
# Vertex Unique Labelled Subgraph Mining

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

**Wen Yu**

September 2015

# Contents

# Illustrations

## List of Figures

## List of Tables

# Abstract

This thesis proposes the novel concept of Vertex Unique Labelled Subgraph (VULS) mining with respect to the field of graph-based knowledge discovery (or graph mining). The objective of the research is to investigate the benefits that the concept of VULS can offer in the context of vertex classification. A VULS is a subgraph with a particular structure and edge labelling that has a unique vertex labelling associated with it within a given (set of) host graph(s). VULS can describe highly discriminative and significant local geometries each with a particular associated vertex label pattern. This knowledge can then be used to predict vertex labels in "unseen" graphs (graphs with edge labels, but without vertex labels). Thus this research is directed at identifying (mining) VULS, of various forms, that "best" serve to both capture effectively graph information, while at the same time allowing for the generation of effective vertex label predictors (classifiers). To this end, four VULS classifiers are proposed, directed at mining four different kinds of VULS: (i) complete, (ii) minimal, (iii) frequent and (iv) minimal frequent. The thesis describes and discusses each of these in detail including, in each case, the theoretical definition and algorithms with respect to VULS identification and prediction. A full evaluation of each of the VULS categories is also presented.

VULS has wide applicability in areas where the domain of interest can be represented in the form of some sort of a graph. The evaluation was primarily directed at predicting a form of deformation, known as springback, that occurs in the Asymmetric Incremental Sheet Forming (AISF) manufacturing process. For the evaluation two flat-topped, square-based, pyramid shapes were used. Each pyramid had been manufactured twice using Steel and twice using Titanium.

The utilization of VULS was also explored by applying the VULS concept to the field of satellite image interpretation. Satellite data describing two villages located in a rural part of the Ethiopian hinterland were used for this purpose. In each case the ground surface was represented in a similar manner to the way that AISF sheet metal surfaces were represented, with the $z$ dimension describing the grey scale value. The idea here was to predict vertex labels describing ground type.

As will become apparent, from the work presented in this thesis, the VULS concept is well suited to the task of 3D surface classification with respect to AISF and satellite imagery. The thesis demonstrates that the use of frequent VULS (rather than the other forms of VULS considered) produces more efficient results in the AISF sheet metal forming application domain, whilst the use of minimal VULS provided promising results

in the context of the satellite image interpretation domain. The reported evaluation also indicates that a sound foundation has been established for future work on more general VULS based vertex classification.

# Acknowledgements

This thesis would have not been completed if not for the help I have received from a number of people whose contribution to my research deserves a special mention. It is a pleasure to convey my gratitude to them all in this modest acknowledgement.

Of all the fantastic and fabulous people involved, my greatest debt of gratitude must go to my first supervisor, Professor Frans Coenen, who has given me the chance to work under his supervision. He has provided me with invaluable assistance, guidance, constant support, encouragement, constructive criticism, research ideas and excellent advice throughout the past four years, which has made it one of the most enjoyable and unforgettable experience of my life. During my PhD study, he also gave me many opportunities to publish papers and attend conferences which have helped me to broaden my horizons. I learned a lot from these conferences and workshops, these experiences remain very valuable to me. Through his extraordinary experience, he has taught me not only to be a good PhD student but also to be a good researcher and an intellectual person. He has enlightened me through his inspiration and endless efforts on how to explain and present academic work simply and clearly. He is the best supervisor any one could hope for, and more. It has been a pleasure and an honour to have been supervised by him. He was the perfect resource which inspired me and enriched my experience making me the person that I am today.

I would also like to express my great gratitude to my second supervisor, Dr. Michele Zito, who provided valuable insights and reviewed many pieces of my writings. I also thank him for providing many constructive suggestions and valuable comments concerning my research work.

I am also thankful to my friends and colleagues Subhieh El Salhi and Kwankamon (Kwan) Dittakan. They have been extremely helpful in providing advice on numerous occasions. I would like to thank Subhieh El Salhi for her valuable collaboration, especially with respect to the pre-processing of the raw sheet metal data sets used in my research. I would also like to extend great appreciation to Kwan for supplying the satellite image data sets used in the research presented here. I can not expect better friends and colleagues than Subhieh and Kwan. We encouraged each other during our PhD study. I do not believe I could have got this far without such good friends. I am also obliged to many other friends who provided encouragement, either directly or indirectly; they are: Maduka Attamah, Eric Schneider, Muhammad Tufail, Esra'a Shdaifat and Jeffery Raphael.

The Department of Computer Science at the University of Liverpool has been an excellent place in which to conduct research, and all members of staff and colleagues have been encouraging and helpful whenever needed. In particular, I would also like to extend my gratitude to my "PhD Advisors": Dr. Russell Martin, Professor Paul Dunne and Dr. David Grossi for providing me with assistance, suggestions and constructive feedback at various times. I would also like to thank The Department of Computer Science at the University of Liverpool for providing me with sufficient financial assistance to attend a number of conferences/workshops and seminars throughout my study years. I am grateful for this assistance.

I would also like to extend my thanks to the Tecnalia Corporation (Spain) and the IBF institute of metal forming (Germany) for providing the before and after sheet metal forming data that I have used extensively throughout my research.

Fundamental to my being able to conduct this research was the financial support of the GuangZhou government in China, so I also would like to convey thanks to the Overseas Study Program of Guangzhou Elite Project (GEP) in China for their financial support, they have given me the opportunity to pursue my dream of studying overseas. Without their support I would have not been able to finish my research and write this thesis.

My deepest and most grateful thanks also go to my family and friends in China. Their support, trust, encouragement and thoughtfulness have been priceless to me. I am grateful to Professor Sheng Yi Jiang and Dr. Yanbo (Justin) Wang for their constant support and encouragement while in the UK, without whom the completion of my Ph.D. would not have been possible to accomplish.

Finally, I would also like to acknowledge my family and friends for all their love, motivation and support that has helped me to conduct my PhD studies. Particular thanks go to my parents A Ming Yu and Fu Ying Deng. Their love provided me with endless inspiration and was the driving force that allowed me to complete my PhD.

# Abbreviations

| | |
|---|---|
| k-NN | k-Nearest Neighbour. |
| VULS | Vertex Unique Labelled Subgraph(s). |
| compVULS | Set of complete VULS. |
| minVULS | Set of Minimal VULS. |
| freqVULS | Set of Frequent VULS. |
| minFreqVULS | Set of minimal Frequent VULS. |
| BMV | Backward-Match-Voting algorithm. |
| 3D | Three Dimensional. |
| AUC | Area Under the receiver operating Curve. |
| SD | Standard Deviation. |
| DM | Data Mining. |
| FTM | Frequent subTree Mining. |
| FSM | Frequent Subgraph Mining. |
| DCP | Downward Closure Property (A graph can only be frequent if all of its subgraphs are also frequent). |
| DFS | Depth First Search. |
| BFS | Breadth First Search. |
| DT | Decision Tree. |
| KDD | Knowledge Discovery in Databases. |
| RGB | The Red, Green and Blue colour model. |
| TCV | Ten-fold Cross Validation. |
| AISF | Asymmetric Incremental Sheet Forming. |
| CAD | Computer Aided Design. |
| CAM | Computer Aided Manufacturing. |
| $G_{train}$ | An input training graph with vertex and edge labels. |
| $max$ | The maximum size of labelled subgraphs such as VULS. |
| $G_k$ | A collection of $k$-edge subgraphs. |
| $c$ | A candidate VULS. |
| $S$ | A list of potential labels for the vertices of $c$. |
| $\mathcal{U}$ | A set of labelled subgraphs such as VULS. |
| $\|x\|$ | The cardinality of a set $x$. |

# Chapter 1

# Introduction

## 1.1 Overview

Data Mining (DM) is the process of extracting implicit, previously unknown, and potentially useful information from large amounts of data [95]. DM is an element in the Knowledge Discovery in Data (KDD) process [74, 75]. The data that data miners wish to mine comes in many different forms including: images, graphs, text and so on. Therefore the field of data mining includes sub-fields such as image mining, graph mining, and text mining. The work described in this thesis is concerned with graph mining. More specifically the work presented in this thesis proposes the concept of Vertex Unique Labelled Subgraph (VULS) mining; the identification and extraction of subgraphs (with specific configurations and edge labelings from a single graph) that have a unique vertex labelling associated with them. In other words, given a subgraph $g$ with a specific structure and edge labelling, but no vertex labelling, and one single graph $G$ with labelled edges and vertices. If wherever $g$ occurs in $G$ it always has the same vertex labelling, then $g$ is a VULS. The utility of VULS, as will be demonstrated later in this thesis, is that they can be used to label vertices in previously "unseen" vertex unlabelled graph. A simple example is given in Figure 1.1 so as to facilitate a better understanding of the concept of VULS. With reference to input graph $G$ in Figure 1.1, subgraph 1 is a VULS since the vertex labelling associated with the specific configuration and edge labelling in the third column is unique in the context of $G$. Subgraph 2 is not a VULS since there are two possible vertex labelings (so not unique). It is important to note when considering whether a subgraph is a VULS or not that it is the vertex labelling in relation to configuration and edge labelling that needs to be unique, not the vertex labels in isolation. A VULS can include labels that appear in other VULS and other subgraphs, it is the relationship between the vertex labels that needs to be unique for a VULS to exist. The lower limit for the size of a VULS is one edge. The upper limit is the size of the input graph, in fact the entire input graph will be a VULS (although with little utility). In practice, as will become apparent later in this thesis, an upper limit is placed on the size of a VULS to ensure their utility. Further detail concerning the VULS concept are presented in Chapters 2 and 4 later in this thesis.

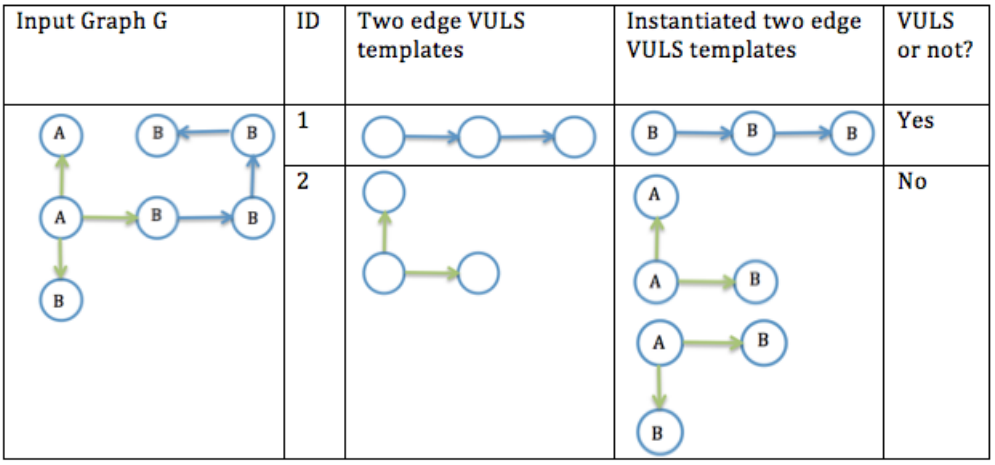



FIGURE 1.1: Example of VULS.

The essence of graph mining is the extraction of useful knowledge from graph represented data. There has been a substantial amount of research effort directed at many aspects of graph mining. This research work can be loosely defined in terms of the following categorisation: (i) frequent subgraph mining [136], (ii) optimal graph pattern mining [204], (iii) correlated graph pattern mining [191], (iv) graph pattern summarization [208], (v) approximate graph pattern mining [240], (vi) graph classification [211], (vii) graph clustering [185], (viii) graph indexing [231, 232], (ix) graph searching [188] and (x) vertex classification [80]. The work presented in this thesis falls into the category of vertex classification. However, to the best knowledge of the author, there has been no comparable work directed at the concept of VULS mining, or the usage of VULS for vertex classification, as presented in this thesis.

Graphs are a powerful mechanism for representing structured data, for example graph vertices can correspond to objects and the edges to relationships or interactions between those objects. Two examples of the usage of graphs to represent protein networks are given in Figure 1.2 [210]. In the figures[1] the vertices represent the proteins and the edges represent interactions among proteins. Generally speaking, the vertices and edges in graphs may be labelled or unlabelled. The edges may be directed or undirected. Graphs may be cyclic or acyclic. A frequently encountered type of graph structure is the Directed Acyclic Graph (DAG) structure [35, 72]. Another particular kind of graph structure, and that of specific relevance with respect to the work presented in this thesis, is the grid graph [92, 145].

Graph mining has been applied to a great variety of domains. Reported examples include: (i) chemical informatics [183], (ii) bioinformatics [54, 179], (iii) video indexing [34, 105], (iv) text retrieval [32, 162, 195], (v) Web mining, XML document mining [27, 30, 33, 38], (vi) face recognition [234] and (vii) Telecommunication and computer network analysis [19, 60, 71, 91].

[1] http://www.liacs.nl/ erwin/dbdm2009/GraphMining.pdf

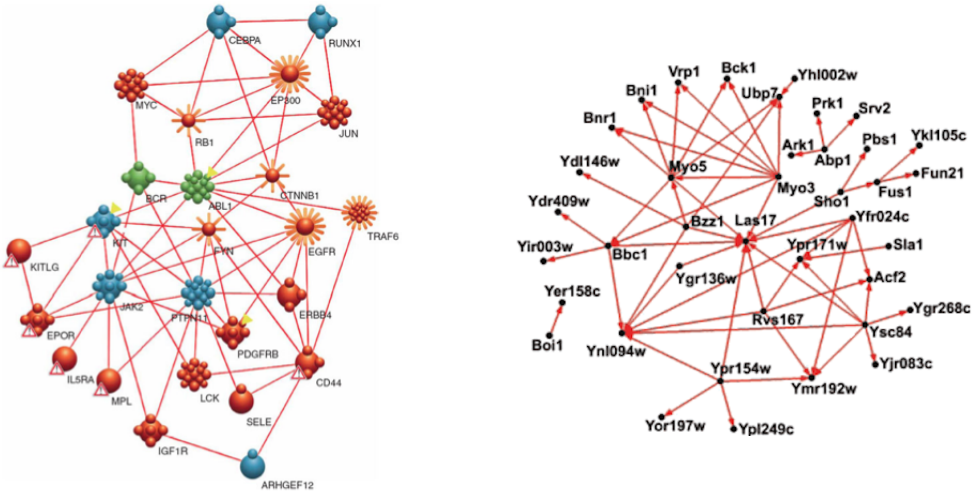



FIGURE 1.2: Graph examples of protein networks [210]

The application domain at which the research presented in this thesis is directed at three Dimensional (3D) surface analysis (interpretation). 3D surface data occurs in the context of many environments. Obvious examples are applications that use map data such as geological studies [141, 154]; we might wish to predict (say) that a certain region within some given map features a particular form of geology. A less immediate example is image analysis, however images can clearly be viewed as 3D surfaces if we consider the third dimension to be "grey scale" intensity. Other applications where 3D surfaces are of significance include manufacturing processes where 3D parts are produced. One example of the latter, and that which is central to motivation for the work presented in this thesis, is sheet metal forming ([77]). The significance of 3D surface analysis, in the context of graphs is that 3D surfaces can be represented in terms of a grid, which in turn can be represented in terms of grid graph where each grid cell centre point is a vertex and each edge represents an immediate neighbourhood relationship linking adjacent grid centre points (vertices). Vertices in such grid graphs may then be labelled, for example with (say) geological labels (as suggested above) or ground surface texture labels. The central idea presented in this thesis is that given a vertex labelled grid graph, VULS mining can be applied to identify a set of VULS that can be used to predict the vertex labelling contained in previously unseen grid graphs (provided they have been drawn from the same application domain).

The rest of this introductory chapter is organised as follows. A more detailed discussion, than that presented above, of the motivation for the work described is presented in Section 1.2. Section 1.3 presents the formulation of the main research question to be addressed by the thesis and also lists the related issues to be addressed to provide an answer to the research question. The research methodology adopted is presented in Section 1.4, whilst the research contributions of the work are presented in Section 1.5. Section 1.6 presents an overview of the organisation of the rest of this thesis. This is followed by a review of the published work to date, resulting from the research described,

in Section 1.7. Finally, this chapter is concluded with a summary presented in Section 1.8.

## 1.2 Motivation

As noted in the previous section the focus for the work is grid graph vertex classification with application to 3D surface analysis and more particularly the sheet steel metal forming application domain. This section elaborates on the motivation for this focus.

The primary motivation for the work presented was a desire to provide a solution to an open problem in sheet metal forming whereby the produced shape is not the desired shape due to distortions introduced during the manufacturing process. In the context of the sheet metal forming motivation for the work presented in this thesis is therefore a demand for accurate and well formed sheet metal components in a variety of industries (such as the automotive and aircraft manufacturing industries). To this end there are a number of manufacturing process that can be adopted. One such process, and that used for evaluation purposes with respect to the work described in this thesis, is Asymmetric Incremental Sheet Forming (AISF). In AISF the metal sheet from which the desired component is to be manufactured is clamped into a "blankholder", a forming tool then follows a predefined tool path to "push out" a desired shape [121]. The main advantage of AISF, over alternative sheet metal forming processes, is that of cost reduction [88, 193] (it doesn't require heating). However, a major limitation of techniques such as AISF is that, as a result of applying the process, deformations called Springback are introduced whereby the produced shape is not the same as the desired shape. Springback is defined as the elastic deformation that occurs in a produced shape as a result of the application of a sheet metal forming process; this deformation only becomes apparent when the manufactured piece is unclamped. In other words, the produced shape differs from the desired shape. Springback is a complex physical phenomenon that is related to the local geometry of the shape to be manufactured. Essentially the shape to be manufactured can be viewed as a 3D surface which in turn can be represented in terms of a grid graph with edges representing slope (the $\delta z$ value between adjacent grid center points). If the nature of the springback associated with individual grid graph vertices (grid cells) can be predicted then some form of mitigation can be applied; an idea first proposed in [65–67, 130]. The VULS concept proposed in this thesis therefore seems idealy suited to providing a solution to the sheet metal forming springback prediction problem. Given a manufactured part with known springback, a grid graph can be formulated with each vertex labelled with a springback value. VULS mining can then be applied and the result applied to the definition of new shapes to be manufactured.

Other than the application domain motivation described above the work presented in this thesis was also motivated by the general need for supervised learning techniques that can be effectively utilised to predict vertex labels in "unseen" graphs. More specifically the desire to investigate an entirely novel approach to vertex label classification.

Current work on vertex classification is directed at exploiting the topology of the graphs considered [17]. Current work is also typically not founded on graph mining techniques (as in the case of the proposed VULS technique). For example in [189, 242], a clustering approach is used, whilst in [86] a probabilistic Bayesian network model is applied and in [56] Markov random walks are used. To the best knowledge of the author the above sets the work described in this thesis apart from other existing work.

In summary the work presented in this thesis was motivated by the following:

1. A desire to address a real world problem (in the domain of sheet metal forming).
2. The need for more effective vertex classification methods directed at graph representations with a focus on grid graphs.
3. The opportunity to research an aspect of graph mining that has not previously received attention (to the best knowledge of the author).

## 1.3 Research Question and Related Issues

Given the motivations presented in the previous section, the main research question to be addressed by this thesis is:

**"How best can the proposed VULS mining be conducted so as to achieve effective vertex classification?"**

To provide an answer to this research question we also address the resolution of the following subsidiary technical research questions:

1. **What is the most appropriate mechanism for identifying VULS?** Although the fundamental idea of VULS mining seems clear, the practicalities of VULS mining, because the idea was entirely novel, remained a subject for detailed investigation.
2. **Can efficiency gains be realised by mining some subset of the complete set of VULS?** Graph mining, of all forms, is known to be computationally expensive [108, 225]; typically graph mining requires a substantial amount of isomorphism testing [158]. Thus instead of identifying the complete set of VULS, we can attempt to identify some appropriately descriptive subset of the complete set of VULS.
3. **Given that we can mine a variety of different categories of VULS which of these are the most useful in terms of effectiveness and efficiency?** Effectiveness and efficiency are important with respect to classification in general although there is typically a trade-off between the two.

4. **How do we measure the quality of a set of VULS without applying them to a test set?** One way of testing a set of identified VULS is to apply them in a vertex classification setting. However, in practice, this opportunity will typically not be available. An alternative VULS quality measure is thus required.

5. **Once a set of VULS have been identified what is the mechanism for uitilising this set of VULS in the context of vertex classification?** VULS are only of benefit if they can be successfully applied with respect to vertex classification, how this can best be achieved was a subject for the research.

The research presented in this thesis used sheet metal forming, particularly AISF, as a focus. There were thus also a number of subsidiary application dependent research questions that the work needed to address. Namely:

1. **How best to generate the desired grid graphs from raw AISF data?** In the sheet metal forming industry desired surfaces are typically specified using CAD data, while the shapes produced can be defined in terms of a point cloud obtained using some optical measuring instrument. How grid graphs can be generated from this data was unclear at the start of the research.

2. **What further applications can the VULS concept be applied to?** For the VULS idea to have general utility it needs to have a wide range of applicability. The nature of these further applications, applications that entail 3D surface analysis of some kind, was unclear at the commencement of the work.

The overall objective of the work presented in this thesis was thus to provide answers to the above research questions.

## 1.4 Research Methodology

The adopted research methodology was to commence by investigating a mechanism for identifying the complete set of VULS in a given grid graph. The start point for this work was the well known gSpan algorithm for frequent subgraph mining [229]. The gSpan algorithm operates in a Depth First Search (DFS) manner and this seemed like an appropriate strategy to be adopted for VULS mining. The gSpan algorithm also features a particular canonical form for graph representation and the concept of right most extension, both of which were adapted for the work described in this thesis.

It was anticipated, as noted above, that the complete VULS algorithm would be computationally intensive. Alternative, more efficient VULS mining algorithms were thus seen to be desirable. Three alternative forms of VULS mining were considered: (i) minimal VULS mining, (ii) frequent VULS mining and (iii) minimal frequent VULS mining. The four different forms of VULS mining could then be compared in the context of effectiveness.

To conduct the desired evaluation "real life" data sets, describing 3D surfaces that had been manufactured, were obtained from industry. More specifically data sets were obtained from the Tecnalia Corporation (Spain) and the IBF institute of metal forming (Germany) with whom (at time of writing) the Department of Computer Science at The University of Liverpool had contact within the context of the INnovative MAnufacturing (INMA) Framework 7 European project. The data sets described two flat-topped pyramid shapes referred to as the Gonzalo and Modified pyramids. In total eight data sets were obtained each comprising before and after point clouds. For further exploration, and to investigate the more general utility of the VULS mining concept, other forms of 3D surface were considered. More specifically satellite image data describing two villages located in a rural part of the Ethiopian hinterland were obtained using the Google Static Map Service and translated in grid graphs.

The evaluation was conducted predominantly in the context of vertex classification. The available data sets were divided into appropriate training and test sets. VULS mining was then applied to the training sets and the resulting set of VULS utilised with respect to the test sets. This process is illustrated in Figure 1.3. The resulting predicted vertex labelling could then be compared to the known labelling. The VULS mining algorithms and prediction algorithm (Backward-Match-Voting algorithm) in Figure 1.3 will be described further in Chapter 5 and Chapter 6 respectively. Standard approaches in the field of data mining, and more specifically classification, were used (such as Ten-fold Cross Validation (TCV) [79]) for testing and evaluation purposes. The metrics used to measure classification performance were accuracy, runtime, and AUC (Area Under the receiver operating Curve [9, 23, 97]). A number of additional metrics, specific to VULS mining, were also used; namely: number of VULS and coverage.

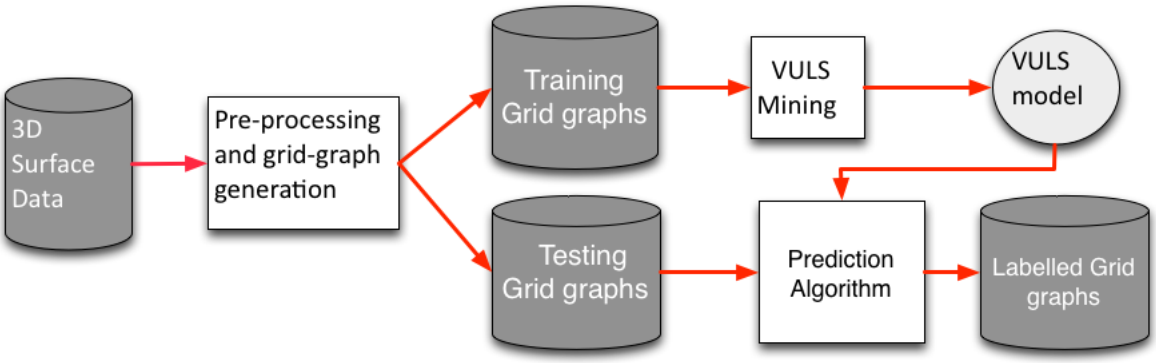



FIGURE 1.3: The VULS mining evaluation process using training and test sets

Comparisons were also conducted using traditional classifiers for vetex classification, namely the J48 decision tree classifier and Naive Bayes. To determine whether the results obtained were also statistically significant the Friedman and Nemenyi tests were applied [29, 81, 143]. Finally, a variant of *Demšar*'s significance diagrams was used to visualise the statistical results obtained.

## 1.5 Contributions

The main contributions of the research presented in this thesis are:

1. The concept of VULS, which is entirely novel within the context of graph mining.
2. An alternative approach to vertex label classification that does not use a training set in the traditional form. Existing wok on vertex classification has typically adopted a traditional approach that uses a training set, comprised of a set of pre-labelled subgraphs each featuring a small number of edges and encoded in a tabular format, which is then used to generate a classifier which can then be applied to new data. This approach assumes the existence of an appropriate set of subgraphs. The proposed VULS mining automates the labelled subgraph identification process (although training data is still required).
3. Four algorithms for mining four types of VULS within a host graph. The algorithms can be used to find the complete set of VULS, only the minimal ones, or frequent ones, or minimal frequent ones. More specifically:
   (a) **compVULSM** The complete VULS mining algorithm that finds all the VULS that exist in a given input graph.
   (b) **minVULSM** The minimal VULS mining algorithm which finds the subset of VULS whose subgraphs are not VULS (but whose supergraphs may be). The conjecture here was that this would be a more efficient form of VULS mining (than complete VULS mining) that would still realise an effective set of VULS.
   (c) **freqVULSM** The frequent VULS mining algorithm which finds the subset of VULS whose occurrence count is above some predefined threshold. Again the conjecture here was that this would be more efficient than complete VULS mining while still providing an effective set of VULS (in terms of vertex classification).
   (d) **minFreqVULSM** The minimal frequent VULS mining algorithm that finds the subset of VULS that are both minimal and frequent. The conjecture here was that minimal frequent VULS mining would combine the advantages of both minimal and frequent VULS mining.
4. The Backward-Match-Voting (BMV) algorithm for vertex classification.
5. A complete evaluation and statistical analysis of the relative merits of VULS mining in the context of vertex classification with respect to sheet steel forming, the primary application domain considered.
6. I wider investigation of the application of VULS mining in the context of vertex classification with respect to satellite image interpretation and analysis, the secondary application domain considered.

## 1.6 Thesis Organization

The organisation of the remainder of this thesis is as follows:

1. **Chapter 2** presents the necessary background to the work described together with a review of related work.

2. **Chapter 3** presents a brief description of the AISF and satellite image application domains. More specifically the datasets which were used for experimental purposes. Recall that the AISF application was used as the primary application focus for the work, while the latter was used to confirm the general applicability of the proposed VULS techniques. The necessary data preparation and image pre-processing is also described in this chapter.

3. **Chapter 4** provides a formalism for the VULS concept including the four identified categories of VULS: (i) complete, (ii) minimal, (iii) frequent and (iv) minimal frequent. The chapter includes simple examples of each.

4. **Chapter 5** presents detail of each of the four proposed VULS mining algorithms: (i) compVULSM, (ii) minVULSM, (iii) freqVULSM and (iv) minFreqVULSM.

5. **Chapter 6** describes the prediction algorithm, namely the Backward-Match-Voting (BMV) algorithm, for applying VULS in the context of vertex classification.

6. **Chapter 7** gives a comprehensive experimental analysis of the nature of VULS in the context of vertex classification with respect to AISF. The chapter includes a detailed statistical analysis.

7. **Chapter 8** presents an experimental analysis of the utility of VULS in the context of vertex classification with respect to an alternative application domain than sheet metal forming; specifically satellite image interpretation.

8. **Chapter 9** concludes the thesis. The chapter presents a summary of the work together with the the main findings in terms of the identified research questions identified above. The chapter also presents a number of potential directions for future work founded on the research presented in this thesis.

## 1.7 Published Work

Some of the work described in this thesis is founded on work previously published by the author in refereed publications. These publications are itemized below, in each case the relevance with respect to the contents of this thesis is highlighted.

1. Journal Papers:

(a) **W. Yu, F. Coenen, M. Zito, and S. El-Salhi (2015). Vertex Unique Labelled Subgraph Based Classification. Submitted for refereeing to the AI Journal.** This journal paper compares the operation of earlier versions of the complete VULSM and minVULSM algorithms with respect to four different styles of grid graph: (i) degree 4 undirected, (ii) degree 8 undirected, (iii) degree 4 directed and (iv) degree 8 directed. The reported experiments indicate that the VULSM algorithm, when applied to directed grid graphs with a degree of 4, tended to produce best results. The relevance of this paper with respect to the thesis is that the paper summarised the work presented here. Note that similar experimental results to those described in the paper are presented in chapter 7.

2. Conference Papers:

(a) **A. Albarrak, F.Coenen, Y.Zheng, W.Yu (2012). Volumetric Image Mining Based on Decomposition and Graph Analysis: An Application to Retinal Optical Coherence Tomography. The 13th IEEE International Symposium on Computational Intelligence and Informatics (CINTI2012), pp. 263-268. Budapest, Hungary. 20th-22th November, 2012.** This paper considered a method for classifying volumetric images using a decomposition and graph analysis based method. In the paper a hierarchical decomposition techniques was used to incrementally divide a given 3D volume into sub-volumes according to some critical function and then to represent the decomposition as a tree. The evaluation was conducted by considering the classification of 3D Optical Coherence Tomography (OCT) retinal images according to whether they feature Age-related Macular Degeneration (AMD) or not (AMD is an eye condition that can result in blindness in old age). A frequent subgraph mining algorithm was applied to the tree representations and the resulting identified frequent subgraphs used to define a feature vector encoding which was then fed into a standard classifier. The significance with respect to the work presented in this thesis is that the work described, although not directly concerned with vertex classification, lead the author to the idea of VULS based vertex classification; the central theme of this thesis.

(b) **W. Yu, F. Coenen, M. Zito, and S. El-Salhi (2013). Vertex Unique Labelled Subgraph Mining. Thirty-third BCS SGAI International Conference on Artificial Intelligence (BCS SGAI2013), Springer Berlin Heidelberg, pp. 21-38. Cambridge, UK. 10th-12th December, 2013.** This was the first paper published by the author that proposed the concept of Vertex Unique Labelled Subgraph Mining (VULSM), a new form of graph mining. The paper presents the Right-most Extension VULS Mining (REVULSM) algorithm for finding the complete set of VULS in a

given input graph. The REVULSM algorithm was a preliminary version of the compVULSM algorithm presented later in this thesis in Chapter 5. The reported experimental results demonstrated that the VULS idea is sound and that the REVULSM algorithm can identify VULS in real data. The evaluation was conducted using the sheet steel forming application data also used in this thesis.

(c) **W. Yu, F. Coenen, M. Zito, and S. El-Salhi (2013). Minimal Vertex Unique Labelled Subgraph Mining. The 15th International Conference on Data Warehousing and Knowledge Discovery (DaWak 2013), Springer Berlin Heidelberg, pp. 317-326. Prague, Czech Republic. 26th-29th August, 2013.** This paper built on 2(b) and proposed a minimal VULS mining algorithm, the Minimum Breadth First Search Rightmost Extension Unique Subgraph Mining (Min-BFS-REUSM) algorithm, to improve the efficiency of the VULSM process (with respect to the REVULSM algorithm presented in 2(b)). The Min-BFS-REUSM algorithm was an early version of the minVULSM algorithm presented later in this thesis in Chapter 5. The reported experimental results indicated that the Min-BFS-REUSM algorithm could successfully identify all minimal VULS in reasonable time and with (in some cases) excellent coverage (an important requirement in the context of the sheet metal forming application used as a focus for the work).

(d) **W. Yu, F. Coenen, M. Zito, and S. El-Salhi (2013). Vertex unique labelled subgraph mining for vertex classification. The 9th International Conference on Advanced Data Mining and Applications (ADMA 2013), Springer Berlin Heidelberg, pp. 542-553. Hangzhou, China. 14th-16th December, 2013.** This paper was the first to suggest that the VULS concept could be used in the context of vertex classification and proposed the Match-Voting algorithm for applying sets of identified VULS in the context of vertex classification. The Match-Voting algorithm was a fore-runner of the Backward-Match-Voting (BMV) algorithm presented later in this thesis in Chapter 6. Experiments were conducted using the REVULSM and Min-BFS-REUSM algorithms from papers 2(b) and 2(c), and the sheet metal forming application also used for evaluation purposes in this thesis. The results reported in this paper indicated that minimal VULS mining is both efficient and effective in term of coverage (at least in the context of the sheet metal forming application used for the evaluation).

(e) **W. Yu, F. Coenen, M. Zito, and K. Dittakan (2014). Classification of 3D Surface Data Using the Concept of Vertex Unique Labelled Subgraphs. The 9th International Workshop on Spatial and Spatiotemporal Data Mining (SSTDM), 2014 IEEE International Conference on Data Mining Workshop (ICDMW), pp. 47-54. Shenzhen, China. 14th-17th December, 2014.** This paper extended the

work from 2(b), 2(c) and 2(d) by introducing the idea of frequent and minimal frequent VULSM. The paper included revised versions of the earlier algorithms for complete and minimal VULSM similar to those presented later in Chapter 5 of this thesis. This paper also presented the Backward Match Voting (BMV) algorithm for predicting (classifying) vertex labels associated with "unseen" graphs using a given collection of VULS. An extended version of the description of the BMV algorithm is presented in Chapter 6 of this thesis. Unlike previous papers in this series the evaluation was conducted using a satellite image interpretation application (earlier papers had all used sheet metal forming as the evaluation application domain). The satellite data used describing two villages located in a rural part of the Ethiopian hinterland data. The collected satellite images were encoded in a grid format which was then converted into a 3D surface formalism by considering the average grey scale value for each grid cell as the $z$ value. The idea here was to predict vertex labels describing ground type. A statistical analysis of the results, using the Friedman test, was also presented so as to demonstrate the statistical significance of the VULS based 3D surface regional classification idea. The results indicate that the VULS concept is also suited to the task of 3D surface regional classification.

## 1.8 Summary

In summary, this chapter has provided an overview and background for the research described in the remainder of this thesis, including details concerning the motivation for the work and the research question and associated issues. The work is broadly focused on four different categories of VULS: (i) complete, (ii) minimal, (iii) frequent and (iv) minimal frequent. The main objective of the work described is to provide an answer to the research question: *"How best can the proposed VULS mining be conducted so as to achieve effective vertex classification?"*. This introductory chapter has also provided a brief description of the programme of work, the evaluation criteria used and the contribution of the work. In the following chapter (Chapter 2) a literature review, intended to provide much more detail regarding the necessary background concerning the research described in this thesis, is presented.

# Chapter 2

# Literature Review

## 2.1 Introduction

This chapter presents a review of the background, related work and the application domain central to the work described in this thesis. The related work is mainly founded on three areas of research study (as shown in the Venn diagram presented in Figure 2.1): (i) Graph mining, (ii) Vertex classification and (iii) 3D surface representation. With respect to Figure 2.1 the research described in this thesis can be conceptually placed at the intersection between the three research themes. Each of the themes is considered in this chapter. We commence the discussion, in Section 2.2, with a review of graph mining techniques. One particular graph mining technique that is of significant relevance with respect to VULS mining is Frequent Subgraph Mining (FSM). This is thus considered in some detail in Section 2.3. We then go on to consider 3D surface representation and vertex classification techniques in Sections 2.4 and 2.5 respectively. A review of the evaluation metrics, and their derivation, used with respect to the work described in this thesis, is then presented in Section 2.6. Finally, this chapter is concluded with a summary in Section 2.7.

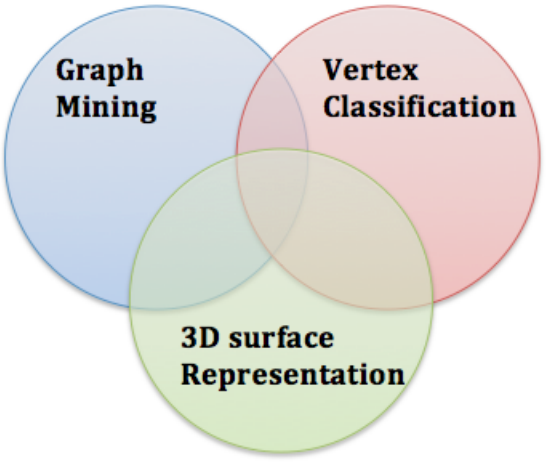



FIGURE 2.1: The three main research themes of this thesis: Graph Mining, Vertex Classification and 3D Surface Representation.

## 2.2 Graph Mining

Graph mining is concerned with the discovery of hidden information in graph data. A graph is a set of vertices and a set of edges connecting pairs of vertices. In this thesis, and in common with much other work, graphs are also considered to have both vertex and edge labels. Thus, without loss of generality, each graph G is defined in terms of a tuple of five elements:

$$\langle V, E, L_V, L_E, F \rangle \tag{2.1}$$

where:

1. $V$ is a set of $n$ vertices. The elements of $V$ are denoted by the letters u or v, occasionally with subscripts if the nature of the collection of vertices is important with respect to a particular context.
2. $E$ is a set of $m$ edges. The elements of $E$ are usually denoted by the letter e, again occasionally with subscripts.
3. $L_V$ is the set of vertex labels.
4. $L_E$ is the set of edge labels.
5. $F$ is a labelling function that defines the mappings $V \rightarrow L_V$ and $E \rightarrow L_E$.

Each vertex (or edge) of the graph is required to have a single label; the same label can be assigned to many vertices (or edges) in the same graph. If the labelling is restricted to the vertices, or the edges, the graph is defined by a four-tuple. If no labelling is present, the graph is defined by the pair $(V, E)$.

Throughout this thesis the expression $|X|$ denotes the cardinality (number of elements) of $X$, if $X$ is a set or an ordered sequence. However, if $G$ is a graph, the size of $G$, is normally defined as the number of vertices or the number of edges. In this thesis $|G| = |V|$ (thus, $|E| = O(|V|)$).

If $E$ is a collection of sets each formed by two elements of $V$ we say that $G$ is an undirected graph. If $E$ is a collection of ordered pairs of elements of $V$ then $G$ is a directed graph. A path $P$ in a graph $G$ is an ordered sequence of vertices $v_1, v_2, \ldots, v_{|P|}$ such that $v_i$ and $v_{i+1}$ form an edge in $G$, for each $i \in \{1, \ldots, |P| - 1\}$. The graph $G$ is connected if every pair of vertices in $G$ are connected by a path.

A subgraph of $G$ is any graph $\phi \equiv \{V_\phi, E_\phi, L_{V_\phi}, L_{E_\phi}, F\}$ with ($V_\phi \subset V$, $E_\phi \subset E$, $L_{V_\phi} \subset L_V$ and $L_{E_\phi} \subset L_E$). A candidate $c$ of $G$ is a subgraph of $G$ without the vertex labelling, thus $c = \{V_c, E_c, L_{E_c}, F\}$. Note that $c$ may appear many times in $G$, there may be a number of instances of $\phi$ that are equivalent (isomorphic) to $c$.

Given a candidate $c$, and a graph $G$, a set of potential vertex labels $L_v$ can be associated with each $v \in V_c$ by inspecting the labelled occurrences of $v$ in $G$. We denote

by $S'$ the collection $\{L_v : v \in V_c\}$. A VULS of $G$ is a connected subgraph where $|L_v| = 1$ for each $L_v \in S'$.

Referring back to the introduction to this chapter, other than "standard" VULS as defined above, we can identify three other kinds of VULS: Minimal VULS, Frequent VULS and Minimal frequent VULS.

> **Definition 1: Minimal VULS.** A minimal VULS is a VULS such that none of its subgraphs are VULS.

> **Definition 2: Frequent VULS.** A Frequent VULS (FVULS) is a VULS $\phi$ whose occurrence count, $Occurrence(\phi)$, within the input graph $G$, is greater than some pre-specified threshold $\sigma$.

> **Definition 3: Minimal frequent VULS.** A minimal frequent VULS is a VULS $\phi$ which is both minimal and frequent.

The relationship between the different forms of VULS is illustrated in Figure 2.2. The relevance of the numbering included in Figure 2.2 will become clear later in chapter 4. The concept of VULS will be considered in further detail, with some worked examples, in chapter 4.

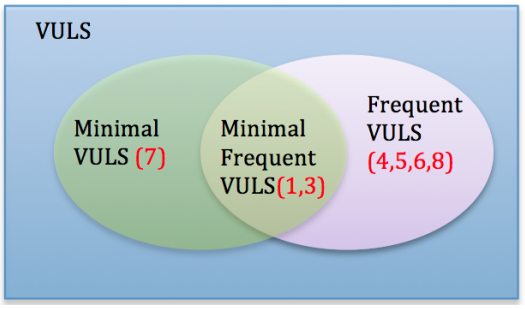



FIGURE 2.2: Venn Diagram showing the relationship between VULS, Minimal VULS, Frequent VULS and Minimal frequent VULS

This section is organised as follows. Section 2.2.1 presents an overview of the research domain of graph mining starting with a categorisation of this domain. Graph mining often involves the identification of patterns (subgraphs or subtrees) in graph data. There are various forms of graph pattern that may be of interest and these are reviewed in section 2.2.2. Subgraph isomorphism is a central activity frequently required for graph mining and this is discussed in section 2.2.3. Another important aspect of graph mining is the mechanisms whereby graphs are represented so as to facilitate mining; some kind of canonical form is required, and this is therefore discussed in section 2.2.4.

### 2.2.1 Graph Mining Categorisation

The domain of graph mining can be categorised in a number of ways, the most straightforward is to consider the domain in terms of: (i) transaction graph mining and (ii)

single graph mining. In transaction graph mining [47, 115, 118, 136, 221] the input comprises a collection of graphs and the objective is to find hidden patterns that exist across this collection. Transaction graph mining is essentially an extension of frequent items set mining [2, 21, 93, 94] as used with respect to tabular data. In single graph mining the input comprises a single large graph and the objective is to find patterns that exist within this single large graph. An application domain where single graph mining is often applied is social network analysis [3, 133, 203]. Examples of social networks include twitter communities, Facebook communities, and co-authoring networks. The work presented in this thesis belongs to the single graph mining category.

Alternatively, graph mining when used for classification purpose can be divided into two categories: (i) graph classification and (ii) vertex classification. With respect to the first category the objective is to use pre-labelled training graphs to produce a classifier with which to classify previously unseen graphs [128]. From the literature we can identify a number of graph classification approaches, these include: LEAP [230], gPLS [182], CORK [205], and COM [123]. With respect to the second category the objective is to train a classifier, using training data, so as to construct a classifier that can be used to classify (label) vertices whose labels were previously unknown. Vertex classification may be conducted within a single graph (where some of the vertices have been pre-labelled for training purposes) or across a collection of graphs (where the vertices in some graphs have been pre-labelled for training purposes). The work presented in this thesis is particularly concerned with the concept of vertex classification and thus the principles of vertex classification are considered in further detail in Section 2.4 later in this chapter.

### 2.2.2 Subgraph Patterns

A central concern of graph mining, either directly or indirectly, is the identification of patterns (subgraphs) within graph data. We are typically interested in identifying subgraphs that feature some particular structure. The patterns of interest can be categorised as follows [46]: (i) topological patterns, (ii) frequent patterns and (iii) relational patterns. The first has to do with the topology of the subgraphs of interest, subgraphs that feature a specified topology. Examples include arbitrary subgraphs, induced subgraphs, maximal complete subgraph (cliques) and paths [28, 118]. The second category has to do with frequency of occurrence, thus subgraphs that are frequent, maximal, or closed. The third category is concerned with the relationships between subgraphs, thus: various embeddings within discovered patterns, embeddings that can have arbitrary overlaps, partial overlaps, being vertex- and/or edge-disjoint and coherent substructures [107]. The particular patterns of interest with respect to this thesis are VULS, which can be considered to fall within the topological category identified above. If we are talking about frequent VULS or minimal frequent VULS then the VULS concept can also be said to fall into the frequent pattern category.

As will become apparent later in this thesis, the proposed VULS mining methods used in this work, especially frequent VULS and minimal frequent VULS mining, "borrow" concepts and techniques from the domain of FSM (Frequent Subgraph Mining), thus VULS mining techniques also fall into the second of the above categories. From the literature the most common types of frequent subgraph that can be mined include: (i) "ordinary" frequent subgraphs [109], (ii) closed frequent subgraphs [39, 43, 228], (iii) maximal frequent subgraphs [106, 207], (iv) approximately frequent substructures [240], (v) contrast substructure [46], and (vi) discriminative frequent substructures [124, 232]. The first of the above, ordinary frequent subgraphs, is the most relevant to VULS mining, especially frequent VULS and minimal frequent VULS mining. Because of its significance with respect to the work presented in this thesis frequent subgraph mining is discussed in further detail in Section 2.3 later in this chapter.

### 2.2.3 Graph Isomorphism

Graph isomorphism is concerned with the problem of deciding whether two graphs are identical or not. In other words, given two graphs, does there exist a one to one mapping of the vertices in one graph to the vertices in the other such that vertex adjacency is preserved [78]. Isomorphism testing is computationally expensive. This is particularly the case with respect to frequent subgraph mining (see below) because of the large number of graph comparisons that need to be made; graph isomorphism plays an essential role in determining whether a candidate subgraph exists within a given graph and how often it exists. Isomorphism testing also plays a significant role with respect to vertex classification (see section 2.5 below). Subgraph isomorphism with respect to non-regular graphs is known to be a NP-complete[1] problem [78], no algorithm has been found that can solve the isomorphism testing problem in polynomial time [49, 85]. However, subgraph isomorphism with respect to trees, permutation graphs, chordal graphs and grid graphs (as used in this thesis) is not a NP-complete problem [78]. Note that a tree is a graph containing no cycles [216], and a grid graph is a node-induced finite subgraph of the infinite grid [119].

There has been much work on algorithms to enhance the efficiency of isomorphism testing; examples include: (i) Ullmann's algorithm [160, 215], (ii) Backtracking algorithm [186], (iii) Nauty [157], (iv) VF and VF2 using a State Space Representation (SSR) [48, 49] and (v) geometric isomorphism [139]. However, none of these existing isomorphism detection algorithms are entirely suited to VULS mining, or VULS based vertex classification, as envisioned in this thesis. The reasons for this are:

1. Most existing isomorphism approaches use vertex invariants (such as the degree of a vertex, the "twopaths" concept, adjacency triangles, k-cliques, independent k-sets and distances) to solve the isomorphism problem [78]. However, calculation

[1]NP-complete: In computational complexity theory, the complexity class NP refers to the set of decision problems whose solutions can be verified in polynomial time. A decision problem $\rho$ is NP-complete if $\rho$ is in NP and every other problem in NP is reducible to $\rho$ in polynomial time [50].

of the invariant values, especially with respect to the more complex invariants, is time consuming; it is also very difficult to determine which invariant is the best for a particular graph. Instead, what is typically done is to leave the decision of if and when to use a vertex invariant up to the user (except for degree which is always used). Given a number of complex graphs, time would be required to experimentally determine what invariant to use.

2. Most of above isomorphism approaches use adjacency matrixes to represent graphs, which is not well suited in the context of the VULS mining proposed in this thesis.

Thus the isomorphism testing approach used with respect to the work presented in this thesis is based on that espoused within the gSpan frequent subgraph mining algorithm [229]. The gSpan isomorphism approach is to take a single graph $G$ and apply some function $C(G)$ which returns a canonical label such that $C(G) = C(H)$ iff $G$ and $H$ are isomorphic. In other words, gSpan addresses the graph isomorphism problem by comparing the "canonical labellings" of two graphs. If these labellings are the same, then these graphs are isomorphic. The concept of canonical labelling (forms) is described in more detail in the next section. The gSpan algorithm is considered in further detail in Section 2.3 later in this chapter.

### 2.2.4 Canonical Forms

A canonical form is an agreed way of representing some object (such as a graph). In the context of graph mining canonical forms are important so that the relevant algorithms can be both effective and efficient [177, 222]. In the case of isomorphism testing canonical forms are significant so as to ensure that, given two identical graphs, they are represented in the same manner (thus simplifying the isomorphism testing). In other words, a canonical form facilitates isomorphism checking because it ensures that if a pair of graphs are isomorphic, then their canonical labellings will be identical [20, 117, 136, 177]. In the case of graph mining algorithms that involve the generation of candidate graphs, such as the VULS mining algorithms with respect to this thesis, the usage of a canonical form is important so as to avoid the generation of duplicates (the same graph but expressed in different ways).

From the literature we can identify a number of canonical labelling strategies such as: (i) Minimum DFS Coding [229], (ii) the Coding proposed by Akihiro Inokuchi [118], (iii) Canonical Adjacency Matrix (CAM) encoding [109, 116], (iv) Canonical Representation of free trees [180], (v) String encodings for rooted ordered trees [152, 171], (vi) Canonical forms for rooted unordered trees [8, 42], (vii) DFS Label Sequence (DFS-LS) encoding [236], (viii) Depth-Label Sequence (DLS) encoding [127], (ix) Breadth-First Canonical String (BFCS) encoding [40], (x) Depth-First Canonical String (DFCS) encoding [43] and (xi) Consolidated Prüfer Sequence(CPS) encoding[200]. The first, Minimum DFS Coding differs from the other canonical forms in that it uses a tree representation of each graph instead of a more traditional adjacency matrix. With respect to the work

described in this thesis, Minimum DFS Coding, was adopted because of its popularity in the context of frequent subgraph mining.

Depth-First Search (DFS) is an algorithm for traversing or searching tree or graph data structures. One starts at the root (some arbitrary vertex in the case of a graph) and explores each branch before backtracking. This requires some ordering of vertex identifiers; given two branches emanating from a vertex some ordering needs to be imposed so that one branch is selected before the other. In the case of graphs a similar mechanism needs to be imposed to select a start vertex.

For one graph $G$, there are lots of ways to construct different DFS trees by selecting a different root $U$ and different growing edges. When performing a depth-first search in a graph, various DFS trees can be constructed each with its own DFS code. In other words, a lot of DFS codes can be adopted to represent a graph $G$, to prevent ambiguity we require a canonical form so that among all these DFS codes, there is only one minimal DFS code. It is this minimal DFS code that will then be employed to represent $G$. In the following, how to generate various DFS codes for graph $G$ is described first. Then an example is given to illustrate such process and how to choose the minimal one as the canonical form from the various DFS codes generated is also explained at the same time.

The DFS code for a graph consists of a sequence of tuples describing the edges in a given graph $G$. The tuples are of the form:

$$\langle U, V, L_U, L_E, L_V \rangle$$

where: (i) $U$ is the identifier for the start vertex, (ii) $V$ is the identifier for the end vertex, (iii) $L_U$ is the vertex label for $U$, (iv) $L_E$ is the edge label and (v) $L_V$ is the vertex label for $V$. The DFS codes for $G$ is generated as follows:

1. Impose an ordering on the vertices (a sequential set of unique vertex identifiers). Mark all edges as "unread".
2. Select the first vertex in the ordering as the root (the start point) $U$.
3. Create an ordered list of edges $\{e_1, e_2, \ldots, e_n\}$ emanating from root $U$. List backward edges prior to forward edges.
4. Process the edge list $\{e_1, e_2, \ldots, e_n\}$, $i = 1$ $(1 \leqslant i \leqslant n)$:
   (a) If $e_i$ is marked as being "unread", create a code describing the edge and store, mark the edge as "read" and proceed to (b).
   (b) If $e_i$ is a forward edge, choose the end vertex of $e_i$ as root $U$, repeat from 3.
   (c) Otherwise, $e_i$ is a backward edge, increment i $(i++)$, and repeat from (a).

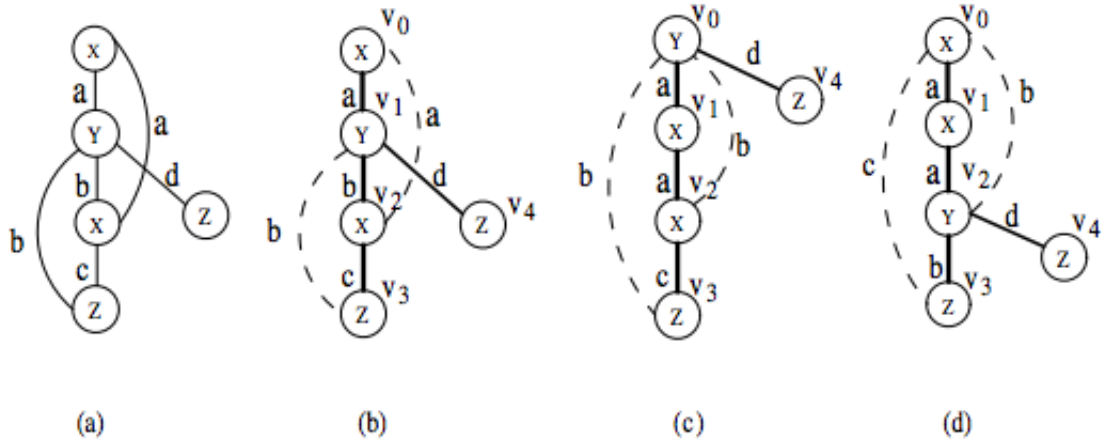



FIGURE 2.3: Depth-First Search Tree and its Forward/Backward Edge Set [229]. Note that forward edges are represented by solid lines and backward edges dashed lines.

TABLE 2.1: DFS code for Figure 2.9 (b), (c) and (d) [229]

| edge NO. | (b) | (c) | (d) |
|---|---|---|---|
| 0 | $\langle 0, 1, X, a, Y\rangle$ | $\langle 0, 1, Y, a, X\rangle$ | $\langle 0, 1, X, a, X\rangle$ |
| 1 | $\langle 1, 2, Y, b, X\rangle$ | $\langle 1, 2, X, a, X\rangle$ | $\langle 1, 2, X, a, Y\rangle$ |
| 2 | $\langle 2, 0, X, a, X\rangle$ | $\langle 2, 0, X, b, Y\rangle$ | $\langle 2, 0, Y, b, X\rangle$ |
| 3 | $\langle 2, 3, X, c, Z\rangle$ | $\langle 2, 3, X, c, Z\rangle$ | $\langle 2, 3, Y, b, Z\rangle$ |
| 4 | $\langle 3, 1, Z, b, Y\rangle$ | $\langle 3, 0, Z, b, Y\rangle$ | $\langle 3, 0, Z, b, X\rangle$ |
| 5 | $\langle 1, 4, Y, d, Z\rangle$ | $\langle 0, 4, Y, d, Z\rangle$ | $\langle 2, 4, Y, d, Z\rangle$ |

For instance consider the graph $G$ given in Figure 2.3 (a). From this graph various DFS trees can be generated depending on the choices of the root $U$ among the vertices. Figure 2.3(b), (c) and (d) give three different DFS trees for the graph in Figure 2.3(a). If we choose the first "X" in Figure 2.3(a) as the root $U$, the DFS tree shown in Figure 2.3(b) will be generated. In the same manner, if we choose "Y" or the second "X" as the root. The trees shown in Figures 2.3(c) and (d) would be generated. Thus $G$ can be represented as three different ways. Therefore there are three different DFS codes one per DFS tree as shown in Table 2.1 to represent the same graph $G$ (Figure 2.3(a)). Note that forward edges are represented by solid lines and backward edges dashed lines. The question is how to find the minimal DFS codes from these three DFS codes.

The minimum DFS code, as the name suggests, is the minimum code according to some lexicographic order of the graph edges. In other words, an ordering is imposed on the element values with respect to five tuples $\langle U, V, L_U, L_E, L_V\rangle$. Thus, in Table 2.1, edge 0 is compared first with respect to (b), (c) and (d). The first and second elements are the same (0 and 1). The third element "X" is lexicographic before "Y", thus DFS codes (b) and (d) are "smaller" than (c). We need to find the minimal DFS code, thus (c) is out, (b) and (d) are left so we continue by comparing edge 1; obviously the first

two elements are the same (1 and 2), the third element "X" is before "Y", thus DFS code (d) is smaller than (b). Thus, (d) is the minimum DFS code for $G$ and thus it is the unique canonical form with which to represent $G$ (Figure 2.3(a)). Note that a subgraph is a duplicate subgraph if and only if its DFS code is not minimum.

Given two graphs $G$ and $G'$, $G$ is isomorphic to $G'$ if and only if the minimum DFS code of $G$ is identical to the minimum DFS code of $G'$. The isomorphism testing process is given in algorithm 1. If the minimum DFS codes of two graphs are not the same, the two graphs are not identical. Thus the problem of mining subgraphs can be said to be equivalent to analysing their corresponding minimum DFS codes. Note that the minimal DFS code is dependent upon the global set of labels in an input graph or set of input graphs. Thus any two subgraphs that subscribe to this global labelling can still be compared; of course if they do not subscribe to this global labelling then comparison is not possible. As will become apparent later in section 2.3.3.2, the use of minimum DFS codes can enhance the process of frequent subgraph mining by comparing the minimal DFS code of two graphs to do the isomorphism testing. Thus for the VULS classification proposed in this thesis, minimum DFS coding was adopted for subgraph matching (subgraph isomorphism) with respect to both the VULS mining and vertex classification by VULS.

**Algorithm 1**

```
1:  procedure IsomorphismTest(G, G')
2:      S= the set of minimal DFS codes for graph G
3:      S'= the set of minimal DFS codes for graph G'
4:      result=true
5:      if |S| != |S'| then
6:          result=false
7:      else
8:          for all i from 1 to |S| do
9:              if S_i ≠ S'_i then
10:                 result=false
11:                 return result
12:             end if
13:         end for
14:     end if
15:     return result
16: end procedure
```

## 2.3 Frequent Subgraph Mining

Frequent Subgraph Mining (FSM) is, as the phrase suggests, concerned with the identification of subgraphs that occur frequently in the input data. A good way of perceiving the FSM process is to conceptualise it as one of searching through a lattice describing all possible patterns and selecting those that are (in some sense) frequent. The work

presented in this thesis "borrows" ideas from the concept of FSM and especially the well known gSpan algorithm [229]. More specifically: (i) some general ideas concerning FSM were incorporated into the proposed VULS mining; and (ii) frequent VULS mining and minimal frequent VULS mining, a variation of the proposed generic VULS mining, was founded on the idea of FSM. FSM algorithms, typically operate using a candidate generation, frequency counting and pruning loop; the proposed VULS mining algorithms operate in a similar manner although (except in the case of frequent VULS or minimal frequent VULS mining) without the frequency counting element.

There are a great variety of FSM algorithms. Some of which were developed for transaction graph mining and have subsequently been used for single graph mining and others that have been specifically designed for single graph mining. It should also be noted that many subgraph mining algorithms are extensions of subtree mining algorithms which in turn are often extensions of frequent item set mining algorithms as used with respect to binary valued tabular data [11, 122]. Recall also that a tree is a special form of graph that offers advantage in the context of graph mining, namely that candidate generation and isomorphism testing is much more straight forward than in the case of graphs.

In the context of FSM, particularly when applied to transaction graph data, a concept known as the Downward Closure Property (DCP) or the anti-monotone property of itemsets is often utilised. This is an important concept with respect to both FSM and frequent VULS mining, thus is discussed in further detail in Section 2.3.1 below. A central element of FSM algorithms is the mechanism used to determine the frequency of subgraphs, this is thus discussed in Section 2.3.2. To determine whether a subgraph is frequent or not its occurrence count is typically compared with a frequency threshold $\sigma$, the nature of this threshold is discussed in Section 2.3.3. Candidate generation is then considered in Section 2.3.3.1.

### 2.3.1 The Downward Closure Property

In the context of transaction FSM, where frequency is counted in terms of one count per transaction in which some subgraph $g$ appears, the downward closure or anti-monotone property states that if $g$ is not frequent none of its supergraphs can be frequent. This is significant with respect to FSM candidate generation as discussed in the following subsection. In some cases the Downward Closure Property (DCP) may apply in the context of single graph mining depending on how the frequency counting is conducted [187, 197, 218]. However, in the context of VULS mining, as will be demonstrated in Chapters 4 and 5, this does not apply; if $g$ is not a VULS this does **not** mean that none of the supergraphs of $g$ are also not VULS.

### 2.3.2 Frequency Counting

As noted above, mechanisms for conducting frequency counting play an essential role in FSM, and by extension frequent (and minimal frequent) VULS mining. In FSM the term

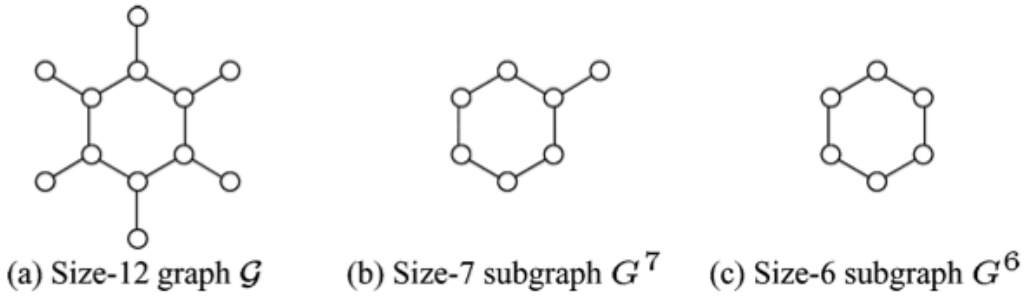



FIGURE 2.4: Patterns with the non-monotonic frequency [138].

