G

Greedy Search Approach of Graph Mining

Lawrence Holder
School of Electrical Engineering and Computer Science,
Box 642752,
Washington State University,
Pullman, WA 99164, USA

Definition

Greedy search is an efficient and effective strategy for searching an intractably large space when sufficiently informed heuristics are available to guide the search. The space of all subgraphs of a graph is such a space. Therefore, the greedy search approach of graph mining uses heuristics to focus the search toward subgraphs of interest while avoiding search in less interesting portions of the space. One such heuristic is based on the compression afforded by a subgraph; that is, how much is the graph compressed if each instance of the subgraph is replaced by a single vertex. Not only does compression focus the search, but it has also been found to prefer subgraphs of interest in a variety of domains.

Motivation and Background

Many data mining and machine learning methods focus on the attributes of entities in the domain, but the relationships between these entities also represents a significant source of information, and ultimately, knowledge. Mining this relational information is an important challenge both in terms of representing the information and facing the additional computational obstacles of analyzing both entity attributes and relations. One efficient way to represent relational information is as a graph, where vertices in the graph represent entities in the domain, and edges in the graph represent attributes and relations among the entities. Thus, mining graphs is an important approach to extracting relational information. The main alternative to a graph-based representation is first-order logic, and the methods for mining this representation fall under the area of inductive logic programming. Here, the focus is on the graph representation.

Several methods have been developed for mining graphs (Washio & Motoda, 2003), but most of these methods focus on finding the most frequent subgraphs in a set of graph transactions (e.g., FSG (Kuramochi & Karypis, 2001), gSpan (Yan & Han, 2002), Gaston (Nijssen & Kok, 2004)) and use efficient exhaustive, rather than heuristic search. However, there are other properties besides frequency of a subgraph pattern that are relevant to many domains. One such property is the amount of compression afforded by the subgraph pattern, when each instance of the pattern is replaced by a single vertex. Searching for the most frequent subgraphs can be made efficient mainly through the exploitation of the downward closure property, which essentially says one can prune any extension of a subgraph that does not meet the minimum support frequency threshold. Unfortunately, the compression of a subgraph does not satisfy the downward closure property; namely, while a small extension of a subgraph may have less compression, a larger extension may have greater compression. Therefore, one cannot easily prune extensions and must search a larger portion of the space of subgraphs. Thus, one must resort to a greedy search method to search this space efficiently.

As with any greedy search approach, the resulting solution may sometimes be suboptimal, that is, the resulting subgraph pattern is not the pattern with maximum compression. The extent to which optimal solutions are missed depends on the type of greedy search and the strength of the heuristics used to guide the search. One approach is embodied in the graph-based induction (GBI) method (Matsuda, Motoda, Yoshida, & Washio, 2002; Yoshida, Motoda, & Indurkhya, 1994).

C. Sammut, G. Webb (eds.), *Encyclopedia of Machine Learning*, DOI 10.1007/978-0-387-30768-8, © Springer-Verlag Berlin Heidelberg, 2010

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.

A similar approach recommended here is the use of a beam search strategy coupled with a compression heuristic based on the ►minimum description length (MDL) principle (Rissanen, 1989). The goal is to perform unsupervised discovery of a subgraph pattern that maximizes compression, which is essentially a trade-off between frequency and size. Once the capability to find such a pattern exists, it can be used in an iterative discovery-and-compress fashion to perform hierarchical conceptual clustering, and it can be used to perform supervised learning, that is, find patterns that compress the positive graphs, but not the negative graphs. This approach has been well studied (Cook & Holder, 2000, 2007; Gonzalez, Holder, & Cook, 2002; Holder & Cook, 2003; Jonyer, Cook, & Holder, 2001; Kukluk, Holder, & Cook, 2007) and has proven successful in several domains (Cook, Holder, Su, Maglothin, & Jonyer, 2001; Eberle & Holder, 2006; Holder, Cook, Coble, & Mukherjee, 2005; You, Holder, & Cook, 2006).

