our substructure performs, the smaller the compression ratio will be. The description length of a graph (or substructure) consists of the number of bits needed to encode the vertex labels, the adjacency matrix, the number of edges between vertices, and the edge labels. C.f. (Cook and Holder 1994) for a full discussion of the MDL computation used by Subdue to encode graphs.

$$Compression = \frac{DL(S) + DL(G \mid S)}{DL(G)} \qquad \text{Eq. 1}$$

Subdue's evaluation algorithm ranks the best substructure by measuring the inverse of the compression value in Equation 1. Favoring larger values serves to pick a substructure that minimizes *DL(S)* + *DL(G|S)*, which means we have found the most descriptive substructure.

For I-Subdue, we must use a modified version of the compression metric to find the globally best substructure, illustrated in Equation 2.

$$Compress_m(S_i) = \frac{DL(S_i) + \sum_{j=1}^{m} DL(G_j \mid S_i)}{\sum_{j=1}^{m} DL(G_j)} \qquad \text{Eq. 2}$$

With Equation 2 we calculate the compression achieved by a particular substructure, S_i, up through and including the current data increment *m*. The $DL(S_i)$ term is the description length of the substructure, S_i, under consideration. The term

$$\sum_{j=1}^{m} DL(G_j \mid S_i)$$

represents the description length of the accumulated graph after it is compressed by the substructure S_i. Finally, the term

$$\sum_{j=1}^{m} DL(G_j)$$

represents the full description length of the accumulated graph.

$$arg\ max(i)\left[\frac{\sum_{j=1}^{m} DL(G_j)}{DL(S_i) + \sum_{j=1}^{m} DL(G_j \mid S_i)}\right] \qquad \text{Eq. 3}$$

At any point we can then reevaluate the substructures using Equation 3 (inverse of Equation 2), choosing the one with the highest value as globally best.

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 4. The top *n=3* substructures from each iteration are shown in Figure 6. Table 1 lists the values returned by Subdue for the local top *n* substructures discovered in each iteration. The second best substructures in iterations 2 and 3 (S_{22}, S_{32}) are the same as the second best substructure in iteration 1 (S_{12}), which is why the column corresponding to S_{12} has a value for each iteration. The values in Table 1 are the result of the compression evaluation metric from Equation 1. The locally best substructures illustrated in Figure 4 have the highest values, demarcated by the highlighted cells in Table 1.

Table 2 depicts our application of I-Subdue to the increments from Figure 4. After each increment is received, we apply Equation 3 to select the globally best substructure. The values in Table 2 are the inverse of the compression metric from Equation 2. As an example, the calculation of the compression metric for substructure S_{12} after iteration 3 would be:

$$\frac{DL(S_{12}) + DL(G_1 \mid S_{12}) + DL(G_2 \mid S_{12}) + DL(G_3 \mid S_{12})}{DL(G_1) + DL(G_2) + DL(G_3)}$$

Consequently the value of S_{12} would be:

$$\frac{117 + 117 + 116}{15 + 96.63 + 96.63 + 96.74} = 1.1474$$

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. Figure 7 illustrates the basic algorithm, where Subdue is invoked to discover the candidate substructures and the byproduct evaluation metrics are

Table 1. Substructure values computed independently for each iteration. Highlighted cells indicate maximum values in each iteration.

Iteration #	New Substructures from Iteration #1			New Substructures from Iteration #2		New Substructures from Iteration #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	0.98151		
3		1.03804				1.15126	0.96602

Table 2. Using I-Subdue to calculate the global value of each substructure. The description length of each graph iteration (G_j) and of each substructure (S_i) are shown. Highlighted cells indicate the global best substructure at each iteration.

Global Best Calculation After Iteration #	New Substructures from Iteration #1			New Substructures from Iteration #2		New Substructures from Iteration #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	25.755	15	25.7549	15	26.5098	

*measured in bits

```
//Call I-Subdue on the new data increment Gj
I-Subdue(Gj)
   //Subdue returns description length values and top n substructures for current data increment,
   //which are stored for global calculations
   CandidateSubstructures[], SubstructureSizes[], CompressedGraphSizes[], size_Gj ⇐ Subdue(Gj)
   total_graph_size = total_graph_size + size_Gj

/*************************************************************************/
Get_Global_Best(total_graph_size,CandidateSubstructures[], SubstructureSizes[], CompressedGraphSizes[])
   best_value = 0
   global_best_substructure = nil
   for(i=1 to sizeof(CandidateSubstructures))
            size_si = CandidateSubstructureSizes[i]
                     compressed_graph_size = 0
                     for(j=1 to num_data_increments)
                              compressed_graph_size = compressed_graph_size +
                                       CompressedGraphSizes[i][j])          //DL(Gj|Si)
                     value_si = graph_size/(size_si + compressed_graph_size)
                     if value_si > best_value
                              best_value = value_si
                              global_best_substructure = CandidateSubstructures[i]
   return global_best_substructure
```