*support*[2] is used to describe the occurrence or frequency count of a graph. There are a variety of ways that support can be defined. In part this is dependent on the nature of the FSM problem: transaction graph based or single graph based. In transaction graph mining the options are whether: (i) to consider the presence of a particular subgraph $g$ in a transaction graph $t$ to represent a count of one regardless of how many times $g$ actually occurs in $t$, or (ii) to take into account the occurrences of $g$ in $t$. We refer to the second as *occurrence counting*. In single graph mining the only option is to adopt occurrence counting and take into account occurrences of $g$. Given that the focus of this thesis is on single graph mining the discussion on support counting presented in this section is directed solely at mechanisms in the context of single large graphs [26, 76].

In general, in the context of single graph mining, there are two possible methods for determining the frequency of a subgraph. In the first, two embeddings of a subgraph are considered different, as long as they differ by at least one edge (non-identical). As a result, arbitrary overlaps of embeddings of the same subgraph are allowed. In the second, two embeddings are considered different, only if they are edge-disjoint. These two methods are illustrated in Figures 2.5 and 2.4. In the example presented in Figure 2.5 there are three possible embeddings of the subgraph shown in Figure 2.5(a) in the input graph given in Figure 2.5(b). Two of these embeddings (shown in Figures 2.5(c) and (e)) do not share any edges, whereas the second embedding (Figure 2.5(d)) shares edges with the other two. Thus, if we allow overlaps, the frequency of the subgraph is 3, and if we do not it is 2.

[2]Support can be defined in general as the proportion of occurrence of a subgraph over a total number of graph transactions in a transaction graph mining data set.

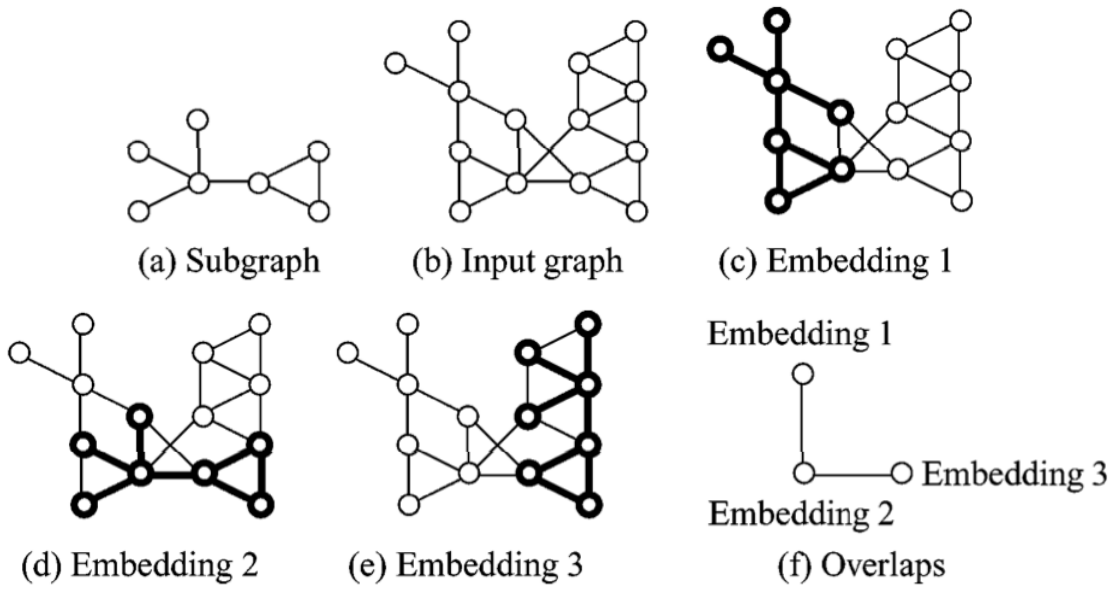



FIGURE 2.5: Overlapped embeddings [138].

The above two ways of counting the frequency of a subgraph lead to problems with dramatically different characteristics. If we allow arbitrary overlaps between non-identical embedding the DCP (see above) does not hold. This is illustrated in Figure 2.4 where both the 7-edge subgraph $G^7$ and the 6-edge subgraph $G^6$ are subgraphs of $\mathcal{G}$. The smaller 6-edge subgraph $G^6$ has only one non-identical embedding whilst the larger 7-edge subgraph $G^7$ has six non-identical embeddings. The occurrence count for $G^6$ is therefore less than the occurrence count for $G^7$, thus in this case the DCP does not necessarily hold; if the frequency threshold $\sigma$ is defined as 6, then $G^7$ is frequent but its subgraph $G^6$ is not frequent. On the other hand, if we determine the frequency of each subgraph by counting the maximum number of its edge-disjoint embeddings[3], then the DCP does hold [217].

With respect to the proposed frequent VULS and minimal frequent VULS mining the first frequency counting method, where two embeddings of a subgraph are considered different as long as they differ by at least one edge (thus allowing two embeddings to share some edges), was adopted. Thus the proposed frequent VULS and minimal frequent VULS mining algorithms presented later in this thesis do not subscribe to the DCP (see Chapter 5).

### 2.3.3 The Minimum support threshold $\sigma$

As noted in the foregoing subsection a subgraph is considered to be frequent if its support exceeds a given threshold $\sigma$. One option is to fix the value for $\sigma$ prior to commencing the FSM. However, it should be noted that: (i) if $\sigma$ is low we will extract a large number subgraphs, including many spurious subgraphs; and (ii) if $\sigma$ is high we may miss many interesting patterns occurring at low levels of support. Most existing FSM algorithms

[3]Two embeddings of a subgraph are considered different, as long as they don't share any edge.

require that $\sigma$ be pre-specified. These algorithms are then evaluated by considering a range of values for $\sigma$ [26, 70, 76]. This is not appropriate with respect to VULS mining due to two drawbacks noted above. Also by fixing the value of $\sigma$ in advance, the occurrence distribution of generated subgraphs is not taken into account. An alternative is to compute $\sigma$ dynamically in a self-adaptive manner. Thus, in the context of frequent VULS and minimal frequent VULS, a bespoke mechanism for calculating $\sigma$ dynamically was developed by the author; this will be presented later in Chapter 5.

In the context of the above there are other measures that can be used to identify "interesting" subgraphs such as all-confidence [170], h-confidence [226], Jaccard similarity measure [44], MiNImum image based (MNI) support metric [26], Harmful Overlap (HO) support metric [76], and Maximum Independent Sets (MIS) support metric [76]. However, the ordinary support measure is the most frequently used and hence it was adopted with respect to the work presented in this thesis.

#### 2.3.3.1 Candidate Generation

As noted above, candidate generation forms a central part of FSM as well as the proposed VULS mining. FSM typically operates in an iterative manner starting with one edge candidate subgraphs, then two edge candidate subgraphs, and so on. The proposed VULS mining also operates in this manner. On each iteration the graphs from the previous iteration are grown by adding a node connected to the previous graph by an edge. The important issue in candidate generation is to grow subgraphs so that, as the process continues, duplicate candidate graphs are not generated. This is why it is important to use a canonical form as described above in Subsection 2.2.4.

From the literature the most representative candidate generation methods that can be identified are: (i) Right most extension [1, 196, 198], (ii) Equivalence class based extension [237], (iii) Right-and-left tree join [100], (iv) Extension and join [38, 41, 110, 117], (v) Level-wise join [136], and (vi) gFSG-join [139]. With respect to the proposed VULS mining the right most extension mechanism was adopted due to its reported effectiveness and efficiency [1, 196, 198]. The right most extension mechanism is described in further detail in the remainder of this subsection.

The right most path extension method is reported to be complete in that all valid candidates are enumerated once (it is a non-redundant method) [1, 196, 198]. The right most extension method views candidate graphs in the form of trees. The tree is "grown" by: (i) adding backward edges from the rightmost vertex, and then (ii) by attaching new nodes only to the vertices along the "rightmost path", the dashed lines indicate the newly added edges. This is illustrated in Figures 2.6 and 2.7. Where the red vertices indicate the rightmost path. Figure 2.6 gives a current $k-$edge graph while Figure 2.7 gives the possible one edge extensions of this graphs to give a sequence of

$(k+1)-$edge subgraphs. Note that backward edges[4] can only grow from the rightmost vertex, while forward edges can grow from vertices on the rightmost path.

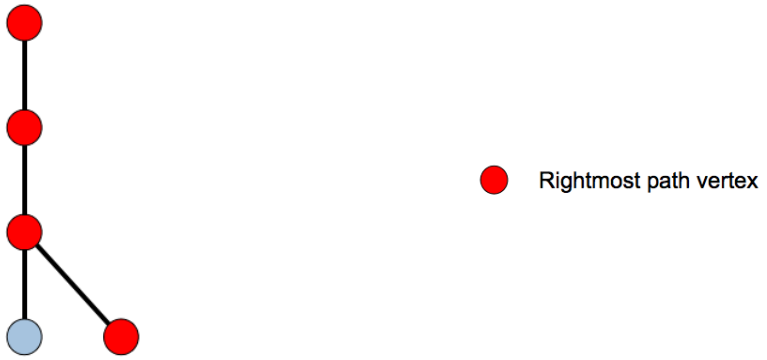



FIGURE 2.6: k-edge subgraph (k=4).

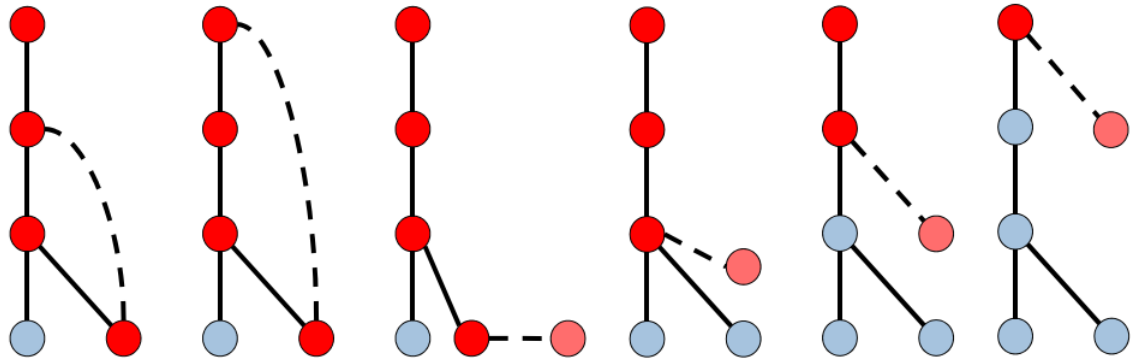


FIGURE 2.7: (k+1)-edge subgraphs generated by right most extension from the k-edge subgraph given in Figure 2.6.

#### 2.3.3.2 Frequent Subgraph Mining Algorithms

As noted above, FSM can be directed at either: (i) a set of transaction graphs or (ii) one large single graph. Broadly FSM techniques can also be divided into two categories according to the search strategy adopted: (i) Breadth First Strategy (BFS) also referred to as the Apriori approach; and (ii) Depth First Strategy (DFS) also referred to as the pattern growth approach. These two categories are similar in spirit to their counterparts found in Association Rule Mining (ARM), namely the Apriori algorithm [4] and the FP-growth algorithm [93] respectively.

[4] A forward edge is an edge that extends to a new end node while a backward edge extends to an existing end node.

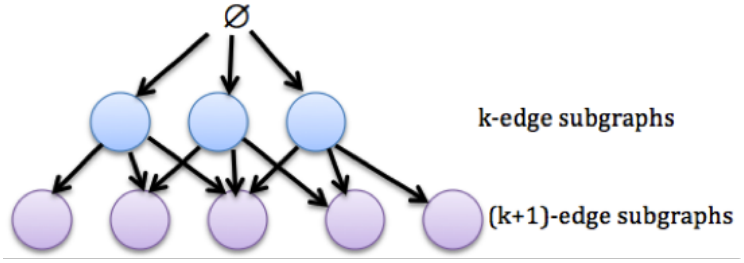



FIGURE 2.8: Apriori-based (BFS).

The Apriori approach is illustrated in Figure 2.8. Note that each node corresponds to a k-edge subgraph. The figure shows that the search for frequent subgraphs starts with 1-edge subgraphs and then moves on to 2-edge subgraphs and so on in a BFS manner. The new (k+1)-edge subgraphs are generated by growing the previously identified k-edge subgraphs by using, for example, right most extension. The frequency of the newly formed (k+1)-edge subgraphs is then checked and only frequent subgraphs are extended. The whole process iterates in this manner level by level as k is increased. Essentially, the Apriori-based approach proceeds in a "generate-and-test" manner using a BFS strategy to explore the subgraph lattice of the given graphbase.

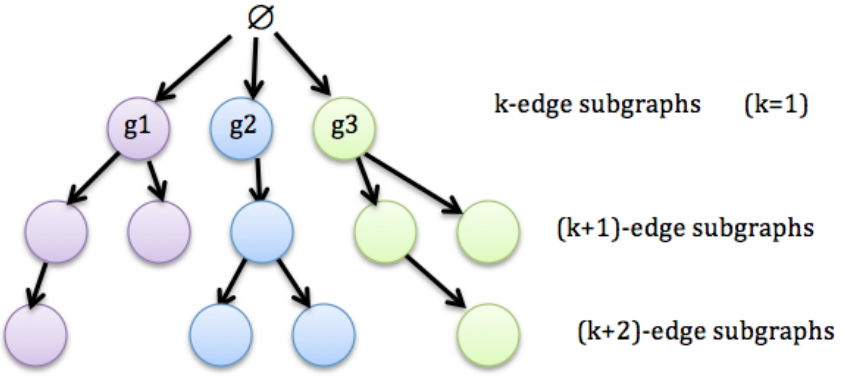



FIGURE 2.9: Pattern-growth (DFS). Only frequent $K$-edge subgraphs will be grown to $(K+1)$-edge subgraphs.

In Figure 2.9 the DFS (pattern growth) approach is shown. Note again that each node corresponds to a k-edge subgraph. For each discovered subgraph $g$, using the DFS strategy, $g$ is extended recursively until all the frequent supergraphs of $g$ have been discovered [229]. All subgraphs containing $g1$ (coloured in purple) are generated, then all subgraphs containing $g2$ (coloured in blue) are generated excluding these containing $g1$ and so on. In this manner the pattern growth approaches can avoid duplicate subgraphs efficiently. The approach also requires less memory than the Apriori approach. Moreover, as opposed to Apriori approaches; pattern growth approaches do not need to generate all the k-edge subgraphs first so as to explore (k+1)-edge subgraphs. Therefore a pattern growth approach DFS strategy was adopted for VULS candidates generation and BFS strategy was adopted for identifying VULS from VULS candidates with respect to this thesis.

Table 2.2: Frequent subgraph mining algorithm categorisation [122, 140]

| | | |
|---|---|---|
| Transaction graphs | BFS strategy | AGM [116] |
| | | AcGM [117] |
| | | FSG [136] |
| | | gFSG [139] |
| | | Farmer [167] |
| | | ISG[206] |
| | | MUSE[244] |
| | | DPMine [89] |
| | DFS strategy | MoFa [22] |
| | | gSpan [229] |
| | | ADI-Mine [220] |
| | | FFSM [109] |
| | | SPIN [111] |
| | | TSP[104] |
| | | RP-FP[144] |
| | | RP-GD[144] |
| | | JPMiner[150] |
| | | MSPAN[146] |
| | | GASTON [168] |
| Single graph | HSIGRAM [138] | |
| | VSIGRAM [138] | |
| | FPF [187] | |
| | DPMine [89] | |
| | SUBDUE [45] | |
| | Grew [137] | |
| | gApprox [36] | |
| | RAM [239] | |

Table 2.2, taken from [122, 140], gives a categorisation of a range of well known FSM algorithms according to whether they were intended for transaction graph FSM or single graph FSM; and in the case of transaction graph FSM, whether they operate in a BFS or a DFS manner.

The remainder of this section gives a detailed overview of the gSpan algorithm because of its relevance with respect to the research on VULS mining presented later in this thesis.

**Algorithm 2** gSpan($\mathbb{G}$,$\sigma$).

Sort the labels in $\mathbb{G}$ by their frequency;
Remove infrequent vertices and edges from $\mathbb{G}$;
Relabel the remaining vertices and edges in $\mathbb{G}$ according to frequency;
$\mathbb{S} = \{\}$;
$\mathbb{S}^1 \leftarrow$ All frequent 1-edge subgraphs in $\mathbb{G}$;
Sort $\mathbb{S}^1$ in DFS lexicographic order;
$\mathbb{S} = \mathbb{S} \cup \mathbb{S}^1$;
**for each** edge $e \in \mathbb{S}^1$ **do**
    $s = \{\}$;
    $s = s \cup e$; /* $s$ is the set of all subgraphs contain $e$ */
    $\mathbb{S} = \mathbb{S} \cup$ Subgraph_Mining($\mathbb{G}$,$s$); /* Algorithms 3 */
    $\mathbb{G} = \mathbb{G} - e$; /* Remove $e$ from graph dataset $\mathbb{G}$ */
    **if** The number of graphs in $\mathbb{G} < \sigma \times |\mathbb{G}|$ **then**
        exit;
    **end if**
**end for**

**Algorithm 3** Subgraph_Mining($\mathbb{G}$,$s$)

**if** $s \neq mininmalDFScode(s)$ **then**
    **return null;**
**end if**
Enumerate s in each graph in $\mathbb{G}$ by right most extension new edge to $s$ and count its children c's occurrence support(c);
**for each** c **do**
    **if** support(c) $\geqslant \sigma$ **then**
        $s = s \cup c$;
        Subgraph_Mining($\mathbb{G}$,$s$);
    **end if**
**end for**
**return** $s$**;**

In the context of transaction graph mining, given a database $\mathbb{G}$ comprised of a collection of graphs and a threshold $\sigma$ $(0 < \sigma \leq 1)$; where $\mathbb{G} = \{\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_n\}$ and $\mathbb{G}_i$ represents a graph in the graph dataset $\mathbb{G}$. The frequency of a subgraph $g$ in $\mathbb{G}$ is defined by Occurrence($g$) $= |\{\mathbb{G}_i \in \mathbb{G} | g \subseteq \mathbb{G}_i\}|$. The support of a graph $g$ is defined as the fraction of the graphs in $\mathbb{G}$ to which $g$ is subgraph isomorphic.

$$support(g) = Occurrence(g)/|\mathbb{G}| \tag{2.2}$$

A subgraph $g$ is frequent if and only if support($g$) $\geqslant \sigma$. The objective of FSM algorithms such as gSpan (graph-based Substructure pattern mining) algorithm [229] is

to find all frequent subgraphs in $\mathbb{G}$. gSpan is a DFS transaction graph FSM algorithm. The basic gSpan algorithm is given in Algorithm 2 [229]. The input is a graph dataset $\mathbb{G}$ and a minimum support threshold $\sigma$. The output is the set of identified frequent subgraphs $\mathbb{S}$. Note that the concept of frequency is generic regardless of whether we are considering FSM with respect to transaction graphs or one single large graph (as in the case of the proposed VULS mining). In both cases frequency is defined using a threshold $\sigma$. The distinction is in how the threshold is defined. In transaction graph mining it can be expressed simply as a percentage of the transaction graphs in which a candidate subgraph needs to exist for it to be frequent. This is not so straightforward in the context of a single graph mining where $\sigma$ needs to be calculated in some other way or user specified.

Returning to Algorithm 2, assume we have a label set of {A,B,C, ...} for the vertices, and a label set of {a,b,c, ...} for the edges. In Algorithm 2, lines 8-16, on the first iteration we will discover all the frequent subgraphs containing an edge $\langle A, a, A\rangle$. On the second iteration we will discover all the frequent subgraphs containing $\langle A, a, B\rangle$, but not $\langle A, a, A\rangle$. This procedure repeats until all the frequent subgraphs are discovered. Algorithm 3 is recursively called to grow the graphs and find all their frequent descendants. Algorithm 3 starts (line 1) with a check whether DFS code of subgraph $s$ is minimal. If the DFS code of subgraph $s$ is not minimal, $s$ has been discovered before (it is the duplicate subgraph), thus return null, otherwise keep on generating supergraphs of $s$ by right most extension. The Subgraph_Mining procedure stops searching either when support of the subgraph is less than $\sigma$, or its code is not a minimum code, which means this subgraph and all its descendants have been discovered before.

The VULS classifiers presented later in this thesis "borrow" three techniques from gSpan: (i) the use of the canonical form, more specifically minimal DFS coding as described in section 2.2.4 above; (ii) the candidate generation mechanism, more specifically right most extension as described in section 2.3.3.1; and (iii) the DFS strategy as introduced above. Private correspondence with the author (XiFeng Yan) of gSpan confirms that since gSpan has subgraph isomorphism checking embedded within in it. Thus the time complexity of gSpan can't be computed in any meaningful manner.

## 2.4 3D Surface Representation Techniques and Grid Graphs

This section introduces the concept of surface representation, used with respect to various application domains such as: medicine and bioinformatics [54, 179, 209, 235], manufacturing [98, 103] and especially computer graphics in the context of animation and video gaming [99, 102]. The significance is, that as noted in the introduction to this thesis, the AISF primary application domain of interest involves 3D surfaces (as does the secondary satellite image interpretation domain).

There are many ways of representing 3D surfaces. The most straight forward is a point cloud [134], these can be used in the context of both the sheet metal forming

and the satellite image interpretation applications as will be described in chapter 3. A point cloud is simply a collection of unordered and unorganised data points, P = $\{p_i = (x_i, y_i, z_i) | 1 \leqslant i \leqslant n\}$. A variety of techniques are available to generate such point clouds, these include: (i) laser and optical scanners, (ii) 3D digitizers (coordinate measuring machines), (iii) some Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) systems and (iv) range data converters[5] such as in [5, 214]. In the case of the work described in this thesis concerning sheet metal forming the clouds of interested were generated either using a CAD system or an optical measuring system[6].

Although point cloud representations are simple there is often a need, depending on a particular application domain, for a higher level representation. A wide range of 3D representation techniques have been proposed. An overview of these different representations is presented in this section. These are categorised as follows: (i) Mathematical representations (Parametric and Implicit representation)[151, 201, 202], (ii) Mesh (Polygonization) representations (grid graph representation), and (iii) other representations such as Constructive Solid Geometry (CSG) [223], Boundary Representation (B-Rep) [178] and Voxel Representation[87, 126].

Most 3D surface representations listed above are directed at visualisation and not further interpretation. These types of representations are therefore not well suited to surface representation for springback prediction and satellite image interpretation (the application focus with respect to this thesis). Note that in the context of the work described in this thesis we are interested in 3D representations that are not only able to capture the geometrical information contained in a given surface but also support the application of classification techniques. Mesh (Polygonization) representations meet such needs because they provide a useful mechanism for representing 3D surfaces in such a way that they can be mined (as will be demonstrated later in this thesis). A regular mesh grid representations, more precisely a square grid representation, was adopted with respect to this thesis, and hence mesh representations are considered in some further detail in the remainder of this section.

A Mesh representation, of a given 3D object, is defined in terms of a collection of 0-dimensional cells (vertices), 1-dimensional cells (edges) and 2-dimensional cells (facets) in $R^3$. Thus, an object is defined by a pair of ordered lists:

$$\langle \mathbb{P}, \mathbb{V} \rangle$$

where:

- $\mathbb{P} = \{p_1, p_2, \ldots, p_m\}$ is a set of polygons described in such a way that $p_i = v_{i1}, v_{i2}, \ldots, v_{ij}$ is the polygon comprised of j vertices (thus if $j = 3$ we would have a triangular polygon [31], see Figure 2.10). The set $\mathbb{P}$ is normally used to describe topological information (information about the connectivity between $\mathbb{V}$). In

[5]Range data is 2D data where each "pixel" value describes the distance between the points in a 3D scene (object) to a specific point such as a camera. It is sometimes considered to be a special form of 3D data referred to as "2.5D" data.

[6]The actual system used was a GOM (Gesellschaft fur Optische Messtechnik).

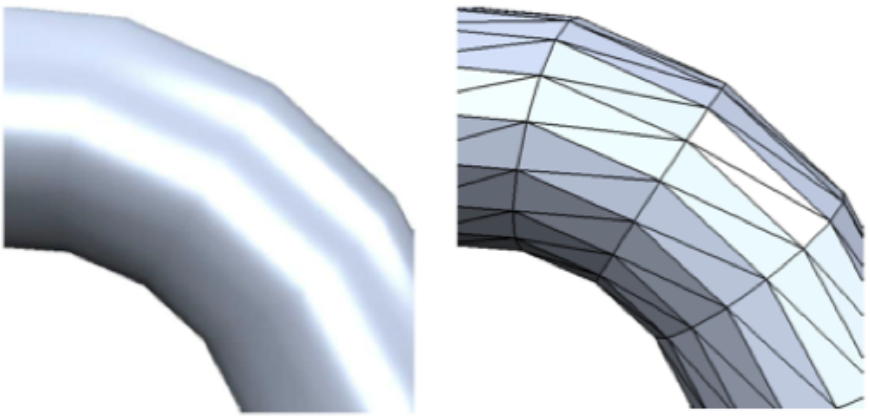


FIGURE 2.10: Triangular mesh representation using $j = 3$ [134].

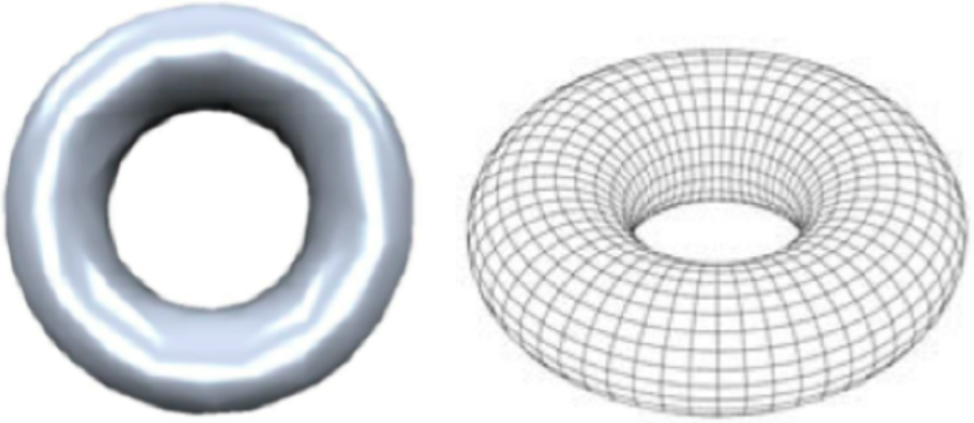


FIGURE 2.11: Rectangular mesh representation using $j = 4$ [134].

other words, the mesh can be seen as unstructured grid. For the work presented in this thesis, $j$ was set to 4 indicating the usage of square polygons with respect to a "regular mesh grid" (or square mesh as shown in Figure 2.11) as will be described further in chapter 3.

- $\mathbb{V} = \{v_1, v_2, \ldots, v_n\}$ is a set of n vertices defined in terms of a 3D coordinate system where $v_i = (x_i, y_i, z_i)$.

Constructing a mesh representation from a point cloud is a well known problem in the field of 3D geometrical modelling referred to as "surface reconstruction". Surface reconstruction has been widely investigated in the context of various domains such as reverse engineering (reconstructing a representation given a visualisation of a 3D object) and face recognition. Alternatively the polygons that frequently make up mesh representations can be generated from parametric mathematical representations [135] and from implicit mathematical representations [169]. Recall that the representation techniques used with respect to this thesis should facilitate graph translation (as well as VULS vertex classification). Mesh representations inherently support such graph conversion. The popularity of mesh representations comes from their simplicity and high manageability

in the context of 3D design (easy to define and modify). With respect to the work described in this thesis, the advantages of the mesh representation are incorporated into a uniform equal grid representation where the centre points of these grids represent the set $\mathbb{V}$. Using a regular grid the connectivity information can be easily attained, therefore the regular grid (mesh) representation adopted in this thesis is considered to be a special form of the mesh representation without the set $\mathbb{P}$. More specifically a regular (square) grid representation was adopted because:

1. It was compatible with the CAD/CAM systems used to generate desired shape descriptions in the context of sheet metal forming.
2. It was simple to generate and process.
3. When combined with other representation forms (such as grid graphs) it readily supported edge connection (as will be demonstrated later in this thesis).
4. It facilitated the capture of local geometries (again as will be demonstrated later in this thesis).
5. It provided a useful mechanism whereby geometrical information could be stored (and subsequently used or reused).
6. It provided flexibility in that it supported translation into other forms such as grid graphs.

As noted above further detail of the grid (mesh) representation used with respect to this thesis will be presented in Chapter 3. However, it should be noted here that grid representations have been used elsewhere, for example with respect to texture analysis in the context of satellite image processing [87, 126] and terrain analysis in Geographical Information System (GIS) [153]; thus adding to the case for their adoption with respect to the work presented in this thesis. Of course the effectiveness of mesh representations can be influenced by surface complexity. However, in the case of the 3D surfaces considered in this thesis, surfaces representing objects that can be produced using AISF and surfaces representing satellite images, this is not an issue.

An additional advantage of regular square grid mesh representations, of specific interest with respect to the VULS mining proposed in this thesis, is that they can easily be represented as grid graphs. More specifically graphs where each vertex represented a grid square (with a value associated with it) which is connected to its immediate neighbouring grid squares by edges labelled with slope in each case. The advantage offered by this representation is that it lends itself to the concept of VULS mining to identify unique subgraphs that have a particular error pattern (vertex labelling) associated with them. Given a new shape $S$ the identified VULS can then be used to predict a resulting shape $S'$ as a result of applying some process $P$. Of course good coverage of $S$ is necessary so as to generate effective predictions. Further detail concerning grid graphs will be presented in Chapters 3 and 4.

Finally, at the end of this section, it should also be noted that there are alternative mesh representations, to that proposed and utilised in this thesis, that have been adopted elsewhere with respect to research related to AISF, such as the work presented in [149, 159]. However the work presented in [149, 159] was not directed at using data mining (vertex classification) techniques to predict springback. It is the view of the author that usage of a regular square grid mesh is best suited to the derivation of grid graphs to which VULS mining, and by extension vertex classification, can easily be applied; than the more complex alternative mesh representations used in [149, 159].

## 2.5 Classification

A central theme of this thesis is classification, specifically vertex classification. Classification is concerned with the construction of a "classifier" which can be used to label previously unseen data. Classification algorithms can be categorised as being: (i) supervised, (ii) unsupervised and (iii) semi-supervised. Supervised learning requires the availability of (costly) pre-labelled training data with which to build (train) a classifier. Unsupervised learning requires no training data but tends to produce less accurate classifiers. Semi-supervised learning is where use is made of both labelled and unlabelled data for training purposes, typically a small amount of labeled data is used together with a large amount of unlabelled data. Semi-supervised lies somewhere between unsupervised learning (without any labelled training data) and supervised learning (with only labeled training data).

The single graph vertex classification sub-domain can be divided into two application dependent approaches: (i) semi-supervised learning and (ii) supervised learning. In the case of semi-supervised learning we consider a single input graph which is partially labelled and the objective is to build a classifier, using the vertices with known classes, to predict the labels for the remaining vertices. From the literature there are many reported techniques directed at semi-supervised learning for vertex classification, such as: (i) graph regularization [15, 53] (where the graph is represented as a weighted matrix), (ii) methods based on the amount of "functional flow" through the vertices [164], (iii) global graph consistency methods [125, 219], (iv) methods that use Markov Random Fields [56], (v) methods that use Gaussian Random Fields [163, 212] and (vi) more recently, methods founded on kernelized score functions (for vertex ranking) [175]. Supervised learning, in turn, is founded on the use of one or more fully vertex pre-labelled training graphs which are then used to train a classifier to be applied to previously "unseen" graphs (graphs without vertex labels). So far there is limited research directed at supervised vertex classification. Existing techniques include: (i) "robust vertex classification" using a sparse signal representation [37] and (ii) universally consistent vertex classification for latent position graphs [199]. The work presented in this thesis falls into the supervised single graph vertex classification category and this is therefore discussed further in this section.

The rest of this section is organised as follows. As noted above, vertex classification is typically conducted in either a semi-supervised or supervised manner, the first is discussed in Section 2.5.1 while the second is discussed in Section 2.5.2. More general classification, as opposed to vertex classification, is of significance in the context of this thesis because, for comparison purposes, a number of traditional classifiers are applied to the vertex classification problem. More specifically J48 and Naive Bayes are used. A brief overview of these is thus given in Section 2.5.3.

### 2.5.1 Semi-supervised Vertex Classification

In the context of semi-supervised vertex classification, the problem of labelling a partially labeled graph is considered. This general problem arises in a number of different settings. For example, in survey sampling, one has a database of individuals along with their preference profiles that determines a graph structure based on similarity of preferences. One wishes to estimate a survey variable (e.g. hours of TV watched, amount of cheese consumed, etc.). Rather than survey the entire set of individuals every time, which might be impractical, one may sample a subset of the individuals and then attempt to infer the survey variable for the rest of the individuals. There has been an increase of interest in semi-supervised learning recently, because of the increase in the number of datasets with large amounts of unlabelled examples and only a few labelled ones. For instances, [53] proposes a non-parametric algorithm using a function induction method that minimizes a regularization criterion applied to an out-of-sample example.

In label prediction problems vertices represent partially labeled instances and the edges pairwise similarities among vertices; the aim is then to label the unlabelled part of the graph by exploiting the topology of the network and the a priori knowledge encoded in the labelled vertices [17]. A variety of methods have been proposed for vertex label classification, examples include: (i) algorithms based on the "guilt-by-association" principle [156, 176], (ii) methods focused on clustering [189, 242] and (iii) methods based on random walks [10, 16, 56, 120, 241, 243]. The main differences between the various semi-supervised vertex classification algorithms, such as spectral methods, random walks, graph mincuts and transductive SVM; lie in their way of realizing the assumption of consistency. Furthermore, some semi-supervised vertex classification techniques may be adapted to supervised vertex classification.

### 2.5.2 Supervised Vertex Classification

In the context of supervised vertex classification, classifiers are generated from training graph data (with both vertex and edge labels). The generated classifiers are then used to predict class labels of vertices in "unseen" graphs. Exemplar techniques include: (i) robust vertex classification via sparse signal representation [37], (ii) universally consistent vertex classification [199], (iii) Bayesian network [86], (iv) Probabilistic Model [56], (v) Markov random walks [56], (vi) SVM (Support Vector Machines) [212] and (vii) the random dot product graph model [194]. The work presented in this thesis is directed

at supervised vertex classification. However, most existing vertex classification methods operate in a very different manner to graph mining (the theme of this thesis). Thus VULS vertex classification proposed in this thesis thus represents a novel approach to the vertex classification problem. For the same reason, it is difficult to compare our VULS classifiers with existing vertex classification methods, therefore, some alternative existing classification approaches have been chosen with which to compare the proposed VULS classifiers, this will be discussed further in the next Subsection.

### 2.5.3 Classification Techniques

As noted above, later in this thesis some comparison is conducted using more standard forms of classification, typically directed at tabular data. More specifically, the J48 decision tree classification and Naive Bayes classification algorithms are utilised because these are popular exemplars of "standard" classification algorithms. Some brief details of these two algorithms is therefore presented in the following two Sections.

#### 2.5.3.1 J48.

The J48 algorithm is a WEKA[7] implementation of the ID3 (Iterative Dichotomiser 3) Decision Tree (DT) algorithm. With respect to the work described in this thesis the J48 algorithm [174] was adopted as it has been considered to be a benchmark DT classifier throughout the data mining community. J48 uses Information Gain (IG) as the splitting criteria whereby the attribute with the highest information gain is selected to be used at the current node. IG(A) is a measure of the difference in entropy from before to after the set S is split on an attribute A. In other words, how much uncertainty in S was reduced after splitting set S on attribute A. IG is calculated using Equation 2.3:

$$IG(A, S) = Entropy(S) - \sum_{t \in T} P_t \times Entropy(t) \tag{2.3}$$

where (i) Entropy(S) is the entropy of set S, (ii) T is the subsets created from splitting set S by attribute A such that $S = \bigcup_{t \in T} t$, (iii) $P_t$ is the proportion of the number of elements in t to the number of elements in set S, and (iv) Entropy(t) is the entropy of subset t. Entropy(D) is a measure of the amount of uncertainty in the (data) set D. It is calculated using Equation 2.4.

$$Entropy(D) = \sum_{i=1}^{i=|c|} -P_i log_2 P_i \tag{2.4}$$

where $P_i$ is the probability of class $i \in c$. Normally, $P_i = \frac{|C_i, D|}{|D|}$ where $|C_i, D|$ is the number of records corresponding to class i with respect to the entire data set D. Intuitively, $0 \leqslant Entropy(D) \leqslant 1$. Entropy is thus a measure of the homogeneity of a given data set. If Entropy(D)= 0, then all the records belong to the same class and therefore the

[7] http://data-mining.business-intelligence.uoc.edu/home/j48-decision-tree

outcome is certain. On the other hand, if Entropy(D)= 1 this would mean that the data set is totally homogeneous and all classes are equally likely. IG is thus a measure of the expected reduction in the entropy for a given attribute. In other words IG indicates the "importance" of a given attribute with respect to the DT construction process. In the context of Equation 2.3 the importance of an attribute is determined by identifying the entropy value of the attribute before and after splitting. The same calculation is made for the complete set of attributes and the attribute that maximises information gain is selected for the DT node in question. The interesting reader can find further detail and a worked example in [173].

#### 2.5.3.2 Naive Bayes.

Bayesian classifiers are statistical classifiers based on the well known Bayes probability theorem:

$$P(H|A) = \frac{P(A|H)P(H)}{P(A)} \tag{2.5}$$

where $H$ is a "hypothesis" and A is an "evidence". The probability of the hypothesis $H$ holding given the evidence A is denoted by the conditional probability $P(H|A)$ and is called the posterior probability. In the context of classification, A is a vector of $n$ features whereby $A = \{a_1, a_2, \ldots, a_n\}$. Similarly, the conditional probability $P(A|H)$ is the posterior probability of a given $H$. $P(H)$ and $P(A)$ are called the prior probability of the hypothesis $H$ and the feature vector $A$ respectively. The prior probability is the distribution probability without considering any event, while the posterior probability considers the event $H$. Bayes theorem assumes that all the attributes in a feature vector are independent, this is why it is sometimes referred to as Naive Bayes; however, the assumption simplifies the calculation. Given a training set of $n$ records $T = \{t_1, t_2, \ldots, t_n\}$ where each $t_i$ comprises $l$ attributes $t_i = \{a_1, a_2, \ldots, a_l\}$. Suppose that there are $m$ classes where $t_i$ belongs to class $k$ if and only if the posterior probability $P(C_k|t_i)$ has the highest probability value amongst other classes:

$$P(C_k|t_i) > P(C_j|t_i), 1 \leqslant j \leqslant m, \text{ and } j \neq k \tag{2.6}$$

Recall that $P(C_k|t_i)$ is calculated using Equation 2.5. A Bayesian classifier is constructed as follows [95].

1. Estimate the prior probability $P(C_k)$ of each class, $P(C_k) = \frac{|C_k,T|}{n}$, where $|C_k, T|$ is the number of tuples in $T$ that belongs to class $C_k$ and $n$ is the total number of tuples in the training set. The prior probability $P(H)$ is constant for all classes.

2. Maximise the posterior probability$P(t_i|c_k)$ by finding the total product of the posterior probability of each attribute in $t_i$ individually as follows[8].

[8]In case of continuous attribute values, the Gaussian distribution is assumed along with a mean $\mu$ and standard deviation $\sigma$ to be used for the calculation of probability. $P(x \mid C_k) = f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$. See [94] for more details.

$$P(t_i|C_k) = \prod_{i=1}^{l} P(a_i|C_k) = P(a_1|C_k) \times P(a_2|C_k) \times P(a_3|C_k) \times \cdots \times P(a_l|C_k) \quad (2.7)$$

3. Finally, calculate $P(t_i|C_k)P(C_k)$ for all $m$ classes. The class $C_k$ that has the highest $P(t_i|C_k)P(C_k)$ value associated with it is selected as the predicted class.

The main limitation of Bayesian classifiers is the Naive Bayes assumption. However, their main advantages are their simplicity and computational efficiency; they only require a single scan of the training data and provide a fast classification of new unlabelled cases. [95].

## 2.6 Evaluation Criteria

One of the aims of the work presented in this thesis was to investigate the utility of the proposed VULS mining in terms of vertex classification. As noted in the introduction to this thesis this investigation was conducted with respect to two application domains (i) springback prediction in the context of sheet metal forming and (ii) ground type identification with respect to satellite image interpretation. In the context of vertex classification the proposed VULS classifiers were evaluated individually and comparatively using the following metrics:

- **Individually**: Using accuracy and Area Under ROC Curve (AUC).
- **Comparatively**: First by comparing collated AUC values, and then statistically by applying the Friedman and the Nemenyi tests to demonstrate whether there was a statistically significant difference between the operation of the proposed classifiers.

These measures are reviewed in this section. This section is organised as follows. Subsection 2.6.1 presents an overview of the evaluation measures used with respect to the individual evaluations (accuracy and AUC). An overview of statistical significance testing is presented in Subsection 2.6.2. The Freidman Test, the statistical comparison technique adopted with respect to the work presented in this thesis, is then described in Subsection 2.6.3.

### 2.6.1 Accuracy, AUC, TCV and SD

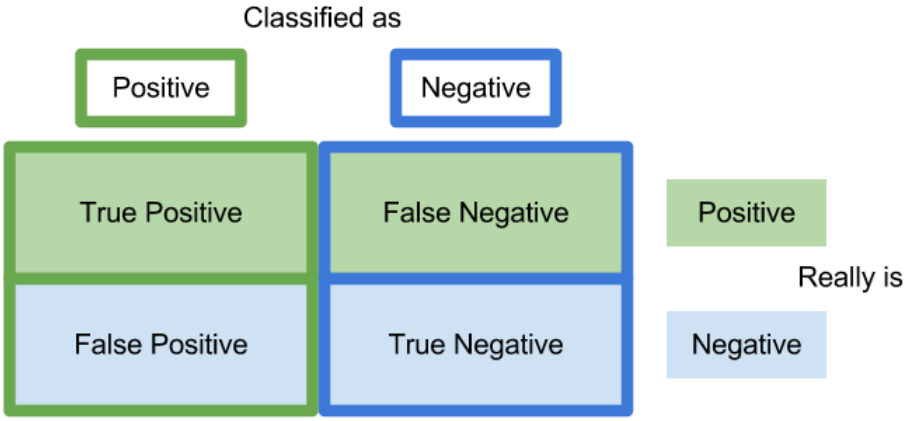



FIGURE 2.12: Confusion matrix.

There are many mechanisms for assessing the effectiveness of classifiers; as noted above, with respect to the work presented in this thesis, accuracy and AUC were used because the usage of these measures is frequently reported in the literature [112, 147]. Alternative methods include sensitivity and specificity. Accuracy is calculated simply as shown in Equation 2.8:

$$accuracy = \frac{number\ of\ records\ correctly\ classified}{total\ number\ of\ records} \tag{2.8}$$

However, although accuracy provides for an easily understandable measure of the overall quality of a classifier (hence its usage in this thesis), it does not take into consideration the distribution of the classes. In this respect AUC is a more appropriate measure [23, 97]. Broadly, the ROC curve concept was originally used in signal detection theory to depict the trade-off between hit rates and false alarm rates. The "hit rate" is called the True Positive Rate (TPR), benefit or sensitivity; while the "false alarm rate" is called the False Positive Rate (FPR), or cost. Both are expressed in the form of a real number ranging from between 0.0 and 1.0. TPR and FPR are calculated using the concept of what is known as a *confusion matrix* as shown in Figure 2.12. Confusion matrices are used with respect to two class problems. With reference to Figure 2.12: (i) the True Positives (TP) value is the number of instances that are correctly classified as belonging to the Positive class, (ii) the False Negatives (FN) value is the number of instances belonging to Positive class that are erroneously predicted as belonging to Negative class, (iii) the True Negatives (TN) value is the number of instances that are correctly classified as belonging to Negative class and (iv) the False Positives (FP) value is the number of instances belonging to Negative class that are erroneously predicted as belonging to Positive class. Using a confusion matrix TPR and FPR are calculated as shown in Equations 2.9 and 2.10 respectively. Note that accuracy can also be derived from a confusion matrix using Equation 2.11.

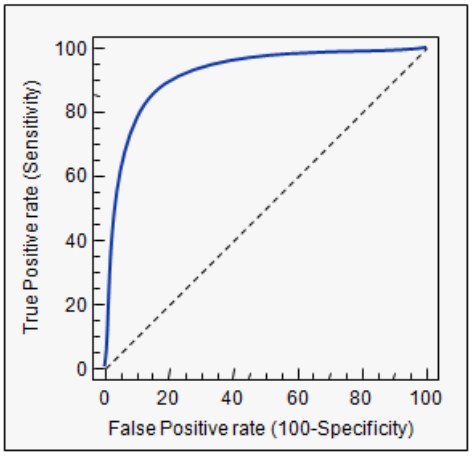



FIGURE 2.13: The ROC curve. The solid blue line indicates a good ROC curve that reaches the upper left corner and the dotted line indicates a random classifier (guessing).

$$TPR = \frac{TP}{TP + FN} = sensitivity \tag{2.9}$$

$$FPR = \frac{FP}{TN + FP} = 1 - specificity \tag{2.10}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.11}$$

A ROC curve is generated by plotting the FPR (False Positive Rate) against the TPR (True Positive Rate) (with the FPR plotted along the X-axis and the TPR along the Y-axis). Both TPR and FPR range from 0 to 1[9]. In the ROC space, the best classification performance exists in the upper left corner (where FPR=0 and TPR=1) while the diagonal represents random classification (guessing); as shown in Figure 2.13. Therefore, a "good" ROC curve is one that reaches the upper left corner.

The Area Under a ROC curve (AUC) is a single value frequently used to measure classifier performance ($0 \leq AUC \leq 1$). In other words AUC is an indicator of the probability that a classifier will correctly classify instances [9, 113, 148, 161]. Note that an AUC value of 0.5 indicates a random classifier (guessing). To illustrate the distinction between accuracy and AUC, consider a 2-class problem where class 1 has 990 instances and class 2 has 10 instances, then the accuracy of the generated model by simply guessing class 1 would be $\frac{990}{990+10} \times 100 = 99\%$; "on the face of it" a good accuracy value. However, a classifier that does this is clearly not a good classifier as indicated by the AUC= 0.5 that would describe this situation. Thus the main advantage of AUC is its ability to deal with unbalanced data sets since it considers the distribution of classes (TPR and FPR values) [73]. Therefore, AUC was chosen to be the other performance measure used with respect to the proposed classifiers presented in this thesis because of

[9] https://www.youtube.com/watch?v=OAl6eAyP-yo

the uneven vertex label distributions within the evaluation datasets. The further detail of AUC can be found in [147].

However, with respect to the evaluation data sets used these featured more than two classes, hence the above confusion matrix based approach to AUC calculation was inappropriate. Instead the Mann-Whitney-Wilcoxon (MWW) statistical method, which employs a ranking concept based on the signal detection theory proposed by [96], was used with respect to the work described in this thesis to calculate AUC values[10]. A full example on how to calculate the AUC value, based on the MWW statistic, is presented in Appendix A.

For the presented evaluations Ten Cross Validation (TCV) [192] was adopted, where appropriate, in order to reduce the likelihood of overfitting [79]. Overfitting mainly occurs when a generated classifier (model) is fitted to the training data in such a perfect manner that the resulting classifier is not suited to classifying anything else (thus defeating the objective of generating the classifier in the first place). TCV is used in order to limit the implication of overfitting. TCV is a well established technique for evaluating the performance of supervised learners whereby the data is divided into ten parts so that class labels are distributed equally (stratified). Using the TCV technique the learner is applied ten times, each time to a different 9/10 of the data set, and tested using the remaining 1/10. On completion, the recorded results of the ten iterations are used to compute an averaged set of results.

Note that in this thesis, when reporting average values, such as those generated using TCV, the associated Standard Deviation (SD) is also reported. SD is a measure of how much variation exists with respect to a given average value. A low SD indicates that the values are close to the average. A high SD indicates that the values are spread out over a large range of values.

### 2.6.2 Overview of Statistical Performance Comparison

Given a collection of classifiers and a particular data set, one will usually produce a better accuracy and AUC then the others. The question is whether this is statistically significant or not. There are many techniques for statistical comparison. This section provides some background concerning the most common (popular) techniques used to perform the statistical comparison of competing classification techniques, and explain the reason why the Friedman's Test (presented in the following Section) was used in this thesis.

A number of different approaches have been proposed to conduct statistical comparison between the operation of different classifiers. Typically, these approaches can be categorised as being either: (i) parametric or (ii) non-parametric. The first is used when the distribution of the data set is drawn from a normal (Gaussian) distribution. The second makes no assumption about the distribution of the data set (Distribution-Free).

[10]The AUC/ROC calculation conducted using the well known Weka data mining workbench is also done using the Mann Whitney statistic [224].

The size of the data sets also has an impact on whether parametric or non-parametric analysis is conducted. For large data sets, both the parametric and the non parametric statistical tests have the same implication (in other words, there is no difference between the parametric and non-parametric tests for large data sets). Several forms of statistical test have been proposed with respect to both parametric and non-parametric testing. For comparison between only two classifiers over different data sets, the paired t-test has been proposed for parametric data, while the Wilcoxon Signed-Rank Test has been proposed for non-parametric data. For more details and examples concerning theses tests see [55].

To determine whether there is a statistical difference between the operation of more than two classifiers the Analysis Of Variance (ANOVA) and Friedman test have been extensively used [190]. The ANOVA statistical test is based on two assumptions: (i) the normal distribution of the classification results and (ii) that the data sets used have equal variance (the homogeneity of variance) [55]. Although both assumptions cannot be guaranteed, the violation of them would cause a greater effect on any post-hoc testing. Thus the ANOVA statistical test is not recommended for classification analysis unless both assumptions are certainly satisfied. However, the Friedman test is mainly directed at non-parametric testing. The Friedman test offers two advantages over parametric techniques (such as ANOVA): (i) ease of computation and interpretation and (ii) its ability to demonstrate the classification performance in terms of ranks rather than vague averages [84]. The Friedman test was thus chosen to evaluate the performance of the different proposed techniques with respect to this thesis. In addition to the practical advantages offered by the Friedman test, it was also chosen because [82, 84, 213]:

- There is no guarantee that the AUC results obtained from the proposed techniques follow the normal (Gaussian) distribution (the data is thus assumed to be non-parametric).
- The Friedman statistical test is generally recommended (see for example [55]) for use with related data sets while the ANOVA test is recommended for unrelated data sets. With respect to the work described in this thesis the data sets used were considered to be related data sets (see Chapter 3).

### 2.6.3 Friedman's Test

This section describes the operation of the Friedman statistical test [29, 55, 81, 83, 143]. The Friedman test is commenced by ranking each classification technique according to their performance in ascending order with respect to each data set considered. The ranking can be done using AUC or accuracy (or any other value of interest), although AUC is used throughout this thesis. The mean rank of each classifier $j$, $AR_j$, was then computed across all the data sets. With $D$ representing the overall number of data sets, $K$ the overall number of classifiers, and $r_j^i$ the rank of classifier $j$ with respect to data set $i$, the Friedman test statistic was then calculated as follows:

$$\chi_F^2 = \frac{12D}{K(K+1))} \left[ \sum_{j=1}^{K} AR_j^2 - \frac{K(K+1)^2}{4} \right] \quad (2.12)$$

$$\mathrm{AR}_j = \frac{1}{D} \sum_{i=1}^{D} r_j^i \quad (2.13)$$

where $\chi_F^2$ is distributed according to the Chi-Square distribution with $K-1$ Degrees of freedom.

The null hypothesis, $H_0$, being tested was that there was no statistically significant difference between the operation of the compared classifiers. In other words that the performance differences observed with respect to (say) the reported AUC results was not statistically significant different, but simply due to random chance. If the value of $\chi_F^2$ is above a given threshold, then the null hypothesis that there is no difference in the operation of the classifiers can be rejected. The smallest level of significance that can result in the rejection of the null hypothesis is represented by a threshold value called the $p$-value. The $p$-value not only provides information about whether a statistical hypothesis test is significant or not, it also indicates "how significant" the result is. Note also that the smaller the $p$-value, the stronger the evidence against $H_0$. If $H_0$ can be rejected, a so-called post hoc test can be applied to detect which specific approaches differ significantly from the rest. In this respect *Demšar* [55] recommended the use of the Nemenyi test. The Nemenyi post-hoc test [166] was thus applied so as to identify significant differences (if any) between the individual approaches. Using the Nemenyi post-hoc test the performances of two or more approaches (classifiers) is significantly different if their average ranks differ by at least a Critical Difference value ($CD$), given by:

$$CD = q_{\alpha,\infty,K} \sqrt{\frac{K(K+1)}{12D}} \quad (2.14)$$

The value $q_{\alpha,\infty,K}$ is based on the "studentized" range statistic and is tabulated in standard statistical textbooks as shown in Figure 2.14. With respect to the evaluation presented here the outcome from applying the Friedman test and the Nemenyi post-hoc tests are presented using a modified version of *Demšar*'s significance diagrams [55, 143]. These diagrams display the ranked performances of competing classifiers, along with their critical difference, to clearly indicate those classifiers whose operation is significantly different from the other classifiers (in terms of recorded AUC value in the case of the evaluation reported in this thesis).

| # of models | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $q_{0.01}$ | 2.576 | 2.913 | 3.113 | 3.255 | 3.364 | 3.452 | 3.526 | 3.590 | 3.646 | 3.696 | 3.741 | 3.781 | 3.818 | 3.853 |
| $q_{0.05}$ | 1.960 | 2.344 | 2.569 | 2.728 | 2.850 | 2.948 | 3.031 | 3.102 | 3.164 | 3.219 | 3.268 | 3.313 | 3.354 | 3.391 |
| $q_{0.10}$ | 1.645 | 2.052 | 2.291 | 2.460 | 2.589 | 2.693 | 2.780 | 2.855 | 2.920 | 2.978 | 3.030 | 3.077 | 3.120 | 3.159 |

| # of models | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $q_{0.01}$ | 3.884 | 3.914 | 3.941 | 3.967 | 3.992 | 4.015 | 4.037 | 4.057 | 4.077 | 4.096 | 4.114 | 4.132 | 4.148 | 4.164 |
| $q_{0.05}$ | 3.426 | 3.458 | 3.489 | 3.517 | 3.544 | 3.569 | 3.593 | 3.616 | 3.637 | 3.658 | 3.678 | 3.696 | 3.714 | 3.732 |
| $q_{0.10}$ | 3.196 | 3.230 | 3.261 | 3.291 | 3.319 | 3.346 | 3.371 | 3.394 | 3.417 | 3.439 | 3.459 | 3.479 | 3.498 | 3.516 |

| # of models | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $q_{0.01}$ | 4.179 | 4.194 | 4.208 | 4.222 | 4.236 | 4.249 | 4.261 | 4.273 | 4.285 | 4.296 | 4.307 | 4.318 | 4.329 | 4.339 |
| $q_{0.05}$ | 3.749 | 3.765 | 3.780 | 3.795 | 3.810 | 3.824 | 3.837 | 3.850 | 3.863 | 3.876 | 3.888 | 3.899 | 3.911 | 3.922 |
| $q_{0.10}$ | 3.533 | 3.550 | 3.567 | 3.582 | 3.597 | 3.612 | 3.626 | 3.640 | 3.653 | 3.666 | 3.679 | 3.691 | 3.703 | 3.714 |

| # of models | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
|---|---|---|---|---|---|---|---|
| $q_{0.01}$ | 4.349 | 4.359 | 4.368 | 4.378 | 4.387 | 4.395 | 4.404 |
| $q_{0.05}$ | 3.933 | 3.943 | 3.954 | 3.964 | 3.973 | 3.983 | 3.992 |
| $q_{0.10}$ | 3.726 | 3.737 | 3.747 | 3.758 | 3.768 | 3.778 | 3.788 |

FIGURE 2.14: Critical values for the two-tailed Nemenyi test[11].

## 2.7 Summary

This chapter has presented the background to the work presented in this thesis. The chapter covered three main areas: (i) Graph mining, (ii) Vertex classification and (iii) 3D surface representation. An overview of graph mining techniques, especially single graph based FSM and transaction graphs based FSM were provided. 3D surface representation techniques, vertex classification techniques and traditional standard classifiers, such as J48 and Naive Bayes, were also summarised. Finally the criteria used to evaluate the operation of the proposed classifier was presented. It was noted that two types of evaluation were conducted: (i) individual evaluation for each proposed classifier using accuracy and AUC; and (ii) overall performance evaluation using statistical approaches. Both evaluation mechanisms were reviewed in this chapter. In the next chapter the two application domains with respect to the evaluation presented later in this thesis will be considered: (i) sheet metal forming (AISF) and (ii) satellite image interpretation.

[11] Source : `http://www.cin.ufpe.br/~fatc/AM/Nemenyi_critval.pdf`

# Chapter 3

# Application Domain and Data Sets

## 3.1 Introduction

This chapter describes the data sets used for evaluation purposes with respect to the work presented in this thesis. The data sets were drawn from two specific application domains: (i) sheet metal forming, and (ii) satellite image interpretation. However, it should be recalled that the primary application focus of the thesis was the sheet metal forming application, while the purpose of the satellite image application was to consider the broader applicability of the VULS concept. It should also be recalled that the identified input to the proposed VULS mining techniques, and the mechanism for using identified VULS for vertex classification, is a grid graph. Using grid graphs allows for comparison with more standard classification approaches. However, it should also be noted, as will become apparent later in this thesis, that the VULS context is applicable to other forms of graph than grid graphs. This point will be revisited when future work is discussed in chapter 9.

In the context of grid graphs, theses are extracted from grids (meshes) describing some 3D surface of interest. Each grid square in such a grid is defined by its centre point which is in turn defined by an X-Y coordinate pair. Each grid point also has a "Z" value associated with it. In the case of the sheet metal forming application this was a height above some reference point. In the case of the satellite image interpretation application this was a pixel intensity value. With respect to the training data used for generating VULS, and the test data used for vertex classification evaluation purposes, each grid cell also had a label of some form associated with it describing some feature of interest. In the case of the sheet metal forming application this was springback, in the case of the satellite image application this was ground type.

The raw data sets related to the two application domains were in very different formats (point clouds and images) thus the mechanisms required to translate the raw data

sets into the desired grid graph format were also different. Consequently the applications are considered in two separate sections; Sub-section 3.2 considers the sheet metal forming application domain while Sub-section 3.3 the satellite image domain.

So as to compare the operation of the proposed VULS techniques with the usage of more standard (traditional) classification approaches (Naive Bayes and J48) the data sets used also had to be pre-processed in a different manner. This pre-processing is described in Sub-section 3.4. It should be noted here that this processing was only applicable because of the nature of the grid represented raw data under consideration. Given other forms of input graph data, this process is unlikely to be applicable. The chapter is concluded with a summary and a "look ahead" in Section 3.5.

## 3.2 Application Domain One: Asymmetric Incremental Sheet Forming (AISF) and Springback Prediction

As noted in the introduction to this thesis the exemplar domain at which the work described in this thesis is directed at sheet metal forming. As described in the introduction to this thesis sheet metal forming is concerned with manufacturing of shapes from sheet metal. There are various ways whereby this can be achieved, but one such process, again as noted in the introduction to this thesis, is Asymmetric Incremental Sheet Forming (AISF). The advantages of AISF are that it is comparatively inexpensive and does not require heating of the metal (heating introduces potential fracture points and adds an additional financial overhead). The disadvantage of AISF metal forming, and similar processes, is that deformation (*springback*) is introduced as a result of the application of the process, consequently the intended shape will not be the same as the actual shape produced. Thus, the AISF process commences with a desired *input* shape, sometimes referred to as the CAD shape, defined in terms of a set of 3-D coordinates; and produces an *output* shape which, as a result of the process, is a "variation" of the desired input shape because of the springback that has been introduced. The resulting output shape can be recorded using an optical measuring system[1] to generate a second set of 3-D coordinates. Thus before and after *coordinate clouds* (input and output) can be identified.

The springback prediction problem is well described in [64, 68, 69, 129, 184] where it is presented as follows. Given a desired shape $T$, a process $P$ and a resulting shape $T'$ it is seen as desirable to be able to predict the correlation $A$ between $T$ and $T'$ so that given a new shape $S$ we can predict the outcome $S'$ and consequently attempt to redefine $S$ so as to minimise the springback. A simple answer to the problem can be expressed as $A = \frac{T+T'}{2}$, where $A$ is the redefined shape. However, the springback introduced by process $P$ is not evenly spread across the entire output shape; it is conjectured by domain experts that the nature of the springback may be dependent on a number

[1] For the work described in this thesis a GOM (Gesellschaft für Optische Messtechnik) optical measuring tool, produced by GOM mbH, was used.

of factors such as tool head shape, tool head speed, tool head pitch, lubricant, blank holder, type of alloy, sheet thickness, sheet size, shape geometry and the forming process used. Whatever the case it is generally acknowledged that a key influencing factor is the geometry of the desired shape. Therefore it can be assumed that the nature of the springback (correlation) between $T$ and $T'$, as a result of the application of the process $P$, is localised according to the geometry of $T$ (and by extension $T'$). In this thesis it is suggested that the VULS concept can be used for predicting the springback associated with local geometries (regional vertex label classification).

