

## Clustering Rules: A Comparison of Partitioning and Hierarchical Clustering Algorithms

A. P. REYNOLDS<sup>★</sup>, G. RICHARDS, B. DE LA IGLESIA  
and V. J. RAYWARD-SMITH

*School of Computing Sciences, University of East Anglia, Norwich, UK.*  
*e-mail: ar@cmp.uea.ac.uk*

**Abstract.** Previous research has resulted in a number of different algorithms for rule discovery. Two approaches discussed here, the ‘all-rules’ algorithm and multi-objective metaheuristics, both result in the production of a large number of partial classification rules, or ‘nuggets’, for describing different subsets of the records in the class of interest. This paper describes the application of a number of different clustering algorithms to these rules, in order to identify similar rules and to better understand the data.

**Mathematics Subject Classifications (2000):** 62H30, 68T05, 68T37.

**Key words:** clustering, partial classification, rule induction.

### 1. Introduction

Partial classification, also known as ‘nugget’ discovery, involves the production of accurate yet simple rules that describe subsets of the records of interest within a database. For example, in a motor insurance database, the class of interest may contain those motorists who have claimed on their insurance. Often, rules produced are conjunctions of simple clauses, where each clause applies to one field within the database (e.g. [2]). For example,

if age ≤ 25 and car\_type = sports then claim = yes

is such a rule.

Although this approach produces useful rules, these rules may describe only small subsets of the records of interest. To describe larger subsets accurately, collections of such rules are required. De la Iglesia et al. [5, 6] have researched the application of multi-objective genetic algorithms to data mining, using confidence and coverage as two separate objectives. This approach leads to the

---

<sup>★</sup> Corresponding author.

production of a collection of Pareto-optimal rules. Richards et al. [17] have created software that produces *all* the rules that apply to the data, subject to constraints on, amongst other things, the rule confidence and coverage.

Both of these approaches can produce hundreds or even thousands of rules. For example, the all-rules algorithm was applied to the ‘adult’ database [3] and instructed to seek rules that met a number of constraints. These included the constraints that the rules should have greater than 15% coverage and greater than 60% confidence. The result was the production of 4,785 rules. Examining these rules by eye immediately reveals that many are very similar. Others, while appearing to be different, may in fact match similar sets of records.

In order to determine which rules match similar sets of records and to improve the presentation of results obtained from such algorithms, a number of different clustering algorithms have been applied to the results obtained by both the all-rules algorithm and from a multi-objective genetic algorithm. This paper extends the work started by Reynolds et al. [16].

Sections 2 and 3 introduce terms used in later sections. Section 2 describes what is meant by a ‘simple’ rule, whereas Section 3 describes the measures used to determine the quality of a rule.

Sections 4 and 5 describe the algorithms used to produce the sets of rules to be clustered and describe the types of rule produced by each. These algorithms were applied to the ‘adult’ database from the UCI repository [3]. Records with unknown attributes were not considered, leaving 30,162 records of adult census data from the 1994 US Current Population Survey for training. The final weight field was removed, leaving five numeric fields and nine categorical fields, including the ‘salary’ field. Rules were produced that predicted salaries of over 50,000 dollars.

Having described the types of rules produced and the algorithms that produce them, Sections 6 and 7 provides the groundwork necessary before clustering can be applied to the rules. Section 6 presents a number of different ways to define rule dissimilarity. The resulting dissimilarity matrices are required, as input, by the clustering algorithms described in later sections. Section 7 gives a description of the use of silhouettes to measure the quality of the clusterings obtained. This allows us to perform some comparison of the algorithms.

The clustering algorithms are described in Sections 8 and 9. Section 8 presents four clustering algorithms, *k*-means [14], *k*-medoids, PAM [13] and CLARANS [15], that partition a set of objects into *k* clusters. *K*-means is shown to be unsuitable for the problem of rule clustering, but results for the other three techniques are presented and compared. This is followed in Section 9 by a description of the application of hierarchical clustering techniques to the rule sets. Algorithm design choices are discussed and the results produced are presented.

Finally Section 10 presents some conclusions and presents a number of ideas for further research.

## 2. Attribute Tests and Simple Rules

Before describing the algorithms used to produce the ‘simple’ rules to be clustered, it is first necessary to describe what is meant by a ‘simple’ rule. The two algorithms used produce slightly different types of rules. However, in this section we will describe the features shared by all the rules produced by both. Differences between the types of rule produced by the two algorithms are described in subsequent sections.

Both of the algorithms to be described produce partial classification rules of the form

$$\text{antecedent} \rightarrow \text{consequent},$$

where both the antecedent and the consequent are constructed from *attribute tests*.

Let  $Q$  be a finite set of attributes, which in practice corresponds with the fields in the database. Each  $q \in Q$  has an associated domain,  $Dom(q)$ . An attribute test (AT),  $b$ , consists of an attribute,  $at(b) \in Q$ , and a value set,  $Val(b) \subseteq Dom(at(b))$ , and may be written  $at(b) \in Val(b)$ . A record satisfies this test if its value for attribute  $at(b)$  belongs in the set  $Val(b)$ .

Each algorithm may allow only certain types of value sets  $Val(b)$ . In particular, both algorithms require that attribute tests on a numerical field have a value set of the form  $Val(b) = \{x \in at(b) : x \leq v(b)\}$  or  $Val(b) = \{x \in at(b) : x \geq v(b)\}$ , where  $v(b) \in Dom(at(b))$ . In this case, the attribute test may be written  $at(b) \leq v(b)$  or  $at(b) \geq v(b)$ , respectively. In the example in the introduction,  $age \leq 25$  is an attribute test of this form.

