# Frequent Sub-graph Mining on Edge Weighted Graphs

Chuntao Jiang, Frans Coenen, Michele Zito

The University of Liverpool
Ashton Building, Ashton Street
Liverpool, L69 3BX, United Kingdom
{c.jiang,coenen,michele}@liv.ac.uk

**Abstract.** Frequent sub-graph mining entails two significant overheads. The first is concerned with candidate set generation. The second with isomorphism checking. These are also issues with respect to other forms of frequent pattern mining but are exacerbated in the context of frequent sub-graph mining. To reduced the search space, and address these twin overheads, a weighted approach to sub-graph mining is proposed. However, a significant issue in weighted sub-graph mining is that the *anti-monotone property*, typically used to control candidate set generation, no longer holds. This paper examines a number of edge weighting schemes; and suggests three strategies for controlling candidate set generation. The three strategies have been incorporated into weighted variations of gSpan: ATW-gSpan, AW-gSpan and UBW-gSpan respectively. A complete evaluation of all three approaches is presented.

**Keywords**: Weighted Transaction Graph Mining, Weighted Frequent Sub-graph Mining, Weighting Schemes.

## 1 Introduction

Graph mining is concerned with the identification of patterns within graph data of various forms. One form of graph mining is frequent sub-graph mining which aims to identify frequently occurring patterns (sub-graphs) across a collection of "small" graphs or within one "large" graph. This paper concentrates on the first (also sometimes referred to as *transaction graph mining*).

Frequent sub-graph mining techniques [3, 5, 6, 8, 11, 12] have parallels with more established frequent pattern mining techniques such as those used in, for example, Association Rule Mining (ARM). Thus, in common with other forms of frequent pattern mining, frequent sub-graph mining entails two significant overheads: candidate set generation and isomorphism checking. However, these overheads are exacerbated because of the nature of graph data. In the case of candidate set generation the potential number of size $K+1$ sub-graphs that can be generated from size $K$ graphs is exponentially greater than in the case of more standard forms of frequent pattern mining. With respect to isomorphism checking, the process of comparing a candidate pattern with the input data to

determine the support (frequency) of the candidate is significantly more complex in the case of frequent sub-graph mining than in more standard forms of frequent pattern mining such as ARM.

The overheads associated with frequent sub-graph mining are compounded when the support threshold is low. The solution advocated in this paper is based on the observation that, for many applications, some edges (nodes) in the input graph set can be considered to be more significant than others. Therefore, sub-graph patterns that include edges (nodes) with high weight values should be considered more important than those with low weight values if they both satisfied the support threshold. This concept is illustrated in this paper by considering a social network mining scenario.

Weighted frequent sub-graph mining advocates the use of *weighted support counts* to identify weighted frequent sub-graphs. Hence, the "computational burden" of sub-graph mining can be considerably alleviated by generating a set of weighted frequent sub-graphs. The concept of edge weightings can be encapsulated in a number of ways (for reasons of clarity only edge weighted graphs are considered in this paper although much of the discussion is equally applicable to to node, or node and edge, weighted graphs).

Regardless of whether edge or node weighting is adopted, a significant issue encountered in weighted sub-graph mining is that the *anti-monotone property*, whereby if a $K$ size sub-graph is not frequent none of its $K+1$ super-graphs will be frequent, typically used to restrict the size of the search space in standard pattern mining, no longer holds if weightings are applied in a naive manner. Thus any proposed weighted sub-graph mining mechanism must either be defined in such a way that the property continues to hold, or an alternative pruning strategy must be adopted.

Three edge weighting schemes are considered in this paper: (i) *Average Total Weighting* (ATW), (ii) *Affinity Weighting* (AW) and (iii) *Utility Based Weighting* (UBW). The three approaches have been incorporated into three weighted variations of the gSpan algorithm (ATW-gSpan, AW-gSpan, and UBW-gSpan).

The rest of this paper is organised as follows. A problem definition overview is presented in Section 2. The proposed edge weighting mechanisms are considered in Section 3. Experiments to evaluate the proposed techniques, and the ensuing results, are presented in Section 4. Some conclusions are presented in Section 5.

