# Knowledge Discovery from Transportation Network Data*

Wei Jiang
Purdue University
250 N. University St.
W. Lafayette, IN 47907-2066
wjiang@cs.purdue.edu

Jaideep Vaidya
Rutgers University and CIMIC
180 University Avenue
Newark, NJ 07102-1803
jsvaidya@rbs.rutgers.edu

Zahir Balaporia
Schneider National, Inc.
Green Bay, WI 54306-2545
balaporiaz@schneider.com

Chris Clifton
Purdue University
250 N. University St.
W. Lafayette, IN 47907-2066
clifton@cs.purdue.edu

Brett Banich
11231 Arrowhead Trail
Indian Head Park, IL 60525
brett_banich@alumni.purdue.edu

## Abstract

*Transportation and Logistics are a major sector of the economy, however data analysis in this domain has remained largely in the province of optimization. The potential of data mining and knowledge discovery techniques is largely untapped. Transportation networks are naturally represented as graphs. This paper explores the problems in mining of transportation network graphs: We hope to find how current techniques both succeed and fail on this problem, and from the failures, we hope to present new challenges for data mining. Experimental results from applying both existing graph mining and conventional data mining techniques to real transportation network data are provided, including new approaches to making these techniques applicable to the problems. Reasons why these techniques are not appropriate are discussed. We also suggest several challenging problems to precipitate research and galvanize future work in this area.*

## 1 Introduction

Transportation and logistics are an important sector of the economy. Transportation consumes 60% of oil worldwide[11], and the number is increasing. Data mining has lead to significant gains in other areas, and should also be used to improve this sector of our economy. Computer use is widespread in transportation and logistics. Inventory management, parcel tracking, and even on-truck location sensors provide a wealth of data. This seems a natural application area for data mining, however, to date, there have been few success stories.

There has been some mining with freight flow data, but with transactional characteristics of freight and events such as safety/accident records rather than the geometry of the network. For example, classification on safety/accident records might find that trucks are prone to accidents at 7:00 AM on east - west roads (i.e., when the sun is in the drivers' eyes.) A similar problem could be to find conditions in which trucks suffer from mechanical failure to predict a requirement for maintenance.

Existing methods that utilize the network structure fall within the domain of optimization. Optimization techniques have long been computerized, but do not provide the kinds of insights that are the goal of data mining. These optimization techniques generally fix certain constraints and allow a manageable number of free variables to be adjusted to achieve minimal cost. The goal of our data mining is not to better optimize these free variables, but to generate knowledge that enables a business to adjust its process to modify the *constraints*, enabling the optimization to generate a lower cost solution. Optimization may realize a feasible solution, but data mining could find patterns where constraints could be modified or re-defined to provide a more robust optimal solution. One of the few reported data mining successes in this area involves management of inventory levels[3, 4], where the constraints on safety stocks were modified to achieve lower inventory and higher in-stock rates.

Transportation networks can be viewed as graph problems, with vertices corresponding to the origins and destinations, and edges corresponding to goods moved from the

*This work supported by a grant from the Purdue's e-Enterprise Center at Discovery Park.

Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)
1084-4627/05 $20.00 © 2005 **IEEE**

origin to the destination. What makes an interesting pattern in this data? One goal is to better utilize truck capacity, eliminating the need to deadhead empty trucks. Another is to get loads delivered more quickly. The optimization community has worked on these problems for years. The goal of data mining must go beyond this. By finding interesting and frequent patterns, we hope to allow transportation experts to identify changes that go beyond the traditional optimization methods.

There are several known "good" shapes in transportation networks. For example, a circular route enables a truck to move from point to point, pick up and drop off goods at each point and regularly return home. A hub-and-spoke built around a central warehouse or sorting facility is also efficient. By finding new patterns in the data, we hope to enable transportation experts to devise new methods to make those a "good" pattern.

Graph mining would seem a natural fit, however existing graph mining work does not map well to this problem. Much of the prior work has focussed on finding frequent sub-graphs from a set of graph "transactions" [10, 13, 20, 23]. This work is not directly applicable to the transportation domain, as transportation data is not easily broken down into a collection of graphs over the same vertices and edges. There has been some work on mining single graphs, notably the SUBDUE system [9, 5], but this also turns out to be a poor match for our problem.

Despite the above, we wish to leverage prior work to quickly find data mining techniques that apply to transportation network data. This paper discusses problems, and presents first-cut solutions based on different ways of partitioning data, so that existing graph mining algorithms can be used to obtain meaningful solutions. These solutions are tested through experiments with actual transportation network data from a real logistics management company, using both non-graph techniques and graph techniques such as SUBDUE[9, 5] and FSG[13] to identify frequent subgraphs. From this, we identify challenges and directions for graph mining research.

We first discuss prior work in graph mining. In Section 3, we give background on our transportation network data. Section 4 formally defines when two sub-graphs within a single graph support the same pattern. We then introduce approaches to transform this data into forms amenable for use with existing graph mining algorithms. Section 5 divides the graph structurally. This enables discovery of patterns that re-occur in multiple geographic locations. In Section 6, we describe a temporal-based partitioning, finding patterns that occur over time. Section 7 converts the data into a pure transactional form, applying conventional data mining tools. From experimental results, Section 8 analyzes the limitations of the existing graph mining softwares and the conventional data mining algorithms. We then conclude with challenges and research directions: What problems are *really* of interest in this domain, and the resulting generalized research challenges.

## 2 Related Work

Graphs are used to model semi-structured data in several domains [13, 16, 17, 22]. Graph Mining is still a relatively new research area. There has been a recent surge in interest in finding frequent patterns from different types of graphs and trees. Many of the proposed algorithms are based on the well known Apriori principle [1]; modifying the functionality of candidate generation, candidate selection, and support pruning to support graph data.