The constraints on the attribute tests for a nominal field differ between the two algorithms, being more restrictive in the case of the all-rules algorithm. These constraints are described in Sections 4 and 5.

Both of the algorithms produce a set of rules, each of which have the same consequent consisting of just one AT. This defines the class of interest targeted by the algorithms, for example, the class of motorists who have claimed on their motor insurance. The antecedent of a rule produced by one of these algorithms is a conjunction of ATs, none of which may share the same attribute as the consequent.

## 3. Confidence, Coverage and Improvement

Let  $D$  be the database from which rules are produced. Let rule  $\rho$  be of the form  $\alpha \rightarrow \beta$ . Define the sets  $L(\rho)$ ,  $R(\rho)$  and  $S(\rho)$  as follows:

$$\begin{aligned} L(\rho) &= \{r \in D : \alpha(r)\}, \\ R(\rho) &= \{r \in D : \beta(r)\}, \\ S(\rho) &= L(\rho) \cap R(\rho). \end{aligned}$$

Here,  $L(\rho)$  is the set of records that match the left hand side of rule  $\rho$ ,  $R(\rho)$  is the set that match the right hand side and  $S(\rho)$  is the set that match both.  $S(\rho)$  is the set of records that *support* rule  $\rho$ .

A rule  $\rho'$  is a sub-rule of  $\rho$  if it shares the same consequent and if the attribute tests in the antecedent of  $\rho'$  form a proper subset of the set of antecedent attribute tests of  $\rho$ .

We can define the confidence, coverage and improvement of a rule  $\rho$  as follows:

- The *confidence* of rule  $\rho$  is defined as  $conf(\rho) = |S(\rho)| / |L(\rho)|$ ,
- The *coverage* of rule  $\rho$  is defined as  $cov(\rho) = |S(\rho)| / |R(\rho)|$ ,
- The *improvement* of rule  $\rho$  is defined as  $imp(\rho) = conf(\rho) - conf(\rho')$ , where  $\rho'$  is the sub-rule of  $\rho$  with greatest confidence.

#### 4. The All-rules Algorithm

The all-rules algorithm of Richards et al. [17] attempts to produce all rules subject to a set of constraints.

- Rules must be of the form described in Section 2.
- Attribute tests for nominal fields must be of the form  $at(b) = v(b)$ , where  $v(b) \in Dom(at(b))$ .
- The confidence, coverage and improvement of a rule must exceed values provided by the user.
- A limit is set on the number of attribute tests that may appear in the antecedent of a rule.

When the database includes numerical fields, the number of rules that may be produced can become very large. Setting the threshold for rule confidence to 60%, coverage to 15% and improvement to 3%, and restricting the number of attribute tests in the antecedent to five or fewer results in the production of 4,785 rules from the adult database. Applying the algorithm to larger databases can also result in excessive time requirements. Therefore, a different version of the algorithm adds the constraint that the rules produced must be *pc-optimal* (population-confidence optimal) [1].

A partial order  $\leq_{pc}$  is defined on rules where  $\rho_1 \leq_{pc} \rho_2$  if and only if

$$S(\rho_1) \subseteq S(\rho_2) \text{ and } conf(\rho_1) < conf(\rho_2), \text{ or} \\ S(\rho_1) \subset S(\rho_2) \text{ and } conf(\rho_1) \leq conf(\rho_2)$$

and  $\rho_1 =_{pc} \rho_2$  if and only if

$$S(\rho_1) = S(\rho_2) \text{ and } conf(\rho_1) = conf(\rho_2).$$

A rule,  $\rho_1$  is pc-optimal if and only if  $\rho_1$  satisfies the constraints and there exists no other rule,  $\rho_2$ , that satisfies the constraints such that  $\rho_1 <_{pc} \rho_2$ . Restricting the algorithm to pc-optimal rules not only reduces the number of rules produced but also reduces time requirements, since additional pruning can be performed.

Searching for pc-optimal rules reduces the number of rules found from 4,785 to 1804. This is still a large number of rules. Applying clustering algorithms will allow us to group similar rules and will help in the presentation of the rule set.

## 5. Multi-objective Genetic Algorithms

An alternative approach to rule induction is to consider rule coverage and confidence to be two conflicting objectives. A multi-objective evolutionary algorithm [7] can then be applied in order to produce a set of rules that balance the trade-off between these objectives in different ways. De la Iglesia et al. [5, 6] applied such an algorithm, producing a different set of rules. Again, these rules were required to satisfy a set of constraints.

- Rules must be of the form described in Section 2.
- Any attribute test is permitted for a nominal field, i.e., any test of the form  $at(b) \in Val(b)$ , where  $Val(b) \subseteq Dom(at(b))$ .

The aim of the algorithm is to find rules that are Pareto-optimal with regards to the objectives of rule confidence and rule coverage.

A partial order  $\leq_{sc}$  is defined on rules where  $\rho_1 <_{sc} \rho_2$  if and only if

$$\begin{aligned} & cov(\rho_1) \leq cov(\rho_2) \text{ and } conf(\rho_1) < conf(\rho_2), \text{ or} \\ & cov(\rho_1) < cov(\rho_2) \text{ and } conf(\rho_1) \leq conf(\rho_2) \end{aligned}$$

and  $\rho_1 =_{sc} \rho_2$  if and only if

$$cov(\rho_1) = cov(\rho_2) \text{ and } conf(\rho_1) = conf(\rho_2).$$

If  $\rho_1 <_{sc} \rho_2$ , rule  $\rho_2$  is said to *dominate* rule  $\rho_1$ . A rule  $\rho_1$  is Pareto-optimal if and only if  $\rho_1$  satisfies the constraints and there exists no other rule,  $\rho_2$ , that satisfies the constraints and dominates  $\rho_1$ . A Pareto-optimal rule may also be referred to as being *sc-optimal* (support-coverage optimal) [1].