Figure 7. Application of I-Subdue to store metrics returned from running Subdue over a single data increment, then calculating the global best substructure using the collected metrics.

collected and used to calculate the globally best substructures after each new data increment is processed.

In circumstances where a specific substructure is not present in a particular data increment, such as S_{31} in iteration 2, then

$$DL(G_2 \mid S_{31}) = DL(G_2)$$

and the substructure's value would be calculated as follows:

$$\frac{117+117+116}{15+117+117+85.76} = 1.0455$$

Conclusions

In this paper we have presented our introductory work on I-Subdue, an extension to the Subdue structure discovery

system. We have presented a set of metrics and associated rules for their application that allows our system to discover globally best substructures from serially processed data increments. This work allows us to overcome a problem inherent to the sequential discovery process, namely that of overlooking globally best substructures because of discoveries that are locally best to the specific data increment being mined.

Future Work

Our preliminary analysis indicates that there would be some benefit to formulating I-Subdue so that the algorithm uses what it has learned from previous data iterations to direct the discovery process in future iterations. By using the ranking of globally best substructures, I-Subdue may be able to prune substructures from the search space that are clearly only good in the local context. However, care must be taken not to prematurely judge a substructure as unimportant. Doing so may inappropriately bias the global discovery process.

Sequential Relationships. Many applications areas in which we are applying our research are event driven. For example the smart home application generates data about events created by the human occupants. The counter-terrorism application domain used throughout this paper is also a collection of events. The goal of structure discovery is then to derive representative patterns from sets of these events. This is complicated in the sequential learning process, since event correlations may transcend multiple data iterations. For example, in the smart home one might assume that the temporal adjacency of two events is significant enough to infer a relationship. This time window is certainly subjective, however, one might also choose to assume a relationship between two events that fall outside of the time window but happen to have some conceptual relationship; the use of a washer and dryer for instance. One may start the wash on one day and dry it on another. Although the events may not be temporally related, they are conceptually related. We will address sequential relationships in our future work.

Shifting Concepts. In the traditional machine learning problem (Mitchell 1997, Vapnik, 1995), it is generally stated that some function F(x) is generating an attribute vector x, based on a fixed relationship, whether probabilistic or deterministic. The attribute vector x represents the observable features of the problem space. This definition extends intuitively to data mining. However, in sequential discovery problems, the applications are such that the underlying relationships between system variables often change. Referring back to our smart home application, changes in lifestyle, seasons, or anomalous weather events may perturb the typical system behavior. There are approaches to machine learning in the presence of shifting concepts, such as the sliding window approach presented in (Widmer and Kubat 1996), but these are often naïve in the sense that they disregard valuable information learned outside of the data window. Our future work will focus on developing methods for structure discovery when the underlying system is undergoing change.

Acknowledgments

This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and managed 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 DARPA, AFRL, or the United States Government.

References

1. Cook, D. and Holder, L. 1994. Substructure Discovery Using Minimum Description Length and Background Knowledge. In *Journal of Artificial Intelligence Research*, Volume 1, pages 231-255.

2. Cook, D., Youngblood, M., Heierman, E., Gopalratnam, K., Rao, S, Litvin, A. and Khawaja, F. 2003. MavHome: An Agent-Based Smart Home, *Proceedings of the* First IEEE International *Conference on Pervasive Computing.* Fort Worth, Texas: IEEE Computer Society.

3. Holder, L., Cook, D., Gonzalez, J., and Jonyer, I. 2002. In *Structural Pattern Recognition in Graphs. Pattern Recognition and String Matching*, Chen, D. and Cheng, X. eds. Kluwer Academic Publishers.

4. Mitchell, Tom. 1997 *Machine Learning*. McGraw Hill.

5. Rissanen, J. 1989. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Company, 1989.

6. Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer, New York, NY, USA.

7. Widmer, G. and Kubat, M. 1996. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*, 23, 69-101