Enhancing Structure Discovery for Data Mining in Graphical Databases Using Evolutionary Programming

Sanghamitra Bandyopadhyay, Ujjwal Maulik, Diane J. Cook, Lawrence B. Holder, and Yousuf Ajmerwala
Department of Computer Science Engineering
University of Texas at Arlington
416 Yates Street, 300 Nedderman Hall
Arlington, TX 76019
Email: {bandyopa,maulik,cook,holder,ajmerwal}@cse.uta.edu

Keywords: Data Mining, Structural Databases, Substructure Discovery, Evolutionary Programming.

Abstract

The purpose of this paper is to develop an evolutionary programming based system that performs data mining on databases represented as graphs. The importance of such an endeavor can hardly be overemphasized, given that much of the data collected nowadays is structural in nature, or is composed of parts and relations between the parts, which can be naturally represented as graphs. The searching capability of evolutionary programming is utilized for discovering concepts or substructures that are often repeating in such structural data. The superiority of the proposed technique over the previously developed SUBDUE system (Cook and Holder 2000), which uses a computationally constrained beam search in the space of substructures, is demonstrated for a number of data sets in the Web domain.

Introduction

In recent times there has been a surge of activity aimed at discovering interesting patterns, concepts and structural repetitions in large amounts of data that is routinely collected (Cheeseman and Stutz 1996; Fisher 1987; Quinlan 1993). Although much of the data collected has an explicit or implicit structural component (e.g., spatial or temporal), few discovery systems are designed to handle this type of data (Fayyad, Piatetsky-Shapiro, and Smyth 1996). One method for discovering knowledge in structural data is the identification of common substructures or concepts that describe interesting and repetitive substructures within structural data. Once discovered, the substructure concept can be used to simplify the data by replacing instances of the substructure with a pointer to the newly discovered concept. The discovered substructure concepts allow abstraction over detailed structure in the original data and provide new, relevant attributes for interpreting the data.

A variety of approaches to discovery using structural data have been proposed (e.g., (Conklin 1995; Thompson and Langley 1991; Cook and Holder 2000)). Many of these approaches use a knowledge base of concepts to classify the structural data. These systems perform concept learning over examples and categorization of observed data. The SUBDUE system developed in (Cook and Holder 2000) discovers interesting substructures in structural data by using a beam search in order to constrain the search space of the substructures. Because any such search algorithm is limited by the choice of the beam width, it may often end up providing sub-optimal results.

In order to overcome this limitation, in this article we propose an evolutionary programming based substructure discovery algorithm. Evolutionary programming (EP) was introduced by Fogel et al. (Fogel 1962; Fogel, Owens, and Walsh 1966; Porto *et al.* 1998) as a means to develop artificial intelligence. EP makes use of a metaphor of natural evolution, according to which, a problem plays the role of an environment wherein lives a population of individuals, each representing a possible solution to the problem. The degree of adaptation of each individual, i.e., the candidate solution, to its environment is expressed by a measure known as a *fitness function.* The phenotype of each individual is generally encoded in some manner into its genome and known as *chromosome.* Like evolution in nature, EP potentially produces progressively better solutions to the problem by introducing new genetic material into the chromosome using only mutation. The effectiveness of the EP-based substructure discovery algorithm proposed in this article is demonstrated on several data sets generated from web pages. Its performance is compared to that of the previously developed SUBDUE system (Cook and Holder 2000) for these data sets.

Concept Discovery from Structured Data

The aim of the concept discovery algorithm is to detect patterns that are often repeating in structured data, such that the structural concepts in it are represented and the original data is compressed. The searching capability of EP, along with its characteristic of coming out of local optima is used for this purpose. In this section, we first model the substructure discovery problem.

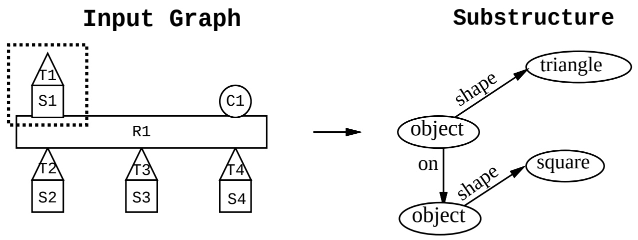

Figure 1: Example substructure in graph form.

This is followed by the description of the EP-based discovery technique.

