Exploring A Bioinformatics Clustering Algorithm

By

Mukhlis Matti

A Thesis Submitted To
Victoria University Of Wellington
In Fulfilment Of The Requirements For The Degree Of
Master Of Science
In Computer Science

Wellington, New Zealand
October 2004

Abstract

This thesis explores and evaluates MAXCCLUS, a bioinformatics clustering algorithm, which was designed to be used to cluster genes from microarray experimental data. MAXCCLUS does the clustering of genes depending on the textual data that describe the genes. MAXCCLUS attempts to create clusters of which it selects only the statistically significant clusters by running a significance test. It then attempts to generalise these clusters by using a simple greedy generalisation algorithm. We explore the behaviour of MAXCCLUS by running several clustering experiments that investigate various modifications to MAXCCLUS and its data. The thesis shows (a) that using the simple generalisation algorithm of MAXCCLUS gives better result than using an exhaustive search algorithm for generalisation, (b) the significance test that MAXCCLUS uses needs to be modified to take into consideration the dependency of some genes on other genes functionally, (c) it is advantageous to delete the non domain-relevant textual data that describe the genes but disadvantageous to add more textual data to describe the genes, and (d) that MAXCCLUS behaves poorly when it attempts to cluster genes that have adjacent categories instead of having two distinct categories only.

Acknowledgments

Firstly, I would like to thank God for helping me and giving me the opportunity to contribute something to human civilisation by letting me to do and finish the research project and write the thesis.

Secondly, I would like to thank my supervisor: Dr Peter Andreae, for his valuable feedback, effort, time, and support.

I would like also to thank the following people:

From Victoria University of Wellington:

From the School of Mathematics, Statistics & Computer Science:

Dr John Haywood and Dr Richard Arnold.

The people that take care of the school's computer systems:
Dr Ray Brownrigg, Chang Chui, Andrew Mellanby, Kirk Jackson, Duncan McEwan, Mark Davies, and Roger Cliffe.

The school administrative staff:
Dr Peter Donelan (Head of School), Millie Byrne, Prema Ram, Elsie Gatfield, Rowan McCaffery, and Ginny Nikorima.

From the School of Biological Sciences:
Dr Rod Lea and Dr Darren J Day.

The staff members of the Faculty of Science Student Administration Office.

From outside Victoria University of Wellington:

Dr Christine M Morris, Associate Professor and Director, Cancer Genetics Research Group, Department of Pathology, Christchurch School of Medicine, Christchurch, New Zealand.

Dr Don Love, Associate Professor, Functional & Chemical Genomics Laboratory, School of Biological Sciences, The University of Auckland, Auckland, New Zealand.

The staff members of:
Wellington Hospital Laboratory, Wellington, New Zealand.
Ministry of Health, Wellington, New Zealand.
Medical Laboratory - Wellington, Wellington, New Zealand.

At the end, I would like to thank my parents, my sister, her husband, and my niece for their valuable support.

Table of contents

Chapter 1

Introduction

This thesis explores MAXCCLUS, a bioinformatics clustering algorithm, by running a set of clustering experiments to study and analyse MAXCCLUS's behaviour, in order to identify the strengths and limitations of MAXCCLUS. The results of the experiments show that in general MAXCCLUS is doing a reasonably good job.

Clustering is the task of grouping instances into sets so that each set reflects the shared properties between its instances, providing that instances of other sets do not have all these properties.

Clustering as an approach of machine learning is applied in many fields. One of the important applications of clustering is in the field of bioinformatics. Bioinformatics is a branch of science that deals with the collection, storage, classification and analyses of the biological and biomedical information. One of the promising technologies used in bioinformatics is the *microarray* (also called *gene chip*) [Molla, M; Waddell, M; Page, D. and Shavlik, J. 2004]. Microarray biotechnology measures *gene expression levels* for thousands of genes in the same time and in a cost effective way. Gene expression level is the amount of protein that the gene produces. Comparing the gene expression rates *before* and *after* a specific event applied to biological cells or tissues (some examples are: antibiotic shock, heating, or cooling) during biological or biomedical experiments, helps the scientists to understand the genes functionality and gives them an insight into what is happening during these experiments.

Microarray gene expression data are produced around the world in *huge* amounts. This data is by no mean easy to analyse manually. Therefore, there is a need for automated techniques to do the job. Clustering is one of the automated techniques that can be handed to a machine (a computer) to cluster the data to groups depending on properties that this data share.

An important source of information for such automated techniques is the data about genes in large shared text databases such as Swiss-Prot [Swiss-Prot].

1.1 Outline of the Thesis

The rest of this chapter introduces the background to the thesis, and describes the MAXCCLUS algorithm in detail.

In chapter two of this thesis, we describe the methodology we used to analyse MAXCCLUS and the results of the clustering experiments. At the end of chapter two we show how much of the data the original MAXCCLUS (without any changes to it) is able to describe.

In chapter three, we explore MAXCCLUS's simple generalisation approach, and we introduce a new generalisation approach, which is much powerful and much more expensive. The result of running the clustering experiment of chapter three is to show that MAXCCLUS's simple generalisation is better than the introduced one, by being able to explain the same genes in much less time.

In chapter four, we explore the significance test that MAXCCLUS depends on to select the clusters that are statistically significant from the total clusters it creates. The result of running the clustering experiment of chapter four is to show that MAXCCLUS is ignoring the fact that some instances are dependent on other ones, which leads MAXCCLUS to select clusters that bigger than they should be to be considered statistically significant. Therefore the result suggests that the significance test should be changed to take into its account the dependency of some instances on other ones.

In chapter five, we explore deleting some of the textual data, which describe the instances that MAXCCLUS attempts to cluster. The result of running the clustering experiment of chapter five shows that it is advantageous to delete some of the textual data providing that it is not domain-relevant. Deleting this data lets MAXCCLUS characterise the clusters it creates in a more scientifically informative way, and to describe almost the same number of genes faster.

In chapter six, also we explore the textual data. This time we attempt to add more textual data (synonyms and/or hypernyms of already existing textual data) to describe the instances. The result of running the clustering experiment of chapter six is to show that it is a bad idea to add more textual data: MAXCCLUS is not able to describe any more genes than before adding more textual data, it creates many alternative characterisations for the clusters that can confuse the user who is trying to make sense of the clusters, and it took much longer to cluster.

In chapter seven, we explore the behaviour of MAXCCLUS when it attempts to cluster instances that have continuous range of categories instead of having two categories separated with a clear gap (the ones that MAXCCLUS originally uses). The result of running the clustering experiment of chapter seven is to show that MAXCCLUS, like some other clustering algorithms, is not able to produce as good results when clustering using a continuous range of categories as when clustering using two distinct categories.

In chapter eight, we briefly describe several tools that we developed for the project of this thesis.

At the end, in chapter nine, we give the overall conclusions for this thesis.

1.2 Clustering

There are many approaches to clustering [Fasulo, D. 1999]. Two of the popular approaches are: *K-means* clustering, and *Agglomerative* clustering.

K-means clustering

In k-means clustering [Falkenauer E. and Marchand A. 2001], the algorithm needs in advance the number of clusters (*k*) that it will cluster the instances to. The algorithm initially randomly assigns *k* cluster centres. Next the algorithm clusters each instance to the cluster whose centre is nearest to this instance. Then the algorithm changes each cluster centre to the mean of the centres of its instances. The last two steps repeat until no changes occur any more. K-means clustering assumes *numerical* attributes because it must be able to compute the cluster mean.

Agglomerative clustering (bottom up)

In agglomerative clustering, the algorithm constructs a tree out of the instances to be clustered. Initially each one of the instances presents a cluster of its own. Then the algorithm finds the closest two clusters and merges them into a new cluster, which represents the parent cluster of the two merged clusters. This step repeats until there are no clusters left to merge. This kind of agglomerative clustering is called *single linkage* (also referred to as *nearest neighbour*). Agglomerative clustering assumes a *distance* measure can be computed.

1.3 MAXCCLUS

MAXCCLUS is a clustering algorithm designed and implemented by Peter Andreae that uses a novel approach for clustering instances [Andreae, P; Shavlik, J; and Molla M. 2001]. This clustering algorithm is designed to cluster the genes of a microarray experiment to find characterisations of subsets of the genes that behave in the same way during the microarray experiment. The goal behind designing MAXCCLUS is to make sense of the data that the microarray experiment produces, so that scientists can interpret the data more easily by having a clearer view of what is happening in the microarray experiments.

1.4 Input data types to MAXCCLUS

MAXCCLUS depends on two types of data to cluster instances: *experimental* data and *textual* data.

1.4.1 Experimental data

The experimental data that is used, as an input to MAXCCLUS, are actual experimental data from microarray experiments. These experiments are from the University of Wisconsin, Madison, USA. They are a set of 46 experiments on E-Coli, a kind of bacteria. Each microarray experiment dataset is about the genes (the instances) that were involved in the microarray experiment. The experimental data is a table (example: Table 1.1) in which each row contains the instance ID, and a number that represents the expression ratio, which describes the regulation of this gene as a result of this microarray experiment, under a physical or chemical event, such as antibiotic shock or temperature shock. This expression ratio in fact represents the category of the regulation that this instance belongs to, under this microarray experiment. If the expression ratio is *greater than 2.0*, we say that the instance is *Up* regulated.

If the expression ratio is *less than 0.5*, we say that the instance is *Down* regulated. If the expression ratio is in the range [0.5,2.0], this means that this expression ratio is uncertain; we will consider it unreliable and we will ignore it, and the gene will not be included in the experimental input data to MAXCCLUS.

So practically, we can replace the expression ratio of Table 1.1, by its category as in Table 1.2. This makes each microarray experiment dataset be a table (Table 1.2) in which each row contains the instance ID, and the instance category (*Up* or *Down*).

Notice that instances with IDs *b0003* and *b0004* are not included in Table 1.2, because their expression ratios are *1.7* and *0.6* respectively (Table 1.1), which make these expression ratios uncertain.

Instance ID	**Expression ratio**
b0001	2.7
b0002	0.3
b0003	1.7
b0004	0.6
b0005	3.2
b0006	0.4
b0007	0.15

***Table 1.1**: An experimental data source example. This experimental data source is a table in which each row contains the instance ID and a number that represents the expression ratio.*

Instance ID	**Category**
b0001	Up
b0002	Down
b0005	Up
b0006	Down
b0007	Down

***Table 1.2**: The same experimental data source example of Table 1.1, but the expression ratios were replaced by their categories. This table (Table 1.2) does not include the instances that have uncertain expression ratios (b0003 and b0004). The data in this table (Table 1.2) will be the experimental data input to the clustering algorithms.*

1.4.2 The textual data

The other type of data that MAXCCLUS uses, when attempting to cluster instances, is the *textual* data. The textual data presents the text that annotates the genes that is available in some fields of the databases about these genes. *Swiss-Prot* [Swiss-Prot] is one of these databases. Swiss-Prot is a protein knowledgebase. Proteins are the life building blocks that genes code for. Swiss-Prot lists the genes with text fields. Figure 1.1 shows a snapshot of the Swiss-Prot database showing the fields for the protein corresponding to gene b0003. Each line has a type [Swiss-Prot User Manual]. Not all the text fields are useful for the purpose of creating the textual data that MAXCCLUS needs to use. For example the last few lines of the field *CC* are the *copyright notice* and they are repeated for all the genes. Including the same textual data for all the genes is not useful because it will not discriminate between the genes.

```
ID   KHSE_ECOLI     STANDARD;      PRT;   310 AA.
AC   P00547;
DT   21-JUL-1986 (Rel. 01, Created)
DT   01-APR-1993 (Rel. 25, Last sequence update)
DT   01-OCT-2000 (Rel. 40, Last annotation update)
DE   HOMOSERINE KINASE (EC 2.7.1.39) (HK).
GN   THRB.
OS   Escherichia coli.
OC   Bacteria; Proteobacteria; gamma subdivision; Enterobacteriaceae;
OC   Escherichia.
OX   NCBI_TaxID=562;
RN   [1]
RP   SEQUENCE FROM N.A.
RX   MEDLINE=81150470; PubMed=6259626;
RA   Cossart P., Katinka M., Yaniv M.;
RT   "Nucleotide sequence of the thrB gene of E. coli, and its two
RT   adjacent regions; the thrAB and thrBC junctions.";
RL   Nucleic Acids Res. 9:339-347(1981).
RN   [2]
RP   SEQUENCE FROM N.A.
RC   STRAIN=K12;
RX   MEDLINE=92334977; PubMed=1630901;
RA   Yura T., Mori H., Nagai H., Nagata T., Ishihama A., Fujita N.,
RA   Isono K., Mizobuchi K., Nakata A.;
RT   "Systematic sequencing of the Escherichia coli genome: analysis of
RT   the 0-2.4 min region.";
RL   Nucleic Acids Res. 20:3305-3308(1992).
RN   [3]
RP   SEQUENCE FROM N.A.
RC   STRAIN=K12 / MG1655;
RX   MEDLINE=95334362; PubMed=7610040;
RA   Burland V.D., Plunkett G. III, Sofia H.J., Daniels D.L.,
RA   Blattner F.R.;
RT   "Analysis of the Escherichia coli genome VI: DNA sequence of the
RT   region from 92.8 through 100 minutes.";
RL   Nucleic Acids Res. 23:2105-2119(1995).
RN   [4]
RP   REVISIONS TO 166-190.
RA   Deborde D.C., Strange J.C., Wright B.E.;
RL   Submitted (XXX-1993) to the EMBL/GenBank/DDBJ databases.
CC   -!- CATALYTIC ACTIVITY: ATP + L-HOMOSERINE = ADP + O-PHOSPHO-L-
CC       HOMOSERINE.
CC   -!- PATHWAY: THREONINE BIOSYNTHESIS.
CC   -!- SIMILARITY: BELONGS TO THE GHMP KINASE FAMILY. HOMOSERINE
CC       KINASE SUBFAMILY.
CC   --------------------------------------------------------------------------
CC   This SWISS-PROT entry is copyright. It is produced through a collaboration
CC   between  the Swiss Institute of Bioinformatics  and the  EMBL outstation -
CC   the European Bioinformatics Institute.  There are no  restrictions on  its
CC   use  by  non-profit  institutions as long  as its content  is  in  no  way
CC   modified and this statement is not removed.  Usage  by  and for commercial
CC   entities requires a license agreement (See http://www.isb-sib.ch/announce/
CC   or send an email to license@isb-sib.ch).
CC   --------------------------------------------------------------------------
DR   EMBL; J01706; AAA83915.1; ALT_SEQ.
DR   EMBL; D10483; BAA01287.1; ALT_SEQ.
DR   EMBL; L13601; AAA20618.1; -.
DR   EMBL; U14003; AAA97302.1; -.
DR   EMBL; AE000111; AAC73114.1; -.
DR   PIR; A00658; KIECM.
DR   PIR; S40532; S40532.
DR   SWISS-2DPAGE; P00547; COLI.
DR   ECOGENE; EG10999; THRB.
DR   INTERPRO; IPR000870; -.
DR   INTERPRO; IPR001745; -.
DR   PFAM; PF00288; GHMP_kinases; 1.
DR   PRINTS; PR00958; HOMSERKINASE.
DR   PROSITE; PS00627; GHMP_KINASES_ATP; 1.
KW   Threonine biosynthesis; Transferase; Kinase; ATP-binding.
FT   NP_BIND      91     101       ATP (POTENTIAL).
SQ   SEQUENCE   310 AA;  33623 MW;  0F225F9F1B634BE8 CRC64;
     MVKVYAPASS ANMSVGFDVL GAAVTPVDGA LLGDVVTVEA AETFSLNNLG RFADKLPSEP
     RENIVYQCWE RFCQELGKQI PVAMTLEKNM PIGSGLGSSA CSVVAALMAM NEHCGKPLND
     TRLLALMGEL EGRISGSIHY DNVAPCFLGG MQLMIEENDI ISQQVPGFDE WLWVLAYPGI
     KVSTAEARAI LPAQYRRQDC IAHGRHLAGF IHACYSRQPE LAAKLMKDVI AEPYRERLLP
     GFRQARQAVA EIGAVASGIS GSGPTLFALC DKPETAQRVA DWLGKNYLQN QEGFVHICRL
     DTAGARVLEN
```

Figure 1.1: *A snapshot of the text annotated to genes from Swiss-Prot database. The snapshot is for the gene b0003.*

1.5 Pre-processing textual data

This section describes the original pre-processing of the textual data.

The raw data of Swiss-Prot needs to be pre-processed to prepare the textual data for MAXCCLUS. Some of the modifications made to the raw data are:

- Select the *DE*, *KW*, *RT*, and *CC* fields from the Swiss-Prot entries.
- Convert the text to lowercase letters.
- Remove the copyright notice from the *CC* lines, and some of the *reference title* lines (*RT*) that appears in the text of most of the genes.
- Break sentences and phrases into their words. For example "*Escherichia coli*" becomes two words "*escherichia*" and "*coli*".
- Break the "words" that have hierarchical structure into their substructures including the original full structure. For example "*EC 2.7.1.39*" becomes "*ec@2@7@1@39*", "*ec@2@7@1*", "*ec@2@7*", and "*ec@2*". The use of "@" instead of "." is because all punctuation symbols are to be removed.
- Remove punctuation.
- Replace words with their stems using the *Porter stemmer algorithm* [Porter, M. F. 1980] that removes suffixes and prefixes from words, because words that have the same stem usually have similar meanings.
- Remove duplicate words in each entry.

After this all the words that belong to a gene comprise the set of descriptors to this gene. The textual data consists of a table in which each row in the table contains the gene id associated with the gene descriptor set (as in Table 1.3).

Instance ID	Descriptors
b0001	acid biosynthesis escherichia isoleucine mutant region structural sulfate …
b0002	active aspartate belong cytoplasmic h(2)o kda molecular pathway protein …
b0003	adjacent adp analysis atp belong biosynthesis coli ec@2@7@1@39 …
b0004	coli compared enzyme mutant observed protein region sulfate transferase …
b0005	abundance dna-binding encoded homodimer integral orthophosphate …
b0006	chromatography hypothetical membrane nucleotide predicts subunit sulfate …
b0007	51.7 acid analysis biosynthesis coli oxidoreductase potential regulatory …

***Table 1.3**: A snapshot of the descriptors table showing instances and their descriptors. The text data in this table is the textual data input to MAXCCLUS.*

When the experimental *and* the textual data are available, we can run MAXCCLUS to create subsets of the instances that behave in the same way during the microarray experiment.

1.6 How MAXCCLUS works

Figure 1.2 shows the MAXCCLUS clustering algorithm. First, MAXCCLUS uses the textual data to construct clusters of instances. Second, from these clusters MAXCCLUS selects the good clusters by using the experimental data. Third, MAXCCLUS attempts to generalise the good clusters producing the generalised clusters. Finally, from the generalised clusters MAXCCLUS attempts to create the cover clusters from which it constructs rules and characterisations.

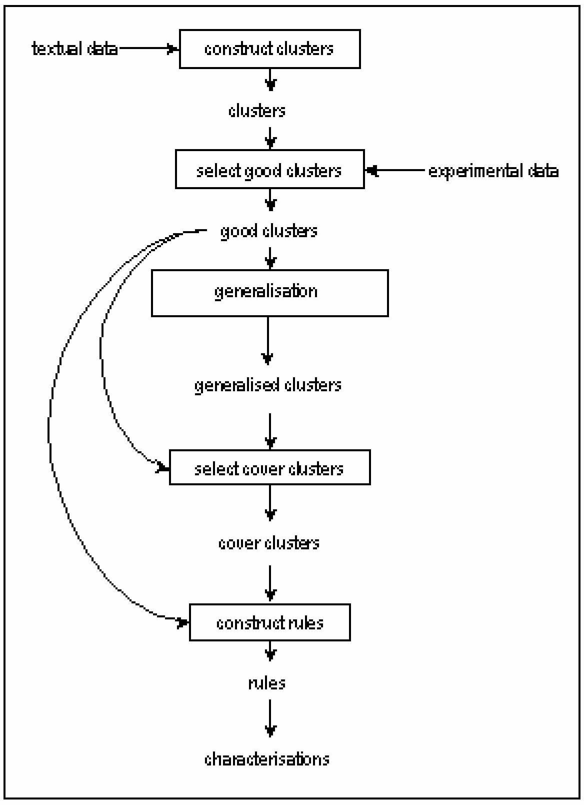


***Figure 1.2**: The MAXCCLUS clustering algorithm.*

1.6.1 Construct the clusters

MAXCCLUS starts clustering the instances (genes) using the textual data by finding, for each pair of instances, the set of descriptors that are shared by these two instances. MAXCCLUS marks each pair of instances with the number of descriptors shared between them. Then for each marked pair of instances MAXCCLUS creates a cluster. Then MAXCCLUS adds to this cluster other instances that share the same descriptors with the cluster's instances.

The clusters that MAXCCLUS creates are determined only by the textual data while ignoring the instances' categories from the experimental data. Each of the constructed clusters has a set of descriptors (features) that is *maximal* in the sense that MAXCCLUS cannot add more descriptors to the cluster without excluding some instances from the cluster. Also each cluster is *maximal* in the sense that MAXCCLUS cannot add more instances to the cluster without reducing the number of descriptors that describe the cluster.

Each cluster MAXCCLUS creates can be viewed either as a set of instances or as a set of word (descriptors) that describe the cluster. We refer to the first view as the *extensional* view, and refer to the second as the *intensional* view.

Not all the clusters MAXCCLUS creates are relevant to the microarray experiment; and therefore not all of the clusters are meaningful. This is because, as mentioned above, the clusters that MAXCCLUS creates are determined only by the textual data while ignoring the instance categories from the experimental data. To obtain useful clusters, MAXCCLUS needs some criteria to determine which of the clusters that it constructs are relevant to the microarray experiment.

1.6.2 Select good clusters

MAXCCLUS considers a cluster to be a good cluster if it passes both of the two tests: the *accuracy* and the *significance* tests.

The accuracy test

One of the two conditions for a cluster to be considered as a good cluster is that a cluster needs to be accurate enough. Accuracy (purity) is a percentage measure of the number of instances in the cluster that have the same category (*Up* or *Down*). The accuracy is also reflects the accuracy of the rules that MAXCCLUS creates (in later step) to describe the cluster. The user that uses MAXCCLUS package determines the minimum accuracy. For example if the user determined that the accuracy should be 95%, this means that each cluster must have 95%, or more, of its instances having the same category for MAXCCLUS to consider this cluster as a good cluster.

The significance test

The other condition for a cluster to be considered a good cluster is that an accurate cluster needs to be statistically significant. The significance test distinguishes between the accurate clusters that may result from a statistical chance, and the clusters that represent a real relationship between the descriptors (features) and the categories. MAXCCLUS runs a significance test on the clusters it creates to determine which clusters are statistically significant. The user of MAXCCLUS determines the minimum significance as a percentage (for example 95%).

MAXCCLUS uses a permutation test. It does this test by repeatedly assigning categories (*Up*s or *Down*s) randomly to the instances. It then evaluates the purity of all the clusters, given these random assignments of categories. Any pure clusters it finds cannot represent significant clusters because the categories are random. There will be many small pure clusters, but fewer large pure clusters. It finds the minimum size cluster for which the probability of finding a pure cluster of this size is less than: 1-significance.

It then assigns the true categories to the instances and selects the pure clusters that are at least this minimum size. These clusters are presumed to be significant, and not the product of statistical chance, and are referred to as "*good clusters*" or "*significant clusters*".

After selecting good clusters, MAXCCLUS can go through one or more of three paths (viewing them from left to right in Figure 1.1):

The first path is that MAXCCLUS attempts to directly construct the rules that characterise the good clusters.

The second path is that MAXCCLUS attempts to select a subset of the good clusters that covers all the instances in the good clusters. It then constructs the rules that characterise these cover clusters.

The third path is that MAXCCLUS attempts to generalise the good clusters to create generalised clusters; then selects a cover for the generalised clusters; after that constructs the rules that characterise the generalised cover clusters.

The following sections explain how MAXCCLUS constructs rules, selects cover clusters, and performs generalisation.

1.6.3 Constructing rules

MAXCCLUS attempts to construct rules that describe each cluster. MAXCCLUS uses the cluster's descriptors (words) to construct all the rules that describe all the instances (genes) that are in the cluster, and at the same time exclude all the instances that are not in the cluster.

MAXCCLUS constructs conjunctive rules (using "*and*" to combine words logically). For example, a rule that MAXCCLUS constructs to describe the instances of an *Up* regulated cluster of instances may look like this "Rule: protein, sequence, subunit, belong". This rule can be interpreted as: *if an instance has all the words "protein", "sequence", "subunit", and "belong", then the instance is Up regulated*. Saying it differently: all the instances in the cluster can be described by the conjunction of the words "*protein*", "*sequence*", "*subunit*", and "*belong*"; this conjunction of words excludes all the instances that are outside the cluster. MAXCCLUS does not construct disjunctive rules (using "*or*" to combine words logically).

Words MAXCCLUS uses in constructing rules

There are two sets of words that MAXCCLUS may use in constructing rules to describe a cluster, the *Necessary* and the *Sufficient* sets.

The *Necessary* set of words contains the words that exist in every rule of the cluster. Each *Necessary* word is in the description of every gene in the cluster, but some genes outside the cluster share all but this word in the cluster's descriptors. These words are necessary to exclude these "*near misses*".

Sometimes the *Necessary* words are not enough to exclude some of the genes that are out the cluster, because some of these genes can have all of the *Necessary* words. MAXCCLUS finds another set of words, the *Sufficient* words. By using at least some of the *Sufficient* words in conjunction with the *Necessary* set of words, MAXCCLUS creates minimal rules of which each rule describes the instances in the cluster, but not true for the instances outside the, cluster even the ones that have all the *Necessary* words. The words from the *Sufficient* set of words are present in *at least* one but *not* all rules of the cluster.

For a rule to be minimal no word can be deleted from the rule without affecting the set of instances this rule describe. Deleting any word from a minimal rule allows the rule to also describe some instances from outside the cluster, which decreases the accuracy of the rule by describing instances that have different categories than the cluster's instances category.

Word sets that MAXCCLUS uses in characterising a cluster

Both the *Necessary* and the *Sufficient* sets of words characterise the instances of the cluster, but they are not the only ones. MAXCCLUS constructs two other sets of words to characterise the instances of a cluster, the S*upplementary* and the *All but X* sets of words.

The S*upplementary* set of words contains the words that are not in any rule of the cluster, but are still true of all instances in the cluster. Since the S*upplementary* words are also true for some instances outside the cluster, these words do not help in distinguishing the cluster from other clusters.

The *All but X* set of words (also called *frequent words*) contains the words that are not in any rule of the cluster, but are still true of all but *X* of the instances in the cluster. For example, a cluster may have 217 instances with characterisation that say "*All but 2: hypothetical*", which means that 215 of the cluster instances have the word "*hypothetical*" in their descriptor set, but the other 2 instances do not have this word.

Both the S*upplementary* and the *All but X* sets of words are to help the user of MAXCCLUS (the scientist) by reflecting more of the characterisations of the instances in the cluster. These extra characterisations are not reflected in the rules.

1.6.4 Generalisation

MAXCCLUS attempts to generalise each of the good clusters by adding some instances from outside the cluster. The instances added share some of the same descriptors as the cluster's characterisation, but must have the same category as the cluster's category. Therefore, generalisation increases the accuracy of the cluster by increasing the number of correct instances. We call the resulting clusters the *generalised clusters*.

MAXCCLUS performs generalisation by constructing generalised rules that cover all the genes in the cluster and do not cover any *incorrect* instances outside the clusters, but may cover some additional correct instances. This small change to the rule constructions leads to a kind of generalisation that is simple and strict, however it is fast in execution and as good in covering instances as the more complex, and more time consuming generalisation introduced in Chapter 3.

1.6.5 Select cover clusters

After constructing the good clusters (or the generalised clusters), MAXCCLUS attempts to select the cover clusters. Because the set of the good (or generalised) clusters may contain clusters that are overlapping and others that are subsumed by bigger clusters in the same set, MAXCCLUS attempts to select a minimal subset of the clusters that cover the same instances. We call this subset the *cover clusters* if it is the cover for the good clusters, or the *generalised cover clusters* if it is the cover clusters for the generalised clusters.

1.7 How MAXCCLUS reports its clustering results

After finishing clustering, MAXCCLUS reports its clustering results as an XML (Extensible Markup Language) file, which was originally rendered to HTML (HyperText Markup Language) file. Rendering was originally done by running a Java program on each of the XML files to create the respective HTML file. This approach is not flexible enough in case one needs to change how the HTML presents the data; furthermore this approach doubles the computer storage size needed to store only the XML files (because of having the HTML files too), which can be considerable when running MAXCCLUS on many data sets (some XML and HTML files can exceed 34MB in size). We modified the rendering by using only one small (7KB) XSL (Extensible Stylesheet Language) file to render, on the fly, any XML file created by MAXCCLUS, to a HTML file by using Microsoft Internet Explorer to open the XML file.

Figure 1.3 shows a snapshot of MAXCCLUS clustering results with Microsoft Internet Explorer using the XSL file.

```
Data/gene-dkcr-mp1s ===> Data/Experiments/exp03-gene-category

ClusterSet
clusterCount: 19
minAccuracy: 95
minConfidence: 95
minSize-Up= "212"
minSize-Dn= "5"
instanceCount: 335

DataSource
instanceSource: Data/Experiments/exp03-gene-category
textSource: Data/gene-dkcr-mp1s

instanceCount: 495
UpCount: 456
DnCount: 39
--------------------------------------------------------------------------------
Cluster
category: Dn
accuracy: 100.0
instanceCount: 8

Characterisation: Necessary: protein , sequence , subunit ,
Sufficient: belong , family ,
Supplementary: coli , escherichia ,

Rule: protein , sequence , belong , subunit ,

Rule: protein , sequence , family , subunit ,

Operons: b0882 , b0911 , b1260 , b1923 , b2614 , b3295 , b3699 , b3829 ,
--------------------------------------------------------------------------------
Cluster
category: Dn
accuracy: 100.0
instanceCount: 7

Characterisation: Necessary: mutant ,
Sufficient: gene , k-12 ,
Supplementary: coli , escherichia , protein , belong , family ,

Rule: gene , mutant ,

Rule: k-12 , mutant ,

Operons: b1136 , b1236 , b1779 , b1923 , b2497 , b3295 , b4177 ,
--------------------------------------------------------------------------------
Cluster
category: Dn
accuracy: 100.0
instanceCount: 5

Characterisation: Necessary: belong , encoded , k-12 , as ,
Supplementary: coli , escherichia , family ,

Rule: belong , encoded , k-12 , as ,

Operons: b1243 , b1852 , b3295 , b3829 , b4177 ,
--------------------------------------------------------------------------------
Cluster
category: Dn
accuracy: 100.0
instanceCount: 5

Characterisation: Necessary: nucleotide , salmonella ,
Supplementary: coli , escherichia , sequence , belong , family ,

Rule: nucleotide , salmonella ,

Operons: b1260 , b1264 , b1779 , b1923 , b3295 ,
--------------------------------------------------------------------------------
Cluster
category: Dn
accuracy: 100.0
instanceCount: 5

Characterisation: Sufficient: sequence , belong , encoded , step , k-12 , as ,
Supplementary: coli , escherichia ,

Rule: step , as ,

Rule: belong , step , k-12 ,

Rule: sequence , encoded , k-12 , as ,

Rule: belong , encoded , k-12 , as ,

Operons: b1852 , b2780 , b3295 , b3829 , b4177 ,
--------------------------------------------------------------------------------
Cluster
category: Dn
accuracy: 100.0
instanceCount: 8
```

Figure 1.3: *A snapshot of MAXCCLUS clustering results.*

Chapter 2

Methodology

This chapter describes the methodology we used to design the different clustering experiments and to analyse the results that MAXCCLUS produced from the experiments.

MAXCCLUS produces the results of its clustering and stores these results into XML files. Some of these XML files can exceed 34MB in size. Browsing this amount of data and attempting to analyse it is a difficult job. Therefore we chose a methodology that can help us to simplify the results and enables us to compare different clustering experiments.

2.1 The clustering experiments

The project designed different clustering experiments to study the behaviour of MAXCCLUS. There are many control variables that one can change in a clustering experiment to create a new clustering experiment. We attempted to design the clustering experiments so that each one of them differs from the reference clustering experiment by one controlling variable. The reference clustering experiment is obtained by running MAXCCLUS using the experimental and textual data that is explained in Chapter 1. Comparing each one of the clustering experiments to the same reference clustering experiment (the normal clustering experiment) gives an understanding of the behaviour of MAXCCLUS and its limitations, and it helps identify improvements to MAXCCLUS, or the experimental or textual data that it uses.

Figure 2.1 shows the reference clustering experiment, which we refer to as the *before* changes case, and the other clustering experiments, which we refer to as *after* changes case.

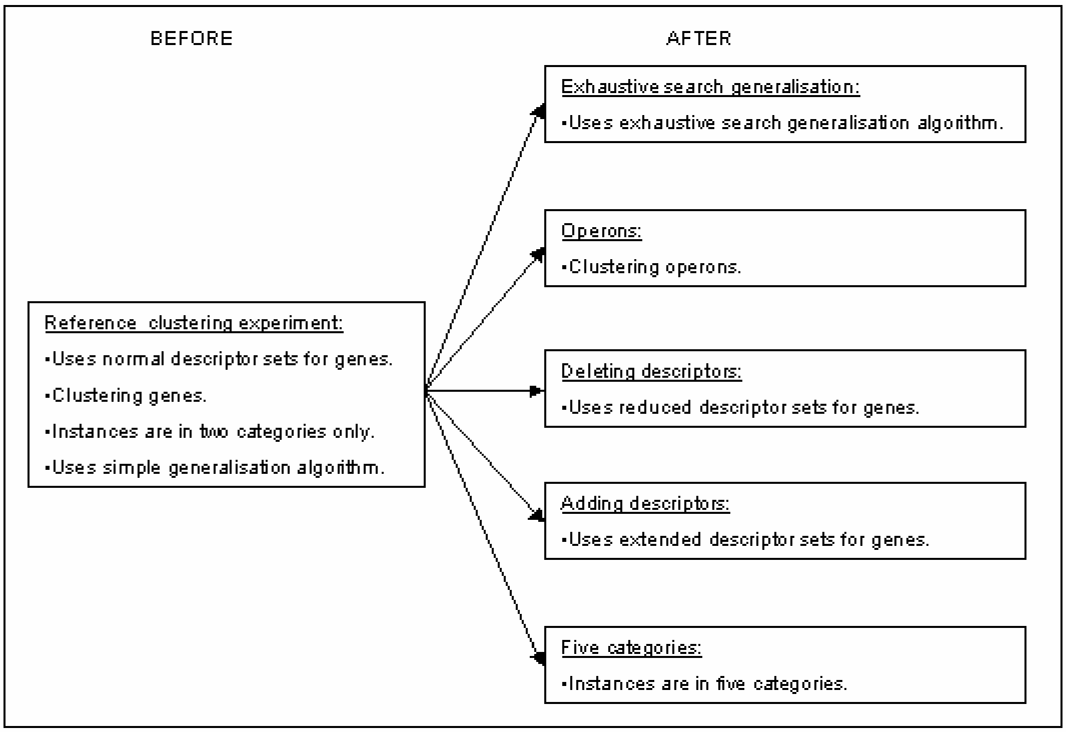

BEFORE
AFTER
Reference clustering experiment:
•Uses normal descriptor sets for genes.
•Clustering genes.
•Instances are in two categories only.
•Uses simple generalisation algorithm.
Exhaustive search generalisation:
•Uses exhaustive search generalisation algorithm.
Operons:
•Clustering operons.
Deleting descriptors:
•Uses reduced descriptor sets for genes.
Adding descriptors:
•Uses extended descriptor sets for genes.
Five categories:
•Instances are in five categories.

Figure 2. 1: The *before* and *after* changes clustering experiments.

