Finding the Most Descriptive Substructures in Graphs with Numeric Labels

Michael Davis, Weiru Liu, and Paul Miller

Centre for Secure Information Technologies (CSIT),
School of Electronics, Electrical Engineering and Computer Science,
Queen's University, Belfast, United Kingdom
{mdavis05, w.liu}@qub.ac.uk, p.miller@ecit.qub.ac.uk

Abstract. Many graph datasets are labelled with numeric attributes. Frequent substructure discovery algorithms usually ignore these attributes; in this paper we show that they can be used to improve discrimination and search performance. Our thesis is that the most descriptive substructures are those which are normative both in terms of their structure and in terms of their numeric values. We propose an outlier-detection step during substructure discovery to prune anomalous vertices and edges, which gives more weight to the most descriptive substructures. Our experiments on a real-world access control database returns similar substructures to Subdue with 30% fewer graph isomorphism tests.

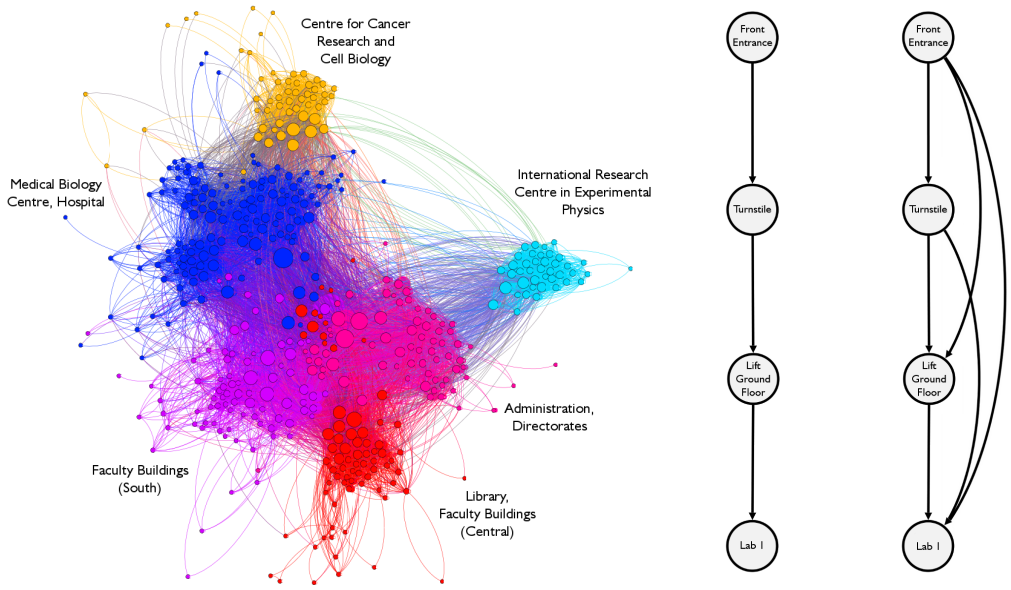
Keywords: graph mining, frequent substructure discovery, numeric attributes, outlier detection

1 Introduction

A common task in graph mining is to discover frequently-occurring substructures for concept learning, clustering or anomaly detection. In this paper, we consider how numeric attributes can be combined with structural data, to refine the search for the most descriptive substructures.

Frequent substructures are defined as those which pass some minimum support threshold [7, 10, 14] or in information-theoretic terms, as the patterns which can be used to maximally compress the input graph [3]. To count the frequency of each pattern, discovery algorithms compare subgraphs for identity, or Graph Isomorphism (GI). As GI is computationally complex to decide [5], discovery algorithms attempt to minimise the number of GI tests by sorting candidate substructures by their canonical labels [7, 10], by organising discovered subgraphs into spanning trees [1, 14] or by performing a heuristic search [3].

Substructure discovery algorithms typically operate on unweighted, labelled graphs, where labels represent discrete attributes of vertices or edges. However, many graph datasets also contain numeric labels or weights, representing attributes such as size, distance, time, frequency or amount. We propose that the best substructures are not only frequent, but also have the most normative numeric attributes. By excluding substructures containing numeric anomalies, we



(a) Single large graph (b) Simple transaction (c) Forward edges

Fig. 1: Graph of Access Control Transactions on a University Campus

prune unlikely candidates early, focussing computational resources on the most descriptive patterns. In our experiments, we were able to discover the most descriptive substructures with a 30% reduction in the number of GI tests required, compared to Subdue.