Note that, since the consequents of all the rules produced are the same,

$$cov(\rho_1) < cov(\rho_2) \Leftrightarrow |S(\rho_1)| < |S(\rho_2)|.$$

Therefore, if  $\rho_1 \leq_{pc} \rho_2$  then  $\rho_1 \leq_{sc} \rho_2$ . Note however, that the reverse is not true. For example, one rule may have greater confidence and coverage than another,

but if the two rules match disjoint sets of records then neither dominates the other according to the population-confidence partial order.

The multi-objective genetic algorithm used, known as NSGA-II (Elitist Non-Dominated Sorting Genetic Algorithm [8]), aims to find a selection of Pareto-optimal rules that range from those with high confidence but low coverage to those with low confidence and high coverage. However, since this metaheuristic cannot examine all possible rules in a reasonable time, the best rules produced are not guaranteed to be Pareto-optimal. Let  $R$  be the set of rules examined by the algorithm. When the algorithm is halted, those rules in  $R$  that are not dominated by any other rule in  $R$  are returned. Running the algorithm on the adult database with a population of size 100 for 100 generations, using a crossover rate of 0.8 and a mutation rate of 0.01, results in 428 different non-dominated rules. For further details of the implementation of the algorithm, see previous papers by de la Iglesia et al. [5, 6].

## 6. Rule Dissimilarity

There are a number of different ways to measure the dissimilarity between two rules. Firstly, the difference measure may be based either on the difference in the appearance of the rules or on the difference in the sets of records that ‘match’ the rules. The measures used in this paper all use the latter approach. Using the apparent difference between rules is a matter for further research.

Having decided to measure the dissimilarity of the sets of records that ‘match’ the rules, it is necessary to specify what is meant by ‘match.’ One approach is to use the set of records that ‘fire’ a rule  $\rho$ , i.e., the set  $L(\rho)$ . The second approach is to use the set,  $S(\rho)$ , of records that ‘support’ a rule  $\rho$ . Although we have performed experiments using distance measures based on both approaches, this paper only reports on those obtained from the support sets.

If  $\rho_1$  and  $\rho_2$  are two arbitrary rules, we can define a dissimilarity measure

$$\begin{aligned} d(\rho_1, \rho_2) &= |S(\rho_1) \cup S(\rho_2)| - |S(\rho_1) \cap S(\rho_2)| \\ &= |S(\rho_1) \setminus S(\rho_2)| + |S(\rho_2) \setminus S(\rho_1)|. \end{aligned}$$

Dividing this measure by the number of records in the database gives the simple matching coefficient [13]. Alternatively, we can use the Jaccard coefficient [12] on the sets of support and define

$$n(\rho_1, \rho_2) = d(\rho_1, \rho_2) / |S(\rho_1) \cup S(\rho_2)|.$$

Both of these dissimilarity measures are pseudometrics [9]. In other words, if we also define an equivalence relation,  $\sim$ , such that  $\rho_1 \sim \rho_2$  if and only if  $S(\rho_1) =$

$S(\rho_2)$ , then the dissimilarity measures define metrics on the equivalence classes induced by  $\sim$ .

Intuitively, two rules that are mutually supported by a thousand records and differ over only six are more similar than two rules that are mutually supported by no records and differ over five. For this reason, the results described in this paper are those obtained using the Jaccard dissimilarity coefficient.

## 7. Silhouettes

Having determined how to measure the distance between a pair of rules, these distances can be used to determine the quality of the results of a clustering algorithm in a number of different ways. The partition based clustering algorithms described in Section 8 attempt to minimize the sum of the distances of each object from its cluster's centre. An alternative measure of clustering quality can be obtained through the use of *silhouettes*.

Kaufman and Rousseeuw [13] suggest the use of silhouettes both to determine which objects lie well within their clusters and which do not and also to judge the quality of the clustering obtained. Suppose the set of objects – in this case rules –  $R$  has been partitioned into clusters. Each object  $i \in R$  belongs to one and only one cluster  $C_i$ . Let  $d(i, j)$  be the distance between objects  $i$  and  $j$ . For each object  $i$ , let  $a(i)$  be the average distance of  $i$  from all other objects in cluster  $C_i$ . For every other cluster  $C \neq C_i$ , let  $d(i, C)$  be the average distance of  $i$  from the objects in  $C$ . After computing  $d(i, C)$  for all clusters  $C \neq C_i$ , let  $b(i)$  be the smallest. The cluster for which this minimum is attained is called the *neighbour* of  $i$ .

The number,  $s(i)$ , is given by

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

and provides a measure of how well object  $i$  fits into cluster  $C_i$  rather than the neighbouring cluster. If  $s(i)$  is close to one, object  $i$  can be said to be 'well classified.' If  $s(i)$  is close to zero, it is unclear whether  $i$  should belong to cluster  $C_i$  or to its neighbour. A negative value suggests that  $i$  has been misclassified.

The following summary values are also defined:

- The mean of  $s(i)$  for all objects  $i$  in a cluster is called the *average silhouette width* of that cluster.
- The mean of  $s(i)$  for all the objects is called the *average silhouette width for the entire data set* and is denoted by  $\bar{s}(k)$ , where  $k$  is the number of clusters.