Before any changes, in the reference clustering experiment, MAXCCLUS uses the normal descriptor sets for the genes (section 1.5), it attempts to cluster genes, the experimental data has instances (genes) that are in two categories only, and MAXCCLUS uses a simple generalisation algorithm when attempting to generalise the good clusters.

In the *after* changes case there are five clustering experiments. Each of these clustering experiments is based on the reference clustering experiment but with a change to one control variable.

The first clustering experiment (*Exhaustive search generalisation*) is designed to study the behaviour of MAXCCLUS when MAXCCLUS uses an *exhaustive* search algorithm to generalise the good clusters, instead of the *simple* generalisation algorithm that it uses in the reference clustering experiment.

The second clustering experiment (*Operons*) is designed to study the behaviour of MAXCCLUS regarding the significance test. In this clustering experiment MAXCCLUS attempts to cluster *operons* instead of genes. Operons are logical groups of genes that should behave in the same way during the microarray experiment. Using operons instead of genes requires different textual data from the one the normal clustering experiment uses. The textual data for the operon clustering experiment has descriptor sets for operons created from the union of the descriptor sets of their genes.

The third and fourth experiments - *Deleting descriptors* and *Adding descriptors* - are designed to study the behaviour of MAXCCLUS when changes are made to the pre-processing of the textual data by changing the descriptor sets of the genes. Deleting descriptors uses reduced descriptor sets for the genes by deleting words from two different dictionaries. The second clustering experiment (*Adding descriptors*) uses extended descriptor sets for the genes by adding synonyms, hypernyms, or both synonyms and hypernyms of the existing descriptors.

The last clustering experiment (*Five categories*) is designed to study the behaviour of MAXCCLUS when it attempts to cluster instances that are in five categories instead of only two categories. In this clustering experiment, MAXCCLUS uses experimental data that includes all the genes (instances), not only the ones that are clearly Up or clearly Down regulated.

2.2 Comparing clustering experiments results

To be able to study the behaviour of MAXCCLUS, we need to find a way to compare the results of using MAXCCLUS for each clustering experiment *after* having changes to the control variables, against the reference clustering experimental *before* changes to the control variables. We refer to the results of the reference clustering experiment with the word “*before*”, and we refer to the results of each other clustering experiment with the word “*after*”.

What to compare

MAXCCLUS reports its clustering results in XML files. The clustering results consist of some “*objects*”. The *objects* can be clusters, instances, rules, or rules’ words. We measured the clustering execution time separately because it is not reported in the XML files. Figure 2.2 list the maximum value of the objects count in an XML file to show the complexity of the comparison problem.

Object	Maximum count
Instances	980
Clusters	1,789
Rules	115,180
Rules’ words	306

Figure 2.2: The maximum count for each object in an XML clustering results file.

The comparison is made between objects in the *before* and the *after* results. The result sections in the following chapters present the compared results as the relative change between the *before* and the *after* results.

For each kind of object, the results are presented as the change to the total number of the objects, the number of retained objects (objects present in both *before* and *after* results), and the number of new objects. For n microarray experiments data set, the calculations are as the following:

Change%

For the percentage *Change%*, C_i is the percentage ratio of the number of $after$ objects to the number of $before$ objects in the ith microarray experiment, calculated as:

$$C_i = \frac{|after_i|}{|before_i|} \times 100$$

Retained%

For the percentage *Retained%*, R_i is the percentage ratio of the number of objects retained in the $after$ clustering experiment, with respect to the number of objects in the $before$ clustering experiment. The number of the retained objects is the number of objects that are in both the $before$ and $after$ results of the ith microarray experiment. R_i is calculated as:

$$R_i = \frac{|before_i \bigcap after_i|}{|before_i|} \times 100$$

New%

For the percentage *New%*, N_i is the percentage ratio of the number of objects in the $after$ results that were not in the $before$ results of the ith microarray experiment. N_i is calculated as:

$$N_i = \frac{|after_i - before_i|}{|after_i|} \times 100$$

Subsumed%

For the percentage *Subsumed%*, S_i is the percentage ratio of the number of objects in the $before$ results that are subsumed by objects in the $after$ results of the ith microarray experiment. S_i is calculated as:

$$S_i = \frac{|subsumed_i|}{|before_i|} \times 100$$

Where an object from $before_i$ is in $subsumed_i$ if there exists an object in $after_i$ that is strictly a generalisation of it.

Then we calculate the average, median, minimum value, maximum value, upper quartile, and lower quartile of C_i, R_i, N_i, and S_i over n microarray experiments data sets. We present the averages as histograms and other statistics as box and whisker plots [Williamson, D. 2002].

For each experiment, we calculate these measures for three sets of results from MAXCCLUS, the significant clusters (the good clusters, which are accurate and significant at the same time), the cover, and the generalised cover cluster sets, and present them side by side for comparison.

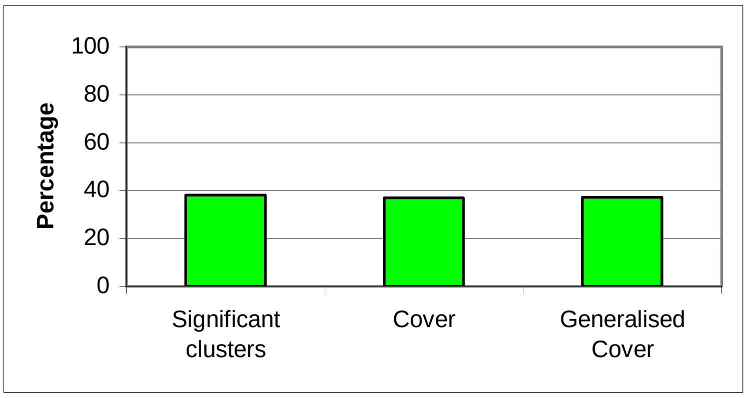
Not all the clustering experiments use the same microarray data sets, because some of the clustering experiments could not handle some of the data sets for one of two reasons: first, not enough computer memory to continue the clustering task, and second, the clustering task takes a very long time, which forces us to kill the running process for these data sets. Figure 2.3 lists the clustering experiments and the microarray data sets used to study the clustering results.

Clustering experiment	**Number of data sets**	**Data sets**
Exhaustive search generalisation	23	3, 6, 7, 8, 10, 11, 12, 13, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 35, 36, 40
Operons	20	6, 7, 10, 11, 12, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 35, 36, 40
Deleting descriptors	23	3, 6, 7, 8, 10, 11, 12, 13, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 35, 36, 40
Adding Descriptors	23	3, 6, 7, 8, 10, 11, 12, 13, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 35, 36, 40
Five categories	19	3, 6, 7, 10, 11, 12, 13, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30

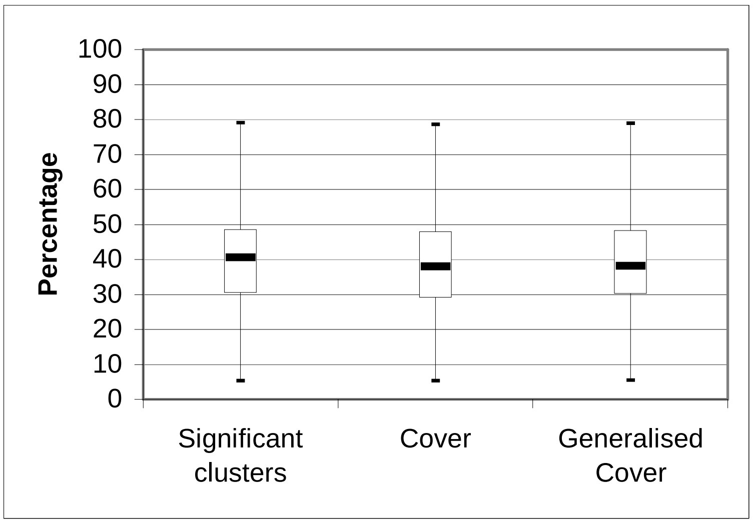
Figure 2.3: The clustering experiments and the microarray data sets used in studying the clustering results.

2.3 How much of the data can MAXCCLUS describe

Figure 2.4 shows the number of instances described originally by MAXCCLUS when running the reference clustering experiment on 38 microarray experimental data sets. On average, MAXCCLUS can describe about 37% of the instances in the data sets. The distributions show that there is a wide variation across the data sets: from about 5% to about 79%.



a: Average



b: Distributions

Figure 2.4: Number of instances described by running the reference clustering experiment.

Chapter 3

Exhaustive Search Generalisation

This chapter describes a clustering experiment designed to study MAXCCLUS's generalisation by introducing an exhaustive search generalisation clustering experiment and comparing its results with MAXCCLUS's simple generalisation used in the reference clustering experiment. The surprising result of this chapter is to show that MAXCCLUS's simple generalisation does very well compared with the more complex and more expensive exhaustive search generalisation, by explaining all the instances with much less execution time.

In the reference clustering experiment, MAXCCLUS uses a simple generalisation that attempts to add, to the cluster that is attempting to generalise, other instances that have the same category as the cluster. It does this using greedy algorithm that effectively drops words from the description of the cluster without including other instances that have different category from the cluster. We thought that having an exhaustive search generalisation would make MAXCCLUS able to generalise a cluster better than the simple greedy generalisation, by using a powerful search an by allowing some instances that have a different category from the cluster's category to be added to the cluster, as long as the accuracy is not sacrificed.

3.1 Data source

Both versions of MAXCCLUS use the same data as their input, the experimental data and the textual data as described in chapter 1.

3.2 Comparison between the two versions of MAXCCLUS

When generalising in the reference clustering experiment (see chapter 1), MAXCCLUS performs a simple generalisation. It only adds correct instances (instances that have the same category as the original cluster's category) and it only generalises by dropping descriptors. Therefore there is a limit for MAXCCLUS's simple generalisation. There is a possibility of getting more general rules and clusters by using more powerful algorithm than MAXCCLUS. MAXCCLUS needs a generalisation that can search the space of the cluster's characterisation words exhaustively, looking through all the possible combinations, and can accept incorrect instances (instances that have categories different than the original cluster's category) providing that the cluster's accuracy is not sacrificed.

3.2.1 MAXCCLUS's two versions

Figure 3.1 shows two versions of MAXCCLUS. Both versions are the same except for the generalisation part: one uses the simple generalisation and the other uses an exhaustive search generalisation. Both versions start by constructing clusters using the textual data annotated with each instance (gene). From these constructed clusters, the two versions select the good clusters (the significant accurate clusters) using the microarray experimental data. Then both versions attempt to generalise the good clusters, either by using the simple generalisation, or by using the exhaustive search generalisation. From these generalised clusters both versions select the cover clusters to get the generalised cover clusters. From these generalised cover clusters, both versions construct the rules that characterise the clusters.

The following sections describe the exhaustive search generalisation method.

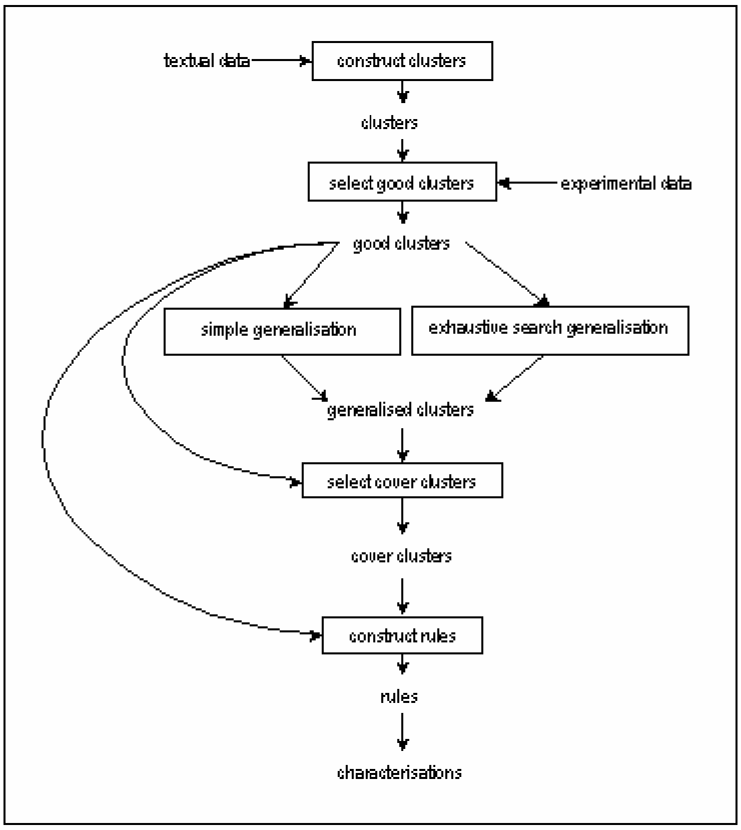


***Figure 3.1**: The two versions of MAXCCLUS. One uses the simple generalisation (the reference clustering experiment) while the other uses the exhaustive search generalisation (this chapters clustering experiment).*

3.2.2 Simple generalisation – algorithm

The simple generalisation that MAXCCLUS uses is based on only dropping words from the rules that describe the cluster, so that the new generalised rule will cover more instances. This generalisation does not allow negative instances (instances that have different category than the cluster's category) to be covered by the generalised rule.

3.2.3 Exhaustive search generalisation

The input to the exhaustive generalisation algorithm is a good cluster (which is a significant *and* pure cluster). The output is the best generalisation of this cluster. The algorithm searches for the best generalised rule, by attempting to generalise one of the cluster's rules (each rule consists of words that describe the cluster's instances). The best generalised rule for a cluster is a rule that has the same category as the cluster and describes more instances than any other rule of the cluster, subject to a constraint on accuracy (described later in section 3.2.3.2). The best generalised rule for the cluster could be one of the original rules of the cluster, or it could be a new rule that also describes the cluster. From the best generalised rule (the best generalisation of the cluster), MAXCCLUS can construct a new cluster, which is bigger than the original cluster (describes more instances than the original cluster).

The exhaustive generalisation algorithm creates a set of *words* that are in the characterisation of the cluster. *Words* contains all the *common* words and all the *frequent* words of the cluster. The *common* words of a cluster are the words that are represented in *all* the instances of the cluster; the *frequent* words of a cluster are the words that are represented in *most* of the instances of the cluster. The exhaustive generalisation algorithm (figure 3.2) searches exhaustively by creating all combinations (all subsets) of the set of words that are in the characterisation of the cluster.
This project implemented two versions of the exhaustive generalisation algorithm, the less restricted version and the more restricted version. They are

similar to each other; they only differ in line 04 of the algorithm (figure 3.2). The more restricted version includes the "a superset of" constraint, and the less restricted version does not.

The exhaustive generalisation algorithm starts by choosing the rule that has fewest words from the rules that describe the cluster. Initially this rule becomes the best rule. From each combination (subset) of *words*, except the empty set ∅, and combinations that are equal to (for the less restricted version), or supersets of (for the more restricted version) the original rules of the cluster, this algorithm creates a new rule. This new rule has the same category of the cluster. The algorithm evaluates the new rule against the best rule by using the *IsBetterThanOrEqualTo* method (described in the next section).

If the new rule is evaluated as better than or equal to the best rule, and if the new rule covers more correct instances than the best rule, or the new rule has fewer words than the best rule, then the new rule becomes the best rule (Note: "*correct instances*" are the instances that have the same category as the rule). Choosing the rule that has fewer words results in a simpler rule to describe the cluster. When the exhaustive search finishes, this algorithm returns the best rule.

```
01: Let the cluster's rule that has fewest words, to become the bestRule.
02: Let words to be the set of common and frequent words of the cluster.
03: For each subset of words (except ∅):
04:     If the subset is not [a superset of] one of the original rules of the cluster:
05:             create newRule from this subset.
06:             If newRule isBetterThanOrEqualTo bestRule:
07:                     If newRule covers more correct instances than bestRule:
08:                             Let newRule to become the bestRule.
09:                     Else: choose between newRule and bestRule,
                                the rule that has fewer words, to become the bestRule.
10: return bestRule.
```

Figure 3.2*: The exhaustive generalisation algorithm*

3.2.3.1 IsBetterThanOrEqualTo – three evaluating criteria

The input to this method is two rules: the best rule found so far, and the new rule that the exhaustive generalisation algorithm trying to evaluate with respect to the current best rule. *IsBetterThanOrEqualTo* returns *true* if the new rule is better than or equal to the best rule. The project implements two versions of the *IsBetterThanOrEqualTo* method, depending on two approaches for evaluating accuracy.

3.2.3.2 Two criteria: Accuracy and Minimum-accuracy

The *IsBetterThanOrEqualTo* method has three conditions that must be satisfied to return the value *true*.

The first two conditions are (see figures 3.3 and 3.4):

1. The number of *newRule*'s correctly covered instances is greater than or equal to the number of *bestRule*'s correctly covered instances.
2. The correctly shared instances between *newRule* and *firstRule* are greater than or equal to a percentage value (the implementation uses 70%).

There are two forms of the third condition referred to as *accuracy* and *minimum-accuracy*.

Accuracy is the accuracy of the rule and is calculated as:

$$A = \frac{C}{I}$$

Where A is the accuracy, C is the number of correct instances the rule covers, and I is the number of all instances the rule covers.

Minimum-accuracy is the accuracy specified by the user of MAXCCLUS. It determines the minimum accuracy for a cluster that MAXCCLUS should handle.

For the *accuracy* form, the third condition is (figure 3.3):

3. *newRule*'s accuracy is greater than or equal to *bestRule*'s accuracy.

For the *minimum-accuracy* form, the third condition is (figure 3.4):

3. *newRule*'s accuracy is greater than or equal to the minimum accuracy for the clusters (usually minimum accuracy is 95% pure). The user of MAXCCLUS specifies the value of the minimum-accuracy.

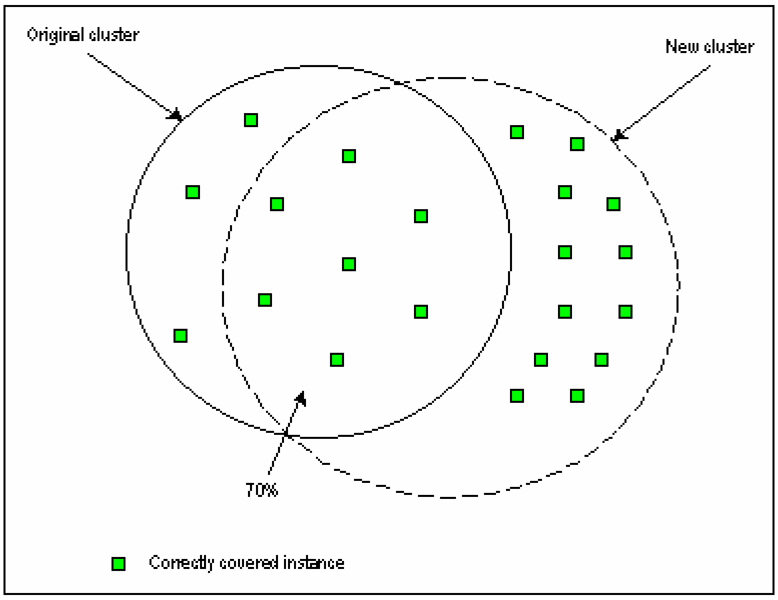


Figure 3.3*: A new cluster, that is created from the best generalised rule, must cover at least 70% of the correctly covered instances of the original cluster. The new cluster’s accuracy is equal to the original cluster’s accuracy (in this example both accuracies are 100%).*

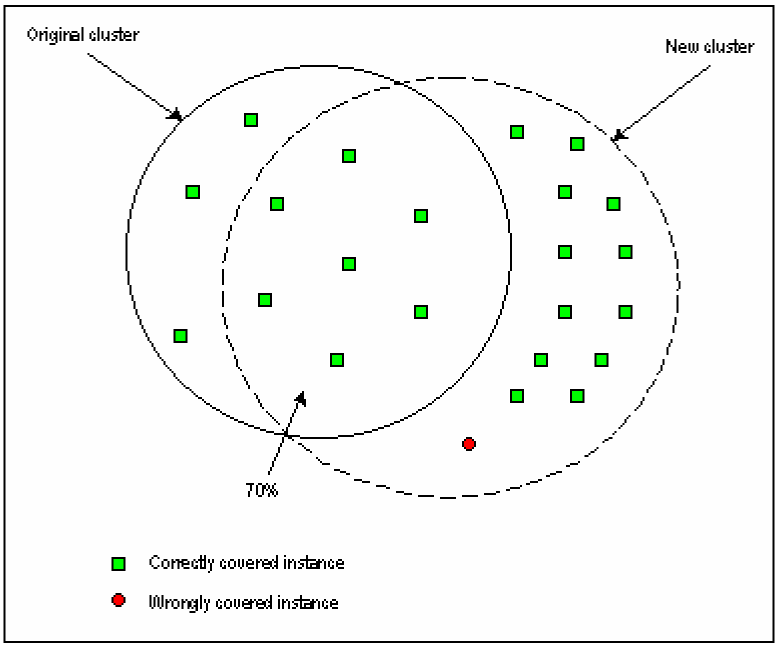


Figure 3.4*: A new cluster, that is created from the best generalised rule, covers more correct instances than the original cluster. The new cluster must also cover at least 70% of the correctly covered instances of the original cluster. Also the new cluster's accuracy must be greater than or equal to minimum accuracy. Supposing that the minimum accuracy is 95%, in this example the new cluster's accuracy is equal to the minimum accuracy.*

3.3 Experimental results

Because the two versions of MAXCCLUS are identical except for the generalisation part, both versions have the same significant clusters and the same cover clusters. Therefore the results section compares only the generalised cover clusters of the two versions.

The results shown in the figures below are for running MAXCCLUS on 23 of the microarray experimental data sets (data set: 3, 6, 7, 8, 10, 11, 12, 13, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 35, 36, and 40), using exhaustive search generalisation instead of the simple generalisation. Figures 3.5 to 3.14 show the results for the number of clusters, the number of rules, the number of words in the rules, the number of words per rule, and the execution time.

We will refer in the remaining of the chapter to the results using accuracy as ***AC***, and using minimum accuracy as ***MA***.

The clusters

Figures 3.5 to 3.7 show the changes to the number of clusters when MAXCCLUS uses the exhaustive search generalisation instead of the simple generalisation. The figures show the changes to the total number of clusters, the number of retained clusters, and the number of new clusters.

On average, the number of clusters when MAXCCLUS uses the exhaustive search generalisation is about 5% more than when MAXCCLUS uses the simple generalisation (figure 3.5).

When MAXCCLUS uses the exhaustive search generalisation, on average, 45% of the clusters are *retained* from the clusters found when MAXCCLUS uses the simple generalisation (figure 3.6).

When MAXCCLUS uses the exhaustive search generalisation, on average, about 56% of the clusters found are *different* from the clusters found when MAXCCLUS uses the simple generalisation (figure 3.7).

On average, almost no clusters found from clustering when MAXCCLUS uses the simple generalisation, are *subsumed* by new clusters found from clustering when MAXCCLUS uses the exhaustive search generalisation.

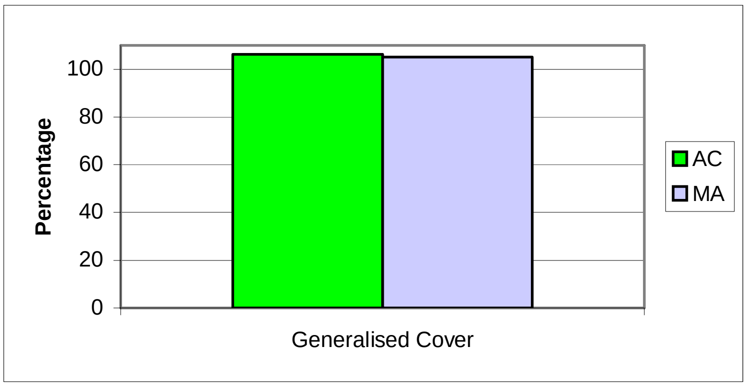


Figure 3.5: Average change to the total number of clusters

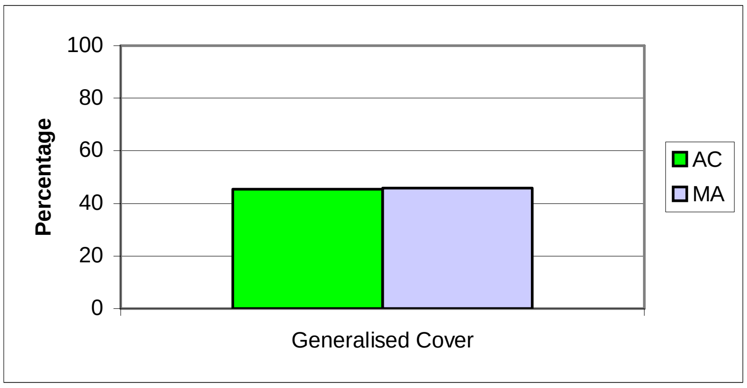


Figure 3.6: Average number of retained clusters

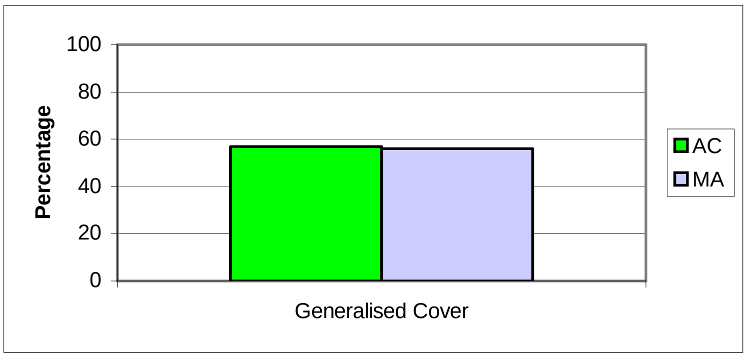


Figure 3.7: Average number of new clusters

The instances

When MAXCCLUS uses the exhaustive search generalisation it describes the *same* instances as it does when uses the simple generalisation.

The rules

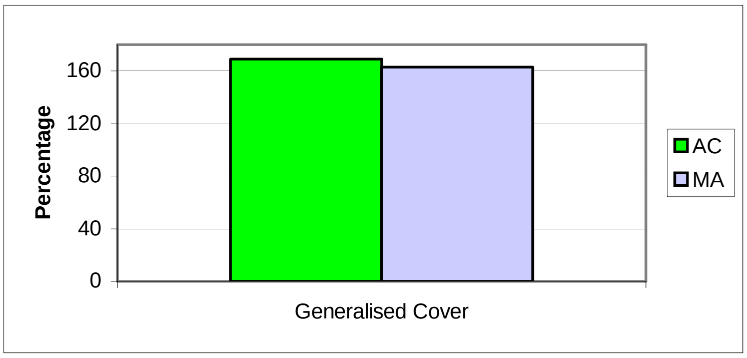
Figures 3.8 to 3.10 show the changes to the number of rules when MAXCCLUS uses the exhaustive search generalisation instead of the simple generalisation. The figures show the changes to the total number of rules, the number of retained rules, and the number of new rules.

On average, the number of rules when MAXCCLUS uses the exhaustive search generalisation is about 65% more that when MAXCCLUS uses the simple generalisation (figure 3.8), but the Accuracy version of the algorithm (AC) has a somewhat wider range.

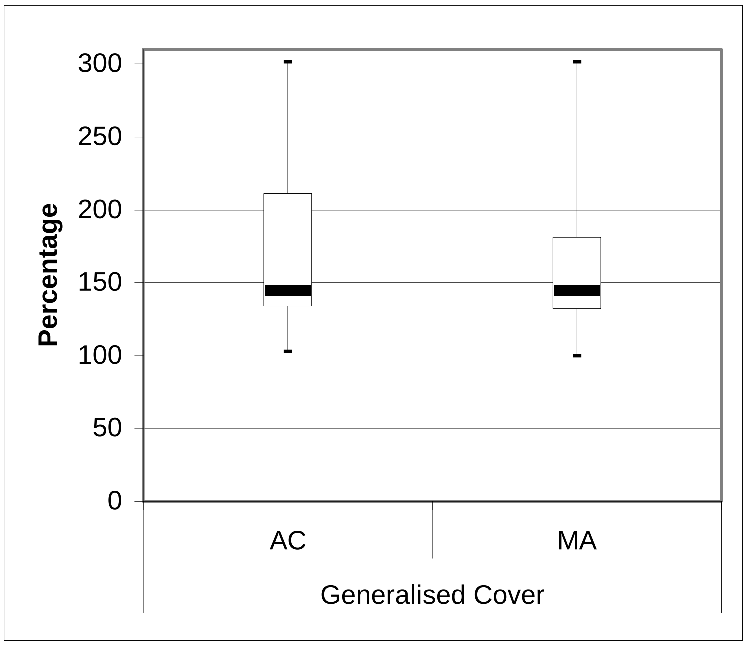
When MAXCCLUS uses the exhaustive search generalisation, about 81% of the rules are *retained* from the rules found when MAXCCLUS uses the simple generalisation (figure 3.9).

When MAXCCLUS uses the exhaustive search generalisation, on average, 46% of the rules are *different* from the rules found when MAXCCLUS uses the simple generalisation (figure 3.10).

On average, almost no rules found from clustering when MAXCCLUS uses the simple generalisation, are *subsumed* by new rules found from clustering when MAXCCLUS uses the exhaustive search generalisation.



a: Average



b: Distributions

Figure 3.8: Change to the total number of rules

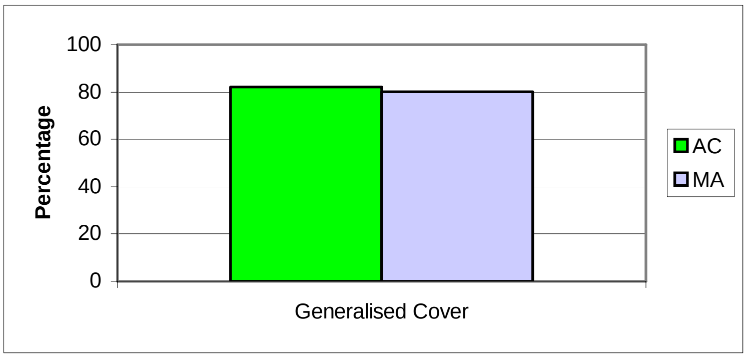


Figure 3.9: Average number of retained rules

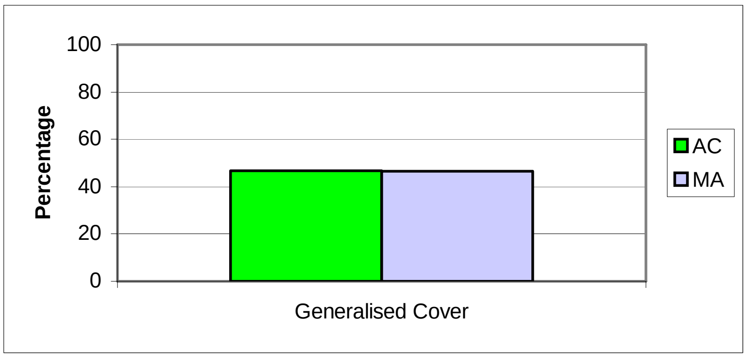


Figure 3.10: Average number of new rules

The rules' words

Figures 3.11 to 3.13 show the changes to the number of rules' words when MAXCCLUS uses the exhaustive search generalisation instead of the simple generalisation. The figures show the changes to the total number of rules' words, the number of retained rules' words, and the number of new rules' words.

On average, the number of rules' words when MAXCCLUS uses the exhaustive search generalisation is about 13% more than that when it uses the simple generalisation (figure 3.11).

When MAXCCLUS uses the exhaustive search generalisation, about 96% of the rules' words are *retained* from that found when MAXCCLUS uses the simple generalisation (figure 3.12).

When MAXCCLUS uses exhaustive search generalisation, on average, about 15% of the rules' words are *different* from the rules' words found when it uses the simple generalisation (figure 3.13).

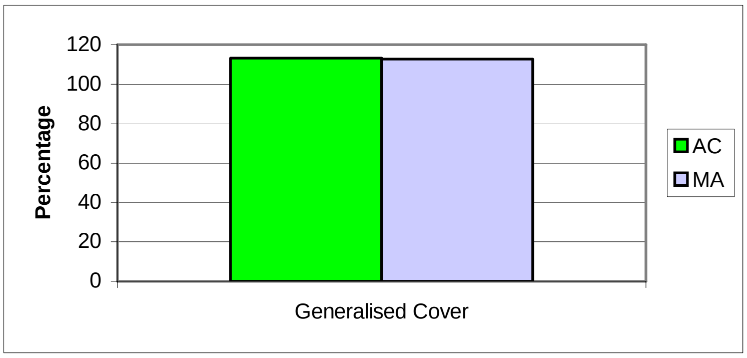


Figure 3.11: Average change to the total number of rules' words

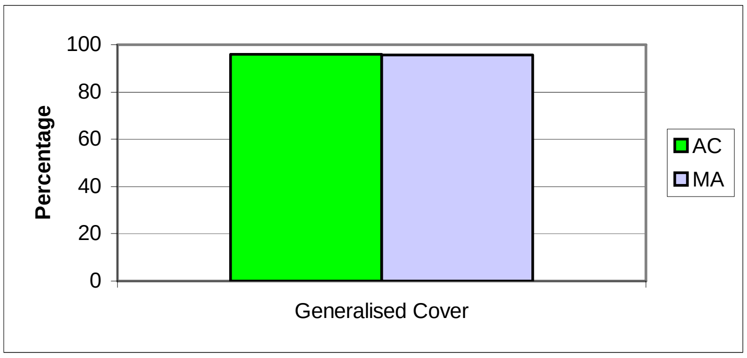


Figure 3.12: Average number of retained rules' words

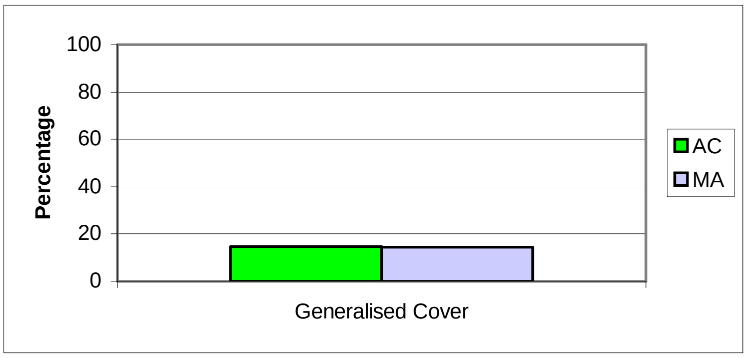


Figure 3.13: Average number of new rules' words

The words per rule

Figure 3.14 shows the changes to the total number of words per rule when MAXCCLUS uses the exhaustive search generalisation instead of the simple generalisation.

On average, the number of words per rule when MAXCCLUS uses the exhaustive search generalisation is almost the same (only about 2% more) as that when MAXCCLUS uses the simple generalisation

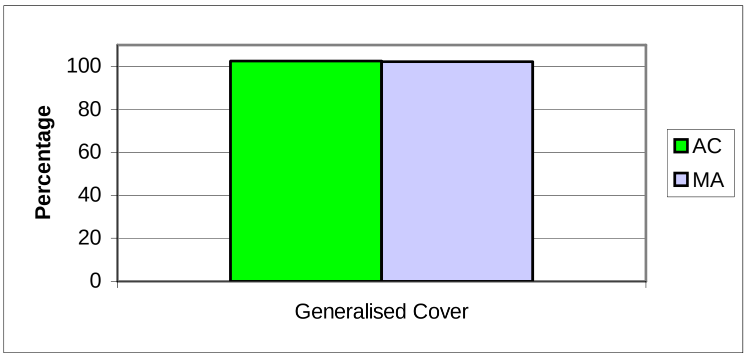


Figure 3.14: Average change to the number of words per rule

The execution time

Figure 3.15 shows the changes to the clustering execution time when MAXCCLUS uses the exhaustive search generalisation instead of the simple generalisation.

When MAXCCLUS uses the exhaustive search generalisation using AC or MA, on average, the execution time is much more than that when MAXCCLUS uses the simple generalisation. On average, the execution time, when clustering using AC, is slightly more than that when using MA; about 7.9 times longer when using AC, but only about 7.6 times longer when using MA, compared with the execution time when MAXCCLUS uses the simple generalisation.