One application of our research is to discover "suspicious" behaviour in secure buildings, such as airports, hospitals and power stations. Our experimental data is from the building access control system for a university campus, represented by the graph in Fig. 1a. Vertices represent door sensors and directed edges represent movements between pairs of sensors. The density of transactions is higher in areas with greater security requirements, *viz.* laboratories for laser, radiation and medical research. As we are interested in paths taken by individuals through the network, we reorganised the graph as a transaction database, where each subgraph represents the movement of an individual within a given day (Fig. 1b–1c). We score each subgraph based on its structural elements and its numeric timing values [4]. Our method shows greater discrimination than scoring based on frequent substructures without numeric attributes.

This paper is organised as follows: Sect. 2 describes the main approaches to substructure discovery in graphs. In Sect. 3, we explain our method of using numeric anomalies to narrow the search for normative substructures. Sect. 4 outlines our experimental method and datasets; the results are presented in Sect. 5. In Sect. 6 we present our conclusions.

2 Related Work

Frequent substructure discovery algorithms attempt to find the subgraphs which occur most frequently in a graph database or in a single large graph. Early approaches were based on Apriori-style itemset mining: AGM [7] and FSG [10] generate candidate substructures by growing them one vertex or one edge at a time, respectively. Frequent substructures are those which exceed a specified minimum support threshold. The main weakness is that candidate generation is expensive, as canonical labels must be calculated for a large number of redundant candidates.

gSpan [14] avoids candidate generation. Canonical labels are determined by the minimum representation of vertex orderings as discovered by a Depth-First Search (DFS). These labels are organised into a hierarchical spanning tree. Frequent structures are discovered by traversing this tree, checking for substructures which exceed minimum support. CloseGraph [15] and SPIN [6] improve on gSpan by mining only "closed" or "maximal" frequent subgraphs, *i.e.* frequent substructures which are not part of any larger frequent substructure. [1] generalises the canonical form found in gSpan and demonstrates that canonical labels based on Breadth-First Search (BFS) are equally valid.

Subdue [3] represents another class of frequent substructure discovery algorithm, based on information theory. Rather than searching for substructures with minimum support, Subdue looks for the substructures which can be used to best compress the input graph based on the Minimum Description Length (MDL) principle. Complexity is managed with a heuristic: candidate substructures are discovered using a greedy beam search (a limited-length queue of the best few patterns which have been found so far). The disadvantage of the greedy search strategy is that some interesting patterns could be missed.

AGM, FSG and gSpan are designed to work on graph transaction databases. Subdue has fewer restrictions on the input graphs: it can also search efficiently in single large graphs.

Numeric labels The above algorithms are designed to work on graphs with discrete labels. There have been a few efforts to incorporate numeric attributes into frequent substructure discovery. [8] extends gSpan by including edge weights into the support calculation, pruning low-weighted substructures from the search. This approach cannot be generalised to other numeric attributes, where a high numeric value does not necessarily indicate greater significance. [12] discusses three ways in which Subdue can handle numeric attributes: exact match (treat numeric attributes as if they are discrete); tolerance match (two numeric values are considered equal if their difference is less than some threshold); and difference match (two numeric values are considered equal if they are drawn from the same Gaussian PDF). The authors propose binning numeric values into discrete ranges; two values are considered equal if they fall into the same range. As each numeric attribute is treated separately, correlations between attributes are ignored during discovery.

3 Frequent Substructure Discovery and Numeric Attributes

It is desirable to prune infrequent substructures as early as possible in the discovery process, to reduce expensive and redundant search operations. We define the most descriptive substructures as those which are normative both in terms of their structure and in terms of their numeric attributes. The corollary is that vertices or edges containing numeric outliers are abnormal and can therefore be pruned early in the discovery process.