Problem Formulation

The structured data is represented as a labeled graph. Objects in the data map to vertices or small subgraphs in the graph, and relationships between objects map to directed or undirected edges in the graph. A *substructure* is a connected subgraph within the graphical representation. An *instance* of a substructure in an input graph is a set of vertices and edges from the input graph that match, graph theoretically, to the graphical representation of the substructure. This graphical representation serves as input to the substructure discovery system. Figure 1 shows a geometric example of a database. The graph representation of the discovered substructure is also shown, and one of the four instances of the substructure is highlighted in the picture.

Evolutionary Programming for Substructure Identification

EP belongs to the same category of evolutionary computation as genetic algorithms (GAs) (Goldberg 1989; Michalewicz 1992; Mitchell 1996), the primary differences being (i) EP does not place any constraint on the representation, while GA usually requires the problem solutions to be encoded as strings, and (ii) EP does not use any sexual reproduction (or crossover), while crossover is one of the fundamental operators in GA. In EP only mutation is applied on the parent chromosomes to produce offspring. The offspring are then evaluated in the same way as their parents. Subsequently the next generation is selected from collection of both the parents and the offspring. This process is repeated for a number of generations until some termination criterion is attained. In this regard, EP is quite similar to evolutionary strategies (ES) (Schwefel 1987), although the two approaches evolved independently.

A chromosome in the EP-based discovery scheme contains a pointer to a substructure in the graph. Since performing crossover between two such chromosomes, each pointing to just a single substructure, is not really meaningful, we mainly rely on the mutation and selection operators to lead us to potentially better solutions. This was the main motivation behind using EP (instead of the more commonly used GA) as the underlying search and optimization tool for the task of substructure discovery from graphical data.

The substructure discovery algorithm used by the EP-based technique is shown in figure 2. Here G is the input graph, *Pop-Size* denotes the number of chromosomes in the population and *Limit* is the maximum number of iterations of EP. The first task in the EP-based discovery algorithm is the generation of the initial population which is done randomly. The substructures in the initial population consist of a single edge with two vertices. These substructures are evaluated, and the value is assigned to the fitness of the corresponding chromosomes. Mutation is used to perturb a chromosome to generate a new one, which are then evaluated. Selection is performed on the set of parent and mutated chromosomes to yield the next generation. The cycle of fitness evaluation, mutation and selection continues for a number of generations until a termination criterion is attained. The different steps of the proposed technique are described below.

Population Initialization: All possible substructures having a single edge with two vertices are generated in the following way: For each unique vertex label, a substructure is assembled whose definition is a vertex with that label, and whose instances are all of the vertices in G with that label. Each of these substructures is inserted in ParentList. Each substructure is then in turn removed from the head of ParentList, and each of its instances is extended in all possible ways. This is done by adding a new edge and vertex in G to the instance. The first instance of each unique expansion becomes a definition for a new child substructure, and all of the child instances that were expanded in the same way (i.e., by adding the same new edge with new vertex to the same old vertex) become instances of that child substructure. In addition, child instances that were generated by different expansions, and that match the child substructure definition, graph theoretically, also become instances of the child substructure. The substructure definition and its instances are inserted in the ChildList. For each chromosome in the initial population, its pointer is set to the address of a substructure randomly selected from ChildList.

Fitness Evaluation: The substructure S pointed to by a chromosome is evaluated based on its size, $SizeOf(S)$, and the size of the graph G compressed using S, $SizeOf(G,S)$. The lower the value of $SizeOf(G,S)+SizeOf(S)$, the better is the discovered substructure in terms of its compressability. Note that since our aim is to minimize this quantity, we define the fitness of a chromosome, which needs to be maximized, to be equal to $\frac{1}{SizeOf(S)+SizeOf(G,S)}$.

Mutation: Mutation is similar to the process of generating the substructures for initializing the population. During mutation of a chromosome, the substructure S that it points to is extracted. Each of the instances of S is extended in all possible ways. The

ALGORITHM EP-SUBSTRUCTURE-DISCOVERY(G, Pop-Size, Limit)

ParentList = {}

ChildList = {}

ChromList = NewChromList = {}

BestChromosome = Chromosome = {}

Generations = 0

Create a substructure from each unique vertex label and its single-vertex instances

Insert the resulting substructures in ParentList

while ParentList is not empty **do**

  Parent = RemoveHead(ParentList)