Kaufman and Rousseeuw suggest that  $\bar{s}(k)$  can be used for the selection of a best value of  $k$ , by choosing that  $k$  for which  $\bar{s}(k)$  is maximal.

Although the emphasis of this paper is on the application of simple clustering algorithms, the papers of Handl and Knowles [10, 11] are of interest. Given two

alternative methods for determining the quality of a clustering, the authors describe how a multi-objective evolutionary algorithm can be used to produce a range of different clusterings that take both measures of clustering quality into consideration.

## 8. Partition-based Clustering Algorithms

### 8.1. OVERVIEW

In this section, four partition based clustering algorithms are considered. Each of these cluster objects into  $k$  clusters, where the value for  $k$  is chosen beforehand. Section 8.2 describes why the  $k$ -means algorithm is unsuitable for the clustering of rules. However, a variant based on medoids is suitable and is described in Section 8.3.

Sections 8.4 and 8.5 describe two methods based on the application of local search to the clustering of objects around medoids. Section 8.4 describes Partitioning around Medoids (PAM), whereas Section 8.5 describes a variant called CLARANS (Clustering Large Applications based on RANdomized Search).

A summary of the results obtained is presented in each of these sections. However, more in-depth analysis of the best clusterings obtained is postponed until Section 8.6.

### 8.2. $K$ -MEANS

The  $k$ -means algorithm [14] is a well known technique for performing clustering on objects in  $\mathbb{R}^n$ . Each cluster is centred about a point called the centroid, where the centroid's coordinates are the mean of the coordinates of the objects in the cluster. The algorithm can be summarized as follows:

1. Select  $k$  points to be the initial candidates for the cluster centroids. These points need not correspond to any of the objects.
2. Assign each object to the cluster associated with the closest centroid.
3. Recalculate the positions of the  $k$  centroids.
4. Repeat Steps 2 and 3 until the centroids become fixed.

Each rule  $\rho$  can be represented by a binary string of length  $|D|$ :

$$\rho[i] = \begin{cases} 1 & \text{if record } i \text{ supports rule } \rho, \\ 0 & \text{otherwise.} \end{cases}$$

As such, the  $k$ -means algorithm can be used to perform the clustering of rules. However, there are two reasons for not doing so. Firstly, although each rule may



be represented as a binary string, centroids will not have this structure and will make little sense as rules. More importantly, the  $k$ -means algorithm requires that distances to the centroids be calculated for each object at each iteration. Since the original database may have tens of thousands of records, such calculations will be slow.

### 8.3. $K$ -MEDOIDS

The  $k$ -medoids algorithm, as used to produce the results of this paper, is an adaptation of the  $k$ -means algorithm. Rather than calculate the mean of the items in each cluster, a representative item, or medoid, is chosen for each cluster at each iteration. Medoids for each cluster are calculated by finding object  $i$  within the cluster that minimizes

$$\sum_{j \in C_i} d(i, j).$$

There are two advantages to using existing rules as the centres of the clusters. Firstly, a medoid rule serves to usefully describe the cluster. Secondly, there is no need for repeated calculation of distances at each iteration, since the  $k$ -medoids algorithm can simply look up distances from a distance matrix.

The  $k$ -medoids algorithm can be summarized as follows:

1. Choose  $k$  objects at random to be the initial cluster medoids.
2. Assign each object to the cluster associated with the closest medoid.
3. Recalculate the positions of the  $k$  medoids.
4. Repeat Steps 2 and 3 until the medoids become fixed.

Step 3 could be performed by calculating  $\sum_{j \in C_i} d(i, j)$  for each object  $i$  from scratch at each iteration. Assuming approximately equal cluster sizes, this requires  $O(n^2/k)$  operations, where  $n$  is the number of items. However, many objects remain in the same cluster from one iteration of the algorithm to the next. Improvements in speed can be obtained by adjusting the sums whenever an object leaves or enters a cluster.

The simplest approach to Step 2 is to compare, for each non-selected item, the distances to each of the  $k$  medoids, requiring  $O(k(n - k))$  operations overall. However, this can also be made more efficient in terms of speed, for larger values of  $k$ . For each object, an array of the other objects, sorted on distance, is maintained. The closest medoid can be found by scanning through this array until a medoid is found, rather than comparing the distance of every medoid. On average, a medoid is found in  $O(n/k)$  operations, so  $O(n(n - k)/k)$  operations are required to assign each item to the correct cluster.

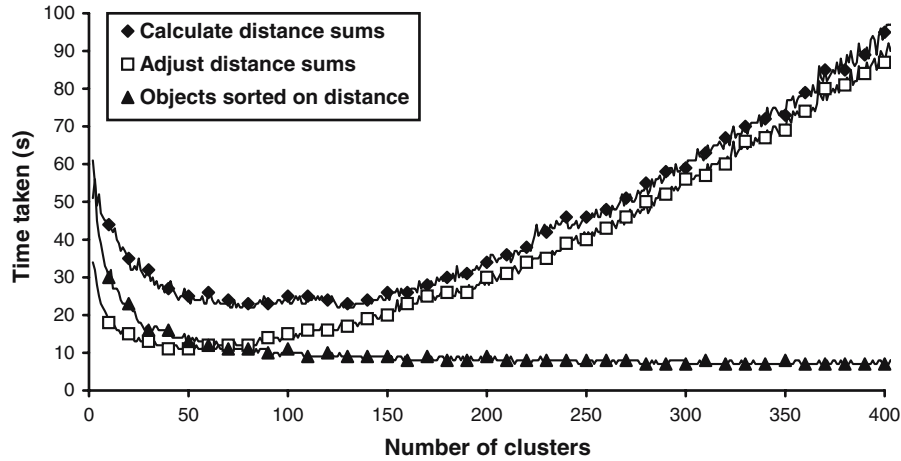


Figure 1. Time requirements of the  $k$ -medoids algorithm.