## 2 Problem Definition

This section introduces the necessary graph-theoretic and mining definitions. In the context of this paper a graph is defined as a finite structure $G$ formed by a set of nodes $V = \{v_1, v_2, \ldots\}$, a set of edges $E = \{e_1, e_2, \ldots\}$, a set of vertex and edge labels $\mathcal{L}$, and a mapping $\phi_{v/e} : \mathcal{L} \rightarrow V/E$. With respect to the work described here the edge labels are assumed to be numeric so that they can be used in the calculation of relative weightings. Depending on the particular application, edges will be either *undirected* pairs over $V$, or *directed* (ordered) pairs.

Let $T = \{G_1, G_2, \cdots, G_t\}$ be a collection of (transaction) graphs. The support set of $g$ is defined as $\delta_T(g) = \{t | g \subseteq G_t\}$, i.e. the set of transaction graphs where $g$ is a sub-graph of $G_t$. The cardinality of the support set, $|\delta_T(g)|$ then defines the support of $g$ with respect to $T$.

**Definition 1.** *Given a database $T$, a graph $g$, and a minimum support $\tau \in (0, 1]$, the graph $g$ is said to be* frequent *(in $T$) if $|\delta_T(g)| \geq \tau \times t$. The* frequent sub-graph mining problem *is thus to find all the frequent sub-graphs in $T$.*

The focus of this paper is on edge weighted graphs. Therefore, the graphs in $T$ are assumed to have weights associated with their edges. Let $W_T$ be a weighting function that assigns a weight to any sub-graph $g$. The *weighted support* of $g$ with respect to $T$, $wsup_T(g)$, is then:

$$wsup_T(g) = W_T(g) \times |\delta_T(g)|. \tag{1}$$

Note that the function of $W_T(g)$ needn't be a number between zero and one. By defining the weighting function, $W_T(g)$, in an appropriate manner it is possible to ensure that the anti-monotone property holds; otherwise other method, such as some heuristic based pruning technique, is required to limit the search space.

## 3 Graph Weighting Mechanisms

Most research work in frequent sub-graph mining [5,6,8,11] assumes each discovered frequent sub-graph is equally important. A lot of redundant and repetitive frequent patterns may therefore exist in the final result. If the size of the graph set is substantial and the minimum support threshold is very low, a typical frequent sub-graph mining task can often not be completed within a fixed period of time due to the exponential complexity of the search space. If we put emphasis on differentiating each discovered frequent sub-graph according to its importance, either as definded by the user or derived from the application domain, the computational complexity can be reduced without compromising the effectiveness of the frequent pattern discovery process. However, when a weighting scheme is integrated into the process of graph mining in a naive manner, the well-known *anti-monotone property*, which is used frequently to reduce the search space, may no longer be satisfied. Two strategies can be identified to address this dilemma: (a) adopt an interestingness measure which does satisfy the property; (b) ignore the property and adopt some alternative heuristic to reduce the computational overhead incurred by not satisfying the property.

In the context of weighted frequent sub-graph mining, weightings associated with a sub-graph pattern $g$ can be defined in a number of manners. Three approaches are introduced in this paper: (i) Average Total Weighting (ATW), (ii) Affinity Weighting (AW), (iii) Utility Based Weighting (UBW). The first two approaches satisfy the anti-monotone property while the last one adopts an alternative pruning heuristic. The last two approaches employ two parameters to control the mining result while the first one uses one parameter only. Each approach is discussed in more detail below. Each approach is discussed in further detail in the following three subsections.

### 3.1 Average Total Weighting (ATW)

In the ATW approach inspired by the work [10], the weight for a sub-graph $g$ is calculated by dividing the sum of the average weights in graphs that contain $g$ with the sum of the average weights across the entire data set $T$. Thus:

**Definition 2.** *Given an edge weighted graph $g$ with edge weights $\{w_1, w_2, \cdots, w_k\}$, the average weight associated with $g$ is defined as $W_{avg}(g) = \frac{\sum_{i=1}^{k} w_i}{k}$.*

Where $w_i$ can be user defined or calculated by some weighting methods.