The first Apriori-based algorithm, WARMR [6], was developed in response to the IJCAI'97 challenge on predicting chemical carcinogenicity. [10] proposed an Apriori-based algorithm, AGM, to discover all frequent (both connected and disconnected) structures from a set of graphs. FSG [13] and gFSG[14] also find frequent graph patterns, but using edges as building blocks. [20] studies the problem of discovering typical patterns of graph data, and proposes a new algorithm for mining graph data based on a novel definition of support that works for both labeled and unlabeled data. [23] proposes a new lexicographic order among graphs and uses this to construct a novel algorithm to mine frequently connected sub-graphs efficiently. *TreeMinerH* and *TreeMinerV* [24] are algorithms for efficiently mining frequent trees in a forest of ordered, rooted trees. *TreeFinder* [19], on the other hand, finds approximate patterns from both ordered and unordered trees.

A common feature of this work is the search for patterns that are common to a set of distinct graphs. This is a natural outgrowth of the IJCAI'97 toxicity challenge, finding common substructures in a variety of chemical compounds. Each compound is represented as a graph, with vertices as molecules and edges as bonds. The goal is to find the common structural features among a set of compounds that are similarly active (e.g., are carcinogenic). Knowledge of the common features would lead to a better understanding of what makes a compound active, improving drug discovery opportunities, speeding toxicity studies, etc.

To give a better understanding of approach to graph mining, we give the problem definition as stated for FSG[13]. A dataset $D$ is divided into transactions, where each transaction $t \in D$ is a labeled undirected graph. Given a minimum support $s$, the goal is to find connected subgraphs that occur in at least $s \cdot |D|$ transactions. A subgraph $g$ occurs in a graph $t$ if $g$ is isomorphic to $t' \subset t$, where isomorphism is defined to include matching the labels as well as the vertex/edge structure. FSG proceeds using the Apriori principle, first finding small frequent subgraphs, then joining these to find candidate larger subgraphs. The candidate

generation phase is more complex than for plain transactions, as joining to size $k-1$ subgraphs can form many size $k$ subgraphs. However, the basic approach is similar.

The problem with this work is that not all graph problems are easily modeled as a set of independent graphs. For example, the entire world wide web can be appropriately modeled as a single graph[7]. Transportation networks can be modeled as a single graph (or perhaps a multigraph: different edges between a pair of vertices may indicate multiple trips), with vertices corresponding to places and edges corresponding to goods to be moved from sources to destinations. There has been some work in mining single graphs, or more specifically from single trees. FREQT [2] was developed to identify frequent patterns in semi-structured data, specifically XML documents. However, there has been little work that applies to general graphs.

SUBDUE [9, 5] is one of the few systems that works on single graphs. The SUBDUE substructure discovery system discovers interesting and repetitive subgraphs in a labeled graph representation using the minimum description length principle. By replacing previously discovered substructures in the data, multiple passes produce a hierarchical description of the structural regularities in the data. It also allows limited inclusion of background knowledge. SUBDUE has been applied to several domains, such as molecular biology, image analysis and computer-aided design. In Section 5.1 we describe experiments we ran with SUBDUE on our transportation network data. As will be described, we found several issues. Suffice it to say that SUBDUE (as it now stands) is not sufficient for our purposes.

In the next two sections, we first provide descriptions of our transportation network data. Then we describe the basic graph structure we are interested in and the sort of patterns we would like to find in it.

## 3 Transportation Network Data Description

We present experiments with six months of origin-destination (OD) data from a large third-party logistic company. The dataset consists of 98,292 transactions. Each transaction has 11 attributes, described in Table 1. There are $4038$ distinct latitude-longitude (LL) pairs in the dataset, with 1797 distinct origins and 3770 distinct destinations (several locations are both). The dataset contains $20,900$ distinct OD pairs (i.e., there are often multiple deliveries between the same origin and destination over the six months). The edges are labeled with the other attributes of the transaction: pickup date, delivery date, distance, hours, weight, and mode.

Labeling edges with the exact values would lead to few frequent patterns being detected, since the edge labels are often unique. Instead, we use a binning strategy. Each label (distance, hours, weight) is divided into ranges, giving a few distinct labels for each type (we used seven for gross weight and ten for transit hours in the experiments reported in this paper). As a result, even edges with similar (though not exactly equal) distances, times, and weight are considered to support the same pattern. Since the range of values is quite large, this makes perfect sense. For example, the range for weight is about 500 tons - in this case, two different transactions from the same source to the same destination with weights 49 tons and 52 tons respectively should be considered equal. Binning facilitates this.

This dataset is naturally represented as a directed graph by mapping locations to vertices. Each transaction can then be represented as the edge of an OD pair. The dataset does give a fully connected graph. Minimum, maximum, and average out-degrees are $1$, $2373$, and $12$ respectively, and Minimum, maximum, and average in-degrees are $1$, $832$, and $6$.

We generate three different graphs from the data, named $OD\_GW, OD\_TH, OD\_TD$. Each graph has the same set of vertices and edges but different labeling scheme for the edges. $OD\_GW$ uses $GROSS\_WEIGHT$, $OD\_TH$ uses $MOVE\_TRANSIT\_HOURS$, while $OD\_TD$ uses $TOTAL\_DISTANCE$ for edge labeling.

## 4 Interesting Patterns

In Section 1, we pointed out some "good" patterns, but in many cases, even the notion of a pattern is open. The real data is composed of a single graph with multiple labels on the edges. However, the concept of a frequent pattern requires that we define when two subgraphs support the same pattern. With a single graph, this raises issues such as overlap (should a single vertex be allowed to belong to two different graphs supporting a pattern?) We instead partition the data into separate sub-graphs (graph transactions), and search for frequent patterns across the sub-graphs. How the partitioning is done determines what the patterns mean.