We define a graph as having an arbitrary number of discrete and numeric labels on its vertices and edges:

Definition 1. *Let L be the set of graph labels, comprising vertex labels L_V and edge labels L_E. L is partitioned into discrete labels L^D and numeric labels L^N, $L^D \cap L^N = \emptyset$. Thus $L = L_V \cup L_E = L^D \cup L^N$. Let A^D be the set of discrete attribute values and $A^N \subset \mathbb{R}$ be the set of numeric attribute values.*

Definition 2. *A labelled graph G is a tuple $\langle V, E, \mathcal{L}_V, \mathcal{L}_E \rangle$. V is a set of vertices and E is a set of edges: $E \subseteq \{\langle v, w \rangle : v, w \in V \times V\}$. If the tuple $\langle v, w \rangle$ is ordered, the edge is directed, otherwise it is undirected.*

Definition 3. *The label-to-value mapping function for vertices is denoted as:*

$$\begin{aligned} \mathcal{L}_V : V \times (L_V \cap L^D) &\rightarrow A^D \\ V \times (L_V \cap L^N) &\rightarrow A^N \end{aligned}$$

The label-to-value mapping function for edges $\mathcal{L}_E$ is denoted in a similar manner:

$$\begin{aligned} \mathcal{L}_E : E \times (L_E \cap L^D) &\rightarrow A^D \\ E \times (L_E \cap L^N) &\rightarrow A^N \end{aligned}$$

For a weighted graph, the edge weight function $\mathcal{W}(e)$ is treated as a special case of numeric edge attributes: $\forall e \in E : \mathcal{W}(e) = \mathcal{L}_E(e, \text{“}weight\text{”})$.

Vertices and edges will be considered equal by the graph isomorphism test if they share the same discrete labels, and their numeric attributes are “normal”. We determine normality by means of a numeric outlier function:

Definition 4. *We define a numeric outlier function $\mathcal{O}$ on a dataset D as:*

$$\mathcal{O} : D \rightarrow \mathbb{R} \qquad \forall d \in D : \mathcal{O}(d) = \begin{cases} q_0 & \text{if } d \text{ is “normal” w.r.t. } D \\ q & \text{otherwise} \end{cases}$$

where q_0 is some constant value and $q \neq q_0$ is a value measuring the degree of outlierness.

The value of q_0 and the range of $\mathcal{O}$ will depend on the specific choice of outlier detection function. In Sect. 4, we use Local Outlier Factors (LOF) [2]. LOF is well-suited to unsupervised learning, as it makes no assumptions about the underlying distribution of the data and can cope with clusters of different sizes and densities. LOF calculates an outlier score based on both the distance from and the relative density of the local neighbourhood. Samples belonging to a dense cluster or deep within a sparse cluster have $\mathrm{LOF}(d) \cong 1$. Outliers have LOF values several times larger. Thus LOF satisfies the property given in Def. 4: $\mathrm{LOF}(d) \cong 1$ for normal values of d and $\mathrm{LOF}(d) \gg 1$ for anomalous values. For a more detailed discussion of the choice of outlier function, see our previous work [4] and the comments at the end of Sect. 6.

In order to calculate numeric outliers over the graph, we partition the vertices and edges into similar disjoint sets or equivalence classes:

$$V = \bigcup_i V_i \qquad E = \bigcup_i E_i$$

Definition 5. *Two vertices are said to be equivalent if their discrete attributes are the same:*

$$\forall v \in V_i \; \forall w \in V_i \; \forall l \in (L_V \cap L^D) : \mathcal{L}_V(v, l) = \mathcal{L}_V(w, l)$$

Two edges $\langle v, w\rangle$ and $\langle x, y\rangle$ are equivalent if their discrete attributes are the same and v is equivalent to x and w is equivalent to y. In the case of an undirected graph, $\langle v, w\rangle$ equivalent to $\langle x, y\rangle \implies \langle v, w\rangle$ equivalent to $\langle y, x\rangle$.

Note that our approach is similar to the algorithms discussed in Sect. 2, in that frequent substructures will not contain vertices or edges with infrequent values for discrete attributes, by definition. However, we may sometimes wish to treat a discrete (enumerated) attribute as numeric: see the discussion of "Day of Week" in Sect. 4 for an example.