**Definition 3.** *Given a set of graphs $T = \{G_1, G_2, \cdots, G_t\}$, the total weight of this set of graphs is defined as $W_{sum}(T) = \sum_{i=1}^{t} W_{avg}(G_i)$.*

**Definition 4.** *Given an arbitrary sub-graph $g$ with its support set $\delta_T(g)$, the weight function of $g$ with respect to $T$, $W_T(g)$, is defined as*

$$W_T(g) = \frac{\sum_{G_i \in \delta_T(g)} W_{avg}(G_i)}{W_{sum}(T)} \tag{2}$$

**Definition 5.** *A sub-graph $g$ is weighted frequent with respect to $T$, if $|\delta(g)| \times W_T(g) \geq \tau \times t$, where $0 < \tau \leq 1$ is a minimum support threshold.*

From the above it can be easily inferred that the function $W_T(g)$, as defined by Equation 2, satisfies the anti-monotone property. Therefore, if a $k$-candidate is not frequent, then any of its $(k+1)$-supersets can be safely pruned from this branch in the lattice of candidates during the $k+1$ candidate generation process. It should be noted, however, that the approach will tend to bias large transaction graphs over smaller transaction graphs, thus is best applied to graph sets where the individual graphs are of a similar size.

### 3.2 Affinity Weighting (AW)

The Affinity Weighting (AW) approach is founded on two elements to restrict the growth of the search space: (i) a graph distance measure, and (ii) a weighting ratio. For a sub-graph $g$ to be frequent both must be greater than specified user thresholds. The graph distance measure is calculated using an appropriately defined support weighting function, $W_T(g)$. This is defined as follows. Let $g$ be a candidate pattern for a database $T = \{G_1, G_2, \cdots, G_t\}$. In the context of AW we define:

$$W_T(g) = \frac{1}{|V(g)|} \sum_{G_i \in \delta_T(g)} \frac{|V(G_i)| - |V(g)|}{|V(G_i)|}. \tag{3}$$

Where $V(G_i)$ is the set of vertices in transaction graph $G_i$ and $V(g)$ is the set of vertices in the sub-graph $g$. Observe that $W_T(g)$ satisfies:

$$W_T(g) = \frac{|\delta_T(g)|}{|V(g)|} - \sum_{G_i \in \delta_T(g)} \frac{1}{|V(G_i)|} \tag{4}$$

It should be noted that adding nodes to $g$ can only reduce the value of the above expression because the support($|\delta_T(g)|$) cannot be increased; the sum contains as many terms as $|\delta_T(g)|$ and each of these cannot be larger than $1/|V(g)|$. Thus $W_T(g)$ as defined above, insures that the weighted support of $g$ is non-increasing (i.e. anti-monotone) in $|V(g)|$.

The graph distance measure is directed at the number of nodes contained in a graph, the weighting ratio concerned with the edge weights (which are assumed to reflex numeric values). The weighting ratio of an edge-weighted graph $g$ is a function $c(g)$ returning a value between zero and one which is decreasing in the number of edges of $g$. Given an edge weighted sub-graph $g$ with edge weights $W = \{w_1, w_2, \cdots, w_k\}$ the weighting ratio function which is similar to [13], $c(g)$, is defined as follows:

$$c(g) = \frac{MIN_{w_i \in W}\{w_i\}}{MAX_{w_j \in W}\{w_j\}}. \tag{5}$$

**Definition 6.** *An edge-weighted graph $g$ is a weighted frequent (i.e. weighted affinity) pattern within a data set $T = \{G_1, G_2, \cdots, G_t\}$, with respect to a support threshold $\tau > 0$ and weighting ratio threshold $\gamma \in [0, 1]$, if the following two conditions (C1 and C2) are satisfied:*

$$\textbf{(C1)}\ wsup_T(g) \geq \tau \times t, \qquad \textit{and} \qquad \textbf{(C2)}\ c(g) \geq \gamma.$$

Definition 6 leads to an alternative pruning strategy which, may be used as part of any frequent sub-graph mining algorithms. During the candidate selection phase, the mining will keep track of the weighted support and weighting ratio of all candidates and discard all those candidates that do not satisfy at least one of **(C1)** and **(C2)**.

### 3.3 Utility Based Weighting (UBW)

The previous two approaches both satisfy the anti-monotone property. In this section an alternative weighting scheme which does not hold the property is proposed. The Utility Based Weighting (UBW) scheme is influenced by ideas suggested in [1, 2]. As in the case of AW scheme, the UBW scheme is founded on two elements: (i) weighted support and (ii) the share (SH) of a sub-graph. Thus:

**Definition 7.** *Given a sub-graph $g$ with edges $E(g) = \{e_1, e_2, \cdots, e_k\}$. For each $e_i \in E(g)$, two vertices connecting $e_i$ are $v_1$ and $v_2$. Their associated support sets (the graphs in $T$ where they appear) are given as $\delta_T(v_1)$ and $\delta_T(v_2)$. The Jaccard similarity coefficient between the two vertices is defined as $jC(e_i) = |\delta_T(v_1) \cap \delta_T(v_2)|/|\delta_T(v_1) \cup \delta_T(v_2)|$. The weighting function of $g$, $W_T(g)$, is then defined as*

$$W_T(g) = \frac{1}{\sum_{e_i \in E(g)} jC(e_i)} \tag{6}$$

*From the above it is clear that $W_T(g)$ satisfies the anti-monotone property. From Section 2 the weighted support is given by $wsup_T(g) = W_T(g) \times |\delta_T(g)|$.*

**Definition 8.** *Given an edge weighted graph set $T = (G_1, \ldots, G_t)$ with edge weights $\{w_1, w_2, \cdots, w_k\}$ for each transaction graph $G_j$ and a sub-graph $g$. Let $g \subseteq G_j$, the weight of $g$ denoted as $W(g, G_j)$, is the sum of the weights of the edges which occurred in $G_j$. That is, $W(g, G_j) = \sum_{e_i \in g, g \subseteq G_j} w_i$. The total weight of $T$, denoted as $TW(T)$, represents the sum of edge weights in $T$, where $TW(T) = \sum_{G_j \in T} \sum_{e_i \in G_j} w_i$. The total weight of $\delta_T(g)$, is defined as $TW(\delta_T(g)) = \sum_{G_j \in \delta_T(g)} \sum_{e_i \in G_j} w_i$.*

**Definition 9.** *The graph weight of $g$ with respect to $T$, denoted as $GW(g)$, is the sum of the weight of the $g$ in each transaction graph $G_j \in \delta_T(g)$. That is, $GW(g) = \sum_{G_j \in \delta_T(g)} W(g, G_j)$.*

**Definition 10.** *The share of a sub-graph $g$, denoted as $SH(g)$, is the ratio of the graph weight of $g$ with respect to $T$ to the total weight of $T$. Thus:*

$$SH(g) = \frac{GW(g)}{TW(T)} \tag{7}$$

*Given a share threshold $\lambda$, a sub-graph $g$ is SH-frequent if $SH(g) \geq \lambda$; otherwise, $g$ is SH-infrequent.*

**Theorem 1.** *Given a $T = (G_1, \ldots, G_t)$, a sub-graph $g$, and a threshold $\lambda$, if $TW(\delta_T(g)) < \lambda \times TW(T)$, all super-graphs of $g$ are SH-infrequent.*

*Proof.* Let $h$ be an arbitrary super-graph of $g$. Clearly, $GW(h) \leq TW(\delta_T(h)) \leq TW(\delta_T(g))$. If $TW(\delta_T(g)) < \lambda \times TW(T)$ holds, $GW(h) < \lambda \times TW(T)$. That is, $SH(h) = GW(h)/TW(T) < \lambda$. Therefore, $h$ is SH-infrequent. □

By Theorem 1, if $TW(\delta_T(g)) < \lambda \times TW(T)$, all super-graphs of $g$ and $g$ are SH-infrequent and can be pruned; otherwise, $g$ is a candidate sub-graph.

**Definition 11.** *An edge-weighted graph $g$ is a weighted frequent pattern for a graph set $T = (G_1, \ldots, G_t)$ with respect to a support threshold $\tau > 0$ and share threshold $\lambda \in (0, 1]$ if the following two conditions are satisfied.*

**(D1)** $wsup_T(g) \geq \tau \times t$, *and* **(D2)** $SH(g) \geq \lambda$.

## 4 Experiments and Results

This section describes a sequence of experiments designed to:

(i) Demonstrate that the proposed weighting schemes can more efficiently generate frequent sub-graphs than without using weightings. In many cases, as will be demonstrated, use of the weighting schemes allows frequent sub-graphs to be identified where this would not be possible using an unweighted approach because of this computational overhead the latter would entail.

**Table 1.** CTS graph set statistics

| | Norfolk | Cornwall | GB |
|---|---|---|---|
| # graphs | 53 | 53 | 53 |
| Max # edges | 77 | 412 | 30107 |
| Average # edges | 54 | 262 | 23055 |
| Max # nodes | 99 | 409 | 23660 |
| Average # nodes | 70 | 284 | 18749 |
| node label count | 614 | 2195 | 81153 |
| Edge label count | 6 | 12 | 46 |

(ii) Compare and contrast the three proposed weighted sub-graph mining techniques.