Here, we formally define when two subgraphs support the same pattern. Let a labeled graph $G$ be represented as $G = (V, E)$, where $V$ is the set of vertices and $E$ is the set of edges. $label(x), x \in V \cup E$ is a function that returns the label of an edge or vertex. $G' = (V', E')$ is a sub-graph of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. Two sub-graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are considered identical if there is an isomorphism between $G_1$ and $G_2$ such that the labels of the edges and vertices match, i.e., the vertices $v1_a \in V_1$ can be mapped to $v2_i \in V_2$ such that if $v1_a$ maps to $v2_i$, $label(v1_a) = label(v2_i)$; if $v1_a$ maps to $v2_i$, $v1_b$ maps to $v2_j$, and there is an edge $e1_c$ from $v1_a$ to $v1_b$, then there is a corresponding edge $e2_k$ from $v2_i$ to $v2_j$ such that $label(e1_c) = label(e2_k)$; and vice-versa.

Assuming a support threshold $s$, we wish to find all

**Table 1. Transportation Network Data Description**

| Name | Description |
|---|---|
| ID | Unique transaction identifier. |
| REQ_PICKUP_DT | Requested date to pick up the load. |
| REQ_DELIVERY_DT | Requested delivery date. |
| ORIGIN_LATITUDE | Latitude of source (to nearest 0.1 degree.) |
| ORIGIN_LONGITUDE | Longitude of source (to nearest 0.1 degree.) |
| DEST_LATITUDE | Latitude of destination (to nearest 0.1 degree.) |
| DEST_LONGITUDE | Longitude of destination (to nearest 0.1 degree.) |
| TOTAL_DISTANCE | Road miles between origin and destination. |
| GROSS_WEIGHT | Weight of load. |
| MOVE_TRANSIT_HOURS | Hours needed to get from origin to destination. |
| TRANS_MODE | Truckload or Less than Truckload. |

sets $P$ of distinct subgraphs of $G$ such that $|P| \geq s$ and $\forall p_i, p_j \in P$, $p_i$ is identical to $p_j$.

In the following two sections, we define two valuable patterns (with distinct underline meanings) for decision making in transportation and logistic domains. We propose first cut approach that utilizes the existing graph mining softwares to discover these patterns. From experimental results, although the existing softwares are not suitable to mine these patterns and the first cut approach is limited, it does present the need for further research in this area.

# 5 Structurally Similar Routes

One problem we have addressed is identifying *self-similarity* within the transportation network. The goal is to identify structurally similar patterns that occur in many locations. For example, a pattern might be a "bow-tie" with several small loads converging on a location, large loads to a distant location, and small loads converging on the distant location. Seeing this pattern, a transportation expert could find a way to better utilize resources outside the bounds of traditional optimization methods. Instead of just optimizing truck routes, the company could use multi-modal transportation, placing trailers on rail cars for the large load long distance portion of the pattern, and using the rail-capable trailers as a pool for the short deliveries in the vicinity of the endpoints. This is just a hypothetical example; best utilizing the discovered patterns requires considerable external knowledge. The key is that since the patterns are *frequent*, innovative transportation approaches that optimize deliveries in those patterns can be applied in many places.

Since we are interested only in structural similarity and not particular locations, we assign all vertices the same label. Thus, vertex labeling is a non-factor in finding frequent sub-graphs. The three variants for edge labels; weight, distance, and time; were described earlier.

## 5.1 Experiments with SUBDUE

SUBDUE [9, 5] is a substructure discovery system that discovers interesting and repetitive subgraphs in a labeled (either connected or disconnected) graph representation. We used release 5.1 of SUBDUE to run experiments on our transportation data. SUBDUE has three methods available for evaluating candidate substructures: Minimum Description Length (MDL), Size, and Set Cover principles. More details on how these work can be found in the user documentation for SUBDUE. We performed experiments with two of those methods (the MDL and Size principle). The Set Cover principle is not relevant, as the value of a substructure S is computed as the number of positive examples containing S plus the number of negative examples not containing S, divided by the total number of examples. The transportation data has no concept of negative examples. In order to get performance estimates, we also ran the experiments on sub-graphs of various sizes. These subgraphs were derived from the original graph by selecting the required number of vertices and then including all of the edges incident on vertices present in the graph. We also ran the experiments on graphs derived from both the Gross Weight attribute as well as the Total Distance attribute. We now describe the results from some of those experiments.

SUDBUE took about 3.25 hours to handle a graph of 100 vertices and 561 edges using the Minimum Description Length principle to find the best 3 patterns of beam size 4. It took about 3 days on the same graph when asked to find the 15 most frequent patterns of up to size 6 using the size principle. Figure 1 [1] shows some of the results obtained using the MDL principle. For example, one of the (very frequent)

[1] Vertex labels on figure 1, 2, 3 did not play any role in the process of pattern discovery. Since we divided the actual edge labeling value into sub-intervals, each edge label on figure 1, 2, 3 represents a disjoint interval. On the other hand, both vertex labels and edge labels on Figure 4 participated in the process of finding patterns.