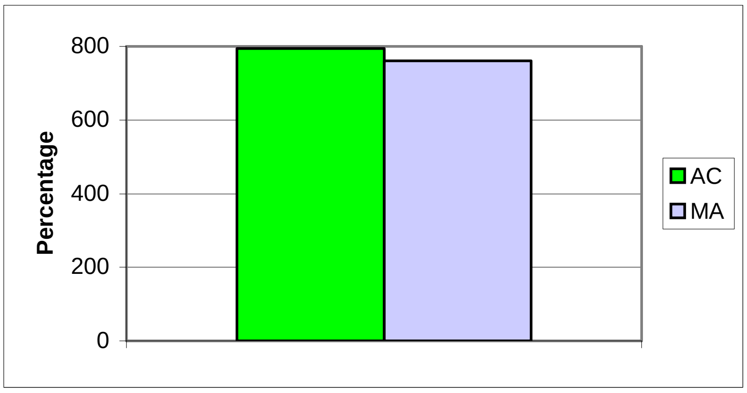


Figure 3.15: Average change to the clustering execution time

3.4 Analysis and discussion

The following sections analyse and discuss the results shown in the previos section.

The MAXCCLUS clustering algorithm has a simpler and faster algorithm for generalisation, compared with more complex and much slower exhaustive search algorithm for generalisation of the modified clustering algorithm. We had expected that the exhaustive algorithm would give noticeably better clusters and rules. However, the results show that MAXCCLUS, using its simple generalisation algorithm, behaved very well with most of the microarray experimental data sets.

The modified clustering algorithm created new clusters, from the best rules, and then it added these new clusters to the original clusters giving a bigger cluster set. Creating and adding the new clusters to the original cluster set, had little effect on the characterisation behaviour of MAXCCLUS, as the results show. There was little change in the number of clusters in the generalised cover and no change to the instances. Therefore the original MAXCCLUS generalisation algorithm, with its simplicity and speed, proved to be preferable to the more complex and more time consuming modified clustering algorithm using either of the two different criteria *accuracy* and *minimum accuracy*.

Although the exhaustive search generalisation allowed adding negative instances that have different category from the cluster, the number of instances did not increase. This could be because of the high user accuracy value (95%), which prevents adding lots of negative instances to keep the high purity of the clusters. We believe that if the user uses low accuracy value (say 70%) then more negative instances would be added to the clusters, but then the clusters would not be interesting because of the low accuracy value.

3.5 Conclusion

Using simple generalisation approach for MAXCCLUS is better, for the microarray data sets used for clustering experiments, than using an exhaustive search generalisation. The simple generalisation can describe all the instances that the exhaustive search generalisation can describe, and it is much faster than the exhaustive search generalisation, at least for the microarray data sets that MAXCCLUS used in this project.

In the future work we will experiment with other data sets (something that was not available during the period of this project) by running this chapter's experiment and the reference experiment and compare their results as we did in this chapter. We expect that the simple generalisation will do as well as the exhaustive search generalisation in describing instances and it will be faster, as it did with the microarray data sets that used in this project.

3.6 Recommendations

For the reasons mentioned in the conclusion section, we recommend that MAXCCLUS keeps its simple generalisation and there is no need to replace it with an exhaustive search generalisation, at least for the microarray data sets that MAXCCLUS used in this project with the given high user accuracy.

Chapter 4

Operons

This chapter explores running MAXCCLUS to cluster operons instead of genes. An operon is a group of genes that behave in the same way regarding regulation. So we expect that within a microarray experiment, a set of genes that belong to an operon will respond to physical and/or chemical effects in the same way. This means an operon is already a cluster of genes, even before using a clustering algorithm (Figure 4.1).

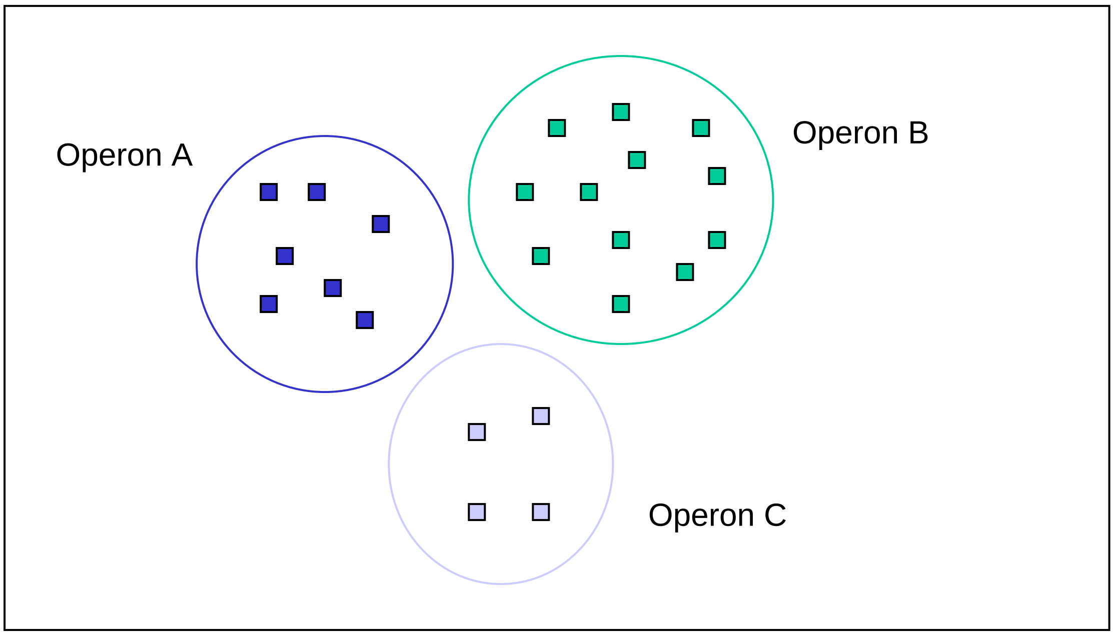


Figure 4.1: An operon is a group of genes that behave in the same way.

By clustering operons instead of genes we expect to reduce the number of clusters produced by MAXCCLUS, and we will find out if the clusters found by clustering genes were scientifically interesting, or if they were clusters of genes that represent operons.

When clustering genes, MAXCCLUS performs a significance test on each cluster it finds. The question “Is this cluster big enough to be significant?” is based on an assumption that all the genes in the cluster were independent of each other. When performing the significance test, MAXCCLUS does not consider that some of the genes belong to operons, and therefore ignores their dependency on each other. When clustering operons, we are questioning the significance of the clusters found by MAXCCLUS when clustering genes.

Because there are not many big operons (figure 4.2), it is not the case that MAXCCLUS could find lots of gene clusters that are operons. But MAXCCLUS can find gene clusters that contain a number of genes that belong to the same operon.

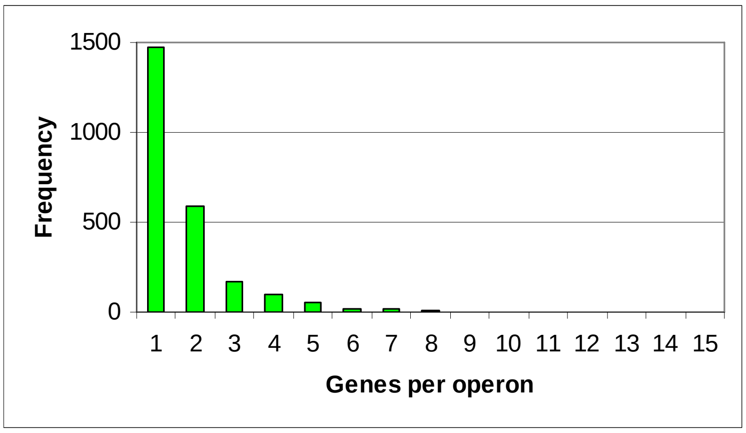


Figure 4.2: The frequency of genes per operon.

Having genes that belong to the same operon, in a cluster found by MAXCCLUS, changes the calculations of the significance test.

4.1 Example

For example, suppose, when clustering genes, MAXCCLUS finds a cluster of 10 genes (figure 4.3 (a)). It may be that this cluster is statistically significant when viewed as having 10 instances. But in another view, the operons view, this cluster consists of only 6 instances (figure 4.3 (b)). In the operons view, it consists of one operon that has three genes, two operons each having two genes, and three single genes (we can imagine these three genes as three operons, each operon having one gene). With the operons view, the same cluster now has 6 instances as operons, and may be that this cluster is statistically insignificant when viewed as having 6 instances, and it should be treated by MAXCCLUS as if it was generated by chance and therefore should be ignored and not considered as a significant cluster.

From this example, we expect that, when clustering operons, MAXCCLUS will find fewer significant clusters. In order to explore significance, we run MAXCCLUS to cluster operons in this chapter.

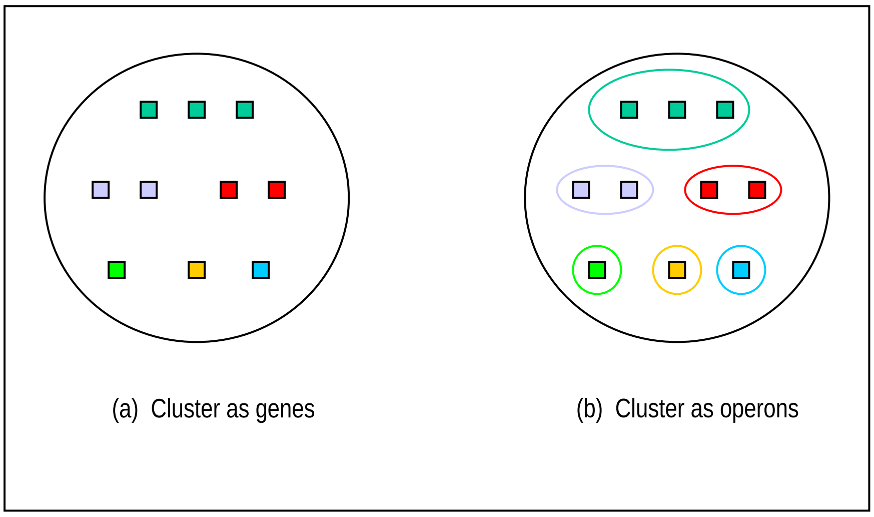


Figure 4.3: Two views for a cluster: (a) as genes, (b) as operons.

4.2 Preparing the data

To be able to run MAXCCLUS on operons instead of genes, we need to prepare the data that MAXCCLUS uses. We have two kinds of data to be used by MAXCCLUS: first the textual data, and second the microarray experimental data.

4.2.1 Preparing the textual data

The textual data for operons was prepared by creating the operon descriptor table and storing it to a file to be ready for use by MAXCCLUS. This table contains a number of rows; each row contains an operon's id associated with the operon's descriptors' sets. We created each operon's descriptor set out of its genes descriptor sets. There are two methods to do this. We implemented the second one only.

First method

The first method is to take the *intersection* of the descriptor sets of the genes in the operon; this gives us all the words that are true of *all* the operon's genes. The operon descriptor sets that we get from this method are not rich enough to be useful for our experiment; therefore we do not use it.

Second method

The second method is to take the *union* of the descriptor sets of the genes in the operon; this gives us the words that are true of *any* of the operon's genes. This method reflects the functional dependency between an operon's genes. As we extracted the gene descriptors from a protein database (the *Swiss-Prot*), the genes descriptors describe the protein that each gene codes for. Because the operon's genes are functionally related, then for each operon, its descriptor set needs to be the union of the descriptor sets of the genes in this operon to keep the knowledge about all the operon's genes. From that, the second method creates a richer set of descriptors that describes the operon; therefore, we chose the second method to create the operon descriptor set.

4.2.2 Preparing the microarray experimental data

The operon microarray experimental data is a file that contains a number of rows, in which each row contains an operon's id associated with the operon's category. For each microarray experiment, the operon microarray experimental data is created from the gene microarray experimental data for this microarray experiment. The problem is that not all genes of every operon have the same category; some genes are not labelled "*Up*" or "*Down*" and it is possible that two genes in the same operon could have opposite categories. Therefore, it is not clear how to assign categories to operons. We took three approaches to creating and implementing the operon microarray experimental data.

First approach (OP100)

The first approach is to accept an operon (include its id in the operon microarray experimental data) if 100% of its genes are present in the gene microarray experimental data *and* if 100% of its genes are associated with the same category. We then associate this operon's id with the same category as its genes' category. If any genes are not present or their categories are mixed, we reject the operon so it will not be included in the operon microarray experimental data. We will refer to this approach as ***OP100***. We will also use this reference to refer to the results of the operon clustering experiment, which uses operon microarray experimental data created by using this approach.

Using this approach, we only accepted 58% of the operons that have genes in the gene microarray experimental data. Because this was too strict, we introduced the second approach with fewer constraints.

Second approach (OP66)

The second approach is to accept an operon if 66% or more of all of its genes' are present in the gene microarray experimental data and if 66% or more of all of its genes are associated with the same category. Then we associate this operon's id with the majority category of its genes. Otherwise we reject the operon. We will refer to this approach as ***OP66***. We will also use this reference to refer to the results of the operon clustering experiment, which uses operon microarray experimental data created by using this approach.

Using the second approach, we accepted 63% of the operons that have genes in the gene microarray experimental data. While the second approach is a little bit less strict than the first approach, the second approach is still strict. Therefore, we introduce the third approach that is looser in its constraints than the first and the second approaches.

Third approach (OP70)

The third approach is to accept an operon if any of its genes are present in the gene microarray experimental data as long as 70% or more of the genes that are present are associated with the same category. Then we associate this operon's id with this category. Otherwise, we reject the operon. We will refer to this approach as ***OP70***. We will also use this reference to refer to the results of the operon clustering experiment, which uses operon microarray experimental data created by using this approach.

Using the third approach we accepted 98.4% of the operons that have genes in the gene microarray experimental data. This third approach is the least strict of the three approaches regarding constrains for accepting operons in the operon's microarray experimental data.

4.3 Experimental results

The results shown in the figures below are for running MAXCCLUS on 20 sets of the operon microarray experimental data (data set: 6, 7, 10, 11, 12, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 35, 36, and 40), and using the operon textual data.

Figures 4.12 to 4.19 show the results for the number of rules, the number of words in the rules, and the number of words per rule. The figures compare the results on the operon data with the corresponding results on gene data. The comparison between the operons data and the genes data is direct.

Figures 4.4 to 4.8 show the results for the number of clusters and the number of instances. However, for these figures the comparison between operons and genes cannot be direct, because the instance ids of genes are different from those of operons. So to be able to compare, we substituted each operon’s id by the ids of its genes, and substituted the clusters of operons by the clusters that contain the genes that these operons map to. By doing this, we can now make a valid comparison between the clusters and instances from the operon data and the clusters and instances from the gene data.

The clusters

Figures 4.4 and 4.5 show the changes to the number of clusters when the genes are replaced by operons. The figures show the changes to the total number of clusters and the number of clusters subsumed by new clusters.

When using any of the approaches OP100, OP66 or OP70, on average, there are fewer clusters when clustering operons than when clustering genes, specially the significant clusters (figure 4.4).

For any of the cluster sets (significant, cover, or generalised cover cluster set), on average, there are fewer clusters when using OP100 than when using OP66, which in turn, are fewer clusters than when using OP70.

When using any of the approaches OP100, OP66 or OP70, on average, the significant cluster set has the highest decrease in the number of clusters.

When clustering operons, on average, almost no clusters are *retained* from the clusters found when clustering genes and almost all the clusters are *different* from the clusters found when clustering genes. On average, very few or no clusters found when clustering genes are *subsumed* by new clusters found when clustering operons, except for OP70 where there is a noticeable percentage of the number of clusters found when clustering genes *subsumed* by the new clusters found when clustering operons, especially the significant clusters set (figure 4.5).

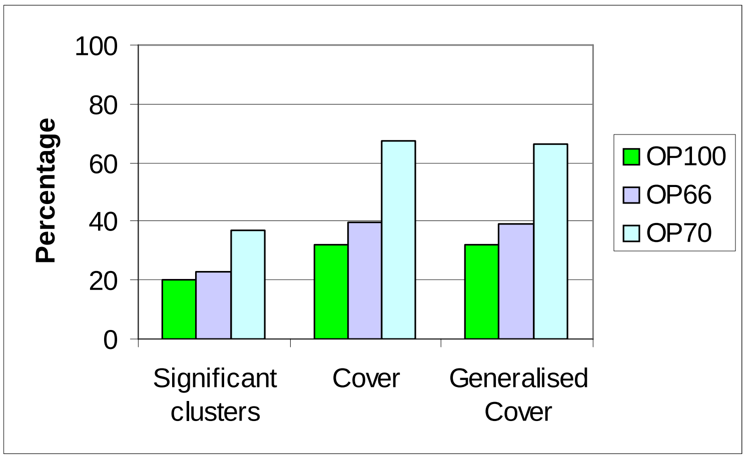


Figure 4.4: Average change to the total number of clusters

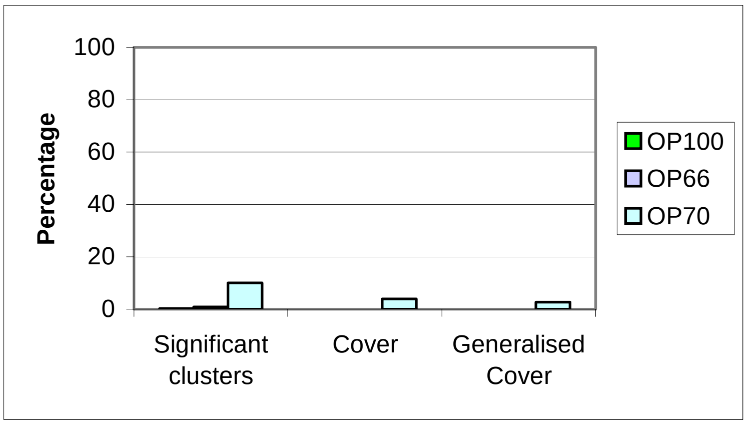


Figure 4.5: Average number of clusters subsumed by new clusters

The instances

Figures 4.6 to 4.8 show the changes to the number of instances when the genes are replaced by operons. The figures show the changes to the total number of instances, the number of retained instances, and the number of new instances.

When using any of the approaches OP100 or OP66, on average, there are fewer instances when clustering operons than when clustering genes, specially the significant clusters (figure 4.6). In contrast, on average, when using approach OP70, there are more instances when clustering operons than when clustering genes.

For any of the cluster sets, significant, cover, or generalised cover cluster set, on average, there are fewer instances when using OP100 than when using OP66, which in turn, there are fewer instances than when using OP70.

When clustering operons, on average, about 40% to 60% of the instances are *retained* from the instances found when clustering genes (figure 4.7). For all cluster sets, significant, cover, or generalised cover cluster set, on average, the retained instances when using OP100 are fewer than that when using OP66, which in turn, are fewer than that when using OP70.

For all the cluster sets, significant, cover, and generalised cover cluster set, on average, the *new* instances that are described when clustering operons, which were not described when clustering genes, are many more when using OP70 than when using OP66, which in turn, are a little bit more than when using OP100 (figure 4.8).

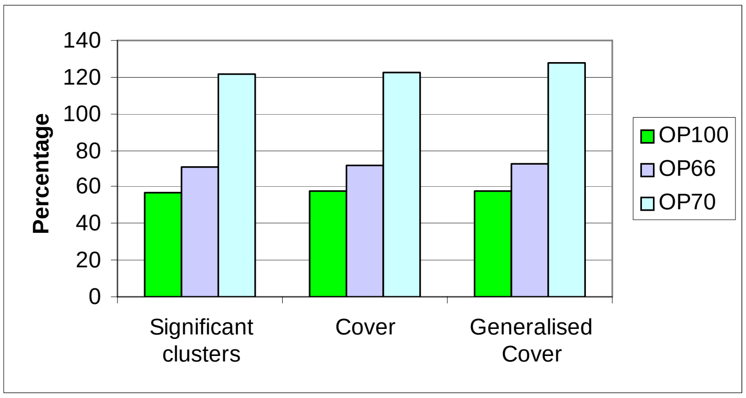


Figure 4.6: Average change to the total number of instances

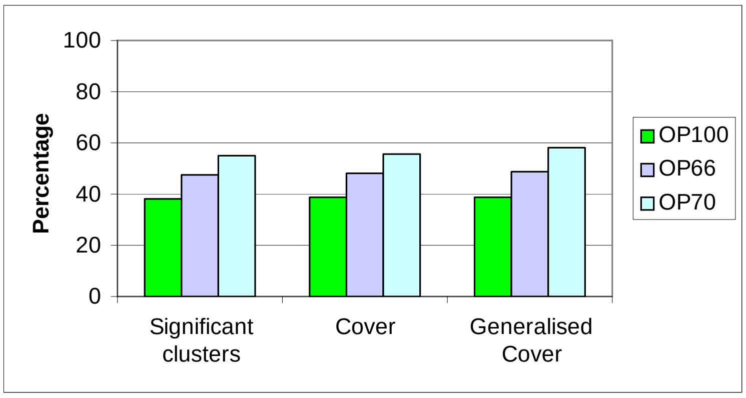


Figure 4.7: Average number of retained instances

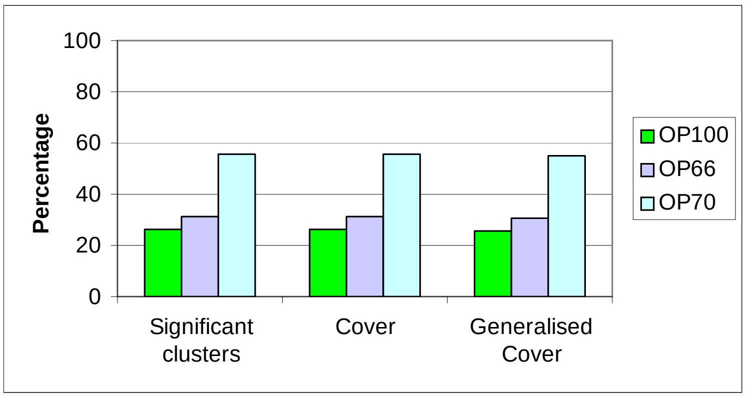


Figure 4.8: Average number of new instances

Figure 4.9 shows the average percentage of the predicted instances (genes) when clustering operons. The predicted genes are the genes that exist in the gene set mapped from the operons after clustering operons but do not exist in the gene experimental data sets. The percentage of the predicted genes $P\%$ for each data set is calculated as the percentage ratio of the number of predicted genes to the number of the genes from the operons clustering:

$$P\% = \frac{|Gp|}{|Go|} \times 100$$

Where Gp represents the predicted genes, and Go represents the genes from operon clustering.

There are no genes predicted when using OP100. On average, the predicted genes when using OP66 are fewer than when using OP70 (figure 4.9).

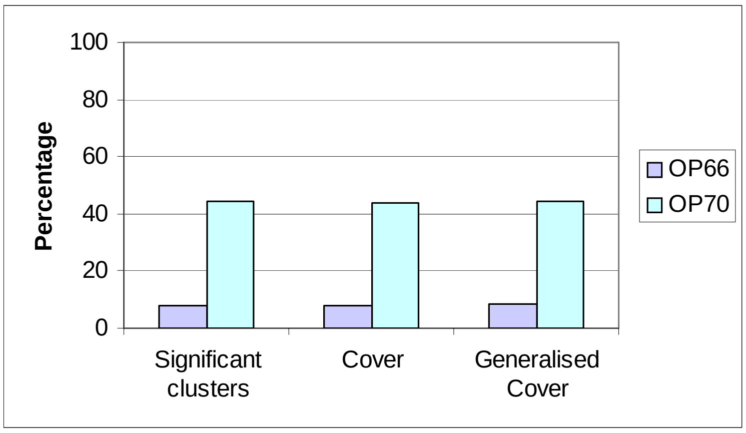


Figure 4.9: Average number of predicted instances

Figure 4.10 shows the average of the correctly predicted genes when clustering operons. The correctly predicted genes are the predicted genes whose real categories are near their predicted categories. A correctly predicted *Up* gene is a gene that its expression ratio is between 1.4 and 2.0 (excluding) and was predicted to be *Up*, while a correctly predicted *Down* gene is a gene that its expression ratio is between 0.5 and 0.7 (excluding) and was predicted to be *Down*. The percentage of the correctly predicted genes $C\%$ for each data set is calculated as the percentage ratio of the number of correctly predicted genes to the number of the predicted genes:

$$C\% = \frac{|C_{Gp}|}{|Gp|} \times 100$$

Where C_{Gp} represents the correctly predicted genes, and again, Gp represents the predicted genes.

There are no correctly predicted genes when using OP100 because there are no predicted genes. On average, the correctly predicted genes when using OP66 are more than that when using OP70 (figure 4.10).

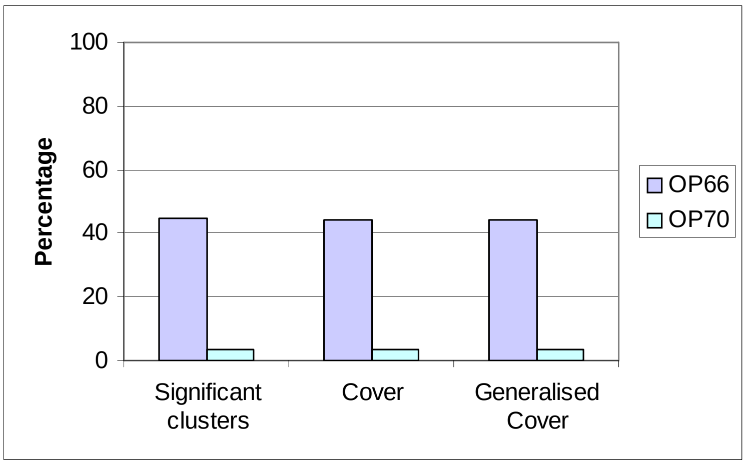


Figure 4.10: Average number of the correctly predicted instances

The Instances with respect to the experimental data

Figures 4.6 to 4.8 show the results for all the genes in the clustered operons. But since this includes genes that were not clearly *Up* or *Down* in the experimental data sets, the comparisons are not completely fair. An alternative is to confine the comparisons to just the *Up* and *Down* Genes in the data sets.

Figure 4.11 shows the number of *Up* and *Down* genes described by the rules when clustering operons.

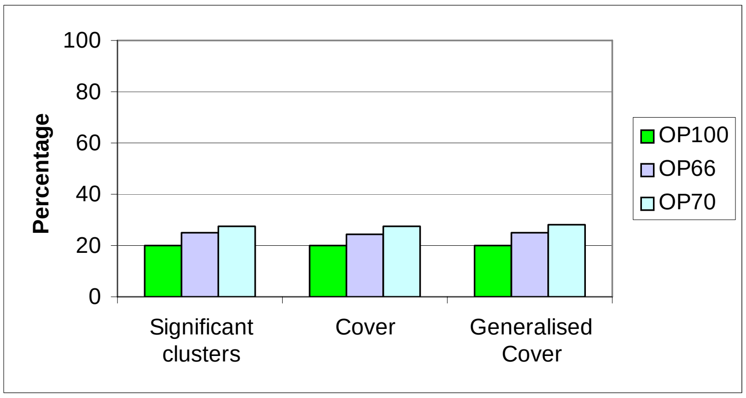
The percentage of the *Up* and *Down* described genes $D_{ud}\%$ for each data set is calculated as the percentage ratio of the number of described genes that have categories *Up* or *Down* when clustering operons:

$$D_{ud}\% = \frac{|Go_{ud}|}{|G_{\exp}|} \times 100$$

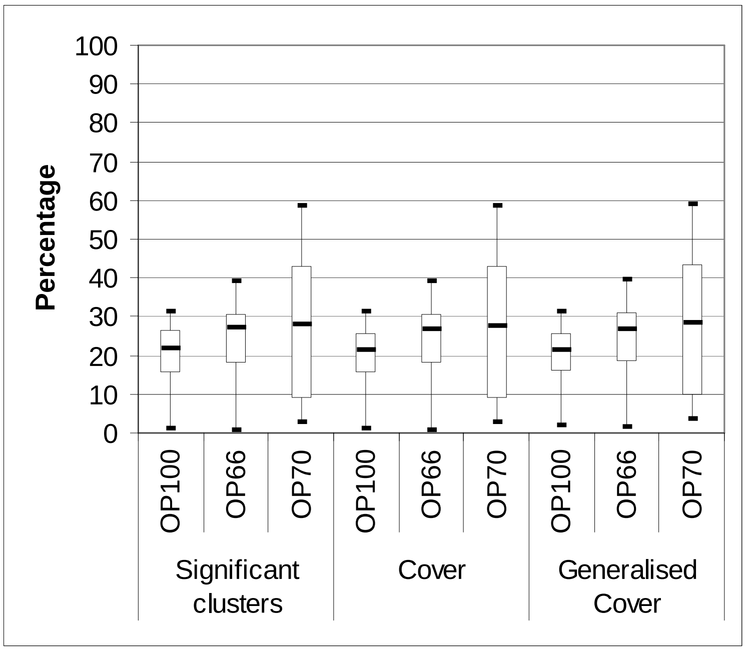
Where Go_{ud} represents the genes from operon clustering whose categories are *Up* or *Down*, and $G_{\exp}$ represents the genes from gene experimental data.

Figure 4.11 shows these results for the three clustering approaches (OP100, OP66, and OP70). It can be compared with figure 4.6.

When clustering operons, for any cluster set, significant, cover, generalised cover cluster set, the *Up* or *Down* described instances are fewer when using OP100 than when using OP66, which in turn, is fewer than when using OP70. The distributions show that there is a relatively wide variation across the data sets especially for OP70.



a: Average



b: Distributions

Figure 4.11 : Number of *Up* and *Down* instances described.

The rules

Figures 4.12 to 4.15 show the changes to the number of rules when the genes are replaced by operons. The figures show the changes to the total number of rules, the number of retained rules, the number of rules subsumed by new rules, and the number of new rules.

Using any of the approaches OP100, OP66 and OP70, on average, there are fewer rules when clustering operons than when clustering genes (Figure 4.12). There are fewer rules when using OP100 than when using OP66, which in turn, are fewer than when using of OP70.

Using any of the approaches OP100, OP66 and OP70, on average, the significant cluster set has the highest decrease in the number of rules. When using OP70, on average, the cover cluster set has the lowest decrease in the number of rules.

When clustering operons, on average, only a few rules are *retained* from the clusters found when clustering genes (Figure 4.13). On average, OP70 retained fewer rules than OP100, which in turn, retained fewer rules than OP66.

On average, there are a reasonable number of rules found when clustering genes that are *subsumed* by new rules found by clustering operons (Figure 4.14). On average, OP70 has fewer rules subsumed than OP66, which in turn, has fewer rules subsumed than OP100. The distributions show that there is a relatively wide variation across the data sets for OP100 significant clusters.

MAXCCLUS replaced the lost rules by *new* ones (Figure 4.15). MAXCCLUS found a lot of new rules when clustering operons. On average, OP66 has fewer new rules than OP100, which in turn, has fewer new rules than OP70.

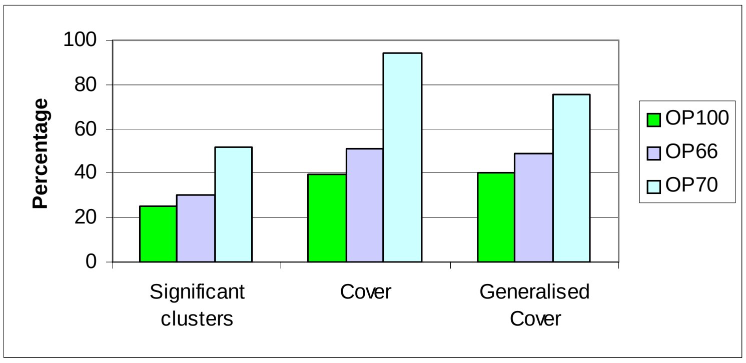


Figure 4.12: Average change to the total number of rules

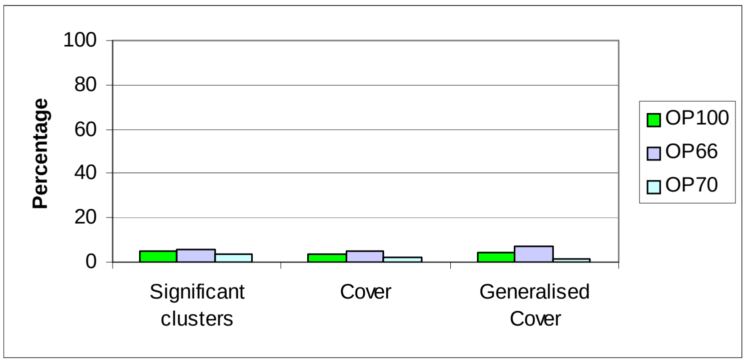
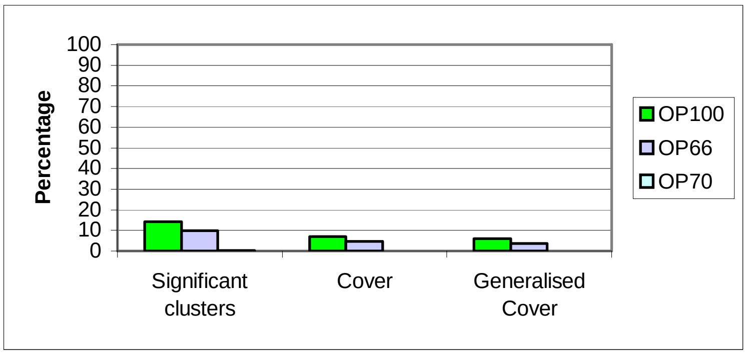
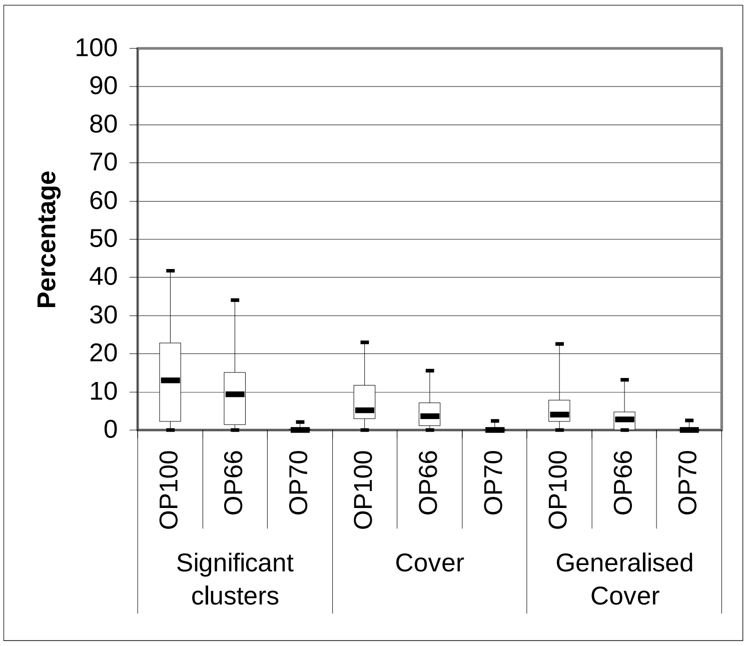


Figure 4.13: Average number of retained rules



a: Average



b: Distributions

Figure 4.14: Number of rules subsumed by new rules

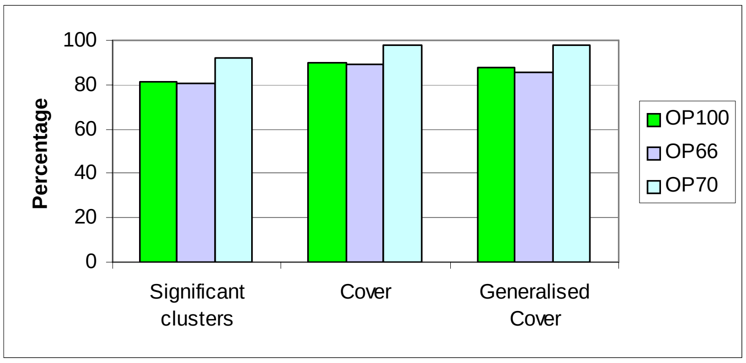


Figure 4.15: Average number of new rules

The rules’ words

Figures 4.16 to 4.18 show the changes to the number of words in the rules when the genes are replaced by operons. The figures show the changes to the total number, the number of retained words of the rules, and the number of new words.