The experiments were conducted using a projection of the cattle movement database in operation in Great Britain (GB). This application domain is described in Section 4.1. The original gSpan algorithm available to the authors could not process directed graphs with self cycles. Therefore an extended gSpan algorithm (extGspan), which can process directed graphs with self cycles, was implemented in order to compare the proposed weighted approaches with the un-weighted case. Results from the experiments are presented in Sub-sections 4.2 and 4.3.

### 4.1 The Cattle Tracking System Database

For the experiments the Cattle Tracking System (CTS) database, in operation in GB, was used. This was provided by the Department for the Environment, Food and Rural Affairs (DEFRA) from the Rapid Analysis and Detection of Animal Risk (RADAR) project[1]. The database provides a record of cattle movements. Each record includes information such as the sender and receiver location IDs, animal ID, animal breed, etc. Three distinct transaction graph datasets were extracted from the CTS database such that nodes represented cattle location (farms, markets, slaughter houses, etc) and edges the movement of cattle between locations (the edges are directed by the direction of the cattle movement). Transaction graph sets for all of Great Britain (GB), and two areas within GB (Norfolk and Cornwall) were extracted. Edges were annotated with a weighting, indicating the number of cattles moved, and a label, indicating the type of movement (e.g. farmToFarm, farmToMarket, etc). For each data set the data from 1 January 2005 to 31 December 2005 was selected and divided into 7-day "episodes" due to the 6-day movement restriction [9] that applies to farms in GB. Statistics for each of the data sets are given in Table 1. Note that the $GB$ data set is significantly larger than the $Cornwall$[2], which in turn was larger than the $Norfolk$ data set. It should also be noted that all the transaction graphs feature directed edges and self cycles.

[1] http://www.defra.gov.uk/foodfarm/farmanimal/diseases/vetsurveillance/radar/project.htm
[2] Cornwall is a county in the SW of GB known for its substantial dairy herds

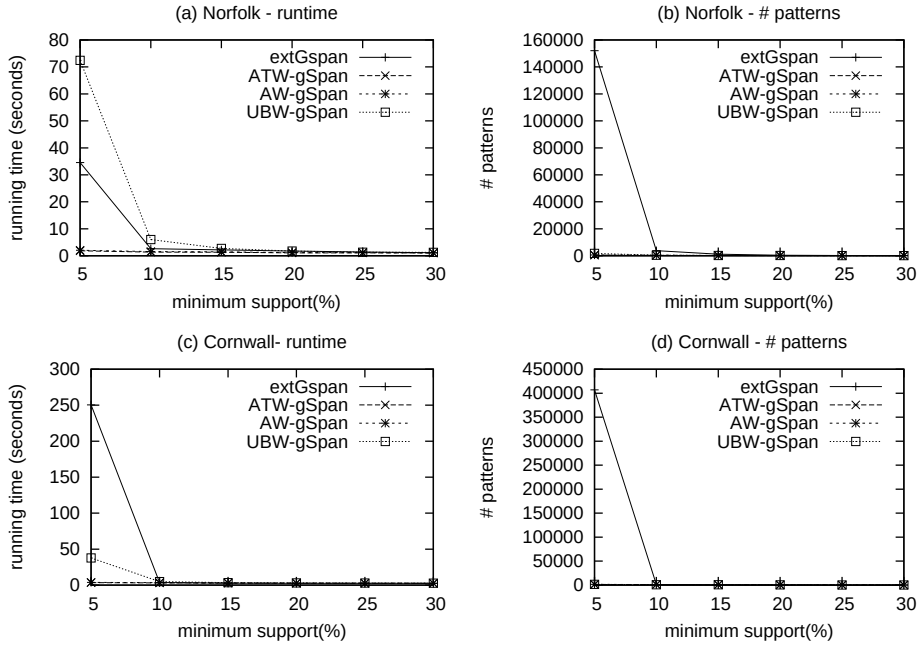


**Fig. 1.** Performance comparison of weighting schemes vs. extGspan on $Norfolk$ and $Cornwall$ data sets (using a range of support values from 5% to 30%)

### 4.2 Comparison Between Weighted and Non-Weighted Approaches

In this subsection the proposed weighting schemes (ATW-gSpan, AW-gSpan, and UBW-gSpan) are compared with the extended gSpan algorithm in terms of efficiency (runtime and the number of frequent sub-graphs generated). For AW-gSpan, $\gamma = 0.6$ was chosen as the weighgting ratio threshold, and $\lambda = 8\%$ was used as the share threshold for UBW-gSpan. The judstification for these $\gamma$ and $\lambda$ values is given in Sub-section 4.3 below.