Figure 1 shows a comparison of the time requirements for each version of the code. The first series shows the requirements of the naive implementation. The second shows the speed improvements obtained by adjusting the sums required in Step 3, rather than recalculating them. In both cases, Step 3 requires most time for small values of  $k$  but Step 2 dominates for larger values of  $k$ . The third series shows the extra improvements obtained, for large values of  $k$ , by maintaining, for each object, an array of the other objects sorted on distance. Each point in the graph indicates the time required to run the  $k$ -medoids algorithm 100 times on 4,785 rules extracted from the ‘adult’ database by the all-rules algorithm, using an AMD Athlon XP 1800+ (1.53 GHz) with 480 Mb of RAM.

Since the  $k$ -medoids algorithm is quick in comparison with the other partition base clustering algorithms described in this paper, an experiment is considered to

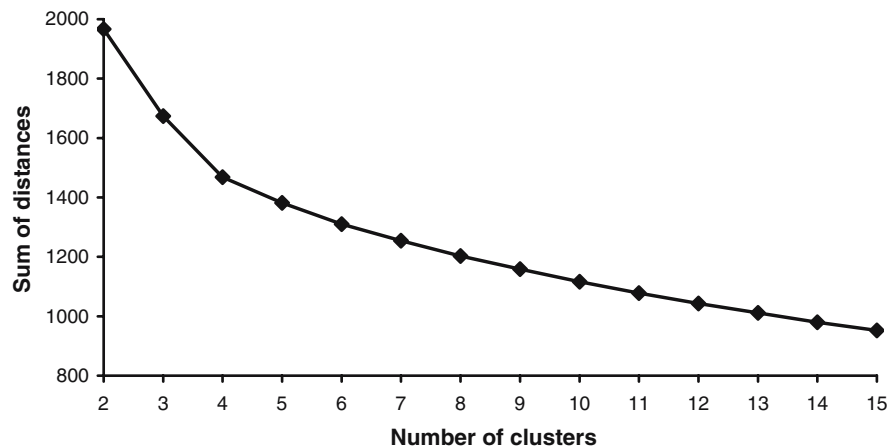


Figure 2. Sum of the distances of rules to the closest medoids.

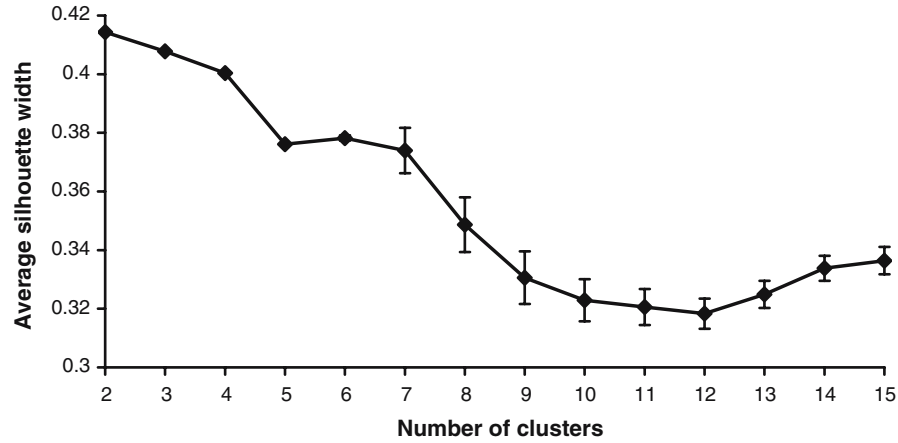


Figure 3. Silhouette widths of clusterings obtained from 100 applications of  $k$ -medoids for each  $k$ .

be the application of 100 runs of the  $k$ -medoids algorithm. The best clustering from these 100 runs, according to the sum of the distances of each rule to its medoid, is recorded. A summary of the results of 100 of these experiments, for each value of  $k$  from 2 to 15, is shown in Figures 2 and 3. Error bars are not shown in Figure 2, since the differences in cluster quality, according to sum of distances, between runs with the same value of  $k$  are insignificant when compared with the variation with  $k$ . The error bars in Figure 3 show 95% confidence intervals. The silhouette widths suggest that, if restricted to a small number of clusters, the best results are usually obtained for two to four clusters.