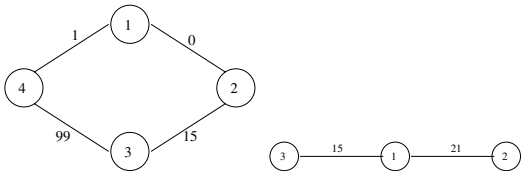


**Figure 1. Frequent patterns found by Subdue on the graph data $OD\_GW$**

patterns we find here demonstrates possible deadheading of trucks. Thus, there seems to be significant traffic from node 2 to node 4 via node 3, but not much return traffic. This might lead to opportunities for pricing changes, e.g., lowering prices on the deadhead routes, that would be outside the realm of traditional route optimization. SUBDUE required about 304 hours (12 days) when ran on a truncated graph consisting of the same 4037 vertices and only about 900 edges, but found fairly trivial results. While we are attempting to run it on the complete graph of over 6 months of data, based on prior performance, we expect completion to take months. All the experiments were done without allowing overlap in the patterns. After running several experiments, it was fairly obvious that running with the MDL principle tends to give trivial results. This matches the expectation from the data, since all the vertices are labeled the same. With MDL, SUBDUE finds a large number of repeated patterns of size 1. It tends to get better compression using these small patterns rather than any of the interesting larger patterns because the larger patterns are relatively infrequent.

We found more interesting results with the Size principle. When run on the truncated graph for Total Distance of 100 vertices and 561 edges looking for the best 5 substructures of maximum beam size 5, it took about 4.9 days (118 hrs) but found very complex patterns. We found a pattern consisting of 31 vertices and 37 edges that was repeated twice in the graph without overlap. We were also looking only for exact matches. Given that there can be at most three non-overlapping patterns of this size in a graph of 100 vertices (31*3 = 93), this seems highly unusual. It would be very interesting to see if this pattern could appear frequently in the full dataset, but the time required for such search with SUBDUE is prohibitive.

To summarize, SUBDUE does not find very interesting patterns from the transportation viewpoint when evaluation is done with the MDL principle. The size principle shows promise of giving interesting results. Unfortunately, in both cases, SUBDUE requires long run times even on small graph sizes. A big challenge is to further improve the running time and make mining on the entire graph feasible.

**Algorithm 1** Finding frequent subgraphs in a single graph

**Require:** $k$ be the number of transactions to partition the graph into, $m$ be the number of repetitions, and $s$ be the support threshold.
$result = \emptyset$
**for** $i = 1..m$ **do**
  $G_1, \ldots, G_k = SplitGraph(G, k)$
  $result = result \cup Find\_Frequent\_Graphs(s, G_1, \ldots, G_k)$
**end for**
return $result$

## 5.2 Experiments with FSG

FSG is another existing software for graph mining, but it merely mines patterns across a set of graph transactions. Therefore, in order to find patterns in a single graph, we attacked the problem with approaches designed for finding patterns across multiple graphs. To support this, we subdivide the single graph into multiple distinct sub-graphs, and each sub-graph is treated as a separate transaction. We then used FSG on these transactions to find frequent sub-graphs This was repeated several times, with a different partitioning each time – if a sub-graph is frequent across a particular partitioning, it is frequent in the entire graph. (Running multiple times decreases the number of false drops, or subgraphs that do not appear frequent because they get split by the partitioning process.) This is described formally in Algorithm 1.

In order to utilize FSG, in the following subsection, we propose two different partitioning strategies breadth / depth first for the $SplitGraph$ procedure. Each strategy is incremental: a single subgraph is pulled from the overall graph $G$ at each step. $G$ is then modified to prevent getting the same subgraph again, as overlapping subgraphs would result in false positives (i.e., the same part of the graph would be repeated in several sub-graphs and thus reported to be frequent). This process is repeated until the graph is fully partitioned. Efficient graph partitioning algorithms are available, e.g., METIS[12]. However, in the experiment with FSG, we adopt breadth / depth first partitioning strategies because they allow us to control the type of patterns preserved after partitioning (e.g., hub-spoke or long chain patterns).

### 5.2.1 Breadth first / Depth first partitioning

The key idea for both partitioning techniques is similar. In both cases, we obtain a sub-graph by randomly choosing a starting vertex in the graph $G$. All edges from that node are added to the sub-graph, along with the endpoint vertices. One of the endpoint vertices is chosen as the next starting vertex, and the process is repeated. This continues until the

**Algorithm 2** $SplitGraph$ Breadth first / Depth first partitioning

**Require:** Graph $G$, number of transactions $k$, ordering structure $q$ (queue for breadth-first, stack for depth-first)
  **for** $transactions = 1..k$ **do**
    $edges = |E|/(k - transactions)$
    Randomly select a vertex from $G$ and insert into $q$
    **while** $edges > 0$ and not $q.empty$ **do**
      $v =$ remove next vertex from $q$
      Add $v$ to the new graph $G'$
      **while** $edges > 0$ and $v$ has edges remaining **do**
        $e =$ get edge adjoining $v$
        add $e$ to $G'$
        remove $e$ from $G$
        $edges = edges - 1$
        Add the vertex on the other end of $e$ to $G$ and $q$
      **end while**
    **end while**
    Return $G'$
    Remove "orphaned" vertices from $G$ (those with no edges)
  **end for**

specified sub-graph size (number of edges) is reached, or the sub-graph cannot be expanded further. At this point, the edges in the sub-graph are removed from the original $G$, along with any orphaned vertices. The process is repeated until no edges remain in $G$.

The choice of which vertex to use as the next vertex gives different ways of growing the graph. We present results using both depth-first search and breadth-first search. This is shown in Algorithm 2. The algorithm attempts to keep similar partition sizes although the potential for disconnected graphs forming as $G$ is partitioned may result in some smaller and larger partitions. Note that in both strategies, as long as the graph $G$ is appropriately modified by removing the selected sub-graph, we should get almost mutually exclusive sub-graphs by this partitioning.