On average, there are fewer words for rules when clustering operons than when clustering genes (figure 4.16).

On average, OP100 has higher decrease in the number of rules’ words than OP66, which in turn, has higher decrease in the number of rules’ words than OP70.

When using any of the approaches, OP60, OP66, or OP70, on average, the significant cluster set has the highest decrease in the number of rules’ words.

On average, OP100 *retained* fewer rules’ words than OP66, which in turn, returned fewer rules’ words than OP70 (figure 4.17).

MAXCCLUS replaced the lost words with *new* words (figure 4.18).

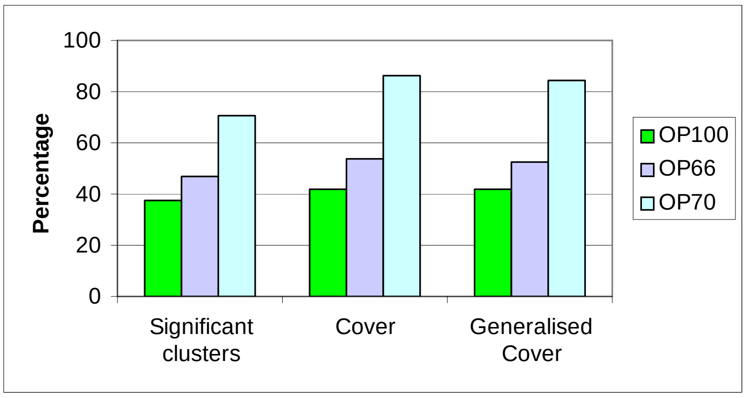


Figure 4.16: Average change to the total number of rules' words

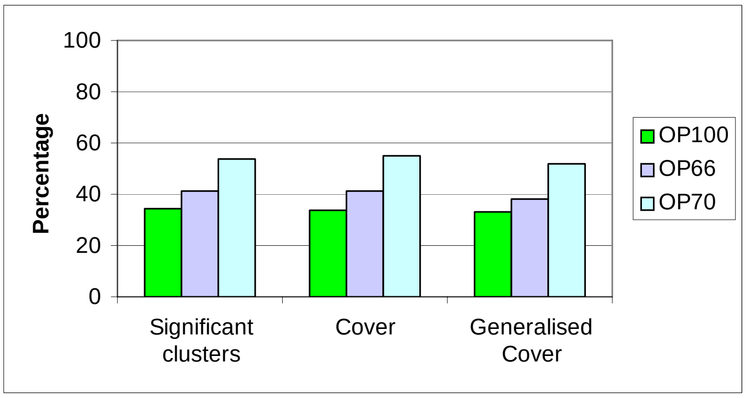


Figure 4.17: Average number of retained rules' words

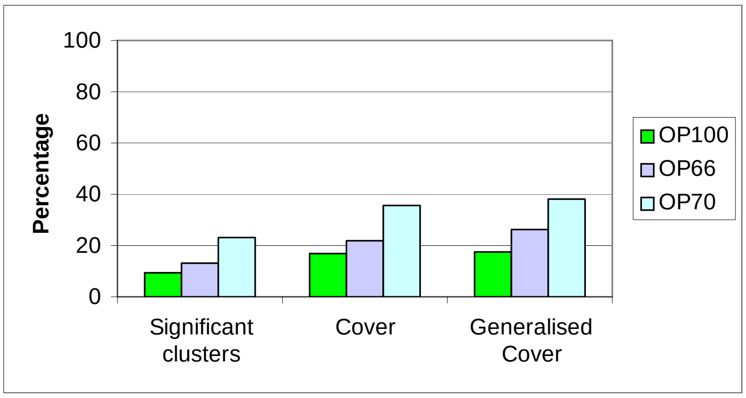


Figure 4.18: Average number of new rules' words

The words per rule

Figure 4.19 shows the changes to the number of words per rule when the genes are replaced by operons.

On average, when using the strictest approach OP100, the number of words per rule when clustering operons is less than that when clustering genes. When using the less strict approach OP66, on average, the number of words per rule when clustering operons is almost the same as that when clustering genes. When using weakest approach OP70, on average, the number of words per rule when clustering operons is more than that when clustering genes.

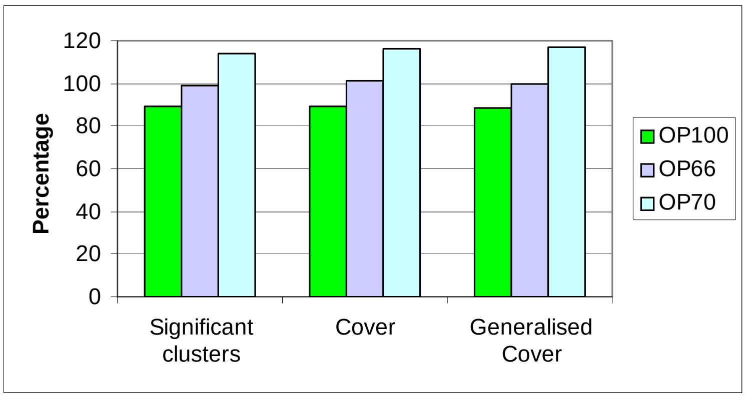


Figure 4.19: Average change to the number of words per rule

4.4 Analysis and discussion

The following sections analyse and discuss the results shown in the results section.

Clusters

Figure 4.4 shows that the average number of the clusters is smaller when clustering operons than it is when clustering genes, specially the significant clusters.

As we mentioned at the beginning of this chapter and showed in the example of section 4.1, it is not the case when clustering genes that MAXCCLUS finds lots of clusters that correspond directly to operons because there are not many big operons (figure 4.2). However it is the case that some clusters have several genes belonging to the same operon, as the example illustrates. Therefore we have fewer clusters when clustering operons than when clustering genes because MAXCCLUS finds fewer significant clusters when clustering operons than when clustering genes. Notice that when clustering operons, especially when using the OP70 approach, MAXCCLUS did not retain any of the gene clusters, there were very few gene clusters subsumed by operon clusters, and almost all the operon clusters were new. Dropping all the clusters and replacing them with new ones could be because of using the union of the descriptor sets of the genes when creating the descriptor sets of the operons. Or it could be because accepting operons that contain genes whose categories are not clearly *Up* or clearly *Down*, will produce a new set of clusters. Some of these new clusters subsumed old clusters of the gene clustering (figure 4.5).

Instances

For the instances, when clustering operons using OP100 or OP66, on average, MAXCCLUS is able to describe fewer genes than it can when clustering genes (figure 4.6). In contrast when using OP70, on average, MAXCCLUS is able to describe more genes than it can when clustering genes. Notice that with OP100, which is the strictest approach, MAXCCLUS describes fewer than the other approaches. This is because in OP100 we included in the operon microarray experimental data only the operons that are 100% pure, therefore we had fewer instances to cluster; in OP66 we included operons that are less than 100% pure, so there were more genes to be described.

An example is that we can have an operon of 10 genes of which nine genes have one category and the other one gene has a different category. By using the OP100 approach this operon (and the other similar operons) would be rejected. But by using the OP66 approach this operon would be accepted. Having a different category for the single gene could be a result of noise or wrong measurements while carrying out the microarray experiment. Therefore it is possible this gene would have the same category as the other genes in the operon if there was no noise or the measurements were right.

Including these extra operons, when using the OP66 approach, added extra genes that were not categorised as clearly *Up* or clearly *Down* genes (and so were not included in the gene microarray experimental data) although they might have been nearly *Up* or nearly *Down*. Including these extra operons helps to correct the noise or the wrong measurements in the microarray experimental data, and to predict that those genes whose categories were not clearly *Up* regulated should have been *Up* regulated, and predict that the ones that were not clearly *Down* regulated should have been *Down* regulated.

When using OP66, on average, about 8% of the genes that MAXCCLUS can describe are predicted genes (figure 4.9). About 45% of these predicted genes are correctly predicted (figure 4.10). Correctly predicted genes are the genes whose real categories are near *Up* or near *Down*, when MAXCCLUS predicted them to be *Up* or *Down* respectively.

By being even less strict in selecting operons in the operon microarray experimental data, as in OP70, MAXCCLUS is able to describe more genes than it can describe using the stricter two approaches. Because with OP70 more operons are accepted than with OP66, the new included operons may have many more genes with categories that are neither *Up* nor *Down*. These categories can be in-between the *Up* and *Down* categories but not so close to *Up* or to *Down*. For each operon, MAXCCLUS predicts that the operon's genes that are not in the gene experimental data are regulated *Up* (or *Down*) on the basis that the operon has 70% or more of its genes that are clearly regulated *Up* (or *Down*). To feel how less strict is the OP70 approach we can have an example.

In this example, we have an operon that has 15 genes. Suppose only one of these genes is in the gene experimental data and this gene has the *Up* category. Then when creating the operon experimental data with the OP70 approach, this operon will be accepted and it will be assigned the *Up* category even though it has 14 other genes with unknown categories of which some may not be even near *Up*, and may even be near *Down*. In contrast, OP66 would reject this operon because only about 7% of its genes are *Up* regulated.

When MAXCCLUS predicts that the 14 other genes of the operon in the above example have *Up* categories, even though some of them are far away from being *Up*, this prediction is not at all reliable. It is probable that some of the 14 genes have categories that are near *Up*, but it is much less probable for all of them to have categories that are near *Up*.

When using the OP70 approach in accepting operons in the operon microarray experimental data, on average, it seems that there is an increase in the number of genes that MAXCCLUS can describe when clustering operons than when clustering genes (figure 4.6). But this is not the case when we look only at the genes of the gene microarray experimental data (figure 4.11 OP70). In this result, when clustering operons MAXCCLUS covers fewer genes than when clustering genes (figure 2.4). In figure 4.6, the extra genes of OP70 described by MAXCCLUS are predicted genes (figure 4.9). Although, when using OP70 MAXCCLUS is able to predict many more genes than when using OP66 (figure 4.9), on average, the correctly predicted genes when using OP70 are much fewer than when using OP66 (figure 4.10). So, on average, when using OP70 most of the genes that MAXCCLUS predicts to have *Up* or *Down* categories are the result of incorrect prediction (on average, only 3% of the predicted genes are correctly predicted). In contrast, when using OP66 about 45% of the predicted genes are correctly predicted by MAXCCLUS, which make using the OP66 approach more sensible than using the OP70 approach in accepting operons in the operon microarray experimental data.

We have considered MAXCCLUS's predictions for genes when using OP66 or OP70, but not when using OP100 because for OP100 the operons are accepted in the operon microarray experimental data are 100% pure and each operon contains genes that are *all Up* or are *all Down*. Therefore, there is no prediction issue when using OP100.

In summary, on average, when clustering genes, MAXCCLUS can explain about 37% of the genes (figure 2.4). But when grouping genes into operons and clustering these operons, on average, MAXCCLUS can explain only about 20% of the genes using the strictest approach OP100 (figure 4.11). This percentage increases a little by using the less strict approach OP66, and again increases by using the least strict approach OP70. This increase is because of accepting more operons that will introduce more genes; these genes were not present in the operons accepted using the strictest version OP100.

Comparing clusters results with instances results

When clustering with operons the number of clusters was lower than when clustering with genes. At the same time, the number of genes went down but not as much as the number of clusters. That means when grouping genes into operons and then clustering these operons MAXCCLUS finds bigger clusters that have more genes.

Rules

The number of rules is smaller when clustering operons than when clustering genes. That means when grouping genes into operons and clustering these operons, MAXCCLUS has got rid of some of the very specific rules. These rules had lots of words that probably had not very many instances. These rules were significant when clustering genes, but they are not significant when clustering operons because these rules represent clusters that are too small to be considered significant. Some of these rules represent groups of genes that were mostly operons. Therefore the rules that MAXCCLUS creates when clustering operons are really significant, especially when using the OP100 approach that accepts pure operons, so we get pure rules from this approach. But not all the rules that MAXCCLUS creates when clustering genes are significant because some of them result from groups of genes that belong to operons.

Comparing instances results with rules results

For OP100 and OP66, on average, between 57% and 72% as many genes are being explained but only between 25% and 49% as many rules are explaining those genes (figures 4.6 and 4.12). So there are fewer rules that each explains more genes. This implies that when clustering operons MAXCCLUS creates rules which are bigger and better. MAXCCLUS has retained rules that explain many more genes than when clustering genes. When clustering operons, MAXCCLUS has got rid of the useless rules that explain groups of genes that in reality belong to operons.

Words per rule

For OP70 (figure 4.19), the results showed that the number of words per rule for the significant cluster set is less than that of the cover cluster set, which in turn is slightly less than that of the generalised cover cluster set. This implies that when creating the cover clusters and generalised cover clusters, MAXCCLUS is creating rules that have more words to describe the instances covered by these clusters. These words come from the descriptors of the extra instances described by the rules.

4.5 Conclusion

As described earlier in this chapter, some genes are related to each other. Each group of the related genes is called an operon. The operon's genes are instances that have shared dependency (the operon's genes functionally depend on each other). By clustering genes, MAXCCLUS ignores the dependency between operon's genes during the significance test. This yields clusters that are statistically big enough to be considered as significant clusters.

When grouping genes into operons, then using MAXCCLUS to cluster the operons, MAXCCLUS could not describe as many genes as when we used it to cluster genes. This drop in the number of genes that MAXCCLUS can describe is because MAXCCLUS is now considering the dependency between some of the genes, which represent operons, during its significance test. But the rules it did obtain were larger (describing more genes) and perhaps more informative. Also, with OP66 and OP70, it is able to make predictions about genes that had ambiguous regulation levels.

4.6 Recommendations

MAXCCLUS significance test needs to be modified to consider the dependency of operons' genes, so that a cluster will not be considered significant only because it is statistically big enough, but also because it consists of enough number of independent instances.

Chapter 5

Deleting Descriptors

This chapter describes a clustering experiment designed to study the effects on MAXCCLUS clustering after deleting some of the descriptors of the instances, and comparing its results with the reference clustering experiment where no descriptors are deleted. The deleted descriptors are the non-domain words that we expect to carry little meaning for characterising the clusters. The result of this chapter is to show that deleting the non-domain descriptors can improve MAXCCLUS clustering by allowing MAXCCLUS to create clusters that are no longer characterised by the non-domain words and to generate the clusters faster. Instead of the non-domain words, the domain-relevant words appear in the rules that characterise the clusters. For the reference clustering experiment, these domain-relevant words are frequently excluded from the rules that characterise the clusters because of the presence of the non-domain words in the descriptors of the instances.

5.1 The experiment

To carry out this chapters clustering experiment, we first prepared the new textual data (new descriptor table file), then we ran MAXCCLUS on the microarray experimental data sets using the new textual data.

The normal textual data (the one that has no descriptors deleted from it) that the reference clustering experiment uses is prepared by cleaning the text extracted from the Swiss-Prot database, and deleting the 15 most common *function words* (*stop words*): "*a*", "*an*", "*and*", "*are*", "*at*", "*by*", "*for*", "*from*", "*in*", "*is*", "*of*", "*on*", "*the*", "*to*", and "*with*". These function words were removed because they are not informative in the rules that describe the clusters, and because they are so common, they may mask more informative words to be included in the rules.

In this chapter's clustering experiment we choose to clean the textual data further, for the same reason that we deleted the stop words above. We expect that further cleaning of the textual data will improve the scientific relevance of the rules by including more informative words. We also expect cleaning the textual data further will increase the efficiency of the clustering process; because there are fewer descriptors, the clustering execution time will be reduced and the computer memory needed for the clustering process will also be reduced. However, we expected that the number of clusters would not reduce very much so that MAXCCLUS should not lose important clusters.

The project chose two approaches to deleting descriptors from the original descriptor set of each instance:

- Deleting descriptors that exist in a relatively *small dictionary*. We chose words in the bible for this purpose.
- Deleting descriptors that exist in a relatively *large dictionary*. We chose WordNet for this purpose.

5.1.1 The bible-words dictionary

The bible-words dictionary contains all the words from a "*Basic English*" version of the bible [Bible in Basic English]. We chose the bible-words dictionary because we expect its words to be far from the domain of biology and genetics where we apply our clustering experiments, and because it is relatively small dictionary. We reduced the size of the bible-words dictionary further by excluding all capitalised nouns (the names), for two reasons. First, we expect these names to not occur in the Swiss-Prot data, and second to make the process of deleting descriptors more efficient. The final version of the bible-words dictionary, used in deleting descriptors, contains 3561 words (about 1000 root words).

5.1.2 WordNet

WordNet is a lexical database for the English language from Princeton University [Princeton University, 2003]. We chose WordNet as an alternative to the bible-words dictionary because WordNet is much bigger than the bible-words dictionary (WordNet contains 144,309 words). At the same time, we still expect WordNet not to contain many domain-relevant words from genetics. WordNet contains only open class words (nouns, verbs, adjectives and adverbs). Note that because WordNet does not contain function words (such as the word "*this*"), WordNet is not a strict superset of the bible-words dictionary.

5.2 The algorithm for deleting descriptors

To delete the descriptors that exist in the dictionary, we created a program using the *Perl* computer-programming language.

Figure 5.1 shows the algorithm that attempts to reduce the size of the descriptor set of each instance in the descriptor table file, by going through each descriptor in the instance's descriptor set, and checks if this descriptor exists in dictionary. If the descriptor is in the dictionary, the algorithm deletes the descriptor from the instance's descriptor set. If the resulting descriptor set is not empty, the algorithm saves the instance and the new version of the instance's descriptor set to the new descriptor table file, which MAXCCLUS will use in this chapter's clustering experiment. Checking whether a word exists in the bible-words dictionary is straight forward, but because WordNet is not a simple list of words, the program uses a module to interface with the WordNet database [Rennie, J. 2002].

The output data will be the new descriptor table file, in which each line contains an instance id (gene id) followed by a set of descriptors (words that describe the instance). This set of descriptors will lack some descriptors that were in the instance's original descriptor set. A possible consequence is that some instances may be lost from this table if their descriptor sets contained only words in the dictionary. Practically, none of the instances were lost, neither by using the bible-words dictionary nor by using WordNet.

```
01 for each instance in the original descriptor table
02    for each descriptor in the instance's descriptor set
03       if descriptor exists in dictionary
04          delete descriptor from the instance's descriptor set.
05 if instance's new descriptor set is not empty
06    save instance and its new descriptor set to the new descriptor table file.
```

Figure 5.1: The algorithm for deleting descriptors.

5.3 Experimental results

This section and its figures show the results for running MAXCCLUS on 23 of the microarray experimental data sets (data set: 3, 6, 7, 8, 10, 11, 12, 13, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 35, 36 and 40) first, after deleting descriptors using the *bible-words* dictionary, and second, after deleting descriptors using *WordNet*. We will refer to the two sets of results by ***BW*** and ***WN*** respectively in the sections and figures below. This section also compares these results with each other and with the results produced by the reference clustering experiment (the one before deleting descriptors). Figures 5.2 to 5.17 show the results for the number of clusters, the number of instances, the number of rules, the number of rules' words, the number of words per rule, and the execution time.

The set of *all-descriptors* - the union of the descriptor sets of all instances in the descriptor table file - contains 14429 words before deleting descriptors. After deleting descriptors from each instance's descriptor set, the size of the set of *all-descriptors* reduces also.

There are 394 words from the bible-words dictionary in the set of *all-descriptors* before reducing the size of the set of *all-descriptors*. Therefore the set of *all-descriptors* after deleting from it the descriptors that exist in the bible-words dictionary shrank to approximately 97% of its original size.

On the other hand there are 2566 words from WordNet in the set of *all-descriptors*. Therefore the set of *all-descriptors* after deleting from it the descriptors that exist in WordNet shrank to approximately 82% of its original size.

The clusters

Figures 5.2 to 5.5 show the changes to the number of clusters when some descriptors are deleted from the descriptor sets of the instances. The figures show the changes to the total number of clusters, the number of retained clusters, the number of clusters subsumed by new clusters, and the number of new clusters.

When using BW or WN, on average, there are fewer clusters when clustering after deleting descriptors than when clustering before deleting descriptors (figure 5.2). However, the change in the BW experiment is small: there is almost no change to the number of the generalised cover clusters of BW.

For any of the cluster sets, significant, cover, or generalised cover cluster set, on average, there are more clusters when using BW than when using WN.

When using BW, on average, the significant cluster set has a slightly greater change in the number of clusters, than the cover cluster set, which in turn, has greater change than the generalised cover cluster set. In contrast, when using WN, on average, the significant cluster set has a significantly greater change than the cover and the generalised cover cluster sets.

When clustering after deleting descriptors, on average, the clusters *retained* from the clusters found when clustering before deleting descriptors, are many more when using BW than when using WN. Between 47% and 59% of clusters are retained when using BW, but only between 2% and 6% of clusters are retained when using WN. For both BW and WN the generalised cover cluster set retained more clusters than the cover cluster set (figures 5.3).

When clustering after deleting descriptors, on average, the *new* clusters found are fewer when using BW than when using WN. Between only 39% and 51% of clusters are new when using BW, but between 91% and 95% of clusters are new when using WN (figure 5.5).

On average, more clusters found from clustering before deleting descriptors, are *subsumed* by new clusters found from clustering after deleting descriptors using BW than that using WN. Between 1% and 19% of clusters are subsumed when using BW, but only between 0% and 9% of clusters are subsumed when using WN. Significant cluster set has *noticeable* number of subsumed clusters especially when using BW. The significant cluster set has more clusters subsumed than the cover cluster set, which in turn has more clusters subsumed than the generalised cover cluster set, which in turn has almost no clusters subsumed using BW or has no clusters subsumed using WN (figure 5.4).

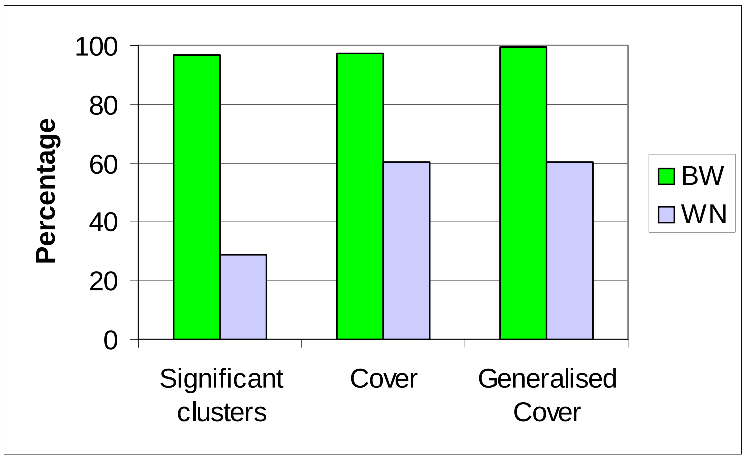


Figure 5.2: Average change to the total number of clusters

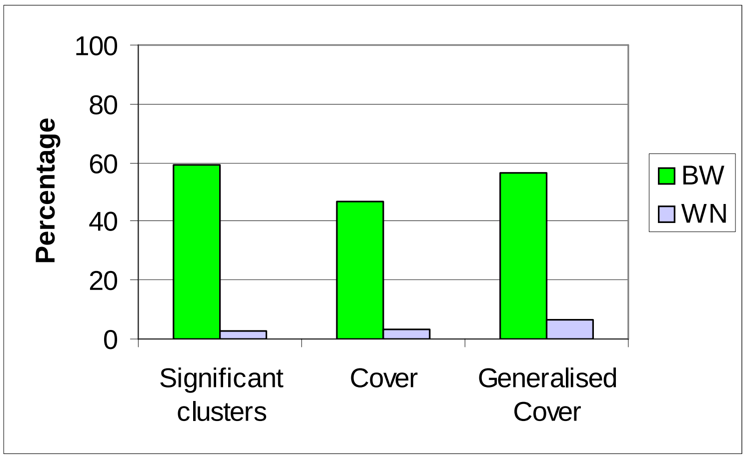


Figure 5.3: Average number of retained clusters

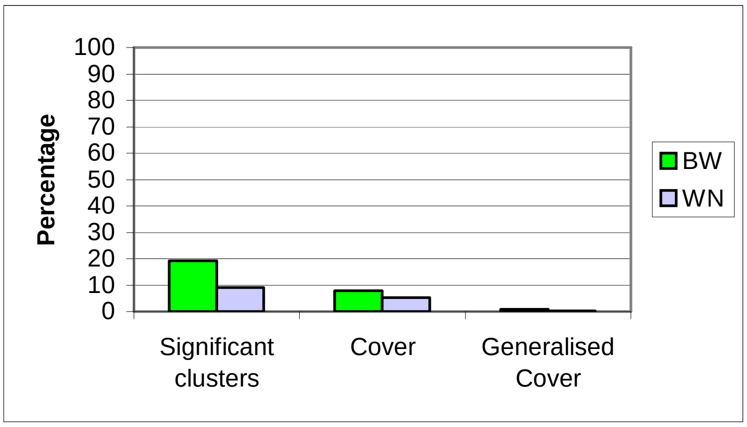


Figure 5.4: Average number of clusters subsumed by new clusters

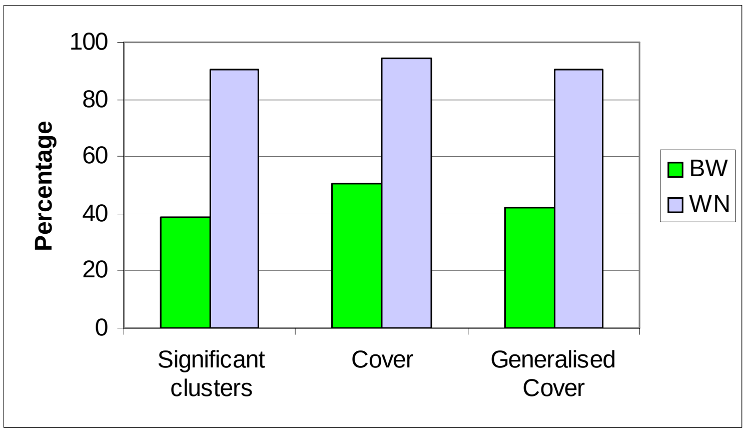


Figure 5.5: Average number of new clusters

Significant clusters minimum size

Figure 5.6 shows the changes to the significant clusters' minimum size for each category when some descriptors are deleted from the descriptor sets of the instances.

When clustering after deleting descriptors using BW or WN, on average, the significant clusters' minimum size for each category is less than that when clustering before deleting descriptors. On average, the reduction in the significant clusters' minimum size, when clustering using BW, is less than that when using WN; with almost no change in the minimum size for *Down* category when using BW. When using any of BW or WN, the reduction in the significant clusters' minimum size for the *Up* category is more than that for the *Down* category.

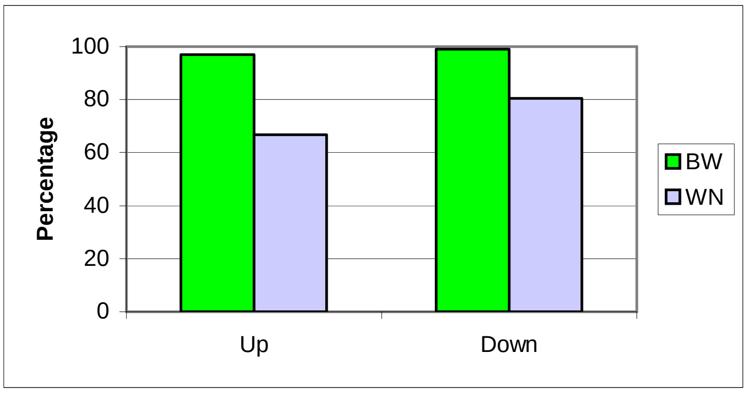


Figure 5.6: Average change to the significant clusters' minimum size for each category

The instances

Figures 5.7 to 5.9 show the changes to the number of instances when some descriptors are deleted from the descriptor sets of the instances. The figures show the changes to the total number of instances, the number of retained instances, and the number of new instances.

On average, when using BW, there is almost no change to the number of instances when clustering after deleting descriptors than when clustering before deleting descriptors. In contrast, when using WN, there are fewer instances when clustering after deleting descriptors than when clustering before deleting descriptors, with between only 55% and 70% as many instances found when clustering after deleting descriptors than when clustering before deleting descriptors (figure 5.7). When using WN, on average, the generalised cover cluster set has the highest decrease in the number of instances.

When clustering after deleting descriptors, on average, the instances *retained* from the clusters found when clustering before deleting descriptors, are more when using BW than when using WN, about 94% of instances are retained when using BW, but only between 45% and 60% of instances are retained when using WN. For WN the generalised cover cluster set retained fewer instances than the other cluster sets (figures 5.8).

When clustering after deleting descriptors, on average, very few instances found are *new*. The new instances found are fewer when using BW than when using WN, with about only 5% of instances are new when using BW, but about 12% of instances are new when using WN (figure 5.9).

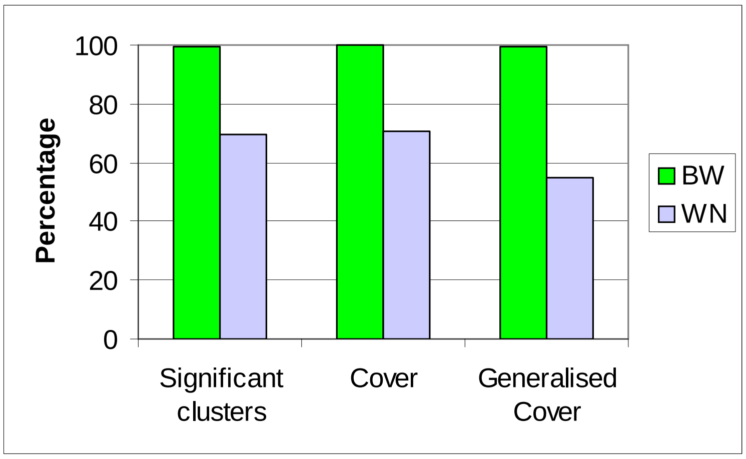


Figure 5.7: Average change to the total number of instances

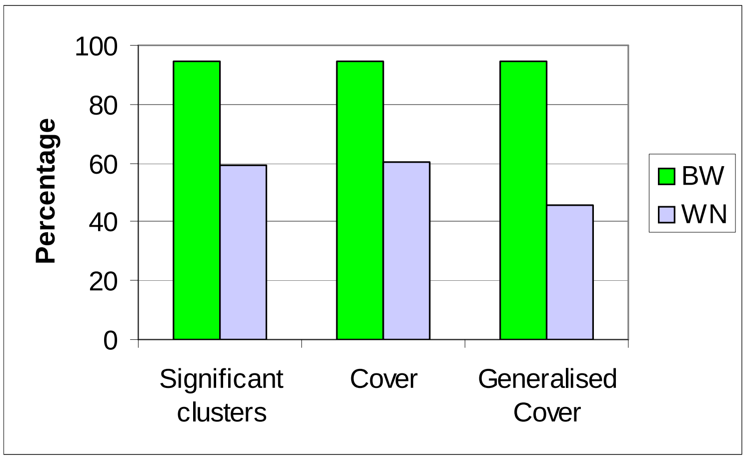


Figure 5.8: Average number of retained instances

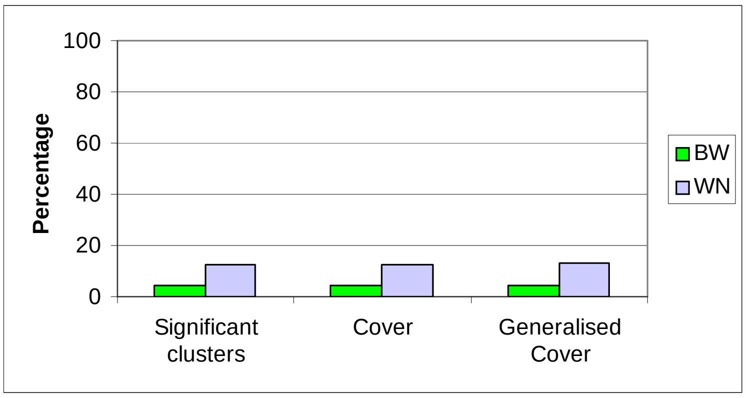


Figure 5.9: Average number of new instances

The rules

Figures 5.10 to 5.12 show the changes to the number of rules when some descriptors are deleted from the descriptor sets of the instances. The figures show the changes to the total number of rules, the number of retained rules, and the number of new rules.

When using BW or WN, on average, there are fewer rules when clustering after deleting descriptors than when clustering before deleting descriptors (figure 5.10). On average, there are more rules when using BW than when using WN, between 67% and 90% as many rules when using BW, but only between 17% and 53% as many rules when using WN.

When using BW or WN, on average, the significant cluster set has greater decrease, in the number of rules, than the cover cluster set, which in turn, has greater decrease than the generalised cover cluster set.

When clustering after deleting descriptors, on average, the rules *retained* from the clusters found when clustering before deleting descriptors, are many more when using BW than when using WN. Between 38% and 46% of rules are retained when using BW, but only about 4% of rules are retained when using WN. For BW, the significant cluster set retained more rules than the generalised cover cluster set, which in turn, retained more rules than the cover cluster set (figures 5.11).

When clustering after deleting descriptors, on average, the *new* rules found are fewer when using BW than when using WN. Between only 30% and 47% of rules are new when using BW, but between 81% and 91% of rules are new when using WN (figure 5.12).

Almost no rules found from clustering before deleting descriptors, are *subsumed* by new rules found from clustering after deleting descriptors using BW or WN.

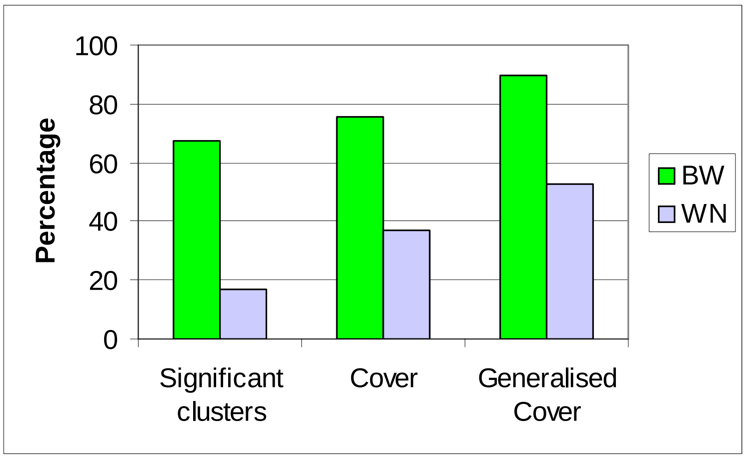


Figure 5.10: Average change to the total number of rules

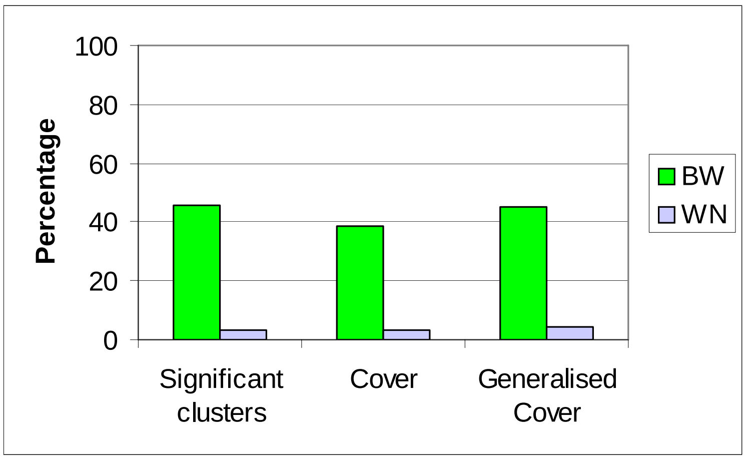


Figure 5.11: Average number of retained rules

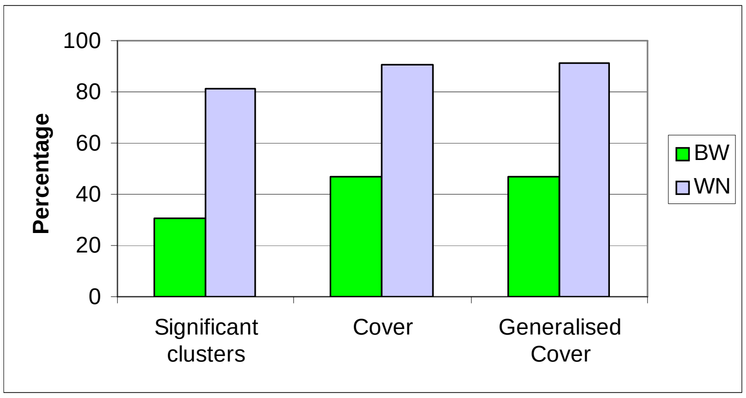


Figure 5.12: Average number of new rules

The rules' words

Figures 5.13 to 5.15 show the changes to the number of rules' words when some descriptors are deleted from the descriptor sets of the instances. The figures show the changes to the total number of words, the number of retained words of the rules, and the number of new words.