Better results are obtained using both PAM and CLARANS, so a more in-depth analysis of the clusters obtained by  $k$ -medoids is not presented here. However,  $k$ -medoids runs quickly for even a large number of clusters. Figure 3

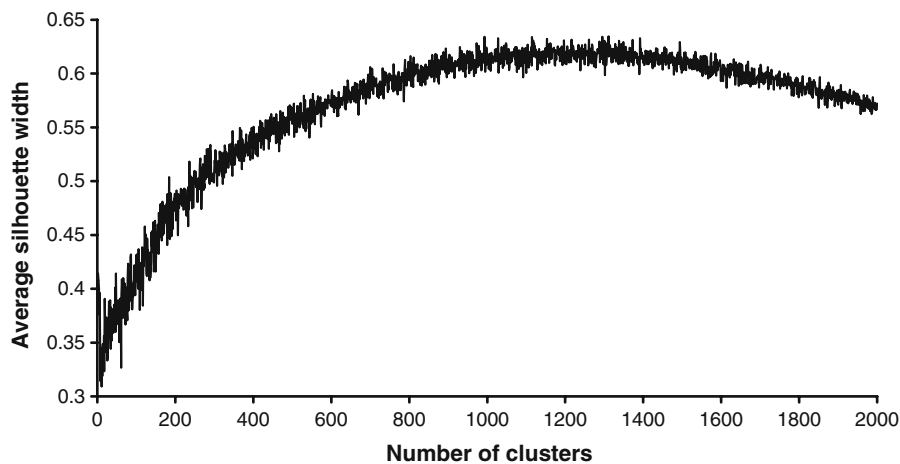


Figure 4.  $K$ -medoid silhouette widths for  $k$  up to 2,000.

does not show any clear pattern in the change in overall silhouette width as  $k$  increases, so the algorithm was run for larger values of  $k$ . Figure 4 shows the silhouette widths obtained when  $k$ -medoid is run 100 times for values of  $k$  up to 2,000.

These results suggest that the best clusterings are obtained for values of  $k$  between 1,000 and 1,400. However, such large values of  $k$  do not result in a simplified view of the data.

#### 8.4. PARTITIONING AROUND MEDOIDS (PAM)

Partitioning around medoids (PAM) [13] also clusters objects about  $k$  medoids, where  $k$  is specified in advance. However, the algorithm takes the form of a steepest ascent hill climber, using a simple swap neighbourhood operation. In each iteration medoid object  $i$  and non-medoid object  $j$  are selected that produce the best clustering when their rôles are switched. The objective function used is the sum of the distances from each object to the closest medoid.

As this search phase of the algorithm is slow, the initial set of medoids is constructed in a greedy build phase. Starting with an empty set of medoids, objects are added one at a time until  $k$  medoids have been selected. At each step, the new medoid is selected so as to minimize the objective function.

At each step of the search phase, it is necessary to calculate the cost of each move to a neighbouring solution. Kaufman et al. [13] suggest that rather than calculating the cost of a neighbouring solution from scratch, the change in cost should be determined.

Consider the effect of removing object  $i$  from the set of medoids and replacing it with object  $h$ . Let the change in the cost of the solution be  $T_{ih}$ , and let the contribution of object  $j$  to this change be  $C_{jih}$ . So

$$T_{ih} = \sum_j C_{jih}.$$

Let  $D_j$  be the distance of object  $j$  from the closest medoid, before the swap is performed. Let  $E_j$  be the distance of object  $j$  from the second closest medoid.

- If  $j$  is further from both  $i$  and  $h$  than from another medoid,  $C_{jih}$  is zero, since item  $j$  will remain in the same cluster.
- If  $j$  is closer to  $i$  than any other medoid before the swap and  $d(j, h) < E_j$ , it will be assigned to the new cluster. Hence, the contribution of object  $j$  to the swap cost is  $C_{jih} = d(j, h) - d(j, i)$ .
- If  $j$  is closer to  $i$  than any other medoid before the swap and  $d(j, h) \geq E_j$ , then  $C_{jih} = E_j - D_j$ .
- If  $j$  is further from  $i$  than from some other medoid, but closer to  $h$  than any, then it will be assigned to the new cluster. So  $C_{jih} = d(j, h) - D_j$ .

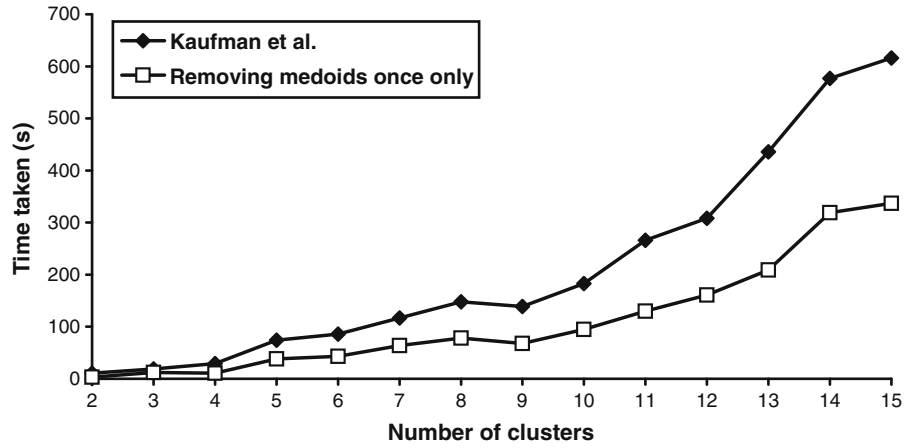


Figure 5. Efficiency improvement obtained by removing medoids once only during the evaluation of the neighbourhood.

If  $T_{ij}$  is negative, then the move gives an improvement in the clustering.

Although the closest medoid to each object must still be found, calculating  $T_{ij}$  in this way does reduce the amount of addition required.

