Graph-Based Knowledge Discovery: Compression Versus Frequency

William Eberle
Department of Computer Science
Tennessee Technological University
Cookeville, TN USA
weberle@tntech.edu

Lawrence Holder
School of Electrical Engineering & Computer Science
Washington State University
Pullman, WA USA
holder@wsu.edu

Introduction

There are two primary types of graph-based data miners: frequent subgraph and compression-based miners. With frequent subgraph miners, the most interesting substructure is the largest one (or ones) that meet the minimum support. Whereas, compression-based graph miners discover those subgraphs that maximize the amount of compression that a particular substructure provides a graph. The algorithms associated with these two approaches are not only different, but they also may result in dramatic performance differences, as well as in the normative patterns being discovered. In order to compare these two types of graph-based approaches to knowledge discovery, in the following sections we will compare two publicly available applications: GASTON and SUBDUE.

Methodologies

The goal of GASTON is to return all frequent substructures in a graph using a depth-first search on the input graphs [Nijssen and Kok 2004]. The core of the GASTON algorithm (and other approaches, like *gSpan* [Yan and Han 2002]) is the construction of its hierarchical search tree based upon the DFS code assigned to each graph. Using its canonical tree structure, the algorithm performs a traversal of the tree in order to discover the *frequent* sub-graphs. This search through the DFS codes is repeated on each edge until frequency drops below the minimum support threshold.

The goal of SUBDUE is different from GASTON (and other frequent subgraph miners), in that it chooses to return the substructures that *compress* the graph the best [Holder et al. 1994]. Using a beam search (a limited length queue of the "best" patterns that have been found so far), the algorithm grows patterns one edge at a time, continually discovering what substructures best compress the description length of the input graph. The core of the SUBDUE algorithm is in its compression strategy. After extending each substructure by one edge, it evaluates the substructure based upon its compression value (the higher the better). A list is maintained of the best substructures, and this process is continually repeated until either there are no more substructures to compress.

Algorithmic Differences

Because of the algorithmic choices each methodology uses, there are differences in the amount of memory/space that is used for their internal structures. GASTON uses a sparse adjacency list representation of the graphs, and with its tree structure and pruning strategy (the depth-first search allows for removal of edges and whole graphs from the search space after all relevant patterns for that edge or graph has been processed), it uses a minimal amount of memory. Whereas, SUBDUE uses a significant amount of memory due to the graph structures it builds and the maintenance of best substructures and subgraph instances.

In addition to the pruning that was mentioned, several optimization techniques have been employed by both of these systems. GASTON, as already mentioned, achieves its optimization through its pruning strategy, and of course, it does not do candidate generation. SUBDUE implements not only an optional aggressive form of pruning the list of best substructures, it also performs some heuristic measures to avoid some repetitive comparisons. Each of these optimizations not only reduces the amount of memory needed, they also increase the relative speed of each of these approaches. However, SUBDUE does make more calls to the graph isomorphism tests than GASTON, and the canonical tree ordering implemented in GASTON minimizes its search space.

Empirical Evaluation

We now present the empirical results from experiments comparing the two approaches. We analyzed graphs which are considered sparse (average degree < 2). While there are many different types of graphs, sparse graphs are very representative of real-world phenomenon such as network traffic, citation networks, and social networks. In addition, we kept the size of the sparse subgraph S constant (arbitrarily, 10 vertices and 9 edges), and varied the other parameters as follows:

Number of Transactions (N): 10, 100, 1000, 10000
Max copies of subgraph per transaction (C): 1, 2, 3
Max random edges per transaction (E):5%,10%,15%, 20%

The synthetic graphs were created using a tool called *subgen* that generates random graphs based upon user-specified parameters.

We chose to use SUBDUE and GASTON because very few graph mining tools are publicly available. For SUBDUE, we will use version 5.2 (www.subdue.org). All SUBDUE runs will use the default options (i.e., no parameters specified). For GASTON, we will use version 1.1 (http://www.liacs.nl/~snijssen/gaston). Again, we will just use the default options (i.e., no parameters specified), except for varying minimum support thresholds (*mst*). Now, here is where we will make a slight modification to the baseline code. We will modify GASTON to output the single (best) substructure that maximizes (*frequency* * size). This will be in direct contrast to what SUBDUE discovers, where the best substructure is defined to be the subgraph that *compresses* the graph the best (MDL).