While ideally we would hope to find any frequent patterns, the partitioning may result in patterns being broken across partitions, and thus not appearing to be frequent. The different partitioning approaches give different types of breaks. Breadth first partitioning allows us to find complex frequent patterns where the nodes have high out-degree. Depth first partitioning, on the other hand, leads to frequent patterns that are long chains. Our experimental results on real data show that the different partitioning approaches do give different results; we do not believe either approach is really finding all the frequent patterns in the data.[2]

[2]Tests on simulated data constructed by joining subgraphs with known frequent patterns to form a single graph, and then partitioned, show recall rates in the 50% and above range with both depth-first and breadth-first partitioning, with better results for smaller graphs.

#### 5.2.2 Experimental Results

We ran experiments on the transportation network data using both partitioning techniques. We also tried different partition sizes; 400, 800, 1200 and 1600. The support was set at 120 occurrences for the depth-first partitioning and 240 for the breadth-first. An average of 667 frequent patterns were found with the breadth-first partitioning, 200 with depth-first. The smaller number of partitions actually gave a larger number of frequent itemsets; this is not surprising, as these produced larger graphs with more potential for overlap.

Some interesting sample results obtained on the real data are displayed in Figures 2 and 3. The pattern displayed in Figure 2 was found to be frequent in 243 instances. This is a hub-and-spoke pattern and common in transportation networks. A likely interpretation is a single delivery source (e.g., a factory) that delivers products to many destinations. Finding this pattern provides some validation of the efficacy of our partitioning strategies.

The pattern in Figure 3 was frequent in 63 instances and is found only by depth first partitioning of the graph. This would be interesting from a transportation point of view, as it points out a likely case for a repeated route, making deliveries and pickups at each node. Such routes are common in transportation networks, where deliveries are made from a common origin to many destinations (e.g., a delivery route). Of note is that this pattern does not reflect a delivery from a common origin, but a combination of pickups and deliveries on relatively short routes – an effective route (since the truck is always utilized) that might be missed in trying to find traditional delivery routes.

One problem we encountered during the experiment is that we had runtime and memory problems with lower supports on the breadth-first partitions. It is safe to say that with low support and large size of sub-graph transactions, even though a single graph can be partitioned into a set of smaller sub-graph transactions, FSG is not an appropriate tool to use for mining recurrence patterns in a large single graph.

## 6 Temporally Repeated Routes

The preceding experiment does not exploit the temporal nature of the transportation graph. We tried a second experiment where we looked for patterns of routes repeated in time, rather than space. In other words, an interesting route would be one that happened many times between the same places.

To find such patterns, we partitioned each graph into a set of graph transactions based on date. Each graph represented all active OD pairs on that date. In other words, if a graph transaction $d$ is between the required-pickup date and the

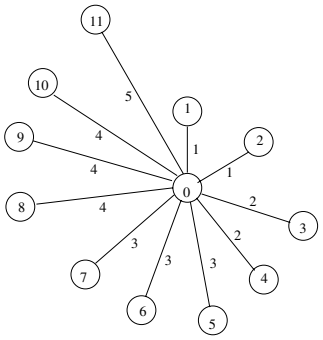


**Figure 2. Sample results on graph data $OD\_TH$ (BF partitioning)**

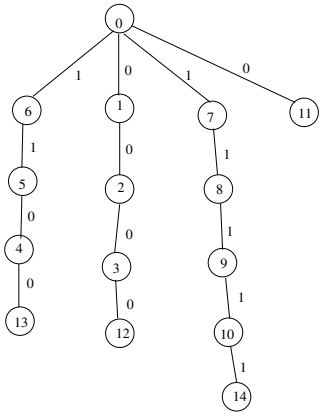


**Figure 3. Sample results on graph data $OD\_TD$ (DF partitioning)**

required-delivery date of an OD pair, the pair is considered an active edge in $d$. Note that the same edge in the original graph may appear in several graph transactions, based on the number of days on which the load may be moved.

We present experiments based on a graph whose edges are labeled based on a range of the gross weight of the load. In other words, a load that would sometimes require a large truck and other times only fill a small truck would not support the same frequent pattern. Since the goal is to find repeated patterns in the same location across time, each vertex is given a unique label based on its latitude and longitude (points within a few miles are coalesced to the same vertex.)

Many individual graph transactions contained multiple connected components. We further broke each disconnected graph transaction into multiple connected graph transactions. Although FSG does not require its input to be a set of connected graphs, breaking a disconnected graph transaction into multiple connected graphs does not affect the outcomes because FSG only finds connected frequent patterns and the distinct labels prevent the different graphs on a single day from supporting a common pattern. Before running through FSG, transactions that have only one edge were also eliminated as not producing interesting patterns. We also had to remove duplicate edges within each transaction, as FSG operates on graphs, not multigraphs. Table 2 shows statistics on the graph transactions after partitioning.

### 6.1 Experimental Results

We were unable to run FSG on the entire data set due to insufficient memory / swap space. This was on a Sparc / Solaris with a 900MHZ processor, 1GB main memory and 1.5GB swap space. We found that the large graphs caused problems – even at 100% support, we could not run FSG on the large graph transactions.

When we limited the data to dates with fewer than 200 distinct vertex labels, FSG produced 22 frequent patterns at 5% support. (The data after partitioning is summarized in Table 3.) Most were small patterns, the largest was a three-edge hub and spoke pattern (Figure 4). While this shows potential, we need the ability to find more complex patterns on full data sets to get results with real-world impact.