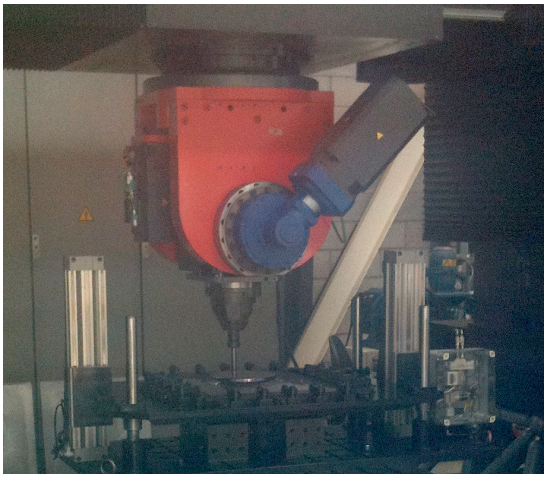


FIGURE 3.1: Example AISF machine 1 [129], the work piece is clamped in position while the tool head "pushes out" the desired shape; on release, springback occurs as a result of which the final shape is not the desired shape.

FIGURE 3.2: Example AISF machine 2, a metal sheet is clamped into a holder and the desired shape is produced using the continuous movement of a simple round-headed forming tool.

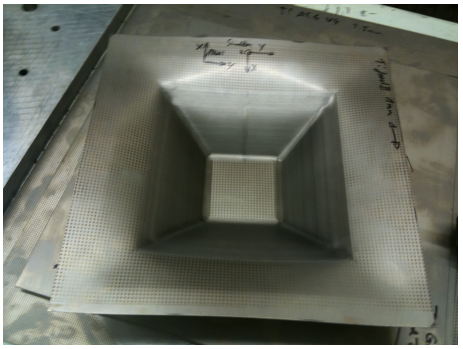


FIGURE 3.3: Square based pyramid (upside down) at the point when it is unclamped after application of the AISF process.

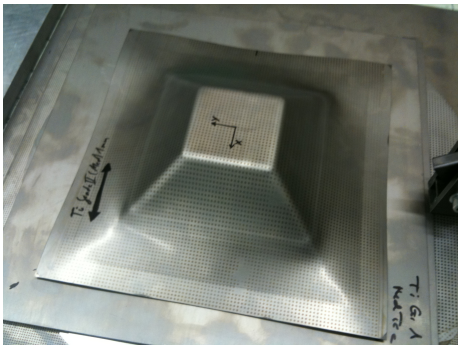


FIGURE 3.4: Square based pyramid (right way up); the markings are used with respect to the GOM optical measuring tool.

The rest of this section is structured as follows. In Sub-section 3.2.1 a brief overview of the AISF process and some related work is presented. In Sub-section 3.2.2 the grid representation commonly adopted with respect to work on AISF is presented (note that this description is based on the work of [64, 68, 69, 129, 184]). A mechanism for measuring the deformation between $T$ and $T'$ is then presented in Sub-section 3.2.3 (again the presented description is based on that of [64, 68, 69, 129, 184]). The AISF data sets used and the adopted translation process are then detailed in Sub-sections 3.2.4 and 3.2.5 respectively. This section is concluded with Sub-section 3.2.6 where some statistics concerning the generated AISF grid graphs are presented.

### 3.2.1 AISF Process

When manufacturing parts using AISF a metal sheet is clamped into a holder and the desired shape is produced using the continuous movement of a simple round-headed forming tool. Two different AISF machines are shown in Figures 3.1 and 3.2. The forming tool is provided with a "tool path" generated by a CAD model and the part is "pressed" out according to the coordinates of the tool path. However, as already noted, due to the nature of both the metal used and the manufacturing process, *springback* occurs; which means that the geometry of the shaped part is different to the geometry of the desired part (deformation has thus been introduced). Figure 3.3 shows a square based pyramid shape at the point when it has been unclamped from the AISF machine; Figure 3.4 shows the same shape "the right way up". The deformation that has been introduced can be seen by inspection of the two figures; the edges of the shape in Figure 3.4 are bent-up towards the corners. In [6] the authors consider a number of products that could potentially be formed using AISF and demonstrated that the accuracy of the formed part needs to be improved before this process could be used in a large scale production. In [101] the authors considered two drawbacks of the AISF process relating to the metal thickness and the geometric accuracy of the resulting shape.

There has been a substantial amount of work on dynamic tool path correction in the context of laser guided tools (see for example [52] and [62]). However, AISF requires that the tool path is specified in advance rather than as the process develops. In [12] a multi-stage forming technique is presented, as opposed to the more standard single pass by the machine tool, several passes are made so that the process can attempt to take into account the deformation that is introduced by springback. As a case study a square based pyramid shape was considered. From [12] it is interesting to note that if the initial geometry comprises corner radii larger than the desired radii, and if a number of forming passes are applied, less distortion results then would be encountered otherwise; in other words, if we have an estimate of springback, we can take this into account when specifying our CAD shape.

From the literature a number of methods have been proposed for calculating springback. For many years the Finite Element Method (FEM) has been used as an industry

standard for calculating the springback associated with sheet metal in forming processes [165]. However, the results of FEM calculations are not very accurate because of the involvement of complex non-linear factors [227]. Not unsurprisingly data mining techniques have also been applied in the context of sheet metal forming springback prediction. There are many examples of the use of neural networks to support sheet metal forming [61, 68, 114, 131, 132, 155, 172, 181]. Considering one example only, in [172] a neural network was trained to predict springback. Several inputs were used for the neural network to train on; such as: thickness, radius and springback. It was observed that the predictions made by the neural networks were very close to the simulation results. Rule based learning techniques have also been popular. For example in [233] rule based mining was used to extract knowledge from data generated by Finite Element Analysis (FEA). Another similar approach was proposed in [238] for the U-draw sheet metal bending process where a rule based system was used to extract knowledge from FEA simulation data. The nature of the material, and various process parameters, were considered with respect to their effect on springback. However, there has been very little reported work (none) on the use of graph mining techniques to predict AISF springback.

In the context of the above previous work, it should be recalled here that the objective of the thesis is to identify the most appropriate mechanisms where VULS mining can be conducted for the purpose of vertex classification. The objective of the thesis is not necessarily to solve the AISF springback prediction problem. This application domain was simply used as a "driver" for the work. The above discussion of previous work on springback has been included here so as to provide the reader with a comprehensive understanding of the AISF springback issue. Hence in the evaluation presented later in this thesis comparisons are not made with the above previously proposed prediction mechanisms; instead the evaluation is directed at vertex classification using the VULS concept.

### 3.2.2 Grid Representation

This section considers the grid representation extracted from the AISF data. The description is based on that previously presented in [64, 68, 69, 129, 184]. From the foregoing, processes such as AISF can be defined in terms of: (i) an input "coordinate cloud" $C_{in}$ (representing $T$) and (ii) an output coordinate cloud $C_{out}$ (representing $T'$). Each coordinate cloud comprises a set of $N$, $\langle x, y, z \rangle$ coordinate triples, such that $x, y, z \in \mathbb{R}$. The number of coordinates per $cm^2$ (within the $x, y$ plane) in each coordinate cloud varies according to how the data is generated/collected. As already noted above, the $C_{in}$ coordinate cloud is typically obtained from a tool path specification generated using a CAD model, while $C_{out}$ is collected using an optical measuring system; $|C_{out}|$ is typically less than $|C_{in}|$. Of course, both coordinate clouds must be registered to the same reference origin and orientation before any comparison can be conducted. The most obvious representation to be adopted is a grid representation, and hence this is the start point for the work presented in this thesis. Figure 3.5 (taken from [129]) shows a grid

where each grid square $c_i$ centre is defined by a $\langle x_i, y_j, z \rangle$ coordinate value pair. The number of grid squares required to represent $C_{in}$ is defined by the grid spacing value $d$ (typically defined in $mm$). Each grid square $c_i$, also has a $z$ value associated with it calculated by averaging the $z$ values of the cloud points contained in the subset of $C_{in}$ that intersect with each $c_i$; this is illustrated in Figure 3.6 [129]. The $C_{out}$ coordinate cloud can then be translated using the same grid format so two grids, $G_{in}$ and $G_{out}$, are produced describing the before and after surfaces ($T$ and $T'$).

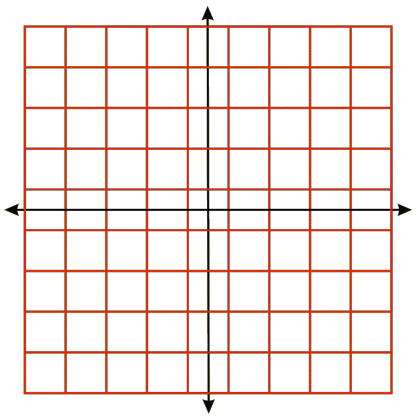


FIGURE 3.5: Example grid referenced to a central origin [129].

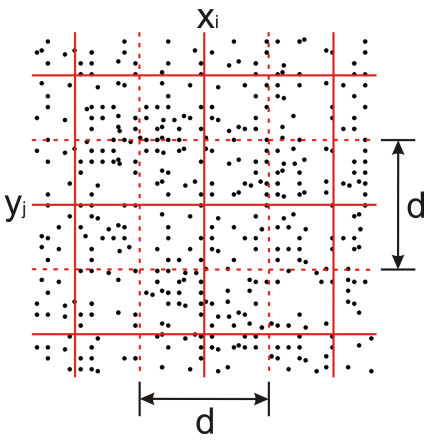



FIGURE 3.6: Coordinate cloud points associated with a grid representation centred on $\langle x_i, y_j \rangle$ (grid spacing = $d$) [129].

### 3.2.3 Springback Measurement

A simple mechanism for establishing the springback value ($e$) at a particular grid point is to measure the difference between the $z$ values between corresponding grid squares in $G_{in}$ and $G_{out}$ as shown in Figure 3.7 [129]. However, a more accurate measure is to determine the distance separating the two surfaces along the surface normal from each grid point in $G_{in}$ to the point where it intersects $G_{out}$. The distance between any two three dimensional points can be calculated using the point to point Euclidean distance formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{3.1}$$

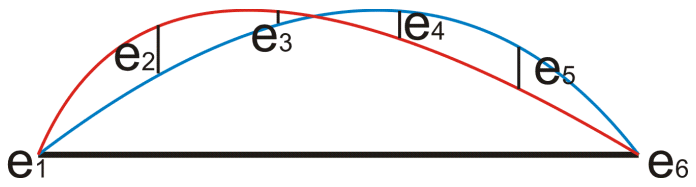



FIGURE 3.7: Cross section at a grid line showing simple vertical springback error calculation between a before (blue line) and an after (red line) shape [129].

However, application of equation 3.1 first requires knowledge of the $x, y, z$ coordinates of the point where the normal intersects $G_{out}$. One method of determining the length of the normal between two surfaces is to use the line-plane intersection method [63] to determine the length of the normal between two surfaces. Using this approach we find the normal to a plane by calculating the cross product of two orthogonal vectors contained within the plane. Once we have the normal we can calculate the equation for the line that includes the start and end points of the normal and then determine the point at which this line cuts $G_{out}$. We can then calculate the length of the normal separating the two planes. The process is well described in [64, 68, 69, 129, 184] and is as follows (with reference to Figure 3.8 taken from [129]):

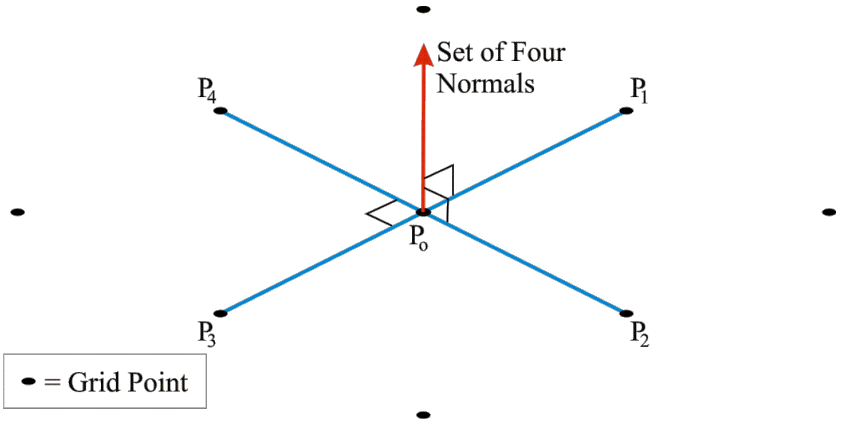



FIGURE 3.8: Error calculation using the line-plane intersection method [129].

1. For each grid point in $G_{in}$ first identify the four neighbouring grid points in the X and Y planes as shown in Figure 3.8 (except at edges and corners where three and two neigbouring grid points will be identified respectively).

2. Define a set of four vectors $V = \{v_1, \ldots, v_4\} = \{\langle p_\emptyset, p_1\rangle, \langle p_\emptyset, p_2\rangle, \langle p_\emptyset, p_3\rangle, \langle p_\emptyset, p_4\rangle\}$, each described in terms of its x-y-z distance from $p_\emptyset$ (the origin for the vector system).

3. Using the four vectors in $V$, four surface normals are calculated, $N = \{n_1 \ldots n_4\}$, by determine the cross product between each pair of vectors: $v_1 \times v_2$, $v_2 \times v_3$, $v_3 \times v_4$, $v_4 \times v_1$. (Note that to validate a surface normal $n_i$, the dot product of one of its associated vectors $v_j$ and $n_i$ must be equal to zero, $n_i \cdot v_j = 0$.)

4. For each normal $n_1 \ldots n_4$ calculate the local plane equation in $G_{in}$ that includes $P_\emptyset$ (thus using, in turn, points $\{P_1, P_\emptyset, P_2\}$, $\{P_2, P_\emptyset, P_3\}$, $\{P_3, P_\emptyset, P_4\}$ and $\{P_4, P_\emptyset, P_1\}$). The plane equation is given by Equation 3.2.

$$ax + by + cz + d = 0 \tag{3.2}$$

5. For each plane equation identified in (4) determine the parametric equations (a set of equations/functions which describe the x, y and z coordinates of the graph of some line in a plane) [63] of the surface normal as a straight line according to the identities given in equation 3.3:

$$x = a + i(t), \qquad y = b + j(t), \qquad z = c + k(t) \tag{3.3}$$

where $t$ is a constant; $a$, $b$ and $c$ are the x-y-z coordinates for the point $p_\emptyset$; and $i$, $j$ and $k$ are the normal components. The constant $t$ is calculated by substituting the parametric equations in plane equation 3.2 for x, y and z.

6. Once the parametric equations for each surface normal are found, they are then used to compute the points of intersection of each normal with $G_{out}$.

7. We then use the coordinates for each of the four points of intersection and $p_\emptyset$ to calculate the Euclidean distance (the springback error) between $p_\emptyset$ and each intersection point to give four springback error values $E = \{e_1 \dots e_4\}$.

8. We then assign each springback error a direction (-ve or +ve) based on the direction of the springback. If springback is "downwards", a -ve direction is assigned to the error. Similarly if the springback is "upwards" a +ve direction is assigned to the error. Note that for each point the direction for each of the four errors is the same.

9. We now have four springback error values for each grid point (except at the corners and edges where we will have two or three respectively), we then find the "overall" error $e$ simply by selecting the minimum error that is nearest to zero. The reason for selecting the minimal error is that it gives us the nearest point to the before surface.

### 3.2.4 AISF Datasets

For the evaluations presented later in this thesis two shapes (3D surfaces) were considered. Both had been specified and manufactured by the IBF (Institut für Bildsame Formgebung) institute of metal forming at Aachen University[2] and both were flat topped pyramid shapes; a shape frequently used in the context of AISF research. The two shapes were referred to as the Gonzalo[3] and Modified pyramids and are shown in Figures 3.9 and 3.10 [184]. Inspection of the figures shows that the two pyramids are not entirely identical (the Gonzalo pyramid has a bulge on one of its side and an indent on the opposite side, while the modified pyramid has indents on two adjacent sides). The overall size of the pyramids was approximately 200 $\times$200 $\times$ 50 $mm$. The shapes were defined

[2] At the time when the research described in this thesis was being conducted the Department of Computer Science at the University of Liverpool was engaged on a European Framework 7 project which included IBF amongst the partners.

[3] The name "Gonzalo" is derived from the name of the person at IBF who designed and manufactured the shape.

in terms of two CAD specifications and each was manufactured four times, twice out of steel and twice out of titanium. By comparing the CAD specification with the output shape, using the process described above, springback measures were obtained. In this manner a total of eight "benchmark", "real world", data sets were produced as follows:

1. Gonzalo Steel Version 1 (GS1).
2. Gonzalo Steel Version 2 (GS2).
3. Gonzalo Titanium Version 1 (GT1).
4. Gonzalo Titanium Version 2 (GT2).
5. Modified Steel Version 1 (MS1).
6. Modified Steel Version 2 (MS2).
7. Modified Titanium Version 1 (MT1).
8. Modified Titanium Version 2 (MT2).

Because each shape was manufactured twice using the same material, corresponding grid graphs could be paired and one used for training purposes and the other for testing (and vice versa). Later in this thesis the abbreviations: GS, GT, MS and MT are used to indicate parings.

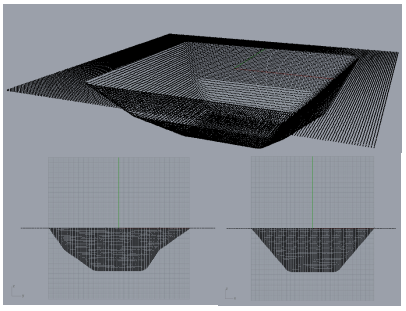


FIGURE 3.9: Gonzalo Pyramid [184].

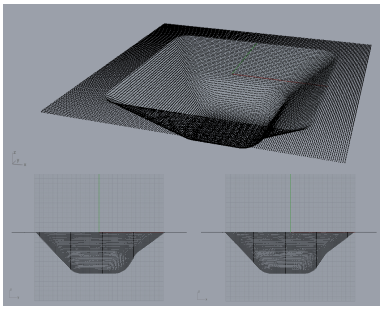


FIGURE 3.10: Modified Pyramid [184].

Thus, in summary, the AISF data sets used for evaluation purposes, in their raw state, consisted of before and after "coordinate clouds"; the first generated by a CAD system, the second using an optical measuring system. These were then transformed into a grid representation with the center points of each grid cell referenced using a X-Y-Z coordinate system. Each grid cell also had a springback value associated with it (calculated as described above). A fragment of some grid data, presented in tabular format, is presented in Table 3.1. The process whereby a grid of the above form was converted into a grid graph is presented in the following Sub-section.

TABLE 3.1: Example of raw input data.

| x | y | z | springback(error) |
|---|---|---|---|
| 0.000 | 0.000 | 0.000 | 0.118 |
| 1.000 | 0.000 | 0.000 | 0.469 |
| 2.000 | 0.000 | 0.000 | 0.469 |
| 3.000 | 0.000 | 0.000 | 0.472 |
| 0.000 | 1.000 | 0.000 | 0.471 |
| 1.000 | 1.000 | -1.402 | 0.088 |
| 2.000 | 1.000 | -4.502 | 1.308 |
| 3.000 | 1.000 | -4.676 | 1.907 |
| ... | ... | ... | ... |

### 3.2.5 AISF Graph Translation

The next step in the data preparation process was to translate the grid data, captured as described above, into a grid graph format such that each grid centre point represented a vertex. Edges were labelled with $z$ difference ($\delta z$) values. Note that springback values were discretinized so that we have a set of springback error labels, $L_V$, to be associated with vertices ($L_V = \{l_{v_1}, L_{v_2}, \dots\}$). Each vertex was then connected to its immediate neighbours by a sequence of either four or eight edges (except at the edge and corner locations), thus Degree = 4 or Degree = 8. If Degree = 4 we refer to this as a "grid graph", if Degree = 8 we refer to this as a "cross-grid graph".

A simple example grid and corresponding grid and cross-grid graphs are given in Figure 3.11. The input grid (lefthand side of Figure 3.11) comprises 9 grid squares. Each grid centre is defined by a $\langle x, y, z\rangle$ coordinate tuple. The number in each grid cell is the $z$ value. This grid can be represented as a grid graph as shown in the middle of figure 3.11 or as a cross-grid graph as shown on righthand side of Figure 3.11. The edges are labelled with "slope" absolute values, the absolute difference in the $z$ coordinate values associated with the two end vertices. In the case of directed graphs (as in the case of the graphs shown in Figure 3.11) the direction of the edges is determined by slope direction, from the lower $z$ value to the higher $z$ value.

### 3.2.6 AISF Grid Graph Statistics

From the foregoing, the translation of grids into grid graphs utilises a number of parameters:

1. The size of the set of vertex labels $|L_V|$.
2. The size of the set of edge labels $|L_E|$.
3. The grid size $d$.
4. Whether the edges in the graph are directed or undirected.
5. Whether the graph is a "grid" graph (degree 4) or a "cross-grid" graph (degree 8).

| | | |
|---|---|---|
| 4 | 2 | 1 |
| 4 | 4 | 3 |
| 1 | 3 | 2 |

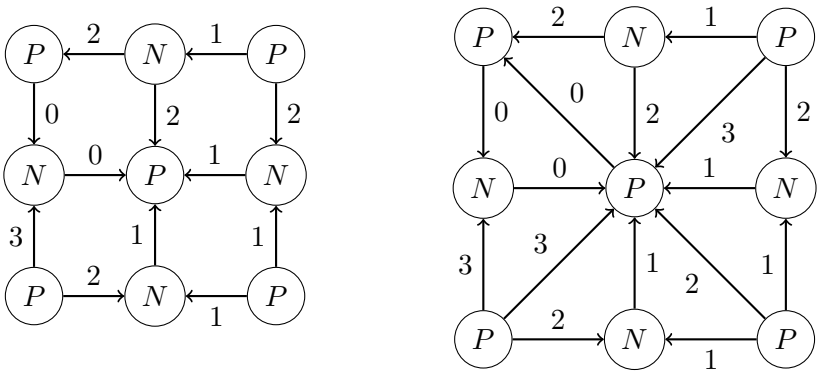



FIGURE 3.11: Grid representation with "Z" values (left), with corresponding grid graph where degree=4 (middle) and cross-grid graph where degree =8 (right), featuring "slope" labels on edges.

For the evaluation presented later in this thesis a range of values were used for: (i) $|L_V|$, and (ii) $|L_E|$; namely $\{2, 3\}$ and $\{2, 3, 4, 5, 6, 7, 8\}$ respectively. Where $|L_V| = 2$ the label set was $\{positive, negative\}$. Where $|L_V| = 3$ the label set was {positive, neutral, negative} with the "neutral" label defined in terms of a positive and negative tolerance value either side of 0; a tolerance value of 0.3 $(mm)$ was chosen so as to achieve some reasonable balance of the distribution of vertex labels when $|L_V| = 3$. Grid sizes of $d = \{28, 23, 18\}$ $(mm)$ were used so as to limit the size of the corresponding grid graphs; also because, as will become apparent later in this thesis, large grid sizes were found to perform better. Thus for each of the eight raw AISF data sets (GS1, GS2, GT1, GT2, MS1, MS2, MT1 and MT2) we have $2 \times 7 \times 3 \times 2 \times 2 = 168$ variations of the associated grid graph. Thus $168 \times 8 = 1344$ graph sets in total.

Tables 3.2 to 3.9 give some statistics concerning the vertex label distributions in each case. With respect to the tables it should be noted that the values given in the "Number of vertices" column is a straight forward function of $d$.

## 3.3 Application Domain Two: Satellite Image Interpretation

The satellite image interpretation application domain is the second application domain considered in this thesis. The primary focus of this thesis was the AISF domain presented above. The main purpose was to use satellite image interpretation to investigate the more general applicability of the VULS concept. For the application a data set comprising ten (cloud free) satellite images, covering a rural area featuring fields and some isolated buildings, within the Harro district in the Oramia Region of Ethiopia (approximately 300 km north-west of Addis Abba) was used. This image set was used because it was readily available within the department of computer science at the university with respect to ongoing research into census estimation [57–59], unrelated to the work, presented in this thesis. The image set was bounded by the parallels of latitude 9.608874N (top) and 9.573583N (bottom), and the meridians of longitude 37.14022E

Table 3.2: Vertex Label distribution for GS1 graph.

| d (mm) | Number of vertices | $\|L_V\|$ | Vextex label | Vertex label distribution(%) |
|---|---|---|---|---|
| 18 | 100 | 2 | positive | 38.00 |
| | | | negative | 62.00 |
| | | 3 | positive | 23.00 |
| | | | neutral | 40.00 |
| | | | negative | 37.00 |
| 23 | 64 | 2 | positive | 43.75 |
| | | | negative | 56.25 |
| | | 3 | positive | 25.00 |
| | | | neutral | 34.38 |
| | | | negative | 40.63 |
| 28 | 36 | 2 | positive | 41.67 |
| | | | negative | 58.33 |
| | | 3 | positive | 27.78 |
| | | | neutral | 36.11 |
| | | | negative | 36.11 |

Table 3.3: Vertex Label distribution for GS2 graph.

| d (mm) | Number of vertices | $\|L_V\|$ | Vextex label | Vertex label distribution(%) |
|---|---|---|---|---|
| 18 | 100 | 2 | positive | 49.00 |
| | | | negative | 51.00 |
| | | 3 | positive | 19.00 |
| | | | neutral | 34.00 |
| | | | negative | 47.00 |
| 23 | 64 | 2 | positive | 45.31 |
| | | | negative | 54.69 |
| | | 3 | positive | 14.06 |
| | | | neutral | 39.06 |
| | | | negative | 46.88 |
| 28 | 36 | 2 | positive | 52.78 |
| | | | negative | 47.22 |
| | | 3 | positive | 22.22 |
| | | | neutral | 33.33 |
| | | | negative | 44.44 |

TABLE 3.4: Vertex Label distribution for GT1 graph.

| d (mm) | Number of vertices | $\|L_V\|$ | Vextex label | Vertex label distribution(%) |
|---|---|---|---|---|
| 18 | 100 | 2 | positive | 31.00 |
| | | | negative | 69.00 |
| | | 3 | positive | 17.00 |
| | | | neutral | 20.00 |
| | | | negative | 63.00 |
| 23 | 64 | 2 | positive | 25.00 |
| | | | negative | 75.00 |
| | | 3 | positive | 14.06 |
| | | | neutral | 21.88 |
| | | | negative | 64.06 |
| 28 | 36 | 2 | positive | 30.56 |
| | | | negative | 69.44 |
| | | 3 | positive | 11.11 |
| | | | neutral | 25.00 |
| | | | negative | 63.89 |

TABLE 3.5: Vertex Label distribution for GT2 graph.

| d (mm) | Number of vertices | $\|L_V\|$ | Vextex label | Vertex label distribution(%) |
|---|---|---|---|---|
| 18 | 100 | 2 | positive | 34.00 |
| | | | negative | 66.00 |
| | | 3 | positive | 20.00 |
| | | | neutral | 20.00 |
| | | | negative | 60.00 |
| 23 | 64 | 2 | positive | 29.69 |
| | | | negative | 70.31 |
| | | 3 | positive | 18.75 |
| | | | neutral | 23.44 |
| | | | negative | 57.81 |
| 28 | 36 | 2 | positive | 33.33 |
| | | | negative | 66.67 |
| | | 3 | positive | 22.22 |
| | | | neutral | 16.67 |
| | | | negative | 61.11 |

Table 3.6: Vertex Label distribution for MS1 graph.

| d (mm) | Number of vertices | $\|L_V\|$ | Vextex label | Vertex label distribution(%) |
|---|---|---|---|---|
| 18 | 100 | 2 | positive | 64.00 |
| | | | negative | 36.00 |
| | | 3 | positive | 45.00 |
| | | | neutral | 30.00 |
| | | | negative | 25.00 |
| 23 | 64 | 2 | positive | 57.81 |
| | | | negative | 42.19 |
| | | 3 | positive | 40.63 |
| | | | neutral | 32.81 |
| | | | negative | 26.56 |
| 28 | 36 | 2 | positive | 55.56 |
| | | | negative | 44.44 |
| | | 3 | positive | 47.22 |
| | | | neutral | 13.89 |
| | | | negative | 38.89 |

Table 3.7: Vertex Label distribution for MS2 graph.

| d (mm) | Number of vertices | $\|L_V\|$ | Vextex label | Vertex label distribution(%) |
|---|---|---|---|---|
| 18 | 100 | 2 | positive | 46.00 |
| | | | negative | 54.00 |
| | | 3 | positive | 28.00 |
| | | | neutral | 36.00 |
| | | | negative | 36.00 |
| 23 | 64 | 2 | positive | 43.75 |
| | | | negative | 56.25 |
| | | 3 | positive | 25.00 |
| | | | neutral | 37.50 |
| | | | negative | 37.50 |
| 28 | 36 | 2 | positive | 52.78 |
| | | | negative | 47.22 |
| | | 3 | positive | 38.89 |
| | | | neutral | 19.44 |
| | | | negative | 41.67 |

TABLE 3.8: Vertex Label distribution for MT1 graph.

| d (mm) | Number of vertices | $\|L_V\|$ | Vextex label | Vertex label distribution(%) |
|---|---|---|---|---|
| 18 | 100 | 2 | positive | 21.00 |
| | | | negative | 79.00 |
| | | 3 | positive | 14.00 |
| | | | neutral | 17.00 |
| | | | negative | 69.00 |
| 23 | 64 | 2 | positive | 17.19 |
| | | | negative | 82.81 |
| | | 3 | positive | 7.81 |
| | | | neutral | 20.31 |
| | | | negative | 71.88 |
| 28 | 36 | 2 | positive | 19.44 |
| | | | negative | 80.56 |
| | | 3 | positive | 11.11 |
| | | | neutral | 22.22 |
| | | | negative | 66.67 |

TABLE 3.9: Vertex Label distribution for MT2 graph.

| d (mm) | Number of vertices | $\|L_V\|$ | Vextex label | Vertex label distribution(%) |
|---|---|---|---|---|
| 18 | 100 | 2 | positive | 20.00 |
| | | | negative | 80.00 |
| | | 3 | positive | 10.00 |
| | | | neutral | 25.00 |
| | | | negative | 65.00 |
| 23 | 64 | 2 | positive | 10.94 |
| | | | negative | 89.06 |
| | | 3 | positive | 3.13 |
| | | | neutral | 26.56 |
| | | | negative | 70.31 |
| 28 | 36 | 2 | positive | 13.89 |
| | | | negative | 86.11 |
| | | 3 | positive | 0 |
| | | | neutral | 27.78 |
| | | | negative | 72.22 |

(left) and 37.17209E (right). The area covered measured 13.6559($km^2$). For identification purposes the images were numbered from 1 to 10. The satellite imagery was extracted using the Google Static Map Service[4] in 2013. One of the images is shown in Figure 3.12. Note that the images were in RGB format. The images measured $256 \times 256$ pixels.


FIGURE 3.12: Example satellite image.

The rest of this section is structured as follows. The process for translating the satellite images to the desired grid-graph format is described in Sub-section 3.3.1. Note that this is much simpler than the process required to translate AISF data to a grid graph format described above. Sub-section 3.3.2 then gives some statistics concerning the resulting grid graphs.

### 3.3.1 Satellite Image Graph Translation

A number of techniques, such as the cluster and Waxman models [7, 13, 14, 18, 51, 90], have previously been proposed for translating images to graph formats. However, none of these existing techniques was entirely suited to the specific grid-graph format required for the application of VULS mining and VULS classification as envisioned in this thesis. A bespoke mechanism was therefore developed by the author. The nature of this mechanism is described in this section.

The process for translating a satellite image grid into a grid graph is illustrated in Figure 3.13. For simplicity only a small $3 \times 3$ section (highlighted in red) of the input grid is used for the example given in the figure. The process is commenced by converting each RGB represent satellite image into a grayscale image. A grid, of grid size $d$ pixels,

[4] https://developers.google.com/maps/documentation/staticmaps/

was then superimposed over each image. A number of example grids are given in Figure 3.14 using $d = 32$, $d = 16$ and $d = 8$ respectively. For each grid square, the mean grayscale value of all pixels in the grid square was calculated. This then became the $z$ value to be associated with each grid center point (the reference point), referred to as the centre pixel. For training and testing purposes each grid cell was assigned a label describing the nature of the ground cover. The mean grayscale intensity value and ground type with respect to a $3 \times 3$ section is shown in two green tables respectively included in Figure 3.13.

The grid was then translated into a grid graph in the same manner as described above. The edge connecting each pair of vertices was again labeled using $z$ difference, although in this case this was the mean grayscale intensity difference between two vertices. As before each vertex was then connected using a sequence of either four or eight edges (except at the edge and corner locations), thus Degree= 4 or Degree= 8. As before the term "grid graph" is used where Degree= 4 or "cross-grid graph" where Degree= 8 (as shown in the bottom-left and bottom-right of Figure 3.13 respectively). Again graphs can be directed or undirected. In the case of directed graphs (as in the case of the graphs shown in Figure 3.13) the direction of the edges is determined by slope direction, from low mean grayscale intensity value to high mean grayscale intensity value.

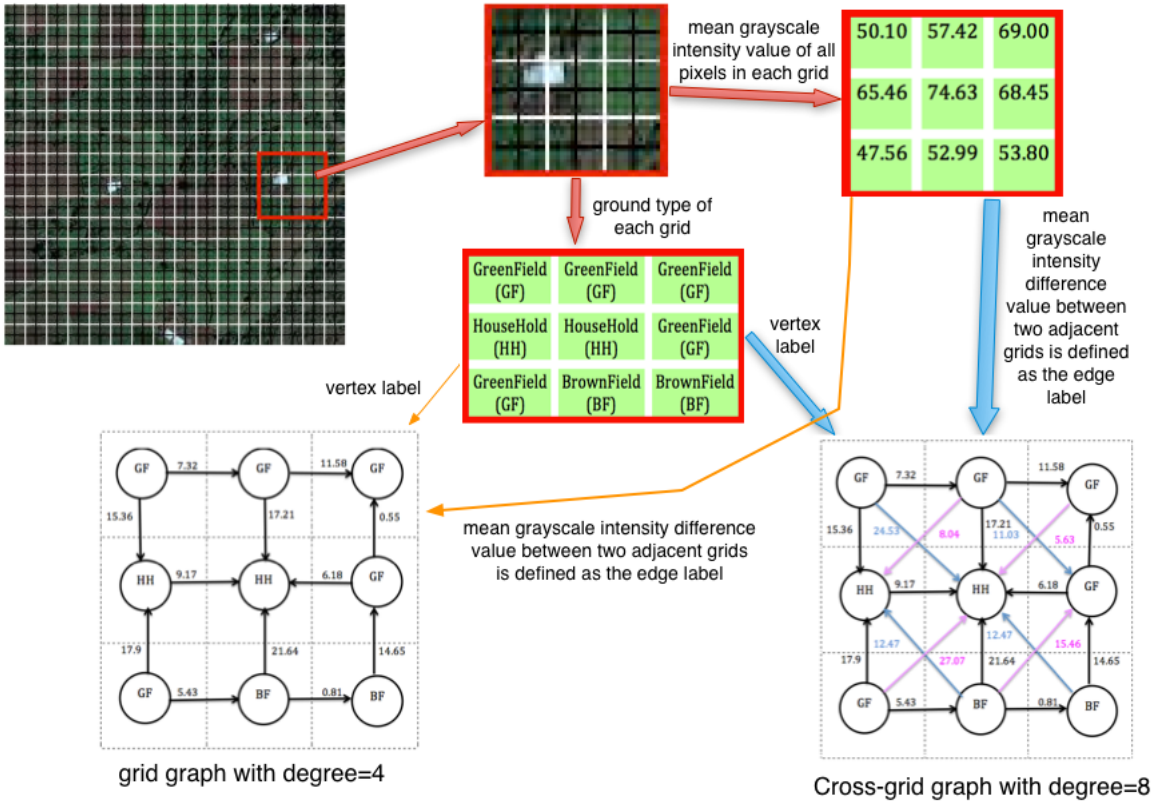



FIGURE 3.13: Process of Translating a satellite image into a "grid graph" and a "cross-grid graph" (the edge colour encoding is for ease of understanding only).

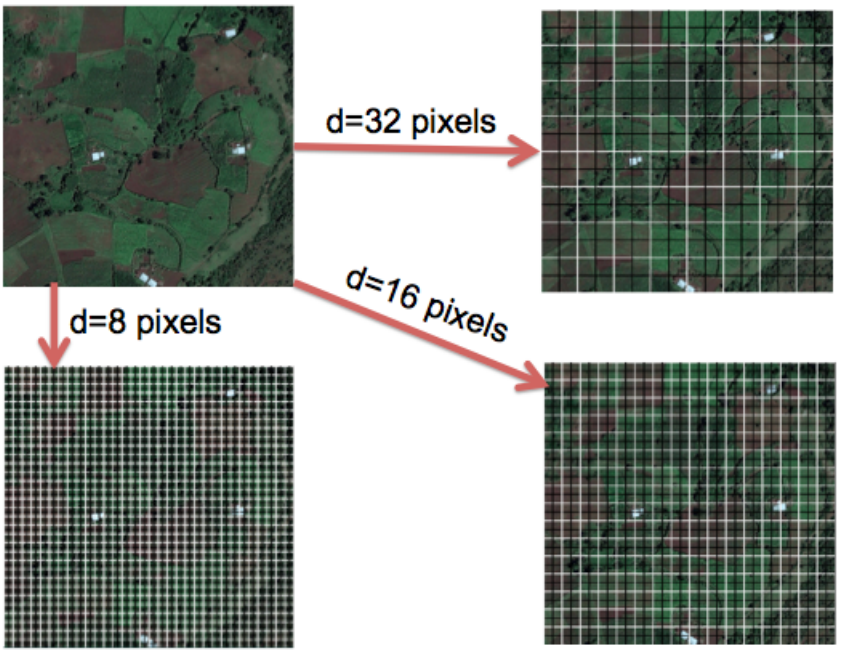



FIGURE 3.14: Satellite image represented in terms of three different grid squares using three different values for $d$.

### 3.3.2 Satellite Grid Graph Statistics

As in the case of the AISF grid graphs the translation of satellite image grids into grid graphs utilises the same six parameters. For $|L_V|$ a range of two values was again used $\{2, 3\}$. Where $|L_V| = 2$, $L_V = \{Field, HouseHold\}$; where $|L_V| = 3$, $L_V = \{BrownField, GreenField, HouseHold\}$. Labels were added to grid squares by hand. Similarly, for $|L_E|$ the range of values $\{2, 3, 4, 5, 6, 7, 8\}$ was again used. For the value of $d$, $\{8, 16, 32\}$ pixels was used. Thus for each satellite image we have $2 \times 7 \times 3 \times 2 \times 2 = 168$ variations of the associated grid graph. Thus $168 \times 10 = 1680$ graph sets in total.

Tables 3.10 to 3.19 present the vertex label distribution for the satellite image grid graphs (numbered from 1 to 10). Note that the distribution is extremely imbalanced. In the same manner as in the case of sheet metal forming, the number of grid squares required to represent a given image will depend on the selected grid size $d$. The number of vertices is consequently also dependent on grid size. Using higher values of $d$ produces fewer grid squares consequently fewer vertices. For example, if a grid size of $d = 8$ is used the number of grid cells (vertices) in each row will be $\frac{256}{d} = 32$ ; the total number of grid cells (vertices) will then be $32 \times 32 = 1024$. The number of edges when degree= 4 will be $2 \times 32 \times (32 - 1) = 1984$, and $2 \times n \times (n - 1) + 2 \times (32 - 1) \times (32 - 1) = 3906$ when degree= 8. In the same manner, when $d = 32$ the number of vertices will be 64.

TABLE 3.10: Vertex Label distribution of Satellite Image graph 1.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 99.80 |
| | | | HouseHold | 0.20 |
| | | 3 | BrownField | 87.30 |
| | | | HouseHold | 0.20 |
| | | | GreenField | 12.50 |
| 16 | 256 | 2 | Field | 99.61 |
| | | | HouseHold | 0.39 |
| | | 3 | BrownField | 87.11 |
| | | | HouseHold | 0.39 |
| | | | GreenField | 12.50 |
| 32 | 64 | 2 | Field | 98.44 |
| | | | HouseHold | 1.56 |
| | | 3 | BrownField | 85.94 |
| | | | HouseHold | 1.56 |
| | | | GreenField | 12.50 |

TABLE 3.11: Vertex Label distribution of Satellite Image graph 2.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 98.73 |
| | | | HouseHold | 1.27 |
| | | 3 | BrownField | 68.16 |
| | | | HouseHold | 1.27 |
| | | | GreenField | 30.57 |
| 16 | 256 | 2 | Field | 97.27 |
| | | | HouseHold | 2.73 |
| | | 3 | BrownField | 67.97 |
| | | | HouseHold | 2.73 |
| | | | GreenField | 29.30 |
| 32 | 64 | 2 | Field | 92.19 |
| | | | HouseHold | 7.81 |
| | | 3 | BrownField | 65.63 |
| | | | HouseHold | 7.81 |
| | | | GreenField | 26.56 |

TABLE 3.12: Vertex Label distribution of Satellite Image graph 3.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 99.41 |
| | | | HouseHold | 0.59 |
| | | 3 | BrownField | 34.38 |
| | | | HouseHold | 0.58 |
| | | | GreenField | 65.04 |
| 16 | 256 | 2 | Field | 98.83 |
| | | | HouseHold | 1.17 |
| | | 3 | BrownField | 34.38 |
| | | | HouseHold | 1.17 |
| | | | GreenField | 64.45 |
| 32 | 64 | 2 | Field | 95.31 |
| | | | HouseHold | 4.69 |
| | | 3 | BrownField | 34.38 |
| | | | HouseHold | 4.68 |
| | | | GreenField | 60.94 |

TABLE 3.13: Vertex Label distribution of Satellite Image graph 4.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 98.83 |
| | | | HouseHold | 1.17 |
| | | 3 | BrownField | 82.32 |
| | | | HouseHold | 1.17 |
| | | | GreenField | 16.50 |
| 16 | 256 | 2 | Field | 97.66 |
| | | | HouseHold | 2.34 |
| | | 3 | BrownField | 82.03 |
| | | | HouseHold | 2.34 |
| | | | GreenField | 15.63 |
| 32 | 64 | 2 | Field | 92.19 |
| | | | HouseHold | 7.81 |
| | | 3 | BrownField | 79.69 |
| | | | HouseHold | 7.81 |
| | | | GreenField | 12.50 |

TABLE 3.14: Vertex Label distribution of Satellite Image graph 5.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 99.80 |
| | | | HouseHold | 0.20 |
| | | 3 | BrownField | 91.80 |
| | | | HouseHold | 0.20 |
| | | | GreenField | 8.00 |
| 16 | 256 | 2 | Field | 99.61 |
| | | | HouseHold | 0.39 |
| | | 3 | BrownField | 91.80 |
| | | | HouseHold | 0.39 |
| | | | GreenField | 7.81 |
| 32 | 64 | 2 | Field | 98.44 |
| | | | HouseHold | 1.56 |
| | | 3 | BrownField | 90.63 |
| | | | HouseHold | 1.56 |
| | | | GreenField | 7.81 |

TABLE 3.15: Vertex Label distribution of Satellite Image graph 6.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 98.83 |
| | | | HouseHold | 1.17 |
| | | 3 | BrownField | 80.66 |
| | | | HouseHold | 1.17 |
| | | | GreenField | 18.16 |
| 16 | 256 | 2 | Field | 96.48 |
| | | | HouseHold | 3.52 |
| | | 3 | BrownField | 79.68 |
| | | | HouseHold | 3.52 |
| | | | GreenField | 16.80 |
| 32 | 64 | 2 | Field | 95.31 |
| | | | HouseHold | 4.69 |
| | | 3 | BrownField | 81.25 |
| | | | HouseHold | 4.69 |
| | | | GreenField | 14.06 |

TABLE 3.16: Vertex Label distribution of Satellite Image graph 7.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 99.61 |
| | | | HouseHold | 0.39 |
| | | 3 | BrownField | 53.13 |
| | | | HouseHold | 0.39 |
| | | | GreenField | 46.48 |
| 16 | 256 | 2 | Field | 99.22 |
| | | | HouseHold | 0.78 |
| | | 3 | BrownField | 53.13 |
| | | | HouseHold | 0.78 |
| | | | GreenField | 46.09 |
| 32 | 64 | 2 | Field | 98.44 |
| | | | HouseHold | 1.56 |
| | | 3 | BrownField | 53.13 |
| | | | HouseHold | 1.56 |
| | | | GreenField | 45.31 |

TABLE 3.17: Vertex Label distribution of Satellite Image graph 8.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 99.61 |
| | | | HouseHold | 0.39 |
| | | 3 | BrownField | 59.57 |
| | | | HouseHold | 0.39 |
| | | | GreenField | 40.04 |
| 16 | 256 | 2 | Field | 99.22 |
| | | | HouseHold | 0.78 |
| | | 3 | BrownField | 59.38 |
| | | | HouseHold | 0.78 |
| | | | GreenField | 39.84 |
| 32 | 64 | 2 | Field | 98.44 |
| | | | HouseHold | 1.56 |
| | | 3 | BrownField | 62.50 |
| | | | HouseHold | 1.56 |
| | | | GreenField | 35.94 |

TABLE 3.18: Vertex Label distribution of Satellite Image graph 9.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 99.61 |
| | | | HouseHold | 0.39 |
| | | 3 | BrownField | 71.88 |
| | | | HouseHold | 0.39 |
| | | | GreenField | 27.73 |
| 16 | 256 | 2 | Field | 99.61 |
| | | | HouseHold | 0.39 |
| | | 3 | BrownField | 71.88 |
| | | | HouseHold | 0.39 |
| | | | GreenField | 27.73 |
| 32 | 64 | 2 | Field | 98.44 |
| | | | HouseHold | 1.56 |
| | | 3 | BrownField | 71.88 |
| | | | HouseHold | 1.56 |
| | | | GreenField | 26.56 |

TABLE 3.19: Vertex Label distribution of Satellite Image graph 10.

| d (pixels) | Number of vertices | $\|L_V\|$ | Vertex label | Vertex label distribution (%) |
|---|---|---|---|---|
| 8 | 1024 | 2 | Field | 99.80 |
| | | | HouseHold | 0.20 |
| | | 3 | BrownField | 82.81 |
| | | | HouseHold | 0.20 |
| | | | GreenField | 16.99 |
| 16 | 256 | 2 | Field | 99.61 |
| | | | HouseHold | 0.39 |
| | | 3 | BrownField | 83.59 |
| | | | HouseHold | 0.39 |
| | | | GreenField | 16.02 |
| 32 | 64 | 2 | Field | 96.88 |
| | | | HouseHold | 3.12 |
| | | 3 | BrownField | 81.25 |
| | | | HouseHold | 3.12 |
| | | | GreenField | 15.63 |

## 3.4 Tabular format for Traditional Classification

As noted in the introduction to this chapter, and earlier in this thesis, the operation of the proposed VULS classification mechanism was compared with the application of more traditional classification approaches (namely Naive Bayes and J48). It should be noted here that the reason we can apply J48 and Naive Bayes is because we are using grid graphs. If we were not using grid graphs, but some other form of graph, we could not use J48 or Naive Bayes and hence such comparison could not be made. Traditional classification approaches, such as J48 and Naive Bayes, require the input data to be in a tabular form. Figure 3.15 illustrates the process of translating a grid graph, in this case with degree= 4 and undirected edges, into such a tabular format. From the figure it can be seen that each vertex in the graph corresponds to a record in the table. The vertex label is then the class label for record. The four edges labels are the attribute values. Thus the record describing the centre vertex in Figure 3.15 would be $\langle E1, E2, E3, E4, V1 \rangle$. Note that it is only possible to translate grid graphs into this tabular format because of the regular nature of grid graphs. It would not be possible to apply this with more general irregular graph formats to which the VULS concept could also be applied.

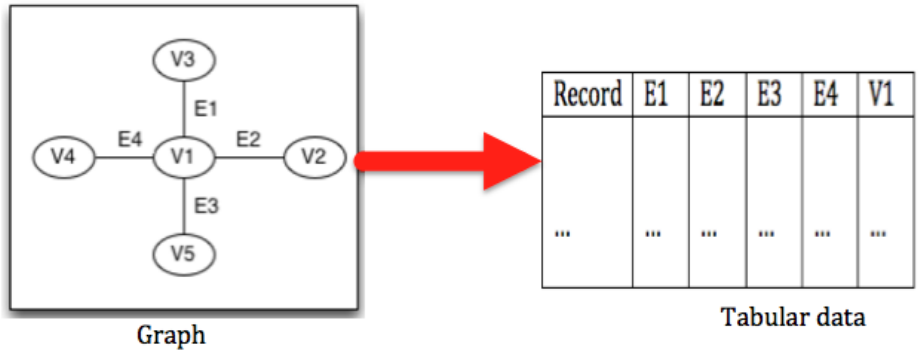



FIGURE 3.15: Process for translating a grid graph into a tabular format.

## 3.5 Summary

In this chapter an overview of the exemplar problem domains (sheet metal forming and satellite image interpretation), and the associated data sets, used to evaluate the proposed VULS mining and vertex classification processes proposed later in this thesis, have been described. In the case of the sheet metal forming data, a total of 1344 grid graphs were generated from eight before and after "point clouds" pairings. In the case of the satellite image interpretation data a total of 1680 grid graphs were generated.

# Chapter 4

# Formalism for VULS

## 4.1 Introduction

This chapter considers the concept of Vertex Unique Labelled Subgraphs (VULS) in detail and presents a formalism for the VULS idea. As already noted, a VULS is a subgraph, within a single large graph, that has vertex labelling associated with it in a unique manner. In other words, VULS means, given a configuration with a set of edges and edge labellings, there is only a fixed set of vertexes matching it, with each vertex has only one possible label. We can identify different forms (categories) of VULS: (i) complete (all possible VULS), (ii) minimal, (iii) frequent and (iv) minimal frequent (a combination of (ii) and (iii)). Each is discussed in this chapter together with examples.

The introduction to this thesis noted that, in the context of VULS mining, there are two significant challenges. The first, in common with other forms of graph mining, is that VULS mining is computationally expensive. Thus anything we can do to reduce the computation overhead associated with VULS mining will be beneficial. The second is that the identified set of VULS $\mathcal{U}$ should describe as wide a range of different configurations as possible. This can be expressed in terms of a metric, *coverage*, which is calculated as follows:

$$coverage = \frac{|V'|}{|V|} \tag{4.1}$$

where $V$ is the set of vertices in a given graph $G$, and $V'$ is the subset of elements of V that appear in at least one VULS in a given set of VULS $\mathcal{U}$. Ideally the coverage will be 1. Thus to achieve good coverage a comprehensive training set is required to generate the set $\mathcal{U}$. As noted above VULS mining is computationally expensive. Thus instead of identifying all VULS, we can attempt to identify some appropriately descriptive (in terms of coverage) subset of the complete set of VULS. Hence the three alternative categories of VULS: (i) minimal VULS, (ii) frequent VULS and (iii) minimal frequent VULS (as listed above).

The rest of this chapter is organized as follows. Section 4.2 presents a formal definition of the VULS problem as conceptualized with respect to this thesis. Section 4.3

demonstrates the relationship (and distinction) between the four categories of VULS listed above using an undirected graph example, while in Section 4.4 the relationship is described in terms of a directed graph example. Finally, in Section 4.5, the chapter is concluded with a brief summary, some discussion and some conclusions.

## 4.2 Formalism

With reference to the definitions given in chapter 2, this section presents a formal definition of the concept of VULS. Recall that labelled graphs are defined as follows:

$$G(V, E, L_v, L_E, f_{map})$$

where:

| | |
|---|---|
| $V$ | is a set of $n$ vertices such that $V = \{v_1, v_2, \ldots, v_n\}$ |
| $E$ | is a set of $m$ edges such that $E = \{e_1, e_2, \ldots, e_m\}$ |
| $L_V$ | is a set of $p$ vertex labels $L_V = \{l_{v_1}, l_{v_2}, \ldots, l_{v_p}\}$ |
| $L_E$ | is a set of $q$ edge labels $L_E = \{l_{e_1}, l_{e_2}, \ldots, l_{e_q}\}$ |
| $f_{map}$ | is some mapping function that maps the vertex and edge labels on to vertices and edges. |

Alternatively we can think of a graph as consisting of a set of $k$ one-edge subgraphs (pairs of vertices linked by an edge). In which case:

$$G = \{P_1, P_2, \ldots, P_k\}$$

where $P_i = \langle v_a, v_b \rangle$ $(v_a, v_b \in V)$. The size of a graph $G$ ($|G|$) can thus be defined in terms of its one edge subgraphs, we refer to 1-edge, 2-edge and $k$-edge subgraphs. For undirected graphs, the edge $\langle v_a, v_b \rangle$ is equivalent to $\langle v_b, v_a \rangle$. We use the notation $P_i.v_a$ and $P_i.v_b$ to indicate the vertices $v_a$ and $v_b$ associated with a particular one-edge subgraph $P_i$. We indicate the labels associated with $P_i.v_a$ and $P_i.v_b$ using the notation $P_i.v_a.label$ and $P_i.v_b.label$ ($P_i.v_a.label \in L_V$ and $P_i.v_b.label \in L_V$). We indicate the edge label associated with $P_i$ using the notation $P_i.label$ ($P_i.label \in L_E$). We also assume that $G$ is connected:

$$\forall P_i \in G \; \exists P_j \in G \mid P_i \neq P_j, P_i.v_a = Pj.v_a \vee$$
$$P_i.v_a = Pj.v_b \vee P_i.v_b = Pj.v_a \vee P_i.v_b = Pj.v_b$$

We can impose a canonical ordering on $G = \{P_1, P_2, \ldots, P_k\}$, in which case the one edge subgraphs in $G$ are best expressed as a list:

$$G = [P_1, P_2, \ldots, P_k]$$

The one edge subgraphs $[P_1, P_2, \ldots, P_k]$ can be ordered using a minimum Depth First Search (DFS) code as used in gSpan. We can use the same notation with respect to any k-edge subgraph $g_k$ ($1 \leqslant k \leqslant |G|$) of $G$ ($g_k \subseteq G$). By default it is assumed that any given graph or subgraph ($G$ or $g_k$) are edge labelled but without any vertex labelling. We can also conceive of vertex unlabelled graphs and subgraphs. With respect to the work described here we are particularly interested in edge labelled subgraphs. In other words, we assume that the configuration $g_k$ with edge labelling is generated from $G$ beforehand.

Given an edge labelled subgraph, $g_k \subseteq G$, comprised of $k$ edges, we can define a function, $getVertexLables$, that determines the potential labels that can be assigned to each vertex in $g_k$ with respect to $G$:

$$getVertexLables(g_k) \rightarrow L$$

where $L = [[L_{v_{a1}}, L_{v_{b1}}], [L_{v_{a2}}, L_{v_{b2}}], \ldots, [L_{v_{ak}}, L_{v_{bk}}]$.

To formally define the concept of a vertex unique labelled subgraph we will commence by defining what we mean by a one-edge vertex unique labelled subgraph and then go on to consider two-edge and $k$-edge vertex unique labelled subgraphs.

Returning to the concept of VULS, given an one-edge subgraph, $g_1 = P_1$ ($g_1 \subseteq G$), such that $P_1.label = l \in L_E$ and $L = [[L_{v_a}, L_{v_b}]]$, so that $L_{v_a} \subseteq L_V$ and $L_{v_b} \subseteq L_V$. If $|L_{v_a}| = 1$ and $|L_{v_b}| = 1$ then $g_1$ is a one-edge VULS with respect to $G$ because there is only one possible vertex labelling (the vertex labelling is unique with the particular edge structure). Given a two-edge edge labelled subgraph $g_2 = [P_1, P_2]$ (where $g_2 \subseteq G$) such that:

$$\begin{aligned} P_{1.label} &= l_1 \in L_E \\ P_{2.label} &= l_2 \in L_E \end{aligned}$$

and:

$$L = [[L_{v_{a1}}, L_{v_{b1}}], [L_{v_{a2}}, L_{v_{b2}}]]$$

and the proviso that $g_2$ is connected.

$$L_{v_{a1}} \cap L_{v_{a2}} \geq 1 \vee L_{v_{a1}} \cap L_{v_{b2}} \geq 1 \vee L_{v_{b1}} \cap L_{v_{a2}} \geq 1 \vee L_{v_{b1}} \cap L_{v_{b2}} \geq 1$$

Then, if $\forall l_i \in L, |l_i| = 1$ the two-edge subgraph $g_2$ will be a two-edge VULS with respect to $G$. The formal definition of the concept of a VULS is then as follows. Given a $k$-edge edge labelled subgraph $g_k = [P_1, P_2, \ldots, P_k]$ ($g_k \subseteq G$) where $L = [[L_{v_{a1}}, L_{v_{b1}}], [L_{v_{a2}}, L_{v_{b2}}], \ldots, [L_{v_{ak}}, L_{v_{bk}}]]$, and the proviso that $g_k$ is connected:

$$\begin{gathered} \forall [L_{v_{ai}}, L_{v_{bi}}] \in L \; \exists \; [L_{v_{aj}}, L_{v_{bj}}] \in L \mid [L_{v_{ai}}, L_{v_{bi}}] \neq [L_{v_{aj}}, L_{v_{bj}}], \\ L_{v_{ai}} \cap L_{v_{aj}} \geq 1 \vee L_{v_{ai}} \cap L_{v_{bj}} \geq 1 \vee L_{v_{bi}} \cap L_{v_{aj}} \geq 1 \vee L_{v_{bi}} \cap L_{v_{bj}} \geq 1 \end{gathered}$$

if $\forall l_i \in L, |l_i| = 1$ then $g_k$ is a $k$-edge VULS with respect to $G$.

Referring back to the introduction to this chapter, other than "standard" VULS as defined above, we can identify three other kinds of VULS: Minimal VULS, Frequent VULS and Minimal frequent VULS. These we defined in general terms in Chapter 2, however maybe more formally defined as follows:

**Definition 1: Minimal VULS.** A minimal VULS is a VULS such that none of its subgraphs are VULS. In other words, given the set of all VULS $T$. A minimal VULS m is a VULS that is in $T$, thus $m \in T$, and whose subsets are not in $T$. Thus given N={ N | all subsets of m }, if $\forall n \in N, n \notin T$=true, then n is a minimal VULS.

**Definition 2: Frequent VULS.** A Frequent VULS (FVULS) is a VULS $\phi$ whose occurrence count, $Occurrence(\phi)$, within the input graph $G$, is greater than some pre-specified threshold $\sigma$. Note that here $\sigma$ is the occurrence count, not the proportion of occurrence of a subgraph over a total number of graph transactions as used in the context of transaction graph mining as described in chapter 2. In the research presented in this thesis $\sigma$ is determined in a dynamic manner (the precise mechanism for calculating $\sigma$ will be presented later in Chapter 5).

**Definition 3: Minimal frequent VULS.** A minimal frequent VULS is a VULS $\phi$ which is both minimal and frequent.

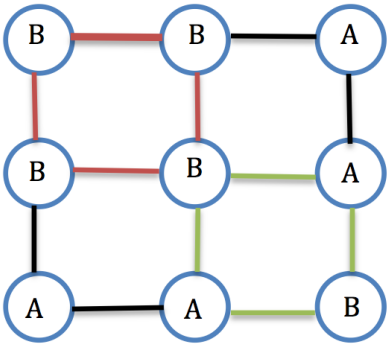



FIGURE 4.1: Example of an undirected graph $G$

## 4.3 Examples of undirected VULS

Consider the undirected graph $G$ given in Figure 4.1. This graph can be encoded as a set of five tuples of the form $\langle u, v, l_u, l_e, l_v \rangle$ where $u, v \in V$, $e = (u, v) \in E$, $l_u, l_v \in L_V$ and $l_e \in L_E$; $u$ and $v$ can also be seen as "vertex position indicators" in a specific subgraph. Thus, with reference to Figure 4.1:

$$L_V = \{A, B\} \qquad L_E = \{black, green, red\}$$

Figures 4.2 and 4.3 show, respectively, the one and two edge VULS candidates that exist in the example undirected graph. The figures also give the occurrence counts for each of the identified VULS candidates. The notation "_" is used to define a vertex without a label (thus a place holder). With reference to Figure 4.2, subgraphs 1 ($\langle 0, 1, \_, red, \_\rangle$) and 3 ($\langle 0, 1, \_, green, \_\rangle$) are VULS because they have unique vertex labelings associated with them. Subgraph 2 is not a VULS because it has the same form ($\langle 0, 1, \_, black, \_\rangle$) but with two potential vertex labelings. In addition if we assume a threshold of $\sigma = 4$ subgraphs 1 and 3 can be said to be frequent VULS (because their occurrence count is at least $\sigma$). Subgraphs 1 and 3 are also minimal VULS because they do not contain any subgraphs that are themselves VULS. By definition subgraphs 1 and 3 are thus also minimal frequent VULS.

With reference to Figure 4.3 subgraphs 4, 5, 6, 7 and 8 are all VULS. However, subgraph 9 is not because it has the same configuration but different (non-unique) vertex labelings. Again, assuming $\sigma = 4$, subgraphs 4, 5, 6, 8 are also frequent VULS. Subgraph 7 is also a minimal VULS because it does not contain any subgraphs that are also VULS. Figure 4.3 does not feature any subgraph that is a minimal frequent VULS.

Note that the VULS ID (the first column of Figures 4.2 and 4.3) are also included in Figure 2.2 in red in chapter 2. This confirms the relationship between the four different categories of VULS. In this case, where the VULS size is less than 2 (recall that VULS size is measured in terms of the number of edges), the coverage with respect to each VULS form is: (i) complete=100%, (ii) minimal=100%, (iii) frequent=100%, and (iv) minimal frequent =7/9=77.78%.