When using BW or WN, on average, there are fewer rules' words after deleting descriptors than before deleting descriptors (figure 5.13). On average, there are more rules' words when using BW than when using WN; between 89% and 93% as many rules' words when using BW, but only between 46% and 50% as many rules' words when using WN.

When using BW or WN, on average, the significant cluster set has greater decrease, in the number of rules' words, than the cover cluster set, which in turn, has greater decrease than the generalised cover cluster set.

When clustering after deleting descriptors, on average, the rules' words *retained* from the clusters found when clustering before deleting descriptors, are many more when using BW than when using WN. Between 74% and 77% of rules' words are retained when using BW, but only between 21% and 25% of rules' words are retained when using WN. For BW, the significant cluster set retained more rules' words than the generalised cover cluster set, which in turn, retained more rules' words than the cover cluster set. In contrast, for WN, the significant cluster set retained more rules' words than the cover cluster set, which in turn, retained more rules' words than the generalised cover cluster set (figures 5.14).

When clustering after deleting descriptors, on average, the rules' words that are *new* found are fewer when using BW than when using WN. Between only 13% and 20% of the rules' words are new when using BW, but between 44% and 56% of the rules' words are new when using WN (figure 5.15). The significant cluster set has fewer rules' words that are new than the cover cluster set, which in turn, has fewer rules' words that are new than the generalised cover cluster set.

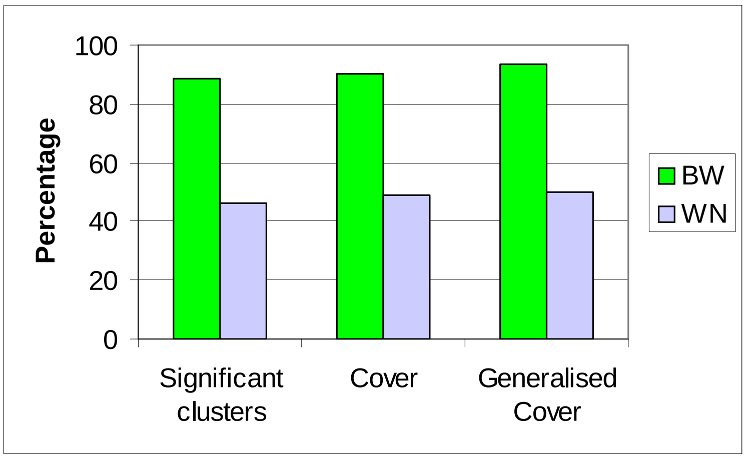


Figure 5.13: Average change to the total number of rules' words

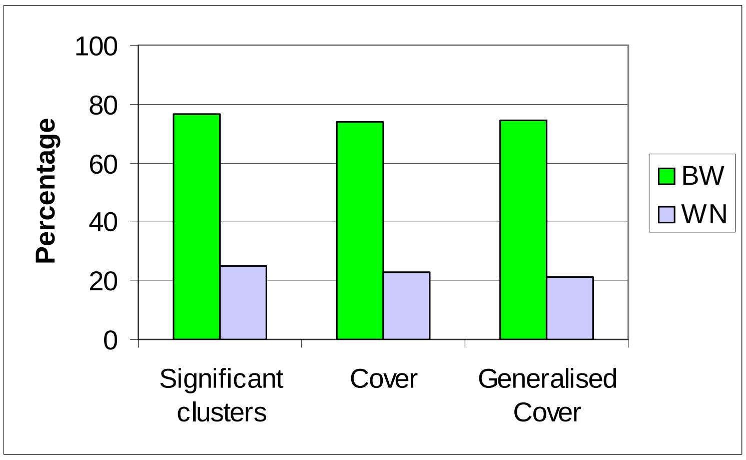


Figure 5.14: Average number of retained rules' words

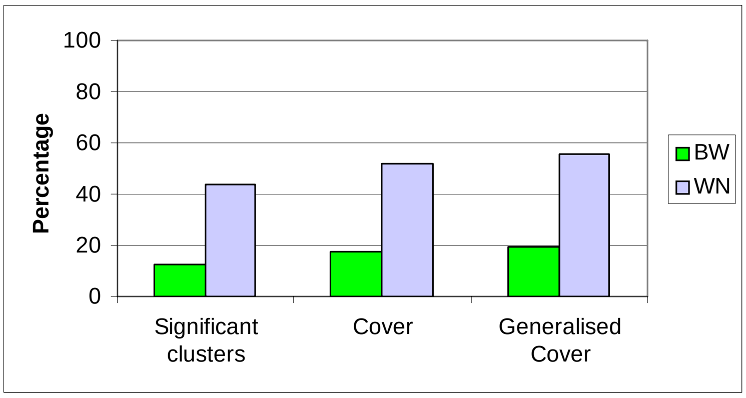


Figure 5.15: Average number of new rules' words

The words per rule

Figure 5.16 shows the changes to the number of words per rule when some descriptors are deleted from the descriptor sets of the instances.

When using BW or WN, on average, there are fewer words per rule when clustering after deleting descriptors than when clustering before deleting descriptors. On average, there are more words per rule when using BW than when using WN. Between 96% and 97% as many words per rule when using BW, but only between 63% and 65% as many words per rule when using WN, compared with words per rule when clustering before deleting descriptors.

On average, for each of BW and WN, the difference in change in the number of words per rule is very little between cluster sets.

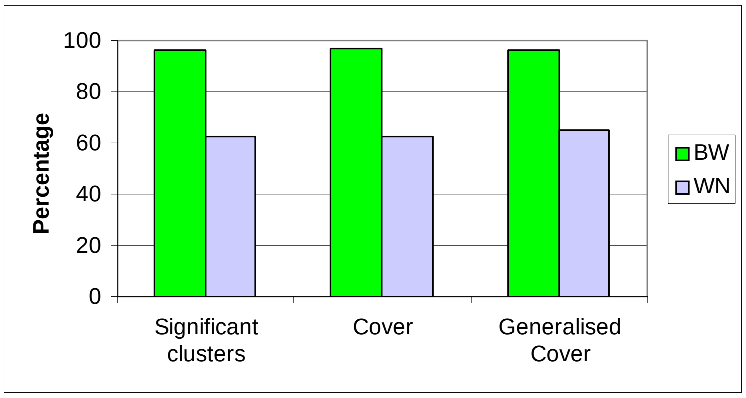


Figure 5.16: Average change to the number of words per rule

The execution time

Figure 5.17 shows the changes to the clustering execution time when some descriptors are deleted from the descriptor sets of the instances.

When clustering after deleting descriptors using BW or WN, on average, the execution time is less than that when clustering before deleting descriptors. On average, the execution time, when clustering using BW, is more than that when using WN; about 74% as much time when using BW, but only about 11% as much time when using WN, compared with the execution time when clustering before deleting descriptors.

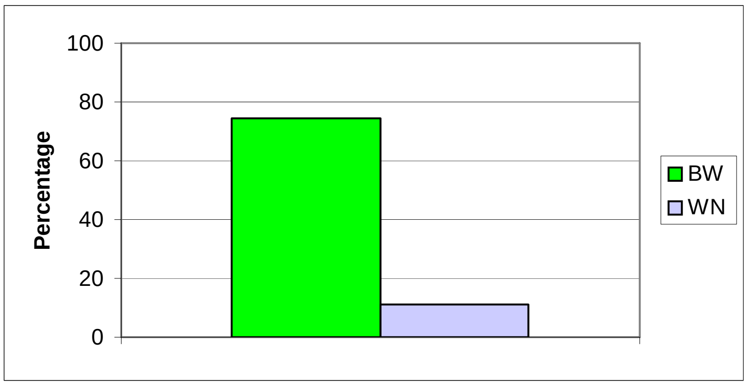


Figure 5.17: Average change to the clustering execution time

5.4 Analysis and discussion

The following sections analyse and discuss the results shown in the results section.

Clusters and instances

Figure 5.2 shows that, on average, when clustering after deleting descriptors using BW, there is almost no change to the number of clusters found by MAXCCLUS, compared to that when clustering before deleting descriptors. The small loss in the number of significant clusters (about 3%) does not have an important effect on the number of instances that MAXCCLUS can describe (figure 5.7). When clustering after deleting descriptors using BW, MAXCCLUS is able to describe almost the same number of instances as when clustering before deleting descriptors. However, MAXCCLUS needs less execution time for clustering (figure 5.17): on average, MAXCCLUS needs only about 75% of the time that it needs for clustering before deleting descriptors.

The reduction in execution time MAXCCLUS needs to cluster instances is because of the reduction in the sizes of the descriptor sets of the instances using BW. MAXCCLUS performs clustering by creating clusters of instances depending on the descriptor sets of these instances. Therefore, when there are fewer descriptors there will be fewer combinations of descriptors to create the clusters of these combinations. Although figure 5.2 showed that the significant cluster set did not change much, the total number of clusters created (the set of *all-clusters*) was much smaller. This reduces the time required for the permutation test. Note that the clusters that were lost were not significant clusters.

In contrast, when clustering after deleting descriptors using WN, the observations mentioned in the previous sections become more noticeable. There is a large reduction in the number of clusters, especially the significant cluster set (figure 5.2). This reduction in the number of clusters is accompanied by a large reduction in the number of instances that MAXCCLUS can describe (figure 5.7). Although the execution time reduces dramatically to an average of 11%, using WN is not preferable because of the loss in the number of instances that MAXCCLUS is able to describe.

Rules and rules' words

Figure 5.10 shows that, on average, when clustering after deleting descriptors using BW or WN, MAXCCLUS finds fewer rules than when clustering before deleting descriptors. On the other hand, figure 5.13 shows that the number of rules' words reduces too, but not as much as the number of rules. The reduction of the number of rules and the number of rules' words is due to the reduction of the sizes of descriptor sets of the instances because of deleting descriptors from these sets. The smaller change for BW than for WN is because of the smaller reduction of the sizes of the descriptor sets of the instances when using BW than when using WN.

Figure 5.4 shows that there are clusters of instances from clustering before deleting descriptors that are *subsumed* by new clusters from clustering after deleting descriptors (a considerable number of significant clusters when using BW). The subsumed clusters found by MAXCCLUS represent sets of instances that MAXCCLUS clustered using non-domain (uninformative) descriptors. When clustering after deleting descriptors, the instances that belong to the subsumed clusters, lack the non-domain words; therefore MAXCCLUS clusters these instances using their descriptors that are now more domain-relevant. Therefore MAXCCLUS finds clusters that describe more instances including the instances that belong to the subsumed clusters. These clusters that MAXCCLUS finds after deleting descriptors are bigger and scientifically more informative.

By looking at the actual rules generated, we observed that words like: “*as*”, “*family*”, “*it*”, “*property*”, “*step*”, “*that*”, “*this*” are lost from the BW rules that characterise the clusters. Instead of these words, new words that are more domain-relevant appeared in the rules: “*coli*”, “*two-dimensional*”, “*gel*”, “*transcription*”, “*transmembrane*”, “*catalyze*”.

For example, figure 5.18 shows a cluster of instances before deleting descriptors. Using the same data set, after deleting descriptors using BW (figure 5.19), the same cluster lost the word "*family*" (which is in the bible-words dictionary and is not informative) from its *Sufficient* set of words. The loss of the word “*family*” made the word "*belong*" (which is not in the bible-words dictionary) become a word of the *Necessary* set of words instead of the *Sufficient* set of words, causing the *Sufficient* set of words to became empty. The effect of this was to reduce the number of rules from 2 to 1 in this example, which is a reduction of 50%. It also made the remaining rule relatively more informative by not including the word "*family*" in it. This is because the word "*family*" is closely connected to the word "*belong*": they appear together in the textual database implying a relationship between them. Some of the examples of the relation between the words "*family*" and "*belong*" from the *Swiss-Prot* database textual data are:
“BELONGS TO THE TRANSALDOLASE FAMILY”
“BELONGS TO THE GHMP KINASE FAMILY”
“BELONGS TO THE SODIUM:ALANINE SYMPORTER FAMILY”

Note, the word “BELONGS” was stemmed to “*belong*” in the pre-processing.

Cluster (Down, 100% accuracy, 8 instances)

Characterisation:
Necessary: protein , sequence , subunit
Sufficient: belong , family
Supplementary: coli , escherichia

Rule: protein , sequence , belong , subunit
Rule: protein , sequence , family , subunit

Genes: b0882 , b0911 , b1260 , b1923 , b2614 , b3295 , b3699 , b3829

Figure 5.18: A cluster before deleting descriptors

Cluster (Down, 100% accuracy, 8 instances)

Characterisation:
Necessary: protein , sequence , belong , subunit
Supplementary: coli , escherichia

Rule: protein , sequence , belong , subunit

Genes: b0882 , b0911 , b1260 , b1923 , b2614 , b3295 , b3699 , b3829

Figure 5.19: The same cluster of figure 5.18, but after deleting descriptors using BW

In contrast, after deleting descriptors using WN, the same cluster of instances is lost; MAXCCLUS is not able to construct this cluster at all because of the severe loss of descriptors in the instances' descriptor sets.

Figures 5.3 to 5.5 show that clustering with WN generates a very different set of clusters, much more so than for BW. This is because the major change in the descriptor sets of the instances after deleting descriptors using WN, including many descriptors that are relevant to the domain. Losing lots of domain-relevant descriptors when using WN cause MAXCCLUS to find clusters that are missing important scientific knowledge (information). Therefore it seams that using WN is a bad thing to do, in spite the fact that the remaining words are the uncommon domain-relevant descriptors, which may lead one to expect a good result.

Some of the words that are retained when using BW are lost when using WN. Examples of these words are: "*escherichia*", "*hypothetical*", "*salmonella*", "*nucleotide*", "*biosynthesis*", "*expression*", "*cytoplasmic*". These are common domain-relevant words.

Instead of the lost words, when clustering using WN, new words appear in the rules. Examples of these new words are: "*ec@2*", "*kda*", "*acetyl-coa*", "*transmembrane*", "*atp-binding*", "*dna-binding*", "*phosphoenolpyruvate*", "*allosteric*", "*3-phosphate*", "*homotetramer*", "*synthetase*", "*l-glutamate*". These are less common domain-relevant words than the words in the previous paragraph.

Figure 5.20 shows an example from another data set, of a cluster after deleting descriptors using WN. The word "*phosphoenolpyruvate*", which is the only word in the *Necessary* set of words, does not exist in WordNet, also it does not exist in any word set of any cluster when clustering before deleting descriptors or when clustering after deleting descriptors using BW. This cluster contains *only* one rule that has *only* one word, which is an uncommon domain-relevant word.

```
Cluster
category: Dn
accuracy: 100.0
instanceCount: 6

Characterisation:
Necessary:       phosphoenolpyruvate
Supplementary: coli

Rule: phosphoenolpyruvate

Genes: b0908 , b1676 , b2169 , b2416 , b2779 , b3956
```

Figure 5.20: A cluster after deleting descriptors using WN

5.5 Conclusion

Deleting the non-domain descriptors in the Bible-words dictionary improves the rules that characterise the clusters created by MAXCCLUS, making the rules more scientifically informative. MAXCCLUS is still able to describe almost the same number of instances compared with what it can cover without deleting descriptors. Also, deleting the non-domain descriptors reduces the clustering time.

However, deleting more descriptors, including common domain-relevant descriptors, using the WordNet dictionary gives bad results, because MAXCCLUS is not able to describe as many instances or create as many clusters as before deleting descriptors.

5.6 Recommendations

These experiments demonstrate that it is advantageous to delete descriptors that are not relevant to the applied domain, because they do not carry any significant scientific information related to the domain. On the other hand, it is important not to delete too many descriptors, especially domain-relevant ones, because it removes essential information needed for the clustering process.

Chapter 6

Adding Descriptors

This chapter describes a clustering experiment designed to study the effects on MAXCCLUS clustering after adding more descriptors to the descriptor sets of the instances, and comparing its results with the reference clustering experiment where no descriptors are added. The added descriptors are the synonyms and hypernyms of the already existing descriptors. We expected this to help MAXCCLUS to form clusters of instances that have semantic relationships but were not previously clustered because they did not share any literal descriptors. Unexpectedly, the result of this chapter is to show that adding descriptors does not improve MAXCCLUS clustering; MAXCCLUS is not able to describe more instances and it generates the clusters *much more* slowly.

6.1 The purpose of the experiment

Some genes can have different descriptor sets and they may not share any descriptor, although some of these descriptors have a very similar meaning. For example, suppose that there are two instances that do not share any descriptor. If the first instance has the word "*expand*" as one of its descriptors, and the second instance has the word "*enlarge*" as one of its descriptors, then these instances may belong to the same cluster because the words "*expand*" and "*enlarge*" have the same meaning. In this case, we expect that adding synonyms will enable MAXCCLUS to describe more instances by being able to see the similarity in their descriptor sets.

6.2 Adding descriptors

We chose three approaches to add more descriptors:

- adding *synonyms* (x is a synonym of y, if x and y have the same meaning). An example: the word “*enlarge*” is a synonym of the word “*expand*”.
- adding *hypernyms* (x is a hypernym of y, if y is a kind of x). An example: the word “*activity*” is a hypernym of the word “*play*”.
- and adding both synonyms and hypernyms

To ensure that any descriptors we add are relevant to the textual data set, we only add a word that is a synonym or hypernym of a descriptor if that word already exists in the set of *all descriptors* (which is the union of the descriptor sets of all instances). This project uses WordNet to find synonyms and hypernyms of a descriptor.

6.2.1 WordNet

WordNet is a lexical database for the English language from Princeton University [Princeton University, 2003]. WordNet contains 144,309 of the open class words (nouns, verbs, adjectives and adverbs) in English. We created a program using the *Perl* computer-programming language, which uses a special module [Rennie, J. 2002] to access the WordNet database to retrieve all synonyms and hypernyms of descriptors.

6.2.2 The algorithm for adding descriptors

Algorithm 6.1 (figure 6.1) attempts to increase the size of the descriptor set of each instance in the descriptor table file (the input to this algorithm) by going through each descriptor in an instance's descriptor set, and attempting to retrieve the relevant *senses* of this descriptor from WordNet (if any senses exist). In WordNet, the *senses* of a word include the synonyms and hypernyms of the word. For each synonym or hypernym retrieved, it checks whether the word exists in the set of *all-descriptors*, and then adds the word to this instance's descriptor set. The set of *all-descriptors* is the union of the descriptor sets of all instances. The algorithm writes each instance's new descriptor set to the new descriptor table file, which MAXCCLUS will use in this chapter's clustering experiment.

```
01 for each instance in the original descriptor table
02    for each descriptor in the instance's descriptor set
03        retrieve the relevant senses of this descriptor
04        for each sense
05            if sense exists in the set of all-descriptors set
06                add sense to the instance's descriptor set.
07 save instance and its new descriptor set to the new descriptor table file.
```

Figure 6.1: Algorithm 6.1: Adding senses to instances' descriptor sets.

Figure 6.2 shows the increase in the average size of the descriptor sets for the three cases. Adding hypernyms gives in lowest increase value, on the other hand, adding synonyms and hypernyms together gives in the highest increase value.

Added senses	**Increase**
Synonyms	69%
Hypernyms	50%
Synonyms and Hypernyms	117%

Figure 6.2: The increase in the average size of the descriptor sets for the three cases.

6.3 Experimental results

This section presents the results of running MAXCCLUS on 23 of the microarray experimental data sets (data set: 3, 6, 7, 8, 10, 11, 12, 13, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 35, 36, and 40) first after adding synonyms, second, after adding hypernyms, and third after adding both synonyms and hypernyms. We will refer to these three sets of results by ***SY***, ***HY***, and ***SH*** respectively in the rest of the chapter. This section also compares these results with each other and with the results produced by the reference clustering experiment (the one before adding descriptors). Figures 6.3 to 6.18 show the results for the number of clusters, the number of instances, the number of rules, the number of words in the rules, the number of words per rule, and the execution time.

The clusters

Figures 6.3 to 6.6 show the changes to the number of clusters when more descriptors are added to the descriptor sets of the instances. The figures show the changes to the total number of clusters, the number of retained clusters, the number of clusters subsumed by new clusters, and the number of new clusters.

When using SY, HY or SH, on average, there are fewer clusters in the significant cluster set when clustering after adding descriptors than when clustering before adding descriptors, but there are more clusters in the cover and the generalised cover cluster sets (figure 6.3).

For the significant cluster set and the generalised cover cluster set, on average, there are more clusters when using HY than when using SH, which in turn, there are more clusters than when using SY. For the cover cluster set, there are more clusters when using SH than when using HY, which in turn, there more clusters than when using SY.

When clustering after adding descriptors, on average, the clusters *retained* from the clusters found when clustering before adding descriptors, are more when using SY than when using HY, which in turn has more retained clusters than when using SH. Between 26% and 36% of clusters are retained when using SY, between 14% and 21% of clusters are retained when using HY, but only between 8% and 14% of clusters are retained when using SH. For SY, HY and SH, the significant cluster set retained more clusters than the cover cluster set or the generalised cover cluster set (figures 6.4).

When clustering after adding descriptors, on average, the *new* clusters found are fewer when using SY than when using HY, which in turn has fewer new clusters than when using SH. Only between 59% and 74% of clusters are new when using SY; between 75% and 85% of clusters are new when using HY, and between 84% and 91% of clusters are new when using SH (figure 6.6).

For the significant cluster set, on average, more clusters found from clustering before adding descriptors, are *subsumed* by new clusters found from clustering after adding descriptors using HY than that using SH, which in turn has more clusters subsumed than when using SY. For the cover cluster set, on average, more clusters are subsumed using SY than that using SH, which in turn has more clusters subsumed than when using HY. For the generalised cover cluster set, on average, more clusters are subsumed using HY than that using SY, which in turn has more clusters subsumed than when using SH. The significant cluster set has more clusters subsumed than the generalised cover cluster set, which in turn has more clusters subsumed than the cover cluster set (figure 6.5).

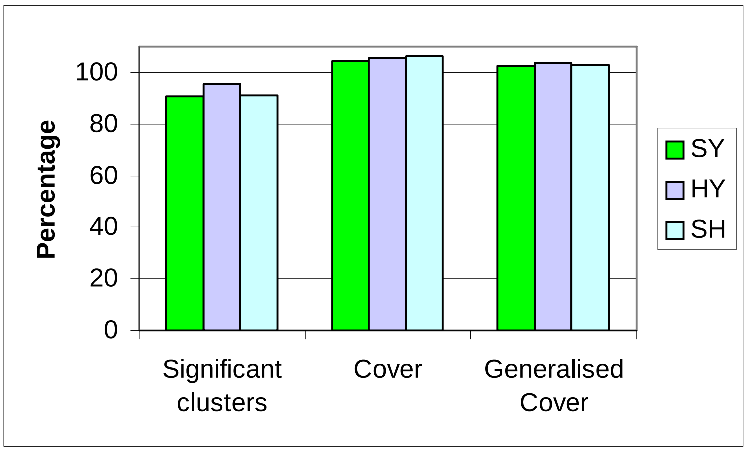


Figure 6.3: Average change to the total number of clusters

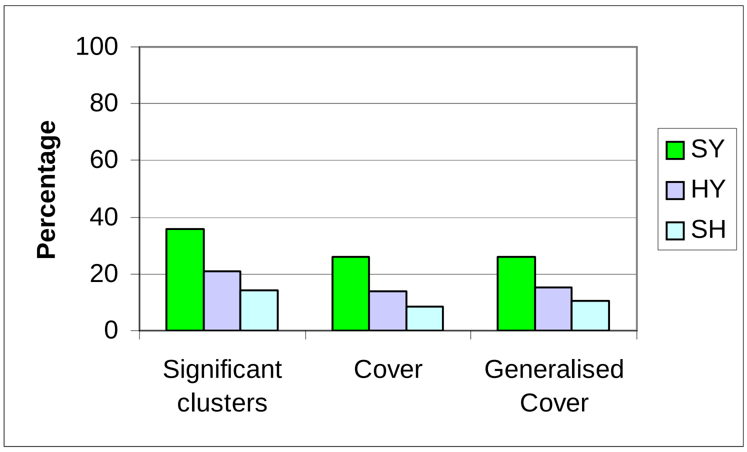


Figure 6.4: Average number of retained clusters

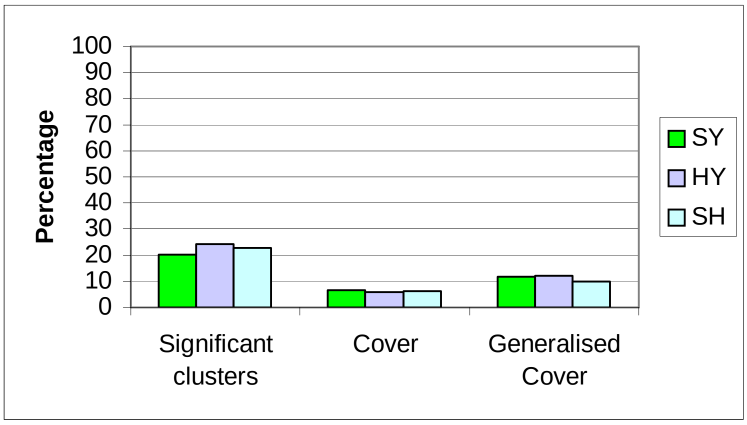


Figure 6.5: Average number of clusters subsumed by new clusters

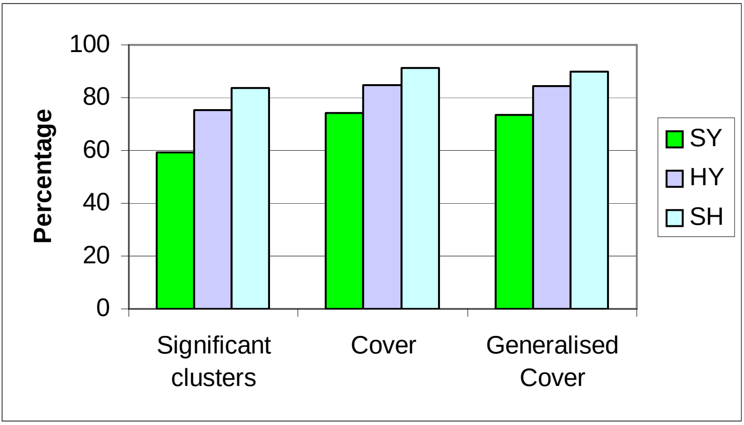


Figure 6.6: Average number of new clusters

Significant clusters minimum size

Figure 6.7 shows the changes to the significant clusters' minimum size for each category when more descriptors are added to the descriptor sets of the instances.

When clustering after adding descriptors using SY, HY, or SH, on average, the significant clusters' minimum size for each category is more than that when clustering before adding descriptors. On average, the increase in the significant clusters' minimum size, when using SY, is less than that when using HY, which in turn is less than that when using SH. When using any of SY, HY, or SH, the increase in the significant clusters' minimum size for the *Up* category is more than that for the *Down* category.

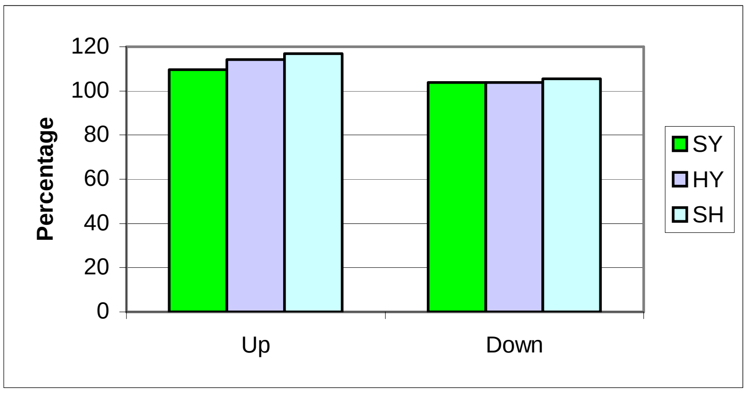


Figure 6.7: Average change to the significant clusters' minimum size for each category

The instances

Figures 6.8 to 6.10 show the changes to the number of instances when more descriptors are added to the descriptor sets of the instances. The figures show the changes to the total number of instances, the number of retained instances, and the number of new instances.

In each figure, the results of the significant and the cover cluster sets are the same because both sets should cover the same instances.

When using SY, HY or SH, on average, there are fewer instances in the significant, cover, and generalised cover cluster sets after adding descriptors than before adding descriptors, with almost no change in the number of instances in the significant cluster set using SY (figure 6.8).

For any of the cluster sets, significant, cover, or generalised cover cluster set, when using SY, on average, there are more instances than when using HY or SH.

When clustering after adding descriptors, on average, the instances *retained* from the clusters found when clustering before adding descriptors, are more when using SY than when using HY, which in turn has more retained instances than when using SH. About 89% of clusters are retained when using SY, about 85% of instances are retained when using HY, but only about 83% instances are retained when using SH. For SY, HY and SH, the generalised cover cluster set retained fewer instances than the significant cluster set or the cover cluster set (figures 6.9).

When clustering after adding descriptors, on average, the *new* instances found are fewer when using SY than when using HY, which in turn has fewer new instances than when using SH. Only about 8% of instances are new when using SY; about 9% of instances are new when using HY, and about 11% of instances are new when using SH (figure 6.10).

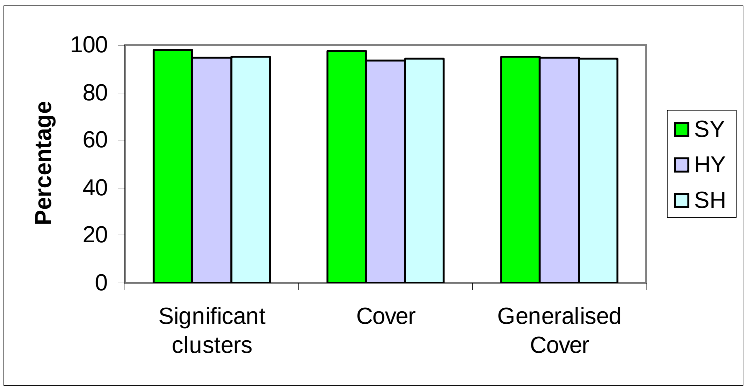


Figure 6.8: Average change to the total number of instances

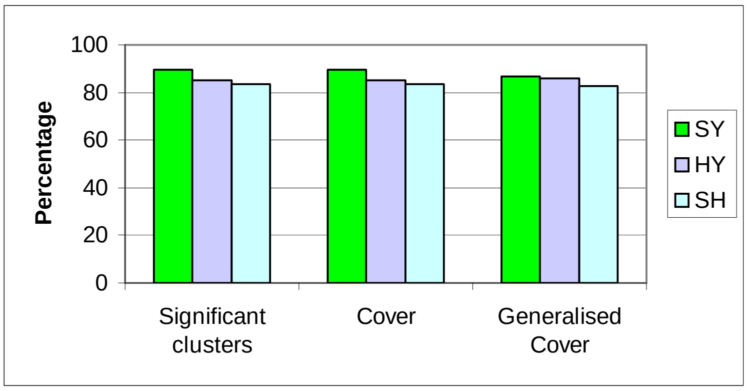


Figure 6.9: Average number of retained instances

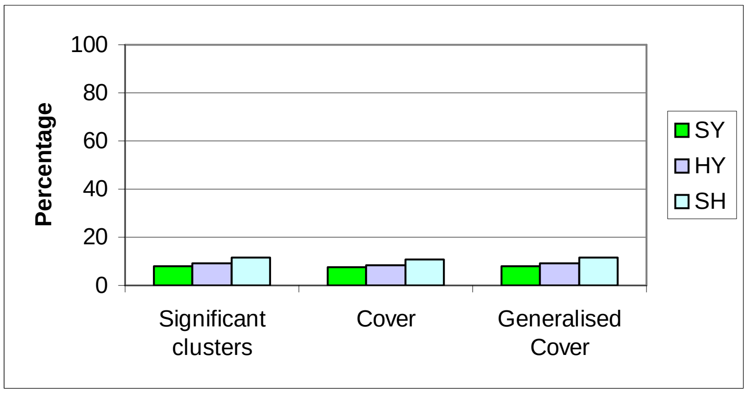


Figure 6.10: Average number of new instances

The rules:

Figures 6.11 to 6.13 show the changes to the number of rules when more descriptors are added to the descriptor sets of the instances. The figures show the changes to the total number of rules, the number of retained rules, and the number of new rules.

When using SY, HY or SH, on average, there are many more rules when clustering after adding descriptors than when clustering before adding descriptors. On average, there are many more rules when using SH than when using SY, which in turn there are more rules than when using HY (figure 6.11).

The significant cluster set has a greater increase in the number of rules than the cover cluster set, which in turn has a greater increase than the generalised cover cluster set.

When clustering after adding descriptors, on average, the number of rules *retained* from the rules found when clustering before adding descriptors, is greater when using SY than when using HY, which in turn has more retained rules than when using SH. Between 26% and 43% of rules are retained when using SY, between 14% and 27% of rules are retained when using HY, but only between 7% and 17% of rules are retained when using SH. For SY, HY and SH, the significant cluster set retained more rules than the cover cluster set, which returned more rules than the generalised cover cluster set (figures 6.12).

When clustering after adding descriptors, on average, the *new* rules found are more when using SH than when using SY or HY. Almost all the rules are new when using SH, especially for the significant and the cover cluster sets (figure 6.13).

On average, almost no rules found from clustering before adding descriptors, are *subsumed* by new rules found from clustering after adding descriptors using any of SY, HY or SH.