Using Defs. 4–5, we can calculate outlier values on the numeric attributes of vertices and edges across all equivalence classes, $\mathcal{O}(\mathcal{L}_V(V_i, L^N))$ and $\mathcal{O}(\mathcal{L}_E(E_i, L^N))$. This now brings us to the definition of each vertex and edge as normal or anomalous:

Definition 6. *A vertex $v \in V_i$ is normal if $\mathcal{O}(\mathcal{L}_V(v, L^N)) \cong q_0$, anomalous otherwise. An edge $e \in E_i$ is normal if $\mathcal{O}(\mathcal{L}_E(e, L^N)) \cong q_0$, anomalous otherwise.*

Alg. 1 shows how our approach can be integrated with the substructure discovery process for Subdue [3]. An initial set of parent substructures is initialised to each of the disjoint single-vertex instance sets, $P = \{V_1, \ldots, V_n\}$. Anomalous vertices (as defined by Def. 6) are excluded. We execute Alg. 1 once for each member of P: **for all** $I \in P$ **do** ExpandSubstructures(G, I). All possible expansions of the instances $g_i \in I$ are evaluated. In line 5, candidate substructures with anomalous vertices and edges are pruned. The algorithm returns a set of child substructures C. At the end of the first iteration, the substructures in C are evaluated using the MDL principle as usual [3]. For the next iteration, P is initialised to C.

Algorithm 1 (Subdue) ExpandSubstructures(G, I)

Require: Graph $G = \langle V, E, \mathcal{L}_V, \mathcal{L}_E \rangle$, Parent instances $I = \{g_1, \ldots, g_n\}$, $\forall i \forall j : g_i$ is isomorphic to g_j

```
Child substructures C ← ∅
for all g_i ∈ I do
    for all v ∈ V_{g_i} do                          ▷ Find all possible edge expansions
        for all e = ⟨v, w⟩ : e ∈ E and e ∉ E_{g_i} do
            if e is normal and w is normal by Def. 6 then        ▷ Prune anomalies
                Create child substructure as copy of g_i
                if w ∉ V_child then add w to child
                Add e to child
                Find I_c ∈ C : child is isomorphic to g_c ∈ I_c
                if not exist I_c then add I_c ← ∅ to C
                Add new child instance to I_c ∈ C
return C
```

Alg. 1 can be adapted to FSG-style substructure discovery by applying it to the initial step of generating all 1-subgraphs and 2-subgraphs. In this case, each $|I_i| : I_i \in C$ is evaluated against the minimum support parameter. The approach can be applied to gSpan and similar algorithms which use spanning trees, by incorporating the anomaly score into the calculation of minimum support.

4 Experiments

Our experiments measured the effect of pruning numeric outliers during substructure discovery on: the number of graph isomorphism tests required; the computation time for substructure discovery; and the effect on the accuracy and meaningfulness of the discovered substructures.

Synthetic Datasets Our experimental setup included Erdős-Rényi random graphs and R-MAT random graphs with up to 10,000 vertices. For the R-MAT graphs, we added edges with mean degree 4 and probabilities that an edge is placed in one of the four quadrants of the graph as $\langle 0.57, 0.19, 0.19, 0.05 \rangle$, to ensure that the random graphs exhibited clustering/community properties. The baseline version of each random graph structure had no numeric attributes. We created multiple copies of each structure with between 1 and 100 numeric attributes on each edge (we could equally have chosen to add attributes to vertices or to both vertices and edges). For "normal" edges, each attribute was selected from a PDF normally distributed across equivalent edges to create a multivariate Gaussian feature vector. We perturbed 1% of edges to simulate the fact that these edges were produced by a different process.

Access Control System Dataset Fig. 1a shows the patterns of movement of students and staff on a large university campus. Each vertex represents a door

sensor; directed edges represent movements between pairs of sensors. There are approximately 6,500 users, 800 sensors and 900,000 transactions. We are interested in finding patterns representing "suspicious" behaviour, particularly in high-security areas such as laboratories for laser, radiation and medical research [4].

As we are interested in paths taken by individuals through the network, we organised the graph as a transaction database. Each graph transaction represents the movement of an individual within a given 24-hour period. Fig. 1b shows an example of a typical graph transaction. Sometimes users may fail to swipe in at a particular sensor (for example, if someone holds open the lift door). This creates missing edges in the graph. We compensated for these missing edges by including forward edges from each sensor to all subsequent sensors visited by the user (Fig. 1c).