Structure of Learning System

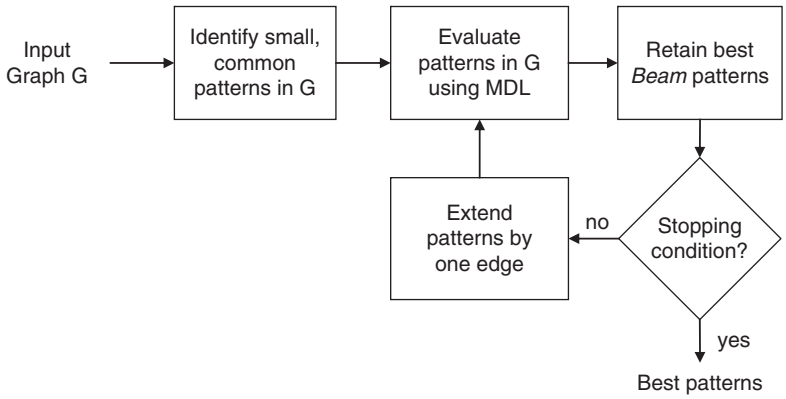
Figure 1 depicts the structure of the greedy search approach of graph mining. The input data is a labeled, directed graph *G*. The search begins by identifying the set of small common patterns in G, that is, all vertices with unique labels having a frequency greater than one. The algorithm then iterates by evaluating the patterns according to the search heuristic, retaining the best patterns, and extending the best patterns by one edge until the stopping condition is met.

The search is guided by the minimum description length (MDL) 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 pattern is the one that minimizes the description length of the input graph when compressed by the pattern. The description length of the pattern *S* given the input graph *G* is calculated as $DL(G,S) = DL(S) + DL(G|S)$, where *DL(S)* is the description length of the pattern, and $DL(G|S)$ is the description length of the input graph compressed by the pattern. The search seeks a pattern *S* that minimizes *DL(G,S)*.

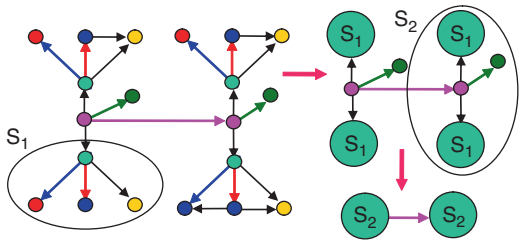
While several greedy search strategies apply here (e.g., hill climbing, stochastic), the strategy that has been found to work best is the beam search. Of the patterns currently under consideration, the system retains only the best *Beam* patterns, where *Beam* is a user-defined parameter. These patterns are then extended by one edge in all possible ways according to the input graph, the extended patterns are evaluated, and then again, all but the best *Beam* patterns are discarded. This process continues until the stopping condition is met. Several stopping conditions are applicable here, including a user-defined limit on the number of patterns considered, the exhaustion of the search space, or the case in which all extensions of a pattern evaluate to a lesser value than their parent pattern. Once meeting the stopping condition, the system returns the best patterns. Note that while the naïve approach to implementing this algorithm would require an NP-complete subgraph isomorphism procedure to collect the instances of each pattern, a more efficient approach takes advantage of the fact that new patterns are always one-edge extensions of existing patterns, and, therefore, the instances of the extended patterns can be identified by searching the extensions of the parent's instances. This process does require several isomorphism tests, which is the computational bottleneck of the approach, but avoids the subgraph isomorphism problem.

Once the search terminates, the input graph can be compressed using the best pattern. The compression procedure replaces all instances of the pattern in the input graph by single vertices, which represent the pattern's instances. Incoming and outgoing edges to and from the replaced instances will point to, or originate from the new vertex that represents the instance. The algorithm can then be invoked again on this compressed graph.

Figure 2 illustrates the process on a simple example. The system discovers pattern S_1, which is used to compress the data. A second iteration on the compressed graph discovers pattern S_2. Because instances of a pattern can appear in slightly different forms throughout the data, an inexact graph match, based on graph edit distance, can be used to address noise by identifying similar pattern instances.