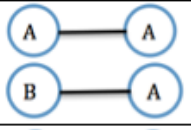
| ID | 1-edge VULS candidates | Candidates matching vertices' labels | 1-edge VULS | Occ. Count | Freq. VULS ? | Min. VULS ? | Min. Freq VULS ? |
|---|---|---|---|---|---|---|---|
| 1 |  |  |  | 4 | yes | yes | yes |
| 2 |  |  | | 4 | no | no | no |
| 3 |  |  |  | 4 | yes | yes | yes |

FIGURE 4.2: One edge subgraphs contained in the example undirected graph $G$ shown in Figure 4.1

## 4.4 Examples of directed VULS

In this section, we consider directed graphs. Consider the directed graph $G$ given in Figure 4.4. We can encode the graph as a set of five tuples of the form $\langle u, v, l_u, l_e, l_v\rangle$ where $u, v \in V$, $e = (u, v) \in E$, $l_u, l_v \in L_V$ and $l_e \in L_E$ (again note that $u$ and $v$ can

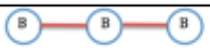
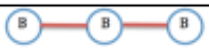
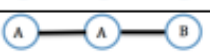
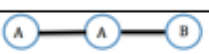
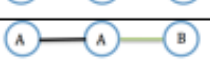
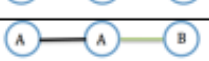
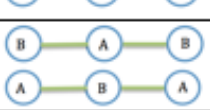
| ID | 2-edge VULS candidates | Candidates matching vertices' labels | 2-edge VULS | Occ. Count | Freq. VULS? | Min. VULS ? | Min. Freq. VULS ? |
|---|---|---|---|---|---|---|---|
| 4 |  |  |  | 4 | yes | no | no |
| 5 |  |  |  | 4 | yes | no | no |
| 6 |  |  |  | 4 | yes | no | no |
| 7 |  |  |  | 2 | no | yes | no |
| 8 |  |  |  | 4 | yes | no | no |
| 9 |  |  | | 4 | no | no | no |

FIGURE 4.3: Two edge subgraphs contained in the example undirected graph $G$ shown in Figure 4.1

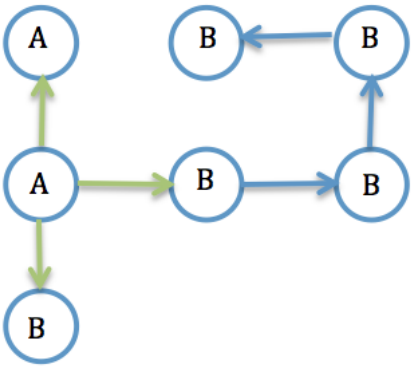



FIGURE 4.4: Example of directed graph $G$

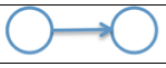
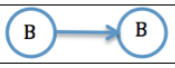
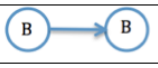
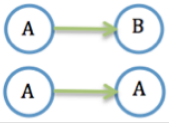
| ID | 1-edge VULS candidates | Candidates matching vertices' labels | 1-edge VULS | Occ. Count | Freq. VULS ? | Min. VULS ? | Min. Freq VULS ? |
|---|---|---|---|---|---|---|---|
| 1 |  |  |  | 3 | yes | yes | yes |
| 2 |  |  | | | | | |

FIGURE 4.5: one edge subgraphs contained in the example directed graph $G$ shown in Figure 4.4

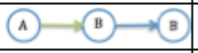
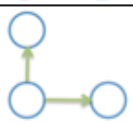
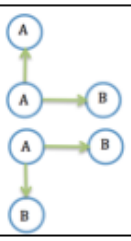
| ID | 2-edge VULS candidates | Candidates matching vertices' labels | 2-edge VULS | Occ. Count | Freq. VULS ? | Min. VULS ? | Min. Freq. VULS ? |
|---|---|---|---|---|---|---|---|
| 3 |  |  |  | 1 | no | no | no |
| 4 |  |  |  | 2 | yes | no | no |
| 5 |  |  | | | | | |

FIGURE 4.6: Two edge subgraphs contained in the example directed graph $G$ shown in Figure 4.4

also be seen as "vertex position indicators" in a specific subgraph). Thus, with reference to Figure 4.4:

$$L_V = \{A, B\} \qquad\qquad L_E = \{green, blue\}$$

Figures 4.5 and 4.6 show respectively the one and two edge VULS candidates that exist in the example directed graph. As before the Figures include the occurrence counts for each of the identified VULS candidates. The notation "_" is again used to define a subgraph without vertex labels. With reference to Figure 4.5, subgraph 1 ($\langle 0, 1, \_, blue, \_ \rangle$) is a VULS, because it has a unique vertex labelling associated with it. Subgraph 2 is not a VULS because it has two potential vertex labelings. In addition, assuming a threshold of $\sigma = 2$, subgraph 1 can be said to be frequent VULS (because the occurrence count is greater than the value for $\sigma$). Subgraph 1 is also a minimal VULS because it does not contain subgraphs that are themselves VULS. Again, by definition subgraph 1 is also a minimal frequent VULS.

With reference to Figure 4.6 subgraphs 3 and 4 are VULS. However, subgraph 5 is not a VULS because different vertex labelings can be applied to this configuration (the labelling is therefore not unique). Again, assuming $\sigma = 2$, subgraph 4 is also a frequent VULS. However, subgraphs 3 and 4 are not minimal VULS because they contain subgraphs that are also VULS. Figure 4.6 does not feature any subgraph that is a minimal frequent VULS.

In this case where the VULS size is less than 2 edges, coverage with reference to each category of VULS is as follows: (i) complete=5/7=71.43%, (ii) minimal=4/7=57.14%, (iii) frequent=4/7=57.14%, and (iv) minimal frequent =4/7=57.14%. The above thus serves to further illustrate the relationship between the four different identified categories of VULS considered in this thesis (as initially illustrated in Figure 2.2).

## 4.5 Summary

This chapter has presents a formalisation of the concept of VULS. It has identified and defined four different categories of VULS: (i) complete, (ii) minimal, (iii) frequent and (iv) minimal frequent, and illustrated the distinction between these different categories using a set of examples. The examples were given in terms of both undirected and directed graphs. This chapter also noted that "coverage" is an important concept in the context of VULS and presented how this is calculated. The significance of the coverage concept will become more apparent later in this thesis. In terms of coverage it is important to note that we can expect the coverage when conducting complete VULS mining to be greater than when conducting frequent or minimal VULS mining. Similarly we can expect the obtained coverage when conducting frequent or minimal VULS mining to be greater than when conducting minimal frequent VULS mining. The extent of the differences in coverage and the significance of these differences will be explored further later in this thesis. In the next chapter the VULS mining process, and associated algorithms, will be considered in detail in terms of each category of VULS.

# Chapter 5

# Algorithms for VULS Mining

## 5.1 Introduction

This chapter presents the four VULS mining algorithms proposed in this thesis: (i) compVULSM, (ii) minVULSM, (iii) freqVULSM and (iv) minFreqVULSM. Each is directed at one of the four different categories of VULS identified in the previous chapter. A schematic of the VULS model generation process is given in Figure 5.1. The process commences with the provision of pre-labelled raw 3D surface training data. The assumption is that this is not in the appropriate graph format and thus needs to be translated into this format. This is the case with respect to the sheet metal forming and satellite image applications used for evaluation purposes later in this thesis, but of course it may also be the case that the input data is already in the required graph format in which case the preprocessing step can be omitted. Whatever the case the graph training data is passed on to the mining stage where one of the proposed algorithms is applied (depending on whether we want to find the complete set of VULS, minimal VULS, frequent VULS, or minimal frequent VULS). The result is a "VULS model" which can then be used to predict vertex labels in unlabelled graph data. The process of predicting vertex labels using a "VULS model" will be described in the next chapter.

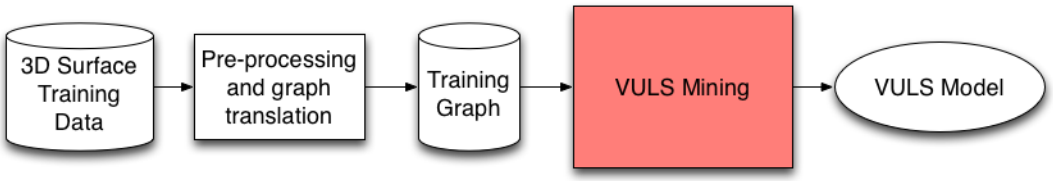



FIGURE 5.1: VULS model generation process.

The rest of this chapter is organized as follows. The pseudo codes for each VULS mining algorithms are presented in Sections 5.2 to 5.5 respectively. In each case the description of the algorithm includes a Worked example using the graph depicted in Figure 5.2. Finally, in Section 5.6, the chapter is concluded with a brief summary.

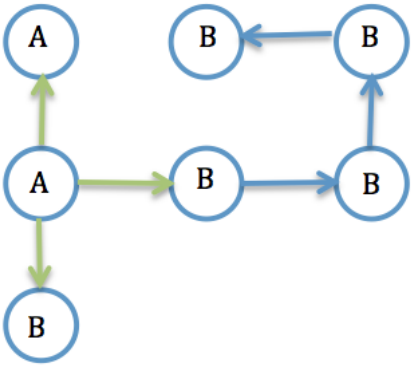



FIGURE 5.2: Example graph $G_{train}$.

## 5.2 The compVULSM Algorithm

This section presents the compVULSM algorithm for mining the complete set of VULS contained in the input graph. Mining the complete set of VULS entails traversing the entire search space starting with $k = 1$ edge subgraphs and continuing until some user specified maximum number of edges ($max$) is reached. Recall that in the context of VULS vertex classification very large VULS are unlikely to be of much use as they will tend to be "overfitted" to the training set and therefore unlikely to appear in any unseen data. Note also that if a $k$-edge subgraph is not a VULS this does not necessarily mean that its $(k + 1)$-edge super-graphs will also not be VULS (and vice versa). Thus the process for finding the complete set of VULS is exhaustive, involving exponential time complexity as the size of the input (training) graph increases. Consequently complete VULS mining is computationally expensive.

The high level pseudo code for the compVULSM algorithm is presented in Algorithm 4. The input is a labelled (training) graph $G_{train}$ and a value $max$ for the maximum size of any identified VULS. Note that the input graph $G_{train}$ is encoded using minimal Depth First Search (DFS) lexicographical ordering; as noted in Section 2.2.4 this is also the canonical form used by the well known gSpan algorithm [229]. The output is a complete set of identified VULS $\mathcal{U}$. Note that we limit the size of the searched-for VULS using the constant $max$; if we do not do this the entire input graph may ultimately be identified as a VULS. As already noted, for vertex classification purposes, large VULS are undesirable. At the commencement of the compVULSM algorithm, the set $\mathcal{U}$ will be empty (line 4). Note also that $max \leq |E_G|$, where $E_G$ is the set of edges in $G_{train}$, as otherwise we will attempt to generate candidate VULS that are bigger than the input graph. We proceed in a breadth first manner commencing with one edge candidate VULS, then two edge candidate VULS, and continuing until $max$ edge candidate VULS are arrived at. On each iteration we first identify the $k$-edge VULS contained in the set of candidate VULS $G_k$ (line 6), using the $genVULS$ procedure, and include them in the set $\mathcal{U}$, then we continue and generate the $k + 1$ edge candidate VULS (line 7) using the $subgraph\_Mining$ procedure (Algorithm 6), and so on.

**Algorithm 4** The compVULSM algorithm

```
1: procedure main(G_train, max)
2:     k = 1
3:     G_k =the set of k-edge subgraphs in G_train (candidate VULS)
4:     U = ∅
5:     while (k < max) do
6:         U = U ∪ genVULS(k,G_k)
7:         G_{k+1}= subgraph_Mining(G_train, k + 1)(Algorithm 6)
8:         k = k + 1
9:     end while
10: end procedure

11: procedure genVULS(k, G_k)
12:     U' = ∅
13:     for all c ∈ G_k do
14:         if isVULS(c) then /* Algorithm 5 */
15:             U' = U' ∪ c
16:             coverage = compute coverage using Equ (4.1)
17:             if coverage == 100% then
18:                 exit
19:             end if
20:         end if
21:     end for
22:     return U'
23: end procedure
```

**Algorithm 5** isVULS procedure

```
1: procedure isVULS(c)
2:     isaVULS = true
3:     F(c) → S' /* The vertex label list S' of subgraph c for each vertex V_i in V_c is
       drawn from L_V and mapped using the function F according to input graph G_train*/
4:     for all L_{v_i} ∈ S' do
5:         if |L_{v_i}| ≠ 1 then
6:             isaVULS = false
7:             break
8:         end if
9:     end for
10:    return isaVULS
11: end procedure
```

**Algorithm 6** subgraph_Mining, return a set of k-edge subgraphs $G_k$. $G_{temp}$ is a local variable represents set of subgraphs generated so far, $G_{temp} = \emptyset$ at the beginning.

```
1: procedure subgraph_Mining(G_train, k)
2:     G_k = ∅
3:     G_1=the set of one-edge subgraphs in G_train
4:     sort G_1 in DFS lexicographic order
5:     for each edge e ∈ G_1 do
6:         G_k = G_k∪ subgraph(e,1,k)
7:         G_train = (G_train − e) (remove e from G_train)
8:     end for
9:     Return G_k
10: end procedure

11: procedure subgraph(e, size, k)
12:     if size == k then
13:         return G_temp
14:     end if
15:     generate all e's potential extension subgraphs g with one edge growth by right
    most extension in G_train
16:     for each g do
17:         if g is minimal DFSCode then
18:             G_temp = G_temp ∪ g
19:             subgraph(G_temp,size+1,k)
20:         end if
21:     end for
22: end procedure
```

The *genVULS* procedure takes as input a set of $k$ edge candidate VULS ($G_k$) and returns a set $\mathcal{U}'$ of discovered $k$-edge VULS (if any). Each candidate subgraph $c$ in $G_k$ is processed in turn (line 14). Note that each subgraph $c$ in $G_k$ has the form $\langle \mathrm{V}_g, E_g, L_{E_g}, F \rangle$ (no specified vertex labels). If a candidate graph $c$ is identified as a VULS it is appended to the set $\mathcal{U}'$ (line 15). To determine whether a candidate VULS $c$ is a VULS or not the isVULS procedure is called.

The isVULS procedure is given in Algorithm 5. In line 3 a list of sets of possible labels for each vertex in the current candidate VULS $c$ is determined (as already noted previously in Chapter 4, each subgraph $c$ in $G_k$ has the form $\langle \mathrm{V}_c, E_c, L_{E_c}, F \rangle$; no specified vertex labels). The vertex label list $S'$ of subgraph c for each vertex $V_i$ in $V_c$ is drawn from $L_V$ and mapped using the function $F$ according to input graph $G_{train}$: $F(c) \rightarrow S'$; thus there is a one to one correspondence between the vertex sets in $V_c$ and the vertices' labels $L_V$ in $S'$. If the size of any of the sets in $S'$ is greater then one, then $c$ is not unique (line 5) and thus $c$ is not a VULS and the procedure returns *false*. Otherwise the procedure returns *true*. The proposed *subgraph_Mining* procedure is presented in algorithm 6. This procedure is similar to that found in gSpan except that we are finding all subgraphs up to a size $k$, as opposed to only frequent subgraphs (the anti-monotone property does not apply when mining VULS). We commence (line 4) by sorting all

the one-edge subgraphs, from input graph $G_{train}$, into DFS lexicographic order and storing them in $G_1$. Then (lines 5-8), for each one edge subgraph $e$ in $G_1$, the *subgraph* procedure is called (line 6), which finds all supergraphs for each one edge graph $e$ up to size $k$, and stores the result in $G_k$ (line 6). As we proceed, each subgraph $e$, whose supergraphs have all been found, is removed from graph $G_{train}$ so as to avoid duplicate supergraph generation.

The $subgraph(e, size, k)$ procedure generates all the supergraphs of the given one edge subgraph $e$ by growing $e$ by adding edges using the right most extension principle as already described previously in Chapter 2. For each candidate subgraph $g$, if $g$ is described by a minimal DFS Code (line 17), the super graph of $e$ is stored in $G_{temp}$ (line 18). The process continues in this recursive manner until the number of edges in the supergraphs to be generated ($size$) is greater than $k$ (line 13).

A Worked example of the process of generating the complete set of VULS using the compVULSM algorithm, with $max = 3$ is given in Figure 5.3. The input to the Worked example is the training graph presented in Figure 5.2. Figure 5.3 is organised in a tabular format with the rows representing iterations and the columns representing: (i) VULS candidates, (ii) labelled candidates and (iii) whether each candidate is a VULS or not. Recall that the input graph $G_{train}$ is encoded using minimal Depth First Search (DFS) lexicographical encoding. Recall also that a candidate VULS is a subgraph without vertex labelling.

The numbers attached to vertices in the candidate VULS column in Figure 5.3 are simply vertex position indicators. Before considering the figure in more detail, with respect to the discussion below. The notation "_" is used to define a vertex without a label (thus a place holder indicated by vertex label list $S'$ later).

On the first iteration where K=1 (second row in Figure 5.3), two one-edge candidate VULS were identified (the set $G_1$): ⟨ 0, 1, _ , green, _ ⟩ and ⟨ 0, 1, _ , blue, _ ⟩. For each of these candidates a vertex label list, $S'$, was generated according to function $F(c) \rightarrow S'$ applied to $G_{train}$. Using the set $S'$, three vertex labelled candidate VULS were identified: ⟨ 0, 1, A, green, A ⟩, ⟨ 0, 1, A, green, B ⟩ and ⟨ 0, 1, B, blue, B ⟩. Thus ⟨ 0, 1, _ , green, _ ⟩ is not a VULS because it has two potential vertex labelings (⟨ 0, 1, A, green, A ⟩, ⟨ 0, 1, A, green, B ⟩); while ⟨ 0, 1, B, blue, B ⟩ is a VULS because it has a unique vertex labelling associated with it.

As K is increased to 2 (coverage is not 100% and the max value for k has not been reached), on the second iteration (K=2), three two-edge candidate VULS are generated from $G_1$ by right most extension. In the same manner as before, for all $c \in G_2$ the vertex label list $S'$ was generated according to function $F(c) \rightarrow S'$ applied to $G_{train}$. In this case, the following set of labelled candidate VULS were identified:

(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, B ⟩), and
(⟨ 0, 1, B, blue, B ⟩, ⟨ 1, 2, B, blue, B ⟩).

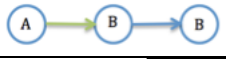
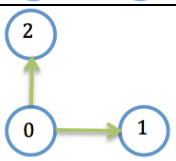
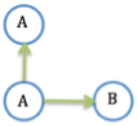
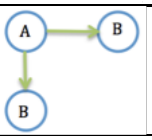
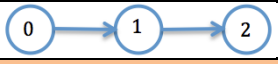
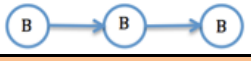
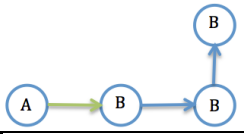
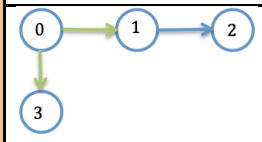
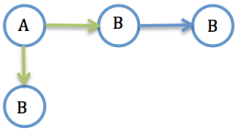
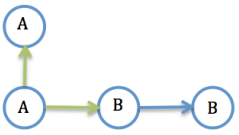
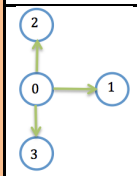
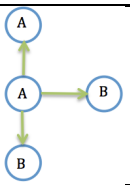
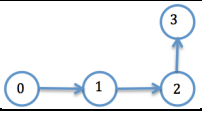
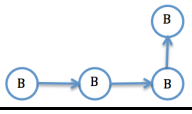
| Iteration number | Candidate VULS | Labeled candidate VULS | VULS Yes/No ? |
|---|---|---|---|
| K=1 |  |  | Not VULS |
| | |  | |
| |  |  | VULS |
| K=2 | Candidate VULS | Labeled candidate VULS | VULS Yes/No ? |
| |  |  | VULS |
| |  |  | Not VULS |
| | |  | |
| |  |  | VULS |
| K=3 | Candidate VULS | Labeled candidate VULS | VULS Yes/No ? |
| |  |  | VULS |
| |  |  | Not VULS |
| | |  | |
| |  |  | VULS |
| |  |  | VULS |

FIGURE 5.3: Worked example of complete VULS mining where $max = 3$

Of these (⟨ 0, 1, _ , green, _ ⟩, ⟨ 0, 2, _ , green, _ ⟩) is not a VULS because it has the same structure but two potential vertex labelings; (⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩) and (⟨ 0, 1, B, blue, B ⟩, ⟨ 1, 2, B, blue, B ⟩) are VULS because they have unique vertex labellings associated with them.

We then move on to k=3 (coverage was not yet 100% and the max value has not yet been reached), a set of four three-edge candidate VULS ($G_3$) will be generated from $G_2$ by right most extension. Vertex labels were again generated as before to give:

(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 2, 3, B, blue, B ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 0, 3, A, green, B ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 0, 3, A, green, A ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩, ⟨ 0, 3, A, green, B ⟩), and
(⟨ 0, 1, B, blue, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 2, 3, B, blue, B ⟩).

Of these (⟨ 0, 1, _ , green, _ ⟩, ⟨ 1, 2, _ , blue, _ ⟩, ⟨ 0, 3, _ , green, _ ⟩) is not a VULS because it has the same structure, but two potential vertex labelings:

(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 0, 3, A, green, B ⟩), and
(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 0, 3, A, green, A ⟩).

The rest of the three-edge VULS candidates are VULS because they do have unique vertex labellings. K has now reached the maximum pre-specified value of $max = 3$, thus the algorithm stops, the complete set of VULS of size less than or equal to 3 have been identified.

## 5.3 Minimal VULS Mining

The main issue with the compVULSM algorithm, as described above, is the significant computational overhead required to identify the complete set of VULS (as will be demonstrated later in Chapter 7). Since the process of identifying the complete set of VULS is slow and time-consuming, one solution to this is to find a subset of the complete set of VULS that requires less computational cost but still gives good coverage. One idea is to find the set of minimal VULS. The intuition here was that the set of minimal VULS will be suitably representative of the complete set of VULS because every VULS in the complete set will either contain at least one minimal VULS or will be a minimal VULS itself. Recall that a minimal VULS $\phi$ is one where none of its subgraphs are VULS (but its supergraphs may be). The advantage offered by the minimal VULS approach is that when we proceed in a breadth first manner, starting with one edge candidate graphs, we do not need to grow the $(k+1)$-edge candidate subgraphs from the identified $k$-edge minimal VULS. In the context of coverage it is conjectured that good coverage will still be maintained because at least one minimal VULS must appear in any non-minimal VULS.

The pseudo code for the minVULSM algorithm is presented in Algorithm 7. The input is again a labelled graph $G_{train}$ and a value $max$. The output is a set of minimal

**Algorithm 7** minVULSM algorithm

```
1: procedure main(G_train, max)
2:     k = 1
3:     G = G_train (Part of input graph not covered by minimal VULS)
4:     G_k =the set of k-edge subgraphs in G (candidate VULS)
5:     T_k = ∅ (the set of k-edge subgraphs which are not VULS)
6:     U = ∅
7:     while (k < max) do
8:         U = U ∪ genMinVULS(k,G_k)
9:         G_{k+1}= subgraph_Mining(G, k + 1)(Algorithm 6 where e ∈ T_k)
10:        k = k + 1
11:    end while
12: end procedure

13: procedure genMinVULS(k, G_k)
14:    U' = ∅
15:    for all c ∈ G_k do
16:        if isVULS(c) then /* Algorithm 5 */
17:            U' = U' ∪ c
18:            coverage = compute coverage using Equ (4.1)
19:            if coverage == 100% then
20:                exit
21:            end if
22:            G = G − c
23:        else
24:            T_k = T_k ∪ c
25:        end if
26:    end for
27:    if T_k == ∅ then
28:        exit
29:    end if
30:    return U'
31: end procedure
```

VULS $\mathcal{U}$. Note that the algorithm is similar to algorithm 4 (presented above) for mining the complete set of VULS, except that on each iteration we capture the current set of $k$-edge VULS in a set $\mathcal{U}$ (line 8) and remove the identified $k$ edge VULS from $G_{train}$ to give $G$ from which the following $(k+1)$-edge subgraphs will be generated (line 9). In other words, we only extend the set of $k$ edge non-VULS to generate the next $(k+1)$-edge VULS candidates on each iteration. Note that the $isVULS$ and $Subgraph\_Mining$ procedures were reused as presented earlier (Algorithms 5 and 6).

Continuing with Algorithm 7, the $genMinVULS$ procedure takes as input the current graph size $k$ (where $k$ is the number of edges) and the set of $k$-edge subgraphs contained in the set $G_k$ as pruned so far. The procedure returns the set of $k$-edge minimal VULS $\mathcal{U}'$. On each call the procedure $genMinVULS$ loops through the input set of $k$-edge subgraphs and (line 15) for each subgraph $c$ in $\mathcal{G}_k$ determines whether it is a

VULS or not by calling Algorithm 5 described earlier. If $c$ is a VULS it is added to the set $\mathcal{U}'$ (line 17). We then (line 18) calculate the coverage so far, if it has reached 100% we have found the complete set of minimal VULS and we exit (line 20). Note that if coverage is equal to 100%, the input set $G$ will now be empty. Otherwise, if the coverage is not 100%, we continue processing and (line 22) remove $c$ from the global set $G$. If $c$ is not a VULS we add it to $T_k$ (line 24), $T_k$ is the set of $k$-edge subgraphs which will be extended to form $G_{k+1}$, the set of $(k+1)$-edge subgraphs, ready for the next level of processing. Eventually all $c$ in $G_k$ will have been processed. If, at this stage $T_k$ is empty there will be no more subgraphs that can be generated and the process will exit (line 28). Otherwise control will return to the *main* procedure at line 8 and the set of $(k+1)$-edge subgraphs will be generated from $T_k$ (the set of $k$-edge subgraphs that are not VULS) using right most extension coupled with isomorphism checking (using the *subgraph_Mining* procedure described previously). We continue in this manner until the maximum value for $k$ is reached or coverage reaches 100%.

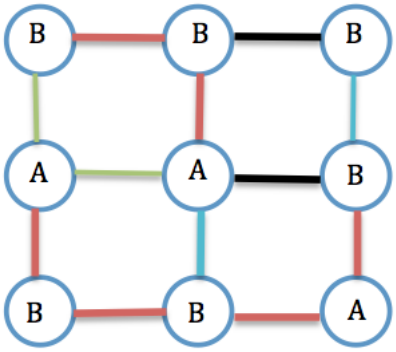



FIGURE 5.4: Input graph $G_{train}$, $G = G_{train}$ at the beginning of the algorithm 7


FIGURE 5.5: Example of 2-edge minimal VULS $c$

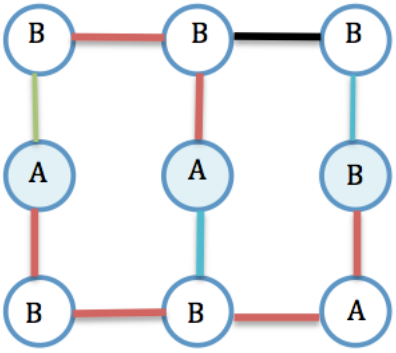



FIGURE 5.6: $G = G - c$

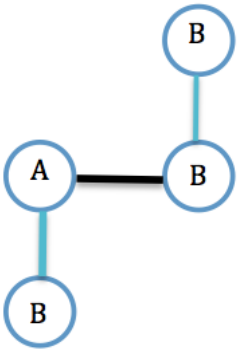



FIGURE 5.7: Example of 3-edge minimal VULS which is missed

Note that as the procedure progresses the identified minimal VULS were "marked up" in the input graph $G$ so that they would not be used further in the VULS identification process. However, this sometimes resulted in the input graph becoming fragmented (disconnected). Fragments that were smaller than the current value for $k$ could be safely

ignored, but the remainder still had to be considered during the VULS identification process. It was also found that, given a particular configuration, some VULS were missed. For example where a three edge and a two edge minimal VULS share a one edge subgraph that was not a VULS, in this case if the two edge occurrences in $c$ were marked up, the three edge minimal VULS might never be identified. This is illustrated with a Worked example in Figures 5.4 to 5.7. Figure 5.4 gives the input graph $G_{train}$, $G = G_{train}$ at the beginning of algorithm 7. A two-edge VULS $c$ (as shown in Figure 5.5) is identified from $G$, as the algorithm proceeds, this two-edge VULS c will be marked and the associated edges will be removed from $G$ as shown in Figure 5.6. In this case, the three-edge VULS (as shown in Figure 5.7), which shares the one-edge subgraph ⟨ A, Black,B ⟩ with $c$, can not be identified from $G$ in Figure 5.6.

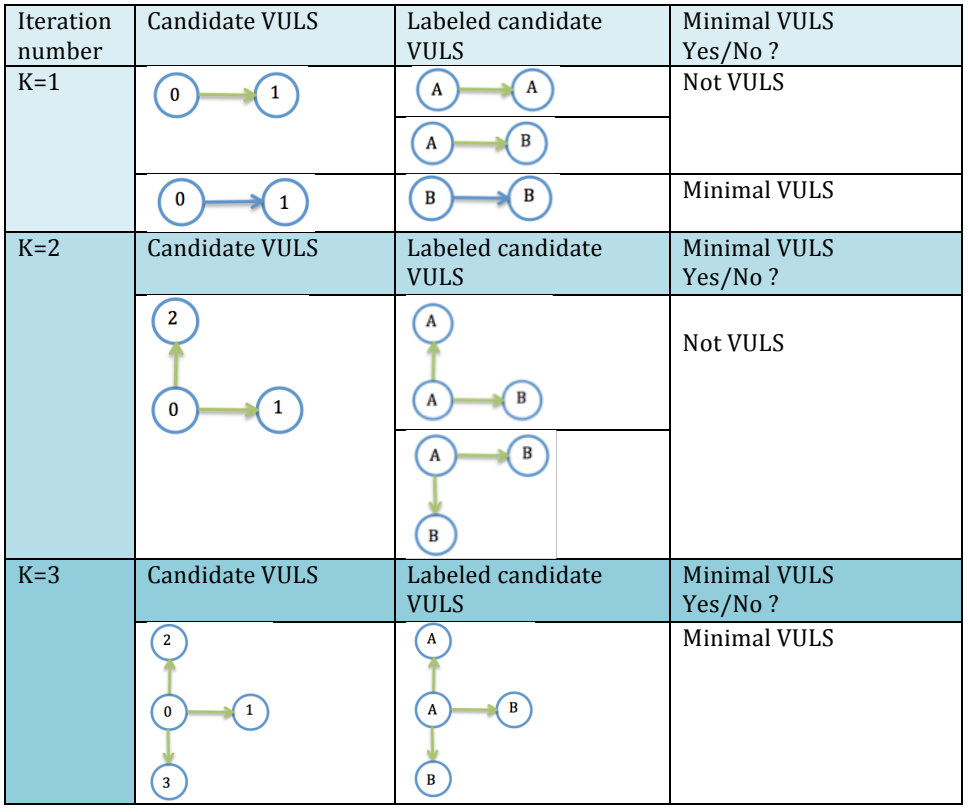



FIGURE 5.8: Worked example of minimal VULS mining where $max = 3$

A Worked example for the process of identifying all minimal VULS using the min-VULSM algorithm (Algorithm 7), with max=3, is given in Figure 5.8. The input to the Worked example is the training graph presented previously in Figure 5.2. On the first iteration where K=1 (second row in Figure 5.8), two one-edge candidate VULS were identified: ⟨0, 1, _ , green, _⟩ and ⟨ 0, 1, _ , blue, _ ⟩ as before, for each of these candidates a vertex label list $S'$ was generated according to function $F(c) \rightarrow S'$ applied to $G_{train}$. Using the set $S'$, three vertex labelled candidate VULS were identified: ⟨ 0, 1, A, green, A ⟩, ⟨ 0, 1, A, green, B ⟩ and ⟨ 0, 1, B, blue, B ⟩. Of these ⟨ 0, 1, _ , green, _

⟩ is not a VULS because it has two potential vertex labelings (⟨ 0, 1, A, green, A ⟩, ⟨ 0, 1, A, green, B ⟩); ⟨ 0, 1, B, blue, B ⟩ is a minimal VULS because it has a unique vertex labelling associated with it, and it does not contain any subgraphs that are themselves VULS. This minimal VULS ⟨ 0, 1, B, blue, B ⟩ will be marked up in $G$ ($G = G_{train}$) so that it won't be extended further. As K is increased to 2 (coverage is not 100% and the max value for k has not been reached), one two-edge candidate VULS ($G_2$) is generated from the set of non-VULS $G_1$ by right most extension. In the same manner as before, for all $c \in G_2$ the vertex label list $S'$ was generated, to give the following set of labelled candidate VULS:

(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩), and
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, B ⟩).

Of these (⟨ 0, 1, _ , green, _ ⟩, ⟨ 0, 2, _ , green, _ ⟩) is not a VULS because it has the same structure but two potential vertex labelings:

(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩), and
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, B ⟩).

We then move on to k=3 (coverage is not yet 100% and the max value has not been reached); in this case, one three-edge candidate VULS ($G_3$) is generated from the set of non-VULS $G_2$ by right most extension. Vertex labels were again generated as before to give: (⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩, ⟨ 0, 3, A, green, B ⟩). This subgraph is a minimal VULS since it does not contain any subgraphs that are themselves VULS. K has now reached the specified maximum value of $max = 3$, thus the algorithm stops. All minimal VULS of size less than or equal to 3 have been identified.

Clearly fewer VULS candidates are generated using minimal VULS mining than when mining the complete set of VULS; thus mining minimal VULS is more efficient than when mining the complete set of VULS as will be confirmed by the evaluation presented later in chapters 7 and 8.

## 5.4 Frequent VULS Mining

An alternative way to reduce the complexity of VULS is to mine only frequent VULS. An idea taken from the concept of frequent subgraph mining [115, 118, 136] and indirectly from the concept of frequent item set mining [2, 21]. As noted in Chapter 4 a frequent VULS is one whose occurrence count exceeds some threshold value $\sigma$. Note that the anti-monotone property utilised in frequent subgraph mining (and frequent item set mining) where if a subgraph is infrequent none of its super graphs will be frequent does not apply.

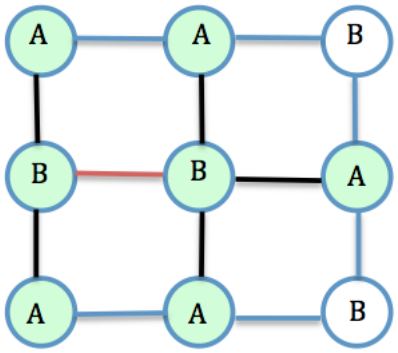



FIGURE 5.9: Example graph $G_{train}$.

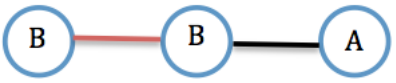



FIGURE 5.10: Example of 2-edge frequent subgraphs extended from the 1-edge infrequent subgraph ⟨ B, red, B ⟩ in Figure 5.9.

The fact that the anti-monotone property does not hold is illustrated by example in Figures 5.9 to 5.10. Figure 5.9 gives an example input graph while Figure 5.10 shows a 2−edge subgraph extracted from the example graph. If we set threshold $\sigma$ as 4. In Figure 5.9, the one edge subgraph ⟨ B, red, B ⟩ occurrs only once, obviously not frequent. However, the frequent two edge subgraph (as shown in Figure 5.10), extended from the infrequent subgraph ⟨ B, red, B ⟩, occurs 5 times, more than threshold $\sigma$. Thus for the Frequent VULS Mining algorithm both frequent and infrequent subgraphs will be extended further. The motivation for frequent VULS mining was that it was anticipated that it would be more computationally efficient than complete VULS mining while at the same time coverage would still be good because we would be identifying commonly occurring VULS. Whether frequent VULS mining would be more effective and/or efficient than minimal VULS mining was initially unclear (this is explored further in Chapters 7 and 8).

An issue with any form of frequent pattern mining is the nature of the threshold $\sigma$. Recall from Chapter 2, that in the case of frequent subgraph mining (transaction graph mining) [47, 115, 118, 136, 221] we express $\sigma$ in terms of a proportion of the number of transaction graphs under consideration (a subgraph is then frequent if it appears in $x\%$ of the available set of transaction graphs). In single graph mining this is not so straight forward, $\sigma$ can of course be predefined in terms of a fixed value but this does not take account of the different numbers of graphs that might be found on each iteration. With respect to the work presented in this thesis the value $\sigma$ is calculated dynamically in terms of the average *Occurrence* (occurrence count) values for the complete set of candidate VULS identified on each iteration as shown in Equation 5.1 where $g_i \in G_k$ and *Occurrence* is a function that returns the occurrence count of $g_i$. For example, there are 5 4-edge subgraphs $g_1$, $g_2$, $g_3$, $g_4$, $g_5$ in $G_4$ on the fourth iteration, thus $|G_4|$=5. The occurrence count of each subgraph is: Occurrence($g_1$)=1, Occurrence($g_2$)=5, Occurrence($g_3$)=3, Occurrence($g_4$)=2, Occurrence($g_5$)=4. Then, according to equation 5.1, $\sigma$ on this iteration will be calculated as $\sigma = (1 + 5 + 3 + 2 + 4)/5 = 3$. Note that this mechanism for calculating $\sigma$ is one of the author's own contributions with respect to this thesis.

$$\sigma = \frac{\sum_{i=1}^{i=|G_k|} Occurrence(g_i)}{|G_k|} \tag{5.1}$$

**Algorithm 8** freqVULSM algorithm

```
1: procedure main(G_train, max)
2:     k = 1
3:     G_k =the set of k-edge subgraphs in G_train (candidate VULS)
4:     U = ∅
5:     while (k < max) do
6:         U = U ∪ genFVULS(k,G_k)
7:         G_{k+1}= subgraph_Mining(G_train, k + 1)(Algorithm 6)
8:         k = k + 1
9:     end while
10: end procedure

11: procedure genFVULS(k, G_k)
12:     /* Calculate σ */
13:     Sup = Array of length |G_K| to hold occurrence counts
14:     total = 0
15:     for each g ∈ G_k do
16:         Sup[g] = Occurrence count of g in G_train
17:         total = total + sup[g]
18:     end for
19:     σ = total/|G_k| (Equation (5.1))
20:     /* Identify frequent candidate VULS */
21:     G'_k = ∅
22:     for each edge g ∈ G_k do
23:         if Sup[g] ≥ σ then
24:             G'_k = G'_k ∪ g
25:         end if
26:     end for
27:     U' = ∅
28:     for all c ∈ G'_k do
29:         if isVULS(c) then /* Algorithm 5 */
30:             U' = U' ∪ c
31:             coverage = compute coverage using Equ (4.1)
32:             if coverage == 100% then
33:                 exit
34:             end if
35:         end if
36:     end for
37:     return U'
38: end procedure
```

The pseudo code for the freqVULSM algorithm is presented in Algorithm 8. As before, the algorithm describes an iterative process. On each iteration five steps are undertaken: (i) calculation of the threshold $\sigma$ using Equation (5.1), (ii) pruning of the

set of candidate VULS according to $\sigma$, (iii) identification of VULS, (iv) coverage check and (v) calculation of the follow on set of candidate VULS. During the calculation of $\sigma$ (lines 12 to 19) the support for each subgraph $g$ in $G_k$ (the candidate set of VULS) is calculated. This requires an isomorphism checking procedure (not explicitly shown) which, given a reasonably sized input graph $G_{train}$, will be computationally expensive. Once the support counts for each $g$ in $G_k$ has been calculated, each $g$ in $G_k$ whose support count is greater than $\sigma$ is added to $G'_k$ (lines 22 to 26) to give a set of candidate $k$-edge frequent VULS. Next, using the candidate set $G'_k$, the $k$ sized VULS are identified in the same manner as before using Algorithm 5. After which a check for coverage is conducted. If the input graph $G_{train}$ is fully covered by the frequent VULS identified so far the algorithm exits. Otherwise the next set of candidate $(k+1)$-edge VULS are generated in the same manner as before using Algorithm 6. Then we proceed to the next iteration and repeat the process until either: (i) the maximum size of VULS is reached (indicated by the input $max$), (ii) 100% overage is reached, or (iii) no more frequent subgraphs (candidate VULS) can be generated.

A Worked example of the process of generating frequent VULS using Algorithm 8, with $max = 3$ is given in Figure 5.11. As in the case of the previous examples, the input to the Worked example is the training graph presented in Figure 5.2. On the first iteration where K=1 (second row in Figure 5.11), two one-edge candidate VULS were identified: ⟨ 0, 1, _ , green, _ ⟩ and ⟨ 0, 1, _ , blue, _ ⟩. For each of these candidates a vertex label list, $S'$, was generated according to function $F(c) \rightarrow S'$ applied to $G_{train}$. Using the set $S'$, three vertex labelled candidate VULS were identified: ⟨ 0, 1, A, green, A ⟩, ⟨ 0, 1, A, green, B ⟩ and ⟨ 0, 1, B, blue, B ⟩.

Meanwhile, threshold $\sigma$ is calculated using Equation 5.1 as 2; ⟨ 0, 1, _ , green, _ ⟩ is not a VULS because it has two potential vertex labelings (⟨ 0, 1, A, green, A ⟩, ⟨ 0, 1, A, green, B ⟩) and is thus not a frequent VULS. ⟨ 0, 1, B, blue, B ⟩ is a frequent VULS because it has a unique vertex labelling and the occurrence count of 3 exceeds $\sigma = 2$.

As K is increased to 2 (coverage is not 100% and the max value for K has not been reached), on the second iteration (K=2), three two-edge candidate VULS are generated from $G_1$ by right most extension. In the same manner as before, for all $c \in G_2$ the vertex label list, $S'$, was generated according to function $F(c) \rightarrow S'$. In this case, the following set of labelled candidate VULS were identified:

(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, B ⟩), and
(⟨ 0, 1, B, blue, B ⟩, ⟨ 1, 2, B, blue, B ⟩).

$\sigma$ is calculated as 1.25. (⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩) is a VULS but not a frequent VULS since the occurrence count (1) is less than $\sigma$ (1.25). (⟨ 0, 1, B, blue, B ⟩, ⟨ 1, 2, B, blue, B ⟩) is a frequent VULS because it has a unique vertex labelling and the occurrence count (2) exceeds $\sigma$ (1.25).

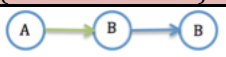
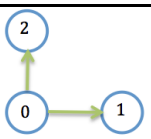
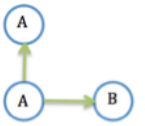
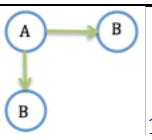
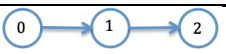
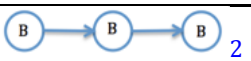
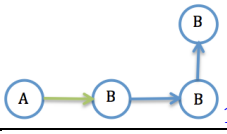
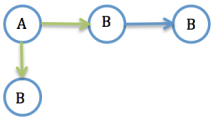
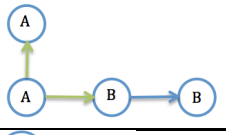
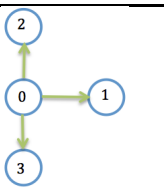
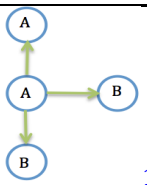
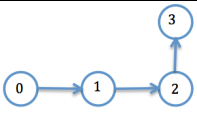
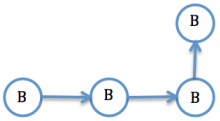
| Iteration Number ($\sigma$) | Candidate VULS | Labeled candidate VULS (Occurrence count) | Frequent VULS Yes/No ? |
|---|---|---|---|
| K=1<br>($\sigma = \frac{1+2+3}{3}=2$) |  |  1 | Not VULS |
| | |  2 | |
| |  |  3 | Frequent VULS |
| K=2<br>($\sigma = \frac{1+1+1+2}{4}=1.25$) | Candidate VULS | Labeled candidate VULS (Occurrence count) | Frequent VULS Yes/No ? |
| |  |  1 | VULS but not frequent VULS |
| |  |  1 | Not VULS |
| | |  1 | |
| |  |  2 | Frequent VULS |
| K=3<br>($\sigma = \frac{1+1+1+1+1}{5}=1$) | Candidate VULS | Labeled candidate VULS (Occurrence count) | Frequent VULS Yes/No ? |
| |  |  1 | Frequent VULS |
| |  |  1 | Not VULS |
| | |  1 | |
| |  |  1 | Frequent VULS |
| |  |  1 | Frequent VULS |

FIGURE 5.11: Worked example of frequent VULS mining where $max = 3$

We then move on to K=3 (coverage is not yet 100% and the max value has not been reached.) In this case, a set of four three-edge candidate VULS $G_3$ is generated from $G_2$ by right most extension. Vertex labels were again generated as before to give:

(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 2, 3, B, blue, B ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 0, 3, A, green, B ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 0, 3, A, green, A ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩, ⟨ 0, 3, A, green, B ⟩), and
(⟨ 0, 1, B, blue, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 2, 3, B, blue, B ⟩).

$\sigma$ is calculated as 1. Thus:

(⟨ 0, 1, A, green, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 2, 3, B, blue, B ⟩),
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩, ⟨ 0, 3, A, green, B ⟩), and
(⟨ 0, 1, B, blue, B ⟩, ⟨ 1, 2, B, blue, B ⟩, ⟨ 2, 3, B, blue, B ⟩)

are frequent VULS because they have unique vertex labellings and their occurrence count of 1 is greater than or equal to $\sigma = 1$. K has now reached the maximum value of $max = 3$, thus the algorithm stops. All frequent VULS of size less than or equal to 3 have been identified.

Unlike when mining minimal VULS, the number of candidate VULS generated is not reduced when mining frequent VULS. However, fewer frequent VULS are identified than mining for the complete set of VULS. This will be confirmed in the next two Chapters 7 and 8, where some experimental results will be presented.

## 5.5 Minimal Frequent VULS Mining

The idea behind the concept of minimal frequent VULS mining is that the time complexity advantages of both minimal and frequent VULS mining can be combined. However, it was anticipated that fewer VULS would be discovered than when finding only minimal or only frequent VULS, and thus coverage might be adversely affected (this is explored further in Chapter 7 and 8). As in the case of the minVULSM algorithm we can proceed in a breadth first manner and exclude subgraphs who have a parent graph that is a minimal VULS.

The pseudo code for the minFreqVULSM algorithm is presented in Algorithm 9. The algorithm is almost identical to Algorithm 8 except that, on each iteration, we store the identified non-minimal VULS in a set $T_k$ (line 39) and then use this set to generate the following set of $k + 1$ edge subgraphs $G_{k+1}$ (line 9).

**Algorithm 9** minFreqVULSM algorithm

1: **procedure** $main(G_{train}, max)$
2: $k = 1$
3: $G = G_{train}$(Part of input graph not covered by minimal VULS)
4: $G_k$ =the set of k-edge subgraphs in $G$ (candidate VULS)
5: $T_k = \emptyset$(the set of k-edge subgraphs which are not VULS)
6: $\mathcal{U} = \emptyset$
7: **while** $(k < max)$ **do**
8: $\mathcal{U} = \mathcal{U} \cup$ genFminVULS(k,$G_k$)
9: $G_{k+1}$= $subgraph\_Mining(G, k+1)$(Algorithm 6 where $e \in T_k$)
10: $k = k + 1$
11: **end while**
12: **end procedure**

13: **procedure** $genFminVULS(k, G_k)$
14: /* Calculate $\sigma$ */
15: $Sup$ = Array of length $|G_K|$ to hold occurrence counts
16: $total = 0$
17: **for** each $g \in G_k$ **do**
18: $Sup[g]$ = Occurrence count of $g$ in $G$
19: $total = total + sup[g]$
20: **end for**
21: $\sigma = \frac{total}{|G_k|}$ (Equation 5.1)
22: /* Identify frequent candidate VULS */
23: $G'_k = \emptyset$
24: **for** each edge $g \in G_k$ **do**
25: **if** $Sup[g] \geq \sigma$ **then**
26: $G'_k = G'_k \cup g$
27: **end if**
28: **end for**
29: $\mathcal{U}' = \emptyset$
30: **for all** $c \in G'_k$ **do**
31: **if** isVULS($c$) **then** /* Algorithm 5 */
32: $\mathcal{U}' = \mathcal{U}' \cup c$
33: $coverage$ = compute coverage using Equ (4.1)
34: **if** $coverage ==$ 100% **then**
35: exit
36: **end if**
37: $G = G - c$
38: **else**
39: $T_k = T_k \cup c$
40: **end if**
41: **end for**
42: **if** $T_k == \emptyset$ **then**
43: exit
44: **end if**
45: **return** $\mathcal{U}'$
46: **end procedure**

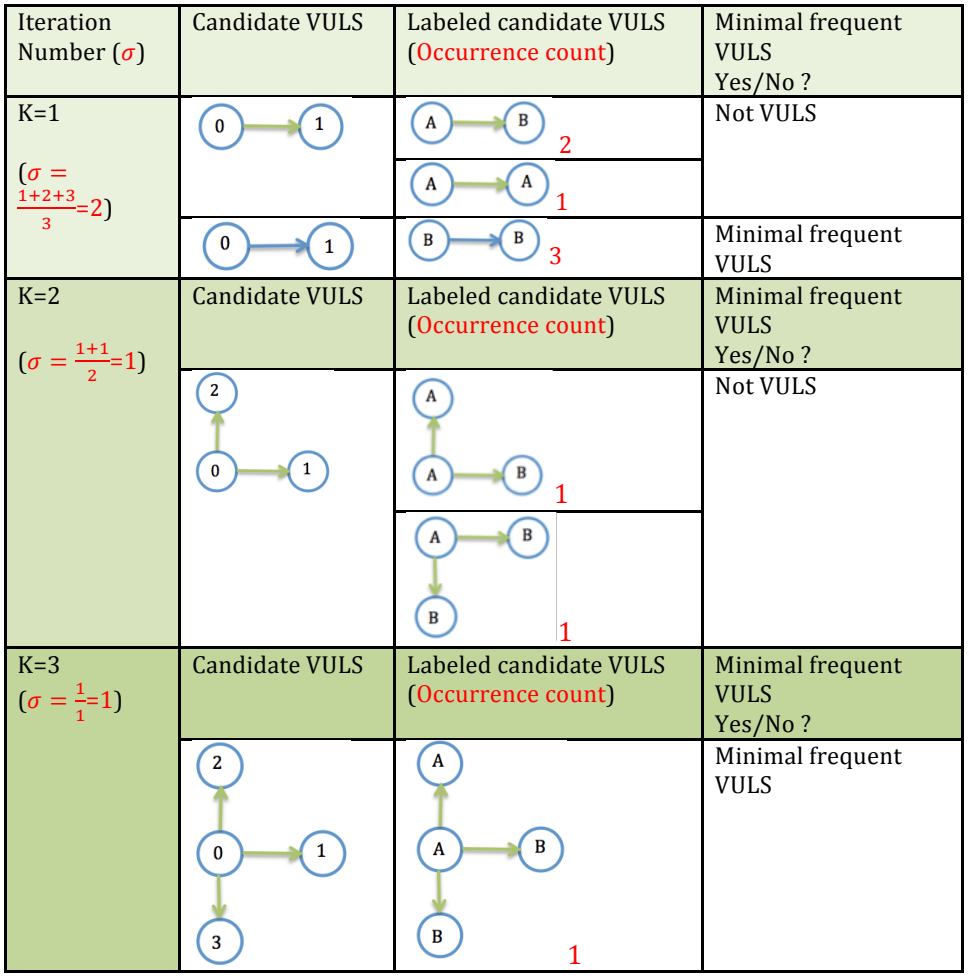



FIGURE 5.12: Worked example of minimal frequent VULSM mining where $max = 3$

A Worked example of the process of generating all minimal frequent VULS using Algorithm 9, with $max = 3$, is given in Figure 5.12. The input to the Worked example is again the training graph presented in Figure 5.2. On the first iteration where K=1 (second row in Figure 5.12), two one-edge candidate VULS were identified: ⟨ 0, 1, _ , green, _ ⟩ and ⟨ 0, 1, _ , blue, _ ⟩, For each of these candidates a vertex label list, $S'$, was generated using the function $F(c) \rightarrow S'$. Using the set $S'$ three vertex labelled candidate VULS were identified: ⟨ 0, 1, A, green, A ⟩, ⟨ 0, 1, A, green, B ⟩ and ⟨ 0, 1, B, blue, B ⟩. Meanwhile, threshold $\sigma$ is calculated using equation 5.1 as 2. ⟨ 0, 1, B, blue, B ⟩ is a minimal frequent VULS because it has a unique vertex labelling, it does not contain any subgraphs that are themselves VULS and its occurrence count of 3 exceeds the threshold $\sigma = 2$. Then the minimal frequent VULS ⟨0, 1, B, blue, B⟩ will be removed from $G$ ($G = G_{train}$) so that it won't be extended further. As K is increased to 2 (coverage is not 100% and the max value for K has not been reached), on the second iteration (K=2), one two-edge candidate VULS is generated from the set of non-VULS, $G_1$, by right most extension. In the same manner as before, for all $c \in G_2$ the vertex label list, $S'$, was generated using the function $F(c) \rightarrow S'$. In this case, the following

set of labelled candidate VULS were identified: (⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩), and (⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, B ⟩).

The threshold $\sigma$ is calculated as 1. (⟨ 0, 1, _ , green, _ ⟩, ⟨ 0, 2, _ , green, _ ⟩) is not a VULS because it has the same structure but two potential vertex labelings:

(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩), and
(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, B ⟩).

and thus it can't be minimal frequent VULS. We then move on to K=3 (coverage is not yet 100% and the max value has not been reached.) In this case, one three-edge candidate VULS ($G_3$) is generated from the set of non-VULS, $G_2$, by right most extension. Vertex label lists were again generated as before to giving:

(⟨ 0, 1, A, green, B ⟩, ⟨ 0, 2, A, green, A ⟩, ⟨ 0, 3, A, green, B ⟩).

This is a minimal frequent VULS since it does not contain any subgraphs that are themselves VULS and the occurrence count of 1 is greater than or equal to $\sigma = 1$. K has reached the specified maximum value of $max = 3$, and thus the algorithm stops. All minimal frequent VULS of size less than or equal to 3 have been identified.

It should be noted that the value for $\sigma$ is not necessarily the same when mining frequent VULS and minimal frequent VULS, although it is computed using the same equation, as the calculation depends on the occurrence count distribution of the generated candidate subgraphs on each iteration, hence slightly different sets of VULS may be generated and consequently slightly different coverage results obtained with respect to frequent VULS and minimal frequent VULS as will be demonstrated later in Chapter 7.

As in the case of mining minimal VULS, fewer VULS candidates will be generated when mining minimal frequent VULS than when mining the complete set of VULS. Thus mining minimal frequent VULS is likely to be more efficient than when mining the complete set of VULS as will be confirmed by the evaluation presented in Chapters 7 and 8.

## 5.6 Summary

This chapter has described the theory and operation of four different algorithms for finding the four different identified categories of VULS. The algorithms were as follows:

1. **compVULSM** for finding the complete set of VULS.
2. **minVULSM** for finding minimal VULS.
3. **freqVULSM** for finding frequent VULS.
4. **minFreqVULSM** for finding minimal frequent VULS.

In the next chapter, the Backward-Match-Voting algorithm for vertex classification will be presented and illustrated using a work example.

# Chapter 6

# Algorithm for Vertex Classification

## 6.1 Introduction

In the previous chapter the generation (mining) of VULS was considered. In this short chapter their utility (in the context of vertex classification) is considered. This chapter presents the Backward-Match-Voting algorithm for vertex classification using pre-labelled subgraphs such as the VULS identified using the VULS mining algorithms presented in the previous chapter. A schematic of the vertex classification process is given in Figure 6.1. Similar to the VULS training process presented in the previous chapter, with respect to Figure 6.1, it is again assumed that the new 3D surface data is not in the appropriate graph format (as in the case of the sheet metal forming and satellite image applications) and thus requires translation. Once in the correct grid graph format pre-labelled subgraphs, such as VULS, can be applied to this new graph data so that the labels of the vertices in the new graph can be predicted (labelled).

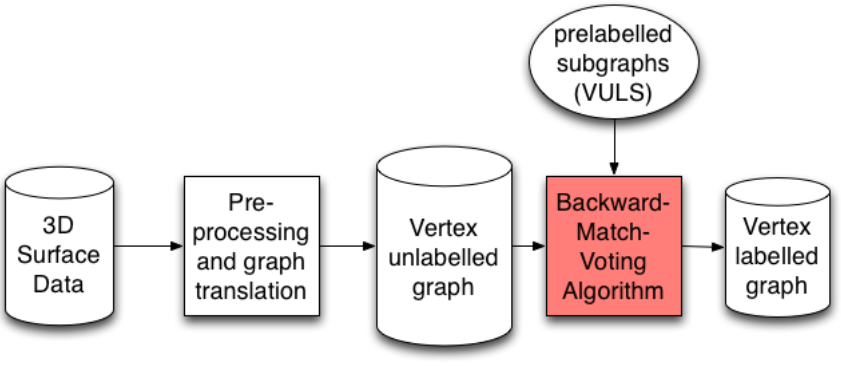



FIGURE 6.1: Schematic for predicting vertex labels given a new 3D surface data set.

Using the presented Backward-Match-Voting (BMV) algorithm labels will be attached to as many vertices in the new graph as possible. In the case of vertices that

are still unlabelled, a default label can be applied or alternatively the most frequently occurring label that features in the pre-labelled subgraphs used for training purposes can be used. The BMV algorithm is intended for application using collections of identified VULS, but this does not have to be the case, it can be used with respect to other subgraph collections.

The rest of this chapter is organized as follows. The BMV algorithm for vertex label classification (prediction) is presented in Section 6.2. A Working Example illustrating how the BMV algorithm operates is presented in Section 6.3. Section 6.4 then concludes the chapter with a brief summary.

## 6.2 Backward-Match-Voting algorithm

As noted in the above introduction, the idea is to use pre-labelled subgraphs, such as sets of identified VULS, to label the vertices in an unseen graph $G$ by matching the individual pre-labelled subgraphs with subgraphs in $G$. A general issue with this approach is that vertices in $G$ may be matched to more than one vertex in the given set of pre-labelled subgraphs, possibly with different vertex labels. In this case the conflicting labelling needs to be resolved. We can identify a potential number of mechanisms for doing this, but propose a voting mechanism here. A variety of voting mechanisms also exist, including: (i) majority voting and (ii) weighted voting. We adopted the first as the most appropriate weighting mechanisms to use was difficult to determine. We refer to the proposed vertex classification approach as the *Backward-Match-Voting* mechanism because: (i) we work "backwards" from the maximum value of $k = max$ to $k = 1$ (because this is more efficient as potentially a larger number of vertices in $G$ will be covered increasing the likelihood of reaching 100% coverage early on in the process), (ii) we "match" graph structures and edge labellings of pre-labelled subgraphs to $G$ so as to label the vertices in $G$ using the labels from pre-labelled subgraphs, and (iii) where more than one vertex label is assigned to a vertex in $G$ we use a majority "voting" scheme.

The Backward-Match-Voting algorithm is presented in Algorithm 10. The algorithm takes as input a collected set $\mathcal{U}$ of pre-labelled subgraphs (such as a set of VULS generated as described in the previous chapter) and a new graph $G$ which has known edge labels but unknown vertex labels. The algorithm also utilises the parameter $max$ as the maximum size of the pre-labelled subgraphs in $\mathcal{U}$ so that the algorithm does not try to find matches beyond this maximum size. In the case of using a set of VULS as the pre-labelled subgraphs, $max$ is set the same value as used by the VULS mining algorithm used to generate the VULS. The output is a vertex labelled graph $G$. The algorithm starts with pre-labelled subgraphs of size $max$ and iteratively proceeds with pre-labelled subgraphs of decreasing size until one edge pre-labelled subgraphs are reached or 100% coverage of the vertices in $G$ is obtained, whichever happens first. At the beginning of the algorithm, a set of vertex labels $L_V$ is initialized (line 11) to hold vertex labels

**Algorithm 10** : Backward-Match-Voting algorithm to predict vertex labels in a vertex-unlabelled graph

1: **Input:**
2: $\mathcal{U}$ = a set of pre-labelled subgraphs (such as VULS)
3: $G = \langle V, E, L_E \rangle$, Edge labelled graph (with unlabelled vertices),
4: $max$ = The maximum size of an element of $\mathcal{U}$
5: $L$ = default vertex label
6: **Output:**
7: Graph $G$ with labelled vertices

8: **procedure** $main(G$ , $max$, $\mathcal{U}$, $L)$
9: coverage=0
10: $k = max$
11: $L_V = \phi$
12: **while** $(k \geq 1)$ **do**
13: $G'$ = the set of all subgraphs in $G$.
14: **for all** $c \in \mathcal{U}$ where $|c| \equiv k$ **do**
15: **for each** $g \subseteq G'$ isomorphic to $c$ **do**
16: add $(v, l_v)$ to $L_V$, for each $v$ in $g$ with $l_v$ equal to the label of $v$ in $c$
17: **end for**
18: **end for**
19: $coverage$ = compute the coverage of $G$ using Equation 4.1
20: **if** $coverage \equiv 100\%$ **then**
21: exit
22: **end if**
23: $k = k - 1$
24: **end while**
25: **for all** $v$ in $G$ **do**
26: **if** $v$ occurs in $L_V$ **then**
27: set the label of $v$ in $G$ to the most frequent label of $v$ in $L_V$
28: **end if**
29: **end for**
30: Set the label of any remaining unlabelled vertex in $G$ to $L$
31: **end procedure**

extracted from relevant pre-labelled subgraphs in $\mathcal{U}$. Note that each element of $L_V$ corresponds to a vertex in $G$ that does not yet have a label associated with it (so all vertices in $G$ on start up). We then, on each iteration, process each pre-labelled subgraph $c$ of size $k$; if a $k$-edge subgraph $g$ within the input graph $G$ is isomorphic to $c$, each vertex in $g$ will be labelled according to vertex labelling in $c$. In this manner, part of $L_V$ will be populated on each iteration. Once all $k$-edge subgraphs in $\mathcal{U}$ have been processed the coverage is calculated (line 19), if coverage is equivalent to 100% the process stops (line 21), otherwise we continue with the $(k-1)$-edge subgraphs. Once all items in $\mathcal{U}$ have been processed, we then process each vertex $v$ in graph $G$. If a vertex $v$ has more than one label associated with it, the voting mechanism is invoked and the most popular label assigned to the corresponding vertex $v$ in $G$. It may be the case that some elements in

$L_V$ are still set to null indicating that the corresponding vertex has not been covered by any pre-labelled subgraph. In this case a default label is assigned. In case of the evaluation presented later in this thesis, the most frequent vertex label from the training data is used as the default label (lines 30).