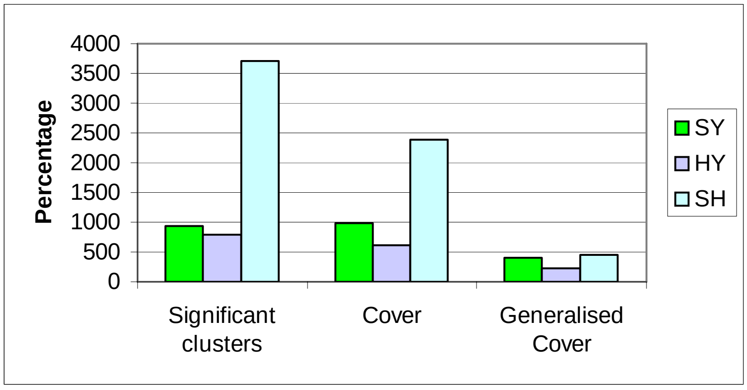


Figure 6.11: Average change to the total number of rules

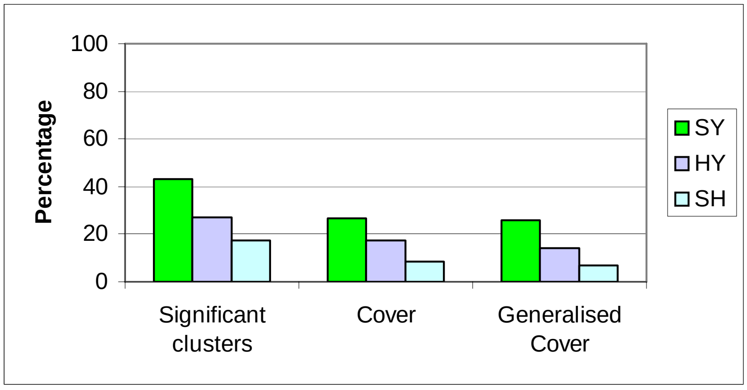


Figure 6.12: Average number of retained rules

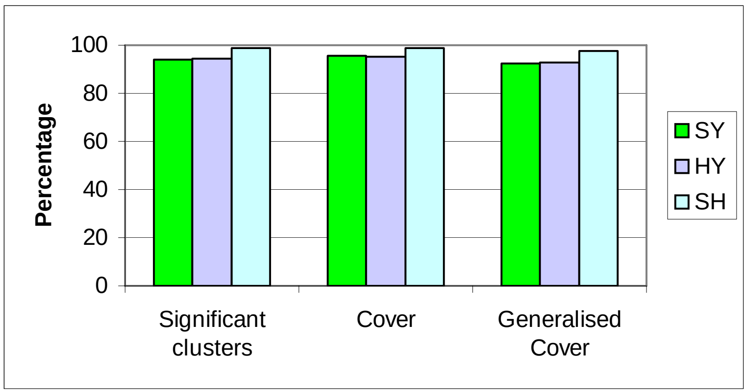


Figure 6.13: Average number of new rules

The rules' words

Figures 6.14 to 6.16 show the changes to the number of rules' words when more descriptors are added to the descriptor sets of the instances. The figures show the changes to the total number, the number of retained words of the rules, and the number of new words.

On average, there are more rules' words after adding descriptors than before adding descriptors. When using SH there are more rules' words than when using SY, which in turn there are more rules words than when using HY. The cover cluster set has greater increase in the number of rules' words than the significant cluster set, which in turn, has greater increase than the generalised cover cluster set (figure 6.14).

When clustering after adding descriptors, on average, the number of rules' words *retained* from the clusters found when clustering before adding descriptors, is greater when using SY than when using HY, which in turn has more retained rules' words than when using SH. Between 67% and 83% of rules' words are retained when using SY, between 57% and 73% of rules' words are retained when using HY, but only between 49% and 69% of rules' words are retained when using SH. For SY, HY and SH, the significant cluster set retained more rules' words than the cover cluster set, which in turn retained more rules' words than the generalised cover cluster set (figures 6.15).

When clustering after adding descriptors, on average, the *new* rules' words found are fewer when using HY than when using SY, which in turn has fewer new rules' words than when using SH. Only between 51% and 57% of rules' words are new when using HY, between 52% and 57% of rules' words are new when using SY, and between 66% and 71% of rules' words are new when using SH. The significant cluster set has fewer new rules' words than the generalised cover set, which in turn has fewer new rules' words than the cover cluster set (figure 6.16).

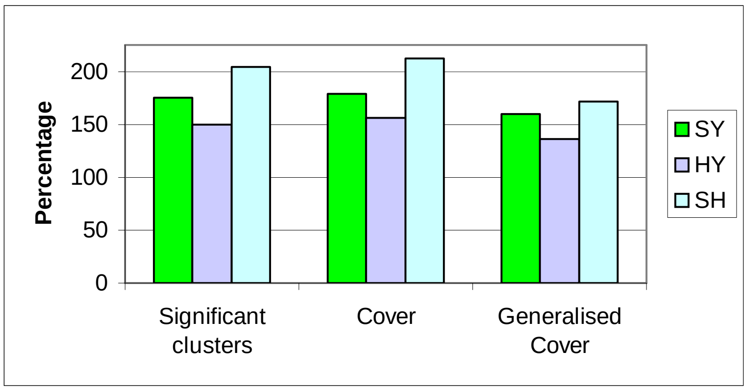


Figure 6.14: Average change to the total number of rules' words

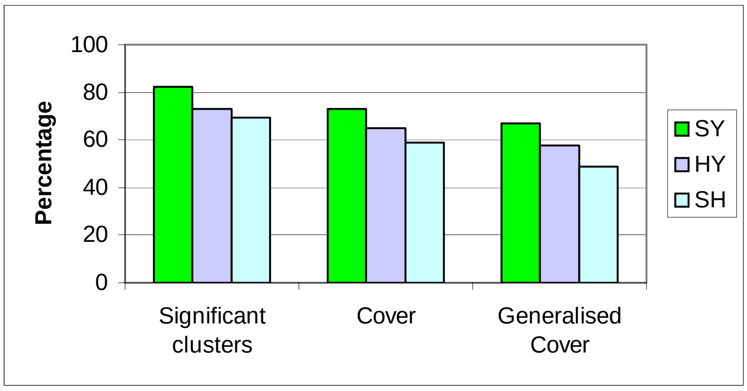


Figure 6.15: Average number of retained rules' words

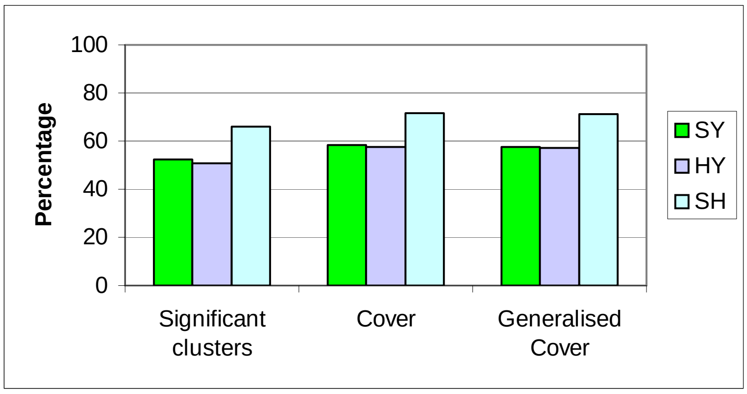


Figure 6.16: Average number of new rules' words

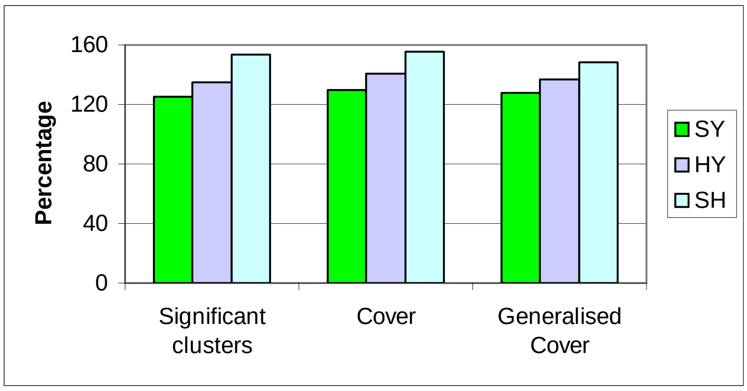
The words per rule

Figure 6.17 shows the changes to the total number of words per rule when more descriptors are added to the descriptor sets of the instances.

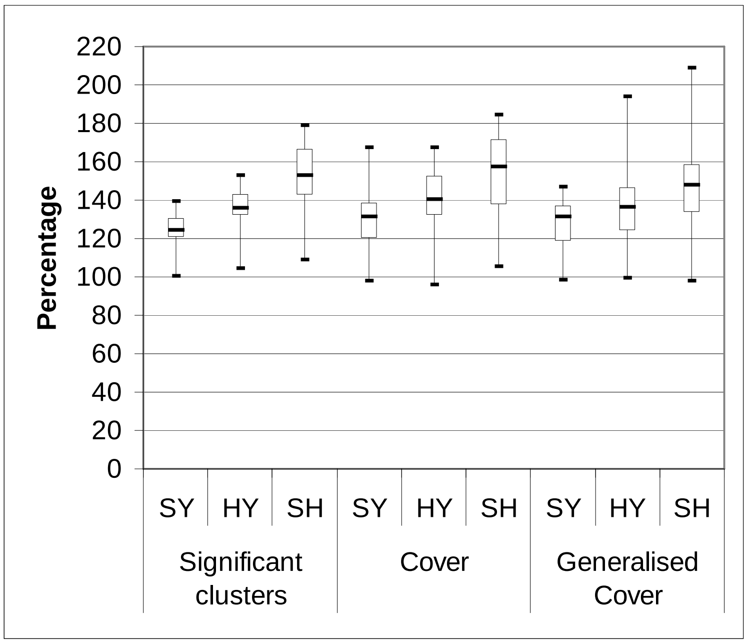
On average, there are more words per rule when clustering after adding descriptors than when clustering before adding descriptors. On average, when using SY there are fewer words per rule than when using HY, which in turn generates fewer rules than when using SH.

On average, when using SY or HY, the cover cluster set has greater increase, in the number of words per rule, than the generalised cover cluster set, which in turn, has greater increase than the significant cluster set. In contrast when using SH, the cover cluster set has greater increase, in the number of words per rule, than the significant cluster set, which in turn, has greater increase than the generalised cover cluster set.

The statistical distributions in the figure show that almost all the data sets have more words per rule when clustering after adding descriptors than when clustering before adding descriptors.



a: Average



b: Distributions

Figure 6.17: Change to the number of words per rule

The execution time

Figure 6.18 shows the changes to the clustering execution time when more descriptors are added to the descriptor sets of the instances.

When clustering after adding descriptors using SY, HY or SH, on average, the execution time is more than that when clustering before adding descriptors. On average, the execution time, when clustering using SH, is more than that when using HY, which in turn is more than that when using SY. About 7.8 times longer when using SH, but only about 2.8 times longer when using HY, and only about 2.6 times longer when using SY, compared with the execution time when clustering before adding descriptors.

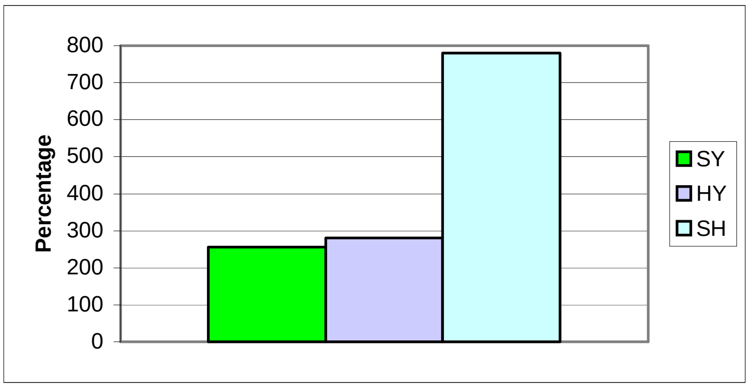


Figure 6.18: Average change to the clustering execution time

6.4 Analysis and discussion

The following sections analyse and discuss the results shown in the results section.

Clusters

On average, there are fewer clusters in the significant cluster set after adding descriptors than before adding descriptors (figure 6.3). Although there is not a great reduction in the number of the significant clusters, the significant clusters when clustering after adding descriptors are quite *different* from when clustering before adding descriptors: there is a relatively low percentage of the retained significant clusters (figure 6.4). Noticing the relative high percentage of the subsumed significant clusters in the new ones (figure 6.5), this means that some of the new significant clusters, when clustering after adding descriptors, are bigger by subsuming other significant clusters from clustering before adding descriptors.

On average, after adding descriptors, the minimum size for significant clusters increased (figure 6.7). This means that some of the clusters from before adding descriptors are no longer considered significant.

This decrease in the number of significant clusters was a surprise, particularly given that the size of the set of *all-clusters* increased (from an average of about 43,000 clusters, before adding descriptors, to 71,000 for SY, 81,000 for HY, and 96,000 for SH after adding descriptors). Usually for a clustering experiment if the size of the set of *all-clusters* increases then the size of the minimum size for the clusters to be considered statistically significant increases too.

We believe that the reason for the decrease in the number of significant clusters is that the minimum size for significant clusters also increased. Therefore, although MAXCCLUS found more clusters, fewer of them were counted as significant, and therefore, it was not able to explain as much of the data.

Instances

On average, after adding descriptors, there is a decrease in the number of instances that MAXCCLUS can describe (figure 6.8). The instances described are *almost* as the same as the ones described before adding descriptors: a *high* percentage of the instances are retained (figure 6.9), and a *low* percentage of the instances are new (figure 6.10). Therefore after adding descriptors MAXCCLUS is not describing more instances than before adding descriptors, instead it is describing fewer instances on average and these instances are almost the same instances that MAXCCLUS was able to describe without the added descriptors.

Having a different set of clusters describing almost the same set of instances means that adding descriptors cause MAXCCLUS to spend more time in clustering (figure 6.18) without being able to increase the number of instances that it is able to describe.

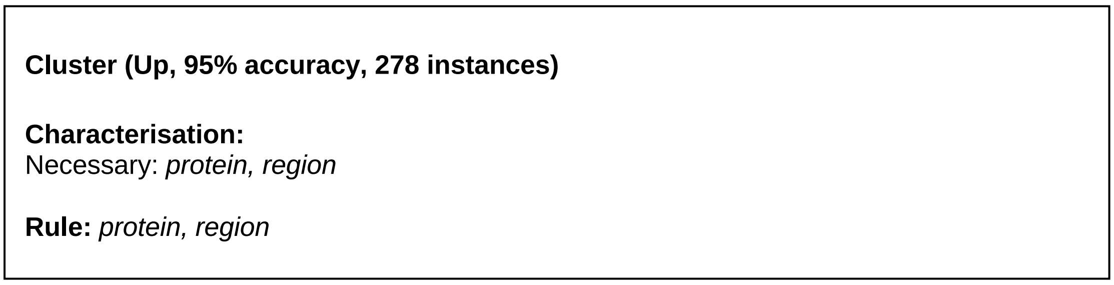
Rules

After adding descriptors, MAXCCLUS creates *many more* rules than before adding words, to characterise the clusters it found (figure 6.11). Note that these rules characterise almost the same set of instances as before. Having many more rules is confusing and does not help a user trying to make sense of the clustering results. The number of rules increased dramatically because MAXCCLUS can create many alternative rules using the added words that have the similar meaning.

In some cases, the number of rules for a cluster does not change, but even then, the rules may be different. For example, figure 6.19 shows a significant cluster that MAXCCLUS creates when clustering before adding descriptors. This cluster has 278 instances that are described by the rule "*protein, region*". Using the same data set, when clustering after adding synonyms, MAXCCLUS creates the same cluster that has the same 278 genes, but with different rule, "*protein*, *neighborhood*", to describe the instances (figure 6.20).

What happened is that the previous version of the cluster has the words "protein" and "region" in the set of the *Necessary* words. After adding synonyms, the descriptors "*area*", "*domain*", "*neighborhood*", and "*part*" were added to all the instances that had the word "*region*" in their descriptor set. These synonyms were added because they exist in the set of *all-descriptors* (other synonyms of the word "*region*" and the synonyms of the word "*protein*" were not added because they do not exist in the set of *all-descriptors*). When clustering after adding synonyms, the word "*neighborhood*" replaced the word "*region*" in the set of the *Necessary* words, therefore the rule that describe the instances became "*protein, neighborhood*" (figure 6.20).

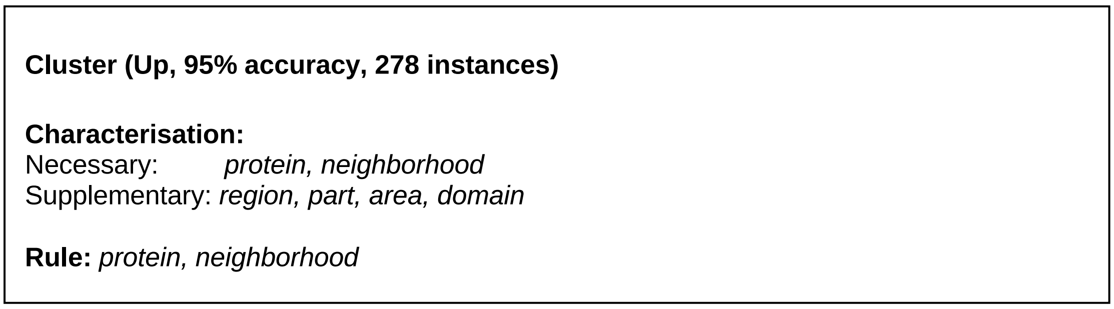
This seems ok but when comparing the frequency of the word "*region*" before adding descriptors, which is 2002, with the frequency of the word "*neighborhood*" which is 1, the word "*region*" is more familiar than the word "*neighborhood*". Before adding descriptors, the word "*region*" appears in the descriptor sets of *2002* instances, in contrast the word "*neighborhood*" appears only in the descriptor set of *1* instance. Which make the word "*region*" more frequent in use than the word "*neighborhood*". It is not desirable to have a much less frequent word replacing a more frequent one in a rule, because the more frequent word is probably more familiar than its synonyms to the people that work in the domain (in this case the genetics scientists), and may have important connections with the domain.


Cluster (Up, 95% accuracy, 278 instances)

Characterisation:
Necessary: *protein, region*

Rule: *protein, region*

Figure 6.19: A cluster before adding synonyms


Cluster (Up, 95% accuracy, 278 instances)

Characterisation:
Necessary: *protein, neighborhood*
Supplementary: *region, part, area, domain*

Rule: *protein, neighborhood*

Figure 6.20: The same cluster of figure A.1 but having different rule after adding synonyms.

On the other hand, in most of the cases the number of rules increases after adding descriptors. For example suppose that, when clustering before adding descriptors, MAXCCLUS finds a rule that has the following words: w1, w2, and w3. Suppose that s1 is a synonym for w1, s2 is a synonym for w2, and s3 is a synonym for w3. Therefore, when clustering after adding using SY, to characterise a cluster MAXCCLUS creates the rule “w1, w2, w3” and its alternatives, which are:
“s1, w2, w3”
“w1, s2, w3”
“w1, w2, s3”
“s1, s2, w3”
“s1, w2, s3”
“w1, s2, s3”
“s1, s2, s3”

to characterise the same cluster. These alternative rules are likely to confuse the user who is trying to make sense of the cluster. Therefore, adding descriptors using SY is not useful. The same process happens when clustering using HY, but the increase in the number of rules, although it is not as great as when using SY, is still a large increase. Suppose that h1 is a hypernym for the w1, h2 is a hypernym for the w2, and h3 is a hypernym for the w3.

The worst case is when using SH where the number of rules jumps very high for the significant clusters. This is because using SH is combining the use of the approaches SY and HY. Therefore, the number of rules when using SH worsens the rules' results, even more than when using SY or HY, by dramatically increasing the number of rules giving the worst results. Although it is still high, the number of rules of the cover cluster set is not as high as that of the significant cluster set, therefore covering the significant clusters with the cover cluster set is working fine. Even more, the generalisation is working fine because the number of rules of the general cluster set, although it is still a large increase, is the best case among the three sets of clusters, which reflects an enhancement of the generalised cover cluster set over the cover cluster set.

6.5 Conclusion and recommendations

Adding synonyms, hypernyms (or both) to the descriptor sets of the instances does not help MAXCCLUS to describe more instances, instead MAXCCLUS describes fewer instances, takes more time for clustering, and produces clusters that are characterised with many alternative rules that can confuse the user who is trying to make sense of the clusters. Therefore adding synonyms, hypernyms or both of them is a bad idea. It may be possible to get better clustering results if we can add synonyms and/or hypernyms of the domain-relevant words but WordNet does not contain all of these. It might be better to replace the descriptors with their most frequent (relevant to the textual data) synonyms or hypernyms; this will prevent MAXCCLUS from creating alternative rules.

A topic for future work is to replace the domain-relevant descriptors with there frequent synonyms and/or hypernyms.

Chapter 7

Five Categories

This chapter describes a clustering experiment designed to study the ability of MAXCCLUS to cluster instances that have five adjacent categories and comparing its results with the reference experiment where MAXCCLUS clusters instances that have two distinct categories only. Unfortunately, the result of this chapter is to show that MAXCCLUS is not as good for clustering instances that fall in multiple adjacent categories, as it is for clustering instances that belong to two clearly distinct categories.

This clustering experiment uses the same data that the reference clustering experiment uses, but with an important difference: the number of the categories and how close the categories are. The reference clustering experiment uses two categories to categorise instances (genes). The two categories are *Up* and *Down*. On the other hand, this chapter's clustering experiment uses five categories: *Up*, *MidUp*, *Mid*, *MidDown*, and *Down*. Of these five categories, the *Up* and *Down* categories are the same *Up* and *Down* categories as in the reference clustering experiment.

The actual raw experimental data allocates each gene a numeric expression ratio, which varies between approximately 0.001 and 1000.0. The discrete categories represent subranges of the expression ratio. In both the two categories and the five categories experiments, the *Up* and *Down* represent the ranges: greater than or equal to 2.0, and less than or equal to 0.5 respectively (Figure 7.1). In the five categories experiment, the categories *MidUp*, *Mid*, and *MidDown* cover the remaining part of the range. It is significant that in the two categories experiment, all the genes in the range between *Up* and *Down* are excluded, so that the two categories are quite distinct. In the five categories experiment, all the genes are included, so that the categories are adjacent to each other and are less distinct. Also, the five categories experiment includes 3,548 genes, whereas, on average, only 921 genes are included in each data set of the two categories experiment.

Category	**Expression ratio X**
Up	$2.0 \le X$
MidUp	$1.4 < X < 2.0$
Mid	$0.7 \le X \le 1.4$
MidDown	$0.5 < X < 0.7$
Down	$X \le 0.5$

Figure 7.1: Categories versus their expression ratios.

7.1 Textual data

The textual data, for the five categories clustering experiment is the same as that of the reference clustering experiment; it represents the association of each gene's id with the gene's descriptors.

7.2 Experimental results

The results shown in the figures below are for running MAXCCLUS on 19 of the microarray experimental data sets (data set: 3, 6, 7, 10, 11, 12, 13, 16, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, and 30), using five categories instead of two categories. Figures 7.2 to 7.13 show the results for the number of clusters, the number of instances, the number of rules, the number of words in the rules, and the number of words per rule.

The clusters

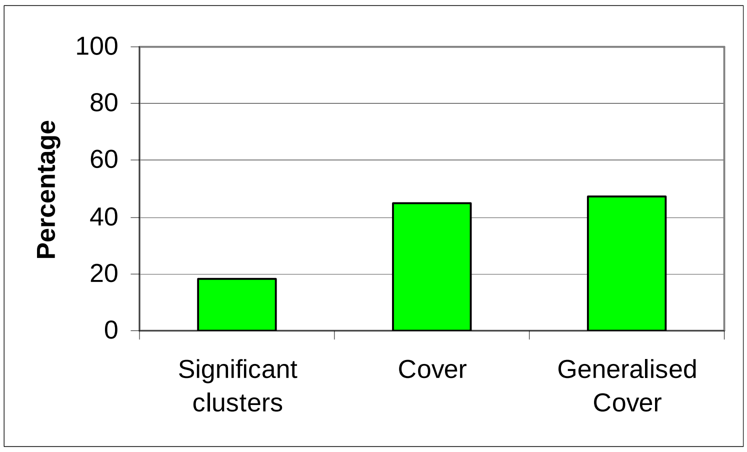
Figure 7.2 shows the changes to the total number of clusters when MAXCCLUS clusters instances using five categories instead of two categories.

On average, there are fewer clusters when clustering instances using five categories, than when clustering instances using two categories (between 18% and 47% as many). The significant cluster set has greater decrease in the number of clusters than the cover cluster set, which in turn has greater decrease than the generalised cover cluster set.

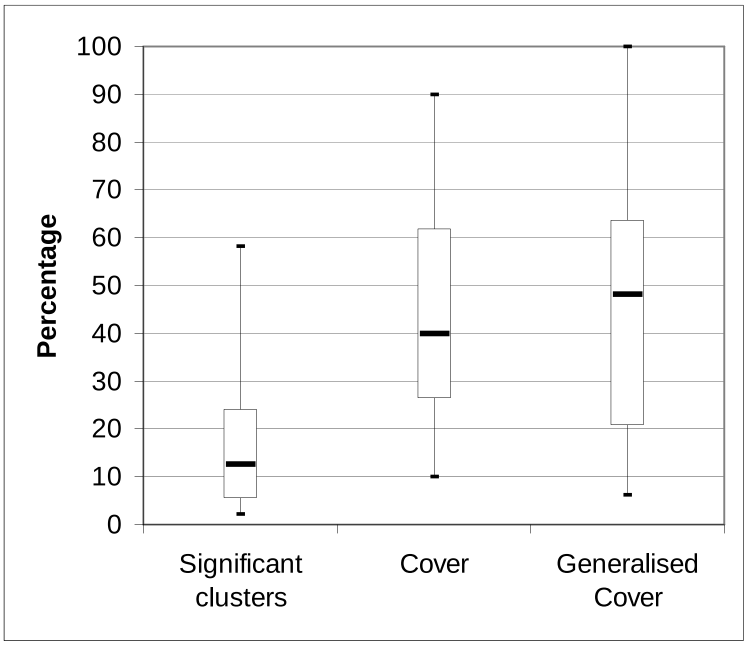
The distributions show that the significant cluster set has the narrowest distribution (the maximum value is an outlier) of the cluster sets, the narrowest inter-quartile range, it has the lowest median, and is skewed to lower values. The cover cluster set is more spread in its distribution, has a higher median but is still skewed to the lower values. The generalised cover is has the most spread distribution and the highest median.

When clustering instances using five categories a completely different set of clusters was found: no clusters were *retained* from the clusters found when clustering instances using two categories, and therefore all the clusters are *different* from the clusters found when clustering instances using two categories.

No clusters found from clustering instances using two categories are *subsumed* by new clusters found from clustering instances using five categories.



a: Average



b: Distributions

Figure 7.2: Change to the total number of clusters

Significant clusters minimum size

Figure 7.3 shows the changes to the minimum size of the significant clusters for the *Up* and *Down* categories when clustering using five categories compared with when clustering using two categories. Only the minimum size for the *Up* and *Down* can be compared because there are no other categories available when clustering genes using two categories.

When clustering instances using five categories, on average, the significant clusters' minimum sizes for the *Up* and *Down* categories are less than that when clustering instances using two categories.

On average, the significant clusters' minimum size for the *Up* category has greater decrease than that for the *Down* category.

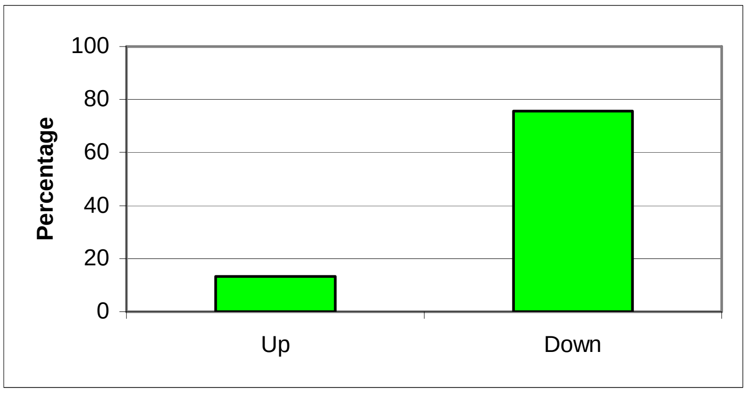


Figure 7.3: Average change to the significant clusters' minimum size for the *Up* and *Down* categories

On the other hand, Figure 7.4 shows the average of the significant clusters' minimum size for each of the five categories (not as a percentage value). The figure shows that the *Mid* category has higher minimum significant cluster size than the other categories.

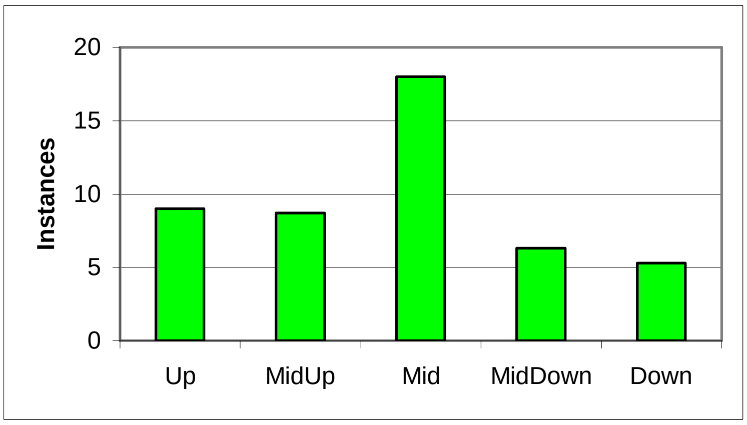


Figure 7.4: Average of the significant clusters' minimum size for each of the five categories (not as a percentage value)

The instances

Figures 7.5 to 7.7 show the changes to the number of instances when MAXCCLUS clusters instances using five categories instead of two categories. The figures show the changes to the total number of instances, the number of retained instances, and the number of new instances.

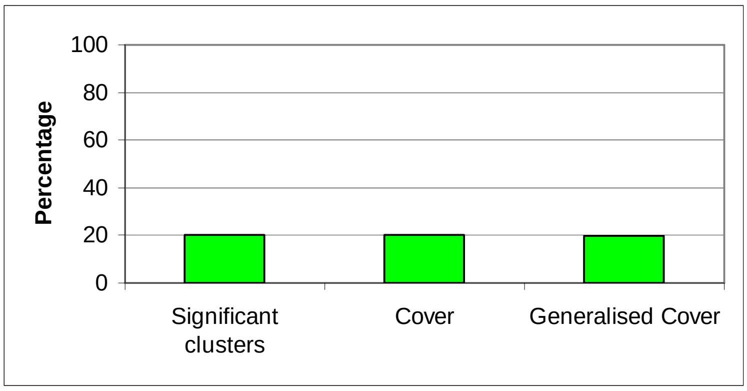
In each figure, the results of the significant and the cover cluster sets are the same because both sets should cover the same instances.

On average, there are fewer instances when clustering instances using five categories, than when clustering instances using two categories (about 20% as many) (figure 7.5), in spite of the fact that there are on average of four times as many instances in the experimental data sets.

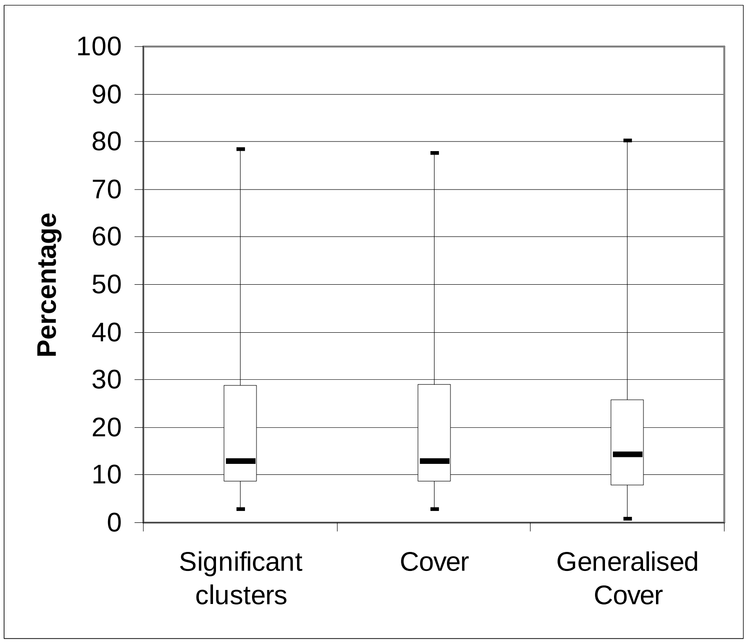
Although the averages for all the cluster sets are the same, the distributions in show that the significant and the cover cluster sets have a narrower inter-quartile range and are more skewed towards the minimum values with slightly lower median than the generalised cluster set (figure 7.5). All the maximum values are outliers and they belong to one data set (data set 13), while the value next to the maximum value is about 45%, which is not an outlier. Also, all the minimum values belong to another single data set (data set 20).

When clustering instances using five categories, on average, very few instances are *retained* from the clusters found when clustering instances using two categories (about 6%) (Figure 7.6), and about 56% of the instances are *different* from the instances found when clustering instances using two categories (Figure 7.7).

Although Figure 7.7 shows that the average of the number of new instances is almost the same for all the cluster sets, the distribution show that the median of the generalised cover cluster set is lower than that of the other cluster sets. Also this figure shows that all the cluster sets have a very widely spread distribution; all have a wide inter-quartile range especially the generalised cover. Three data sets (data set 3, 13, and 23) share all the maximum values, while two data sets (data sets 6 and 7) share all the minimum values.



a: Average



b: Distributions

Figure 7.5: Change to the total number of instances

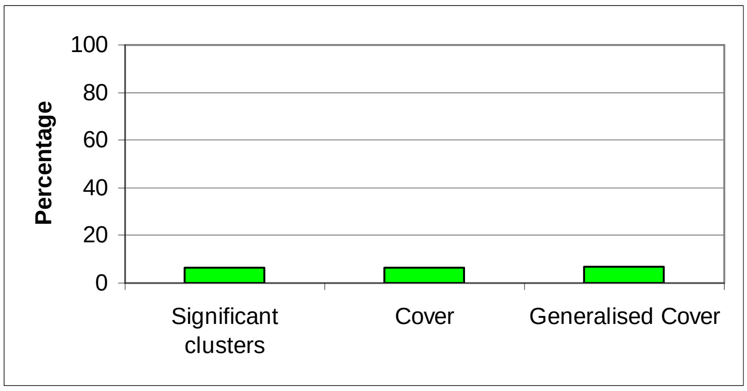
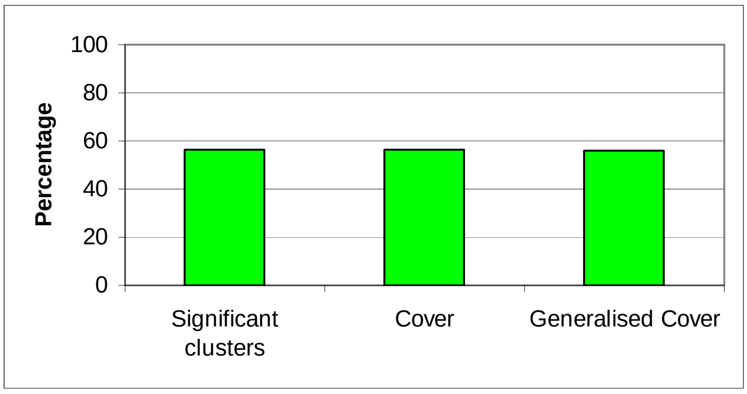
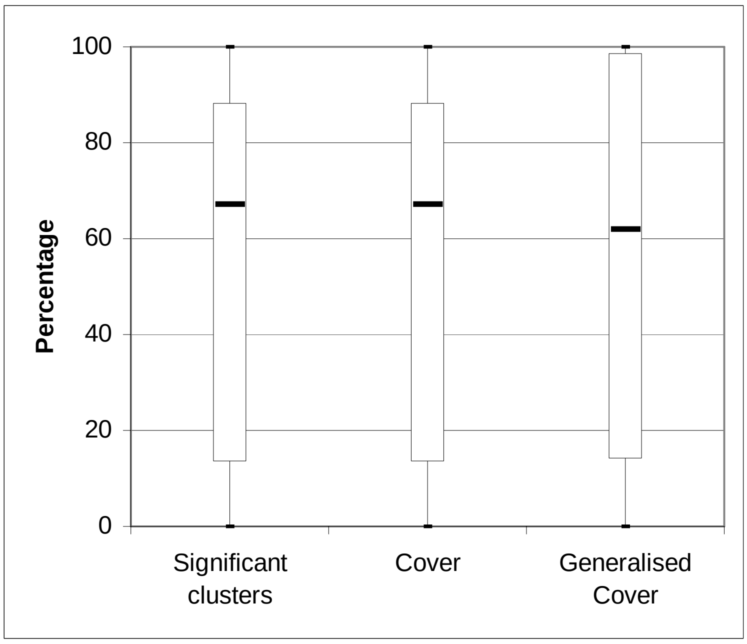


Figure 7.6: Average number of retained instances



a: Average



b: Distributions

Figure 7.7: Number of new instances

Figure 7.8 shows the average of the number of instances per category for each of the five categories when clustering using five categories (not as a percentage value). The figure shows that the *Mid* category has the highest average number of instances per category.