Next, we added numeric attributes to the edges. The attributes for a single path segment in the database are shown in Fig. 2. Absolute time is the time of day (seconds since midnight) when the user presented their ID card to a door sensor at the end of a path segment. Elapsed time is the difference in seconds between the absolute time at the current sensor and the absolute time at the previous sensor. Day of Week (DoW) could have been represented as a discrete attribute, but by combining it with the other attributes in multi-dimensional space, the effect is to cluster different patterns of behaviour on different days. Weekend patterns are quite different from weekday patterns, perhaps representing the movements of security staff or cleaners as well as weekend workers. The experimental results show that we achieved greater discrimination by combining DoW and elapsed time than by treating either attribute in isolation.

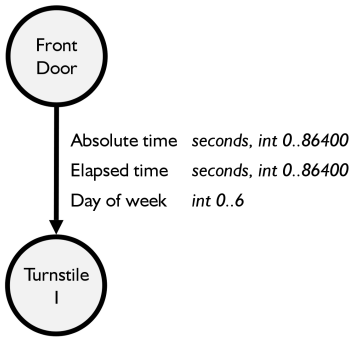
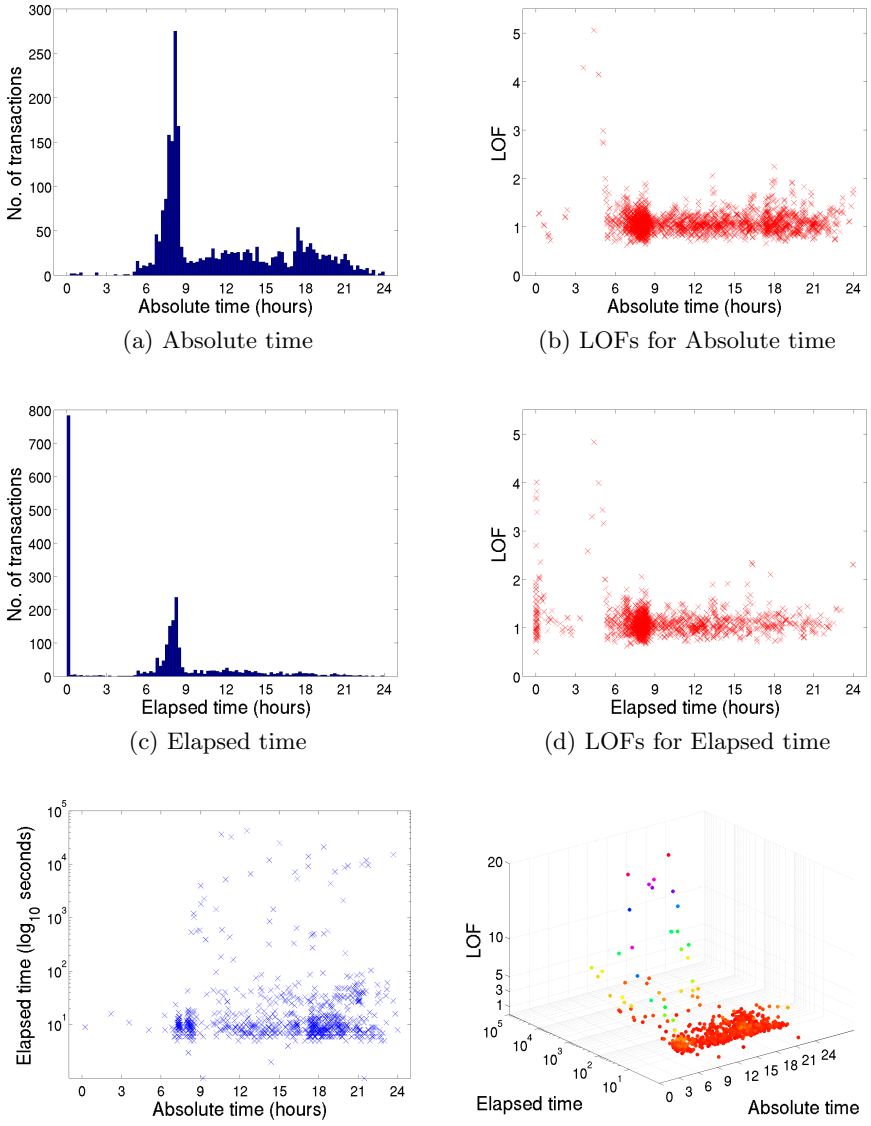