Note that Backward-Match-Voting algorithm start with the largest pre-labelled subgraphs (pre-labelled subgraphs of size $max$) moving down to pre-labelled subgraphs of size 1. The reason for doing this, is that larger pre-labelled subgraphs are better at predicting vertex labels than smaller pre-labelled subgraphs because larger pre-labelled subgraphs define a more specific structure than smaller pre-labelled subgraphs. In other words it is easier to match small pre-labelled subgraphs to subgraphs in $G$, often resulting in an incorrect vertex labelling, than it is to match larger pre-labelled subgraphs. Hence the "backwards" in the Backward-Match-Voting algorithm. The above is illustrated by the example given in Figures 6.2 to 6.5. Figure 6.2 shows an unlabelled graph that features $L_E$ ={black, green, red}. Figures 6.3 to 6.5 show a set of labelled one-edge subgraphs and a labelled four-edge subgraph. The one-edge and four-edge pre-labelled subgraphs is shown in Figure 6.3 and 6.5 respectively. The set $\{V1, V2, \ldots, V9\}$ in Figure 6.4 indicates the identifiers for the vertices in Figure 6.2. Using the largest four-edge pre-labelled subgraph (Figure 6.5) the vertex labelling for the new graph will be: $L_{V1} = \{A\}$, $L_{V2} = \{C\}$, $L_{V4} = \{D\}$, $L_{V6} = \{D\}$, $L_{V8} = \{B\}$. While using the one-edge VULS (Figure 6.3) the vertex labelling will be: $L_{V1} = \{A, B, C, D\}$, $L_{V2} = \{A, C\}$, $L_{V4} = \{A, D\}$, $L_{V6} = \{A, D\}$, $L_{V8} = \{A, B\}$. This example clearly indicates that the vertex's labelling becomes more complicated when using (small) one-edge VULS than when using (large) four-edge VULS.

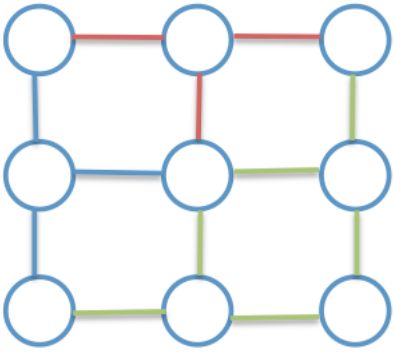


FIGURE 6.2: Input graph $G$.

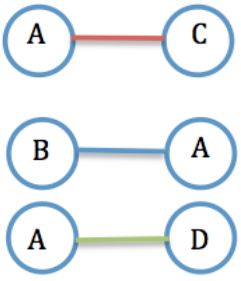



FIGURE 6.3: One-edge pre-labelled subgraphs.

## 6.3 A Working Example Using the Backward-Match-Voting Algorithm

This section presents a working example using a directed graph, to illustrate the operation of the BMV algorithm with max=4.

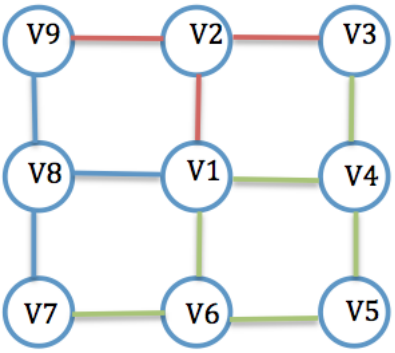



FIGURE 6.4: Graph without vertex labels.

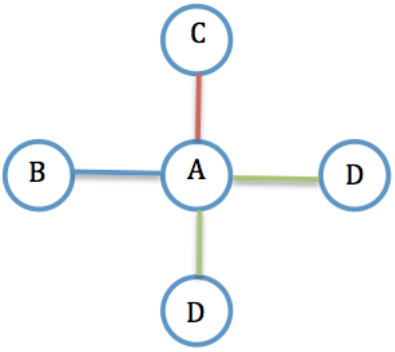



FIGURE 6.5: Four-edge pre-labelled subgraph.

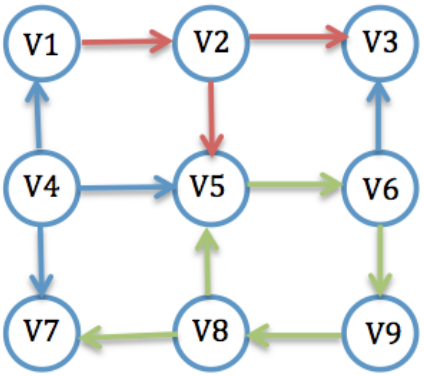



FIGURE 6.6: Input graph $G = \langle V, E, L_E \rangle$, with unlabelled vertices, $\{V1, V2, \ldots, V9\}$ are vertex identifiers.

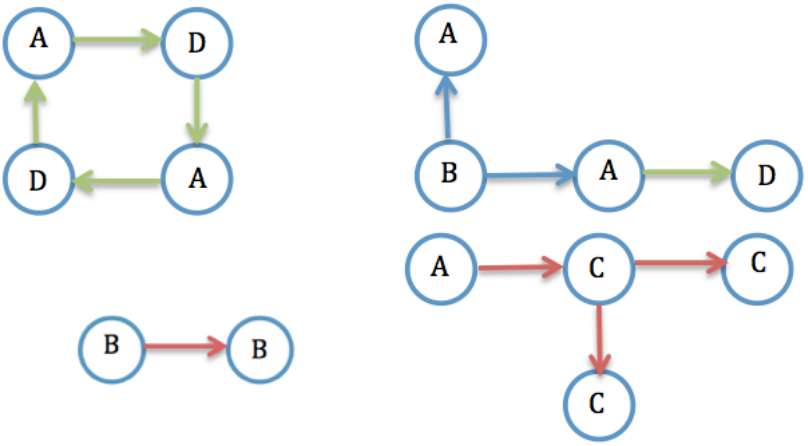



FIGURE 6.7: Four pre-labelled subgraphs.

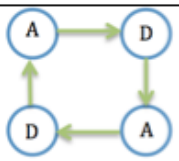
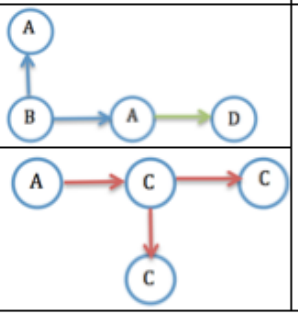
| Iteration number | K-edge pre-labelled subgraphs | Vertices in G covered by K-edge pre-labelled subgraphs | Vertex label allocation |
|---|---|---|---|
| K=4 |  |  Coverage=4/9=44.44% | $L_{V1}$={}<br>$L_{V2}$={}<br>$L_{V3}$={}<br>$L_{V4}$={}<br>$L_{V5}$={A,D,A,D}<br>$L_{V6}$={D,A,D,A}<br>$L_{V7}$={}<br>$L_{V8}$={D,A,D,A}<br>$L_{V9}$={A,D,A,D} |
| K=3 |  |  Coverage=9/9=100% | $L_{V1}$={A,A}<br>$L_{V2}$={C}<br>$L_{V3}$={C}<br>$L_{V4}$={B,B}<br>$L_{V5}$={ A,D,A,D,A,A,C}<br>$L_{V6}$={ D,A,D,A,D,D}<br>$L_{V7}$={A}<br>$L_{V8}$={ D,A,D,A }<br>$L_{V9}$={ A,D,A,D } |

FIGURE 6.8: Worked example of vertex classification using the pre-labelled subgraphs given in Figure 6.7 and the vertex-unlabelled graph given in Figure 6.6.

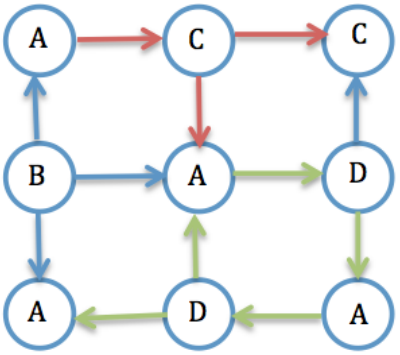



FIGURE 6.9: Output graph $G = \langle V, E, L_E, L_V \rangle$ with predicted vertex label.

The input graph $G$ for the example is presented in Figure 6.6. The input set of pre-labelled subgraphs is given in Figure 6.7. Figure 6.8 is organised in a tabular format with the rows representing $k-$edge pre-labelled subgraph iterations and the columns representing: (i) the relevant $k-$edge pre-labelled subgraphs, (ii) the vertices in $G$ covered by the $k-$edge pre-labelled subgraphs, and (iii) vertex label allocation for $G$. On the first iteration where K=4 (second row in Figure 6.8), the four-edge pre-labelled subgraph is matched to input graph $G$ and vertices V5, V6, V8 and V9 are labelled (covered): $L_{V5} = \{A, D, A, D\}$, $L_{V6} = \{D, A, D, A\}$, $L_{V8} = \{D, A, D, A\}$, and $L_{V9} = \{A, D, A, D\}$. Coverage is now 44.44%. On the second iteration (K=3), the two

three-edge pre-labelled subgraphs are matched to the input graph $G$, vertices V1, V2, V3, V4, V5, V6, V7 are labelled: $L_{V1} = \{A, A\}$, $L_{V2} = \{C\}$, $L_{V3} = \{C\}$, $L_{V4} = \{B, B\}$, $L_{V5} = \{A, D, A, D, A, A, C\}$, $L_{V6} = \{D, A, D, A, D, D\}$, and $L_{V7} = \{A\}$. Note that $L_{V8}$ and $L_{V9}$ were covered on the previous iteration so are not considered here. The coverage has now reached 100%, thus the process stops and the one edge pre-labelled subgraph $\langle B, red, B\rangle$ will not be considered. At the end of this stage, where a vertex is labelled with more than one label, the majority voting mechanism is applied. Note that given an "equal votes situation", a "first come first serve" strategy is used. For example, if $L_{V8} = \{D, A, D, A\}$, (equal votes for A and D), V8 will be assigned the label D because it appeared first. The final output graph $G$, with vertex labelings, is illustrated in Figure 6.9.

## 6.4 Summary

This chapter has described the theory and operation of the BMV prediction algorithm for vertex classification using pre-labelled subgraphs. The algorithm was presented in detail and its operation is illustrated using an example. The experimental analysis and evaluation of all the algorithms proposed in this and the previous chapter, using the data sets introduced in Chapter 3 from the application domains of sheet metal forming (more specifically AISF) and satellite image interpretation, is presented in the following two chapters respectively.

# Chapter 7

# Experimental Results Using The Sheet Metal Forming Application

## 7.1 Introduction

Chapter 5 presented a sequence of algorithms for mining four different categories of VULS: (i) complete, (ii) minimal, (iii) frequent and (iv) minimal frequent. The previous Chapter 6 presented the Backward-Match-Voting algorithm for applying the identified VULS in the context of vertex classification. This chapter presents an evaluation of these algorithms in the context of the sheet metal forming application introduced previously in Chapter 3 (evaluation using the satellite image interpretation application is considered in the following chapter). The principal objectives of the evaluation were as follows:

1. To compare the operation of the proposed VULS mining algorithms, using a range of $max$ values, in terms of coverage, number of identified VULS and runtime.
2. To investigate the effect of using different values for the grid size $d$.
3. To investigate the effect of using different values for $|L_E|$.
4. To compare the distinction between the usage of grid graphs and cross grid graphs, and the usage of directed and undirected graphs.
5. To investigate the effect of using different values for $|L_V|$.
6. To compare the effectiveness of the identified VULS, with respect to vertex classification, and with respect to more standard approaches (J48 and Naive Bayes).
7. To investigate whether there is a statistically significant difference between the results obtained.

Each of these objectives is considered in a separate section below (Sections 7.2 to 7.8). A summary and some conclusions are presented in Section 7.9. All the proposed VULS algorithms were implemented using the JAVA programming language. All the reported

experiments were conducted using a 2.7 GHz Intel Core i5 with 4 GB 1333 MHz DDR3 memory, running OS X 10.8.1 (12B19).

The strategy adopted for reporting the outcomes with respect to the first six of the above objectives, given the large number of possible parameter combinations, was to only vary the parameter(s) of interest with respect to each individual objective, while fixing the remaining parameters. For ease of understanding Table 7.1 gives an overview of this strategy. In the table the columns reference the first six of the above listed objectives while the rows reference the different parameters to be considered. An "F" in a table cell indicates that a fixed value was used for the indicated parameter with respect to the indicated evaluation objective. Where a range of values is given in a table cell this is the range of parameter values reported on with respect to the experiments directed at the indicated objective. Recall that the *max* parameter is used to limit the size of the VULS to be mined, while the remaining parameters are used to define the grid graphs to be used. It should be noted that $|L_V|$ and $d$ (grid size) will typically be application dependent and thus user specified. All experiments reported on in this chapter used $d = 28$ (cm) since as demonstrated in Section 7.3 this tends to produce the best results. Further experiments were conducted using $d = 23$ (cm) with similar outcomes to those reported in this chapter; for completeness, the results from these additional experiments are reported in Appendix C. Note from the Table that for objective 6 (comparison of VULS classification effectiveness) only fixed parameter settings were used.

Table 7.1: Evaluation Strategy Summary

| Parameter | Objective | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| *max* | $\{4, 5, 6\}$ | $F$ | $F$ | $F$ | $F$ | $F$ |
| $d$ | $F$ | $\{18, 23, 28\}$ | $F$ | $F$ | $F$ | $F$ |
| $\|L_E\|$ | $F$ | $F$ | $\{2, 4, 6, 8\}$ | $F$ | $F$ | $F$ |
| Degree | $F$ | $F$ | $F$ | $\{4, 8\}$ | $F$ | $F$ |
| Directed? | $F$ | $F$ | $F$ | $\{Y, N\}$ | $F$ | $F$ |
| $\|L_V\|$ | $F$ | $F$ | $F$ | $F$ | $\{2, 3\}$ | $F$ |

With respect to the last of the above objectives (object 7) the Friedman statistical test was applied to evaluate the performance of the classifiers to determine whether the results produced were truly significant or not with respect to the AUC measure. On completion of the Friedman test, where appropriate Nemenyi test was used to identify the "critical distances" between the techniques so as to identify where the differences actually occurred. Recall that both the Friedman and the Nemenyi test were described in Chapter 2.

## 7.2 Comparison of VULS Mining Algorithms Using a Range of $max$ Values (Objective 1)

This section provides a comparative evaluation of the four different VULS mining algorithms presented in this thesis with respect to a range of max parameter values. As indicated in Table 7.1 the remaining parameters were kept constant at $|L_V| = 2$, $|L_E| = 8$ and $Degree = 4$. All the graphs considered featured directed edges.

The results are presented in Tables 7.2 to 7.4 corresponding to $max$ parameter settings of 4, 5 and 6 respectively. In each table the VULS mining algorithms are listed along the top (CompVULS, Mimimum VULS, Frequent VULS and Minimal frequent VULS). The data sets (graphs) considered are represented by the rows. Note that, because the raw AISF data sets available were paired, the evaluation was conducted by training on one data set and testing on the other. For example in the case of the Gonzalo Steel AISF data, the classifier was trained using GS1 and tested using GS2. The different data sets are indicated using the following notation: GS (Gonzalo Steel), GT (Gonzalo Titanium), MS (Modified Steel) and MT (Modified Titanium) (see chapter 3). For each algorithm coverage, number of identified VULS and run time are presented. Each table also records the average coverage, number of VULS and run time for each algorithm and the associate SD. Note that Standard Deviation (SD) is a measure of how much variation exists from the average. A low standard deviation indicates that the data points tend to be very close to the mean. A high standard deviation indicates that the data points are spread out over a large range of values. The colour coding used in the tables is simply for ease of comparison.

TABLE 7.2: Comparison of VULS Mining Algorithms Using $max = 4$ (Objective 1).

| Graph | Comp.VULS | | | Min. VULS | | | Freq. VULS | | | Min. freq. VULS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time |
| GS | 100.00 | 122.00 | 0.36 | 100.00 | 9.00 | 0.17 | 100.00 | 121.00 | 0.22 | 100.00 | 6.00 | 0.17 |
| GT | 100.00 | 127.00 | 0.23 | 100.00 | 20.00 | 0.15 | 100.00 | 111.00 | 0.24 | 100.00 | 7.00 | 0.17 |
| MS | 100.00 | 86.00 | 0.32 | 100.00 | 20.00 | 0.20 | 97.22 | 169.00 | 0.31 | 88.89 | 24.00 | 0.28 |
| MT | 97.22 | 257.00 | 0.36 | 91.67 | 16.00 | 0.19 | 97.22 | 185.00 | 0.36 | 86.11 | 8.00 | 0.20 |
| Average | 99.31 | | 0.32 | 97.92 | | 0.18 | 98.61 | | 0.28 | 93.75 | | 0.21 |

TABLE 7.3: Comparison of VULS Mining Algorithms Using $max = 5$ (Objective 1).

| Graph | Comp.VULS | | | Min. VULS | | | Freq. VULS | | | Min. freq. VULS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time |
| GS | 100.00 | 122.00 | 0.19 | 100.00 | 9.00 | 0.17 | 100.00 | 121.00 | 0.21 | 100.00 | 6.00 | 0.16 |
| GT | 100.00 | 127.00 | 0.19 | 100.00 | 20.00 | 0.15 | 100.00 | 111.00 | 0.24 | 100.00 | 7.00 | 0.16 |
| MS | 100.00 | 86.00 | 0.16 | 100.00 | 20.00 | 0.18 | 97.22 | 471.00 | 0.60 | 94.44 | 35.00 | 0.38 |
| MT | 97.22 | 700.00 | 0.75 | 91.67 | 16.00 | 0.25 | 97.22 | 465.00 | 0.59 | 86.11 | 8.00 | 0.28 |
| Average | 99.31 | | 0.32 | 97.92 | | 0.19 | 98.61 | | 0.41 | 95.14 | | 0.25 |

TABLE 7.4: Comparison of VULS Mining Algorithms Using $max = 6$ (Objective 1).

| Graph | Comp.VULS | | | Min. VULS | | | Freq. VULS | | | Min. freq. VULS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time |
| GS | 100.00 | 122.00 | 0.21 | 100.00 | 9.00 | 0.20 | 100.00 | 121.00 | 0.23 | 100.00 | 6.00 | 0.21 |
| GT | 100.00 | 127.00 | 0.20 | 100.00 | 20.00 | 0.17 | 100.00 | 111.00 | 0.22 | 100.00 | 7.00 | 0.17 |
| MS | 100.00 | 86.00 | 0.18 | 100.00 | 20.00 | 0.23 | 97.22 | 1245.00 | 0.98 | 94.44 | 44.00 | 0.78 |
| MT | 97.22 | 1814.00 | 1.49 | 91.67 | 16.00 | 0.21 | 97.22 | 1162.00 | 0.85 | 86.11 | 8.00 | 0.35 |
| Average | 99.31 | | 0.52 | 97.92 | | 0.20 | 98.61 | | 0.57 | 95.14 | | 0.38 |

From Tables 7.2 to 7.4 the following can be noted with respect to coverage, number of VULS identified and run time:

- **Coverage.**

  1. When finding Complete VULS, minimal VULS, frequent VULS and minimal frequent VULS, 100% coverage can be obtained in some cases regardless of the *max* value used.
  2. When finding minimal VULS 100% coverage was obtained in three out of the four graphs considered. Where 100% coverage was not obtained this was always with respect to the MT graph. In the same manner, When finding frequent VULS and minimal frequent VULS respectively, 100% coverage was obtained in two out of the four graphs considered.
  3. On average, when finding minimal VULS, coverage was always worse than when finding frequent VULS.
  4. The worst recorded coverage values were obtained when finding minimal frequent VULS.
  5. The relationship between the different VULS algorithms in terms of coverage fits with the more general relationship between the four different categories of VULS illustrated previously in Figure 2.2 of chapter 4.
  6. As the *max* parameter increased from 4 to 6 we can anticipate that more VULS will be identified and hence coverage will also increase. This trend becomes more apparent when $d = 23$ (cm) is used as shown in Appendix C. In the case of $d = 28$ (cm) as used with respect to the experiments presented in this chapter, this trend is not so apparent because if the coverage when $max = 4$ is 100% this cannot be improved upon. In most graphs, 100% has already been reached when $max = 4$. In the case of the Complete VULS, minimal VULS or frequent VULS results, when $max = 5$ and $max = 6$, the recorded coverage values were the same; this was because no additional VULS which cover new vertices could be found by increasing $max = 5$ to $max = 6$.

- **Number of VULS.**

    1. As anticipated many more VULS are discovered when mining for Complete VULS than when mining for any of the other forms of VULS considered (minimal, frequent, minimal frequent). The total number of VULS that exist in a given graph will be greater than the number of minimal VULS that can be found in the same graph. Similarly the total number of VULS will be greater than the number of frequent VULS that can be found in the same graph. Of course the number of minimal frequent VULS that can be found will be less than the number of frequent VULS that can be found.
    2. Generally speaking, with respect to the average number of VULS discovered in each case, we can expect: (i) VULS $\geqslant$ minimal VULS $\geqslant$ minimal frequent VULS; and (ii) VULS $\geqslant$ frequent VULS $\geqslant$ minimal frequent VULS. However, from the tables we can see that the number of frequent VULS identified is sometimes greater than the number identified using the Complete VULS algorithm, this is because the Complete VULS algorithm stops mining further VULS when the coverage reaches 100%.
    3. Not withstanding point 2 above, the recorded results again confirm the relationship between the four different categories of VULS illustrated previously in Figure 2.2 of Chapter 4.
    4. With respect to the *max* parameter, as this was increased from 4 to 6, the number of identified VULS was expected to also increases. This is confirmed by the results recorded in the table (except where 100% coverage was reached using a lower value of *max* as noted in point 2 above).

- **Run time.**

    1. As the *max* parameter was increased from 4 to 6, the required run time for identifying VULS (with respect to all four categories) also increased (as was to be expected).
    2. The frequent VULS mining algorithm required more run time than the Complete VULS and Minimal VULS algorithms because of the additional candidate graph counting (isomorphism testing) that had to be conducted.

From the above, the number of minimal VULS (or frequent VULS) is low (in comparison with the total number of VULS), however coverage is still good. Furthermore, it is interesting to note that good coverage is still obtained using small numbers of minimal VULS or frequent VULS. For example, with respect to the GS directed graphs, the coverage when mining either Complete VULS, minimal VULS, frequent VULS or minimal frequent VULS is always 100% regardless of the *max* value used. However, when using Complete VULS mining we identify 122 VULS, while when using minimal VULS mining we identify only 9; similarly when using frequent VULS mining we identify

121 VULS while when using minimal frequent we identify only 6. For ease of consistent consideration and analysis of the effects of varying the remaining parameters $max = 4$ was used with respect to the following reported evaluations, because in most cases 100% coverage had already been reached when using $max = 4$ and when this was not the case coverage did not increase dramatically as $max$ increased from 4 to 6.

## 7.3 Effect of Grid Size $d$ on Classification Effectiveness (Objective 2)

The parameter considered with respect to the evaluation presented in this section is the grid size $d$. In this section we consider the effect on classification when $d = \{18, 23, 28\}$ respectively. For the experiments $|L_V| = 2$, $|L_E| = 8$ and $max = 4$ were again used. Only directed grid graphs were considered. The results are presented in Table 7.5.

Table 7.5: Classification Effectiveness with Respect to $d$ (Objective 2).

| $d$ | Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | |
|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| 18 | GS | 0.65 | 0.64 | 0.48 | 0.47 | 0.77 | 0.77 | 0.50 | 0.49 |
| | GT | 0.68 | 0.53 | 0.80 | 0.71 | 0.67 | 0.51 | 0.79 | 0.69 |
| | MS | 0.74 | 0.76 | 0.57 | 0.60 | 0.77 | 0.79 | 0.70 | 0.72 |
| | MT | 0.78 | 0.53 | 0.80 | 0.50 | 0.77 | 0.52 | 0.80 | 0.50 |
| Average | | 0.71 | 0.62 | 0.66 | 0.57 | 0.75 | 0.65 | 0.70 | 0.60 |
| SD | | 0.06 | 0.11 | 0.16 | 0.11 | 0.05 | 0.15 | 0.14 | 0.12 |
| 23 | GS | 0.75 | 0.77 | 0.59 | 0.63 | 0.75 | 0.77 | 0.59 | 0.63 |
| | GT | 0.72 | 0.53 | 0.77 | 0.62 | 0.72 | 0.53 | 0.77 | 0.62 |
| | MS | 0.81 | 0.83 | 0.58 | 0.63 | 0.78 | 0.81 | 0.56 | 0.61 |
| | MT | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 |
| Average | | 0.79 | 0.67 | 0.71 | 0.61 | 0.78 | 0.67 | 0.70 | 0.60 |
| SD | | 0.07 | 0.15 | 0.15 | 0.04 | 0.07 | 0.15 | 0.15 | 0.04 |
| 28 | GS | 0.89 | 0.88 | 0.58 | 0.56 | 0.89 | 0.88 | 0.61 | 0.59 |
| | GT | 0.69 | 0.77 | 0.53 | 0.65 | 0.69 | 0.77 | 0.44 | 0.58 |
| | MS | 0.94 | 0.94 | 0.72 | 0.71 | 0.97 | 0.97 | 0.94 | 0.94 |
| | MT | 0.86 | 0.67 | 0.89 | 0.68 | 0.86 | 0.67 | 0.89 | 0.60 |
| Average | | 0.85 | 0.82 | 0.68 | 0.65 | 0.85 | 0.82 | 0.72 | 0.68 |
| SD | | 0.11 | 0.12 | 0.16 | 0.06 | 0.12 | 0.13 | 0.24 | 0.18 |

From Table 7.5, it can be seen that classification effectiveness tends to increase as the grid size $d$ increases from 18 to 28. As noted above the value for $d$ is application domain dependent. In the context of sheet metal forming, if the grid size $d$ is small,

TABLE 7.6: Classification Effectiveness with Respect to $|L_E|$ (objective 3).

| $\|L_E\|$ | Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | | J48 | | Naive Bayes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| 2 | GS | 0.44 | 0.46 | 0.53 | 0.55 | 0.44 | 0.46 | 0.53 | 0.55 | 0.47 | 0.50 | 0.56 | 0.71 |
| | GT | 0.67 | 0.50 | 0.64 | 0.56 | 0.67 | 0.50 | 0.67 | 0.58 | 0.67 | 0.50 | 0.67 | 0.75 |
| | MS | 0.53 | 0.55 | 0.89 | 0.88 | 0.86 | 0.87 | 0.92 | 0.91 | 0.83 | 0.72 | 0.83 | 0.77 |
| | MT | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.80 |
| Average | | 0.63 | 0.50 | 0.73 | 0.62 | 0.71 | 0.58 | 0.75 | 0.64 | 0.71 | 0.56 | 0.73 | 0.76 |
| SD | | 0.18 | 0.04 | 0.17 | 0.17 | 0.20 | 0.19 | 0.18 | 0.19 | 0.18 | 0.11 | 0.14 | 0.04 |
| 4 | GS | 0.58 | 0.60 | 0.50 | 0.52 | 0.58 | 0.60 | 0.50 | 0.52 | 0.47 | 0.50 | 0.83 | 0.79 |
| | GT | 0.69 | 0.56 | 0.67 | 0.50 | 0.72 | 0.58 | 0.67 | 0.50 | 0.67 | 0.50 | 0.69 | 0.82 |
| | MS | 0.92 | 0.91 | 0.83 | 0.82 | 0.97 | 0.97 | 0.97 | 0.97 | 0.94 | 0.91 | 0.94 | 0.98 |
| | MT | 0.86 | 0.67 | 0.89 | 0.68 | 0.86 | 0.67 | 0.92 | 0.70 | 0.86 | 0.50 | 0.89 | 0.95 |
| Average | | 0.76 | 0.69 | 0.72 | 0.63 | 0.78 | 0.71 | 0.77 | 0.67 | 0.74 | 0.60 | 0.84 | 0.89 |
| SD | | 0.16 | 0.16 | 0.18 | 0.15 | 0.17 | 0.18 | 0.22 | 0.22 | 0.21 | 0.21 | 0.11 | 0.09 |
| 6 | GS | 0.75 | 0.76 | 0.42 | 0.44 | 0.75 | 0.76 | 0.42 | 0.44 | 0.47 | 0.50 | 0.75 | 0.83 |
| | GT | 0.69 | 0.54 | 0.69 | 0.54 | 0.69 | 0.54 | 0.69 | 0.54 | 0.67 | 0.50 | 0.67 | 0.79 |
| | MS | 0.94 | 0.94 | 0.89 | 0.88 | 0.94 | 0.94 | 0.94 | 0.94 | 0.94 | 0.91 | 0.89 | 0.98 |
| | MT | 0.86 | 0.67 | 0.89 | 0.68 | 0.86 | 0.67 | 0.89 | 0.60 | 0.86 | 0.50 | 0.86 | 0.88 |
| Average | | 0.81 | 0.73 | 0.72 | 0.64 | 0.81 | 0.73 | 0.74 | 0.63 | 0.74 | 0.60 | 0.79 | 0.87 |
| SD | | 0.11 | 0.17 | 0.22 | 0.19 | 0.11 | 0.17 | 0.24 | 0.22 | 0.21 | 0.21 | 0.10 | 0.08 |
| 8 | GS | 0.89 | 0.88 | 0.58 | 0.56 | 0.89 | 0.88 | 0.61 | 0.59 | 0.47 | 0.50 | 0.75 | 0.88 |
| | GT | 0.69 | 0.77 | 0.53 | 0.65 | 0.69 | 0.77 | 0.44 | 0.58 | 0.67 | 0.50 | 0.72 | 0.87 |
| | MS | 0.94 | 0.94 | 0.72 | 0.71 | 0.97 | 0.97 | 0.94 | 0.94 | 0.83 | 0.77 | 0.89 | 0.99 |
| | MT | 0.86 | 0.67 | 0.89 | 0.68 | 0.86 | 0.67 | 0.89 | 0.60 | 0.86 | 0.50 | 0.86 | 0.90 |
| Average | | 0.85 | 0.82 | 0.68 | 0.65 | 0.85 | 0.82 | 0.72 | 0.68 | 0.71 | 0.57 | 0.81 | 0.91 |
| SD | | 0.11 | 0.12 | 0.16 | 0.06 | 0.12 | 0.13 | 0.24 | 0.18 | 0.18 | 0.14 | 0.08 | 0.05 |

such as 18 ($mm$), the grid graphs tend to be flat which in turn means that the geometry can not be effectively captured, thus the performance tends to deteriorate. Moreover, if the grid size $d$ is too small, it might lead to overfitting.

## 7.4 Effect of $|L_E|$ on Classification Effectiveness (Objective 3)

Although the size of the edge label set $L_E$ will in many case be dictated by the nature of the application domain the effect that the size of $L_E$ has on classification effectiveness, in terms of accuracy and AUC, is considered in this section. The discussion is founded on a sequence of experiments conducted using a range of values for $|L_E|$ from 2 to 8 increasing in steps of 2. Note that the VULS classification was again conducted using the raw data set pairings whereby one raw data set could be used for training while the other could be used for testing. For the experiments $max = 4$ and $|L_V| = 2$ were used. All the graphs considered were grid graphs (Degree=4) and featured directed edges. The results are presented in the table 7.6. The table is organised in a similar manner to the previous results tables presented in this chapter.

From Table 7.6 it can be seen that average accuracy and AUC tends to increase as $|L_E|$ is increased from 2 to 8 regardless of the form of VULS mining adopted (Complete set of VULS, Minimal VULS, Frequent VULS, Minimal Frequent VULS mining) in most cases. The conjectured reason for this is that as number of edge label $|L_E|$ increased, the

edge labels become more diverse, more distinctive VULS can be identified, thus more accurate vertex label prediction results.

## 7.5 Comparison Between Usage of Grid Graphs and Cross Grid Graphs, and Directed and Undirected Graphs (Objective 4)

In Chapter 3 it was noted that VULS mining, as envisioned in this thesis, could be applied to a variety of different types of grid graph: (i) directed grid graphs, (ii) undirected grid graphs, (iii) directed cross-grid graphs, (iv) undirected cross-grid graphs. Recall that the distinction between a grid graph and a cross grid graph is that the first has a degree of 4, while the second has a degree of 8 (except in the part of the graph where the 3D surface edges and corners are represented). In this section the distinction between these different types of grid graph are considered in terms of classification effectiveness (using accuracy and AUC), again using the raw data pairings used previously. For the comparison $max = 4$, $|L_V| = 2$ and $|L_E| = 8$ was used. The later because earlier experiments, reported in Section 7.4 above, had demonstrated that this produced the best result. The results from the conducted experiments are presented in Table 7.7.

Table 7.7: Classification Effectiveness with Respect to Graph Types (Objective 4).

| Graph Category | Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | | J48 | | Naive Bayes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| directed grid | GS | 0.89 | 0.88 | 0.58 | 0.56 | 0.89 | 0.88 | 0.61 | 0.59 | 0.47 | 0.50 | 0.75 | 0.88 |
| | GT | 0.69 | 0.77 | 0.53 | 0.65 | 0.69 | 0.77 | 0.44 | 0.58 | 0.67 | 0.50 | 0.72 | 0.87 |
| | MS | 0.94 | 0.94 | 0.72 | 0.71 | 0.97 | 0.97 | 0.94 | 0.94 | 0.83 | 0.77 | 0.89 | 0.99 |
| | MT | 0.86 | 0.67 | 0.89 | 0.68 | 0.86 | 0.67 | 0.89 | 0.60 | 0.86 | 0.50 | 0.86 | 0.90 |
| Average | | 0.85 | 0.82 | 0.68 | 0.65 | 0.85 | 0.82 | 0.72 | 0.68 | 0.71 | 0.57 | 0.81 | 0.91 |
| SD | | 0.11 | 0.12 | 0.16 | 0.06 | 0.12 | 0.13 | 0.24 | 0.18 | 0.18 | 0.14 | 0.08 | 0.05 |
| undirected grid | GS | 0.56 | 0.57 | 0.47 | 0.50 | 0.53 | 0.55 | 0.47 | 0.50 | 0.47 | 0.50 | 0.75 | 0.88 |
| | GT | 0.69 | 0.60 | 0.67 | 0.50 | 0.75 | 0.63 | 0.67 | 0.5 | 0.67 | 0.50 | 0.72 | 0.87 |
| | MS | 0.89 | 0.89 | 0.81 | 0.8 | 0.94 | 0.94 | 0.56 | 0.53 | 0.83 | 0.77 | 0.89 | 0.99 |
| | MT | 0.92 | 0.7 | 0.92 | 0.7 | 0.92 | 0.7 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.90 |
| Average | | 0.77 | 0.69 | 0.72 | 0.63 | 0.79 | 0.71 | 0.64 | 0.51 | 0.71 | 0.57 | 0.81 | 0.91 |
| SD | | 0.17 | 0.14 | 0.19 | 0.15 | 0.19 | 0.17 | 0.17 | 0.02 | 0.18 | 0.14 | 0.08 | 0.05 |
| directed cross-grid | GS | 0.64 | 0.65 | 0.69 | 0.71 | 0.69 | 0.71 | 0.56 | 0.58 | 0.47 | 0.50 | 0.75 | 0.88 |
| | GT | 0.67 | 0.50 | 0.67 | 0.50 | 0.67 | 0.50 | 0.69 | 0.54 | 0.67 | 0.50 | 0.72 | 0.87 |
| | MS | 0.94 | 0.94 | 0.89 | 0.88 | 0.94 | 0.94 | 0.94 | 0.94 | 0.83 | 0.77 | 0.89 | 0.99 |
| | MT | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.90 |
| Average | | 0.78 | 0.65 | 0.78 | 0.65 | 0.79 | 0.66 | 0.76 | 0.64 | 0.71 | 0.57 | 0.81 | 0.91 |
| SD | | 0.15 | 0.21 | 0.11 | 0.18 | 0.13 | 0.21 | 0.17 | 0.20 | 0.18 | 0.14 | 0.08 | 0.05 |
| undirected cross-grid | GS | 0.47 | 0.49 | 0.58 | 0.60 | 0.61 | 0.63 | 0.39 | 0.41 | 0.47 | 0.50 | 0.75 | 0.88 |
| | GT | 0.67 | 0.50 | 0.67 | 0.50 | 0.67 | 0.50 | 0.67 | 0.50 | 0.67 | 0.50 | 0.72 | 0.87 |
| | MS | 0.89 | 0.89 | 0.94 | 0.94 | 0.94 | 0.94 | 0.89 | 0.88 | 0.83 | 0.77 | 0.89 | 0.99 |
| | MT | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.50 | 0.86 | 0.90 |
| Average | | 0.72 | 0.60 | 0.76 | 0.64 | 0.77 | 0.64 | 0.70 | 0.57 | 0.71 | 0.57 | 0.81 | 0.91 |
| SD | | 0.19 | 0.20 | 0.17 | 0.21 | 0.16 | 0.21 | 0.23 | 0.21 | 0.18 | 0.14 | 0.08 | 0.05 |

From Table 7.7 it can be seen that, regardless of the form of VULS used (Complete VULS, minimal VULS, frequent VULS or minimal frequent VULS) more effective results

are produced using directed graphs than undirected graphs, and grid graphs than cross-grid graphs. More specifically, in most cases, the VULS classifiers' performance (in terms of accuracy and AUC) shows the trends: (i) directed grid > undirected grid; (ii) directed cross-grid > undirected cross-grid; (iii) directed grid >directed cross-grid; and (iv) in some cases, undirected grid >undirected cross-grid. The results indicate that VULS classifiers perform better on directed graphs than undirected graphs, because when using directed graphs a greater number of VULS are identified than when using undirected graphs consequently the vertex classification is more specific and as a result a better overall classification results. However, VULS classifiers tended to perform better when dealing with grid graphs where vertices have a degree of 4 than cross-grid graph where vertexes have a degree of 8. This is probably because when using degree=8 a greater number of edges are included in the input graph which in turn results in more VULS; as a result the likelihood of more than one label being assigned to a vertex is increased, and consequently the allocation (despite the adopted voting mechanism) tends to be less effective.

## 7.6 Effect of $|L_V|$ on Classification Effectiveness (Objective 5)

The parameter considered with respect to the evaluation presented in this section is $|L_V|$. As noted above the value for $|L_V|$ is application domain dependent. However, for completeness, in this section we consider the effect on classification when $|L_V| = 2$ and $|L_V| = 3$ (the nature of the labelling in each case was discussed in Chapter 3). For the experiments $|L_E| = 8$ and $max = 4$ were used. Only the results obtained using directed grid graphs are presented here. The results are presented in Table 7.8.

Table 7.8: Classification Effectiveness with Respect to $|L_V|$ (Objective 5).

| $\|L_V\|$ | Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | | J48 | | Naive Bayes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| 2 | GS | 0.89 | 0.88 | 0.58 | 0.56 | 0.89 | 0.88 | 0.61 | 0.59 | 0.47 | 0.50 | 0.75 | 0.88 |
| | GT | 0.69 | 0.77 | 0.53 | 0.65 | 0.69 | 0.77 | 0.44 | 0.58 | 0.67 | 0.50 | 0.72 | 0.87 |
| | MS | 0.94 | 0.94 | 0.72 | 0.71 | 0.97 | 0.97 | 0.94 | 0.94 | 0.83 | 0.77 | 0.89 | 0.99 |
| | MT | 0.86 | 0.67 | 0.89 | 0.68 | 0.86 | 0.67 | 0.89 | 0.60 | 0.86 | 0.50 | 0.86 | 0.90 |
| Average | | 0.85 | 0.82 | 0.68 | 0.65 | 0.85 | 0.82 | 0.72 | 0.68 | 0.71 | 0.57 | 0.81 | 0.91 |
| SD | | 0.11 | 0.12 | 0.16 | 0.06 | 0.12 | 0.13 | 0.24 | 0.18 | 0.18 | 0.14 | 0.08 | 0.05 |
| 3 | GS | 0.31 | 0.40 | 0.25 | 0.35 | 0.31 | 0.40 | 0.25 | 0.35 | 0.36 | 0.56 | 0.50 | 0.75 |
| | GT | 0.72 | 0.51 | 0.61 | 0.33 | 0.72 | 0.51 | 0.61 | 0.33 | 0.61 | 0.50 | 0.64 | 0.70 |
| | MS | 0.78 | 0.67 | 0.78 | 0.67 | 0.78 | 0.67 | 0.78 | 0.67 | 0.64 | 0.75 | 0.72 | 0.84 |
| | MT | 0.72 | 0.62 | 0.75 | 0.61 | 0.72 | 0.62 | 0.75 | 0.61 | 0.72 | 0.50 | 0.67 | 0.82 |
| Average | | 0.63 | 0.55 | 0.60 | 0.49 | 0.63 | 0.55 | 0.60 | 0.49 | 0.58 | 0.58 | 0.63 | 0.78 |
| SD | | 0.22 | 0.12 | 0.24 | 0.18 | 0.22 | 0.12 | 0.24 | 0.18 | 0.16 | 0.12 | 0.09 | 0.06 |

From Table 7.8, as might be expected, better accuracy and AUC tend to be achieved when the number of vertex labels is small ($|L_V| = 2$). This is because as the number

of vertex labels is increased, for example to $|L_V| = 3$, the graph labelling becomes more diverse; as a result, the classification performance tends to deteriorate. This observation is true for classification in general, the more class labels that need to be considered the more challenging the classification.

## 7.7 Comparison of VULS Vertex Classification Effectiveness (Objective 6)

This section reports on the comparative evaluation conducted with respect to VULS vertex classification in general. This section includes a comparison with using more traditional forms of classification, namely J48 and Naive Bayes (as implemented in the Waikato Environment for Knowledge Analysis (WEKA) machine learning workbench[1]). For usage with J48 and Nave Bayes the data sets were processed as described in chapter 3. For the results presented in the Table 7.9, $max = 4$, $|L_E| = 8$ and $|L_V| = 2$ were used. All the graphs considered were grid graphs (degree= 4) and featured directed edges. The results are presented in Table 7.9. The table is laid out in a similar manner to those included in the previous sections.

Table 7.9: VULS Vertex Classification Comparison (Objective 6).

| Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min. freq. VULS | | J48 | | Naive Bayes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| GS | 0.89 | 0.88 | 0.58 | 0.56 | 0.89 | 0.88 | 0.61 | 0.59 | 0.47 | 0.50 | 0.75 | 0.88 |
| GT | 0.69 | 0.77 | 0.53 | 0.65 | 0.69 | 0.77 | 0.44 | 0.58 | 0.67 | 0.50 | 0.72 | 0.87 |
| MS | 0.94 | 0.94 | 0.72 | 0.71 | 0.97 | 0.97 | 0.94 | 0.94 | 0.83 | 0.77 | 0.89 | 0.99 |
| MT | 0.86 | 0.67 | 0.89 | 0.68 | 0.86 | 0.67 | 0.89 | 0.60 | 0.86 | 0.50 | 0.86 | 0.90 |
| Average | 0.85 | 0.82 | 0.68 | 0.65 | 0.85 | 0.82 | 0.72 | 0.68 | 0.71 | 0.57 | 0.81 | 0.91 |
| SD | 0.11 | 0.12 | 0.16 | 0.06 | 0.12 | 0.13 | 0.24 | 0.18 | 0.18 | 0.14 | 0.08 | 0.05 |

From Table 7.9 it can be seen that, in terms of the VULS algorithms considered, using Complete VULS and frequent VULS produced the most effective results in terms of average accuracy (0.85) and average AUC (0.82). Comparing with the Nave Bayes and J48 classifiers using Complete VULS and frequent VULS still produced the most effective results in terms of average accuracy (0.85), while Nave Bayes produced the most effective results in terms of AUC (0.91). J48 produced the worst vertex classification performance in terms of AUC.

Note that the above demonstrated VULS mining does not always work as well as Naive Bayes. However, the experiments reported here are in the context of grid graphs and not more general graphs formats to which Naive Bayes could not be applied.

[1] http://www.cs.waikato.ac.nz/ml/weka/

## 7.8 Statistical Comparison of the Proposed VULS Approaches (Objective 7)

This section reports on a statistical comparison of the vertex classification effectiveness of the proposed VULS mining by considering a range of classifiers defined in terms of: (i) the nature of the edges (directed or undirected), (ii) the vertex degree (4 or 8), (iii) a range of alternative values for $max$ $\{4, 5, 6\}$, and (iv) the VULS types (Complete VULS, MinVULS, freqVULS or minFreqVULS). Thus 48 ($2 \times 2 \times 3 \times 4$) forms of VULS classifiers were considered. The adopted naming convention for each form of VULS classifiers, used in this section, is:

$$< d, `-', `Degree', e, `-', `max', m, `-', V >$$

where $d$ is the nature of the edges (directed or undirected), $e$ is the vertex degree value (4 or 8), $m$ is the $max$ parameter (range of $\{4, 5, 6\}$) and $V$ is the VULS types (Complete VULS, MinVULS, freqVULS, minFreqVULS). For example, "Directed-Degree4-max5-minFreqVULS" indicates a minimal frequent VULS classifier, generated using $max = 5$ and a directed grid graph with $degree = 4$. In addition standard Naive Bayes and J48 classification was included in the experimentation, hence a total of 50 forms of classifier were considered.

Two separate sets of experiments were conducted, one using $|L_V| = 2$ and one using $|L_V| = 3$. For each of the 48 different forms of VULS classifier noted above, 28 graphs were generated by considering the four possible graph data pairings (GS, GT, MS, MT) and a range of seven values for $L_E$ ($\{2, 3, 4, 5, 6, 7, 8\}$), $7 \times 4 = 28$.

The average ranking of the classifiers, with respect to $|L_V| = 2$ and $|L_V| = 3$, is presented in Tables 7.10 and 7.11 respectively, where the "AR" column gives the average AUC ranked performance and the "AR+CD" column the average AUC ranked performance plus the calculated critical difference. Recall that the performance of two classifiers is significantly different if the corresponding Average Ranks (AR) differ by at least the Critical Difference (CD). In other words the values in the "AR+CD" column should be interpreted as indicating that the operation of any other classifier with a rank outside of the AR to AR+CD range is significantly different from the classifier in question.

With respect to Table 7.10, and in the context of the Freidman test, recall that $K$ is the number of classifiers, and $D$ is the number of data sets. Thus with respect to the statistical evaluation presented in this section $k = 50$ (48 VULS classifiers, plus J48 and Naive Bayes) and $D = 28$. Consequently the chi-squared ($\chi^2_F$) Friedman test value calculated using $(K - 1) \times (D - 1) = (50 - 1) \times (28 - 1) = 1323$ degrees of freedom, is as follows:

$$\begin{aligned}\chi^2_F &= \tfrac{12 \times 28}{50 \times (51)} \left[35335.78316326532 - \tfrac{50 \times (51)^2}{4}\right] \\ &= 0.13176470588235295 \times 2823.283163265318 \\ &= 372.0090756302537\end{aligned}$$

The p-value (threshold) was then $1.574E-10$ and the corresponding F-distribution, $F(49, 1323)$, was 10.044. The critical value for $F(49, 1323)$, with a critical difference level of $\alpha = 0.05$, is 1.364. Thus the p-value ($1.574E-10$) is significantly smaller than 0.005 and that the F-distribution (10.044) is larger than the corresponding F-distribution critical value of 1.364. Therefore we can reject the null hypothesis $H_0$ (that the observed performance differences among the classifiers is simply a matter of chance). Referring back to Table 7.10 the average rank results for the different VULS classifiers proposed in this thesis were significant ($p < 0.005$). A Nemenyi post-hoc test was therefore deemed to be applicable to determine which particular classifiers differ significantly from the others. The significance diagram is presented in Figure 7.1. The figure displays the AUC performance rankings for the classifiers, along with Nemenyi's Critical Difference (CD) tail. The CD value for the diagram is equal to 10.997 calculated as follows:

$$\begin{aligned} CD &= q_{\alpha,\infty,50}\sqrt{\tfrac{50\times 51}{12\times 28}} \\ &= 3.992 \times 2.755 \\ &= 10.997 \end{aligned}$$

The critical difference diagram given in 7.1 shows classifiers listed in ascending order of their ranked AUC performance along the y-axis; whilst the mean ranked AUC performance value, across all 28 datasets, is displayed on the x-axis. Two red vertical dashed lines have been inserted in the figure to clearly identify the end of the best performing classifier's tail and the start of the next significantly different approach. From the figure it can be seen that with respect to graphs where $|L_V| = 2$ and $d = 28$ (mm), the best performing classifier was Naive Bayes (a recorded AR value of 4.5). Indicated in blue in the figure: whilst the Directed-Degree4-max4-freqVULS, Directed-Degree4-max6-minFreqVULS and Degree4-max5-minFreqVULS classifiers produced statistically comparable performances to Naive Bayes. From the figure it can also be noted that all the classifiers highlighted in grey "before" the Directed-Degree8-max5-freqVULS classifier (highlighted in red) perform statistically better than J48.

TABLE 7.10: Average Rankings of classifiers where $|L_V| = 2$ and $d = 28$ (mm)

| Classifier | AR | AR+CD |
|---|---|---|
| NaiveBayes | 4.500 | 15.497 |
| Directed-Degree4-max5-minFreqVULS | 13.518 | 24.515 |
| Directed-Degree4-max6-minFreqVULS | 15.143 | 26.140 |
| Directed-Degree4-max4-freqVULS | 15.250 | 26.247 |
| Undirected-Degree4-max6-compVULS | 16.125 | 27.122 |
| Directed-Degree4-max5-compVULS | 16.464 | 27.461 |
| Undirected-Degree4-max5-compVULS | 16.518 | 27.515 |
| Directed-Degree4-max5-minVULS | 17.286 | 28.283 |
| Directed-Degree4-max6-freqVULS | 17.482 | 28.479 |
| Undirected-Degree4-max6-freqVULS | 17.696 | 28.693 |

| | | |
|---|---|---|
| Directed-Degree4-max5-freqVULS | 17.875 | 28.872 |
| Directed-Degree4-max4-compVULS | 18.304 | 29.301 |
| Directed-Degree4-max6-minVULS | 18.536 | 29.533 |
| Undirected-Degree4-max5-freqVULS | 18.768 | 29.765 |
| Directed-Degree4-max6-compVULS | 18.911 | 29.908 |
| Directed-Degree4-max4-minFreqVULS | 19.643 | 30.640 |
| Undirected-Degree4-max4-compVULS | 21.607 | 32.604 |
| Directed-Degree8-max4-freqVULS | 22.196 | 33.193 |
| Directed-Degree8-max5-compVULS | 23.036 | 34.033 |
| Directed-Degree8-max5-freqVULS | 23.625 | 34.622 |
| Directed-Degree8-max4-compVULS | 23.946 | 34.943 |
| Directed-Degree4-max4-minVULS | 24.054 | 35.051 |
| Undirected-Degree4-max6-minVULS | 24.804 | 35.801 |
| Undirected-Degree4-max4-freqVULS | 25.054 | 36.051 |
| Directed-Degree8-max6-freqVULS | 25.393 | 36.390 |
| Directed-Degree8-max6-compVULS | 26.393 | 37.390 |
| Directed-Degree8-max6-minFreqVULS | 28.679 | 39.676 |
| Undirected-Degree8-max5-minVULS | 28.768 | 39.765 |
| Undirected-Degree8-max6-freqVULS | 28.893 | 39.890 |
| Undirected-Degree4-max5-minVULS | 29.393 | 40.390 |
| Undirected-Degree8-max6-minVULS | 29.589 | 40.586 |
| Undirected-Degree8-max5-freqVULS | 29.821 | 40.818 |
| Undirected-Degree8-max4-minVULS | 30.036 | 41.033 |
| Directed-Degree8-max5-minFreqVULS | 30.571 | 41.568 |
| Directed-Degree8-max4-minFreqVULS | 31.286 | 42.283 |
| Directed-Degree8-max6-minVULS | 31.357 | 42.354 |
| Directed-Degree8-max5-minVULS | 31.464 | 42.461 |
| Undirected-Degree8-max6-compVULS | 31.911 | 42.908 |
| Undirected-Degree4-max6-minFreqVULS | 31.964 | 42.961 |
| Undirected-Degree8-max5-compVULS | 32.304 | 43.301 |
| Undirected-Degree8-max6-minFreqVULS | 32.393 | 43.390 |
| Undirected-Degree4-max4-minVULS | 32.607 | 43.604 |
| Undirected-Degree8-max4-freqVULS | 32.768 | 43.765 |
| Undirected-Degree8-max4-compVULS | 33.000 | 43.997 |
| Directed-Degree8-max4-minVULS | 34.589 | 45.586 |
| J48 | 34.893 | 45.890 |
| Undirected-Degree4-max5-minFreqVULS | 35.571 | 46.568 |
| Undirected-Degree8-max5-minFreqVULS | 35.911 | 46.908 |
| Undirected-Degree8-max4-minFreqVULS | 37.446 | 48.443 |
| Undirected-Degree4-max4-minFreqVULS | 37.661 | 48.658 |

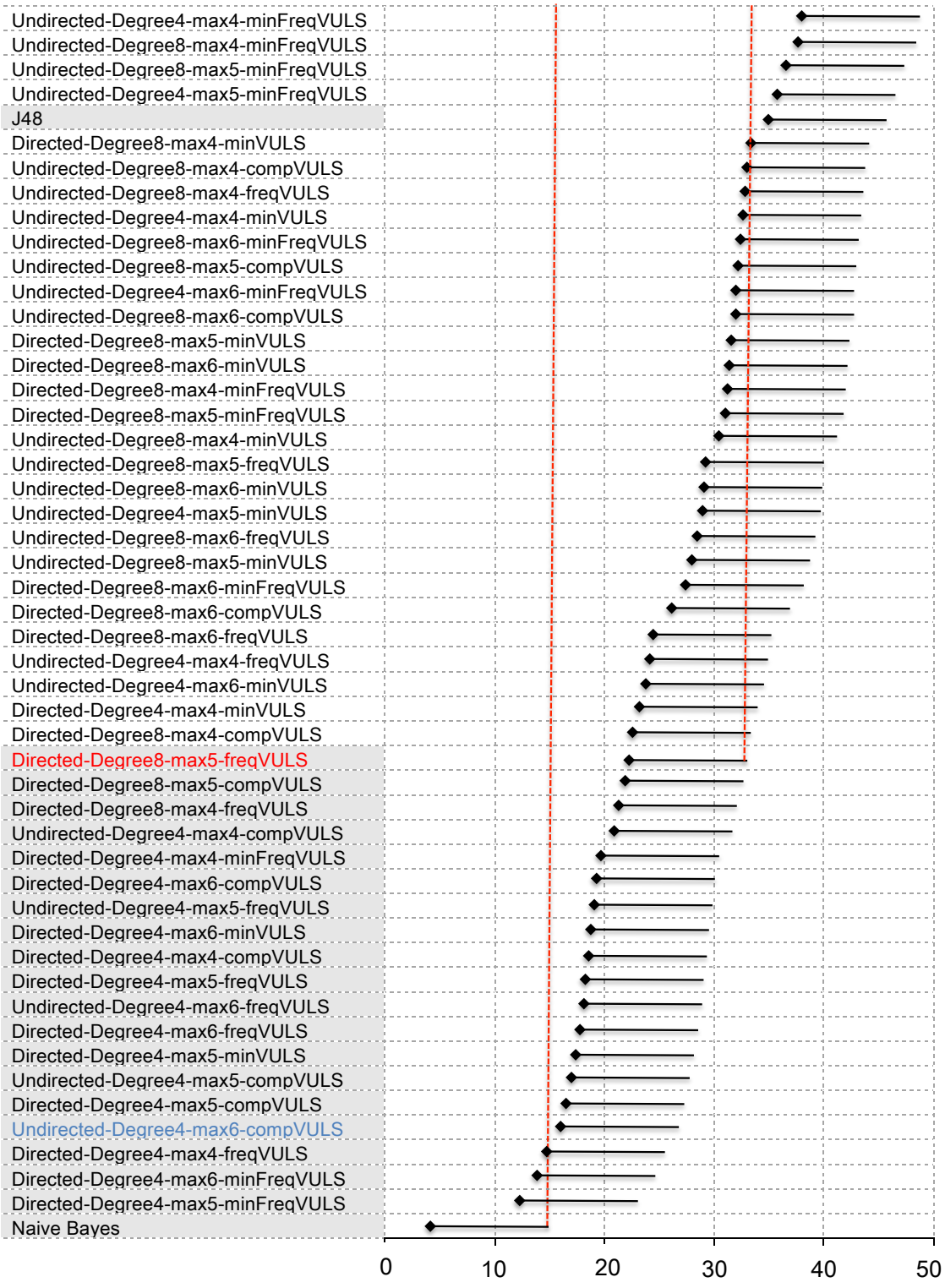



FIGURE 7.1: Critical difference diagram generated using Nemenyi's post hoc test with $\alpha = 0.05$ for graphs where $|L_V| = 2$ and $d = 28$ (mm).

Similarly, with respect to Table 7.11 ($|L_V| = 3$), the chi-squared ($\chi^2_F$) Friedman test value, with $K - 1 = 50 - 1 = 49$ degrees of freedom, was calculated as follows:

$$\begin{aligned}\chi^2_F &= \frac{12\times 28}{50\times(51)}\left[35368.0593112245 - \frac{50\times(51)^2}{4}\right] \\ &= 0.13176470588235295 \times 2855.559311224497 \\ &= 376.26193277311023\end{aligned}$$

The associated p-value (threshold) was then $1.344E - 10$. The F-distribution with $K - 1 = 49$ and$(K - 1) * (D - 1) = (50 - 1) * (28 - 1) = 1323$ degrees of freedom, $F(49, 1323)$, is 10.20. The critical value for $F(49, 1323)$, with a critical difference level of $\alpha = 0.05$, is 1.364. Thus, as in the case of $|L_V| = 3$, it can be noted that the p-value ($1.344E - 10$) is less than 0.005 and that the F-distribution (10.20) is larger than the corresponding F-distribution critical value (1.364). Therefore we can again reject the null hypothesis $H_0$ (that there is no statistical difference in operation between the classifiers). Note also that, referring back to Table 7.11, the average rank results were all significant ($p < 0.005$). Consequently a Nemenyi post-hoc test was deemed to be applicable to detect which particular classifiers differ from each other in a statistically significant manner.

The significance diagram is presented in Figure 7.2. As in the case where $|L_V| = 3$, the figure shows the AUC performance rankings for the classifiers, along with the Nemenyi's Critical Difference (CD) tail in each case. As before, the CD value for the diagram is equal to 10.997.