## 7 Patterns Discovered by Using Conventional Mining Algorithms

One approach to pattern discovery in this data is to map the dataset into a standard "transactional" representation and use traditional data mining approaches. In this section, we investigate the kinds of results that can be accomplished by several conventional data mining algorithms. While this loses much of the information contained in the network structure, we want to evaluate this approach to present the

**Table 2. Summary of Temporally Partitioned Graph Data**

| | |
|---|---|
| Number of Input Transactions: | 146 |
| Number of Distinct Edge Labels: | 7 |
| Number of Distinct Vertex Labels: | 3835 |
| Average Number of Edges In a Transaction: | 1092 |
| Average Number of Vertices In a Transaction: | 601 |
| Max Number of Edges In a Transaction: | 4462 |
| Max Number of Vertices In a Transaction: | 2140 |
| | |
| The Number of Graph Transactions with Size between 1 to 10: | 73 |
| The Number of Graph Transactions with Size between 10 to 100: | 5 |
| The Number of Graph Transactions with Size between 100 to 1000: | 3 |
| The Number of Graph Transactions with Size between 1000 to 2000: | 31 |
| The Number of Graph Transactions with Size between 2000 to 5000: | 34 |

**Table 3. Summary of Data Used in Frequent Pattern Discovery**

| | |
|---|---|
| Number of Input Transactions: | 53 |
| Number of Distinct Edge Labels: | 7 |
| Number of Distinct Vertex Labels: | 154 |
| Average Number of Edges In a Transaction: | 4 |
| Average Number of Vertices In a Transaction: | 5 |
| Max Number of Edges In a Transaction: | 8 |
| Max Number of Vertices In a Transaction: | 9 |

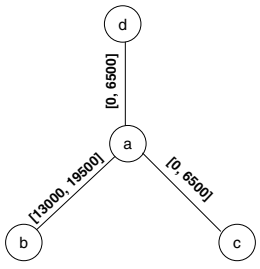


**Figure 4. Temporally frequent pattern with weight ranges as edge labels**

need for novel data mining algorithms that emphasize the structures of the transportation network data.

We used **Weka**[21] for association rule mining, instance (tuple) classification and cluster analysis on the transportation data. The trucking data contained two dates (REQ_PICKUP_DT, REQ_DELIVERY_DT) that are of critical importance when trying to discover periodicity patterns. Since Weka maps the DATE attribute type to a REAL, interpreting experiment results is non-trivial. This led to our exclusion of these two attributes in the following experiments.

### 7.1 Association Based Experiments

We ran the following two experiments to mine association rules:

- Experiment 1: Discretize original data set and run Apriori
- Experiment 2: Find associations, if any, between origin and destination pairs (this experiment was run using only the origin and destination latitude and longitude attributes from the data set).

The results from Experiment 1 yielded associations of the form: $GROSS\_WEIGHT(X, (-\infty, -4501]) \rightarrow TRANS\_MODE(X, LTL)$, where TRANS_MODE has two distinct values: Less than Truckload (LTL) and Truckload (TL). The conclusion is trivial: a lightweight load is usually an LTL shipment, and the reverse holds also. An interesting rule found in the result sets of both experiments with a confidence of 0.87 concluded that: $ORIGIN\_LONGITUDE(X, (-84.76, -75.43]) \rightarrow ORIGIN\_LATITUDE(X, (39.8, 44.08])$. This rule is useful to generalize the geographical area a shipment originates from.

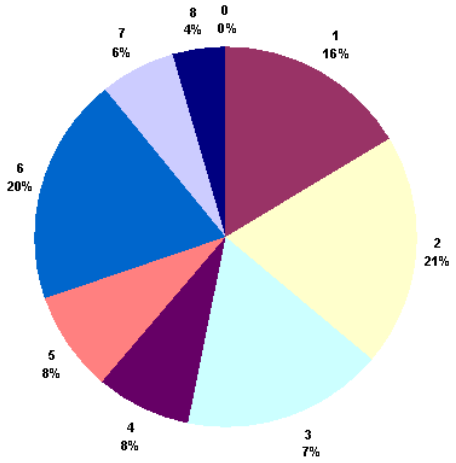


**Figure 5. Clustering statistics for Transportation Data**

## 7.2 Classification Based Experiments

We first ran Weka's version of the C4.5 algorithm (J4.8) on the discretized dataset. The J4.8 model was 96% accurate in classifying instances based on the class attribute TRANS_MODE {LTL, TL}. The classification tree first splits on the GROSS_WEIGHT attribute, which is consistent with the strong association rules generated by the Apriori algorithm in section 7.1. J4.8 was also run on the discretized dataset with the TRANS_MODE attribute removed and TOTAL_DISTANCE set as the class attribute. The results from this experiment were interesting in the sense that TOTAL_DISTANCE and MOVE_TRANSIT_HOURS were not as highly correlated as either TOTAL_DISTANCE and DESTINATION_LATITUDE or TOTAL_DISTANCE and ORIGIN_LATITUDE.

## 7.3 Clustering Based Experiments

We used the original undiscretized data set as the training set input to the EM (expectation-maximization) algorithm. The algorithm works by assigning each object to a cluster based on a weight representing the probability of membership. Figure 5 summarizes the results produced by the EM algorithm. The training data were split into nine clusters varying in size from 3 instances in cluster 0 to 19,386 instances in cluster 2.

Figure 6(a) reflects the mean TOTAL_DISTANCE within each cluster. Similarly Figure 6(b) reflects the mean TRANSIT_HOURS within each cluster. As is evident in the figures, Cluster 0 contains the outliers in this data set. These three shipments have on average, traveled over 3,000 miles in less than 24 hours. Looking at the latitude and longitude