Fig. 2: Numeric edge labels

The distribution of each attribute cannot be represented by a single Gaussian: the PDF is not uniform between attributes, nor even between the same attribute across non-equivalent edges. Usually, there will be a mixture of distributions representing different underlying processes: some people work in a lab and stay for many hours; others may go in simply to speak to a colleague for a few minutes. The advantage of a density-based approach to outlier detection is that we do not have to make any assumptions about the underlying distribution of the data.

Fig. 3 shows the distributions of the numeric attributes from Fig. 2. Fig. 3a shows absolute time. The most common process is that people enter the building between 8–9am, but there are at least two other processes whereby people enter later in the day. Fig. 3c shows that the primary distribution for elapsed time on this path segment is power-law: most people pass between the two sensors in a matter of seconds. But there is a secondary process at around 7–9 hours.



(a) Absolute time (b) LOFs for Absolute time

(c) Elapsed time (d) LOFs for Elapsed time

(e) Absolute time vs. Elapsed time (f) LOFs for Absolute vs. Elapsed time

Fig. 3: Examples of the distribution of numeric attributes across a specific edge in the graph. The figures on the left show the empirical distributions of two numeric attributes (one and two dimensions). The figures on the right show the corresponding distribution of Local Outlier Factors (LOF).

We can discriminate between normal and anomalous values for either of these attributes: Figs. 3b and 3d show the corresponding LOF scores. Normal timing values produce a LOF score close to 1 (Def. 4). Anomalous timing values produce LOF scores in the range 2–5.

However, we can gain more information by considering the correlations between attributes. A two-dimensional feature vector combining absolute and elapsed time is shown in Fig. 3e. Fig. 3f shows the corresponding LOF scores. By calculating LOFs across a multidimensional feature vector containing all the numeric attributes, we can detect outliers with greater discrimination than if we considered each attribute in isolation.

5 Results

Synthetic Datasets The experiments on random single large graphs were repeated 10 times for each combination of no. of vertices, no. of unique vertex labels and no. of numeric attributes, and the results averaged. The results for graphs with 500 and 1,000 vertices and 26 unique labels are shown in Fig. 4. On synthetic graphs, we measured an average 66% reduction in the number of GI tests required when using numeric outliers to prune the graph, compared to Subdue without numeric attributes. The number of GI tests was always directly proportional to discovery time.

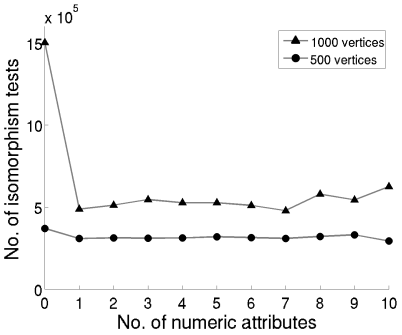
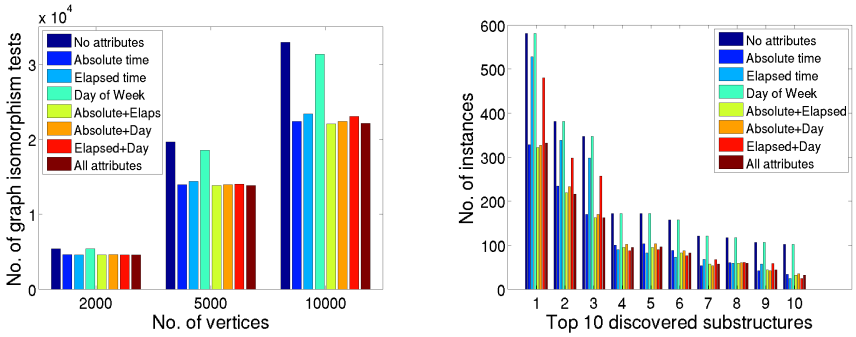


Fig. 4: Effect of numeric attributes on frequent substructure discovery (R-MAT random graphs)

The time taken to calculate the numeric outliers was trivial compared to the time to discover substructures: in the order of 0.2 seconds for a graph with 10,000 vertices and 100 numeric attributes. The complexity of LOF is polynomial, but the size of the numeric datasets across each edge remain small in relation to the size of the graph. In graphs up to 10,000 vertices, the power-law distribution ensures that the majority of edge datasets have no more than a few hundred elements, so LOF's $O(n^2m)$ complexity is acceptable.