TABLE 7.11: Average Rankings of classifiers where $|L_V| = 3$ and $d = 28$ (mm)

| Classifier | AR | AR+CD |
|---|---|---|
| NaiveBayes | 1.768 | 12.765 |
| J48 | 12.179 | 23.176 |
| Directed-Degree4-max4-freqVULS | 16.518 | 27.515 |
| Directed-Degree4-max4-compVULS | 17.625 | 28.622 |
| Undirected-Degree4-max6-compVULS | 19.107 | 30.104 |
| Directed-Degree4-max5-compVULS | 19.661 | 30.658 |
| Directed-Degree4-max5-freqVULS | 19.875 | 30.872 |
| Directed-Degree4-max6-minVULS | 20.018 | 31.015 |
| Undirected-Degree4-max5-freqVULS | 20.089 | 31.086 |
| Directed-Degree8-max5-minFreqVULS | 20.482 | 31.479 |
| Directed-Degree8-max6-minFreqVULS | 20.625 | 31.622 |
| Directed-Degree4-max6-minFreqVULS | 20.946 | 31.943 |
| Undirected-Degree4-max6-freqVULS | 21.304 | 32.301 |
| Directed-Degree8-max5-freqVULS | 21.643 | 32.640 |
| Directed-Degree4-max5-minFreqVULS | 21.804 | 32.801 |
| Directed-Degree8-max5-compVULS | 21.875 | 32.872 |
| Directed-Degree4-max6-freqVULS | 21.893 | 32.890 |

| | | |
|---|---|---|
| Directed-Degree4-max5-minVULS | 22.107 | 33.104 |
| Directed-Degree4-max6-compVULS | 22.161 | 33.158 |
| Directed-Degree8-max6-compVULS | 22.393 | 33.390 |
| Directed-Degree8-max6-freqVULS | 22.519 | 33.515 |
| Directed-Degree8-max4-compVULS | 22.714 | 33.711 |
| Directed-Degree4-max4-minFreqVULS | 22.929 | 33.926 |
| Directed-Degree4-max4-minVULS | 23.143 | 34.140 |
| Directed-Degree8-max4-freqVULS | 23.179 | 34.176 |
| Undirected-Degree4-max5-compVULS | 23.268 | 34.265 |
| Directed-Degree8-max5-minVULS | 23.554 | 34.551 |
| Directed-Degree8-max6-minVULS | 23.590 | 34.586 |
| Undirected-Degree8-max5-compVULS | 24.768 | 35.765 |
| Directed-Degree8-max4-minVULS | 24.946 | 35.943 |
| Undirected-Degree8-max4-compVULS | 25.161 | 36.158 |
| Undirected-Degree4-max4-compVULS | 25.679 | 36.676 |
| Undirected-Degree8-max6-compVULS | 25.750 | 36.747 |
| Undirected-Degree4-max4-freqVULS | 27.179 | 38.176 |
| Directed-Degree8-max4-minFreqVULS | 27.304 | 38.301 |
| Undirected-Degree8-max5-freqVULS | 28.375 | 39.372 |
| Undirected-Degree8-max6-freqVULS | 29.268 | 40.265 |
| Undirected-Degree8-max4-freqVULS | 32.339 | 43.336 |
| Undirected-Degree8-max5-minVULS | 32.768 | 43.765 |
| Undirected-Degree8-max4-minVULS | 32.929 | 43.926 |
| Undirected-Degree8-max6-minVULS | 33.857 | 44.854 |
| Undirected-Degree4-max5-minVULS | 34.089 | 45.086 |
| Undirected-Degree4-max6-minVULS | 34.982 | 45.979 |
| Undirected-Degree4-max4-minVULS | 35.339 | 46.336 |
| Undirected-Degree8-max6-minFreqVULS | 36.625 | 47.622 |
| Undirected-Degree4-max6-minFreqVULS | 38.125 | 49.122 |
| Undirected-Degree4-max5-minFreqVULS | 38.643 | 49.640 |
| Undirected-Degree8-max5-minFreqVULS | 38.911 | 49.908 |
| Undirected-Degree8-max4-minFreqVULS | 39.446 | 50.443 |
| Undirected-Degree4-max4-minFreqVULS | 39.554 | 50.551 |

From Figure 7.2 it can be seen that with respect to graphs where $|L_V| = 3$ (and $d = 28$ (mm)) the best performing classifier was again Naive Bayes (a recorded AR value of 1.768). In this case performing significantly better than the proposed VULS classifiers, whilst J48 achieved a comparable performance to Naive Bayes. The VULS classifiers highlighted in grey, before the Directed-Degree8-max4-freqVULS classifier (highlighted in red) achieve statistically comparable performances to J48.

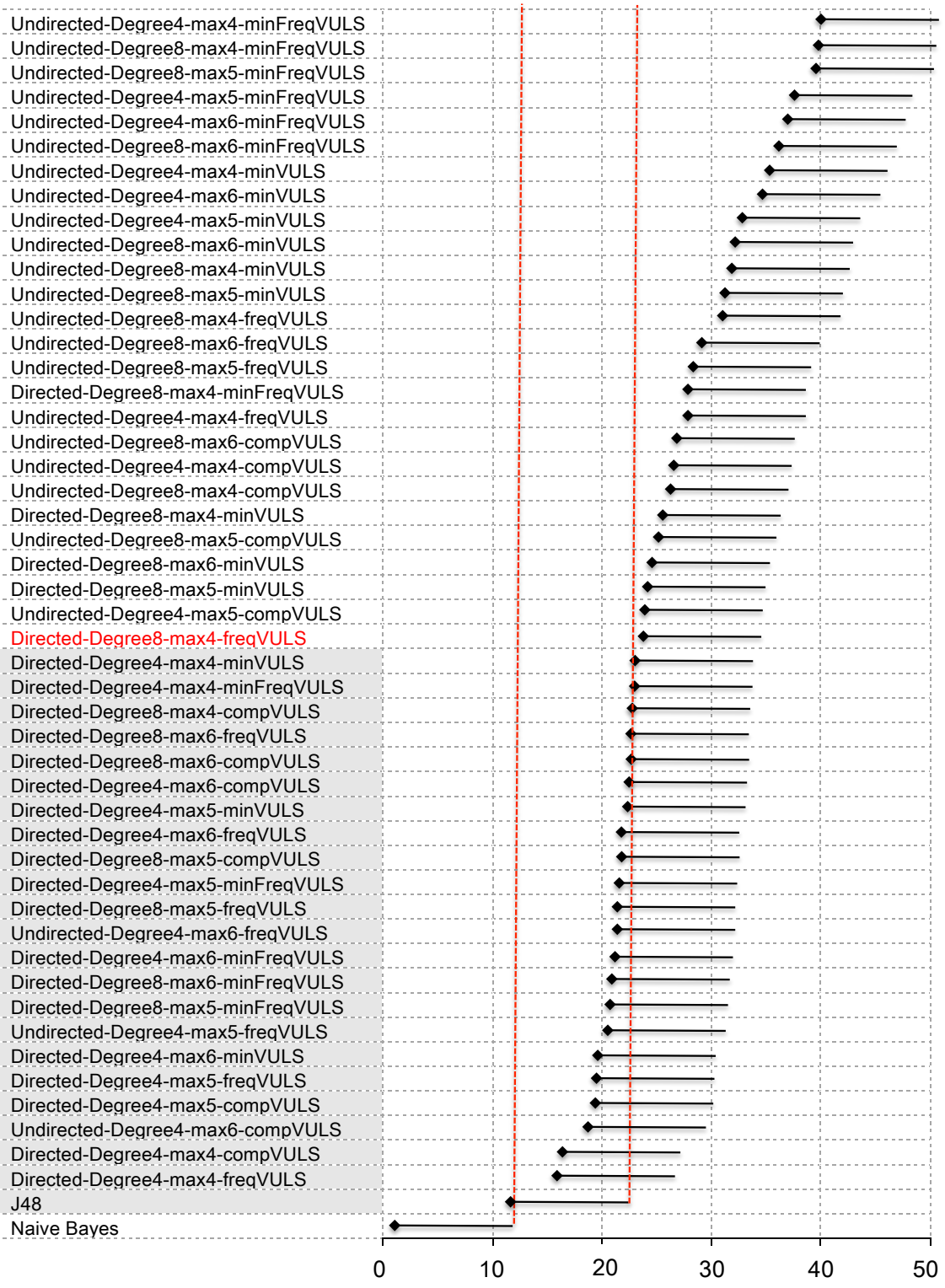



FIGURE 7.2: Critical difference diagram generated using Nemenyi's post hoc test with $\alpha = 0.05$ for graphs where $|L_V| = 3$ and $d = 28$ (mm).

## 7.9 Summary

This chapter has presented an evaluation of the proposed VULS mining and vertex classification processes in the context of the AISF grid graph data sets. From the reported evaluation the following overall observations can be made:

1. With respect to the VULS mining the best coverage was produced using the Complete VULS mining algorithm.
2. The most effective VULS mining algorithms, in the context of vertex classification, were the Complete VULS and Frequent VULS mining algorithms in terms of both accuracy and AUC.
3. Frequent VULS mining is more efficient than Complete VULS mining since fewer VULS were generated when considering only frequent VULS.
4. There is no significant difference between the VULS classifiers with respect to the $max$ parameter settings (at least when $max = \{4, 5, 6\}$). However, using $max = 4$ is more efficient in the context of VULS mining (and in some cases produces a better classification result).
5. The set of VULS produced using directed grid graphs resulted in the most effective vertex classification (in terms of AUC).
6. With respect to the comparison with more traditional classifiers, Naive Bayes produced the best performance. However VULS vertex classification tended to outperform J48 (in terms of AUC) when using larger values of $|L_E|$ and $|L_V| = 2$.
7. When considering different values for $|L_E|$, the higher the $|L_E|$ (for example when $|L_E| = 8$) the more significant the VULS that can be identified, thus a better vertex classification performance (in terms of AUC) can be achieved.
8. VULS classifiers tend to perform better when dealing with "grid" graphs (degree= 4) than "cross-grid" graphs (degree= 8).
9. Relatively speaking, VULS classifiers tend to perform better when dealing with directed graphs than undirected graphs.
10. With respect to the size of $|L_V|$, as the number of vertex labels increases from 2 to 3, the graph labelling becomes more diverse; as a result, the classification performance tends to deteriorate.
11. With respect to the grid size $d$, as $d$ increases from 18 to 28, the classification performance tends to increase as well.

This completes the evaluation of the proposed VULS concept with respect to the sheet metal forming application used as the primary application focus for the work

described. The following chapter reports on the evaluation of the VULS concept in the context of the satellite image interpretation application, the secondary application focus for the work described in this thesis.

# Chapter 8

# Experimental Results Using The Satellite Image Interpretation Application

## 8.1 Introduction

In this chapter, an alternative application for VULS vertex classification is investigated. More specifically the labelling of objects in satellite images using the graph data sets presented previously in Chapter 3. The motivation for this second group of experiments was to investigate an alternative application for the concept of VULS mining. More specifically to analyse the operation of the proposed VULS mining in a different context, but with the same set of objectives as the AISF experiments presented in Chapter 7. For completeness these objectives were as follows:

1. To compare the operation of the proposed VULS mining algorithms, using a range of $max$ values, in terms of coverage, number of identified VULS and runtime.
2. To investigate the effect of using different values for the grid size $d$.
3. To investigate the effect of using different values for $|L_E|$.
4. To compare the distinction between the usage of grid graphs and cross grid graphs.
5. To investigate the effect of using different values for $|L_V|$.
6. To compare the effectiveness of the identified VULS, with respect to vertex classification, and with respect to more standard approaches (J48 and Naive Bayes).
7. To investigate whether there is a statistically significant difference between the results obtained.

As before, each of these objectives is considered in a separate section (Sections 8.2 to 8.8). A summary and some conclusions are presented in Section 8.9. Note that only

directed graphs are considered in this chapter. This is because the AISF experimental results presented in the previous chapter, in the context of sheet metal forming, had demonstrated that usage of directed graphs for vertex classification produced better results than when using undirected graphs. Some preliminary experiments, not reported here, with respect to the satellite graph data, also demonstrated this to be the case. Thus the discussions presented in this chapter are all directed at the usage of directed graphs.

The strategy adopted for reporting the outcomes with respect to the first six of the above objectives, given the large number of possible parameter combinations, is the same as in the previous chapter. Thus for each objective directed at analysing the effect of a particular parameter (Objectives 1 to 5 above) a range of values were considered for the parameter of interest while a static value was used for the remaining parameters as shown in Table 8.1. For objective 6, only fixed parameters were used. Unlike in the case of the AISF experiments presented in Chapter 7, where pairings of graphs were used for training and testing, in the case of the experiments presented in this chapter Ten Cross Validation (TCV) (as described in subsection 2.6.1 of chapter 2) was adopted. Thus results are reported for each 10 fold together with the overall average result.

Table 8.1: Evaluation Strategy Summary

| Parameter | Objective | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| $max$ | $\{4, 5, 6\}$ | $F$ | $F$ | $F$ | $F$ | $F$ |
| $d$ | $F$ | $\{8, 16, 32\}$ | $F$ | $F$ | $F$ | $F$ |
| $\|L_E\|$ | $F$ | $F$ | $\{2, 4, 6, 8\}$ | $F$ | $F$ | $F$ |
| degree | $F$ | $F$ | $F$ | $\{4, 8\}$ | $F$ | $F$ |
| $\|L_V\|$ | $F$ | $F$ | $F$ | $F$ | $\{2, 3\}$ | $F$ |
| Directed? | $F$ | $F$ | $F$ | $F$ | $F$ | $F$ |

With respect to the last objective (objective 7), investigation of the statistical significance of the outcomes, the Friedman statistical test was again applied to determine whether the results produced were statistically significant or not. On completion of the Friedman test, the Nemenyi test was then again used to identify the "critical distances" between the techniques so as to identify where differences actually occurred.

The naming convention used for the satellite image graphs was as follows: ⟨ 'SId',d, 'E',e, 'V',v ⟩; where: (i) $d$ is the grid size (measured in pixels in this case as opposed to mm in the case of the AISF graphs), (ii) $e$ is the number of edge labels ($|L_E|$) and (iii) $v$ is the number of vertex labels ($|L_V|$). For example, "SId8E2V2" indicates a graph derived from a satellite image with grid size $d$ of 8 pixels using $|L_E| = 2$ and $|L_V| = 2$.

Before considering the experimental results obtained it should also be recalled that both the vertex and edge labelling distributions for the satellite image graphs (as described in section 3.3.2 of Chapter 3) was extremely imbalanced. The results presented

in this chapter should thus be viewed in the context of the unbalanced nature of the data sets used.

## 8.2 Comparison of VULS Mining Algorithms Using a Range of $max$ values (Objective 1)

This section provides a comparative evaluation of the four different VULS mining algorithms presented in this thesis in the context of the satellite image interpretation application and with respect to a range of max parameter values. The remaining parameters were kept constant (as indicated in Table 8.1: $|L_V| = 2$, $|L_E| = 4$, $d = 32$ and $degree = 4$ (grid graphs)). As noted in the introduction to this chapter all the graphs considered featured directed edges. $|L_V| = 2$ was chosen because this had been shown to works well in the case of the AISF experiments presented in chapter 7. $|L_E| = 4$ was chosen because this was a good median value.

The comparative evaluation with respect to objective 1 was undertaken in terms of: (i) coverage, (ii) number of identified VULS and (iii) run time of ten folds. As noted above, because we used TCV to evaluate the VULS classifiers, the result for each fold is presented with respect to coverage and number of identified VULS. The recorded run time is the total time take for all ten folds together. The results are presented in Tables 8.2 to 8.4 corresponding to $max$ parameter settings of 4, 5 and 6 respectively. Coverage is computed in terms of the training set as in the case of AISF data. The colour coding used in the tables with respect to coverage is for ease of comparison.

TABLE 8.2: Comparison of VULS Mining Algorithms Using $max = 4$ (Objective 1).

| Fold | Comp.VULS | | | Min. VULS | | | Freq. VULS | | | Min. freq. VULS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time |
| 1 | 100.00 | 76 | | 98.44 | 13 | | 100.00 | 86 | | 98.44 | 6 | |
| 2 | 98.44 | 196 | | 98.44 | 21 | | 98.44 | 61 | | 90.63 | 6 | |
| 3 | 100.00 | 368 | | 9.38 | 1 | | 100.00 | 26 | | 9.38 | 1 | |
| 4 | 98.44 | 296 | | 100.00 | 11 | | 98.44 | 145 | | 98.44 | 6 | |
| 5 | 100.00 | 94 | | 98.44 | 30 | | 100.00 | 28 | | 96.88 | 25 | |
| 6 | 100.00 | 60 | 2.29 | 82.81 | 9 | 1.77 | 100.00 | 19 | 2.72 | 81.25 | 4 | 1.51 |
| 7 | 100.00 | 68 | | 98.44 | 9 | | 100.00 | 23 | | 96.88 | 4 | |
| 8 | 100.00 | 274 | | 100.00 | 22 | | 100.00 | 14 | | 100.00 | 5 | |
| 9 | 100.00 | 235 | | 98.44 | 56 | | 23.44 | 71 | | 98.44 | 50 | |
| 10 | 100.00 | 299 | | 100.00 | 13 | | 100.00 | 28 | | 100.00 | 7 | |
| Average | 99.69 | 196.60 | | 88.44 | 18.50 | | 92.03 | 50.10 | | 87.03 | 11.40 | |
| SD | 0.66 | 114.25 | | 28.25 | 15.49 | | 24.11 | 41.38 | | 27.88 | 15.06 | |

TABLE 8.3: Comparison of VULS Mining Algorithms Using $max = 5$ (Objective 1).

| Fold | Comp.VULS | | | Min. VULS | | | Freq. VULS | | | Min. freq. VULS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time |
| 1 | 100.00 | 76 | | 98.44 | 13 | | 100.00 | 86 | | 98.44 | 13 | |
| 2 | 98.44 | 861 | | 98.44 | 21 | | 98.44 | 377 | | 95.31 | 10 | |
| 3 | 100.00 | 368 | | 9.38 | 1 | | 100.00 | 26 | | 9.38 | 1 | |
| 4 | 98.44 | 917 | | 100.00 | 11 | | 98.44 | 145 | | 98.44 | 21 | |
| 5 | 100.00 | 94 | | 100.00 | 226 | | 100.00 | 28 | | 100.00 | 221 | |
| 6 | 100.00 | 60 | 2.54 | 98.44 | 88 | 1.85 | 100.00 | 19 | 4.09 | 96.88 | 32 | 3.19 |
| 7 | 100.00 | 68 | | 98.44 | 9 | | 100.00 | 23 | | 96.88 | 34 | |
| 8 | 100.00 | 274 | | 100.00 | 22 | | 100.00 | 14 | | 100.00 | 5 | |
| 9 | 100.00 | 235 | | 98.44 | 81 | | 40.63 | 197 | | 98.44 | 57 | |
| 10 | 100.00 | 299 | | 100.00 | 13 | | 100.00 | 28 | | 100.00 | 7 | |
| Average | 99.69 | 325.20 | | 90.16 | 48.50 | | 93.75 | 94.30 | | 89.38 | 40.10 | |
| SD | 0.66 | 316.69 | | 28.39 | 69.38 | | 18.68 | 117.27 | | 28.15 | 65.81 | |

TABLE 8.4: Comparison of VULS Mining Algorithms Using $max = 6$ (Objective 1).

| Fold | Comp.VULS | | | Min. VULS | | | Freq. VULS | | | Min. freq. VULS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time |
| 1 | 100.00 | 76 | | 98.44 | 13 | | 100.00 | 86 | | 98.44 | 13 | |
| 2 | 98.44 | 3257 | | 98.44 | 21 | | 98.44 | 1531 | | 95.31 | 10 | |
| 3 | 100.00 | 368 | | 9.38 | 1 | | 100.00 | 26 | | 9.38 | 1 | |
| 4 | 98.44 | 2100 | | 100.00 | 11 | | 98.44 | 145 | | 98.44 | 22 | |
| 5 | 100.00 | 94 | | 100.00 | 226 | | 100.00 | 28 | | 100.00 | 221 | |
| 6 | 100.00 | 60 | 7.69 | 98.44 | 88 | 2.30 | 100.00 | 19 | 18.13 | 96.88 | 32 | 3.96 |
| 7 | 100.00 | 68 | | 98.44 | 9 | | 100.00 | 23 | | 96.88 | 34 | |
| 8 | 100.00 | 274 | | 100.00 | 22 | | 100.00 | 14 | | 100.00 | 5 | |
| 9 | 100.00 | 235 | | 98.44 | 145 | | 60.94 | 596 | | 98.44 | 121 | |
| 10 | 100.00 | 299 | | 100.00 | 13 | | 100.00 | 28 | | 100.00 | 7 | |
| Average | 99.69 | 683.10 | | 90.16 | 54.90 | | 95.78 | 249.60 | | 89.38 | 46.60 | |
| SD | 0.66 | 1091.88 | | 28.39 | 75.40 | | 12.26 | 483.96 | | 28.15 | 70.53 | |

In the context of Tables 8.2 to 8.4 the following can be observed with respect to coverage, number of VULS identified and run time:

- **Coverage.**

  1. Excellent coverage was obtained, greater than 99.69% with respect to all VULS categories with *max* of 4.
  2. Although there is little difference between the coverage results, best overall coverage was obtained when using Complete VULS (this was also the case with respect to the AISF experiments reported previously).
  3. The coverage results confirm the relationship between the four different categories of VULS illustrated previously in Figure 2.2 of Chapter 4.

- **Number of VULS.**

  1. As in the case of the AISF experiments, many more VULS were discovered when using the Complete VULS algorithm than when using any of the other proposed algorithms (Minimal, Frequent, or Minimal Frequent). Recall that

the total number of VULS that exist in a given graph will typically be greater than the number of minimal VULS that can be found in the same graph. Similarly the total number of VULS will typically be greater than the number frequent VULS that can be found in the same graph. Of course the number of minimal frequent VULS that can be found will typically be less than (or possibly equal to) the number of frequent VULS.

2. Again as in the case of the AISF experiments, and as might be expected, as $max$ is increased from 4 to 6 more VULS will be identified, thus coverage also increases. Although, if the coverage when $max = 4$ is 100%, this can not be improved upon as max is increased further from 4 to 6. In most graphs, 100% coverage had already been reached when $max = 4$ with respect to Complete VULS mining.

- **Run time.**

1. As the $max$ parameter was increased from 4 to 6 the required time for identifying VULS also increased (as expected), thus confirming the runtime result observed when using the AISF graph data.

Overall, with respect to the AISF results presented in Section 7.2 in Chapter 7 the results presented above confirm the results obtained previously.

## 8.3 Effect of grid size $d$ on Classification Effectiveness (Objective 2)

The parameter considered with respect to the evaluation presented in this section is grid size $d$. In this section we consider the effect on classification when $d = \{8, 16, 32\}$ respectively. For the experiments $|L_V| = 2$ and $|L_E| = 4$ were again used; $max = 4$ was used because this tended to produce effective results as demonstrated in the preceding section. The results are presented in Table 8.5.

From Table 8.5 it can be seen that, as opposed to the sheet metal forming application considered in the previous chapter, classification effectiveness tends to deteriorate as the grid size $d$ increases from 8 to 32. The conjectured reasons for this are as follows:

- When images are translated into graphs with larger grid sizes ($d = 16$ or $d = 32$ pixels) different ground types are likely to be incorporated into the individual grid squares. Consequently the average grayscale intensity value of the grid squares tends to become confused which will in turn effect the edge labelling. Hence the identified VULS become less effective at discriminating between classes. When using smaller grid sizes, such as $d = 8$ pixels, each grid square is much more likely to comprise only one ground type.

TABLE 8.5: Classification Effectiveness with Respect to $d$ (Objective 2).

| $d$ | Fold | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | |
|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| 8 | 1 | 100.00 | 1.00 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 2 | 99.41 | 0.77 | 99.02 | 0.62 | 98.73 | 0.50 | 98.73 | 0.50 |
| | 3 | 99.61 | 0.67 | 99.61 | 0.67 | 99.41 | 0.50 | 99.41 | 0.50 |
| | 4 | 99.41 | 0.79 | 99.22 | 0.67 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 5 | 99.90 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 6 | 99.22 | 0.67 | 98.93 | 0.54 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 7 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 8 | 99.80 | 0.75 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 9 | 99.80 | 0.75 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 10 | 99.71 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| Average | | 99.67 | 0.77 | 99.56 | 0.66 | 99.40 | 0.50 | 99.40 | 0.50 |
| SD | | 0.25 | 0.09 | 0.37 | 0.09 | 0.44 | 0 | 0.44 | 0 |
| 16 | 1 | 99.61 | 0.50 | 99.61 | 0.50 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 2 | 97.27 | 0.50 | 97.27 | 0.50 | 97.27 | 0.50 | 97.27 | 0.50 |
| | 3 | 98.83 | 0.50 | 98.83 | 0.50 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 4 | 98.05 | 0.58 | 98.05 | 0.58 | 97.66 | 0.50 | 97.66 | 0.50 |
| | 5 | 99.61 | 0.50 | 100.00 | 1.00 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 6 | 96.48 | 0.50 | 96.48 | 0.50 | 96.48 | 0.50 | 96.48 | 0.50 |
| | 7 | 99.22 | 0.50 | 99.22 | 0.50 | 99.22 | 0.50 | 99.22 | 0.50 |
| | 8 | 99.22 | 0.50 | 99.22 | 0.50 | 99.22 | 0.50 | 99.22 | 0.50 |
| | 9 | 99.61 | 0.50 | 99.61 | 0.50 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 10 | 99.22 | 0.50 | 99.22 | 0.50 | 99.61 | 0.50 | 99.61 | 0.50 |
| Average | | 98.71 | 0.51 | 98.75 | 0.56 | 98.71 | 0.50 | 98.71 | 0.50 |
| SD | | 1.09 | 0.03 | 1.13 | 0.16 | 1.15 | 0 | 1.15 | 0 |
| 32 | 1 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 |
| | 2 | 92.19 | 0.50 | 92.19 | 0.50 | 92.19 | 0.50 | 92.19 | 0.50 |
| | 3 | 95.31 | 0.50 | 95.31 | 0.50 | 95.31 | 0.50 | 95.31 | 0.50 |
| | 4 | 92.19 | 0.50 | 92.19 | 0.50 | 92.19 | 0.50 | 92.19 | 0.50 |
| | 5 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 |
| | 6 | 95.31 | 0.50 | 95.31 | 0.50 | 95.31 | 0.50 | 95.31 | 0.50 |
| | 7 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 |
| | 8 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 |
| | 9 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 | 98.44 | 0.50 |
| | 10 | 96.88 | 0.50 | 96.88 | 0.50 | 96.88 | 0.50 | 96.88 | 0.50 |
| Average | | 96.41 | 0.50 | 96.41 | 0.50 | 96.41 | 0.50 | 96.41 | 0.50 |
| SD | | 2.56 | 0 | 2.56 | 0 | 2.56 | 0 | 2.56 | 0 |

- When using larger values of $d$ the resulting grid graphs become relatively sparse, to the extent that useful geometric information is hard to capture, hence the VULS classification effectiveness tends to deteriorate.

Overall when using smaller grid sizes, such as 8 pixels, each grid square is more likely to include only one ground type, the above issues can thus be relieved to some extent by using small grid sizes (but at the cost of additional run time). This is why the VULS classifiers when applied in the context of satellite image interpretation tend to perform better with smaller grid sizes.

## 8.4 Classification Effectiveness with Respect to $|L_E|$ (Objective 3)

In this section the effect that the size of the edge label set $L_E$ has on classification effectiveness, in terms of accuracy and AUC, is discussed. The discussion is founded on a sequence of experiments conducted using a range of values for $|L_E|$ from 2 to 8 increasing in steps of 2. Recall that the same range of values was used with respect to the AISF experiments. For the remaining parameters the following was used: $max = 4$ and $|L_V| = 2$. The first was chosen because the previously reported experiments had already demonstrated that good coverage (more than 99.69%) was reached when $max = 4$. The second so as to be consistent with the experiments associated with Objective 1. All the graphs considered were grid graphs (degree=4) featuring directed edges, and used grid size $d = 8$ as the previous section had demonstrated that small grid sizes produce better results. The results are presented in Table 8.6.

From Table 8.6 it can be observed that, with respect to AUC, there was no significant difference between the recorded values regardless of the value used for $|L_E|$. However, Complete VULS mining using $|L_E| = \{4, 6, 8\}$ can achieve AUC of 1 (highlighted in green in the table), indicating that VULS classifiers have potential in the context of satellite images. This will be explored further in the next three sections.

With respect to the above, and in contrast to the results obtained using the AISF grid graphs as reported in the previous chapter, the selected value of $|L_E|$ did not have a remarkable effect on classification effectiveness. Recall that in the case of the AISF data accuracy and AUC tended to increase as $|L_E|$ increased.

## 8.5 Comparison Between Usage of Grid Graphs and Cross Grid Graphs (Objective 4)

This section focuses on the distinction between the usage of grid graphs (degree 4) and cross grid graphs (degree of 8) in terms of classification effectiveness. For the experiments the following parameters were used: $max = 4$, $|L_V| = 2$, $d = 8$, and $|L_E| = 4$. The results are presented in Table 8.7.

TABLE 8.6: Classification Effectiveness with Respect to $|L_E|$ (Objective 3).

| $\|L_E\|$ | Fold | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | |
|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| 2 | 1 | 98.93 | 0.99 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 2 | 99.22 | 0.69 | 99.02 | 0.62 | 98.73 | 0.50 | 98.73 | 0.50 |
| | 3 | 99.61 | 0.67 | 99.61 | 0.67 | 99.41 | 0.50 | 99.41 | 0.50 |
| | 4 | 99.51 | 0.79 | 99.32 | 0.71 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 5 | 99.90 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 6 | 99.12 | 0.63 | 99.02 | 0.58 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 7 | 99.80 | 0.75 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 8 | 99.71 | 0.75 | 99.61 | 0.62 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 9 | 99.80 | 0.75 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 10 | 99.51 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| Average | | 99.51 | 0.75 | 99.58 | 0.68 | 99.40 | 0.50 | 99.40 | 0.50 |
| SD | | 0.32 | 0.10 | 0.34 | 0.07 | 0.44 | 0 | 0.44 | 0 |
| 4 | 1 | 100.00 | 1.00 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 2 | 99.41 | 0.77 | 99.02 | 0.62 | 98.73 | 0.50 | 98.73 | 0.50 |
| | 3 | 99.61 | 0.67 | 99.61 | 0.67 | 99.41 | 0.50 | 99.41 | 0.50 |
| | 4 | 99.41 | 0.79 | 99.22 | 0.67 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 5 | 99.90 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 6 | 99.22 | 0.67 | 98.93 | 0.54 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 7 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 8 | 99.80 | 0.75 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 9 | 99.80 | 0.75 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 10 | 99.71 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| Average | | 99.67 | 0.77 | 99.56 | 0.66 | 99.40 | 0.50 | 99.40 | 0.50 |
| SD | | 0.25 | 0.09 | 0.37 | 0.09 | 0.44 | 0 | 0.44 | 0 |
| 6 | 1 | 100.00 | 1.00 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 2 | 99.41 | 0.77 | 99.22 | 0.69 | 98.73 | 0.50 | 98.73 | 0.50 |
| | 3 | 99.61 | 0.67 | 99.51 | 0.58 | 99.41 | 0.50 | 99.41 | 0.50 |
| | 4 | 99.41 | 0.79 | 99.41 | 0.75 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 5 | 99.90 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 6 | 99.02 | 0.58 | 99.02 | 0.58 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 7 | 99.80 | 0.75 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 8 | 99.71 | 0.63 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 9 | 99.71 | 0.63 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 10 | 99.80 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| Average | | 99.64 | 0.73 | 99.61 | 0.69 | 99.40 | 0.50 | 99.40 | 0.50 |
| SD | | 0.29 | 0.12 | 0.31 | 0.07 | 0.44 | 0 | 0.44 | 0 |
| 8 | 1 | 100.00 | 1.00 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 2 | 99.32 | 0.73 | 98.83 | 0.54 | 98.73 | 0.50 | 98.73 | 0.50 |
| | 3 | 99.61 | 0.67 | 99.61 | 0.67 | 99.41 | 0.50 | 99.41 | 0.50 |
| | 4 | 99.51 | 0.79 | 99.12 | 0.63 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 5 | 99.90 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 6 | 99.12 | 0.63 | 99.02 | 0.58 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 7 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 8 | 99.71 | 0.63 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 9 | 99.80 | 0.75 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 10 | 99.80 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| Average | | 99.66 | 0.75 | 99.53 | 0.64 | 99.40 | 0.50 | 99.40 | 0.50 |
| SD | | 0.27 | 0.11 | 0.40 | 0.09 | 0.44 | 0 | 0.44 | 0 |

TABLE 8.7: Classification Effectiveness with Respect to graph types (Objective 4).

| Graph Category | Fold | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | |
|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| directed grid | 1 | 100.00 | 1.00 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 2 | 99.41 | 0.77 | 99.02 | 0.62 | 98.73 | 0.50 | 98.73 | 0.50 |
| | 3 | 99.61 | 0.67 | 99.61 | 0.67 | 99.41 | 0.50 | 99.41 | 0.50 |
| | 4 | 99.41 | 0.79 | 99.22 | 0.67 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 5 | 99.90 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 6 | 99.22 | 0.67 | 98.93 | 0.54 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 7 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 8 | 99.80 | 0.75 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 9 | 99.80 | 0.75 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 10 | 99.71 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| Average | | 99.67 | 0.77 | 99.56 | 0.66 | 99.40 | 0.50 | 99.40 | 0.50 |
| SD | | 0.25 | 0.09 | 0.37 | 0.09 | 0.44 | 0 | 0.44 | 0 |
| directed cross-grid | 1 | 100.00 | 1.00 | 100.00 | 1.00 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 2 | 99.51 | 0.81 | 99.41 | 0.77 | 98.73 | 0.50 | 98.73 | 0.50 |
| | 3 | 99.71 | 0.75 | 99.61 | 0.67 | 99.41 | 0.50 | 99.41 | 0.50 |
| | 4 | 99.61 | 0.87 | 99.41 | 0.79 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 5 | 99.90 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 6 | 99.02 | 0.58 | 99.12 | 0.63 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 7 | 99.80 | 0.75 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 8 | 99.90 | 0.88 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 9 | 99.80 | 0.75 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 10 | 99.90 | 0.81 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| Average | | 99.72 | 0.80 | 99.67 | 0.75 | 99.40 | 0.50 | 99.40 | 0.50 |
| SD | | 0.29 | 0.11 | 0.28 | 0.11 | 0.44 | 0 | 0.44 | 0 |

From Table 8.7 it can be observed that, regarding Complete VULS and minimal VULS, cross-grid graphs produced more effective vertex classification results than when grid graphs were used. Recall that the opposite tended to be observed with respect to the sheet metal forming experiments reported in Chapter 7. It is conjectured that the reasons for this is because:

- the vertex label distribution of graphs in the context of the satellite image data set is extremely imbalanced and relatively balanced in the context of the AISF data set (see chapter 3).
- during the image to graph translation process, we divide each image into grid squares evenly, some grid squares might contain more than one ground type but are labelled with a single ground type, as a result, some ground type information is lost during the translation process. Thus using cross grid graphs with degree 8 can compensate for this information loss to some extent (more edges).
- when using degree 8 a greater number of edges are included in the input graph which in turn results in more VULS, and with the direction restriction, a more comprehensive (effective) set of VULS are generated.

TABLE 8.8: Classification Effectiveness with Respect to $|L_V|$ (Objective 5).

| $\|L_V\|$ | Fold | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | |
|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| 2 | 1 | 100.00 | 1.00 | 100.00 | 1.00 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 2 | 99.51 | 0.81 | 99.41 | 0.77 | 98.73 | 0.50 | 98.73 | 0.50 |
| | 3 | 99.71 | 0.75 | 99.61 | 0.67 | 99.41 | 0.50 | 99.41 | 0.50 |
| | 4 | 99.61 | 0.87 | 99.41 | 0.79 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 5 | 99.90 | 0.75 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| | 6 | 99.02 | 0.58 | 99.12 | 0.63 | 98.83 | 0.50 | 98.83 | 0.50 |
| | 7 | 99.80 | 0.75 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 8 | 99.90 | 0.88 | 99.71 | 0.63 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 9 | 99.80 | 0.75 | 99.80 | 0.75 | 99.61 | 0.50 | 99.61 | 0.50 |
| | 10 | 99.90 | 0.81 | 99.90 | 0.75 | 99.80 | 0.50 | 99.80 | 0.50 |
| Average | | 99.72 | 0.80 | 99.67 | 0.75 | 99.40 | 0.50 | 99.40 | 0.50 |
| SD | | 0.29 | 0.11 | 0.28 | 0.11 | 0.44 | 0 | 0.44 | 0 |
| 3 | 1 | 10.74 | 0.42 | 86.23 | 0.67 | 87.30 | 0.33 | 87.79 | 0.35 |
| | 2 | 34.47 | 0.44 | 69.43 | 0.56 | 68.16 | 0.33 | 67.29 | 0.34 |
| | 3 | 23.14 | 0.44 | 36.62 | 0.50 | 34.38 | 0.33 | 33.69 | 0.32 |
| | 4 | 43.16 | 0.47 | 81.74 | 0.59 | 82.32 | 0.33 | 79.98 | 0.34 |
| | 5 | 2.73 | 0.18 | 91.41 | 0.50 | 91.80 | 0.33 | 91.80 | 0.33 |
| | 6 | 27.93 | 0.24 | 79.69 | 0.39 | 80.66 | 0.33 | 80.37 | 0.35 |
| | 7 | 22.17 | 0.31 | 54.20 | 0.51 | 53.13 | 0.33 | 53.42 | 0.34 |
| | 8 | 39.06 | 0.50 | 60.74 | 0.51 | 59.57 | 0.33 | 59.57 | 0.33 |
| | 9 | 31.15 | 0.45 | 73.14 | 0.60 | 71.88 | 0.33 | 73.63 | 0.37 |
| | 10 | 41.70 | 0.55 | 80.57 | 0.67 | 82.81 | 0.33 | 80.18 | 0.34 |
| Average | | 27.63 | 0.40 | 71.38 | 0.55 | 71.20 | 0.33 | 70.77 | 0.34 |
| SD | | 13.27 | 0.12 | 16.69 | 0.09 | 17.82 | 0 | 17.70 | 0.01 |

## 8.6 Effect of $|L_V|$ on Classification Effectiveness (Objective 5)

As noted previously, the value for $|L_V|$ is usually application domain dependent. However, for completeness, in this section we consider the effect on classification when $|L_V| = 2$ and $|L_V| = 3$ (the nature of the labelling in each case was discussed in Chapter 3). For the experiments $|L_E| = 4$, $d = 8$, and $max = 4$ were used. Only cross-grid graphs were considered because the previous experiments, reported in Section 8.5, indicated that these tended to produce the best results. The results are presented in Table 8.8.

From Table 8.8, as expected, better accuracy and AUC can be achieved when the number of vertex labels is small ($|L_V| = 2$). When the number of vertex labels is increased to 3, the classification performance tends to deteriorate dramatically (AUC values of less than 0.5 were recorded). Similar, but less dramatic, results were observed with respect to the AISF experiments reported in the previous chapter. The reason will be further explored in the next section.

TABLE 8.9: Classification Effectiveness with Respect to graph types (Objective 6).

| Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | | J48 | | Naive Bayes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| SIh8E2V2 | 0.90 | 0.76 | 0.99 | 0.82 | 0.99 | 0.50 | 0.99 | 0.50 | 0.99 | 0.87 | 0.99 | 0.87 |
| SIh8E3V2 | 0.99 | 0.81 | 0.99 | 0.60 | 0.99 | 0.50 | 0.99 | 0.50 | 0.99 | 0.98 | 0.99 | 0.98 |
| SIh8E4V2 | 0.99 | 0.81 | 0.99 | 0.75 | 0.99 | 0.50 | 0.99 | 0.50 | 0.99 | 0.99 | 0.99 | 0.99 |
| SIh8E5V2 | 0.99 | 0.78 | 0.99 | 0.68 | 0.99 | 0.50 | 0.99 | 0.50 | 0.99 | 0.99 | 0.99 | 0.99 |
| SIh8E6V2 | 0.99 | 0.76 | 0.99 | 0.69 | 0.99 | 0.50 | 0.99 | 0.50 | 0.99 | 0.99 | 0.99 | 0.99 |
| SIh8E7V2 | 0.99 | 0.75 | 0.99 | 0.69 | 0.99 | 0.50 | 0.99 | 0.50 | 0.99 | 0.99 | 0.99 | 0.99 |
| SIh8E8V2 | 0.99 | 0.70 | 0.99 | 0.71 | 0.99 | 0.50 | 0.99 | 0.50 | 0.99 | 0.99 | 0.99 | 0.99 |
| SIh16E2V2 | 0.99 | 0.50 | 0.99 | 0.67 | 0.99 | 0.50 | 0.99 | 0.50 | 0.99 | 0.75 | 0.99 | 0.86 |
| SIh16E3V2 | 0.99 | 0.56 | 0.99 | 0.71 | 0.99 | 0.50 | 0.99 | 0.50 | 0.99 | 0.90 | 0.99 | 0.97 |
| SIh16E4V2 | 0.99 | 0.61 | 0.99 | 0.61 | 0.99 | 0.55 | 0.99 | 0.50 | 0.99 | 0.95 | 0.99 | 0.96 |
| SIh16E5V2 | 0.99 | 0.61 | 0.99 | 0.56 | 0.99 | 0.60 | 0.99 | 0.50 | 0.99 | 0.97 | 0.99 | 0.97 |
| SIh16E6V2 | 0.99 | 0.61 | 0.99 | 0.62 | 0.99 | 0.60 | 0.99 | 0.50 | 0.99 | 0.98 | 0.99 | 0.98 |
| SIh16E7V2 | 0.99 | 0.61 | 0.99 | 0.66 | 0.99 | 0.55 | 0.99 | 0.50 | 0.99 | 0.99 | 0.99 | 0.98 |
| SIh16E8V2 | 0.99 | 0.69 | 0.99 | 0.61 | 0.99 | 0.61 | 0.99 | 0.50 | 0.99 | 0.94 | 0.99 | 0.99 |
| SIh32E2V2 | 0.96 | 0.50 | 0.96 | 0.50 | 0.96 | 0.50 | 0.96 | 0.50 | 0.98 | 0.62 | 0.97 | 0.86 |
| SIh32E3V2 | 0.96 | 0.50 | 0.97 | 0.55 | 0.96 | 0.50 | 0.96 | 0.50 | 0.98 | 0.70 | 0.97 | 0.94 |
| SIh32E4V2 | 0.96 | 0.50 | 0.96 | 0.55 | 0.96 | 0.50 | 0.96 | 0.50 | 0.98 | 0.74 | 0.97 | 0.95 |
| SIh32E5V2 | 0.96 | 0.50 | 0.96 | 0.50 | 0.96 | 0.50 | 0.96 | 0.50 | 0.99 | 0.88 | 0.98 | 0.98 |
| SIh32E6V2 | 0.96 | 0.50 | 0.97 | 0.55 | 0.96 | 0.50 | 0.96 | 0.50 | 0.98 | 0.77 | 0.98 | 0.98 |
| SIh32E7V2 | 0.96 | 0.50 | 0.96 | 0.50 | 0.97 | 0.55 | 0.96 | 0.50 | 0.98 | 0.64 | 0.98 | 0.99 |
| SIh32E8V2 | 0.96 | 0.50 | 0.97 | 0.55 | 0.96 | 0.50 | 0.96 | 0.50 | 0.98 | 0.69 | 0.98 | 0.99 |
| Average | 0.98 | 0.62 | 0.98 | 0.62 | 0.98 | 0.52 | 0.98 | 0.50 | 0.99 | 0.87 | 0.99 | 0.96 |
| SD | 0.02 | 0.12 | 0.01 | 0.09 | 0.01 | 0.04 | 0.01 | 0 | 0.01 | 0.13 | 0.01 | 0.04 |

## 8.7 Comparison of VULS Vertex Classification Effectiveness (Objective 6)

This section reports on the comparative evaluation conducted with respect to VULS vertex classification. This section includes a comparison with using more traditional forms of classification, namely J48 and Naive Bayes (as implemented in the Waikato Environment for Knowledge Analysis (WEKA) machine learning workbench[1]). For usage with J48 and Nave Bayes the data sets were processed as described in Chapter 3. For the results presented in Table 8.9, $max = 4$, $|L_E| = \{2, 3, 4, 5, 6, 7, 8\}$ and $|L_V| = 2$. All the graphs considered were cross grid graphs (degree= 8) and featured directed edges. The results are presented in Table 8.9. The table is laid out in a similar manner to those included in the previous sections except that in this case only the average results from the TCV are reported.

With reference to Table 8.9, the VULS based vertex classification did not work as well as expected. The main reasons why VULS based vertex classification did not perform as well as in the context of sheet metal forming are as follows:

- Minimal VULS can produce as good a result as when using the complete set of VULS. This was not the case with respect to the AISF application; where frequent VULS produce as good a result as when using the complete set of VULS. This is probably because minimal VULS are good at capturing significant geometric patterns with rare vertex labellings, recall that the vertex label distribution is

[1] http://www.cs.waikato.ac.nz/ml/weka/

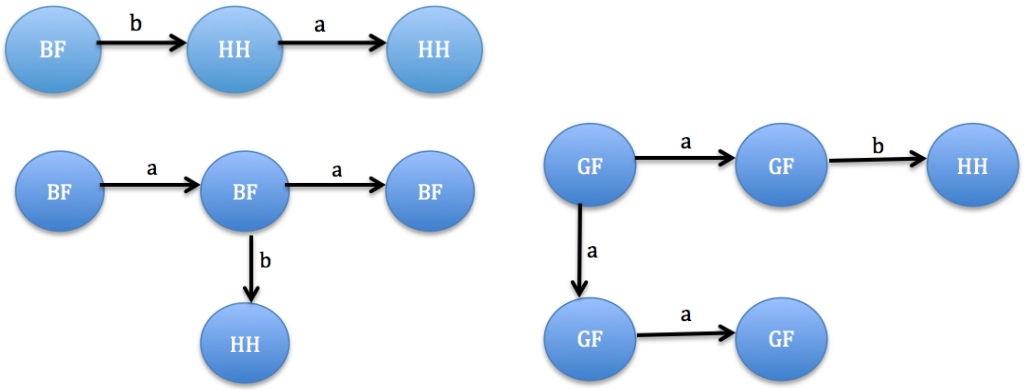



FIGURE 8.1: Examples of VULS identified in graphs where $d = 8$ pixels.

extremely imbalanced in the context of the satellite image data set as noted in Chapter 3. Thus minimal VULS can achieve as good a performance as when using the complete set of VULS in unbalanced graphs.

- As noted above, the vertex label distribution within the satellite image graph data was extremely imbalanced. This in turn has an effect on classifier training (lack of examples with respect to infrequent vertex labels) and consequently the application of the classifier.

- The grayscale intensity of each ground type ("BrownField", "GreenField" and "HouseHold") varies, but the grayscale intensity difference (represented as an edge label) between pairs of vertices might be identical, which in turn may lead to the proposed Backward-Match-Voting algorithm being unable to identify the correct vertex labelling. For example, Figure 8.1 shows a number of VULS identified using $d = 8$ pixels. Recall that vertex label BF represents "BrownField", GF represents "GreenField" and HH represents "HouseHold". From the figure it can be seen that labelling of the vertices associated with edge label 'a' would be hard to distinguish. Similarly the end vertex labels for edges labelling with 'b' cannot be distinguished. However, it should of course be appreciated that if a vertex labelling associated with an edge is not unique we would not have a VULS in the first place.

In the image to graph translation process, if we group pixels using some clustering technique, instead of grid squares, the grayscale intensity difference (represented as edge labels) between vertex pairs might produce a better result, however, this is an issue for future study. For the current research presented here, if the grid size $d$ is small enough to ensure each grid square only contains one ground type and the ground types of neighbouring grid squares are more distinctive, the above issue can be relieved slightly. Thus our proposed VULS classifiers are more likely to perform well using small grid sizes as already demonstrated in section 8.3.

## 8.8 Statistical Comparison of the Proposed VULS Approaches on Satellite Image data (Objective 7)

This section reports on the statistical comparison of the operation of the proposed VULS mining with respect to classification effectiveness (AUC) in the context of the satellite image application. For the experiments both grid and cross grid graphs were considered (degree 4 and 8). A value of $max$=4 was used throughout. All 4 VULS algorithms (compVULS, MinVULS, freqVULS, and minFreqVULS) were considered. Thus eight different kinds of VULS classifiers in total (using $degree = 4$ and $degree = 8$). As in the case of the AISF statistical comparison reported in Chapter 7, more traditional classification techniques J48 and Naive Bayes were also considered. Thus a total of 10 different classifiers.

Two separate sets of experiments were conducted, one using $|L_V| = 2$ and one using $|L_V| = 3$. For each of the ten different kinds of classifier, identified in the previous paragraph, 21 individual classifiers were generated by considering: (i) a range of seven values for $L_E$ ($\{2, 3, 4, 5, 6, 7, 8\}$ and (ii) a range of three values for $d$ ($\{8, 16, 32\}$ ); giving a total of $7 \times 3 = 21$ different classifiers of each type.

The average ranking of the classifiers, with respect to $|L_V| = 2$ and $|L_V| = 3$, are presented in Tables 8.10 and 8.11 respectively. In the tables the "AR" column gives the average AUC ranked performance value and the "AR+CD" gives the average AUC ranked performance value plus the calculated Critical Difference (CD). Recall that the values in the "AR+CD" column can be used to identify classifiers whose operation is statistically different from others (this is the case if its AR value is outside of the range AR to AR+CD).

With respect to Table 8.10, where $|L_V| = 2$, the Friedman test value, using $K - 1 = 10 - 1 = 9$ degrees of freedom, is as follows:

$$\begin{aligned}\chi_F^2 &= \frac{12\times21}{10\times(11)}\left[355.83106575963734 - \frac{10\times(11)^2}{4}\right]\\ &= 2.29090909 \times 53.33106575\\ &= 122.17662337662372\end{aligned}$$

Recall that K is the number of classifiers, 10 in this case (8 VULS classifiers, together with J48 and Naive Bayes), D is the number of data sets (21 as noted above). The p-value (threshold) is then $9.916E-11$. The F-distribution with 9 ($K - 1 = 10 - 1 = 9$) and 180 ($(K - 1) * (D - 1) = (10 - 1) * (21 - 1) = 180$) degrees of freedom, $F(9, 180)$, is 36.57. The critical value for $F(9, 180)$, with a critical difference level of $\alpha = 0.05$, is 1.932. Thus, from the foregoing, we can note that the p-value ($9.916E-11$) is less than 0.005 and that the F-distribution (36.57) is larger than the corresponding F-distribution critical value (1.932). Therefore we can reject the null hypothesis $H_0$ that there is no statistically significant difference in the observed performance differences among the classifiers considered. Thus a Nemenyi post-hoc test was deemed to be applicable to detect which particular classifiers differed in a statistically significantly manner from the others.

The resulting significance diagram, corresponds to the information presented in Table 8.10, is presented in Figure 8.2. The classifiers are listed along the Y axis in ascending order of their ranked AUC performance. The AUC performance rankings, along with the Nemenyi's Critical Difference (CD) tail, are presented along the X axis. The CD value for the diagram is equal to 2.09 calculated as follows:

$$\begin{aligned} CD &= q_{\alpha,\infty,10}\sqrt{\tfrac{10\times 11}{12\times 21}} \\ &= 3.164 \times 0.66068 \\ &= 2.090415 \end{aligned}$$

The two vertical dashed lines included in the figure indicate the end of the best performing classifier's tail and the start of the next statistically different classifier. From the figure it can be seen that with respect to graphs where $|L_V| = 2$, the best performing classifier was Naive Bayes (a recorded AR value of 1.262), J48 achieved comparable performance whilst VULS classifiers perform significantly worse than Naive Bayes and J48. Relatively speaking, minimal VULS with degree 8 produced the best performance amongst the VULS classifiers, whilst Complete VULS and minimal VULS performed significantly better than frequent VULS and minimal frequent VULS regardless of the degree value used.

TABLE 8.10: Average Rankings of classifiers where $|L_V| = 2$

| Algorithm | AR | AR+CD |
|---|---|---|
| Directed-Degree4-max4-compVULS | 5.905 | 7.995 |
| Directed-Degree4-max4-minVULS | 6.048 | 8.138 |
| Directed-Degree4-max4-freqVULS | 7.571 | 9.662 |
| Directed-Degree4-max4-minFreqVULS | 7.976 | 10.067 |
| J48 | 1.738 | 3.829 |
| Naive Bayes | 1.262 | 3.352 |
| Directed-Degree8-max4-compVULS | 4.952 | 7.043 |
| Directed-Degree8-max4-minVULS | 4.452 | 6.543 |
| Directed-Degree8-max4-freqVULS | 7.024 | 9.114 |
| Directed-Degree8-max4-minFreqVULS | 8.071 | 10.162 |

TABLE 8.11: Average Rankings of classifiers where $|L_V| = 3$

| Algorithm | AR | AR+CD |
|---|---|---|
| Directed-Degree4-max4-compVULS | 7.071 | 9.162 |
| Directed-Degree4-max4-minVULS | 5.286 | 7.376 |
| Directed-Degree4-max4-freqVULS | 7.4045 | 9.495 |
| Directed-Degree4-max4-minFreqVULS | 7.976 | 10.067 |
| J48 | 1.381 | 3.471 |
| Naive Bayes | 1.619 | 3.709 |
| Directed-Degree8-max4-compVULS | 5.190 | 7.281 |
| Directed-Degree8-max4-minVULS | 3.952 | 6.043 |
| Directed-Degree8-max4-freqVULS | 7.786 | 9.876 |

| | | |
|---|---|---|
| Directed-Degree8-max4-minFreqVULS | 7.333 | 9.424 |

With respect to Table 8.11 ($|L_V| = 3$), the chi-squared ($\chi^2_F$) Friedman test value, with $K - 1 = 10 - 1 = 9$ degrees of freedom, was calculated as the follows:

$$\begin{aligned}\chi^2_F &= \frac{12\times21}{10\times(11)}\left[357.8798185941044 - \frac{10\times(11)^2}{4}\right]\\ &= 2.29090909 \times 55.37981859410439\\ &= 126.870129870\end{aligned}$$

The associated p-value (threshold) was then $6,246E - 11$. The F-distribution with $K - 1 = 9$ and $(K - 1) * (D - 1) = (10 - 1) * (21 - 1) = 180$ degrees of freedom, $F(9, 180)$, was then 40.84. The critical value for $F(9, 180)$, with a critical difference level of $\alpha = 0.05$, is 1.932. As in the case of $|L_V| = 2$, the p-value is less than 0.005 and the F-distribution is larger than the corresponding F-distribution critical value (1.932). Therefore, as before, the null hypothesis $H_0$ can be rejected. Consequently a Nemenyi post-hoc test was deemed to be appropriate. The significance diagram is presented in Figure 8.3. As in the case of the $|L_V| = 2$ results, the figure shows the AUC performance rankings for the classifiers, along with the Nemenyi Critical Difference (CD) tail in each case. As before, the CD value for the diagram is equal to 2.09 calculated as shown above.

From Figure 8.3 it can again be seen that, with respect to graphs where $|L_V| = 3$, the best performing classifier was Naive Bayes (a recorded AR value of 1.619), J48 achieved comparable performance whilst the VULS classifiers performed significantly worse than Naive Bayes and J48. As before, and relatively speaking, minimal VULS with degree 8 produced the best performance amongst the VULS classifiers considered, whilst minimal VULS and Complete VULS with degree 8 perform significantly better than frequent VULS and minimal frequent VULS regardless of degree.

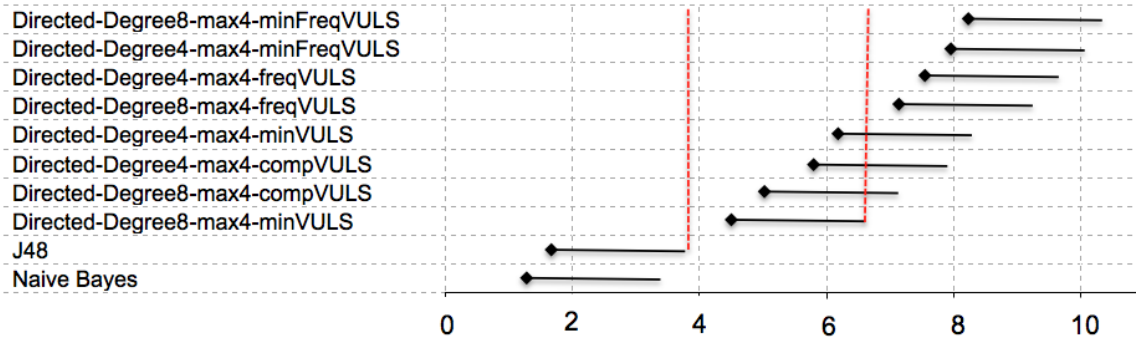



FIGURE 8.2: Critical difference diagram generated using Nemenyi's post hoc test with $\alpha = 0.05$ for graphs where $|L_V| = 2$ .

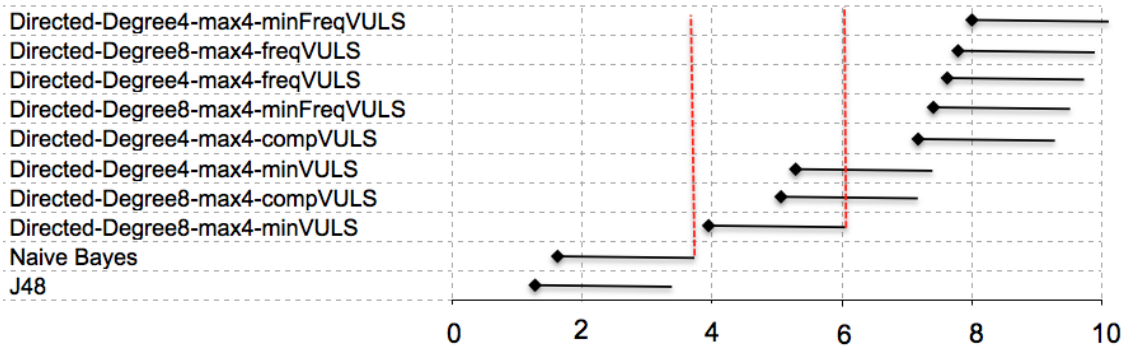



FIGURE 8.3: Critical difference diagram generated using Nemenyi's post hoc test with $\alpha = 0.05$ for graphs where $|L_V| = 3$ .

## 8.9 Summary

This chapter has presented an evaluation of the proposed VULS mining and vertex classification processes in the context of satellite image interpretation. From the evaluation the following main findings can be made:

1. Minimal VULS mining when applied to directed cross-grid graphs (graphs of degree 8) produced the most effective results with respect to the satellite image application domain.

2. There is no significant difference in classification effectiveness, measured in terms of accuracy and AUC, between VULS classifiers built using different $max$ parameter settings (at least in the range of $\{4, 5, 6\}$). However, using $max = 4$ is more efficient in the context of VULS mining (and in some cases produces a slightly better classification result).

3. The set of VULS produced using directed cross-grid graphs resulted in the most effective vertex classification (in terms of AUC).

4. The VULS classifiers performed better on graphs that featured a small grid size, such as $d = 8$ pixels, than graphs that featured a larger grid size such as $d = 16$ and $d = 32$ pixels.

5. With respect to the size of $|L_V|$, as the number of vertex labels increases from 2 to 3, the graph labelling becomes more diverse; as a result, the classification performance tended to deteriorate.

6. When the operation of the VULS classifiers, in terms of accuracy and AUC, was compared with operation of more standard (traditional) classifiers, namely J48 and Nave Bayes, the latter were unfortunately found to produce a better performance.

7. Referring back to chapter 7 the evaluated VULS classifiers produced better results with respect to the sheet metal forming AISF application domain than the satellite

image interpretation domain. It was conjectured that this was largely because the vertex label distribution of the satellite image graphs was extremely unbalanced.

This chapter concludes the evaluation of the proposed VULS mining concept considered in this thesis. The thesis is concluded in the following chapter with a summary of the work presented, along with the main findings in the context of the research objectives presented in Chapter 1, and some suggestions for future work.

# Chapter 9

# Conclusion and Future Research

## 9.1 Introduction

This concluding chapter presents an overall summary of the work described in this thesis along with the main findings and contributions. This chapter also provides some suggestions for future work. The chapter is organised as follows. In Section 9.2 an overall summary of the thesis is presented. The main findings and contributions are reported in Section 9.3. Finally, some suggested ideas for future work are presented in Section 9.4 in the context of further potential research based on the work described in this thesis.

## 9.2 Summary

This section presents an overall summary of the work presented. The thesis commenced in Chapter 2 with a review of previous work, and then went on to consider the data sets used in Chapter 3. Two application domains were considered. The first, and the main focus of the work, was a sheet metal forming application, namely the prediction of a phenomena known as springback. In this context the data sets used were generated using a process known as Asymmetric Incremental Sheet Forming (AISF), and were supplied by the IBF (Institut für Bildsame Formgebung) institute of metal forming at Aachen University. The second application domain considered was a satellite image interpretation application where the aim was to identify ground types. This second application was not the main focus for the work, but was intended to illustrate the wider applicability of the VULS idea. In both cases the raw data was translated into a grid format from which grid graphs, where each node represented a grid square, were generated. In the case of the AISF application the vertex labels were springback values; in the case of the satellite image interpretation the labels were ground types. The advantage offered by this representation was that the VULS Mining concept could be applied in a relatively straight forward manner.

The formalism for the proposed VULS concept was established in Chapter 4, where four different categories of VULS were also defined: (i) Complete VULS, (ii) Minimal VULS, (iii) Frequent VULS and (iv) Minimal Frequent VULS. In Chapter 5 four VULS

mining algorithms were proposed, one for each of the identified categories. The main issues these algorithms sought to address were:

1. The need to identify a sufficiently comprehensive set of VULS (a collection of VULS that will ensure good "coverage" with respect to unseen data).
2. The large number of potential VULS that can be contained in a reasonably sized graph (the identification process thus needed to be efficient).

Chapter 6 proposed the Backward Match Voting algorithm for using pre-labelled sub-graphs (such as VULS generated using VULS mining algorithms presented in Chapter 5) to predict vertex labels. The main issues here were how best to address the situations where, with respect to a specific vertex in an unseen data set, either: (i) several competing VULS can be used to label the vertex, or (ii) no appropriate VULS can cover the vertex.

The evaluation of the proposed VULS algorithms was presented in Chapters 7 and 8 in the context of AISF springback prediction and satellite image ground type prediction respectively. The evaluation was conducted mostly in the context of vertex classification and with respect to a number of input parameters ($max$, $|L_V|$, $L_E$ and $d$). Evaluation was also conducted comparing the usage of graphs with degree four and eight; and, in the case of the AISF data, comparing the usage of directed and undirected graphs. In addition some overall evaluation was conducted with respect to classification effectiveness, including comparison with Naive Bayes and J48. The recorded evaluation unfortunately indicated that Naive Bayes produced the best performance, while the proposed VULS classifiers could achieve a comparable performance to J48, and sometimes a better performance, depending on the vertex label distribution of the input graph. However, it should be noted that the Naive Bayes and J48 classifiers could only be used because of the particular, very structured, nature of the grid (and cross grid) graphs considered, they could not be easily applied in the case of more unstructured input graph data where the VULS concept would still be applicable. In terms of efficiency the evaluation reported in Chapters 7 and 8 indicated that the Frequent VULS mining algorithm was the fastest in the case of the sheet metal forming AISF application, whilst the Minimal VULS mining algorithm was the fastest in the context of the satellite image interpretation application (the reasons for this were discussed in the relevant chapters). Both chapters 7 and 8 were concluded with a statistical analysis of the results using the Friedman and the Nemenyi post-hoc tests. .