Figure 1 shows the performance of the weighting schemes and extGspan on the $Norfolk$ and $Cornwall$ data sets (recall that extGspan does not make any use of weightings). It can be clearly seen from the figure that all four algorithms display a similar behaviour when the support value is between 10% to 30%, however the number of patterns generated by the extGspan algorithm increase abruptly when the support value is decreased to below 10%. From Figure 1 it can be observed that: (i) significantly more frequent sub-graphs (at support threshold below 10%) are found using the non-weighted extGspan algorithm than using any of the weighting schemes, indicating the advantages offered using the weighted approaches, (ii) the ATW and AW schemes run faster than the UBW scheme, this is because the pruning technique adopted by UBW schem is not strong enough compared with the anti-monotone based pruning methods used by ATW and AW schemes.

Experiments (not shown) using extGspan and the $GB$ data set failed to produce any results (because of memory errors) unless the support thresholod

was set to 30% or above, a threshold at which only one node size sub-graph are discovered. Thus it was not possible to conduct any meaningful comparison between the weighted frequent sub-graph mining algorithms and a non-weighted approach using the $GB$ data set.

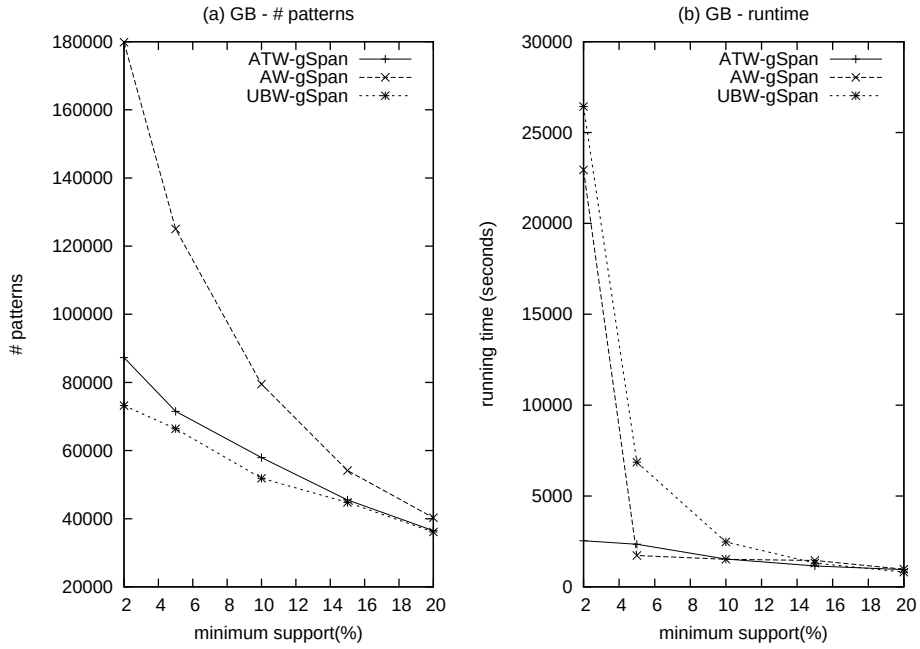


**Fig. 2.** Performance comparison of three weighting schemes using the $GB$ data set

### 4.3 Comparison of Weighting Schemes

In this subsection the three proposed weighting schemes are compared with one another using the large $GB$ dataset. As above, $\gamma$ was initially set to 0.6 and $\lambda$ to 8% for use with AW-gSpan and UBW-gSpan algorithms. Figure 2 shows the performance of the weighting schemes on the $GB$ dataset. In Figure 2 (a), each curve depicts the number of patterns generated against the minimum support value used. From the figure it can be seen that UBW-gSpan produces the least number of patterns while AW-gSpan produces the most. Figure 2 (b) indicates the "run time" for the approaches using the same sequence of support threshold values. From the figure it can be seen that UBW-gSpan is the most "expensive", indicating that the cost of finding a minimum number of patterns is higher compared to the other two mechanisms. ATW-gSpan is the most economical. Reference to Figures 1(a) and (b) confirm these results. UBW-gSpan is also expensive with respect to the $Norflok$ and $Cornwall$ data sets. In fact inspection

of Figures 1(a) indicates that UBW-gSpan is more expensive than applying extGspan in the case of the$Norfolk$ data indicating that the cost of reducing the number of patterns is high when using UBW-gSpan. Although it should be noted that with respect to the $GB$ data set extGspan was unable to process this data set at all (using realistic support thresholds). It is interesting to note in Figure 2 (b) that as the support threshold is reduced the effect on run-time is much smaller for ATW-gSpan than the other two weighting schemes. More generally, from Figure 2, it can be seen that (as might be expected) runtime increases significantly as the support threshold is reduced.