  Extend each instance of Parent in all possible ways

  Group extended instances into Child substructures

  foreach Child **do**

    Insert Child in ChildList

for i = 1 to Pop-Size **do**

  Randomly select one substructure from ChildList

  Chromosome = pointer to this substructure

  Evaluate the substructure

  Assign the value to the *fitness* of Chromosome

  ChromList = ChromList + Chromosome

while Generations < Limit **do**

  foreach chromosome in ChromList **do**

    Mutate chromosome and append to ChromList

    Evaluate the mutated substructure

    Assign value to the *fitness* of chromosome

  Store the chromosome with maximum fitness in BestChromosome

  Introduce BestChromosome in NewChromList

  Select (Pop-Size - 1) chromosomes from ChromList and append to NewChromList

  Assign NewChromList to ChromList

  Generations = Generations + 1

return BestChromosome

Figure 2: EP-based discovery algorithm.

first instance of each unique expansion becomes a definition for a new child substructure S', and all of the child instances that were expanded in the same way (i.e., by adding the same new edge or new edge with new vertex to the same old vertex) become instances of that child substructure. In addition, child instances that were generated by different expansions, and that match S' within the matchcost threshold also become instances of S'. The substructure definition S' and its instances are inserted in the ChildList. After all the instances of S have been extended, and the ChildList is complete, a substructure S'' is randomly selected from the ChildList. The chromosome is then made to point to S''. The mutated chromosomes are appended to the parent population, ChromList.

Selection and Elitism: Fitness proportional selection that mimics the Darwinian principle of *survival of the fittest* is adopted for selecting *Pop-Size* chromosomes that are put into NewChromList. Here each chromosome receives a number of copies, which is proportional to its relative fitness in the population, for transmittal to the next generation. Note that since selection is performed stochastically, the best chromosome has a small probability of disappearing from the population. In order to retain the best chromosome seen up to the current generation (BestChromosome), it is deterministically selected and introduced in NewChromList.

Experimental Results

For the purpose of demonstrating the effectiveness of the EP-based technique, we have used ten graphs created from web sites (Gonzalez 2001) which are extracted from the world wide web (WWW) which provides an immense source of information. Relationships between different web sites are established through hyperlinks. These relationships may include a hierarchical relation between a top-level page and a child page containing more detailed information, relationships between the current, previous and next sites in a sequence of pages, or an implicit endorsement of a page which represents an authority on a particular topic. The WWW can be represented as a labeled graph, and then searched using the EP-based data mining system. For this purpose, we transform web data to a labeled graph using a web robot. In this graph, each URL is represented as a vertex labeled "_page_", and edges labeled "_hyperlink_" point from child URLs to parent URLs (the parentURL contains a link to the child site). The graph includes files with extensions .html (.htm), .txt, .exe, .gif, .pdf, .ps, .tar, .gz, .jpg, or .mpg. In the case of text files, the content of the file is filtered to remove punctuation, conjunctions, and other non-disambiguating symbols. The remaining words are added to the graph with an edge labeled "_word_" connecting the page in which the word is found to the word vertex. Using such graphs as inputs, we propose to compare the performance of SUBDUE with the EP-based discovery system in terms of the compression achieved by the substructures discovered by the two techniques.

Table 1 presents the number of vertices and edges present in the ten graphs. The aim is to compare the performance of SUBDUE with that of the EP-based technique in terms of the compression achieved on these graphs. Note that SUBDUE uses a computationally-constrained beam search for discovering patterns in the data. Both the methods use the size of the graphs as the criterion for evaluating the discovered patterns.

Table 1: Description of graphs used in the experiments

Graph	Number of vertices	Number of edges
W1	8	30
W2	31	43
W3	19	159
W4	94	106
W5	22	314
W6	85	303
W7	27	492
W8	25	532
W9	86	501
W10	732	16725

The population size is set to 50 for the proposed EP-based method. Note that there is no guideline available in the literature for selecting proper parameter values for algorithms in the evolutionary computation paradigm. Here we have selected the population size to be 50 so that it is neither too low (thereby performing very limited exploration of the search space), nor too high (thereby requiring too much computation time). The beam width for SUBDUE is set to 4. The results of SUBDUE and the EP-based technique are provided in Tables 2 and 3 respectively. The tables show the number of vertices, edges, instances and the compression of the best substructure discovered by SUBDUE and the EP-based method respectively. Compression is defined as $\frac{SizeOf(S)+SizeOf(G,S)}{SizeOf(G)}$. Therefore, the lower the value of the compression, the better is the discovered substructure. As can be seen, in eight of the ten cases the EP-based technique provides better results than SUBDUE, while in the other two the results are the same. Since SUBDUE uses a computationally constrained beam search, it discards some of the less promising solutions at an early stage; thereby terminating the possibility that these sub solutions can grow to yield better substructures later on. On the contrary, EP, being a randomized search technique, does not suffer from this limitation. Hence EP is often found to outperform SUBDUE as is evident from the tables.