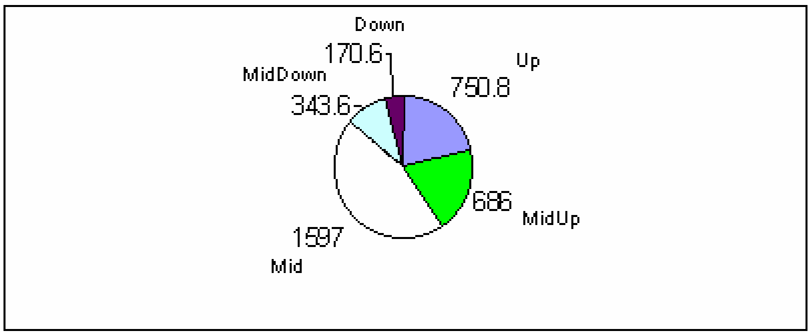


Figure 7.8: Average of the number of instances per category for each of the five categories when clustering using five categories (not as a percentage value)

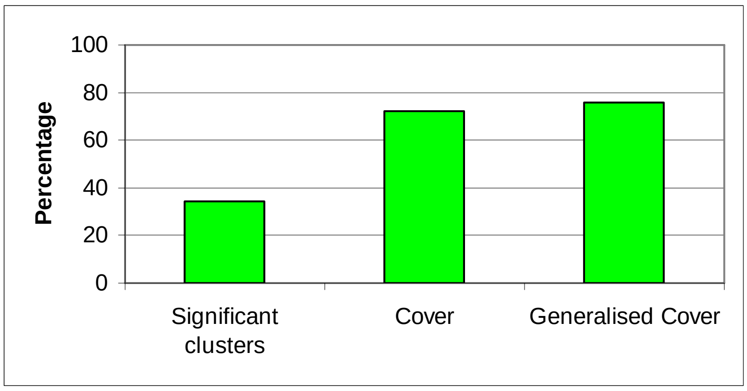
The rules

Figure 7.9 shows the changes to the total number of rules when clustering instances using five categories instead of two categories.

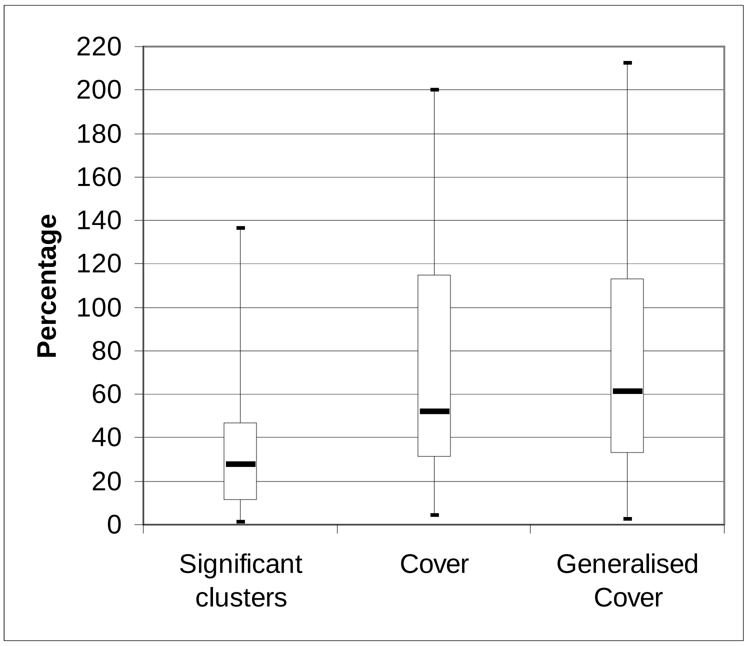
On average, there are fewer rules when clustering using five categories, than when clustering using two categories (between 34% and 76% as many). The significant cluster set has greater decrease in the number of rules than the cover cluster set, which in turn has greater decrease than the generalised cover cluster set.

When clustering instances using five categories, no rules are *retained* from the rules found when clustering instances using two categories, and no rules found from clustering using two categories are *subsumed* by new rules found from clustering using five categories.

The distributions show that the significant clusters have a narrower distribution than the other cluster sets. The maximum values of the cover and the generalised cover cluster sets belong to data set 12, while the minimum values of the significant and the cover cluster sets belong to data set 24.



a: Average



b: Distributions

Figure 7.9: Average change to the total number of rules

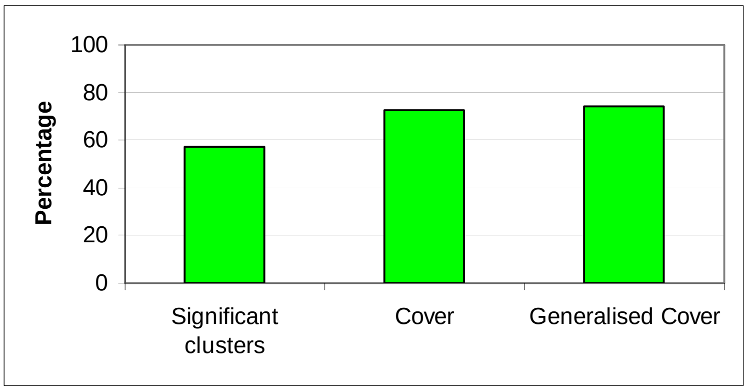
The rules' words

Figures 7.10 to 7.12 show the changes to the number of rules' words when clustering using five categories instead of two categories. The figures show the changes to the total number of rules' words, the number of retained rules' words, and the number of new rules' words.

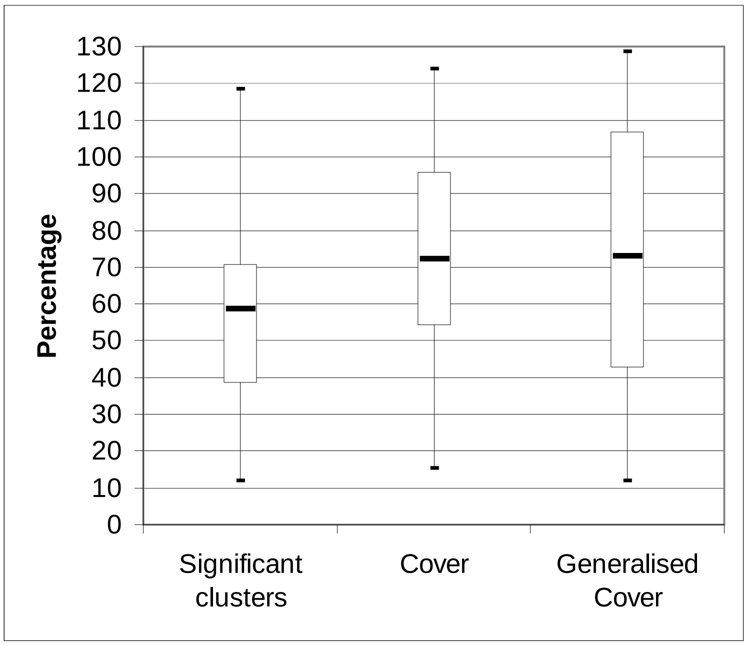
On average, there are fewer rules' words when clustering using five categories, than when clustering using two categories (between 57% and 74% as many). The significant cluster set has greater decrease in the number of rules' words than the cover and the generalised cover cluster set (figure 7.10). The distributions show that there is a wide variation across the data sets.

When clustering using five categories, on average, between 33% and 35.4% of the rules' words are *retained* from the rules found when clustering using two categories (figure 7.11). The average is almost the same for all the cluster sets.

When clustering using five categories, on average, between 42% and 53.5% of the rules' words are *different* compared with the rules' words found when clustering using two categories (figure 7.12). The significant cluster set has fewer different rules' words than the other cluster sets.



a: Average



b: Distributions

Figure 7.10: Change to the total number of rules' words

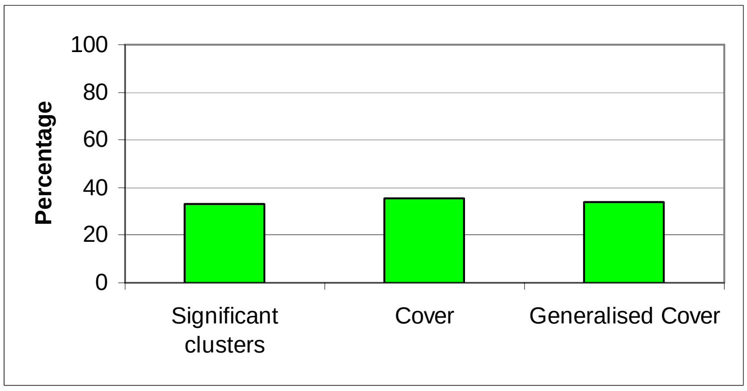


Figure 7.11: Average number of retained rules' words

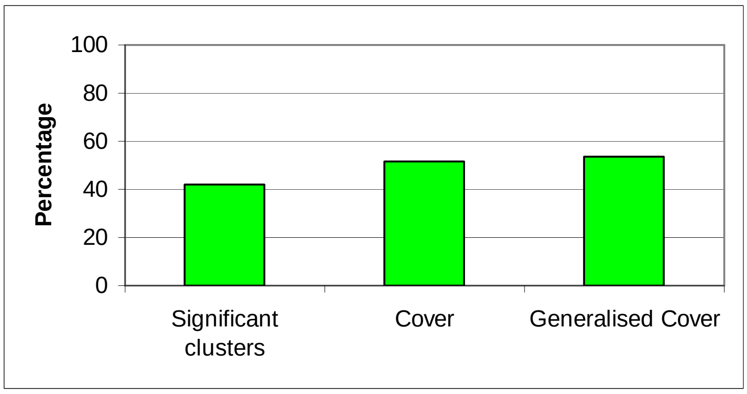


Figure 7.12: Average number of new rules' words

The words per rule

Figure 7.13 shows the changes to the number of words per rule when clustering using five categories instead of using two categories

On average, there are more words per rule when clustering using five categories, than when clustering using two categories (between 136% and 151% as many). The generalised cover cluster set has a greater increase in the number of words per rule than the other cluster sets.

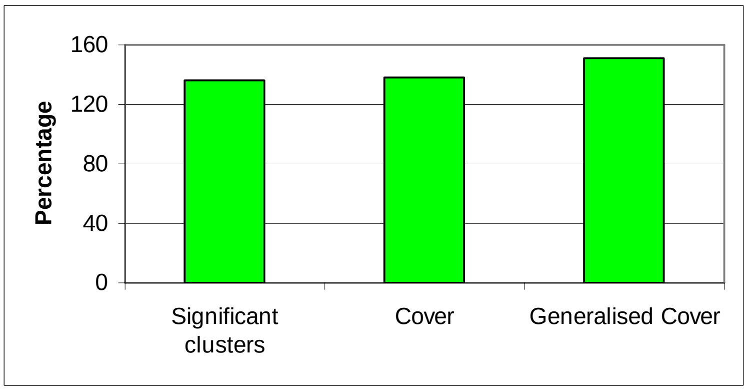


Figure 7.13: Average change to the number of words per rule

The execution time

When clustering using five categories, on average, the execution time is much more than that when clustering using two categories - about 15 times longer when using five categories compared with the execution time when clustering using two categories.

7.3 Analysis and discussion

The following sections analyse and discuss the results shown in the experimental results section.

Clusters

On average, there are fewer clusters when clustering instances using five categories, than when clustering instances using two categories, especially the significant cluster set (figure 7.2). The clusters MAXCCLUS creates when clustering instances using five categories are completely *different* from the clusters it creates when clustering instances using two categories.

The difference in the clusters is the result of including many new instances (the ones with categories *MidUp, Mid* or *MidDown*) in the data sets in addition to the ones used in the reference clustering experiment (the ones with categories *Up* or *Down*).

When clustering instances using five categories, the minimum size of clusters to be considered statistically significant clusters by MAXCCLUS, decreases for the *Up* and *Down* categorised clusters compared to that for clustering instances using two categories (figure 7.3). The minimum size of the *Down* categorised clusters decreased from about 7 instances to about 5 instances, and the minimum size of the *Up* categorised clusters decreased from about 89 instances to about 9 instances.

The reason for the reduction in the minimum size for the *Up* and *Down* clusters is not because the size of the set of *all-clusters* decreased, since that increased to 450,000 clusters for all the data sets when clustering instances using five categories (it is on average only 35,000 clusters, when clustering instances using two categories). The set of *all-clusters* is the set of clusters that MAXCCLUS creates from the descriptor sets of the data set's instances. Usually for a clustering experiment if the size of the set of *all-clusters* increases then the size of the minimum size for the clusters to be considered statistically significant clusters increases too.

When MAXCCLUS clusters instances using only two possible categories (*Up* and *Down*) MAXCCLUS would have to have large clusters to be considered significant, because there is a good chance that most of some cluster's instances belong to one category when there are only two possibilities. On the other hand when MAXCCLUS clusters instances using five possible categories, a small pure cluster would be considered significant because it is very unlikely that, by chance, most of any cluster's instances belong to a single category when there are five possibilities.

Notice that, on average, the minimum size of a cluster to be considered statistically significant (figure 7.4), increases with the number of instances for this category in the data set (figure 7.8). The minimum size of the *Mid* categories clusters is more than that of any other category clusters, because the number of the instances categorised as *Mid* regulated instances is more than the number of any other category instances.

Although the minimum size of a cluster to be considered significant reduces when clustering using five categories, the number of significant pure clusters unexpectedly decreases considerably instead of increasing (figure 7.2). The reduction in the number of significant pure clusters means it is *harder* to find significant clusters that are pure in the real data when clustering using five categories than when clustering using two categories. To explain this, we give an example.

Suppose, when clustering using two categories, there is an *Up* cluster of size 7, and the purity is required by the user to be 100%, which means that a cluster can be pure only if all its instances have the same category; in other words MAXCCLUS excludes from the significant pure cluster set any *Up* cluster that has at least one instance not of *Up* category. On the other hand when clustering using five categories two more instances that have some words in common with the cluster were added to it to become a cluster of size 9. These two new instances are *MidUp* regulated. MAXCCLUS excludes this cluster from the significant pure cluster set even though the *MidUp* instances (which are not clearly *Up* regulated) could be in reality *Up* ones but their regulations were measured as *MidUp* because of microarray experimental noise. Therefore, the cluster becomes bigger but also becomes impure.

When clustering using two categories MAXCCLUS can easily create rules that distinguish between the two separated categories (*Up* and *Down*). But when using five categories MAXCCLUS has difficulty creating rules that can distinguish between any two *adjacent* categories that do not have a gap in between (*Up* and *MidUp*, *MidUp* and *Mid*, *Mid* and *MidDown*, or *MidDown* and *Down*). This is not the behaviour of MAXCCLUS only. In [Sahami, M; Dumais, S; Heckerman, D; and Horvitz, E. 1998] they mentioned the same behaviour with their Spam filtering system, where the system was able to distinguish between ordinary and Spam messages, but it behaved poorly when it attempts to distinguish between three kind of messages: ordinary messages, one kind of Spam messages and other kind of Spam messages.

Instances

On average, when clustering instances using five categories, there is a large decrease in the number of instances that can be described - only 20%, even though this includes the instances that have the categories *MidUp*, *Mid*, or *MidDown* which cannot be described when clustering using two categories (figure 7.5). This decrease in the number of described instances is despite the fact that all the 3548 instances, available from the microarray raw experimental data, are included in all the experimental data sets that MAXCCLUS used to cluster using five categories.

Furthermore, the individual instances that MAXCCLUS is able to describe when clustering using five categories are quite *different* from the instances it can describe when clustering using two categories: a *high* percentage of the instances are new, about 56% (figure 7.7). The primary reason is the reduction in the number of significant pure clusters (as explained in the previous section), since only instances in significant pure clusters are explained.

The algorithm does not scale well to several thousands of instances; the cost of the search grows too fast. The time was much longer because of more instances for the data sets and more descriptors contributed by them. Note that it is the significance test, rather than the construction of the clusters that dominates the time.

Rules and Words per rule

On average, there are fewer rules when clustering instances using five categories, than when clustering instances using two categories, especially the significant cluster set (figure 7.9), and all the rules are different. The number of rules does not decrease as much as the number of clusters (figure 7.2), nor as much as the number of instances (figure 7.5). This means that each cluster must have more possible rules because there are many alternative rules due to adding new instances that contribute with their descriptors.

This is a consequence of having smaller clusters, which therefore are likely to have more words in common. If there are many words associated with the cluster, there are likely to be many possible combinations of these words (that is many rules) that will distinguish just the instances in the cluster.

The words per rule increased (figure 7.13) because of adding new instances (the ones that have categories *MidUp*, *Mid*, or *MidDown*) to the data sets for clustering using five categories. Because there are more instances altogether, MAXCCLUS needs to use more words in each rule to be able to exclude the instances that do not belong to the cluster that the rule characterise (describe).

Rules' words

In spite of more words per rule, there are still fewer words altogether in the rules when clustering instances using five categories, than when clustering instances using two categories, especially the significant cluster set (figure 7.10). The rules' words when clustering instances using five categories are quite *different* from the rules' words when clustering instances using two categories: on average, only about 34% of the rules' words are retained (figure 7.11), and between 42% and 54% of the rules' words are new (figure 7.12).

The difference in the rules' words is because of the difference in the rules that previous section describes.

7.4 Conclusion

MAXCCLUS is not able to produce as good results when clustering using continues range of categories, as when clustering using two categories separated with a clear gap. It is not able to explain nearly as many instances, even though it constructs almost as many rules as when clustering using two categories. When using continues range of categories (*Up*, *MidUp*, *Mid*, *MidDown*, and *Down*) it is difficult for MAXCCLUS to distinguish between any two *adjacent* categories that do not have a gap in between (*Up* and *MidUp*, *MidUp* and *Mid*, *Mid* and *MidDown*, or *MidDown* and *Down*). With respect to the clustering time, the algorithm does not scale well to several thousands of instances as the cost of the search grows too fast.

7.5 Recommendations

If one needs to use MAXCCLUS to cluster instances that fall in more than the clearly *Up* or clearly *Down* categories, it is better to be sure that the categories are separated with a gap. Therefore one might be able to use MAXCCLUS to cluster instances that fall in three categories *Up*, *Mid*, and *Down*, where there is a gap between *Up* and, *Mid*, and between *Mid* and *Down* (figure 7.14). Or clustering by using two categories separated by a gap, one category is *Up-MidUp* (by merging *Up* and *MidUp*), and the other is *Down-MidDown* (by merging *Down* and *MidDown*), in this case the gap would be ignoring the instances that belong to the *Mid* category (figure 7.15). These two suggested clustering experiments are for future research.

Category	**Expression ratio X**
Up	2.0 ≤ X
Mid	0.7 ≤ X ≤ 1.4
Down	X ≤ 0.5

*Figure 7.14: Recommended **three** categories versus their expression rations with **gaps** in between*

Category	**Expression ratio X**
Up-MidUp	1.4 < X
Down-MidDown	X < 0.7

*Figure 7.15: Recommended **two** categories versus their expression ration separated with a **gap** (it is **different** version than that we used in this project)*

Chapter 8

Tools

The focus of the project was on the experiments that investigated various modifications to MAXCCLUS and its data. However, the project also required the development of several tools to assist in the conduct of these experiments. This chapter briefly describes these tools.

8.1 Modifications to MAXCCLUS and its data

We modified MAXCCLUS by designing and implementing the exhaustive search generalisation (used in chapter 3) using the *Java* computer programming language. We modified the data that MAXCCLUS used using the *Perl* computer programming language.

8.2 Comparison used in methodology

To carry out the task of comparison of thousands of the objects in tens of XML files that MAXCCLUS produced as the results of its clustering throughout all the clustering experiments, we designed and implemented computer programs using the *Java* computer language with *SAX* (Simple API for XML) and *Xerces* parser from Apache, to parse the XML files, extract the knowledge, compare, and report the comparison results.

8.3 Rendering

MAXCCLUS reports its clustering results for a microarray experiment data set as an XML file. To easily view these results, they need to be rendered to HTML (HyperText Markup Language) files. Rendering was originaly done by running a Java program on each of the XML files to create the respective HTML file. This approach was not flexible enough in case one needs to change how the HTML presents the data; furthermore this approach doubles the computer storage size needed to store the clustering results (because of having both the XML and the HTML files of the results), which can be considerable when running MAXCCLUS on many data sets (some XML and HTML files can exceed 34MB in size). We modified the rendering by using only one small (7KB) XSL (Extensible Stylesheet Language) file to render, on the fly, any XML file created by MAXCCLUS, to a HTML file by using Microsoft Internet Explorer to open the XML file.

8.4 Web robot

Some microarray experiment data sets have gene ids that are not in the same format of the *Swiss-Prot* textual database. To be able to use these gene ids, we need to map these ids to the format that *Swiss-Prot* uses. This can be done by accessing websites of some gene databases and submitting a query to get back a web page that contains different gene id formats from which we can select the id that has the *Swiss-Prot* format.

Doing this manually is not a problem for few genes, but for thousands of genes it is time consuming, error prone, and frustrating. Therefore we designed and implemented a *web robot*, using the *Perl* computer programming language, to carryout the task automatically.

To retrieve the *Swiss-Prot* format gene ids of the non *Swiss-Prot* format gene ids, we chose the web site of the *NBCI* (*National Center for Biotechnology Information*) at "http://www.ncbi.nih.gov", which is a resource for molecular biology information and public databases.

In one run of the web robot program, the web robot access the web site of the *NBCI* thousands times (depending on the number of genes). In each access, it submits a query for one gene id (saved previously in a file of gene ids), waits for the web page that holds the results of the query, extracts the *Swiss-Prot* gene id format from the results page, and save the gene id with its *Swiss-Prot* id format into a gene ids mapping file. We can then use the gene ids mapping file when we extract the textual data from the *Swiss-Prot* database.

8.5 Clusters visualisation

We needed a way to visualise the clusters to get some idea about how the instances are distributed among the clusters of the results of MAXCCLUS clustering for a microarray experiment data set.

We designed and implemented a program using the Java computer programming language to automatically produce *PNG* (Portable Network Graphics) image files from the *XML* (Extensible Markup Language) clustering result files that MAXCCLUS produce. We chose the *PNG* format for the images because it is patent-free unlike, some other image formats.

The images produced are usually big. Figure 8.1 shows one of the relatively small images that it can be viewed clearly on an A4 size page. On the other hand Figure 8.2 shows another image, which is not clearly viewable on an A4 size page because the image was shrunk to fit the page size, but it can still give some idea about the clusters and their instances (if the reader is viewing a PDF – Portable Document Format – file of this thesis, the image can be viewed clearly by magnifying the document).

In the images, the *x* and *y* coordinates represent the clusters and the instances respectively. The clusters are sorted from left to right in ascending order with respect to the number of instances they have. The instances are sorted from bottom to top in an ascending order with respect to the number of clusters they are in.

The vertical coloured lines are for the different categories of the clusters. The horizontal coloured boxes represents the instances, we made them coloured to make it easy to distinguish between the adjacent instances.

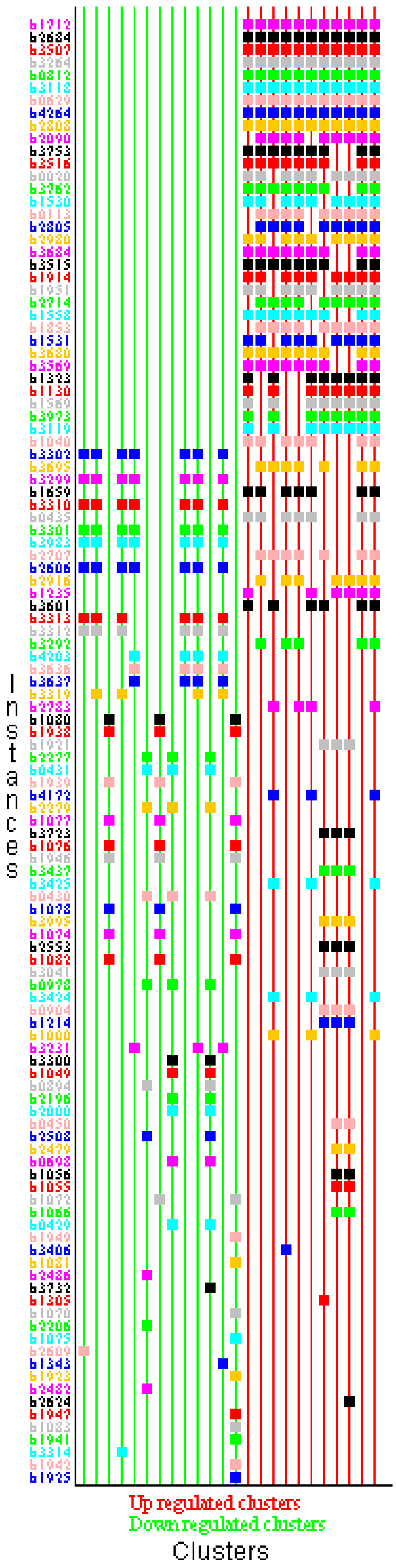


Figure 8.1: *A visualisation for the clustering results of one data set*

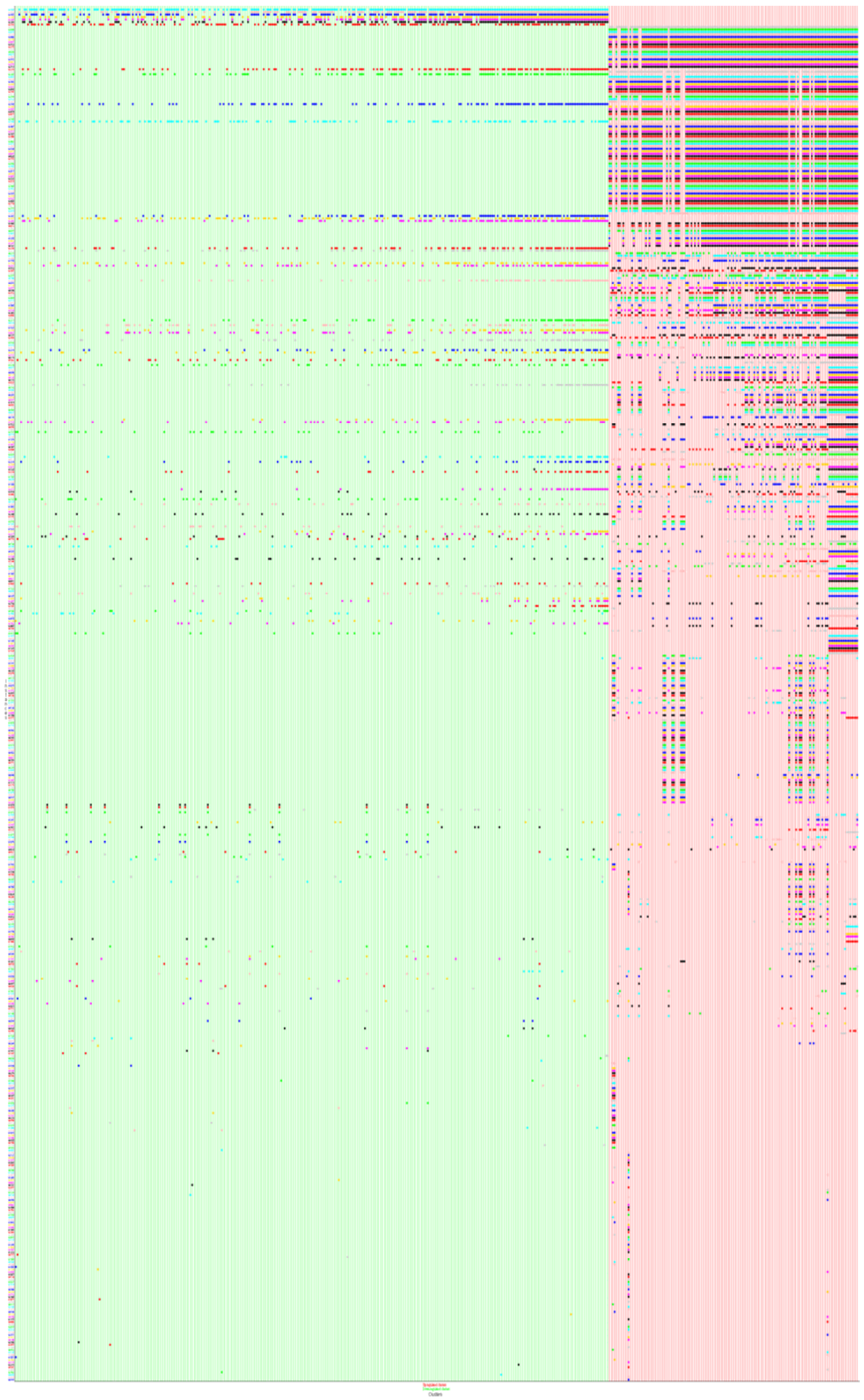


Figure 8.2: *A visualisation for the clustering results of another data set*

Chapter 9

Conclusion

The clustering experiments we carried out through chapters three to seven showed that in general MAXCCLUS is doing a reasonably good job in clustering instances and characterising them to make it easy for the user (the scientist) to make sense of the microarray experimental data. The experiments also identified some limitations of MAXCCLUS, and showed that some improvements that we had expected would help did not in fact improve MAXCCLUS.

Although MAXCCLUS was able to describe, on average, only about 37% of the instances of the microarray experimental data (with a range between about 5% to about 79%) using its simple generalisation approach (see section 2.3), we could not increase the number of instances described by MAXCCLUS by using an exhaustive search for generalisation. This could be because of the relatively high accuracy value (95%) that was used in the clustering experiment (see Chapter 3). However, the exhaustive search generalisation was much more expensive than the simple generalisation.

MAXCCLUS depends on the significance test to select from the set of all-clusters only the clusters that are statistically significant as specified by the user accuracy value (95%). The clustering experiment of chapter four examined the significance test by using MAXCCLUS to cluster operons instead of genes. An operon is a group of genes that are functionally dependent on each other during a microarray experiment; therefore they have dependency relation. The result of the clustering experiment of chapter four showed that, when clustering genes, MAXCCLUS ignored the fact that some genes are dependent on other genes. Ignoring this fact led MAXCCLUS to select clusters that are big enough to be considered statistically significant but were not really significant. The MAXCCLUS significance test needs to be modified to consider the dependency of some genes on others, so that a cluster will be considered significant only if it consists of enough independent instances.

On the side of textual data, the clustering experiment of chapter five showed that it is advantageous to delete some of the descriptors of the instances providing that these descriptors are not domain-relevant ones. Deleting these descriptors let MAXCCLUS to create rules that are more scientifically informative, and to describe almost the same number of instances faster (only 74% as much time) compared with clustering with all the descriptors. On the other hand, the clustering experiment showed that when deleting more descriptors, including some domain-relevant ones, MAXCCLUS performed poorly, describing many fewer instances and finding fewer clusters than before deleting descriptors.

Also, on the side of textual data, the clustering experiment of chapter six showed that adding more descriptors (synonyms and/or hypernyms of already existing descriptors) to the descriptor sets of the instances did not improve MAXCCLUS clustering. After adding descriptors, MAXCCLUS behaved badly: it was not able to describe more instances than before adding descriptors, it created many alternative rules for clusters characterisations that can confuse the user who is trying to make sense of the clusters, and it took much longer to cluster. This result may be because the added synonyms and/or hypernyms retrieved from WordNet were for the non-domain descriptors only (because WordNet does not contain the domain-relevant words for genetics). It may be that MAXCCLUS would behave better if we could find the frequent synonyms and/or hypernyms for the domain-relevant descriptors. It might also be better to replace descriptors by the most frequent synonyms or hypernyms, rather than simply adding all synonyms or hypernyms, but this is a topic for future work.

The clustering experiment of chapter seven showed that the instances that MAXCCLUS clusters need to have a clearly district categories with a gap between the categories. MAXCCLUS was not able to produce as good results when clustering using a continuous range of categories as when clustering using two categories separated with a clear gap. This behaviour is known for other clustering algorithms too. If one needs to use MAXCCLUS to cluster instances, one must be sure that these instances have clear gaps in between their categories.

The experiment also showed that there is a problem of scaling to larger number of instances with respect to clustering time. As MAXCCLUS attempts to cluster thousands of instances the time increases dramatically.

Note that it is the significance test, rather than the construction of the clusters that dominates the time.

Appendix

The words for the Bible in “Simple English”

This appendix lists the words that were used to delete the descriptors using the first method in chapter 5 (deleting descriptors using the Bible words dictionary). There are 3561 words (about 1000 root words).