## 9.3 Main Findings

This section revisits the overriding research question and the subsidiary technical and application dependent research questions presented in Chapter 1 (Section 1.3) and describes how each was resolved in terms of the "main findings" of the research presented in this thesis. The section is organised by considering each of the subsidiary technical

and application research questions in turn and then returning to the overriding research question. The technical research sub-questions postulated in the introduction to this thesis are considered first as follows.

1. *What is the most appropriate mechanism for identifying VULS?* The work conducted suggested that the most appropriate mechanism for identifying VULS was to adopt a candidate generation and test approach along the lines used in the well established gSpan algorithm. Right most extension was selected for the purpose of candidate VULS generation without duplication. The most appropriate mechanism for defining VULS was considered to be the usage of some canonical form. This was seen as important because it allowed graphs to be represented in a unique manner thus allowing for straight forward isomorphism testing. More specifically minimal Depth First Search (DFS) lexicographic encoding was adopted.

2. *Can efficiency gains be realised by mining some subset of the complete set of VULS?* The presented evaluation clearly indicated that efficiency gains can be made by, instead of mining the complete set of VULS, mining only frequent VULS or minimal VULS. More specifically, recall that VULS mining is computationally expensive. Thus instead of identifying the complete set of VULS, we can attempt to identify some appropriately descriptive (in terms of coverage) subset of the complete set of VULS: (i) minimal VULS, (ii) frequent VULS or (iii) minimal frequent VULS. The relationship between these four VULS categories can be expressed as: (i) VULS $\supseteq$ minimal VULS $\supseteq$ minimal frequent VULS; and (ii) VULS $\supseteq$ frequent VULS $\supseteq$ minimal frequent VULS. For minimal VULS and minimal frequent VULS, only the k-edge subgraphs that were found not to be VULS were extended to generate (k+1)-edge VULS candidates for the next iteration so as to reduce the number of over all VULS candidates and improve the VULS mining efficiency. Above all, the number of minimal VULS, frequent VULS and minimal frequent VULS were typically less than the complete set of VULS. As a consequence VULS classifiers using these categories of VULS required less run time. The evaluation of the proposed techniques indicated that minimal frequent VULS in some cases, and frequent VULS, could produce more efficient results in the case of the AISF sheet metal forming application domain, whilst minimal VULS demonstrated promising results in the context of the satellite image interpretation application domain. In order to statistically differentiate between the operation of the proposed VULS classifiers, a statistical evaluation based on the Friedman and Nemenyi statistical tests was performed. This confirmed the results obtained.

3. *Given that we can mine a variety of different categories of VULS which of these are the most useful in terms of effectiveness and efficiency?* As noted above, it was found that there was no definite answer to this question. Usage of each VULS entailed both advantages and disadvantages. The answer depends on the application domain under consideration. Frequent VULS and minimal frequent VULS

were found to perform well on graphs featuring a balanced vertex label distribution, such as in the context of the AISF sheet metal forming application. Minimal VULS were found to produce promising results on graphs featuring unbalanced vertex label distributions, such as in the context of the satellite image interpretation application. Complete VULS can perform effectively in most application domains, however, this is computationally expensive. Thus instead of using the complete set of VULS, it was found that a subset of VULS (such as: (i) minimal VULS, (ii) frequent VULS or (iii) minimal frequent VULS) could be successfully applied to improve efficiency while still producing an effective performance regardless of vertex distribution.

4. *How do we measure the quality of a set of VULS without applying them to a test set?* One way of testing the quality of a set of identified VULS is to apply them within the context of a vertex classification setting. However, in practice, this opportunity may not available. An alternative VULS quality measure was thus considered to be necessary. To this end the coverage metric was proposed. Coverage was thus used as one of the measures to compare the operation of the proposed VULS mining algorithms. To obtain good coverage the identified set of VULS should describe as wide a range of different configurations as possible; the more vertices with various configurations the better.

5. *Once a set of VULS have been identified what is the most appropriate mechanism for uitilising this set of VULS in the context of vertex classification?* In other words, how best to predict vertex labels in a previously unseen graph, $G$, using an identified set VULS. The first issue here was the matching process to be adopted, this was addressed by using the same canonical form and isomorphism testing strategy proposed in the context of VULS mining. The second was resolved using a voting mechanism. A variety of voting mechanisms could have been adopted, including: (i) majority voting and (ii) weighted voting. The first was adopted. The proposed vertex classification algorithm was termed the *Backward-Match-Voting* algorithm because: (i) it operates in a "backwards" manner from the maximum value for $k$, thus $k = max$, to $k = 1$; (ii) it "matches" graph structures and edge labelings in the identified set of VULS with graph structures and edge labelling in $G$ so as to label the vertices in $G$ using the labels from the VULS; and (iii) where more than one vertex label was assigned to a vertex in $G$ a majority "voting" scheme was applied. The evaluation indicated that a good performance could be achieved using the Backward-Match-Voting algorithm. Note also that the application of the algorithm is not limited to VULS it can be used in other situations where we wish to conduct vertex classification using a set of pre-labelled subgraphs.

The application research sub-questions were addressed as follows:

1. *How best to generate grid graphs given a 3D surface?* In other words, how best to translate "point cloud" raw data, expressed in terms of a set of x-y-z coordinates,

to the desired graph format in a simple, easy and effective manner so that the proposed VULS classifiers can be built? The foundation for this part of the work was taken from [65–67, 130, 184]. The given "point clouds" (in the case of the sheet metal forming application) or pixel sets (in the case of the satellite image interpretation application) were first converted into a grid format. The centre point for each grid square was then considered to represent a vertex within a grid graph and labelled with a vertex class label (springback or ground type). Each vertex could be connected by an edge to its neighbours in a number of manners, a logical approach was to consider either the four cardinal, or eight cardinal and sub-cardinal neighbours. Each edge was labelled using a "slope" categorical label (because the proposed VULS vertex classification could only operate using such labels); equal width discretisation was adopted. Edges could be directed or undirected. In total 32 different raw data sets with respect to the sheet metal forming application, and 10 with respect to satellite image data, were generated in this manner. Experiments were conducted using a variety of values for: (i) the size of the vertex (class) label set $L_V$, (ii) the size of the edge label set $L_E$ and (iii) grid size $d$. The conducted experimental analysis established that VULS based vertex classification performed better using: (i) directed graphs than undirected graphs, (ii) graphs of degree 4 in the context of sheet metal forming and degree 8 in the context of satellite image interpretation.

2. *What further applications can the VULS concept be applied to?* In other words, how could the proposed VULS based vertex classification mechanism be usefully adopted in the broader context? With respect to the work presented in this thesis the main application domain was sheet metal forming. The secondary application domain was satellite image interpretation. More generally the VULS concept has wider applicability in all areas where the domain of interest can be represented in the form of some sort of a graph. This is explored further in the following future work section.

Returning to the main research question:

**"How best can the proposed VULS mining be conducted so as to achieve effective grid graph vertex classification?"**

From the foregoing a number of alternative mechanism for mining VULS were considered founded on four different categories of VULS: Complete VULS, Minimal VULS, Frequent VULS and Minimal Frequent VULS. From the evaluation conducted each of these provided different advantages. Best coverage tended to be produced using Complete VULS. The most efficient was frequent VULS in the context of graphs with a balanced vertex label distribution, and minimal VULS in the context of graphs with an imbalanced vertex label distribution. Note that frequent VULS can capture significant VULS which occur commonly, and thus will from the most representative set of VULS;

whilst Minimal VULS can capture significant geometric patterns including patterns related to rare vertex labels. In the context of vertex classification the Backward Match Voting algorithm was proposed. The reported evaluation indicated that by using this algorithm effective classification could be conducted. More specifically that frequent VULS was the most effective in the context of the AISF application and minimal VULS in the satellite image application.

## 9.4 Future Work

The work presented in this thesis has demonstrated that, in the context of sheet metal forming and satellite image interpretation, vertex classification can be effectively achieved using VULS. Despite the results produced, enhancements and improvements can be envisioned. This concluding section suggests some potential areas for future work as follows:

1. **Application to alternative forms of graph.** Only grid graphs were considered with respect to the work presented in this thesis. This was because of the AISF sheet metal forming application domain, that acted as the main motivation for the work, where the 3D shapes to manufactured are frequently considered in terms of a 2D grid. Translation into grid graphs is therefore an obvious step to take. However, the regular structure of grid graphs is such that, as demonstrated, the graph information can be simply translated into a feature vector format to which standard (tabular data) classifier generators can be applied. As a consequence, and as also demonstrated in the thesis, classifiers such as Naive Bayes were found to sometimes outperform the proposed VULS vertex classification approach. However, it would not be so easy to apply such standard classification techniques to unstructured graphs. The VULS concept is clearly applicable to non-regular graphs. A suggested fruitful direction for future investigations into the utility of VULS is thus their application to non-regular graphs. In the future, running experiments where VULS classifiers are applied to more complex graphs could be interesting. Such as: (i) geometric networks built on point clouds, and (ii) "Variable resolution" networks (similar in nature to quad-trees) so that less informative parts of the 3D surface under investigation could be modelled by coarse grids and others by fine grids.

2. **KNN Graphs.** Following on from the previous item one mechanism for generating an alternative to grid graphs is to produce KNN graphs [142]. A KNN graph is constructed by considering a set of records to be set as vertices. Vertices are connected by edges if they are in some sense similar. The parameter $K$ is then the maximum number of permitted connections. An alternative is to use some similarity threshold to connect graphs. We can then label the vertices using a class label and then generate a set of VULS, as envisioned in this thesis, which can then be applied to previously unseen data. Of course this approach will only be

applicable to data that can be conceived of as a set of independent records. In the context of the AISF sheet metal forming application, one way this could simply be achieved is by considering individual grid squares as records.

3. **Alternative Graph formats for image data using pixel clustering.** In the context of the evaluation using the satellite image interpretation application the intention behind the evaluation was to investigate how well the VULS concept could perform in the context of vertex classification with respect to an alternative application domain than the AISF domain used as the central focus for the research. The reported results indicated that, although the VULS concept worked well in the context of the AISF sheet metal forming, the performance was not as good in the context of the satellite image interpretation application. It was conjectured that this was because, during the preprocessing of the satellite images into grid graphs, pixel intensity information was lost as a result of the process of simply dividing the images into even grid squares. It was not an objective of the work described in this thesis to consider techniques for translating image data into a grid graph format, however, the manner in which this is conduced will clearly have some influence on the final VULS classifiers' performance. An alternative mechanism for generating grid graphs therefore seems an appropriate avenue for further work. One idea is to use clustering techniques to group connected pixels with similar intensity values. In this manner a given image will be divided into a collection of clusters instead of grid squares. Each cluster can then be represented as a vertex in a graph with edges linking neighbouring clusters. As before edges can be labelled using grayscale intensity difference. By adopting this approach it might be possible to improve VULS classifier performance, not only with respect to the satellite image interpretation application considered in this these, but also in the context of alternative image interpretation applications. It is thus anticipated that the use of clustering techniques will improve the performance of the proposed VULS classifiers, this is thus anticipated to be another fruitful avenue for future work.

4. **Pixel Clustering for AISF data.** To improve the application of the VULS concept with respect to the AISF application the pixel clustering idea identified above can equally well be applied in the context of AISF data. In this case, instead of pixels, some atomic point formalism will need to be considered. This will mean that different size areas in a 3D surface to be manufactured, that feature constant springback, can be considered in terms of a single vertex. The result of course will be an irregular graph, however, better springback prediction might result.

5. **Additional evaluation**. To date the VULS concept has only been applied to the AISF sheet metal forming and the satellite image interpretation applications considered in this thesis. Much wider evaluation seems desirable. Even in the context of the satellite image interpretation application the images used were limited to a

rural area; urban areas should also be considered. With respect to the KNN graph idea presented above it would be possible to consider benchmark datasets such as those available within the UCI machine learning repository[1].

6. **VULS pruning.** Not all generated VULS are equally significant in the context of vertex classification. This is why the proposed Backward-Match-Voting (BMV) algorithm prioritises "large" VULS. Using some form of principal component analysis it may be possible to identify the most discriminating VULS with respect to some identified set of class labels. Consequently it might be possible to prune the set of identified VULS to reduce there overall number, which might have an impact on efficiency. Alternatively it might be possible to rank the VULS so that the most discriminating VULS are applied first. This will also necessitate an alternative to the proposed backward match voting algorithm, however, it is suggested that this might enhance the overall performance of VULS vertex classification.

7. **Graph Classification and Clustering Using The VULS Concept.** In this thesis VULS have only been used for vertex classification. However, VULS can also be used for graph classification and clustering. More specifically the VULS identified within a transaction graph set can be used to construct a feature space which can then be used to encode a set of class labelled transaction graphs. The features (VULS) that are frequent for one class may be infrequent for other classes and serve as highly discriminative features (VULS). The identified features can then be used construct a classifier using standard classification approaches which can then be used to classify previously unseen transaction graphs. Various classification approaches may be adopted, for example SVM (Support Vector Machines), Naive Bayes and associative classification. Similarly, cluster analysis can be explored using mined VULS patterns. Sets of graphs that share a large number of similar VULS patterns could be considered to be highly similar and thus grouped into a single clusters.

8. **Transaction graph mining.** So far the usage of VULS has been limited to single graph mining, but it can be easily adapted to transaction graphs. This was partly done with respect to the satellite image interpretation image data, but for future work it might be interesting to consider more extensive (larger) transaction graph sets.

9. **Candidate VULS generation Method.** The well known gSpan algorithm combines subgraph extension and isomorphism testing into one procedure; thus it can achieve competitive, and in some cases better performance than other frequent subgraph mining algorithms in terms of run time. However, gSpan does not work well when the size of the graphs considered are large. For the proposed VULS

[1] http://archive.ics.uci.edu/ml/

algorithms the gSpan data structure was borrowed, thus the efficiency of the proposed VULS mining algorithms might meet a similar "bottleneck" when it comes to very large graphs. Exploring more efficient data structures and techniques for generating candidate VULS, that will permit the processing of very large graphs, would therefore be beneficial.

10. **Frequency measures.** The proposed frequent VULS and minimal frequent VULS mining algorithms rely on a support-based pruning strategy to prune the combinatorial search space. The threshold $\sigma$ plays a significant role in this strategy, especially when it comes to graphs with skewed support distributions. Instead of dynamically setting $\sigma$ according to the average occurrence of subgraphs on each iteration, as proposed in this thesis, the usage of other measures, such as h-confidence [226] might be usefully explored.

11. **Comparison with other vertex classification methods.** As noted in the literature review presented in chapter 2 there are a variety of mechanisms whereby vertex classification can be conducted. Although none of these mechanisms use the graph based approach considered in this thesis. It would be interesting to conduct a comparison between the VULS approach presented in this thesis and a number of these alternative mechanisms.

Overall the research work on the generation and usage of VULS for vertex classification, in the context of sheet metal forming and satellite image interpretation, as presented in this thesis, has produced some interesting outcomes and provided a sound foundation for future work.

# Bibliography


[1] Kenji Abe, Shinji Kawasoe, Tatsuya Asai, Hiroki Arimura, and Setsuo Arikawa. Optimized substructure discovery for semi-structured data. In *Principles of Data Mining and Knowledge Discovery*, pages 1–14. Springer, 2002.

[2] Ramesh C. Agarwal, Charu C. Aggarwal, and V.V.V. Prasad. A tree projection algorithm for generation of frequent item sets. *Journal of Parallel and Distributed Computing*, 61(3):350 – 371, 2001.

[3] Charu C Aggarwal. An introduction to social network data analytics. In Charu C. Aggarwal, editor, *Social Network Data Analytics*, pages 1–15. Springer US, 2011.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[5] A. Albarrak, F. Coenen, and Yalin Zheng. Classification of volumetric retinal images using overlapping decomposition and tree analysis. In *Computer-Based Medical Systems (CBMS), 2013 IEEE 26th International Symposium on*, pages 11–16, June 2013.

[6] J. M. Allwood, G. P. F. King, and J. Duflou. A structured search for applications of the incremental sheet-forming process by product segmentation. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 219(2):239–244, 2005.

[7] Karl-Heinrich Anders, Monika Sester, and Dieter Fritsch. Analysis of settlement structures by graph-based clustering. *Semantische Modellierung*, pages 41–49, 1999.

[8] Tatsuya Asai, Hiroki Arimura, Takeaki Uno, and Shin-ichi Nakano. Discovering frequent substructures in large unordered trees. In Gunter Grieser, Yuzuru Tanaka, and Akihiro Yamamoto, editors, *Discovery Science*, volume 2843 of *Lecture Notes in Computer Science*, pages 47–61. Springer Berlin Heidelberg, 2003.

[9] K. Ataman, W.N. Street, and Yi Zhang. Learning to rank by maximizing auc with linear programming. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 123–129, 2006.

[10] Arik Azran. The rendezvous algorithm: Multiclass semi-supervised learning with markov random walks. In *Proceedings of the 24th international conference on Machine learning*, pages 49–56. ACM, 2007.

[11] Emna Bahri and Stéphane Lallich. Pruning for extracting class association rules without candidate generation. In *Proceedings of The 2009 International Conference on Data Mining, DMIN 2009, July 13-16, 2009, Las Vegas, USA*, pages 11–17, 2009.

[12] M. Bambach, B. Taleb Araghi, and G. Hirt. Strategies to improve the geometric accuracy in asymmetric single point incremental forming. *Production Engineering Research and Development*, 3(2):145–156, 2009.

[13] Michael J. Barnsley and Stuart L. Barr. Distinguishing urban land-use categories in fine spatial resolution land-cover data using a graph-based, structural pattern recognition system. *Computers, Environment and Urban Systems*, 21(34):209 – 225, 1997. Remote Sensing of Urban Systems.

[14] Th Bauer and K Steinnocher. Per-parcel land use classification in urban areas applying a rule-based technique. *GeoBIT/GIS*, 6:24–27, 2001.

[15] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *Learning theory*, pages 624–638. Springer, 2004.

[16] Mikhail Belkin and Partha Niyogi. Using manifold structure for partially labeled classification. In *Advances in neural information processing systems*, pages 929–936, 2002.

[17] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. *Label Propagation and Quadratic Criterion*, page 193216. MIT Press, 2006.

[18] Ursula C Benz, Peter Hofmann, Gregor Willhauck, Iris Lingenfelder, and Markus Heynen. Multi-resolution, object-oriented fuzzy analysis of remote sensing data for gis-ready information. *ISPRS Journal of photogrammetry and remote sensing*, 58(3):239–258, 2004.

[19] Khalida binti Oseman, Norazrina Abu Haris, and Faizin bin Abu Bakar. Data mining in churn analysis model for telecommunication industry. *Journal of Statistical Modeling and Analytics Vol*, 1(19-27), 2010.

[20] Christian Borgelt. Canonical forms for frequent graph mining. In Reinhold Decker and Hans-J. Lenz, editors, *Advances in Data Analysis*, Studies in Classification,

Data Analysis, and Knowledge Organization, pages 337–349. Springer Berlin Heidelberg, 2007.

[21] Christian Borgelt. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):437–456, 2012.

[22] Christian Borgelt and Michael R Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 51–58. IEEE, 2002.

[23] Andrew P. Bradley. The use of the area under the {ROC} curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145 – 1159, 1997.

[24] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[25] U Brandes, M Eiglsperger, M Kaufmann, J Lerner, and C Pich. The graphml file format. 2000.

[26] Bjrn Bringmann and Siegfried Nijssen. What is frequent in a single graph? In Takashi Washio, Einoshin Suzuki, KaiMing Ting, and Akihiro Inokuchi, editors, *Advances in Knowledge Discovery and Data Mining*, volume 5012 of *Lecture Notes in Computer Science*, pages 858–863. Springer Berlin Heidelberg, 2008.

[27] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer Networks*, 33(16):309 – 320, 2000.

[28] Coen Bron and Joep Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, September 1973.

[29] Iain Brown. An experimental comparison of classification techniques for imbalanced credit scoring data sets using sas. In *SAS Global Forum 2012,Data Mining and Text Analytics*, 2012.

[30] Gregory Buehrer and Kumar Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 95–106, New York, NY, USA, 2008. ACM.

[31] Richard J. Campbell and Patrick J. Flynn. A survey of free-form object representation and recognition techniques. *Comput. Vis. Image Underst.*, 81(2):166–210, February 2001.

[32] Claudio Carpineto and Giovanni Romano. Using concept lattices for text retrieval and mining. In *Formal Concept Analysis*, pages 161–179. Springer, 2005.

[33] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 38(1), June 2006.

[34] Hyun Sung Chang, Sanghoon Sull, and Sang Uk Lee. Efficient video indexing scheme for content-based retrieval. *Circuits and Systems for Video Technology, IEEE Transactions on*, 9(8):1269–1279, 1999.

[35] Gary Chartrand and Ping Zhang. *A first course in graph theory*. Courier Corporation, 2012.

[36] Chen Chen, Xifeng Yan, Feida Zhu, and Jiawei Han. gapprox: Mining frequent approximate patterns from a massive network. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 445–450. IEEE, 2007.

[37] L. Chen, J. Vogelstein, and C. Priebe. Robust Vertex Classification. *ArXiv e-prints*, November 2013.

[38] Yun Chi, Richard Muntz, Siegfried Nijssen, and Joost Kok. Frequent subtree mining–an overview. *Fundamenta Informaticae*, 21:1001–1038, 2001.

[39] Yun Chi, Yi Xia, Yirong Yang, and Richard R Muntz. Mining closed and maximal frequent subtrees from databases of labeled rooted trees. *Knowledge and Data Engineering, IEEE Transactions on*, 17(2):190–202, 2005.

[40] Yun Chi, Yirong Yang, and Richard R. Muntz. Indexing and mining free trees. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 509–512, Nov 2003.

[41] Yun Chi, Yirong Yang, and Richard R. Muntz. Hybridtreeminer: an efficient algorithm for mining frequent rooted trees and free trees using canonical forms. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 11–20, June 2004.

[42] Yun Chi, Yirong Yang, and Richard R Muntz. Mining frequent rooted trees and free trees using canonical forms. *Knowledge and Information Systems*, 2004.

[43] Yun Chi, Yirong Yang, Yi Xia, and Richard R. Muntz. Cmtreeminer: Mining both closed and maximal frequent subtrees. In *In The Eighth Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD'04*, 2003.

[44] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J.D. Ullman, and C. Yang. Finding interesting associations without support pruning. *Knowledge and Data Engineering, IEEE Transactions on*, 13(1):64–78, Jan 2001.

[45] Diane J. Cook and Lawrence B. Holder. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, pages 231–255, 1994.

[46] Diane J. Cook and Lawrence B. Holder, editors. *Mining graph data.* Hoboken, N.J. Wiley-Interscience, 2007.

[47] R. Cooley, B. Mobasher, and J. Srivastava. Grouping web page references into transactions for mining world wide web browsing patterns. In *Knowledge and Data Engineering Exchange Workshop, 1997. Proceedings*, pages 2–9, Nov 1997.

[48] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. In *In: 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen*, pages 149–159, 2001.

[49] Luigi P Cordella, Pasquale Foggia, Carlo Sansone, Francesco Tortorella, and Mario Vento. Graph matching: a fast algorithm and its evaluation. In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 2, pages 1582–1584. IEEE, 1998.

[50] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.

[51] J-P de Almeida, JG Morley, and IJ Dowman. Graph theory in higher order topological analysis of urban scenes. *Computers, environment and urban systems*, 31(4):426–440, 2007.

[52] G. Dearden, S.P. Edwardson, E. Abed, K. Bartkowiak, and K.G. Watkins. Correction of distortion and design shape in aluminium structures using laser forming. In *25th International Congress on Applications of Lasers and Electro Optics(ICALEO 2006)*, pages 813–817, 2006.

[53] Olivier Delalleau, Yoshua Bengio, and Nicolas Le Roux. Efficient non-parametric function induction in semi-supervised learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 96–103, 2005.

[54] Konstantinos K Delibasis, Aristides Kechriniotis, and Ilias Maglogiannis. A novel tool for segmenting 3d medical images based on generalized cylinders and active surfaces. *Computer methods and programs in biomedicine*, 111(1):148–165, 2013.

[55] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

[56] Minghua Deng, Ting Chen, and Fengzhu Sun. An integrated probabilistic model for functional prediction of proteins. *Journal of Computational Biology*, 11(2-3):463–475, 2004.

[57] Kwankamon Dittakan, Frans Coenen, and Rob Christley. Towards the collection of census data from satellite imagery using data mining: A study with respect to the ethiopian hinterland. In Max Bramer and Miltos Petridis, editors, *Research*

*and Development in Intelligent Systems XXIX*, pages 405–418. Springer London, 2012.

[58] Kwankamon Dittakan, Frans Coenen, and Rob Christley. Satellite image mining for census collection: A comparative study with respect to the ethiopian hinterland. In Petra Perner, editor, *Machine Learning and Data Mining in Pattern Recognition*, volume 7988 of *Lecture Notes in Computer Science*, pages 260–274. Springer Berlin Heidelberg, 2013.

[59] Kwankamon Dittakan, Frans Coenen, Rob Christley, and Maya Wardeh. Population estimation mining using satellite imagery. In Ladjel Bellatreche and MukeshK. Mohania, editors, *Data Warehousing and Knowledge Discovery*, volume 8057 of *Lecture Notes in Computer Science*, pages 285–296. Springer Berlin Heidelberg, 2013.

[60] Paul Dokas, Levent Ertoz, Vipin Kumar, Aleksandar Lazarevic, Jaideep Srivastava, and Pang-Ning Tan. Data mining for network intrusion detection. In *Proc. NSF Workshop on Next Generation Data Mining*, pages 21–30, 2002.

[61] S. Dunston, S. Ranjithan, and E. Bernold. Neural network model for the automated control of springback in rebars. *IEEE Expert: Intelligent Systems and Their Applications*, pages 45–49, 1996.

[62] S.P. Edwardson, K.G. Watkins, G. Dearden, and J. Magee. Generation of 3D shapes using a laser forming technique. In *Proceedings of ICALEO'2001*, pages 2–5, 2001.

[63] P.A. Egerton and W W. Hall. *Computer graphics: Mathematical first steps*. Simon and Schuster International, 1998.

[64] S. El-Salhi, F. Coenen, C. Dixon, and M. Khan. Identification of correlations between 3d surfaces using data mining techniques: Predicting springback in sheet metal forming. In *Proc. AI 2012*, page To appear. Springer, 2012.

[65] Subhieh El-Salhi, Frans Coenen, Clare Dixon, and M. Sulaiman Khan. Identification of correlations between 3d surfaces using data mining techniques: Predicting springback in sheet metal forming. In *Research and Development in Intelligent Systems XXIX, Incorporating Applications and Innovations in Intelligent Systems XX: Proceedings of AI-2012, The Thirty-second SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, England, UK, December 11-13, 2012*, pages 391–404, 2012.

[66] Subhieh El-Salhi, Frans Coenen, Clare Dixon, and M. Sulaiman Khan. Predicting features in complex 3d surfaces using a point series representation: A case study in sheet metal forming. In *Advanced Data Mining and Applications, 9th International*

*Conference, ADMA 2013, Hangzhou, China, December 14-16, 2013, Proceedings, Part I*, pages 505–516, 2013.

[67] Subhieh El-Salhi, Frans Coenen, Clare Dixon, and M. Sulaiman Khan. Predicting "springback" using 3d surface representation techniques: A case study in sheet metal forming. *Expert Syst. Appl.*, 42(1):79–93, 2015.

[68] Subhieh El-Salhi, Frans Coenen, Clare Dixon, and Muhammad Sulaiman Khan. Identification of correlations between 3d surfaces using data mining techniques: Predicting springback in sheet metal forming. In *Springer,Research and Development in Intelligent Systems XXIX*, pages 391–404, 2012.

[69] Subhieh El-Salhi, Frans Coenen, Clare Dixon, and Muhammad Sulaiman Khan. Predicting features in complex 3d surfaces using a point series representation: A case study in sheet metal forming. *Springer, Advanced Data Mining and Application, 9th International Conference ADMA 2013 Proc*, 8346:505–516, 2013.

[70] Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. GRAMI: frequent subgraph and pattern mining in a single large graph. *PVLDB*, 7(7):517–528, 2014.

[71] William Hill Clement Skorupka Lisa M. Talbot Jonathan Tivel Eric Bloedorn, Alan D. Christiansen. Data mining for network intrusion detection: How to get started. *Technical report, The MITRE Corporation*, 2001.

[72] Shimon Even. *Graph algorithms*. Cambridge University Press, 2011.

[73] Tom Fawcett. An introduction to {ROC} analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. {ROC} Analysis in Pattern Recognition.

[74] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[75] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, et al. Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88, 1996.

[76] Mathias Fiedler and C. Borgelt. Subgraph support in a single large graph. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, pages 399–404, Oct 2007.

[77] M. Firat, B. Kaftanoglu, and O. Eser. Sheet metal forming analyses with an emphasis on the springback deformation. *Journal of Materials Processing Technology*, 196(1-3):135–148, 2008.

[78] Scott Fortin. The graph isomorphism problem. Technical report, The University of Alberta, 1996.

[79] Eibe Frank and Ian H. Witten. Making better use of global discretization. In *Proc. of the Sixteenth International Conference on Machine Learning*, pages 115–123, 1999.

[80] Marco Frasca, Alberto Bertoni, Matteo Re, and Giorgio Valentini. A neural network algorithm for semi-supervised node label learning from unbalanced data. *Neural Networks*, 43(0):84 – 98, 2013.

[81] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[82] S. Garca, A. Fernndez, J. Luengo, and F. Herrera. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Computing*, 13(10):959–977, 2009.

[83] Salvador Garca and Francisco Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9(12):2677–2694, 2008.

[84] Salvador Garca, Daniel Molina, Manuel Lozano, and Francisco Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: acase study onthecec2005 special session onreal parameter optimization. *Journal of Heuristics*, 15(6):617–644, 2009.

[85] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[86] Maxime Gasse, Alex Aussem, and Haytham Elghazel. A hybrid algorithm for bayesian network structure learning with application to multi-label learning. *Expert Systems with Applications*, 41(15):6755–6772, 2014.

[87] Dan Gordon and R. Anthony Reynolds. Image space shading of 3-dimensional objects. *Computer Vision, Graphics, and Image Processing*, 29(3):361 – 376, 1985.

[88] Alexander Göttmann, Jö Diettrich, Georg Bergweiler, Markus Bambach, Gerhard Hirt, Peter Loosen, and Reinhart Poprawe. Laser-assisted asymmetric incremental sheet forming of titanium sheet metal parts. *Production Engineering*, 5(3):263–271, 2011.

[89] Ehud Gudes, Solomon Eyal Shimony, and Natalia Vanetik. Discovering frequent graph patterns using disjoint paths. *Knowledge and Data Engineering, IEEE Transactions on*, 18(11):1441–1456, 2006.

[90] Cigdem Gunduz, Bülent Yener, and S Humayun Gultekin. The cell graphs of cancer. *Bioinformatics*, 20(suppl 1):i145–i151, 2004.

[91] Şimşek Gürsoy and Umman Tuğba. Customer churn analysis in telecommunication sector. *Journal of the School of Business Administration, Istanbul University*, 39(1):35–49, 2010.

[92] FO Hadlock. A shortest path algorithm for grid graphs. *Networks*, 7(4):323–334, 1977.

[93] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 1–12, New York, NY, USA, 2000. ACM.

[94] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[95] J.W. Han and M. Kamber. *Data Mining: Concepts and Techniques 2nd edition.* China Machine Press, 2006.

[96] David J Hand and Robert J Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine learning*, 45(2):171–186, 2001.

[97] J A Hanley and B J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982. PMID: 7063747.

[98] Wang Hao and Stephen Duncan. Optimization of tool trajectory for incremental sheet forming using closed loop control. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, pages 779–784. IEEE, 2011.

[99] Gabor T Herman. *Fundamentals of computerized tomography: image reconstruction from projections.* Springer Science & Business Media, 2009.

[100] S. Hido and H. Kawano. Amiot: induced ordered tree mining in tree-structured databases. In *Data Mining, Fifth IEEE International Conference on*, pages 8 pp.–, Nov 2005.

[101] G. Hirt, J. Ames, M. Bambach, R. Kopp, and R. Kopp. Forming strategies and process modelling for cnc incremental sheet forming. *CIRP Annals - Manufacturing Technology*, 53(1):203–206, 2004.

[102] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. *Surface reconstruction from unorganized points*, volume 26. ACM, 1992.

[103] A. S. M. Hoque, P. K. Halder, M. S. Parvez, and T. Szecsi. Integrated manufacturing features and design-for-manufacture guidelines for reducing product cost under cad/cam environment. *Comput. Ind. Eng.*, 66(4):988–1003, December 2013.

[104] Hsun-Ping Hsieh and Cheng-Te Li. Mining temporal subgraph patterns in heterogeneous information networks. In *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, pages 282–287. IEEE, 2010.

[105] Weiming Hu, Nianhua Xie, Li Li, Xianglin Zeng, and Stephen Maybank. A survey on visual content-based video indexing and retrieval. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(6):797–819, 2011.

[106] J. Huan, W. Wang, J. Prins, and J. Yang. SPIN: mining maximal frequent subgraphs from graph databases. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 581–586, 2004.

[107] J Huan, W Wang, A Washington, J Prins, R Shah, and A Tropsha. Accurate classification of protein structural families using coherent subgraph analysis. In *Proceedings of the Ninth Pacific Symposium on Biocomputing (PSB)*, pages 411–422, 2003.

[108] Jun Huan, Wei Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 549–552, Nov 2003.

[109] Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 549–552. IEEE, 2003.

[110] Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 549–552. IEEE, 2003.

[111] Jun Huan, Wei Wang, Jan Prins, and Jiong Yang. Spin: mining maximal frequent subgraphs from graph databases. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 581–586. ACM, 2004.

[112] Jin Huang and C.X. Ling. Using auc and accuracy in evaluating learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 17(3):299–310, March 2005.

[113] Jin Huang and C.X. Ling. Using auc and accuracy in evaluating learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 17(3):299–310, March 2005.

[114] M. Inamdar, P.P. Date, K Narasimhan, S.K. Maiti, and U.P. Singh. Development of an articial neural network to predict springback in air vee bending. *International Journal of Advanced Manufacturing Technology*, 16(5):376–381, 2000.

[115] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In DjamelA. Zighed, Jan Komorowski, and Jan ytkow, editors, *Principles of Data Mining and Knowledge Discovery*, volume 1910 of *Lecture Notes in Computer Science*, pages 13–23. Springer Berlin Heidelberg, 2000.

[116] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer, 2000.

[117] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. A fast algorithm for mining frequent connected subgraphs. In *Research Report RT0448, IBM Research, Tokyo Research Laboratory*. 2002.

[118] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.

[119] Alon Itai, Christos H Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

[120] Martin Szummer Tommi Jaakkola and Martin Szummer. Partially labeled classification with markov random walks. *Advances in neural information processing systems (NIPS)*, 14:945–952, 2002.

[121] J. Jeswiet, F. Micari, G. Hirt, A. Bramley, and J. Duflou andJ. Allwood. Asymmetric single point incremental forming of sheet metal. *CIRP Annals Manufacturing Technology*, 54(2):88–114, 2005.

[122] Chuntao Jiang. *Frequent subgraph mining algorithms on weighted graphs*. PhD thesis, University of Liverpool, 2011.

[123] Ning Jin, Calvin Young, and Wei Wang. Graph classification based on pattern co-occurrence. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 573–582. ACM, 2009.

[124] Ning Jin, Calvin Young, and Wei Wang. Gaia: graph classification using evolutionary computation. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 879–890. ACM, 2010.

[125] Ulas Karaoz, TM Murali, Stan Letovsky, Yu Zheng, Chunming Ding, Charles R Cantor, and Simon Kasif. Whole-genome annotation by using evidence integration in functional-linkage networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2888–2893, 2004.

[126] Arie Kaufman. Voxels as a computational representation of geometry. *The Computational Representation of Geometry. SIGGRAPH?94 Course Notes*, 1994.

[127] ABE Kenji, Shinji Kawasoe, Hiroshi SAKAMOTO, Hiroki Arimura, and Setsuo Arikawa. Efficient substructure discovery from large semi-structured data. *IEICE TRANSACTIONS on Information and Systems*, 87(12):2754–2763, 2004.

[128] Nikhil S Ketkar, Lawrence B Holder, and Diane J Cook. Empirical comparison of graph classification algorithms. In *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*, pages 259–266. IEEE, 2009.

[129] M. Sulaiman Khan, Frans Coenen, Clare Dixon, and Subhieh El-Salhi. Finding correlations between 3-d surfaces: A study in asymmetric incremental sheet forming. *Machine Learning and Data Mining in Pattern Recognition Lecture Notes in Computer Science*, 7376:366–379, 2012.

[130] M. Sulaiman Khan, Frans Coenen, Clare Dixon, and Subhieh El-Salhi. Finding correlations between 3-d surfaces: A study in asymmetric incremental sheet forming. In *Machine Learning and Data Mining in Pattern Recognition - 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings*, pages 366–379, 2012.

[131] D.J. Kim and B.M. Kim. Application of neural network and fem for metal forming processes. *International Journal of Machine Tools and Manufacture*, 40(6):911–925, 1999.

[132] B. Kinsey, J. Cao, and S. Solla. Consistent and minimal springback using a stepped binder force trajectory and neural network control. *Journal of Engineering Materials and Technology*, 122(1113):113–118, 2000.

[133] Jon M. Kleinberg. Challenges in mining social network data: Processes, privacy, and paradoxes. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 4–5, New York, NY, USA, 2007. ACM.

[134] Leif Kobbelt and Mario Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[135] Venkat Krishnamurthy and Marc Levoy. Fitting smooth surfaces to dense polygon meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 313–324, New York, NY, USA, 1996. ACM.

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

[137] Michihiro Kuramochi and George Karypis. Grew-a scalable frequent subgraph discovery algorithm. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 439–442. IEEE, 2004.

[138] Michihiro Kuramochi and George Karypis. Finding frequent patterns in a large sparse graph*. *Data mining and knowledge discovery*, 11(3):243–271, 2005.

[139] Michihiro Kuramochi and George Karypis. Discovering frequent geometric subgraphs. *Information Systems*, 32(8):1101–1120, 2007.

[140] K Lakshmi. Frequent subgraph mining algorithms-a survey and framework for classification. 2012.

[141] J-F Lalonde, Ranjith Unnikrishnan, Nicolas Vandapel, and Martial Hebert. Scale selection for classification of point-sampled 3d surfaces. In *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*, pages 285–292. IEEE, 2005.

[142] Daniel T. Larose. *k-Nearest Neighbor Algorithm*, pages 90–106. John Wiley Sons, Inc., 2005.

[143] Stefan Lessmann, Bart Baesens, Christophe Mues, and Swantje Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Trans. Software Eng.*, 34(4):485–496, 2008.

[144] Jianzhong Li, Yong Liu, and Hong Gao. Efficient algorithms for summarizing graph patterns. *Knowledge and Data Engineering, IEEE Transactions on*, 23(9):1388–1405, Sept 2011.

[145] Xiang Li, Christophe Claramunt, and Cyril Ray. A grid graph-based model for the analysis of 2d indoor spaces. *Computers, Environment and Urban Systems*, 34(6):532 – 540, 2010. GeoVisualization and the Digital City Special issue of the International Cartographic Association Commission on GeoVisualization.

[146] Yuhua Li, Quan Lin, Gang Zhong, Dongsheng Duan, Yanan Jin, and Wei Bi. A directed labeled graph frequent pattern mining algorithm based on minimum code. In *Multimedia and Ubiquitous Engineering, 2009. MUE'09. Third International Conference on*, pages 353–359. IEEE, 2009.

[147] Charles X. Ling, Jin Huang, and Harry Zhang. Auc: a better measure than accuracy in comparing learning algorithms. In *IN PROC. OF IJCAI03*, pages 329–341. Springer, 2003.

[148] CharlesX. Ling, Jin Huang, and Harry Zhang. Auc: A better measure than accuracy in comparing learning algorithms. In Yang Xiang and Brahim Chaib-draa, editors, *Advances in Artificial Intelligence*, volume 2671 of *Lecture Notes in Computer Science*, pages 329–341. Springer Berlin Heidelberg, 2003.

[149] RA Lingbeek, W Gan, RH Wagoner, T Meinders, and J Weiher. Theoretical verification of the displacement adjustment and springforward algorithms for springback compensation. *International Journal of Material Forming*, 1(3):159–168, 2008.

[150] Yong Liu, Jianzhong Li, and Hong Gao. Jpminer: Mining frequent jump patterns from graph databases. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on*, volume 5, pages 114–118, Aug 2009.

[151] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987.

[152] Fabrizio Luccio, Antonio Mesa Enriquez, P Olivares Rieumont, and Linda Pagli. Bottom-up subtree isomorphism for unordered labeled trees. 2004.

[153] David J. Maguire. The raster gis design model: A profile of erdas. *Comput. Geosci.*, 18(4):463–470, May 1992.

[154] Jason Mah, Claire Samson, Stephen D McKinnon, and Denis Thibodeau. 3d laser imaging for surface roughness analysis. *International Journal of Rock Mechanics and Mining Sciences*, 58:111–117, 2013.

[155] K. Manabe, M. Yang, and S. Yoshihara. Artificial intelligence iidentification of process parameters and adaptive control system for deep drawing process. *Journal of Materials Processing Technology*, 80-81:421–426, 1998.

[156] Edward M Marcotte, Matteo Pellegrini[2], Michael J Thompson, Todd O Yeates, and David Eisenberg. A combined algorithm for genome-wide prediction of protein function. *Proc. Natl Acad. Sci. USA*, 93:4787–4792, 1996.

[157] Brendan D. McKay. Nauty users guide (version 1.5). In *Technical Report Technical Report, TR-CS-90-02, Department of computer Science, Australian National University*, 1990.

[158] Brendan D McKay et al. *Practical graph isomorphism*. Department of Computer Science, Vanderbilt University, 1981.

[159] T Meinders, IA Burchitz, MHA Bonte, and RA Lingbeek. Numerical product design: springback prediction, compensation and optimization. *International Journal of Machine Tools and Manufacture*, 48(5):499–514, 2008.

[160] Bruno T Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5):493–504, 1998.

[161] Charles E. Metz. Basic principles of {ROC} analysis. *Seminars in Nuclear Medicine*, 8(4):283 – 298, 1978.

[162] Manuel Montes-y Gómez, Aurelio López-López, and Alexander Gelbukh. Information retrieval with conceptual graph matching. In *Database and Expert Systems Applications*, pages 312–321. Springer, 2000.

[163] Sara Mostafavi, Debajyoti Ray, David Warde-Farley, Chris Grouios, and Quaid Morris. GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome biology*, 9 Suppl 1(Suppl 1):S4+, 2008.

[164] Elena Nabieva, Kam Jim, Amit Agarwal, Bernard Chazelle, and Mona Singh. Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *Bioinformatics*, 21(suppl 1):i302–i310, 2005.

[165] N. Narasimhan and M. Lovell. Predicting springback in sheet metal forming an explicit to implicit sequential solution procedure. *Finite Elements in Analysis and Design*, 33(1):29–42, 1999.

[166] P. Nemenyi. *Distribution-free Multiple Comparisons*. Princeton University, 1963.

[167] Siegfried Nijssen and Joost Kok. Faster association rules for multiple relations. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, pages 891–896, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[168] Siegfried Nijssen and Joost N. Kok. A quickstart in frequent structure mining can make a difference. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 647–652, New York, NY, USA, 2004. ACM.

[169] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. *Computer Graphics and Applications, IEEE*, 13(6):33–41, 1993.

[170] Edward R Omiecinski. Alternative interest measures for mining associations in databases. *Knowledge and Data Engineering, IEEE Transactions on*, 15(1):57–69, 2003.

[171] Fabrizio Luccio Antonio Mesa Enriquez Pablo, Olivares Rieumont, and Linda Pagli. Exact rooted subtree matching in sublinear time. 2001.

[172] K.K. Pathak, S. Panthi, and N. Ramakrishnan. Application of neural network in sheet metal bending process. *Defence Science Journal*, 55(2):125–131, 2005.

[173] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.

[174] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[175] Matteo Re and Giorgio Valentini. Cancer module genes ranking using kernelized score functions. *BMC bioinformatics*, 13(Suppl 14):S3, 2012.

[176] Matteo Re and Giorgio Valentini. Cancer module genes ranking using kernelized score functions. *BMC bioinformatics*, 13(Suppl 14):S3, 2012.

[177] Ronald C Read and Derek G Corneil. The graph isomorphism disease. *Journal of Graph Theory*, 1(4):339–363, 1977.

[178] A. Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973.

[179] Jordan Ringenberg, Makarand Deo, Vijay Devabhaktuni, Omer Berenfeld, Brett Snyder, Pamela Boyers, and Jeffrey Gold. Accurate reconstruction of 3d cardiac geometry from coarsely-sliced mri. *Comput. Methods Prog. Biomed.*, 113(2):483–493, February 2014.

[180] Ulrich Rückert and Stefan Kramer. Frequent free tree discovery in graph data. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 564–570. ACM, 2004.

[181] R. Rufni and J. Cao. Using neural network for springback minimization in a channel forming process. *Journal of Materials and Manufacturing*, 107(5):65–73, 1998.

[182] Hiroto Saigo, Nicole Krämer, and Koji Tsuda. Partial least squares regression for graph mining. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 578–586. ACM, 2008.

[183] Hiroto Saigo and Koji Tsuda. Graph mining in chemoinformatics. *Chemoinformatics and Advanced Machine Learning Perspectives: Complex Computational Methods and Collaborative Techniques*, pages 95–128, 2010.

[184] S. El Salhi, F. Coenen, C. Dixon, and M.S. Khan. Predicting springback using 3d surface representation techniques: A case study in sheet metal forming. *Expert Systems with Applications*, 42(1):79 – 93, 2015.

[185] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[186] Douglas C. Schmidt and Larry E. Druffel. A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. *J. ACM*, 23(3):433–445, July 1976.

[187] Falk Schreiber and Henning Schwöbbermeyer. Frequency concepts and pattern detection for the analysis of motifs in networks. In *Transactions on computational systems biology III*, pages 89–104. Springer, 2005.

[188] Paul D Seymour and Robin Thomas. Graph searching and a min-max theorem for tree-width. *Journal of Combinatorial Theory, Series B*, 58(1):22–33, 1993.

[189] Roded Sharan, Igor Ulitsky, and Ron Shamir. Network-based prediction of protein function. *Molecular systems biology*, 3(1), 2007.

[190] D. J. Sheskin. Handbook of parametric and nonparametric statistical procedures. In *Chapman Hall/CRC*, 2007.

[191] Arlei Silva, Wagner Meira, Jr., and Mohammed J. Zaki. Structural correlation pattern mining for large graphs. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, MLG '10, pages 119–126, New York, NY, USA, 2010. ACM.

[192] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 111–147, 1974.

[193] Matteo Strano. Technological representation of forming limits for negative incremental forming of thin aluminum sheets. *Journal of Manufacturing Processes*, 7(2):122 – 129, 2005.

[194] D.L. Sussman, Minh Tang, and C.E. Priebe. Consistent latent position estimation and vertex classification for random dot product graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(1):48–57, Jan 2014.

[195] Ah-Hwee Tan et al. Text mining: The state of the art and the challenges. In *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, volume 8, pages 65–70, 1999.

[196] H. Tan, T. Dillon, L. Feng, E. Chang, and F. Hadzic. X3-miner: Mining patterns from xml database. In *Data Mining VI: Data Mining, Text Mining and their Business Applications*, volume 35 of *Information and Communication Technologies*, pages 287–296, Ashurst, Southampton, UK, 2005. WIT Press.

[197] Henry Tan, Tharam S Dillon, Fedja Hadzic, Elizabeth Chang, and Ling Feng. Imb3-miner: mining induced/embedded subtrees by constraining the level of embedding. In *Advances in Knowledge Discovery and Data Mining*, pages 450–461. Springer, 2006.

[198] Henry Tan, Tharam S. Dillon, Fedja Hadzic, Ling Feng, and Elizabeth Chang. Mb3 miner: mining embedded sub-trees using tree model guided candidate generation. In *In Proc. of the 1st International Workshop on Mining Complex Data, held in conjunction with ICDM?05*, 2005.

[199] Minh Tang, Daniel L. Sussman, and Carey E. Priebe. Universally consistent vertex classification for latent positions graphs. *Ann. Statist.*, 41(3):1406–1430, 06 2013.

[200] Shirish Tatikonda, Srinivasan Parthasarathy, and Tahsin Kurc. Trips and tides: New algorithms for tree mining. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 455–464, New York, NY, USA, 2006. ACM.

[201] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(11):1115–1138, November 1991.

[202] Alexandru C Telea. *Data visualization: principles and practice*. CRC Press, 2014.

[203] Mike Thelwall, David Wilkinson, and Sukhvinder Uppal. Data mining emotion in social network communication: Gender differences in myspace. *Journal of the American Society for Information Science and Technology*, 61(1):190–199, 2010.

[204] Marisa Thoma, Hong Cheng, Arthur Gretton, Jiawei Han, Hans-Peter Kriegel, Alexander J Smola, Le Song, S Yu Philip, Xifeng Yan, and Karsten M Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *SDM*, pages 1076–1087. SIAM, 2009.

[205] Marisa Thoma, Hong Cheng, Arthur Gretton, Jiawei Han, Hans-Peter Kriegel, Alexander J Smola, Le Song, S Yu Philip, Xifeng Yan, and Karsten M Borgwardt. Near-optimal supervised feature selection among frequent subgraphs. In *SDM*, pages 1076–1087. SIAM, 2009.

[206] Lini Thomas, S Valluri, and Kamalakar Karlapalem. Isg: Itemset based subgraph mining. Technical report, Citeseer, 2009.

[207] Lini T Thomas, Satyanarayana R Valluri, and Kamalakar Karlapalem. Margin: Maximal frequent subgraph mining. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 1097–1101. IEEE, 2006.

[208] Yuanyuan Tian, Richard A Hankins, and Jignesh M Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 567–580. ACM, 2008.

[209] Gabriella Tognola, Marta Parazzini, Giorgio Pedretti, Paolo Ravazzani, Cesare Svelto, Michele Norgia, and Ferdinando Grandori. Three-dimensional reconstruction and image processing in mandibular distraction planning. *Instrumentation and Measurement, IEEE Transactions on*, 55(6):1959–1964, 2006.

[210] Amy Hin Yan Tong, Becky Drees, Giuliano Nardelli, Gary D Bader, Barbara Brannetti, Luisa Castagnoli, Marie Evangelista, Silvia Ferracuti, Bryce Nelson, Serena Paoluzi, et al. A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules. *Science*, 295(5553):321–324, 2002.

[211] Koji Tsuda. Graph classification methods in chemoinformatics. In Henry Horng-Shing Lu, Bernhard Schlkopf, and Hongyu Zhao, editors, *Handbook of Statistical Bioinformatics*, Springer Handbooks of Computational Statistics, pages 335–351. Springer Berlin Heidelberg, 2011.

[212] Koji Tsuda, Hyunjung Shin, and Bernhard Schölkopf. Fast protein classification with multiple networks. *Bioinformatics*, 21(suppl 2):59–65, 2005.

[213] Shusaku Tsumoto and Shoji Hirano. Contingency matrix theory i: Rank and statistical independence in a contigency table. In MarinaL. Gavrilova, C.J.Kenneth Tan, Yingxu Wang, Yiyu Yao, and Guoyin Wang, editors, *Transactions on Computational Science II*, volume 5150 of *Lecture Notes in Computer Science*, pages 161–179. Springer Berlin Heidelberg, 2008.

[214] Akadej Udomchaiporn, Frans Coenen, Marta Garca-Fiana, and Vanessa Sluming. 3-d mri brain scan feature classification using an oct-tree representation. In Hiroshi Motoda, Zhaohui Wu, Longbing Cao, Osmar Zaiane, Min Yao, and Wei Wang, editors, *Advanced Data Mining and Applications*, volume 8346 of *Lecture Notes in Computer Science*, pages 229–240. Springer Berlin Heidelberg, 2013.

[215] Julian R Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.

[216] Maarten Van Steen. Graph theory and complex networks.

[217] N. Vanetik, E. Gudes, and S. E. Shimony. Computing frequent graph patterns from semistructured data. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, ICDM '02, pages 458–, Washington, DC, USA, 2002. IEEE Computer Society.

[218] Natalia Vanetik, Solomon Eyal Shimony, and Ehud Gudes. Support measures for graph data*. *Data Mining and Knowledge Discovery*, 13(2):243–260, 2006.

[219] Alexei Vazquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. Global protein function prediction from protein-protein interaction networks. *Nature biotechnology*, 21(6):697–700, 2003.

[220] Chen Wang, Wei Wang, Jian Pei, Yongtai Zhu, and Baile Shi. Scalable mining of large disk-based graph databases. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–325. ACM, 2004.

[221] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, July 2003.

[222] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, July 2003.

[223] Michael A. Wesley, Tomas Lozano-Perez, Lawrence I. Lieberman, Mark A. Lavin, and David D. Grossman. A geometric modeling system for automated mechanical assembly. *IBM Journal of Research and Development*, 24(1):64–74, 1980.

[224] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[225] Marc Wrlein, Thorsten Meinl, Ingrid Fischer, and Michael Philippsen. A quantitative comparison of the subgraph miners mofa, gspan, ffsm, and gaston. In AlpioMrio Jorge, Lus Torgo, Pavel Brazdil, Rui Camacho, and Joo Gama, editors, *Knowledge Discovery in Databases: PKDD 2005*, volume 3721 of *Lecture Notes in Computer Science*, pages 392–403. Springer Berlin Heidelberg, 2005.

[226] Hui Xiong, Pang-Ning Tan, and Vipin Kumar. Hyperclique pattern discovery. *Data Mining and Knowledge Discovery*, 13(2):219–242, 2006.

[227] J. Xu, Z. Zhang, and Y. Wu. Application of data mining method to improve the accuracy of springback prediction in sheet metal forming. *Journal of Shanghai University (English Edition)*, 8(3):348–353, 2004.

[228] X. Yan and J. Han. Close Graph: mining closed frequent graph patterns. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 286–295, 2003.

[229] X. Yan and J.W. Han. gSpan: Graph-based substructure pattern mining. In *Proceedings of the 2002 International Conference on Data Mining*, pages 721–724, 2002.

[230] Xifeng Yan, Hong Cheng, Jiawei Han, and Philip S. Yu. Mining significant graph patterns by leap search. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 433–444, New York, NY, USA, 2008. ACM.

[231] Xifeng Yan, Philip S Yu, and Jiawei Han. Graph indexing: a frequent structure-based approach. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 335–346. ACM, 2004.

[232] Xifeng Yan, Philip S Yu, and Jiawei Han. Graph indexing based on discriminative frequent structure analysis. *ACM Transactions on Database Systems (TODS)*, 30(4):960–993, 2005.

[233] J.L. Yin and D.Y. Li. Knowledge discovery from finite element simulation data. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, pages 1335–1340, 2004.

[234] Lijun Yin, Xiaochen Chen, Yi Sun, T. Worm, and M. Reale. A high-resolution 3d dynamic facial expression database. In *Automatic Face Gesture Recognition, 2008. FG '08. 8th IEEE International Conference on*, pages 1–6, Sept 2008.

[235] Dong-Jin Yoo. Three-dimensional surface reconstruction of human bone using a b-spline based interpolation approach. *Computer-Aided Design*, 43(8):934–947, 2011.

[236] Mohammed J. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 71–80, New York, NY, USA, 2002. ACM.

[237] Mohammed J. Zaki. Efficiently mining frequent embedded unordered trees. *Fundam. Inf.*, 66(1-2):33–52, November 2004.

[238] S. Zhang, C. Luo, Y.H. Peng, D.Y. Li, and H.B. Yang. Study on factors affecting springback and application of data mining in springback analysis. *Journal of Shanghai Jiaotong University*, E-8(2):192–196, 2003.

[239] Shijie Zhang and Jiong Yang. Ram: Randomized approximate graph mining. In *Scientific and Statistical Database Management*, pages 187–203. Springer, 2008.

[240] Shijie Zhang, Jiong Yang, and V Cheedella. Monkey: Approximate graph mining based on spanning trees. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1247–1249. IEEE, 2007.

[241] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004.

[242] Wei Zhu, Jingyu Hou, and Yi-Ping Phoebe Chen. Semantic and layered protein function prediction from ppi networks. *Journal of theoretical biology*, 267(2):129–136, 2010.

[243] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.

[244] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Mining frequent subgraph patterns from uncertain graph data. *Knowledge and Data Engineering, IEEE Transactions on*, 22(9):1203–1218, Sept 2010.

# Appendix A

# AUC Calculation based on Mann-Whitney-Wilcoxon.

## A.1 Introduction

This Appendix presents a full example of the process of AUC calculation using the Mann-Whitney-Wilcoxon (MWW) statistical method. The Mann-Whitney-Wilcoxon (MWW) statistical method[1] was proposed in [96] to estimate AUC values based on a ranking concept as shown in Equation A.1 where c is the number of classes. In Equation A.1 $A_{i,j}$ is calculated based on the MWW values obtained with respect to class $i$ and class $j$ as shown in Equation A.2. The MWW value is calculated based on the Man-Whitney-Wilcoxon (MWW) statistic (or rank sum) [24]. The ranking concept used in this context is mainly based on signal detection theory [96], and hence the actual value of an attribute is denoted by (signal $S$) and the predicted value is denoted by (response $R$) with respect to a binary classification. Thus the MWW value of class $i$ with respect to class $j$ is calculated as shown in Equation A.3 where: (i) $n_1$ denotes the number of positive examples[2] with respect to class $i$, (ii) $n_2$ denotes the number of negative examples[3] with respect to class $j$ and (iii) $r_i$ is the sum rank for the positive examples with respect to class i. The rank for a given record of class $i$ and with respect to class $j$ is obtained from a value obtained according the combinations of $R$ and $S$.

$$AUC = \frac{2}{c(c-1)} \sum_{i<j} A_{i,j} \tag{A.1}$$

$$A_{i,j} = \frac{MWW(i|j) + MWW(j|i)}{2} \tag{A.2}$$

[1]The Wilcoxon rank sum test makes no assumption about the probability distributions and is completely based on the scores (rank) of the tuples.

[2]Positive examples are the number of records that are actually labelled with class $i$ with respect to $j$ and this means that $S = 1$.

[3]Negative examples are the number of records that are actually not labelled with class $i$ with respect to $j$ and this means that $S = 0$.

The MWW value is calculated by first drawing up a MWW ranked table comprising two columns (sometimes referred to as vectors). The first column is the response column (R), and the second the signal column (S). The rows are numbered from 1 to N where N is the number records to be considered with respect to the MWW calculation (see examples below). The ranking is as follows (in ascending order): false positives ($R_i$=0, $S_i$=1), true negatives ($R_i$=0, $S_i$=0), false negatives ($R_i$=1, $S_i$=0), true positives ($R_i$=1, $S_i$=1). The calculation is then as follows:

$$MWW(i|j) = \frac{\sum r_i - \frac{n_1(n_1+1)}{2}}{n_1 n_2} \tag{A.3}$$

Where $\sum r_i$ is the sum of the rankings of the single values (column S); n1 is the sum of the response values (1s) in the signal column and n2 is the sum of the noise values (0s) in the signal column. Responses can be signal values or noise values, 1 or 0. Signal values (1) are given a higher ranking than noise values (0).

Table A.1 presents the values (denoted as a group ID in this example) of the different potential combinations of $R$ and $S$. The group ID value associated with each record is used to order the records sequentially starting from group ID 1 to group ID 4 and consequently all the records of group ID 1 will be followed by all the records of group ID 2 and so on. Then a sequential number (rank) is given to records starting from the first record in group ID 1 to the last record in group ID 4.

An example of estimating the AUC values using the ranking process for 51 records using three classes $c_1$, $c_2$ and $c_3$ is presented below. The actual and the predicted label of the data set, that are denoted using the $S$ and $R$ variables respectively, are indicated by a value of 1 with respect to the set of classes $c_1$, $c_2$ and $c_3$ as shown in Table A.2. The MWW value of $c_1$ with respect to class $c_2$ is presented in Table A.3 and the MWW value of $c_2$ with respect to class $c_1$ is presented in Table A.4. Both tables show the associated group ID and the rank values. The numbering starts with the first record in group ID 1 to last record in group ID 4 sequentially. According to the given ranks, the last rows in Tables A.3 and A.4 indicate the number of positive and negative records, the sum of ranks and the MWW values with respect to class $c_1$ and $c_2$ respectively.

Similarly, Tables A.5 and A.6 presents the MWW value of class $c_1$ with respect to class $c_3$ and class $c_3$ with respect to class $c_1$; while Tables A.7 and A.8 presents the MWW values of class $c_2$ with respect to class $c_3$ and $c_3$ with respect to class $c_2$. Finally Table A.9 presents the calculation of the overall AUC value using Equation A.3 which is based on the MWW values calculated with respect to classes $c_1$, $c_2$ and $c_3$.