Figure 3 displays the effect on performance of different values for the weighting ratio threshold ($\gamma$) used in conjunction with AW-gSpan, and the share threshold ($\lambda$) used with UBW-gSpan, for a range of support threshold values from 4% to 12%. From Figures 3 (a) and (c) it can be seen that the run time increased as the $\gamma$ value is decreased, while a marginal increase in the number of patterns is witnessed. With respect to Figures 3 (b) and (d) it can be seen that the run time increases as the $\lambda$ value is decreased, while a small corresponding increase in the number of identified patterns is witnessed. However, increasing the $\lambda$ value beyond 8% seems to have very little effect on the number of patterns. Overall it was found that a $\gamma$ value of 0.6 and a $\lambda$ value of 0.8% was the most appropriate.

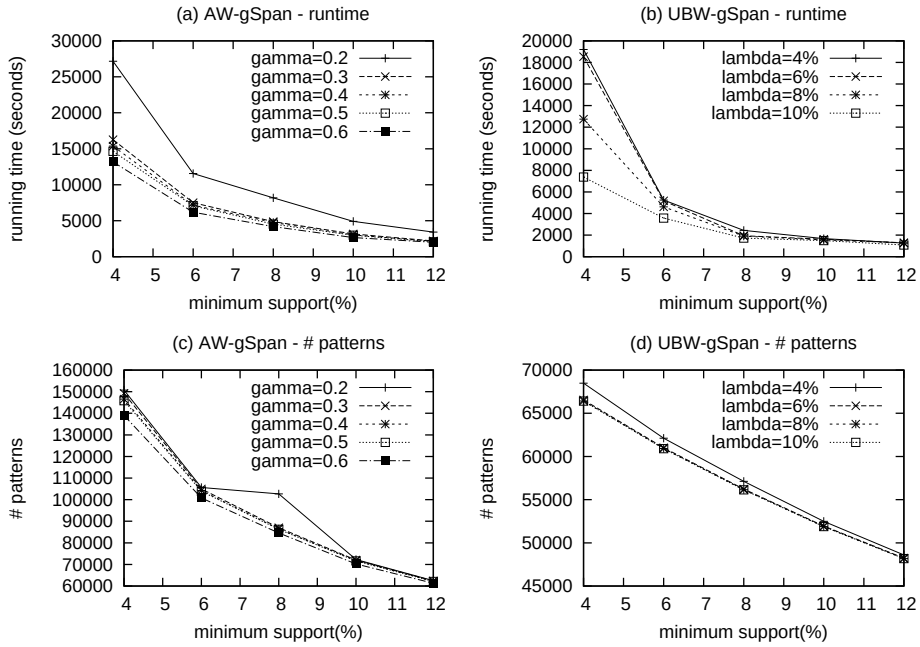


**Fig. 3.** Analysis of the Performance of AW-gSpan and UBW-gSpan using different $\gamma$ and $\lambda$ values

### 4.4 Quality of Results

The above experiments indicate that the proposed weighting approaches can be successfully applied so that frequent sub-graphs can be identified in large collections of graphs (such as those extracted from the CTS database) which could not otherwise be mined using more conventional graph mining approaches. The proposed weighting mechanisms operate by identifying the most "significant" edges. The question that remains is then to ask "are we finding the right frequent sub-graphs?". To answer this question the research team applied the weighting techniques to a number of classification problems. Two data sets were used, an MRI scan data set and a text mining data set where the scans and documents had been processed into a graph representation and labelled. Weighted graph mining techniques were then applied to the graph sets to produce collections of frequent sub-graphs. These sub-graphs were then interpreted as features in a feature space and used to represent the individual records using a standard feature vector representation (where each element represents a frequent sub-graph). Standard classification algorithms were then applied. The results generated were comparable with results obtained using alternative, more conventional, classification approaches thus indicating that the "right sub-graphs" had been identified. Space limitations prevent a full presentation and discussion of these results in this paper, however interested readers can refer to [4] and [7] for reports on the MRI scan and text mining experiments respectively.