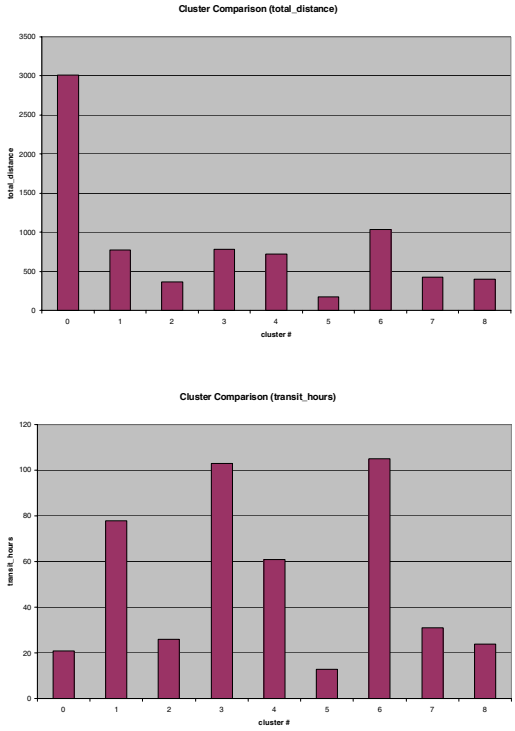


**Figure 6. Clustering statistics for Transportation Data**

of the origin and destination points (as well as the gross weight), one can conclude the shipments are handled as air freight (originating in the Pacific Northwest and delivered to Hawaii). Similar analyses can be applied to characterize the remaining clusters. Clusters 2, 5, 7, and 8 can be grouped together and labeled "short-haul". Similarly, clusters 1, 3, 4, and 6 can be referred to as "long-haul" routes.

Traditional data mining techniques have produced interesting and meaningful results to summarize our data despite the exclusion of two critical temporal attributes. Further experimentation with traditional mining algorithms is required to explore the potential and limitations of these techniques on temporal transportation network data. In addition, from these data mining results, it is obvious that the insights from the structural characteristics of the data cannot be derived. As a consequence, conventional methods can only produce limited insights on this transportation and logistic dataset.

## 8 Limitations of Existing Techniques and Opportunities for Improvements

From the previous sections, some of the limitations of SUBDUE and FSG are obvious. Even though SUBDUE can discover patterns from connected and unconnected graphs, from a transportation perspective, it does not find very interesting patterns with the MDL principle. With the size principle, SUBDUE produced interesting results. Unfortunately, SUBDUE requires long running times even on small graphs, especially with the size principle. A first challenge is to substantially improve the running time and make mining on the entire graph feasible.

Section 6 highlights the limitations of FSG, and most likely other Apriori-style graph mining algorithms. While breaking a disconnected graph transaction into multiple connected graph transactions increases the number of graph transactions in each transaction set; it should allow FSG to speed up because the decrease in average transaction size should substantially decrease the required amount of computation of sub-graph isomorphism. With graph data that naturally forms large graphs, as in the transportation and logistic domain, partitioning the data in natural ways (e.g., temporally) can produce very large graph transactions. In addition, transaction sets in our problem domain had a much larger number of distinct vertex labels than the chemical compound dataset used in [13] and others, which has 4 edge labels, 66 vertex labels and 340 transactions with average size 27.4 edges and 27 vertices.[3] That the large number of distinct labels can cause very large candidate sets leads to insufficient memory to store these candidate sets. Since the temporal partitioning strategy produced very large graphs that caused FSG to fail, in order for FSG to work, people may wonder if we could use the breadth / depth first partitioning strategies to further break these large sub-graphs. Although it is possible to further divide a large partition into smaller sub-graph transactions, we may lose many potentially frequent sub-graph patterns. Based on the fact that sub-graph isomorphism is NP-Complete [8], if any partitioning strategy could preserve all possible sub-graph patterns, the partitioning strategy would generate exponential number of sub-graphs. Therefore, it is hypothetically reasonable to believe smaller partitioning size could make Apriori-based graph mining algorithms to work, but many more frequent sub-graph patterns would not be discovered.

Traditional data mining tasks can be performed on the transportation data in their pure transactional forms. However, they only produce limited insights. Consequently, the existing graph mining algorithms need to be enhanced to handle large graph transactions or new graph mining algorithms need to be investigated within the domain of transportation and logistic networks.

## 9 Conclusion: Challenges for Data Mining Research

We have presented some ideas for new challenges in the field of graph mining, based on transportation network transaction data. We have shown two experiments where we reduce this data to a problem that can be addressed by existing graph mining approaches. The first is based on finding structurally similar patterns of origin-destination transactions; the second on temporally similar patterns, or repeated routes. While both are interesting as a first cut, they are perhaps most interesting not for the results produced, but for the challenges to the data mining community.

We believe that the biggest shortcoming of current methods is their inability to handle the temporal aspects of graphs. For useful mining of transportation data, it is critical to incorporate the temporal nature of the links into the mining algorithms. Thus, one of the biggest challenge problems is how to do mining of dynamic graphs, where a dynamic graph is defined as a graph for which an edge / vertex exists only for certain periods of times. A pertinent question is to consider what is a pattern over time. One example is to find frequently repeated connection paths, where the entire path is not connected at any given time instant but adjacent edges and vertices always co-exist. Patterns could also occur spatially or temporally, possibly with an unknown period. Including the temporal aspects adds a whole new dimension to the problem. Note that interesting patterns need not be cycles, but patterns including elements that are not spatio/temporally close are unlikely to be of interest. For example, one could find that every time there is a load from Green Bay to Lafayette, there is also one from Portland to

[3] We also used the synthetic graph generator used in [13] to generate a set of graph transactions with a large number of distinct vertex labels; this produced the same out of memory problems.

Sacramento. However, this is not necessarily very helpful. Some filtering / constraints are needed.

Incorporating the notion of events into a graph is another interesting problem. An event can be many different things. Some examples are: A customer starting to have service failures or weather incidents that cause longer delays or even closure of some roads, or seasonality (including holiday behavior) which brings about known variability. As a first cut, it is quite natural to represent events as a change in the value of a set of nodes and links. Being able to find frequent patterns caused due to the connectivity structure of a graph would help in determining emergent patterns. Analysis of the fallout of temporal/spatial events could lead to figuring out the nature of causality between emergent patterns and a triggering event (i.e., bounce effect). This would be extremely useful to the transportation industry.