Further improvements can be made by taking advantage of the fact that the whole neighbourhood is evaluated in each iteration of the algorithm. Neighbouring solutions can be evaluated in two steps by first removing a medoid and then adding the new medoid.  $C_{jih}$  can be given as the sum of  $CR_{ji}$  – the change in cost

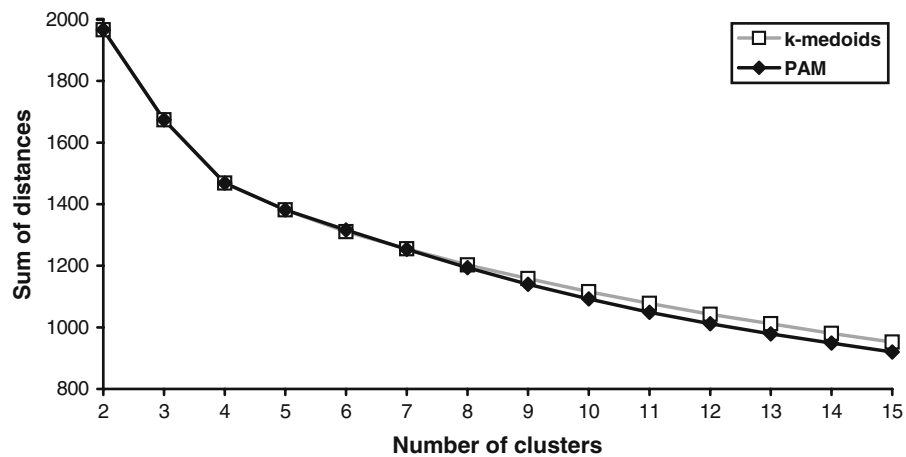


Figure 6. PAM: Sum of the distances of the rules to the closest medoids.

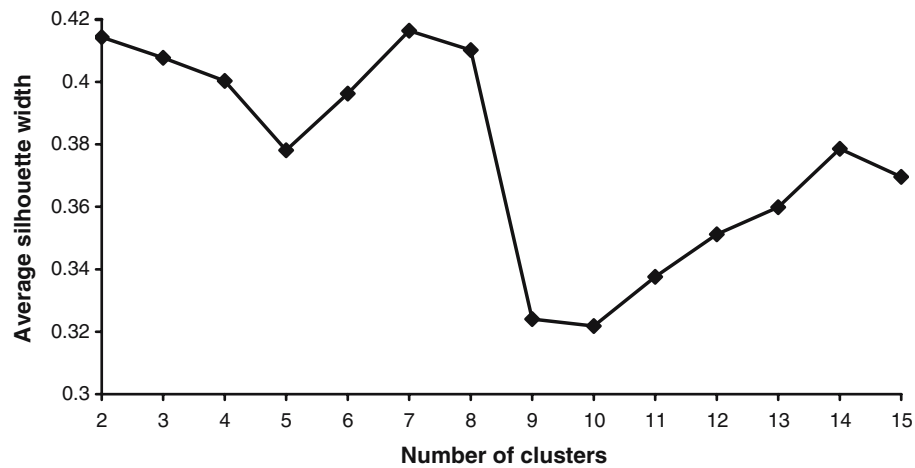


Figure 7. Silhouette widths of clusterings obtained from PAM.

for object  $j$  when medoid  $i$  is removed – and  $CA_{jh}$  – the change in cost for object  $j$  when medoid  $h$  is added.

- If  $j$  is closer to  $i$  than any other medoid,  $CR_{ji} = E_j - D_j$ , otherwise  $CR_{ji} = 0$ .
- After removing medoid  $i$ , let  $F_j$  be the distance of  $j$  from the closest remaining medoid.
- If  $j$  is closer to  $h$  than any remaining medoid,  $CA_{jh} = d(j, h) - F_j$ .

Since the entire neighbourhood is evaluated, a medoid can be removed just once before the addition of all potential alternative medoids, producing a further improvement in the efficiency of the algorithm, as shown in Figure 5.

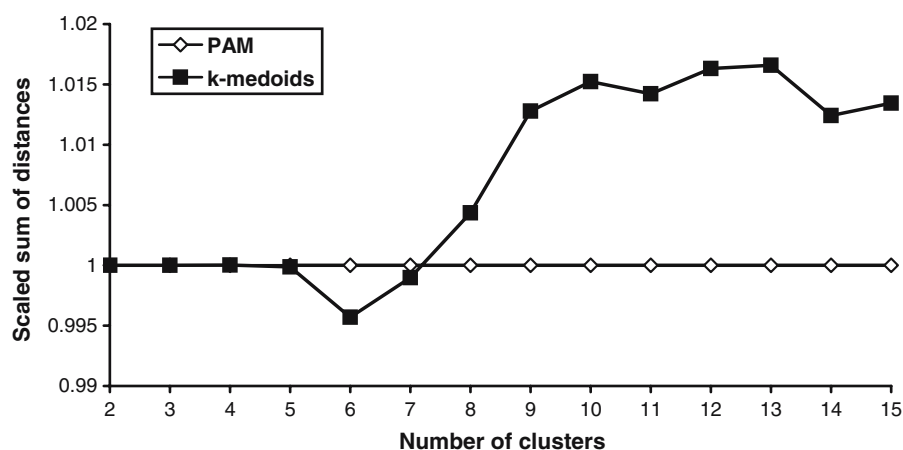


Figure 8. A comparison of  $k$ -medoids and PAM.

The results of running PAM for values of  $k$  from 2 to 15 on the 4,785 rules produced by the all-rules algorithm are shown in Figures 6 and 7. Figure 7 suggests that the best clusterings are obtained when  $k$  equals two or seven. The clusters obtained using these values of  $k$  are discussed further in Section 8.6.