## 5 Conclusions

This paper has proposed a solution to frequent sub-graph mining where the size of the input data is such that standard graph mining algorithms (such as gSpan) are unable to derive any appropriate results because of the computational overheads involved. Three weighting mechanisms are proposed (ATW-gSpan, AW-gSpan, and UBW-gSpan) designed to reduce to overall search space by identifying the most relevant sub-graphs. The weighting schemes assume edge weightings, but similar techniques may be applied with respect to nodes. Experiments comparing the operation of the weighting schemes to a non-weighted version of gSpan indicate that many fewer patterns are derived. The research team have established that the reduced pattern set are the "right" pattern set by applying the results using classification scenarios. The reported experiments indicate that UBW-gSpan finds the least number of patterns will requiring the largest amount of run-time. ATW-gSpan provides the best compromise, a limited number of patterns found in reasonable time (especially at low support threshold values). Experiments were also conducted with respect to the most suitable $\gamma$ and $\lambda$ to be used with respect to AW-gSpan and UBW-gSpan respectively. Overall it was found that a $\gamma$ value of 0.6 and a $\lambda$ value of 0.8% was the most appropriate.

## 6 Acknowledgements

We would like to thank the Department for the Environment, Food and Rural Affairs (DEFRA) for providing us the data. We are grateful to Dr. Christian Setzkorn from the Faculty of Veterinery Science, University of Liverpool for extracting the simplified form of the data and Mrs Puteri Nor Ellyza Nohuddin for assisting us to get the data.

## References

1. Barber,B., Hamilton,H.J.: Extracting Share Frequent Itemsets with Infrequent Subsets. In:*Journal of Data Mining and Knowledge Discovery*, V7, pp. 153-185, (2003)
2. Carter,C.L., Hamilton,H.J., and Cercone,N.: Share based Measures for Itemsets. In:*Komorowski,H.J., Zytkow,J.M.(eds.):1st European Conference on the Principles of Data Mining and Knowledge Discovery.*, Lecture Notes in Computer Science, Vol. 1263, Spring-Verlag, pp. 14-24, (1997)
3. Cook,D.J.,Holder,L.B.:Substructure Discovery Using Minimum Description Length and Background Knowledge. In:*Journal of Artificial Intelligenc Research*, 1:231-255, (1994)
4. Elsayed, A., Coenen, F., Jiang, C., Garca-Fiana, M. ana Sluming, V.: Corpus Callosum MR Image Classification. To appear in the Journal of Knowledge Based Systems, (2010).
5. Huan,J., Wang,W., and Prins,J.: Efficient Mining of Frequent Subgraph in the Presence of Isomorphism. In:*Proceedings of the 2003 International Conference on Data Mining(ICDM'03)*, 2003.
6. Inokuchi,A.,Washio,T. and Motoda,H.: An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data. In: *Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, (2000)
7. Jiang, C., Coenen, F., Sanderson, R. and Zito, M.: Text Classification using Graph Mining-Based Feature Extraction. To appear in the Journal of Knowledge Based Systems, (2010).
8. Kuramochi,M. and Karypis,G.: Frequent Subgraph Discovery. In:*Proceedings of IEEE International Conference on Data Mining*, (2001)
9. Robinson,S.E., and Christley,R.M.: Identifying Temporal Variation in Reported Births, Deaths and Movements of Cattle in Britain. In:*Journal of BMC Verterinary Research*, DOI:10.1186/1746-6148-2-11, (2006)
10. Tao,F., Murtagh,F. and Farid,M.: Weighted Association Rule Mining using Weighted Support and Significance Framework. In: *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD 2003)*, pp. 661-666, Washington DC, USA, (2003)
11. Yan,X. and Han,J.:gSpan: Graph-based Substructure Pattern Mining. In:*Proceedings of 2002 International Conference on Data Mining* (2002)
12. Yan,X. and Han,J.:CloseGraph: Mining Closed Frequent Graph Patterns. In:*Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Pages: 286-295, Washington D.C., USA (2003)
13. Yun,U.:WIS: Weighted Interesting Sequential Pattern Mining with a Similar Level of Support and/or Weight. In:*ETRI Journal*, Vol. 29, No. 3, Pages: 336-352, (2007)