Greedy Search Approach of Graph Mining. Figure 1. Structure of the greedy search approach of graph mining



Greedy Search Approach of Graph Mining. Figure 2. Example of the greedy search approach of graph mining

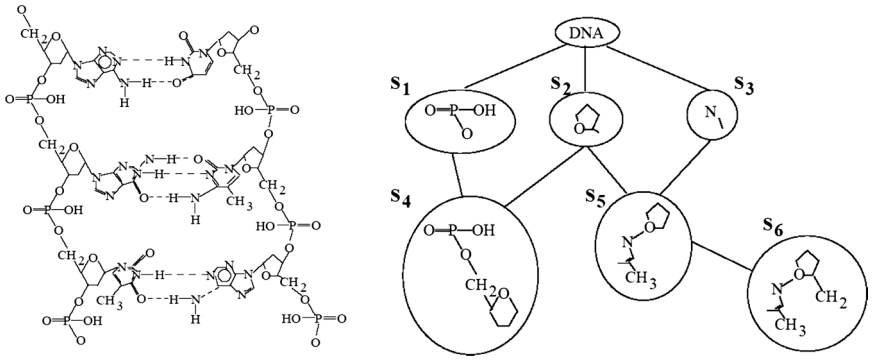
Graph-Based Hierarchical Conceptual Clustering

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 (Jonyer, Cook, & Holder, 2001). On the *i*th iteration, the best subgraph S_i is used to compress the input graph, introducing new vertices labeled S_i in the graph input to the next iteration. Therefore, any subsequently discovered subgraph S_j can be defined in terms of one or more of S_is, where $i < j$. The result is a *lattice*, where each cluster can be defined in terms of more than one parent subgraph. For example, Fig. 3 shows such a clustering done on a DNA molecule.

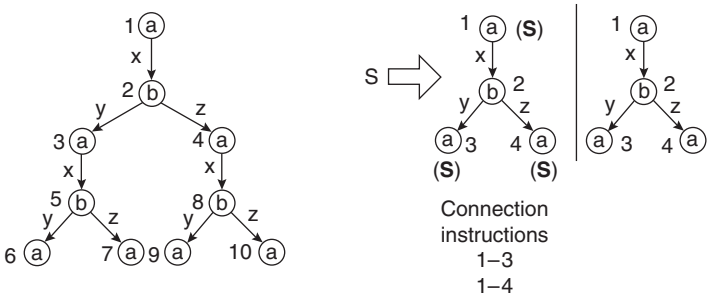
Graph-Based Supervised Learning

Extending a graph-based data mining approach to perform supervised learning involves the need to handle negative examples (focusing on the two-class scenario). In the case of a graph the negative information can come in three forms. First, the data may be in the form of numerous smaller graphs, or graph transactions, each labeled either positive or negative. Second, data may be composed of two large graphs: one positive and one negative. Third, the data may be one large graph in which the positive and negative labeling occurs throughout. The first scenario is closest to the standard supervised learning problem in that one has a set of clearly defined examples (Gonzalez et al., 2002). 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 simply an error-based measure. This approach will lead the search toward a small subgraph that discriminates well. However, such a subgraph does not necessarily compress well, nor represent a characteristic description of the target concept.

One 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 one 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.



Greedy Search Approach of Graph Mining. Figure 3. **Iterative application of the greedy search approach of graph mining yields the hierarchical, conceptual clustering on the right given an input graph representing the portion of DNA structure depicted on the left**



Greedy Search Approach of Graph Mining. Figure 4. **The node-replacement graph grammar (right) inferred from the input graph (left). The connection instructions indicate how the pattern can connect to itself**

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. Note that the compression is a lossy one, that is, one does not keep enough information in the compressed graph to know how the instance was connected to the rest of the graph. This approach is consistent with the goal of learning general patterns, rather than mere compression.

Graph Grammar Inference

In the above algorithms the patterns are limited to non-recursive structures. In order to learn subgraph motifs, or patterns that can be used as the building blocks to generate arbitrarily large graphs, one needs the ability to learn graph grammars. The key to the inference of a graph grammar is the identification of overlapping structure. One can detect the possibility of a recursive graph-grammar production by checking if the instances of a pattern overlap. If a set of instances overlap by a single vertex, then one can propose a recursive node-replacement graph grammar production. Figure 4 shows an example of a node-replacement graph grammar (right) learned from a simple, repetitive input graph (left). The input graph in Fig. 4 is composed of three overlapping substructures. Based on how the instances overlap, one can also infer connection instructions that describe how the pattern can connect to itself. For example, the connection instructions in Fig. 4 indicate that the graph can grow by connecting vertex 1 of one pattern instance to either vertex 3 or vertex 4 of another pattern instance.