Table A.2: Example data set

| Records | $S$ | | | $R$ | | |
|---|---|---|---|---|---|---|
| | $c_1$ | $c_2$ | $c_3$ | $c_1$ | $c_2$ | $c_3$ |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 1 | 0 | 0 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 1 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 6 | 0 | 1 | 0 | 0 | 1 | 0 |
| 7 | 0 | 1 | 0 | 0 | 1 | 0 |
| 8 | 0 | 1 | 0 | 0 | 1 | 0 |
| 9 | 0 | 1 | 0 | 0 | 1 | 0 |
| 10 | 0 | 1 | 0 | 0 | 1 | 0 |
| 11 | 0 | 1 | 0 | 0 | 1 | 0 |
| 12 | 0 | 1 | 0 | 0 | 1 | 0 |
| 13 | 0 | 1 | 0 | 0 | 1 | 0 |
| 14 | 0 | 1 | 0 | 0 | 1 | 0 |
| 15 | 0 | 1 | 0 | 0 | 1 | 0 |
| 16 | 0 | 1 | 0 | 0 | 0 | 1 |
| 17 | 0 | 1 | 0 | 0 | 0 | 1 |
| 18 | 0 | 1 | 0 | 0 | 0 | 1 |
| 19 | 0 | 0 | 1 | 0 | 0 | 1 |
| 20 | 0 | 0 | 1 | 0 | 0 | 1 |
| 21 | 0 | 0 | 1 | 0 | 0 | 1 |
| 22 | 0 | 0 | 1 | 0 | 0 | 1 |
| 23 | 0 | 0 | 1 | 0 | 0 | 1 |
| 24 | 0 | 0 | 1 | 0 | 0 | 1 |
| 25 | 0 | 0 | 1 | 0 | 0 | 1 |
| 26 | 0 | 0 | 1 | 0 | 0 | 1 |
| 27 | 0 | 0 | 1 | 0 | 0 | 1 |
| 28 | 0 | 0 | 1 | 0 | 0 | 1 |
| 29 | 0 | 0 | 1 | 0 | 0 | 1 |
| 30 | 0 | 0 | 1 | 0 | 0 | 1 |
| 31 | 0 | 0 | 1 | 0 | 0 | 1 |
| 32 | 0 | 0 | 1 | 0 | 0 | 1 |
| 33 | 0 | 0 | 1 | 0 | 0 | 1 |
| 34 | 0 | 0 | 1 | 0 | 0 | 1 |
| 35 | 0 | 0 | 1 | 0 | 0 | 1 |
| 36 | 0 | 0 | 1 | 0 | 0 | 1 |
| 37 | 0 | 0 | 1 | 0 | 0 | 1 |
| 38 | 0 | 0 | 1 | 0 | 0 | 1 |
| 39 | 0 | 0 | 1 | 0 | 0 | 1 |
| 40 | 0 | 0 | 1 | 0 | 0 | 1 |
| 41 | 0 | 0 | 1 | 0 | 0 | 1 |
| 42 | 0 | 0 | 1 | 0 | 0 | 1 |
| 43 | 0 | 0 | 1 | 0 | 0 | 1 |
| 44 | 0 | 0 | 1 | 0 | 0 | 1 |
| 45 | 0 | 0 | 1 | 0 | 0 | 1 |
| 46 | 0 | 0 | 1 | 0 | 0 | 1 |
| 47 | 0 | 0 | 1 | 0 | 0 | 1 |
| 48 | 0 | 0 | 1 | 0 | 0 | 1 |
| 49 | 0 | 0 | 1 | 0 | 0 | 1 |
| 50 | 0 | 0 | 1 | 0 | 0 | 1 |
| 51 | 0 | 0 | 1 | 0 | 0 | 1 |

TABLE A.1: The values (Group ID) of different combinations of $R$ and $S$ based on Hand et al. [96].

| $R$ | $S$ | Group ID |
|---|---|---|
| 0 | 1 | 1 |
| 0 | 0 | 2 |
| 1 | 0 | 3 |
| 1 | 1 | 4 |

TABLE A.3: The MWW($c_1|c_2$) value

| $S$ | $R$ | group ID | ranks |
|---|---|---|---|
| 0 | 0 | 2 | 1 |
| 0 | 0 | 2 | 2 |
| 0 | 0 | 2 | 3 |
| 0 | 0 | 2 | 4 |
| 0 | 0 | 2 | 5 |
| 0 | 0 | 2 | 6 |
| 0 | 0 | 2 | 7 |
| 0 | 0 | 2 | 8 |
| 0 | 0 | 2 | 9 |
| 0 | 0 | 2 | 10 |
| 0 | 0 | 2 | 11 |
| 0 | 0 | 2 | 12 |
| 0 | 0 | 2 | 13 |
| 0 | 0 | 2 | 14 |
| 0 | 0 | 2 | 15 |
| 1 | 1 | 4 | 16 |
| 1 | 1 | 4 | 17 |
| 1 | 1 | 4 | 18 |
| $n_1$=3<br>$n_2$=15<br>$\sum r_i$=51 | | MWW($c_1|c_2$) $=\frac{51-6}{45}=1$ | |

TABLE A.4: The MWW($c_2|c_1$) value

| $S$ | $R$ | group ID | ranks |
|---|---|---|---|
| 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 2 |
| 1 | 0 | 1 | 3 |
| 0 | 0 | 2 | 4 |
| 0 | 0 | 2 | 5 |
| 0 | 0 | 2 | 6 |
| 1 | 1 | 4 | 7 |
| 1 | 1 | 4 | 8 |
| 1 | 1 | 4 | 9 |
| 1 | 1 | 4 | 10 |
| 1 | 1 | 4 | 11 |
| 1 | 1 | 4 | 12 |
| 1 | 1 | 4 | 13 |
| 1 | 1 | 4 | 14 |
| 1 | 1 | 4 | 15 |
| 1 | 1 | 4 | 16 |
| 1 | 1 | 4 | 17 |
| 1 | 1 | 4 | 18 |
| $n_1$=15<br>$n_2$=3<br>$\sum r_i$=156 | | MWW($c_2|c_1$) $=\frac{156-120}{45}=0.80$ | |

TABLE A.5: The MWW($c_1|c_3$) value

| $S$ | $R$ | group ID | ranks |
|---|---|---|---|
| 0 | 0 | 2 | 1 |
| 0 | 0 | 2 | 2 |
| 0 | 0 | 2 | 3 |
| 0 | 0 | 2 | 4 |
| 0 | 0 | 2 | 5 |
| 0 | 0 | 2 | 6 |
| 0 | 0 | 2 | 7 |
| 0 | 0 | 2 | 8 |
| 0 | 0 | 2 | 9 |
| 0 | 0 | 2 | 10 |
| 0 | 0 | 2 | 11 |
| 0 | 0 | 2 | 12 |
| 0 | 0 | 2 | 13 |
| 0 | 0 | 2 | 14 |
| 0 | 0 | 2 | 15 |
| 0 | 0 | 2 | 16 |
| 0 | 0 | 2 | 17 |
| 0 | 0 | 2 | 18 |
| 0 | 0 | 2 | 19 |
| 0 | 0 | 2 | 20 |
| 0 | 0 | 2 | 21 |
| 0 | 0 | 2 | 22 |
| 0 | 0 | 2 | 23 |
| 0 | 0 | 2 | 24 |
| 0 | 0 | 2 | 25 |
| 0 | 0 | 2 | 26 |
| 0 | 0 | 2 | 27 |
| 0 | 0 | 2 | 28 |
| 0 | 0 | 2 | 29 |
| 0 | 0 | 2 | 30 |
| 0 | 0 | 2 | 31 |
| 0 | 0 | 2 | 32 |
| 0 | 0 | 2 | 33 |
| 1 | 1 | 4 | 34 |
| 1 | 1 | 4 | 35 |
| 1 | 1 | 4 | 36 |
| $n_1$=3<br>$n_2$=33<br>$\sum r_i$=105 | | MWW($c_1|c_3$) $=\frac{105-6}{99}=1$ | |

TABLE A.6: The MWW($c_3|c_1$) value

| $S$ | $R$ | group ID | ranks |
|---|---|---|---|
| 0 | 0 | 2 | 1 |
| 0 | 0 | 2 | 2 |
| 0 | 0 | 2 | 3 |
| 1 | 1 | 4 | 4 |
| 1 | 1 | 4 | 5 |
| 1 | 1 | 4 | 6 |
| 1 | 1 | 4 | 7 |
| 1 | 1 | 4 | 8 |
| 1 | 1 | 4 | 9 |
| 1 | 1 | 4 | 10 |
| 1 | 1 | 4 | 11 |
| 1 | 1 | 4 | 12 |
| 1 | 1 | 4 | 13 |
| 1 | 1 | 4 | 14 |
| 1 | 1 | 4 | 15 |
| 1 | 1 | 4 | 16 |
| 1 | 1 | 4 | 17 |
| 1 | 1 | 4 | 18 |
| 1 | 1 | 4 | 19 |
| 1 | 1 | 4 | 20 |
| 1 | 1 | 4 | 21 |
| 1 | 1 | 4 | 22 |
| 1 | 1 | 4 | 23 |
| 1 | 1 | 4 | 24 |
| 1 | 1 | 4 | 25 |
| 1 | 1 | 4 | 26 |
| 1 | 1 | 4 | 27 |
| 1 | 1 | 4 | 28 |
| 1 | 1 | 4 | 29 |
| 1 | 1 | 4 | 30 |
| 1 | 1 | 4 | 31 |
| 1 | 1 | 4 | 32 |
| 1 | 1 | 4 | 33 |
| 1 | 1 | 4 | 34 |
| 1 | 1 | 4 | 35 |
| 1 | 1 | 4 | 36 |
| $n_1$=33<br>$n_2$=3<br>$\sum r_i$=660 | | MWW($c_3|c_1$) $=\frac{660-561}{99}=1$ | |

TABLE A.7: The MWW($c_2|c_3$) value

| $S$ | $R$ | group ID | ranks |
|---|---|---|---|
| 1 | 0 | 1 | 47 |
| 1 | 0 | 1 | 48 |
| 1 | 0 | 2 | 1 |
| 1 | 0 | 2 | 2 |
| 1 | 0 | 2 | 3 |
| 0 | 0 | 2 | 4 |
| 0 | 0 | 2 | 5 |
| 0 | 0 | 2 | 6 |
| 0 | 0 | 2 | 7 |
| 0 | 0 | 2 | 8 |
| 0 | 0 | 2 | 9 |
| 0 | 0 | 2 | 10 |
| 0 | 0 | 2 | 11 |
| 0 | 0 | 2 | 12 |
| 0 | 0 | 2 | 13 |
| 0 | 0 | 2 | 14 |
| 0 | 0 | 2 | 15 |
| 0 | 0 | 2 | 16 |
| 0 | 0 | 2 | 17 |
| 0 | 0 | 2 | 18 |
| 0 | 0 | 2 | 19 |
| 0 | 0 | 2 | 20 |
| 0 | 0 | 2 | 21 |
| 0 | 0 | 2 | 22 |
| 0 | 0 | 2 | 23 |
| 0 | 0 | 2 | 24 |
| 0 | 0 | 2 | 25 |
| 0 | 0 | 2 | 26 |
| 0 | 0 | 2 | 27 |
| 0 | 0 | 2 | 28 |
| 0 | 0 | 2 | 29 |
| 0 | 0 | 2 | 30 |
| 0 | 0 | 2 | 31 |
| 0 | 0 | 2 | 32 |
| 0 | 0 | 2 | 33 |
| 0 | 0 | 2 | 34 |
| 0 | 0 | 2 | 35 |
| 0 | 0 | 2 | 36 |
| 1 | 1 | 4 | 37 |
| 1 | 1 | 4 | 38 |
| 1 | 1 | 4 | 39 |
| 1 | 1 | 4 | 40 |
| 1 | 1 | 4 | 41 |
| 1 | 1 | 4 | 42 |
| 1 | 1 | 4 | 43 |
| 1 | 1 | 4 | 44 |
| 1 | 1 | 4 | 45 |
| 1 | 1 | 4 | 46 |
| $n_1$=15<br>$n_2$=33<br>$\sum r_i$=516 | | MWW($c_2|c_3$) $=\frac{516-120}{495}=0.8$ | |

TABLE A.8: The MWW($c_3|c_2$) value

| $S$ | $R$ | group ID | ranks |
|---|---|---|---|
| 0 | 0 | 2 | 1 |
| 0 | 0 | 2 | 2 |
| 0 | 0 | 2 | 3 |
| 0 | 0 | 2 | 4 |
| 0 | 0 | 2 | 5 |
| 0 | 0 | 2 | 6 |
| 0 | 0 | 2 | 7 |
| 0 | 0 | 2 | 8 |
| 0 | 0 | 2 | 9 |
| 0 | 0 | 2 | 10 |
| 0 | 0 | 2 | 11 |
| 0 | 0 | 2 | 12 |
| 0 | 1 | 3 | 13 |
| 0 | 1 | 3 | 14 |
| 0 | 1 | 3 | 15 |
| 1 | 1 | 4 | 16 |
| 1 | 1 | 4 | 17 |
| 1 | 1 | 4 | 18 |
| 1 | 1 | 4 | 19 |
| 1 | 1 | 4 | 20 |
| 1 | 1 | 4 | 21 |
| 1 | 1 | 4 | 22 |
| 1 | 1 | 4 | 23 |
| 1 | 1 | 4 | 24 |
| 1 | 1 | 4 | 25 |
| 1 | 1 | 4 | 26 |
| 1 | 1 | 4 | 27 |
| 1 | 1 | 4 | 28 |
| 1 | 1 | 4 | 29 |
| 1 | 1 | 4 | 30 |
| 1 | 1 | 4 | 31 |
| 1 | 1 | 4 | 32 |
| 1 | 1 | 4 | 33 |
| 1 | 1 | 4 | 34 |
| 1 | 1 | 4 | 35 |
| 1 | 1 | 4 | 36 |
| 1 | 1 | 4 | 37 |
| 1 | 1 | 4 | 38 |
| 1 | 1 | 4 | 39 |
| 1 | 1 | 4 | 40 |
| 1 | 1 | 4 | 41 |
| 1 | 1 | 4 | 42 |
| 1 | 1 | 4 | 43 |
| 1 | 1 | 4 | 44 |
| 1 | 1 | 4 | 45 |
| 1 | 1 | 4 | 46 |
| 1 | 1 | 4 | 47 |
| 1 | 1 | 4 | 48 |
| $n_1$=33<br>$n_2$=15<br>$\sum r_i$=1056 | | MWW($c_3|c_2$) $=\frac{1056-561}{495}=1$ | |

TABLE A.9: The overall AUC value for the given data set

| MWW(i,j) | MWW(j,i) | A(i,j) |
|---|---|---|
| MWW$(c_1, c_2)$=1 | MWW$(c_2, c_1)$=0.8 | $A(c_1\|c_2) = \frac{1+0.8}{2} = 0.9$ |
| MWW$(c_1, c_3)$=1 | MWW$(c_3, c_1)$=1 | $A(c_1\|c_3) = \frac{1+1}{2} = 1$ |
| MWW$(c_2, c_3)$=0.8 | MWW$(c_3, c_2)$=1 | $A(c_2\|c_3) = \frac{0.8+1}{2} = 0.9$ |
| AUC=$\frac{2\times 2.8}{3\times 2} = 0.93$ | | |

# Appendix B

# Graph File Format and Raw Data Format

For the work described in this thesis the GraphML format was employed as the standard format for describing grid graphs. The raw data for the AISF sheet metal forming application was provided in a "point cloud" format in terms of a set of "X-Y-Z" coordinates which was converted into grids and then into grid graphs. The raw data sets for the satellite image interpretation application were provided directly in the form a grids from which grid graphs could be generated. In both cases, prior to further processing, the graphs were presented in GraphML format [25]. For completeness this format is presented in this appendix.

GraphML is an XML based file format originally proposed by the graph analysis community so as to provide a suitable file exchange format for graph data. Figure B.2 presents an encoding of the graph presented in Figure B.1 using GraphML. With respect to Figure B.2, lines 1 to 5 define a common header, line 7 to 15 define graph vertex information, and line 16 to 25 define graph edge information. In GraphML there is no required ordering for the node and edge elements. More advanced features offered by the GraphML format can be found in the GraphML Primer available from the GraphML website[1].

[1] http://graphml.graphdrawing.org/primer/graphml-primer.html

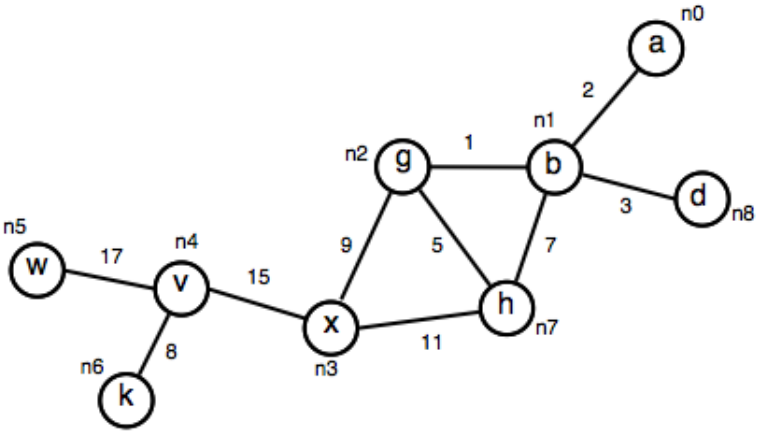



FIGURE B.1: An example graph [122].

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<graph id="G0" edgedefault="undirected">
<node id="n0"/><data>a</data></node>
      <node id="n1"/><data>b</data></node>
      <node id="n2"/><data>g</data></node>
     <node id="n3"/><data>x</data></node>
     <node id="n4"/><data>v</data></node>
     <node id="n5"/><data>w</data></node>
     <node id="n6"/><data>k</data></node>
     <node id="n7"/><data>h</data></node>
     <node id="n8"/><data>d</data></node>
     <edge id="e0" source="n0" target="n1"><data>2</data></edge>
     <edge id="e1" source="n1" target="n2"><data>1</data></edge>
     <edge id="e2" source="n1" target="n7"><data>7</data></edge>
     <edge id="e3" source="n1" target="n8"><data>3</data></edge>
     <edge id="e4" source="n2" target="n3"><data>9</data></edge>
     <edge id="e5" source="n2" target="n7"><data>5</data></edge>
     <edge id="e6" source="n3" target="n4"><data>15</data></edge>
     <edge id="e7" source="n3" target="n7"><data>11</data></edge>
     <edge id="e8" source="n4" target="n5"><data>17</data></edge>
     <edge id="e9" source="n4" target="n6"><data>8</data></edge>
 </graph>
 </graphml>
```

FIGURE B.2: GraphML encoding for the graph given in Figure B.1

# Appendix C

# Additional Experimental Results

## C.1 Introduction

In this appendix some additional experimental results to those given in Chapter 7, in the context of VULS based vertex classification, are presented. More specifically results obtained using a grid size $d = 23$ (mm) for the sheet metal forming AISF data is presented. Recall that in Chapter 7 only results using $d = 28$ (mm) were considered; this was because of of space restrictions within the main body of this thesis. For completeness these additional results are thus presented here. The objectives of the evaluation were the same as before, namely:

1. To compare the operation of the proposed VULS mining algorithms, using a range of $max$ values, in terms of coverage, number of identified VULS and runtime.
2. To compare the distinction between the usage of grid graphs and cross grid graphs, and the usage of directed and undirected graphs.
3. To investigate the effect of using different values for $|L_V|$.
4. To investigate the effect of using different values for $|L_E|$.
5. To compare the effectiveness of the identified VULS, with respect to vertex classification, and with respect to more standard approaches (J48 and Naive Bayes).
6. To investigate the statistical significance of the outcome.

This appendix is organised in the same manner as Chapter 7 since the objectives above are the same. Each of these objectives is considered in a separate section below (Sections C.2 to C.7).

## C.2 Comparison of VULS Mining Algorithms Using a Range of $max$ Values (Objective 1)

In this section the additional results obtained, using $d = 23$ (mm), to compare the operation of the four proposed VULS mining algorithms, are presented. As previously,

the results are considered in terms of coverage, number of identified VULS and run time. The results are listed in Tables C.1 to C.3, corresponding to $max$ parameter settings of 4, 5 and 6 respectively. As before $|L_V|$ was set to 2 and $|L_E|$ was set to 8. All the graphs considered were grid graphs (Degree=4) and featured directed Edges.

TABLE C.1: Comparison of VULS Mining Algorithms Using $max = 4$ (Objective 1).

| Graph | Comp.VULS | | | Min. VULS | | | Freq. VULS | | | Min. freq. VULS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time |
| GS | 100.00 | 192.00 | 0.39 | 100.00 | 52.00 | 0.33 | 100.00 | 173.00 | 0.31 | 100.00 | 48.00 | 0.39 |
| GT | 87.50 | 1044.00 | 0.65 | 84.38 | 53.00 | 0.47 | 87.50 | 1009.00 | 0.81 | 82.81 | 38.00 | 0.49 |
| MS | 100.00 | 79.00 | 0.30 | 96.88 | 18.00 | 0.40 | 100.00 | 50.00 | 0.28 | 93.75 | 11.00 | 0.74 |
| MT | 79.69 | 455.00 | 0.53 | 78.13 | 40.00 | 0.41 | 79.69 | 278.00 | 0.62 | 73.44 | 31.00 | 0.38 |
| Average | 91.80 | 442.50 | 0.47 | 89.85 | 40.75 | 0.40 | 91.80 | 377.50 | 0.51 | 87.50 | 32.00 | 0.50 |
| SD | 9.99 | 430.83 | 0.15 | 10.32 | 16.28 | 0.06 | 9.99 | 431.19 | 0.25 | 11.76 | 15.64 | 0.17 |

TABLE C.2: Comparison of VULS Mining Algorithms Using $max = 5$ (Objective 1).

| Graph | Comp.VULS | | | Min. VULS | | | Freq. VULS | | | Min. freq. VULS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time |
| GS | 100.00 | 192.00 | 0.28 | 100.00 | 52.00 | 0.42 | 100.00 | 173.00 | 0.32 | 100.00 | 48.00 | 0.47 |
| GT | 87.50 | 3621 | 1.55 | 84.38 | 65.00 | 0.45 | 87.50 | 3586.00 | 1.22 | 82.81 | 50.00 | 0.55 |
| MS | 100.00 | 79.00 | 0.29 | 96.88 | 37.00 | 0.50 | 100.00 | 50.00 | 0.28 | 93.75 | 30.00 | 0.79 |
| MT | 79.69 | 1475.00 | 1.24 | 78.13 | 78.00 | 0.44 | 79.69 | 844.00 | 0.98 | 76.56 | 53.00 | 0.50 |
| Average | 91.80 | 1341.75 | 0.84 | 89.85 | 58.00 | 0.45 | 91.80 | 1163.25 | 0.70 | 88.28 | 45.25 | 0.58 |
| SD | 9.99 | 1646.13 | 0.65 | 10.32 | 17.57 | 0.03 | 9.99 | 1652.43 | 0.47 | 10.56 | 10.37 | 0.15 |

TABLE C.3: Comparison of VULS Mining Algorithms Using $max = 6$ (Objective 1).

| Graph | Comp.VULS | | | Min. VULS | | | Freq. VULS | | | Min. freq. VULS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time | Cov. | # VULS | Time |
| GS | 100.00 | 192.00 | 0.34 | 100.00 | 52.00 | 0.23 | 100.00 | 173.00 | 0.31 | 100.00 | 48.00 | 0.53 |
| GT | 87.50 | 11541.00 | 4.30 | 87.50 | 187.00 | 0.73 | 87.50 | 11506.00 | 1.77 | 87.50 | 172 | 1.30 |
| MS | 100.00 | 79.00 | 0.26 | 96.88 | 50.00 | 0.84 | 100.00 | 50.00 | 0.32 | 93.75 | 43.00 | 1.21 |
| MT | 92.19 | 4648.00 | 2.38 | 78.13 | 105.00 | 0.67 | 92.19 | 2488.00 | 1.62 | 76.56 | 80.00 | 0.85 |
| Average | 94.92 | 4115.00 | 1.82 | 90.63 | 98.50 | 0.62 | 94.92 | 3554.25 | 1.01 | 89.45 | 85.75 | 0.97 |
| SD | 6.17 | 5388.53 | 1.92 | 9.88 | 64.26 | 0.27 | 6.17 | 5418.48 | 0.80 | 10.00 | 59.79 | 0.35 |

From Tables C.1 to C.3 the following can be noted with respect to coverage, number of VULS identified and run time:

- **Coverage.**

  1. When finding all VULS, minimal VULS, frequent VULS and minimal frequent VULS, 100% coverage can be obtained in some cases regardless of the $max$ value used.
  2. On average, when finding minimal VULS coverage was always worse than when finding frequent VULS.
  3. The worst recorded coverage values were obtained when finding minimal frequent VULS.

4. The relationship between the different VULS algorithms in terms of coverage fits with the more general relationship between the four different categories of VULS illustrated previously in Figure 2.2 of chapter 4.
5. As the *max* parameter is increased from 4 to 6 we can anticipate that more VULS will be identified and hence coverage would be expected to increase (as indicated by the results).

- **Number of VULS.**

1. Many more VULS are discovered when mining for all VULS than when mining for any of the other forms of VULS considered (minimal, frequent, minimal frequent).
2. Generally speaking, with respect to the average number of VULS identified, we can expect: (i) VULS $\geqslant$ minimal VULS $\geqslant$ minimal frequent VULS; and (ii) VULS $\geqslant$ frequent VULS $\geqslant$ minimal frequent VULS.
3. Not withstanding point 2 above the recorded results again confirm the relationship between the four different categories of VULS illustrated previously in Figure 2.2 of chapter 4.
4. With respect to the *max* parameter, as this was increased from 4 to 6, the number of identified VULS increased (as expected).

- **Run time.**

1. As the *max* parameter was increased from 4 to 6, the required run time for identifying VULS (with respect to all four categories) also increased (as was to be expected).
2. The frequent VULS mining algorithm required more run time than the Complete VULS and Minimal VULS algorithms because of the additional candidate graph counting (isomorphism testing) that had to be conducted.

Comparing these results with the results presented in Section 7.2 of Chapter 7, it can be seen that as grid size $d$ decreased from 28 (mm) to 23 (mm), more VULS (regardless of the VULS form) were generated, but coverage did not increase as expected, and more run time was consumed.

## C.3 Comparison Between Usage of Grid Graphs and Cross Grid Graphs, and Directed and Undirected Graphs (Objective 2)

In this section results are presented from experiments, using $d = 23$ (mm), comparing the operation of VULS based vertex classification with respect to the four different

types of grid graph identified in this thesis: (i) directed grid graphs, (ii) undirected grid graphs, (iii) directed cross-grid graphs, (iv) undirected cross-grid graphs. Recall that the distinction between a grid graph and a cross grid graph is that the first has $degree = 4$ and the second $degree = 8$. As before the experiments were conducted using AISF grid graph pairings. Note also that $|L_V| = 2$, $|L_E| = 8$ and $max = 4$ was used. The results are presented in Table C.4.

TABLE C.4: Classification Effectiveness with Respect to graph types (Objective 2).

| Graph Category | Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | | J48 | | Naive Bayes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| directed grid | GS | 0.75 | 0.77 | 0.59 | 0.63 | 0.75 | 0.77 | 0.59 | 0.63 | 0.83 | 0.77 | 0.78 | 0.74 |
| | GT | 0.72 | 0.53 | 0.77 | 0.62 | 0.72 | 0.53 | 0.77 | 0.62 | 0.70 | 0.50 | 0.73 | 0.90 |
| | MS | 0.81 | 0.83 | 0.58 | 0.63 | 0.78 | 0.81 | 0.56 | 0.61 | 0.83 | 0.92 | 0.80 | 0.99 |
| | MT | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.89 | 0.50 | 0.89 | 0.72 |
| Average | | 0.79 | 0.67 | 0.71 | 0.61 | 0.78 | 0.67 | 0.70 | 0.60 | 0.81 | 0.67 | 0.80 | 0.84 |
| SD | | 0.07 | 0.15 | 0.15 | 0.04 | 0.07 | 0.15 | 0.15 | 0.04 | 0.08 | 0.21 | 0.07 | 0.13 |
| undirected grid | GS | 0.52 | 0.49 | 0.45 | 0.41 | 0.50 | 0.47 | 0.55 | 0.50 | 0.83 | 0.77 | 0.78 | 0.74 |
| | GT | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.73 | 0.90 |
| | MS | 0.91 | 0.92 | 0.75 | 0.78 | 0.89 | 0.90 | 0.72 | 0.75 | 0.83 | 0.92 | 0.80 | 0.99 |
| | MT | 0.89 | 0.50 | 0.83 | 0.59 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.72 |
| Average | | 0.76 | 0.60 | 0.68 | 0.57 | 0.75 | 0.59 | 0.72 | 0.56 | 0.81 | 0.67 | 0.80 | 0.84 |
| SD | | 0.18 | 0.21 | 0.16 | 0.16 | 0.19 | 0.21 | 0.14 | 0.13 | 0.08 | 0.21 | 0.07 | 0.13 |
| directed cross-grid | GS | 0.45 | 0.42 | 0.50 | 0.47 | 0.56 | 0.54 | 0.50 | 0.47 | 0.83 | 0.77 | 0.78 | 0.74 |
| | GT | 0.70 | 0.50 | 0.72 | 0.53 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.73 | 0.90 |
| | MS | 0.94 | 0.94 | 0.77 | 0.79 | 0.95 | 0.95 | 0.84 | 0.86 | 0.83 | 0.92 | 0.80 | 0.99 |
| | MT | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.72 |
| Average | | 0.75 | 0.59 | 0.72 | 0.57 | 0.78 | 0.62 | 0.73 | 0.58 | 0.81 | 0.67 | 0.80 | 0.84 |
| SD | | 0.22 | 0.24 | 0.16 | 0.15 | 0.18 | 0.22 | 0.17 | 0.19 | 0.08 | 0.21 | 0.07 | 0.13 |
| undirected cross-grid | GS | 0.50 | 0.47 | 0.47 | 0.44 | 0.55 | 0.52 | 0.48 | 0.45 | 0.83 | 0.77 | 0.78 | 0.74 |
| | GT | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.73 | 0.90 |
| | MS | 0.94 | 0.94 | 0.86 | 0.87 | 0.92 | 0.93 | 0.66 | 0.69 | 0.83 | 0.92 | 0.80 | 0.99 |
| | MT | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.72 |
| Average | | 0.76 | 0.60 | 0.73 | 0.58 | 0.77 | 0.61 | 0.68 | 0.54 | 0.81 | 0.67 | 0.80 | 0.84 |
| SD | | 0.20 | 0.23 | 0.19 | 0.20 | 0.17 | 0.21 | 0.17 | 0.11 | 0.08 | 0.21 | 0.07 | 0.13 |

From Table C.4 it cam be seen that, regardless of the form of VULS used (Complete VULS, minimal VULS, frequent VULS or minimal frequent VULS) more effective results were obtained using directed graphs than undirected graphs, and using grid graphs than cross-grid graphs in most cases. More specifically, in most cases VULS classification performance (in terms of accuracy and AUC) featured the following trends: (i) directed grid > undirected grid, (ii) directed cross-grid > undirected cross-grid and (iii) directed grid >directed cross-grid. This trend is consistent with that found in section 7.5 of Chapter 7 where grid size $d = 28$ (mm).

## C.4 Effect of $|L_V|$ on Classification Effectiveness (Objective 3)

The third parameter considered in this appendix is $|L_V|$. The effect on classification when $|L_V| = 2$ and $|L_V| = 3$ was considered. For the experiments $|L_E| = 8$ and

$max = 4$ were used. As before all graphs were directed. The results are presented in Table C.5. From the Table, as expected, better accuracy and AUC can be achieved when the number of vertex labels is small ($|L_V| = 2$). As the number of vertex labels increases to 3, the graph labelling becomes more diverse; as a result, the classification performance tends to deteriorate. This trend is consistent with that found in section 7.6 of Chapter 7 where experiments using a grid size $d = 28$ (mm) were considered.

TABLE C.5: Classification Effectiveness with Respect to $|L_V|$ (Objective 3).

| $\|L_V\|$ | Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | | J48 | | Naive Bayes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| 2 | GS | 0.75 | 0.77 | 0.59 | 0.63 | 0.75 | 0.77 | 0.59 | 0.63 | 0.83 | 0.77 | 0.78 | 0.74 |
| | GT | 0.72 | 0.53 | 0.77 | 0.62 | 0.72 | 0.53 | 0.77 | 0.62 | 0.70 | 0.50 | 0.73 | 0.90 |
| | MS | 0.81 | 0.83 | 0.58 | 0.63 | 0.78 | 0.81 | 0.56 | 0.61 | 0.83 | 0.92 | 0.80 | 0.99 |
| | MT | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.89 | 0.50 | 0.89 | 0.72 |
| Average | | 0.79 | 0.67 | 0.71 | 0.61 | 0.78 | 0.67 | 0.70 | 0.60 | 0.81 | 0.67 | 0.80 | 0.84 |
| SD | | 0.07 | 0.15 | 0.15 | 0.04 | 0.07 | 0.15 | 0.15 | 0.04 | 0.08 | 0.21 | 0.07 | 0.13 |
| 3 | GS | 0.47 | 0.37 | 0.38 | 0.32 | 0.47 | 0.37 | 0.38 | 0.32 | 0.58 | 0.74 | 0.69 | 0.73 |
| | GT | 0.61 | 0.40 | 0.58 | 0.33 | 0.61 | 0.40 | 0.58 | 0.33 | 0.58 | 0.50 | 0.67 | 0.89 |
| | MS | 0.58 | 0.63 | 0.28 | 0.36 | 0.58 | 0.63 | 0.27 | 0.35 | 0.58 | 0.75 | 0.50 | 0.81 |
| | MT | 0.73 | 0.38 | 0.72 | 0.37 | 0.73 | 0.38 | 0.70 | 0.35 | 0.70 | 0.50 | 0.67 | 0.69 |
| Average | | 0.60 | 0.45 | 0.49 | 0.35 | 0.60 | 0.45 | 0.48 | 0.34 | 0.61 | 0.62 | 0.63 | 0.78 |
| SD | | 0.11 | 0.12 | 0.20 | 0.02 | 0.11 | 0.12 | 0.19 | 0.02 | 0.06 | 0.14 | 0.09 | 0.09 |

## C.5 Effect of $|L_E|$ on Classification Effectiveness (Objective 4)

In this section further results concerning classification effectiveness, obtained using $d = 23$ (mm), and a range of values for $L_E$ from 2 to 8, increasing in steps of 2, are presented in terms of accuracy and AUC. Note that the VULS classification was again conducted using the AISF data pairings whereby one graph was used for training purposes, while the other was used for testing purposes. As before $|L_V|$ was set to 2 and the $max$ was set to 4; all the graphs considered were grid graphs ($degree = 4$) and featured directed edges only. The results are presented in the Table C.6

TABLE C.6: Classification Effectiveness with Respect to $|L_E|$ (Objective 4).

| $\|L_E\|$ | Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min.freq VULS | | J48 | | Naive Bayes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| 2 | GS | 0.53 | 0.50 | 0.58 | 0.54 | 0.53 | 0.50 | 0.55 | 0.50 | 0.69 | 0.65 | 0.66 | 0.74 |
| | GT | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.65 |
| | MS | 0.44 | 0.47 | 0.44 | 0.50 | 0.44 | 0.50 | 0.44 | 0.50 | 0.67 | 0.82 | 0.67 | 0.87 |
| | MT | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.52 |
| Average | | 0.64 | 0.49 | 0.65 | 0.51 | 0.64 | 0.50 | 0.65 | 0.50 | 0.74 | 0.62 | 0.73 | 0.70 |
| SD | | 0.20 | 0.02 | 0.19 | 0.02 | 0.20 | 0 | 0.20 | 0 | 0.10 | 0.15 | 0.11 | 0.15 |
| 4 | GS | 0.56 | 0.53 | 0.42 | 0.39 | 0.56 | 0.53 | 0.42 | 0.39 | 0.81 | 0.77 | 0.75 | 0.78 |
| | GT | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.70 | 0.50 | 0.72 | 0.85 |
| | MS | 0.70 | 0.74 | 0.69 | 0.72 | 0.67 | 0.71 | 0.55 | 0.60 | 0.86 | 0.91 | 0.75 | 0.96 |
| | MT | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.50 | 0.89 | 0.67 |
| Average | | 0.71 | 0.57 | 0.68 | 0.53 | 0.71 | 0.56 | 0.64 | 0.50 | 0.82 | 0.67 | 0.78 | 0.82 |
| SD | | 0.14 | 0.12 | 0.19 | 0.14 | 0.14 | 0.10 | 0.20 | 0.09 | 0.08 | 0.20 | 0.08 | 0.12 |
| 6 | GS | 0.53 | 0.49 | 0.47 | 0.43 | 0.53 | 0.49 | 0.47 | 0.43 | 0.67 | 0.74 | 0.77 | 0.73 |
| | GT | 0.70 | 0.50 | 0.69 | 0.55 | 0.70 | 0.50 | 0.69 | 0.55 | 0.70 | 0.50 | 0.72 | 0.85 |
| | MS | 0.80 | 0.82 | 0.73 | 0.76 | 0.80 | 0.82 | 0.70 | 0.74 | 0.86 | 0.92 | 0.81 | 0.98 |
| | MT | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.89 | 0.56 | 0.89 | 0.50 | 0.89 | 0.72 |
| Average | | 0.73 | 0.59 | 0.69 | 0.57 | 0.73 | 0.59 | 0.69 | 0.57 | 0.78 | 0.67 | 0.80 | 0.82 |
| SD | | 0.15 | 0.16 | 0.17 | 0.14 | 0.15 | 0.16 | 0.17 | 0.13 | 0.11 | 0.20 | 0.07 | 0.12 |
| 8 | GS | 0.75 | 0.77 | 0.59 | 0.63 | 0.75 | 0.77 | 0.59 | 0.63 | 0.83 | 0.77 | 0.78 | 0.74 |
| | GT | 0.72 | 0.53 | 0.77 | 0.62 | 0.72 | 0.53 | 0.77 | 0.62 | 0.70 | 0.50 | 0.73 | 0.90 |
| | MS | 0.81 | 0.83 | 0.58 | 0.63 | 0.78 | 0.81 | 0.56 | 0.61 | 0.83 | 0.92 | 0.80 | 0.99 |
| | MT | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.89 | 0.50 | 0.89 | 0.72 |
| Average | | 0.79 | 0.67 | 0.71 | 0.61 | 0.78 | 0.67 | 0.70 | 0.60 | 0.81 | 0.67 | 0.80 | 0.84 |
| SD | | 0.07 | 0.15 | 0.15 | 0.04 | 0.07 | 0.15 | 0.15 | 0.04 | 0.08 | 0.21 | 0.07 | 0.13 |

From Table C.6 it can be seen that the average accuracy and AUC tends to rise as $|L_E|$ is increased from 2 to 8 with respect to each of the VULS form considered (Complete VULS, minimal VULS, frequent VULS, minimal frequent VULS). This trend is consistent with that found in section 7.4 of Chapter 7.

## C.6 Comparison of VULS Vertex Classification Effectiveness (Objective 5)

This section reports on the comparative evaluation conducted with respect to VULS vertex classification when $d = 23$ (mm). As before the comparison includes results obtained when using J48 and Naive Bayes. For the experiments the following parameter settings were used: $|L_E| = 8$ and $|L_V| = 2$. All the graphs considered were grid graphs ($degree = 4$) and featured directed Edges. The results are presented in Table C.7. From the table it can be seen that out of the VULS algorithms considered, using Complete VULS and frequent VULS produced comparable results as obtained using J48, in terms of average AUC (0.67). Naive Bayes produced the best vertex classification performance in terms of AUC (0.84).

TABLE C.7: VULS Classification Comparison where $|L_V| = 2$ (Objective 5).

| Graph | Comp.VULS | | Min. VULS | | Freq. VULS | | Min. freq. VULS | | J48 | | Naive Bayes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC | Acc. | AUC |
| GS | 0.75 | 0.77 | 0.59 | 0.63 | 0.75 | 0.77 | 0.59 | 0.63 | 0.83 | 0.77 | 0.78 | 0.74 |
| GT | 0.72 | 0.53 | 0.77 | 0.62 | 0.72 | 0.53 | 0.77 | 0.62 | 0.70 | 0.50 | 0.73 | 0.90 |
| MS | 0.81 | 0.83 | 0.58 | 0.63 | 0.78 | 0.81 | 0.56 | 0.61 | 0.83 | 0.92 | 0.80 | 0.99 |
| MT | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.88 | 0.55 | 0.89 | 0.50 | 0.89 | 0.72 |
| Average | 0.79 | 0.67 | 0.71 | 0.61 | 0.78 | 0.67 | 0.70 | 0.60 | 0.81 | 0.67 | 0.80 | 0.84 |
| SD | 0.07 | 0.15 | 0.15 | 0.04 | 0.07 | 0.15 | 0.15 | 0.04 | 0.08 | 0.21 | 0.07 | 0.13 |

## C.7 Statistical Comparison of the Proposed VULS Approaches (Objective 6)

The statistical comparison of the above results is presented in this section. The Friedman statistic derived from the recorded AUC values for the 28 graphs considered (using different parameter settings) are summarized in Tables C.8 to C.9 respectively where, as before, the "AR" column gives the average AUC ranked performance and the "AR+CD" column gives the average AUC value plus the calculated critical difference. Table C.8 shows the average ranking of each classifier on graphs where $|L_V| = 2$ and $d = 23$ (mm). In this case the Friedman test value, calculated using $K - 1 = 50 - 1 = 49$ degrees of freedom, was 216.91. The corresponding p-value (threshold) was $9.823E - 11$; and the F-distribution with 49 and 1323 degrees of freedom, $F(49, 1323)$, was 5.07. The critical value for $F(49, 1323)$, with a critical difference level of $\alpha = 0.05$, was 1.364. Thus the p-value ($9.823E - 11$) is less than 0.005 and that the F-distribution (5.07) is larger than the corresponding F-distribution critical value (1.364); therefore we can reject the null hypothesis $H_0$. Referring back to Table C.8 the average rank results for the different VULS classifiers considered were all significant ($p < 0.005$), and thus a Nemenyi post-hoc test was deemed to be applicable to detect which particular classifiers differed significantly from each other. The significance diagram, corresponding to the information presented in Table C.8, is presented in Figure C.1. The CD value for the diagram is equal to 10.997.

From Figure C.1 it can be seen that, with respect to graphs where $|L_V| = 2$ (and $d = 23$ (mm)) the best performing classifier was Naive Bayes (a recorded AR value of 1.679); its performance was significantly better than all the other classifiers. In the diagram the classifiers highlighted in grey achieved comparable performance with J48. Similar results were obtained using $d = 28$ (mm) as reported in Chapter 7. However, as noted in Chapter 7, it should be high-lighted that although in the first instance it would appear that standard classifiers outperform the proposed VULS based classifiers, and this is indeed the case with respect to grid graphs, such classifiers would be difficult to apply in the case of un-regular graphs. However, VULS classifiers can be applied to any graphs regardless of their form (grid or non-grid).

TABLE C.8: Average Rankings of classifiers where $|L_V| = 2$ and $d = 23$ (mm)

| Algorithm | AR | AR+CD |
|---|---|---|
| NaiveBayes | 1.679 | 12.676 |
| Undirected-Degree4-max6-VULS | 15.661 | 26.658 |
| Undirected-Degree4-max6-freqVULS | 17.625 | 28.622 |
| Undirected-Degree4-max5-VULS | 19.250 | 30.247 |
| J48 | 19.339 | 30.336 |
| Directed-Degree8-max5-freqVULS | 21.589 | 32.586 |
| Directed-Degree8-max5-VULS | 21.607 | 32.604 |
| Undirected-Degree4-max5-freqVULS | 21.696 | 32.693 |
| Directed-Degree4-max6-freqVULS | 21.911 | 32.908 |
| Directed-Degree8-max6-freqVULS | 22.214 | 33.211 |
| Directed-Degree8-max6-minFreqVULS | 22.500 | 33.497 |
| Directed-Degree8-max6-VULS | 22.571 | 33.568 |
| Directed-Degree4-max6-VULS | 22.589 | 33.586 |
| Directed-Degree4-max4-VULS | 22.607 | 33.604 |
| Undirected-Degree4-max4-VULS | 22.661 | 33.658 |
| Directed-Degree8-max4-VULS | 22.750 | 33.747 |
| Directed-Degree4-max5-VULS | 22.982 | 33.979 |
| Directed-Degree4-max6-minVULS | 23.036 | 34.033 |
| Directed-Degree4-max5-freqVULS | 23.232 | 34.229 |
| Directed-Degree8-max5-minFreqVULS | 23.357 | 34.354 |
| Directed-Degree4-max6-minFreqVULS | 23.500 | 34.497 |
| Directed-Degree4-max5-minVULS | 23.714 | 34.711 |
| Directed-Degree8-max4-freqVULS | 23.893 | 34.890 |
| Directed-Degree4-max4-freqVULS | 23.964 | 34.961 |
| Directed-Degree8-max4-minFreqVULS | 24.125 | 35.122 |
| Directed-Degree4-max5-minFreqVULS | 25.179 | 36.176 |
| Undirected-Degree8-max6-freqVULS | 25.286 | 36.283 |
| Undirected-Degree8-max6-VULS | 25.732 | 36.729 |
| Directed-Degree4-max4-minVULS | 26.018 | 37.015 |
| Undirected-Degree8-max5-VULS | 26.054 | 37.0506 |
| Directed-Degree8-max4-minVULS | 27.161 | 38.158 |
| Undirected-Degree8-max5-freqVULS | 27.946 | 38.943 |
| Undirected-Degree4-max6-minVULS | 28.357 | 39.354 |
| Directed-Degree4-max4-minFreqVULS | 28.375 | 39.372 |
| Undirected-Degree8-max4-VULS | 28.679 | 39.676 |
| Undirected-Degree4-max4-freqVULS | 28.732 | 39.729 |
| Directed-Degree8-max5-minVULS | 29.036 | 40.033 |
| Directed-Degree8-max6-minVULS | 29.286 | 40.283 |
| Undirected-Degree8-max4-freqVULS | 29.768 | 40.765 |

| | | |
|---|---|---|
| Undirected-Degree4-max5-minVULS | 29.857 | 40.854 |
| Undirected-Degree4-max4-minVULS | 30.339 | 41.336 |
| Undirected-Degree4-max5-minFreqVULS | 32.036 | 43.033 |
| Undirected-Degree4-max6-minFreqVULS | 32.107 | 43.104 |
| Undirected-Degree8-max6-minVULS | 32.179 | 43.176 |
| Undirected-Degree8-max4-minVULS | 33.196 | 44.193 |
| Undirected-Degree8-max6-minFreqVULS | 33.696 | 44.693 |
| Undirected-Degree4-max4-minFreqVULS | 33.750 | 44.747 |
| Undirected-Degree8-max4-minFreqVULS | 33.839 | 44.836 |
| Undirected-Degree8-max5-minVULS | 34.161 | 45.158 |
| Undirected-Degree8-max5-minFreqVULS | 34.179 | 45.176 |

Similarly, Table C.2 show the average ranking of each classifier on graphs where $|L_V| = 3$ and $d = 23$. The Friedman test value on this case was 263.83, whilst the p-value (threshold) was $1.138E - 10$. The F-distribution with 49 and 1323 degrees of freedom, $F(49, 1323)$, was 6.428. The critical value for $F(49, 1323)$, with a critical difference level of $\alpha = 0.05$, was 1.364. Thus, from the foregoing we can note that the p-value ($1.138E - 10$) is smaller than 0.005 and that the F-distribution (6.428) is larger than the corresponding F-distribution critical value (1.364). Therefore we can reject the null hypothesis $H_0$ (that the observed performance differences among classifiers is simply a matter of chance). The significance diagram is presented in Figure C.2. The CD value for the diagram was 10.997. From the diagram it can be seen that the best performing classifier was Naive Bayes (a recorded AR value of 1.286), whilst J48 achieved a comparable performance. Classifiers highlighted in grey in the diagram indicate classifiers that achieved a comparable performance. Again similar results were reported in Chapter 7, and again the same observation with respect to grid graphs and the use of more standard classification techniques applies.

TABLE C.9: Average Rankings of the classifiers where $|L_V| = 3$ and $d = 23$ (mm)

| Algorithm | AR | AR+CD |
|---|---|---|
| NaiveBayes | 1.286 | 12.283 |
| J48 | 3.446 | 14.443 |
| Directed-Degree4-max6-VULS | 20.143 | 31.140 |
| Undirected-Degree4-max6-freqVULS | 20.339 | 31.336 |
| Undirected-Degree8-max5-freqVULS | 20.500 | 31.497 |
| Undirected-Degree8-max6-freqVULS | 20.536 | 31.533 |
| Directed-Degree4-max5-VULS | 20.893 | 31.890 |
| Undirected-Degree8-max6-VULS | 21.661 | 32.658 |
| Directed-Degree4-max4-VULS | 21.929 | 32.926 |
| Undirected-Degree8-max5-VULS | 21.946 | 32.943 |
| Directed-Degree8-max5-VULS | 22.125 | 33.122 |
| Directed-Degree4-max6-freqVULS | 22.500 | 33.497 |

| | | |
|---|---|---|
| Directed-Degree8-max6-VULS | 22.821 | 33.818 |
| Undirected-Degree8-max5-minVULS | 23.071 | 34.068 |
| Directed-Degree4-max5-freqVULS | 23.161 | 34.158 |
| Directed-Degree4-max4-freqVULS | 23.446 | 34.443 |
| Directed-Degree8-max4-VULS | 23.589 | 34.586 |
| Undirected-Degree8-max4-VULS | 23.696 | 34.693 |
| Directed-Degree4-max5-minVULS | 24.321 | 35.318 |
| Directed-Degree4-max6-minVULS | 24.625 | 35.622 |
| Undirected-Degree4-max6-VULS | 24.625 | 35.622 |
| Directed-Degree8-max6-freqVULS | 24.732 | 35.729 |
| Directed-Degree4-max6-minFreqVULS | 25.000 | 35.997 |
| Undirected-Degree4-max5-freqVULS | 25.036 | 36.033 |
| Directed-Degree8-max4-freqVULS | 25.071 | 36.068 |
| Directed-Degree8-max5-minVULS | 25.286 | 36.283 |
| Directed-Degree8-max4-minVULS | 25.357 | 36.354 |
| Directed-Degree8-max5-freqVULS | 25.464 | 36.461 |
| Directed-Degree4-max5-minFreqVULS | 25.554 | 36.551 |
| Undirected-Degree4-max4-VULS | 26.679 | 37.676 |
| Undirected-Degree4-max5-VULS | 27.232 | 38.229 |
| Directed-Degree8-max6-minVULS | 27.304 | 38.301 |
| Directed-Degree8-max5-minFreqVULS | 27.714 | 38.711 |
| Undirected-Degree8-max6-minVULS | 28.393 | 39.390 |
| Directed-Degree4-max4-minVULS | 28.857 | 39.854 |
| Directed-Degree8-max6-minFreqVULS | 29.304 | 40.301 |
| Directed-Degree8-max4-minFreqVULS | 29.429 | 40.426 |
| Undirected-Degree4-max4-freqVULS | 29.679 | 40.676 |
| Undirected-Degree8-max4-freqVULS | 30.018 | 41.015 |
| Undirected-Degree4-max6-minFreqVULS | 30.286 | 41.283 |
| Undirected-Degree8-max5-minFreqVULS | 30.482 | 41.479 |
| Directed-Degree4-max4-minFreqVULS | 31.179 | 42.176 |
| Undirected-Degree8-max6-minFreqVULS | 31.214 | 42.211 |
| Undirected-Degree8-max4-minVULS | 31.571 | 42.568 |
| Undirected-Degree4-max5-minFreqVULS | 32.071 | 43.068 |
| Undirected-Degree4-max5-minVULS | 32.661 | 43.658 |
| Undirected-Degree8-max4-minFreqVULS | 33.804 | 44.801 |
| Undirected-Degree4-max4-minVULS | 34.179 | 45.176 |
| Undirected-Degree4-max6-minVULS | 34.482 | 45.479 |
| Undirected-Degree4-max4-minFreqVULS | 36.304 | 47.301 |

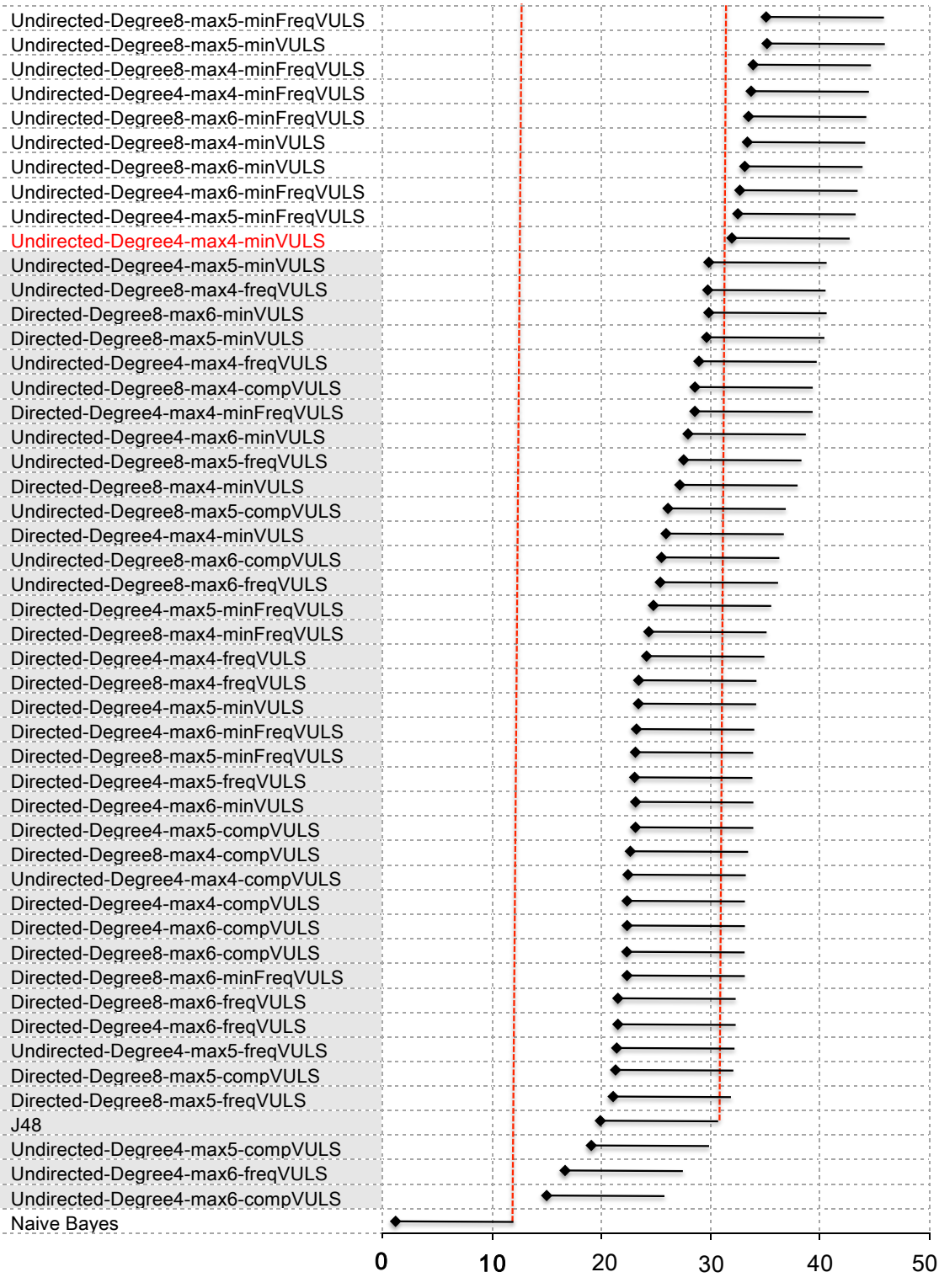



FIGURE C.1: Critical difference diagram generated using Nemenyi's post hoc test with $\alpha = 0.05$ for graphs where $|L_V| = 2$ and $d = 23$ (mm).

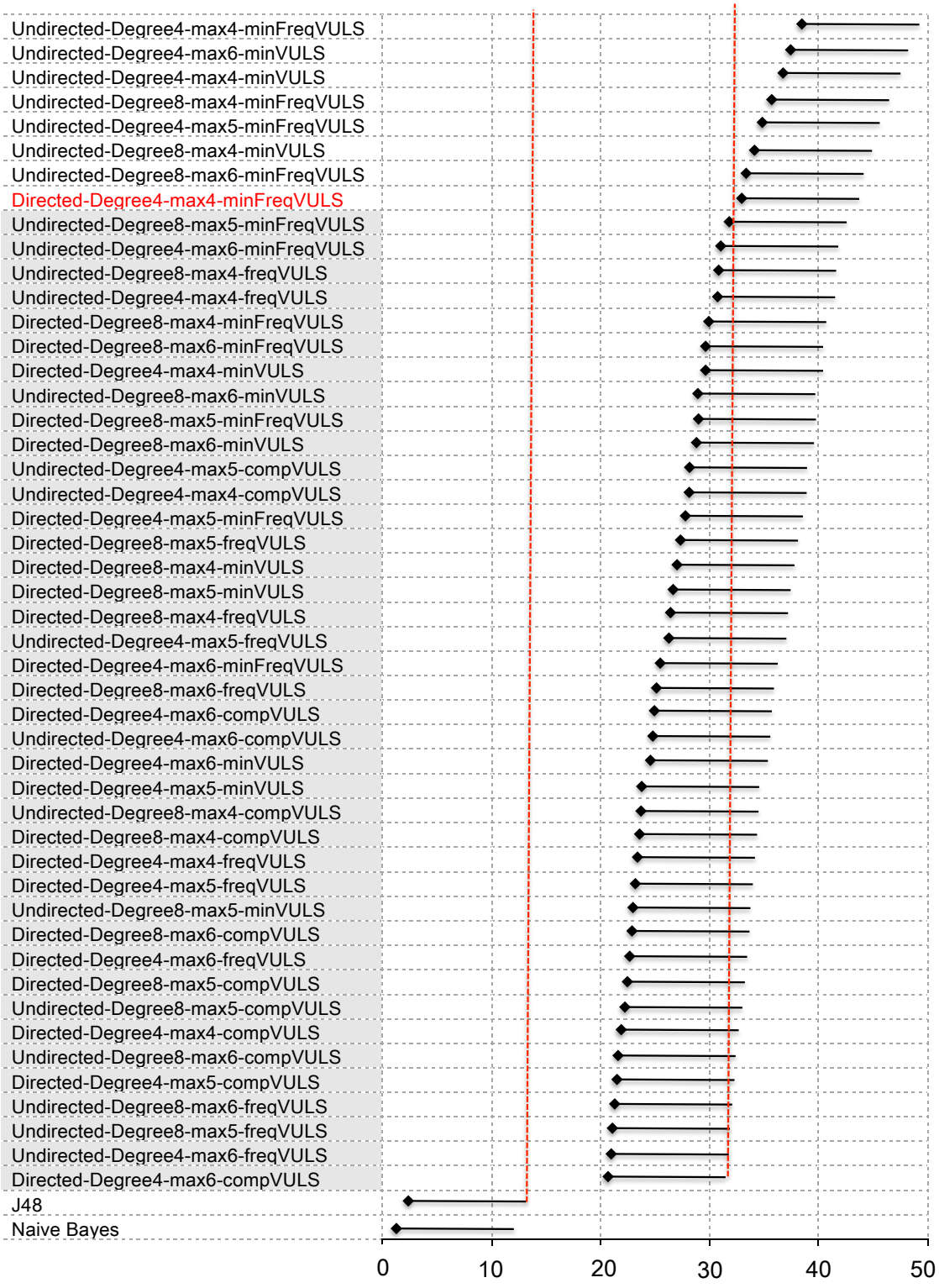



FIGURE C.2: Critical difference diagram generated using Nemenyi's post hoc test with $\alpha = 0.05$ for graphs where $|L_V| = 3$ and $d = 23$ (mm).

## C.8 Summary

This appendix has presented an evaluation of the proposed VULS mining and VULS classification processes in terms of AISF grids where $d = 23$. From the evaluation the following overall observations can be made:

1. With respect to the VULS mining the best coverage was produced using Complete VULS.
2. The most effective VULS mining algorithms were the Complete VULS and frequent VULS mining algorithms in terms of accuracy and AUC.
3. Frequent VULS mining is more efficient than Complete VULS mining.
4. There is no significant difference in terms of classification effectiveness between the VULS classifiers built with $max$ parameter settings in the range of $\{4, 5, 6\}$. However, using $max = 4$ is more efficient in the context of VULS mining.
5. The set of VULS produced using directed grid graph resulted in the most effective vertex classification (in terms of AUC).
6. With respect to the comparison with more traditional classifiers, Naive Bayes produced the best performance.
7. When considering different values for $|L_E|$, when using higher values for $|L_E|$ more significant VULS can be identified, thus a better vertex classification performance (in terms of AUC) can be achieved.
8. VULS classifiers tend to perform better when dealing with "grid" graphs ($degree = 4$) than "cross-grid" graphs ($degree = 8$).
9. Relatively speaking, VULS classifiers tend to perform better when dealing with directed graphs than undirected graphs.
10. With respect to the size of $|L_V|$, as the number of vertex labels increases from 2 to 3, the graph labelling becomes more diverse and as a result the classification performance tends to deteriorate.
11. VULS vertex classification tended to deteriorate when using $d = 23$ (mm) compared with results obtained using $d = 28$ (mm) as reported in chapter 7.

Broadly the above findings corroborate the results obtained using $d = 28$ (mm) as reported in Chapter 7.