We found that graphs with 10,000 vertices and 26 unique labels were not tractable without using numeric attributes, as there were millions of instances of each pattern, requiring more memory to process than was available in our experimental setup. However, we were able to process the 10,000 vertex graphs with numeric attributes, as we had to consider an order of magnitude fewer instances at each iteration. This suggests that our method could be useful when processing Very Large Graphs.



(a) Number of GI tests required (b) Frequency of top 10 substructures

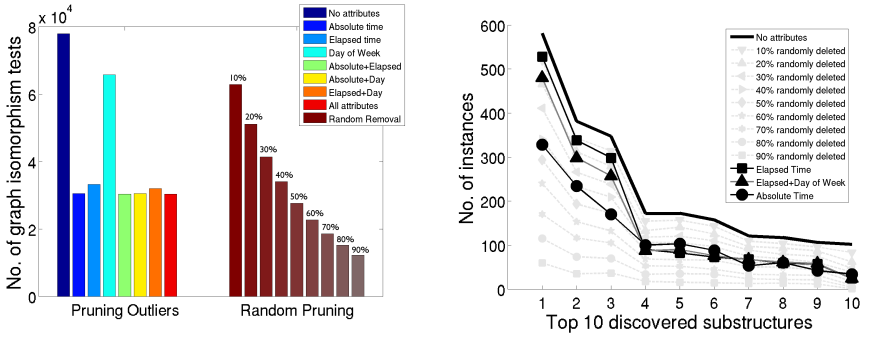
Fig. 5: Effect of pruning numeric anomalies on frequent substructure discovery

Access Control System Dataset The results on the real-world graph transaction database are shown in Fig. 5. Fig. 5a shows the number of GI tests for datasets of 2,000–10,000 vertices. In the real-world data, numeric attributes vary in their ability to discriminate between normal and anomalous patterns, but combining attributes gives the best performance overall. Absolute and elapsed time are both good discriminators, but day of week is very poor. However, if we combine day of week with elapsed time, we get a slightly better result than using elapsed time on its own; and the best results were achieved by combining all three attributes. The benefit of our approach increased with increasing size of database, as we were able to prune more anomalous substructures. In the dataset with 10,000 vertices, representing around three months' transactions, we achieved a performance improvement of around 30%.

Next, we wanted to validate that the discovered substructures are meaningful. Fig. 5b compares the ten best substructures discovered by Subdue (with no numeric attributes) to the substructures discovered when we added attributes. Our approach discovered the same substructures as Subdue, but fewer instances of each. The relative order of the top ten substructures was changed slightly. Where there is a large difference in relative frequency (*e.g.* 1^{st}–3^{rd} substructures), the ordering was unchanged: these substructures are robust against the removal of anomalous edges. In cases where the relative frequencies were very similar, the order was sometimes transposed (*e.g.* 4^{th} and 5^{th} substructures exchanged places). This is because greater weight is given to substructures with normal numeric values.

To investigate this effect further, we compared our method of pruning anomalous edges to random removal of edges from the graph. We conducted experiments where we randomly deleted 10%–90% of the edges in the graph before searching for frequent substructures. The results are shown in Fig. 6.

Fig. 6a shows the effect on performance. It is necessary to remove around 45% of the graph in order to reduce the number of GI tests by a similar amount as our approach.



(a) Number of GI tests required (b) Frequency of top 10 substructures

Fig. 6: Effect of randomly removing graph edges on frequent substructure discovery

Fig. 6b shows the frequency counts of the top 10 substructures. Substructures with edges randomly removed are shown in grey; substructures discovered by our approach are superimposed in black. Randomly removing edges increases the entropy of the graph: the shape of the curve becomes flatter as more of the graph is removed. Substructure 1 is quite distinct; even with 90% edge removal, it remains the most descriptive pattern. Substructures 2–3 are also quite robust; discrimination is lost at around 70% edge removal. However, discrimination between substructures 4–10 diminishes after 20% edge removal and by 50% edge removal, the order is random. These results show that randomly deleting graph edges does not preserve the meaningfulness of the output.