If a set of pattern instances overlap by an edge, then one can propose a recursive edge-replacement graph grammar production. Figure 5 shows an example of an edge-replacement graph grammar (right) learned from

the input graph (left). Connection instructions describe how the motifs can connect via the edge labeled "a" or the edge labeled "b."

Apart from the inclusion of recursive patterns, the greedy search approach of graph mining is unchanged. Both recursive and non-recursive patterns are evaluated according to their ability to compress the input graph using the ►MDL heuristic. After several iterations of the approach, the result is a graph grammar consisting of recursive and non-recursive productions that both describe the input graph and provide a mechanism for generating graphs with similar properties.

Programs and Data

Most of the aforementioned functionality has been implemented in the SUBDUE graph-based pattern learning system. The SUBDUE source code and numerous sample graph data files are available at ►http://www.subdue.org.

Applications

Many relational domains, from chemical molecules to social networks, are naturally represented as a graph, and a graph mining approach is a natural choice for extracting knowledge from such data. Three such applications are described below.

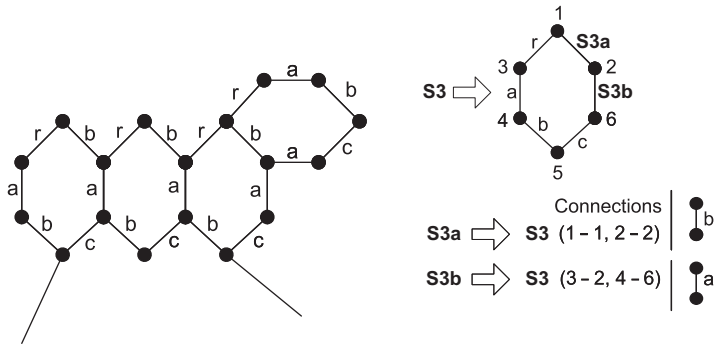
A huge amount of biological data that has been generated by long-term research encourages one to move one's focus to a systems-level understanding of bio-systems. A biological network, containing various biomolecules and their relationships, is a fundamental way to describe bio-systems. Multi-relational data mining finds the relational patterns in both the entity attributes and relations in the data. A graph consisting of vertices and edges between these vertices is a natural data structure to represent biological networks. The greedy search approach of graph mining has been applied to find patterns in metabolic pathways (You et al., 2006). Graph-based supervised learning finds the unique substructures in a specific type of pathway, which help one understand better how pathways differ. Unsupervised learning shows hierarchical clusters that describe the common substructures in a specific type of pathway, which allow one to better understand the common features in pathways.

Social network analysis is the mapping and measuring of relationships and flows between people, organizations, computers, or other information processing entities. Such analysis is naturally done using a graphical representation of the domain. The greedy approach of graph mining has been applied to distinguish between criminal and legitimate groups based on their mode of communication (Holder et al., 2005). For example, terrorist groups tend to exhibit communications chains; whereas, legitimate groups (e.g., families) tend to exhibit more hub-and-spoke communications.

Anomaly detection is an important problem for detecting fraud or unlawful intrusions. However, anomalies are typically rare and, therefore, present a challenge to most mining algorithms that rely on regularity and frequency to detect patterns. With the graph mining approach's ability to iteratively compress away regularity in the graph, what is left can be construed as anomalous. To distinguish this residual structure from noise, one can compare its regularity with the probability that such structure would appear randomly. The presence of rare structure that is unlikely to appear by chance suggests an anomaly of interest. Furthermore, most fraudulent activity attempts to disguise itself by mimicking legitimate activity. Therefore, another method for finding such anomalies in graphs is to first find the normative pattern using the greedy search approach of graph mining and then find unexpected deviations to this normative pattern. This approach has been applied to detect anomalies in cargo data (Eberle & Holder, 2006).