Conclusions

In view of the increasing structural component of today's databases, there is a requirement for developing data mining systems that can handle structural information. In this direction we have described an EP-based system that can discover knowledge in structural

Table 2: Substructures Discovered by SUBDUE

Graph	Discovered Substructure			Compression
	No. of vertices	No. of edges	No. of instances	
W1	2	1	4	0.868421
W2	2	1	5	0.905405
W3	3	2	5	0.915730
W4	3	2	5	0.925000
W5	2	2	8	0.940476
W6	3	2	19	0.817010
W7	4	3	6	0.944123
W8	3	2	7	0.958707
W9	4	3	17	0.838160
W10	2	1	345	0.960646

Table 3: Substructures Discovered by EP-based Technique

Graph	Discovered Substructure			Compression
	No. of vertices	No. of edges	No. of instances	
W1	4	5	2	0.815789
W2	6	5	3	0.743243
W3	3	3	5	0.893258
W4	8	7	3	0.865000
W5	3	5	5	0.919643
W6	3	2	19	0.817010
W7	3	9	6	0.895954
W8	4	16	5	0.865350
W9	4	6	16	0.771721
W10	2	1	345	0.960646

databases. EP has been used as the underlying search and optimization tool since it has the capability of coming out of local optima. Moreover, currently we are mainly concerned with the task of perturbing (mutating) a solution, rather than combining the information in different solutions (crossover).

Results of the EP-based method are provided for several web data sets. Encouragingly, we find that the proposed technique is often able to find better substructures that can effectively compress the data as compared to another system SUBDUE. It may be mentioned in this context that SUBDUE is computationally constrained by the beam width. If we increase the beam width, then it will also be able to come up with better substructures. However for large problems, significant increases in SUBDUE's beam width will considerably increase the running time. It is in such situations that EP is expected to be more effective as the underlying search technique. We aim to investigate the trade-off between the beam width in SUBDUE and the processing time of the EP-based method as the next step of this research. Toward this end, we will have to first optimize the current working version of EP code for efficiency.

As a part of further research, we aim to apply the developed scheme hierarchically. This means that once a substructure is discovered, the substructure will be used to simplify the data by replacing instances of the substructure with a pointer to the newly discovered substructure. Iteration of the substructure discovery and replacement process can construct a hierarchical description of the structural data in terms of the discovered substructures. The implementation for the hierarchical discovery scheme is complete, and the experimental results are awaited. In addition, since the method of substructure discovery is generally applicable to many structural databases such as computer aided design (CAD) circuit data, molecular data, computer programs, chemical data, etc., we propose to apply it in these areas. It is our hope that these discovered patterns will be of use to the respective communities.

Acknowledgment:

Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-01-2-0570. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

References

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

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

Cook, D. J., and Holder, L. B. 2000. Graph-based data mining. *IEEE Intelligent Systems* 15(2):32–41.

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

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

Fogel, L.; Owens, A.; and Walsh, M. 1966. *Artificial Intelligence through Simulated Evolution*. New York: John Wiley & Sons.

Fogel, L. 1962. Autonomous automata. *Industrial Research* 4:14–19.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison-Wesley.

Gonzalez, J. A. 2001. *Empirical and Theoretical Analysis of Relational Concept Learning Using a Graph-Based Representation*. Ph.D. Dissertation, Department of Computer Science and Engineering, University of Texas at Arlington, USA.

Michalewicz, Z. 1992. *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag.

Mitchell, M. 1996. *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press.

Porto, V. W.; Saravanan, N.; Waagen, D.; and Eiben, A., eds. 1998. *Proc. of the Seventh Annual Conference on Evolutionary Programming*. Lecture Notes in Computer Science. Springer.

Quinlan, J. R. 1993. *C 4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc.

Schwefel, H. P. 1987. Collective phaenomena in evolutionary systems. In *31st Annual Meeting of the International Society for General Systems Research*. 1025–1033.

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