6 Conclusions

In this paper, we have presented a method of using numeric outliers to refine the search for frequent substructures in graphs. Our thesis is that the "best" substructures are those which are not only the most frequent, but which are also normative in terms of their numeric attributes. In our experiments, we find similar substructures to Subdue, with around 30% fewer graph isomorphism tests. Where discovered substructures are of similar frequency, we are better able to discriminate between them, because we give greater weight to substructures with normal numeric attributes.

Future Work We used LOF to calculate numeric outlier scores for each edge in the graph, as it fulfils the conditions of Def. 4. In a sparse graph, the size of the numeric datasets across each edge remain small in relation to the size of the graph, so LOF's $O(n^2m)$ complexity is acceptable. This would not be the case for very large or very dense graphs. This could be addressed by replacing LOF with an approximation algorithm such as aLOCI [11] or PINN [13], which has sub-quadratic complexity, $O(mn \log n)$.

Our experiments tested graphs with up to 100 numeric attributes. As we project into a very high dimensional space, the numeric feature vectors become very sparsely distributed and LOF's ability to discriminate between normal and anomalous values is diminished. However, not all attributes are of equal importance to all clusters. For high-dimensional data, we could amend our approach to detect numeric anomalies in subspaces rather than in full space by choosing only locally-relevant attributes on which to calculate our outlier score [9].

A variant of Subdue allows inexact substructure matching. A match cost parameter specifies how many vertices or edges can be different before we consider two substructures to be distinct. It would be possible to extend our approach to inexact matching by increasing the tolerance for numeric outliers (for example, only discarding outlier vertices or edges when their LOF score exceeds 2) or by only pruning substructures when some percentage of its vertices or edges are outliers.

References

1. Borgelt, C.: Canonical forms for frequent graph mining. In: 30th Annual Conf. German Classification Society (GfKl 2006). pp. 337–349. Springer (2006)
2. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: Identifying density-based local outliers. SIGMOD Rec. 29(2), 93–104 (2000)
3. Cook, D.J., Holder, L.B.: Graph-based data mining. IEEE Intelligent Systems 15, 32–41 (March 2000)
4. Davis, M., Liu, W., Miller, P., Redpath, G.: Detecting anomalies in graphs with numeric labels. In: CIKM 2011. pp. 1197–1202. ACM (2011)
5. Fortin, S.: The graph isomorphism problem. Tech. rep., Univ. of Alberta (1996)
6. Huan, J., Wang, W., Prins, J., Yang, J.: SPIN: Mining maximal frequent subgraphs from graph databases. In: KDD 2004. pp. 581–586. ACM (2004)
7. Inokuchi, A., Washio, T., Motoda, H.: An Apriori-based algorithm for mining frequent substructures from graph data. In: PKDD 2000. pp. 13–23. Springer (2000)
8. Jiang, C., Coenen, F., Zito, M.: Frequent sub-graph mining on edge weighted graphs. In: DaWaC 2010. Springer (2010)
9. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Transactions on Knowledge Discovery in Data 3(1), 1:1–1:58 (March 2009)
10. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: ICDM 2001. pp. 313–320. IEEE (2001)
11. Papadimitriou, S., Kitagawa, H., Gibbons, P., Faloutsos, C.: Loci: fast outlier detection using the local correlation integral. In: Data Engineering, 2003. Proceedings. 19th International Conference on. pp. 315 – 326 (march 2003)
12. Romero, O.E., Holder, L.B., Gonzalez, J.A.: A new approach for handling numeric ranges for graph-based knowledge discovery. In: DMIN 2011. pp. 40–46 (2011)
13. de Vries, T., Chawla, S., Houle, M.: Finding local anomalies in very high dimensional space. In: ICDM 2010. pp. 128–137. IEEE (2010)
14. Yan, X., Han, J.: gSpan: Graph-based substructure pattern mining. In: ICDM 2002. pp. 721–724. IEEE (2002)
15. Yan, X., Han, J.: CloseGraph: Mining closed frequent graph patterns. In: KDD 2003. pp. 286–295. ACM (2003)