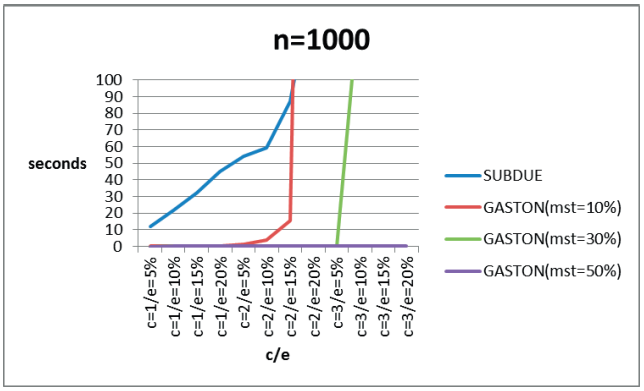

Figure 1. Comparison of running-times for n=1000.

For the experiments where n=10, when c=3, SUBDUE discovers the normative pattern in under 0.3 seconds, whereas for GASTON discovery of the normative pattern with an *mst* of 10% and number of extra random edges (e) at 10%, takes ~2 hours, and more than 16 hours when e > 10%. For the experiments where n=100, when c=3, SUBDUE completes in under 5 seconds, whereas for GASTON discovery of the normative pattern with an *mst* of 10% and number of extra random edges (e) at 5%, takes ~7 minutes, and more than 3 hours when e > 5%. Also, with an *mst* of 30%, runs with e=15% take ~19 minutes and runs with e > 15% do not complete in less than 3 hours. For the experiments where n=1000, when c=3, SUBDUE discovers the best substructure in under 4 minutes, whereas for GASTON discovery of the normative pattern with an *mst* of 10% and number of extra random edges (e) at 5%, takes ~13 minutes, and more than 3 hours when e > 5%. Also, with an *mst* of 30%, runs with e=15% take over 2 hours and runs with e > 15% do not complete in less than 10 hours. (Due to space constraints, only the graph from this set of experiments is shown in Figure .)

For the experiments where n=10000, SUBDUE runs range from a maximum of < 16 hours to minimum running times > 3 hours. GASTON performs well on all runs when the *mst* is at least 50% (less than one second in the worst case), but even at an *mst* of 30%, the runs grow significantly when more connectivity is introduced into the graph. Most runs with an *mst* of 10% had to be terminated after a few hours.

We also need to examine what normative patterns are discovered and how many instances of the normative pattern are reported. For these synthetic graphs, the subgraph seeded into all transactions is as shown in Figure 2. For all SUBDUE and GASTON runs, this normative pattern is reported as the best substructure, with no extra random edges. In addition, in all cases, both approaches report all instances of the best substructure. So, not only does the frequent subgraph mining approach produce identical results to the compression approach, but it is clearly faster when the number of transactions is large. It should also be noted that some of the performance degradation in GASTON, when the *mst* is low, is due to the *connectedness* of the graph. The synthetic graphs that were generated consisted of a mixture of edges between vertices in the same subgraph and vertices in different subgraphs. The less connected the graph, the faster the running times. While it does not appear to affect GASTON when the *mst* is at least 50%, runs under that *mst*, in some cases, do not finish in a reasonable amount of time.

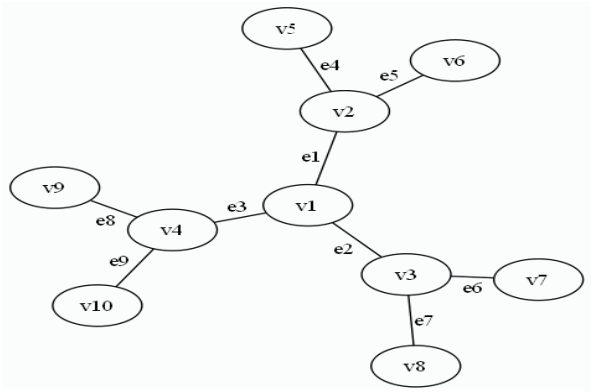

Figure 2. Normative pattern.

References

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

Holder, L, Cook, D. and Djoko, *Substructure Discovery in the SUBDUE System*, AAAI Workshop on KDD, 1994.

Nijssen, S. and Kok, J., *A Quickstart in Frequent Structure Mining Can Make a Difference*, SIGKDD, pp. 647-652, 2004.

Yan, X. and Han, J. *gSpan: Graph-Based Substructure Pattern Mining*. ICDM, pp. 51-58, 2002.