The list of words from WordNet (the second method used in chapter 5 for deleting descriptors) is too large to include (44,309 words), and would not be informative.

a	armbands	beautifully	bones	burn	child's
able	armchains	became	book	burned	chin
about	armed	because	booklearning	burning	chins
acacia	armholes	become	books	burnings	chrysolite
acaciatrees	armies	becomes	boot	burst	chrysoprase
account	arming	becoming	bottle	bursting	church
accounts	armrings	bed	bow	bursts	churches
acid	arms	bedcover	bowcords	business	cinnamon
acre	army	bedroom	bowman	but	circle
across	arrow	bedrooms	bowmen	butter	circled
act	arrows	beds	bows	by	circles
acted	arrowsnake	bed's	box	cake	circling
acting	art	bee	boxedoff	cakes	circumcision
acts	arts	been	boxes	calamus	circumcisionnot
actunnatural	as	bees	boxwood	called	clean
adder	ashtree	before	boy	calling	cleaner
addition	ass	behaviour	boys	calm	cleaner's
additions	asses	being	boy's	calmer	cleanhearted
after	ass's	beings	branch	calmly	cleaning
afterbirth	at	beka	branches	came	clear
again	atha	belief	branching	camel	cleared
against	attack	beliefs	brass	camels	clearing
agate	attacked	bell	bread	camel's	clearly
agreement	attacker	bells	breadbasin	camels'	cloth
agreements	attackers	bent	breadbasins	cameltrains	clothed
aha	attacking	berries	breadmaker	captain	clothing
air	attacks	beryl	breadmakers	captains	cloths
all	attempt	beryls	breadmaking	captain's	clothworker's
allburning	attempting	best	breadmeal	carbuncle	clothworking
allwise	attempts	better	breadpaste	carbuncles	cloud
almond	attention	betterlooking	breast	care	cloudburst
almondtree	attraction	between	breastplate	cared	clouded
almost	authorities	bird	breastplates	cares	clouds
aloes	authority	birdlike	breasts	caretakers	coal
altar	awake	birdnet	breath	caring	coals
altars	awakening	birds	breather	carnelian	coat
am	awaking	birth	breathing	carriage	coated
amethyst	away	birthday	breathingspace	carriagehorses	coating
among	axe	birthpains	brickmaking	carriages	coats
amount	axes	birthplace	bricks	carriagetowns	cock
amounts	babies	birthright	brickwork	carriagewheels	cock's
an	baby	births	brickworks	cart	cold
and	back	bit	bride	carts	coldly
angel	backbone	bite	bridebed	cartwheel	colony
angels	backs	bites	bridefeast	cartwheels	colour
angel's	bad	biting	brideoffering	cassia	coloured
anger	badhumoured	bits	brideprice	cattle	colours
angle	badly	bitter	brides	cattlefood	come
angleplate	bag	bitterly	bride's	cattlehouse	comes
angleplates	bagpipe	bittern	bridesong	cause	comfort
angles	bags	bittertasting	bridetent	caused	comforted
anglestone	balancing	bittertongued	bright	causes	comforter
angrily	balancings	black	brighter	causing	comforters
angry	ball	blackberries	brightly	cedar	comforting
animal	balsamtrees	blackberry	broken	cedars	comforts
animals	band	blacker	brokenhearted	cedartrees	coming
animals'	banded	blackest	broomplant	cedarwood	comings
another	banding	blade	brother	certain	common
another's	bands	blades	brotherinlaw	certainly	commonly
answer	bank	blessing	brotherman	chain	company
answered	baptism	blessings	brotherprisoner	chained	comparison
answering	baptisms	blind	brothers	chaining	comparisons
answers	barley	blinding	brother's	chainornaments	competition
ant	base	blood	brothers'	chains	complete
antelope	based	bloodred	brotherservant	chalcedony	completely
ants	bases	blow	brotherworker	chalk	completing
any	basin	blower	brotherworkers	chameleon	condition
anyone	basing	blowing	brow	chance	conditions
anyone's	basins	blows	brows	chances	coney
anything	basket	blue	brush	change	conies
anywhere	baskets	board	brushed	changed	connection
apparatus	bat	boarded	brushing	changers	conscious
apples	bath	boards	brushwood	changes	consciously
appletree	bathed	boat	bucket	changing	control
approval	bathing	boatmen	buckets	cheap	controlled
arch	bathingplace	boats	bud	cheese	controller
arched	baths	bodies	budding	cheeses	controllers
archer	bdellium	body	buds	chest	controlling
archers	be	bodycover	builder	chests	cook
arches	bear	bodycovers	builders	chief	cooked
are	bears	bodyservant	building	chiefs	cooking
argument	beast	boiling	buildingmaterial	child	cookingpot
arguments	beasts	boilingplaces	buildingpaste	childbirth	cooks
ark	beasts'	boilingrooms	buildings	children	copied
arm	beautiful	bone	buildingstone	children's	copies

copper	cutting	different	eagle	evildoings	feet
copperworker	cuttinginstruments	direction	eagles	evilminded	female
copy	cuttingoff	directions	eagle's	evils	females
copying	cypress	dirty	eagles'	evilsmelling	ferret
cor	cypresstrees	disciple	ear	example	fertile
coral	cypresswood	disciples	earlier	examples	fiction
corals	damage	discovery	earliest	exchange	fictions
cord	damaged	discussion	early	exchanged	field
corded	damaging	discussions	earrings	exchanging	fieldfly
cords	dance	disease	ears	existence	fields
cormorant	dancers	diseased	earth	experience	fieldwork
cornelian	dances	diseases	earthshaking	experienced	fieldworkers
cotton	dancing	disgust	earthshock	experiences	fifteen
coughedup	danger	disgusted	earthshocks	expert	fifteenth
countries	dangers	disgusting	earthwork	expertly	fifth
country	darics	distance	earthworks	experts	fifties
countryman	dark	distribution	east	eye	fiftieth
countrymen	darkest	division	ebony	eyeballs	fifty
countryside	darkly	divisions	edge	eyes	fiftyfive
cover	daughter	do	edged	eyesall	fiftyfour
covered	daughterinlaw	doe	edges	eyewitnesses	fiftynine
covering	daughters	doer	edging	face	fiftyone
covers	daughter's	doers	education	facebones	fiftysecond
cow	daughters'	does	effect	faced	fiftyseven
cows	daughtersinlaw	dog	effected	faces	fiftysix
cow's	daughtertowns	dogfly	effecting	facing	fiftythree
crack	dawn	dogs	effects	fact	fiftytwo
cracked	dawning	dog's	egg	facts	fig
cracking	day	doing	eggs	fair	fight
cracks	daylet	doings	eight	fairer	fighter
crane	daylight	done	eighteen	faith	fighters
credit	days	door	eighteenth	falcon	fighting
credited	day's	doorkeeper	eighth	fall	fightingman
creditor	days'	doorkeepers	eightieth	falling	fightingmen
creditors	daytime	dooropening	eighty	false	fightings
criers	dead	doorpillars	eightyeight	falsehearted	fights
cries	dear	doors	eightyfive	falsely	figs
crime	dearer	doorstep	eightyfour	families	figtree
crimes	dearest	doorsteps	eightyseven	family	figtrees
crocodile	dearly	doorway	eightysix	far	finger
cross	death	doorways	eightythree	faraway	fingerrings
crosses	deathblow	doubt	eightytwo	farm	fingers
crossing	deathgiving	doubted	electrum	farmed	fire
crossingplaces	deaths	doubters	elem	farmer	firebaskets
crossroads	deathwound	doubting	eleven	farmers	fireoffering
crown	debt	doubts	eleventh	farming	fireplace
crowned	debtor	dove	emerald	faroff	fires
crowning	debtors	doves	emeralds	farseeing	firespoon
crowns	debts	doves'	empire	farstretching	firetrays
cruel	deceit	down	end	farther	firewood
cruelly	deceits	downfall	ended	farthest	firing
crushed	decision	dragon	ending	farthing	firkins
crusher	decisions	dragons	ends	farthings	first
crushing	deep	dragon's	engines	fat	firstfruit
crushingfloor	deeper	drain	enough	fate	firstfruits
crushingplaces	deepest	drained	envies	father	firtree
crushingstone	deeply	draining	envy	fatherinlaw	firtrees
crushingstones	deepseated	dream	ephah	fathers	fish
cry	degree	dreamer	ephod	father's	fisher
crying	degrees	dreamers	equal	fathers'	fishermen
crystal	delicate	dreaming	equally	fathersand	fishers
cubit	delicately	dreams	error	fathersaw	fishes
cubits	delight	dress	errors	fatter	fishhook
cumi	delighted	dressed	eternal	fattest	fishhooks
cummin	delighting	dresses	even	fear	fishing
cup	delights	dressing	evening	feared	fishinglines
cups	dependent	drink	evenings	fearing	fishingnet
current	design	drinkers	event	fears	fishspears
curse	designed	drinking	events	feast	fitches
cursed	designedly	drinkingplace	ever	feastday	five
curser	designer	drinkingplaces	everburning	feasters	fivesided
curses	designers	drinkingvessels	everflowing	feasting	fixed
cursing	designing	drinkingwater	everliving	feasts	fixedly
curtain	designs	drinks	evershining	feastthey	flag
curtained	desire	driver	every	feathered	flags
curtains	desired	drivers	everybody	feathers	flame
curve	desires	driving	everyman's	feeble	flames
curved	desiring	drop	everyone	feeblefooted	flaming
curving	destruction	dropped	everyone's	feeblehearted	flat
cushion	detail	dropping	everything	feebleminded	flax
cushions	detailed	droppings	everywhere	feebler	flesh
cut	details	drops	evil	feebly	fleshif
cutoff	dew	dry	evildoer	feed	fleshpots
cuts	diamond	drying	evildoers	feeling	flies
cutters	did	dust	evildoing	feelings	flight

flock	fruittree	grainstems	haters	ho	instruments
flocks	fruittrees	grainstores	hates	hole	interest
floor	full	grape	hating	holes	interested
floors	fuller	grapecakes	hats	holies	interests
flow	fullest	grapecrusher	have	hollow	into
flower	fully	grapecrushing	having	hollowing	invention
flowering	further	grapecutting	hawk	hollowminded	inventions
flowers	future	grapes	hawks	hollows	iron
flowing	galbanum	grapevine	hawk's	holy	irons
fly	garden	grass	he	homer	ironworker
fold	gardener	grasses	head	homers	is
folded	gardens	grassland	headband	honey	island
folding	gave	grasslands	headbands	honeyed	islands
folds	gazelle	great	headcovers	honour	it
food	gazelles	greater	headdress	honoured	its
foodbags	general	greatest	headdresses	honouring	itself
foodplace	generally	greatly	heading	honours	ivory
foods	generation	green	heads	hook	jacinth
foodstore	generations	grey	headstone	hooks	jackals
foolish	gentle	greyhaired	headway	hoopoe	jasper
foolishly	gentlehearted	greyheaded	healthy	hope	javelin
foot	gently	grief	hearer	hoped	jewel
footchains	gerahs	grip	hearers	hopes	jewelled
footmen	get	gripped	hearing	hoping	jeweller
footrest	gets	gripping	heart	horn	jewels
footrings	getting	group	hearts	horned	join
footstep	gettingin	grouped	heart's	hornet	joined
footsteps	giereagle	groups	heartsearchings	hornets	joining
footway	girl	growth	heat	horns	joins
for	girls	growths	heated	horse	journey
force	girl's	guest	heating	horseback	journeying
forced	give	guestroom	heatingpot	horseman	journeys
forces	given	guests	heaven	horsemen	joy
forcing	giver	guest's	heavens	horses	judge
forgiveness	gives	guide	hedgehog	horse's	judged
forgivenessbut	giving	guided	hegoat	horses'	judges
fork	glad	guides	hegoats	hour	judge's
forks	gladhearted	guiding	helamb	hours	judging
form	gladly	guidingblade	helambs	hour's	jumping
formed	glass	guidingblades	hell	house	keep
forming	glories	had	help	housed	keeper
forms	glory	hair	helped	houses	keepers
fortieth	glorying	haircloth	helper	houseservant	keeping
forty	glorysuch	haircutter's	helpers	houseservants	keeps
fortyeight	go	hairs	helping	housetop	keepwashings
fortyfirst	goat	half	her	housetops	kept
fortyfive	goats	halfcurtain	herd	housing	key
fortyfour	goat's	halfheartedly	herdman	how	keys
fortynine	goats'	halfshekel	herdmen	however	keystone
fortyone	goatskins	halftribe	herds	hundred	kidneys
fortyseven	gobetween	halfway	herdsman's	hundreds	kind
fortysix	god	hammer	herdsmen	hundredth	kinder
fortythree	goddess	hammered	here	husband	kindhearted
fortytwo	godfearing	hammering	heritage	husbands	kindly
forward	gods	hammers	heritages	husband's	king
fountain	gods'	hand	heron	hyssop	kingdom
fountains	goes	handbags	hers	ice	kingdoms
four	going	handed	herself	icedrops	kinglike
fourfooted	goings	handing	hesheep	icestorm	kings
fourteen	gold	handpart	high	idea	king's
fourteenth	goldworker	hands	higher	ideas	kings'
fourth	goldworkers	hand's	highest	if	kiss
fowlhouse	gone	handstretch	highhearted	ill	kisses
fowls	good	handwork	highlands	image	kissing
fox	goodday	handworker	highly	images	kite
foxes	goodfornothing	handwriting	highsounding	important	knee
frame	goodlooking	hanging	highway	impossible	knees
framed	goods	hangings	highways	impulse	knife
frames	gopher	happy	hill	impulses	knives
framework	got	harbour	hillcountry	in	knotted
framing	government	hard	hills	increase	knotting
frankincense	grace	harder	hillside	increased	know
free	grain	hardest	hillsides	increasing	knowledge
freeing	graincleaning	hardfaced	hilltops	industry	lama
freely	graincrushers	hardhearted	him	ink	lamb
frequent	graincrushing	hardly	himself	inkpot	lambs
frequently	graincutter	hare	himthat	inlet	land
friend	graincutters	harmony	hin	inlets	landand
friends	graincutting	harp	hind	inmost	landmark
frogs	grainfield	hart	his	inner	landmarks
from	grainfields	harts	hiss	insect	lands
fromfrom	grainfloor	has	hisses	insects	landthough
front	grainfloors	hate	hissing	inside	language
fruit	grainplants	hated	histories	insides	languages
fruits	grains	hater	history	instrument	last

late	locust	mealtime	mulberrytree	noblest	or
later	locusts	measure	mule	nobly	order
laughed	locust's	measured	mules	nobody	ordered
laughing	log	measurers	muscle	nohair	ordering
law	long	measures	muscles	noise	orders
lawgiver	longer	measuring	music	noises	ornament
lawgivers	longhaired	measuringline	musicinstruments	nor	ornamented
lawgivers'	look	measuringrod	musicmaker	normal	ornamenting
laws	looked	meat	musicmakers	normally	ornaments
lead	looking	meathook	musicpipe	north	ospray
leaf	lookingglass	meathooks	mustard	northeast	ostrich
learner	lookingglasses	meats	my	nose	ostriches
learning	lookout	medical	myrrh	nosejewels	other
least	looks	meeting	myrtle	nosering	otherperson
leather	loose	meetingplace	myself	noserings	others
leatherworker	lord	meetingplaces	myselfwith	noses	other's
leaven	lords	meetings	nail	not	others'
leavened	lord's	melodies	nailed	note	our
leaves	loss	melody	nailing	noted	ours
left	losses	memories	nails	notes	ourselves
lefthanded	loud	memory	name	nothing	out
leg	louder	men	namebecause	noting	outburst
legchains	loudly	men's	named	now	outbursts
legmuscles	loudsounding	menservants	names	nowhere	outcome
legs	loudtongued	menthey	naming	number	outcries
leopard	loudvoiced	mercies	narrow	numbered	outcry
leopards	love	mercy	narrower	numbering	outer
leper	loved	mercyseat	narrowminded	numbers	outflowing
lepers	lovefeasts	metal	nation	nuts	outgoing
leper's	lovefruits	metaltester's	nations	oak	outgoings
less	lover	metalworker	natural	oaks	outlaw
let	lovers	metalworkers	naturally	oaktree	outlaws
lets	loves	mice	near	oaktrees	outlet
letter	love's	middle	nearer	oath	outlines
lettered	loving	might	nearest	oaths	outshining
letters	low	mile	necessaries	observation	outside
letting	lower	miles	necessary	octave	outskirts
level	lowest	military	neck	of	outstretched
levelled	lowland	milk	neckornaments	off	oven
life	lowlands	milkcheeses	necks	offand	ovenfire
lifeblood	lowly	million	need	offer	ovens
lifegiver	machine	mind	needed	offered	over
lifegiving	machines	minds	needing	offering	overcame
lifetime	made	mine	needle	offerings	overcome
lifted	make	minute	needle's	offspring	overcomes
lifter	maker	minute's	needlework	oil	overcoming
lifting	makers	mist	needs	oiled	overdriving
light	makes	mists	neighbour	oilgiving	overflow
lighted	making	mixed	neighbouring	oils	overflowing
lighting	male	money	neighbours	old	overfull
lights	males	moneybag	neighbour's	older	overgreat
lightsupport	man	moneybags	neighbours'	oldest	overhanging
lightsupports	manager	moneybox	net	olive	overhard
like	managers	moneychangers	nets	olivegardens	overhead
lily	manchild	monkeys	network	oliveleaf	overhigh
limit	maneh	month	never	olives	overinterested
limits	manna	months	neverending	olivetree	overlong
line	man's	moon	new	olivetrees	overlooked
lined	manservant	moonornaments	newly	olivewood	overlooking
linen	many	moons	newlymarried	omer	overmuch
linenwork	mark	more	news	omers	overpowering
linenworker's	marked	morning	night	on	overquick
lines	market	mornings	nightbird	once	overready
lion	marketing	most	nightfall	one	overruling
lions	marketplace	mother	nighthawk	ones	overrunning
lion's	marketplaces	motherinlaw	nightlet	one's	oversee
lions'	marking	mothers	nights	oneself	overseeing
lips	markings	mother's	night's	onlookers	overseer
liquid	marks	mothers'	nightspirit	only	overseers
liquids	married	mothertown	nighttime	onrush	overstepping
list	mass	motion	nightvision	onto	overtake
listed	massed	mountain	nightwatches	onycha	overtaken
lists	masses	mountains	nine	onyx	overtakes
little	massing	mountaintop	nineteen	open	overtook
liver	master	mountaintops	nineteenth	opener	overturned
lives	masterbuilder	mouse	ninety	openhanded	overturning
living	masters	mouth	ninetyeight	opening	overweight
livingplace	master's	mouthbit	ninetyfive	openly	owl
livingplaces	masters'	mouthbone	ninetynine	openminded	owner
livingspace	material	mouths	ninetysix	openwork	owners
lizard	materials	move	ninetytwo	operation	owning
load	may	moved	ninth	operations	ox
locked	me	mover	no	opinion	oxcords
locking	meal	moving	noble	opinions	oxdriving
locks	meals	much	noblehearted	opposite	oxen

oxstick	pitied	praising	rained	rocks	scribe
oxyokes	pity	prayer	raining	rod	scribes
pain	place	prayers	rains	rods	scribe's
pained	placed	preacher	rainstorm	roe	sea
pains	places	preachers	range	roes	seabeast
paint	placing	preaching	rapture	roe's	seabeasts
painted	plane	present	rate	roes'	seaforce
painting	planetrees	price	rating	roll	seagrass
palm	plant	priced	raven	rolled	seahawk
palmtree	planted	pride	ravens	rolling	sealand
palmtrees	planter	priest	rays	rolls	sealands
paper	planters	priests	reader	roof	seamen
papers	planting	priest's	readers	roofed	search
papyrus	plantingplaces	priests'	readily	roofing	searched
parallel	plantings	prince	reading	roofs	searcher
parcel	plants	princes	ready	room	searchers
parcels	plantworm	prince's	reason	rooms	searching
park	plate	princess	reasoning	root	searchings
part	plated	print	reasonings	rooted	seas
parted	plates	printed	reasons	rooting	sea's
parting	plating	prison	record	roots	seaside
partridge	play	prisoned	recorded	rose	seat
parts	played	prisoner	recorder	rough	seated
passion	player	prisoners	recorders	rougher	seating
passions	players	prisonhouse	records	roughly	seats
past	playing	prisoning	red	round	second
paste	plaything	prisons	redder	roundabout	secret
pasting	please	private	redhaired	rubbed	secretary
payment	pleased	privately	reedboats	rubbing	secretly
payments	pleasers	produce	regret	rubies	secrets
peace	pleasing	produced	regretted	ruby	see
peacegiving	pleasure	producing	regular	rule	seed
peaceloving	pleasuremaking	profit	regularly	ruled	seeds
peacemaker	pleasures	profits	rehokim	ruler	seed's
peacemakers	plough	property	relation	rulers	seeif
peaceoffering	ploughblades	prophet	relations	ruler's	seeing
peaceofferings	ploughed	prophets	relation's	rulers'	seem
peacocks	ploughing	prophet's	religion	rules	seemed
pearl	ploughman	protest	representative	ruling	seeming
pearls	ploughmen	protesting	representatives	run	seems
pelican	ploughs	protests	request	runner	seen
pen	pockets	psaltery	requested	runners	seer
pence	point	public	requesting	running	seers
pencil	pointed	publicly	requests	rush	sees
penknife	pointing	pulled	respect	rushing	selection
pennies	points	pulling	respected	sabachthani	self
penny	poison	punishment	respecter	sad	selfcontrol
people	poisoned	punishments	respecting	sadfaced	selfcontrolled
peopled	poisoning	purple	responsible	sadly	selfglory
peoples	poisonplant	purpose	rest	safe	selfjudged
people's	poisonsnake	purposed	resting	safely	selfordered
peoples'	poisonsnakes	purposes	restingplace	said	selfrespect
peres	police	purposing	restingplaces	sail	selfrespecting
perfume	polished	purslain	revelation	sailing	send
perfumeboxes	polisher	pushed	revelations	sailors	sending
perfumed	pomegranate	pushing	reward	sails	sends
perfumemaker	pomegranates	put	rewarded	sailsupport	sense
perfumemakers	pool	puts	rewarder	saint	senses
perfumer's	pools	putting	rewarding	saints	sent
perfumes	poor	pygarg	rewards	salt	separate
perfumevessel	poorer	qualities	right	salted	separately
person	poorest	quality	righteousness	salvation	separating
persons	poorlooking	quarter	rightly	same	serious
person's	poorly	quarters	rights	sand	seriously
phylacteries	porcupine	queen	ring	sandalwood	seriousminded
picture	porter	queenmother	ringed	sandlizard	servant
pictured	position	queens	rings	sands	servantgirl
pictures	positions	queen's	river	sapphire	servantgirls
pig	possible	question	riverbeast	sapphires	servants
pigeon	postrunner	questioned	rivercrossing	sardius	servant's
pigeons	pot	questioning	rivergrass	sardonyx	servants'
pigs	pots	questionings	riverplant	saviour	servantwife
pig's	potter	questions	riverplants	saviours	servantwives
pigs'	potters	quick	rivers	saw	servantwoman
pillar	potter's	quicker	river's	sawbecause	servantwomen
pillared	pound	quickfooted	riverside	say	seven
pillars	pounds	quickly	road	saying	seventeen
pin	powder	quickmoving	roads	sayings	seventeenth
pinetree	power	quickrunning	roadside	says	seventh
pinning	powers	quiet	robe	scale	seventy
pinpoints	praise	quietly	robed	scales	seventyfive
pins	praised	quite	robes	school	seventyfour
pipe	praiseoffering	railing	rock	scissors	seventyseven
pipes	praiseofferings	rain	rockbadger	scorpion	seventysix
piping	praises	raindrops	rockgoats	scorpions	seventythree

seventytwo	sisters	sorrows	stoning	takers	thirtyfour
sex	sister's	sort	stop	takes	thirtynine
shade	six	sorts	stopped	taking	thirtyninth
shades	sixteen	soul	stopping	talent	thirtyone
shake	sixteenth	souls	store	talents	thirtysecond
shaker	sixth	soul's	stored	talents'	thirtyseven
shaking	sixty	sound	storedup	talk	thirtyseventh
shakiphwood	sixtyeight	sounded	storehouse	talked	thirtysix
shame	sixtyfive	sounding	storehouses	talker	thirtysixth
shamed	sixtyfour	sounds	storerooms	talkers	thirtythree
shaming	sixtynine	soup	stores	talking	thirtytwo
sharp	sixtyone	south	storetowns	tall	this
sharper	sixtyseven	southeast	stories	taller	thistles
sharply	sixtysix	southland	storing	taste	thorn
sharppointed	sixtytwo	space	stork	tasted	thorns
she	size	spaced	storm	tastes	thorntree
sheasses	sizes	spaces	stormcloud	tasting	those
shebears	skies	spade	stormcrushed	tax	though
sheep	skin	spades	stormflames	taxed	thought
sheepfarmer	skindisease	sparrow	stormwind	taxes	thoughts
sheepkeeper	skinmark	sparrows	stormwinds	taxfarmer	thousand
sheepkeepers	skinned	spear	story	taxfarmers	thousands
sheepmarket	skinning	spearmen	straight	taxing	thread
sheep's	skinplates	spears	straightforward	teacher	threads
sheepskins	skins	special	straightforwardly	teachers	three
sheeptraders	skirt	specially	strange	teaching	threeyear
shegoat	skirts	spelt	strangely	teachings	throat
shegoats	sky	spice	stream	tears	throats
shekel	sleep	spices	streaming	teeth	through
shekels	sleeping	spicetrees	streams	tekel	thumb
shekels'	sleepingrooms	spider's	street	ten	thumbs
shelamb	slip	spirit	streets	tencorded	thunder
shelf	slipping	spirits	strength	tens	thunderflame
shelion	slope	sponge	stretch	tent	thunderflames
shelions	slopes	spoon	stretched	tentcircle	thundering
sherbintree	sloping	spoons	stretchedout	tentcircles	thunderings
shining	slow	sport	stretcher	tentcord	thunders
ship	slowly	sports	stretching	tentdoor	thunderstorm
shipmaster	small	spring	strong	tenth	thunderstorms
ships	smaller	springing	stronger	tenths	tight
ship's	smallest	springs	strongest	tentmakers	tightly
ships'	smashed	square	strongly	tentpin	till
shock	smashing	squared	structure	tentpins	time
shocked	smell	squares	substance	tents	times
shocking	smelling	stacte	such	tenttowns	tin
shoe	smells	stage	sudden	test	tired
shoes	smile	stages	suddenly	testament	to
short	smoke	stamp	suggestion	tested	today
shorter	smoking	stamped	suggestions	tester	toe
shower	smooth	stamping	summer	testing	toes
showered	smoother	stamps	summerhouse	tests	together
showers	smoothing	star	sun	than	tomorrow
shut	smoothly	stars	sundown	that	tongue
shutin	smoothsounding	start	sunimages	the	tongues
shutting	snake	started	sunjewels	theatre	tonight
side	snakebite	starting	sunlight	their	took
sidebones	snakes	statement	support	theirs	tooth
sideroads	snake's	statements	supported	them	top
siderooms	sneezings	station	supporter	themsaid	topaz
sides	snow	stationed	supporters	themselves	tops
sidewalls	so	stations	supporting	themthe	touch
siding	soap	stem	supports	then	touched
sign	society	stems	surprise	theories	touching
signed	soda	step	surprised	there	tower
signing	soft	stepping	swallow	these	towers
signs	softer	steps	sweet	they	town
silk	softly	stick	sweeter	thick	towndoor
silver	softlyflowing	sticks	sweetsmelling	thicker	towns
silverworker	solid	sticky	swimming	thickly	townsman
simple	some	stiff	sword	thief	townsmen
simpleminded	someone	stiffhearted	swords	thieves	townspeople
simpler	someone's	stiffnecked	swordsmen	thin	trade
simply	something	still	sycamores	thing	traded
sin	sometimes	stitched	sycamoretrees	things	trader
sinned	somewhere	stitchedup	system	thinner	traders
sinner	son	stitching	table	third	trading
sinners	song	stomach	tableland	thirteen	tradingships
sinner's	songs	stomachs	tablelands	thirteenth	train
sinning	soninlaw	stone	tables	thirtieth	trained
sinoffering	sons	stonecutters	tablevessels	thirty	training
sinofferings	son's	stoned	tail	thirtyeight	transport
sins	sons'	stones	tails	thirtyeighth	transported
sir	sonsinlaw	stonework	take	thirtyfifth	traveller
sister	sorrow	stoneworkers	taken	thirtyfirst	travellers
sisterinlaw	sorrowing	stoneworks	taker	thirtyfive	travelling

travels	undergoing	uses	waster	which	woodlands
tray	undergone	using	wasters	whichever	woods
trays	undertake	valley	wastes	while	woodwork
tree	undertaken	valleys	wasting	whip	woodworker
trees	undertaking	value	watch	whipped	woodworkers
tribe	undertakings	valued	watched	whipping	woodworker's
tribes	undertook	valuer	watcher	whips	wool
tribesmen	underwent	values	watchers	whistling	woolcutters
trick	underworld	valuing	watches	white	woolcutting
tricked	undid	veil	watching	whitehaired	word
tricking	undo	veiled	watchings	whiter	words
tricks	undoes	veiling	watchman	whitewash	work
trigon	undone	veils	watchman's	whitewashed	worked
trouble	unending	verse	watchmen	who	worker
troubled	unequal	very	watchmen's	whoever	workers
troublemaker	unfertile	vessel	watchnight	whom	working
troubler	unfolded	vessels	watchtower	whose	workman
troublers	unfolding	view	water	why	workman's
troubles	unformed	viewing	waterdoor	wide	workmen
troubling	unfree	vine	waterdoorway	widely	works
trousers	unhappy	vinebranch	watered	wider	workstrange
true	unhealthy	vinecuttings	waterer	wideshining	workthen
truehearted	unholy	vinegarden	waterfalls	widestretching	world
truly	unhonoured	vinegardens	waterhen	widow	worldrulers
turn	unimportant	vinekeepers	waterhole	widowed	world's
turned	unit	vineknives	waterholes	widows	worm
turning	united	vineplants	watering	widow's	worms
turnings	uniting	vines	wateringplaces	wife	wormwood
twelfth	unkind	vinetree	waterpipe	wife's	worse
twelve	unleavened	violent	waterplants	will	worship
twentieth	unlifted	violently	waterpot	willowtree	worshipped
twenty	unlike	virgin	waters	wind	worshipper
twentyeight	unlimited	virgins	waterside	windinstruments	worshippers
twentyfifth	unmarked	virgin's	waterskin	window	worshipping
twentyfirst	unmarried	virtue	waterskins	windowframes	worst
twentyfive	unmeasured	virtues	waterspring	windows	would
twentyfour	unmixed	vision	watersprings	winds	wound
twentyfourth	unmoved	visions	waterstreams	wine	wounded
twentynine	unnatural	voice	watertight	winecrusher	wounding
twentyone	unnaturally	voices	watertown	winecrushing	wounds
twentysecond	unnumbered	vulture	watervessel	winedrinking	wrath
twentyseven	unpeopled	waited	waterways	wines	writer
twentyseventh	unplanted	waiting	wave	wineservant	writers
twentysix	unpleasing	waitingwoman	waved	wineservants	writer's
twentysixth	unploughed	walk	waves	wineskin	writing
twentythird	unready	walked	waving	wineskins	writingboard
twentythree	unresting	walking	wax	winestore	writings
twentytwo	unsafe	walks	way	wing	wrong
twice	unseen	wall	ways	winged	wrongdoer
twist	unsexed	wallbuilders	wayside	wings	wrongdoers
twisted	unshaking	walled	we	winter	wrongdoing
twisting	untested	walls	wealth	wires	wrongdoings
twists	untouched	wanderer	weariness	wisdom	wronghearted
two	untrained	wanderers	weasel	wise	wrongly
twoedged	untroubled	wandering	weather	wisehearted	wrongminded
twohorned	untrue	wanderings	week	wisely	wrongs
twos	unused	war	weeks	wiser	yardsticks
unable	unveiled	warcarriage	weeping	wisest	year
unanswered	unveiler	warcarriages	weight	with	years
unarmed	unveiling	warcries	weighted	without	year's
unbalanced	unwalled	warcry	weights	witness	years'
unbreathing	unwashed	ward	well	witnessed	yellow
unbroken	unwatched	wardress	wellarmed	witnesses	yes
uncared	unwatered	wardresses	wellbeing	witnessing	yesterday
uncaredfor	unwell	warhorn	welldoing	wives	yoke
uncertain	unwise	warhorse	welldressed	wolf	yoked
uncertainly	unworked	warhorses	wellloved	wolves	yokes
unchanged	up	warm	wellpleased	woman	yoking
unchanging	upkeep	warming	wellpleasing	woman's	you
unclean	upland	warmly	wellsaid	womanservant	young
unclothed	uplifted	waronly	wellwatered	women	younger
uncomforted	uplifting	warring	went	women's	youngest
uncommonly	upon	wars	were	womenservants	your
unconscious	upper	was	west	wonder	yours
unconsciously	upright	wash	wet	wondered	yourself
uncontrolled	uprightly	washed	what	wondering	yourselves
uncooked	uprights	washerman's	whatever	wonders	
uncovered	uprooted	washing	wheat	wonderworker	
uncovering	uprooting	washingbasin	wheel	wonderworkers	
uncut	upside	washings	wheeled	wonderworking	
undamaged	us	washingvessel	wheels	wood	
under	use	washingvessels	when	woodcutter	
underfoot	used	washpot	whenever	woodcutters	
undergo	user	waste	where	woodcutting	
undergoes	users	wasted	wherever	woodland	

Bibliography

Andreae, P; Shavlik, J; and Molla M. 2001
Characterising Genes From Gene-array Data.
A presentation talk.
http://www.mcs.vuw.ac.nz/~pondy/projects/maxcclus-talk.html

Bassett, D; Eisen, M; and Boguski, M. 1999
Gene expression informatics - it's all in your mine
nature genetics supplement, volume 21, january 1999

Bible in Basic English
http://www.biblestudytools.net/Information/BibleinBasicEnglish.html

Blaschke, C; Oliveros, J; and Valencia, A. 2001
Mining functional information associated with expression arrays
Funct Integr Genomics (2001) 1:256-268

Boeckmann, B; Bairoch, A; Apweiler, R; Blatter, M.C; Estreicher, A; Gasteiger, E; Martin, M.J; Michoud, K; O'Donovan, C; Phan, I; Pilbout, S; and Schneider, M. 2003
The Swiss-Prot protein knowledgebase and its supplement TrEMBL in 2003.
Nucleic Acids Res. 31:365-370(2003).

Carr, D; Somogyi, R; and Michaels, G. 1999
Templates for Looking at Gene Expression Clustering
Statistical Computing & Statistical Graphics Newsletter April 97, 20-29

Chen, Y; Bittner, M; and Dougherty, E. 1999
Issues associated with microarray data analysis and integration
Nature Genet. 22, 213-215; 1999

Claverie J-M. 1999
Computational Methods for the identification of
differential and coordinated gene expression
Human Molecular Genetics, 1999, Vol 8, No. 10, 1821-1832

D'haeseleer, P; Liang, S; and Somogyi, R. 1999
Gene Expression Data Analysis and Modeling
Tutorial
Session on Gene Expression and Genetic Networks
Pacific Symposium on Biocomputing, 1999
Hawaii , January 4-9, 1999

D'haeseleer, P; Liang, S; and Somogyi, R. 2000
Genetic network inference:
from co-expression clustering to reverse engineering
BIOINFORMATICS, Pages 707-726, Vol. 16, no. 8, 2000

Falkenauer, E. and Marchand, A. 2001
Using k-Means? Consider ArrayMiner.
In Proceedings of the 2001 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'2001), Las Vegas, Nevada, USA, June 25-28, 2001.

Fasulo, D. 1999
An Analysis of Recent Work on Clustering Algorithms.
Technical Report # 01-03-02
Department of Computer Science & Engineering, University of Washington, Seattle, USA.

Komili, S. 2002
Clustering
Genomics & Computational Biology, Section 6: October 29, 2002

Molla, M; Andreae, P; Glasner, J; Blattner, F. and Shavlik, J. 2002.
Interpreting Microarray Expression Data Using Text Annotating the Genes.
Information Scienses, 146, pp. 75-88.

Molla, M; Waddell, M; Page, D. and Shavlik, J. 2004.
Using Machine Learning to Design and Interpret Gene-Expression Microarrays.
AI Magazine, 25, pp. 23-44.

Pellegrini, M. 2001
Computational methods for protein function analysis
Curr Opin Chem Biol. 2001 Feb;5(1):46-50.

Porter, M. F. 1980.
An algorithm for suffix stripping.
Program, Vol. 14, no. 3, pp 130-137, July 1980.

Princeton University, 2003
WordNet 2.0
Copyright © 2003. All rights reserved.
http://www.cogsci.princeton.edu/~wn/

Rennie,J. 2002
WordNet::QueryData : a Perl module for accessing the WordNet database.
Copyright © 2000, 2001, 2002. All rights reserved.
http://www.ai.mit.edu/people/jrennie/WordNet

Sahami, M; Dumais, S; Heckerman, D; and Horvitz, E. 1998.
A Bayesian Approach to Filtering Junk E-mail.
AAAI'98 Workshop on Learning for Text Categorization, July 27, 1998, Madison, Wisconsin.

Smet, F; Mathys, J; Marchal, K; Thijs, G; Moor, B; and Moreau, Y. 2002
Adaptive quality-based clustering of gene expression profiles.
BIOINFORMATICS Vol. 18 no. 5 2002 Pages 735-746

Swiss-Prot
http://au.expasy.org/sprot/

Swiss-Prot User Manual
http://au.expasy.org/sprot/userman.html

Szabo, A; Boucher, K; Carroll, W; Klebanov, L; Tsodikov, A; and Yakovlev, A.
Variable Selection and Pattern Recognition with Gene Expression Data Generated by the Microarray Technology
MATHEMATICAL BIOSCIENCES, 2002, 176, pp. 71-98

Tsodikov, A; Szabo, A; and Jones, D. 2002
Adjustments and measures of differential expression for microarray data.
BIOINFORMATICS Vol. 18 no. 2 2002 Pages 251-260

Turcato, D; Popowich, F; Toole, J; Fass, D; Nicholson, D; and Tisher, G.2000
Adapting a synonym database to specific domains.
In Proceedings of the ACL'2000.

Vilo, J; Brazma, A; Jonassen, I; Robinson, A; and Ukkonen, E. 2000
Mining for putative regulatory elements in the yeast genome using gene expression data
ISMB-2000, August 2000, pages 384-394

Waddell, P. J. and Kishino, H. 2000
Cluster Inference Methods and Graphical Models Evaluated on NCI60 Microarray Gene Expression Data
Genome Informatics 11: 129-140 (2000)

Williamson, D. 2002
Box and Whisker Diagrams: getting Microsoft Excel to plot them for you.
http://www.duncanwil.co.uk/boxplot.html
Copyright © 17 October 2002

Zhao, Y. and Karypis, G.
Clustering in Life Sciences
Technical Report
Department of Computer Science, University of Minnesota,
Minneapolis, MN 55455