Future Directions

One of the main challenges in approaches to graph mining is scalability. Since most relevant graph operations (e.g., graph and subgraph isomorphism) are computationally expensive, they can be applied to only modest-sized graphs that can fit in the main memory. Clearly, there will always be graphs larger than can fit in main memory, so efficient techniques for mining in such graphs are needed. One approach is to keep the graph in a database and translate graph mining operations into database queries. Another approach is to create abstraction hierarchies of large graphs so that mining can occur at higher-level, smaller graphs to identify interesting regions of the graph before descending down into more



Greedy Search Approach of Graph Mining. Figure 5. The edge-replacement graph grammar (right) inferred from the input graph (left). The connection instructions indicate how the pattern can connect to itself

specific graphs. Traditional high-performance computing techniques of partitioning a problem into subproblems, solving the subproblems, and then recomposing a solution do not always work for graph mining problems, because partitioning the problem means breaking links which may later turn out to be important. New techniques and architectures are needed to improve the scalability of graph mining operations.

Another challenge for graph mining techniques is dynamic graphs. Most graphs represent data that can change over time. For example, a social network can change as people enter and leave the network, new links are established and old links are discarded. First, one would like to be able to mine for static patterns in the presence of the changing data, which will require incremental approaches to graph mining. Second, one would like to mine patterns that describe the evolution of the graph over time, which requires mining of time slice graphs or the stream of graph transaction events. Third, the dynamics can reside in the attributes of entities (e.g., changing concentrations of an enzyme in a metabolic pathway), in the relation structure between entities (e.g., new relationships in a social network), or both. Research is needed on efficient and effective techniques for mining dynamic graphs.

Recommended Reading

Cook, D., & Holder, L. (March/April 2000). Graph-based data mining. *IEEE Intelligent Systems*, 15(2), 32–41.

Cook, D., & Holder, L. (Eds.). (2007). *Mining graph data*. New Jersey: Wiley.

Cook, D., Holder, L., Su, S., Maglothin, R., & Jonyer, I. (July/August 2001). Structural mining of molecular biology data. *IEEE Engineering in Medicine and Biology, Special Issue on Genomics and Bioinformatics*, 20(4), 67–74.

Eberle, W., & Holder, L. (2006). Detecting anomalies in cargo shipments using graph properties. In *Proceedings of the IEEE intelligence and security informatics conference*, San Diego, CA, May 2006.

Gonzalez, J., Holder, L., & Cook D. (2002). Graph-based relational concept learning. In: *Proceedings of the nineteenth international conference on machine learning*, Sydney, Australia, July 2002.

Holder, L., & Cook, D. (July 2003). Graph-based relational learning: Current and future directions. *ACM SIGKDD Explorations*, 5(1), 90–93.

Holder, L., Cook, D., Coble, J., & Mukherjee, M. (March 2005). Graph-based relational learning with application to security. *Fundamenta Informaticae, Special Issue on Mining Graphs, Trees and Sequences*, 66(1–2), 83–101.

Jonyer, I., Cook, D., & Holder, L. (October 2001). Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research*, 2, 19–43.

Kukluk, J., Holder, L., & Cook, D. (2007). Inference of node replacement graph grammars. *Intelligent Data Analysis*, 11(4), 377–400.

Kuramochi, M., & Karypis, G. (2001). Frequent subgraph discovery. In *Proceedings of the IEEE international conference on data mining (ICDM)* (pp. 313–320), San Jose, CA.

Matsuda, T., Motoda, H., Yoshida, T., & Washio, T. (2002). Mining patterns from structured data by beam-wise graph-based induction. In *Proceedings of the fifth international conference on discovery science* (pp. 323–338), Lubeck, Germany.

Nijssen, S., & Kok, J. N. (2004). A quickstart in frequent structure mining can make a difference. In *Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining (KDD)* (pp. 647–652), Seattle, WA.

Rissanen, J. (1989). *Stochastic complexity in statistical inquiry*. New Jersey: World Scientific.

Washio, T., & Motoda H. (July 2003). State of the art of graph-based data mining. *ACM SIGKDD Explorations*, 5(1), 59–68.

Yan, X., & Han, J. (2002). gSpan: Graph-based substructure pattern mining. In *Proceedings of the IEEE international conference on data mining (ICDM)* (pp. 721–724), Maebashi City, Japan.

Yoshida, K., Motoda, H., & Indurkhya, N. (1994). Graph-based induction as a unified learning framework. *Journal of Applied Intelligence*, 4, 297–328.

You, C., Holder, L., & Cook, D. (2006). Application of graph-based data mining to metabolic pathways. In *Workshop on data mining in bioinformatics, IEEE international conference on data mining*, Hong Kong, China, December 2006.