The prediction of events itself might be possible if a chain of events occur due to causality and graph structure. This is not necessarily spatial, but there are interesting spatial examples (e.g., balance of flow in/out of a certain market, etc.)

While we only tested with FSG[13], the problems posed by a mix of small and large graphs, and particularly by the presence of very large graphs, may also pose problems for other graph mining algorithms. Further experimentation will be necessary to validate this hypothesis, however expanding graph mining evaluation experiments beyond data similar to molecular structures would be a worthwhile exercise for graph mining researchers. Recent work in finding maximal graph patterns, i.e., ignoring sub-patterns of a frequent patterns, may address this challenge.

Another problem is with determining what makes a graph pattern interesting. Even at high support levels (very high frequency), we found many frequent patterns. However, many of these patterns turn out to be trivial or uninteresting. A variety of metrics have been developed to evaluate the interestingness of association rules [15, 18]. Similar metrics are needed for graph mining.

More interesting are the problems that are not easily reduced to the current notion of graph mining. Interesting features of this data that are not well supported by existing work in graph mining include:

- Graphs are directed. Kuramochi and Karypis point out that FSG could easily be extended to directed graphs, and likewise much of the other work in the field could probably be so extended. Little work has actually been reported in this direction outside the domain of trees.

- Finding patterns within a single graph is interesting. Work is first needed to define what a pattern means within the concept of a single graph; we have given one definition in Section 5. Further definitions, and algorithms that efficiently find *all* such patterns, is an open problem.

- The data is temporal. Concepts such as periodicity in routes, or expectation of changes over time, could be important factors. This is an area where merging existing ideas in mining time-series data could lead to new advances.

- Patterns that appear over a time window are more relevant than those appearing at one instant. Knowing that a cycle from Melbourne to Lafayette to Atlanta and back to Melbourne exists on a single day is less relevant than knowing that the cycle exists over a space of a week. Ideas such as frequent episodes could be useful, although it would be necessary to extend them to recognize that not only must the pattern occur within a time window, but that the transactions composing the pattern must be separated by a *minimum or maximum* time.

These challenges provide ample room for further research in the field of graph mining.

## 10 Acknowledgments

We thank Jim Jeray, Mike Zeimer, Bob Gremley, Rao Panchalavarapu, Vishvesh Oza, Erick Wikum, Ted Gifford and Bradley Utz from Schneider National for discussions on interesting challenge problems and value of results. Fruitful discussions with Profs. Carla Brodley, Ananth Iyer, Reha Uzsoy, and Dr. Richard Cho led to our initial focus on this topic. We also thank Mr. Kuramochi for providing us with the graph generator and the FSG executable, as well as Prof. Washio for providing us with the executable for AGM.

## References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, Sept. 12-15 1994. VLDB.

[2] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Satamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In *Proceedings of the Second SIAM International Conference on Data Mining*, Santiago, Chile, Apr. 11-13 2002. SIAM.

[3] K. Bansal, S. Vadhavkar, and A. Gupta. Neural networks based data mining applications for medical inventory problems. *International Journal of Agile Manufacturing*, 1(2):187–200, 1998.

[4] K. Bansal, S. Vadhavkar, and A. Gupta. Neural networks based forecasting techniques for inventory control applications. *Data Mining and Knowledge Discovery*, 2(1):97–102, 1998.

[5] D. Cook and L. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255, 1994.

[6] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 30–36, New York, New York, Aug. 1998. AAAI Press.

[7] S. Dill, S. R. Kumar, K. S. McCurley, S. Rajagopalan, D. Sivakumar, and A. Tomkins. Self-similarity in the web. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases*, pages 69–78, Roma, Italy, Sept. 11-14 2001. Morgan Kaufmann.

[8] M. R. Garey and D. S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.

[9] L. Holder, D. Cook, and S. Djoko. Substructure discovery in the SUBDUE system. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*, pages 169–180, 1994.

[10] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proceedings of PKDD'00*, pages 13–23, 2000.

[11] The IEA and transport, Feb. 27 2003.

[12] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[13] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 313–320, 2001.

[14] M. Kuramochi and G. Karypis. Discovering frequent geometric subgraphs. In *Proceedings of the International Conference on Data Mining*, pages 258–265, Maebashi City, Japan, Dec. 9-12 2002. IEEE.

[15] C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1):39–68, Jan. 1998.

[16] S. Su, D. J. Cook, and L. B. Holder. Knowledge discovery in molecular biology: Identifying structural regularities in proteins. *Intelligent Data Analysis*, 3:413–436, 1999.

[17] Y. Takahashi, Y. Satoh, and S. Sasaki. Recognition of largest common fragment among a variety of chemical structures. *Analytical Sciences*, 3:23–28, 1987.

[18] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 32–41, Edmonton, Alberta, Canada, 2002.

[19] A. Termier, M.-C. Rousset, and M. Sebag. Treefinder, a first step towards XML data mining. In *Proceedings of the International Conference on Data Mining*, Maebashi City, Japan, Dec. 9-12 2002. IEEE.

[20] N. Vanetik, E. Gudes, and S. Shimony. Computing frequent graph patterns from semistructured data. In *Proceedings of the International Conference on Data Mining*, pages 458–465, Maebashi City, Japan, Dec. 9-12 2002. IEEE.

[21] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Fransisco, Oct. 1999.

[22] E. K. Wong. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25(3):287–304, 1992.

[23] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 721–724, Maebashi City, Japan, Dec. 9-12 2002. IEEE.

[24] M. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, 2002.