The graph for PAM in Figure 6 is very similar to that for  $k$ -medoids. Another graph for CLARANS would also be very similar. In order to perform a better comparison of the partitioning algorithms, the value of the clusterings (the sum of distances) is scaled with respect to the results obtained with PAM. Furthermore, each run of  $k$ -medoids allows the algorithm to be restarted whenever a local optimum is reached for as long as it takes for one run of PAM to be completed, in order to produce a fair comparison. Finally, since  $k$ -medoids starts from a randomly generated solution, it is run 10 times and the average result is taken. The resulting graph is shown in Figure 8. Note that since PAM is entirely deterministic, no advantage is gained by running it more than once.

Figure 8 indicates that PAM outperforms  $k$ -medoids except in the cases where  $k$  is six or seven, provided PAM is given sufficient time to finish. Figure 9 shows how the quality of the solutions obtained improve over time for both of the algorithms, when  $k$  is set to 15. As PAM starts with a build phase, the initial solution for this algorithm requires time to be constructed. Therefore the initial solution for PAM is marked in Figure 9. Note that provided PAM is given enough time to build the initial solution, it outperforms  $k$ -medoids.

While PAM usually produces better results than  $k$ -medoids for small values of  $k$ , it is impractical for large  $k$  due to increased time requirements. Figure 10 shows how the time requirements for PAM increase with  $k$ . (It can be shown that each iteration of the algorithm has a time complexity of  $O(k(n - k)^2)$ ). Furthermore, the number of iterations required can be expected to be higher

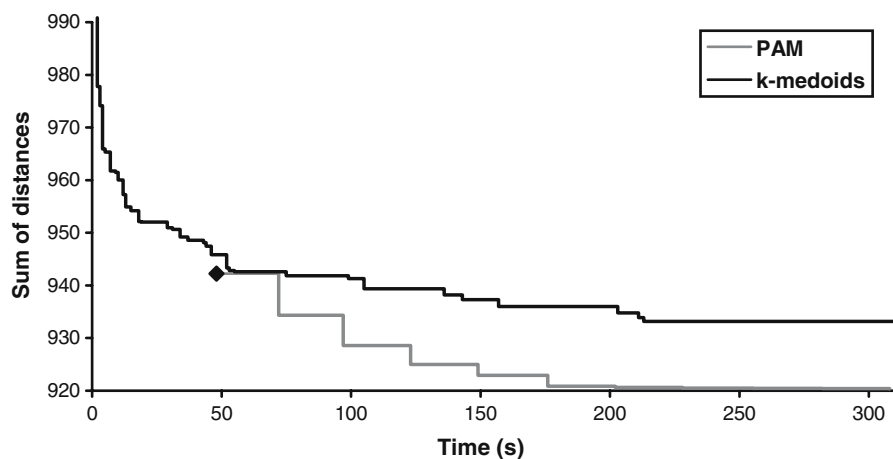


Figure 9. Solution improvement graphs for  $k$ -medoids and PAM.

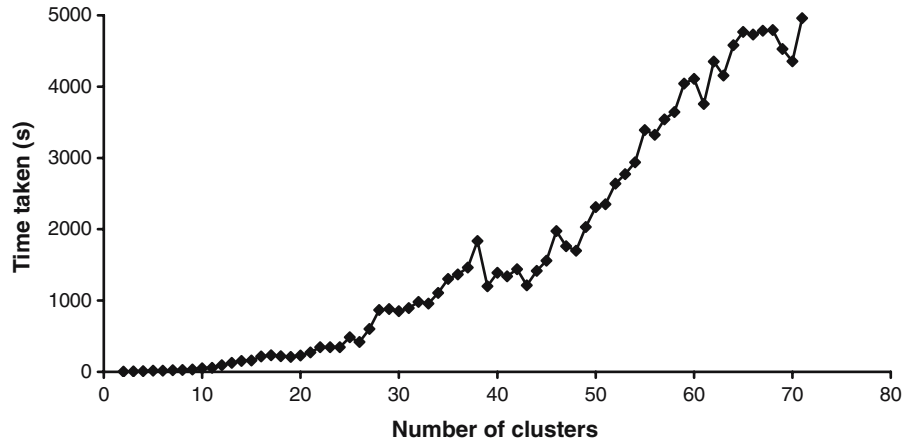


Figure 10. Time requirements for the PAM algorithm.

when both  $k$  and  $n - k$  are large, since the search space is larger and each solution has more neighbours).

#### 8.5. CLARANS

CLARANS [15] is a variant of PAM that uses the same neighbourhood operation but takes the form of a stochastic first-found hill climber. In each iteration, a medoid object,  $i$ , and non-medoid object,  $j$ , are selected at random until the clustering produced when their rôles are switched is better than the current clustering. No build phase is required; the algorithm starts with a randomly selected set of  $k$ -medoids.

Ng et al. [15] describe how the algorithm has two parameters, *numlocal* and *maxneighbour*. The first of these indicates how many runs of the local search algorithm are performed. At the end of each run, the algorithm restarts at a randomly selected solution. The second parameter, *maxneighbour*, indicates the maximum number of neighbours the algorithm examines at each step. If this many neighbours are examined without finding one that is better than the current solution, the current solution is declared to be a ‘local minimum’ and the next run of the local search is started.

In order to compare CLARANS fairly with the other algorithms tested, it was decided that it would be more useful to provide a time limit as a parameter, rather than *numlocal*. Setting the time limit to be the time required by PAM produced the results summarized in Figure 11. Here CLARANS was run with three different values for *maxneighbour*, 100, 1,000 and 10,000. As before, the cost of the clusterings obtained are scaled with respect to PAM, so that PAM’s results are represented by the horizontal line. Also, CLARANS was run



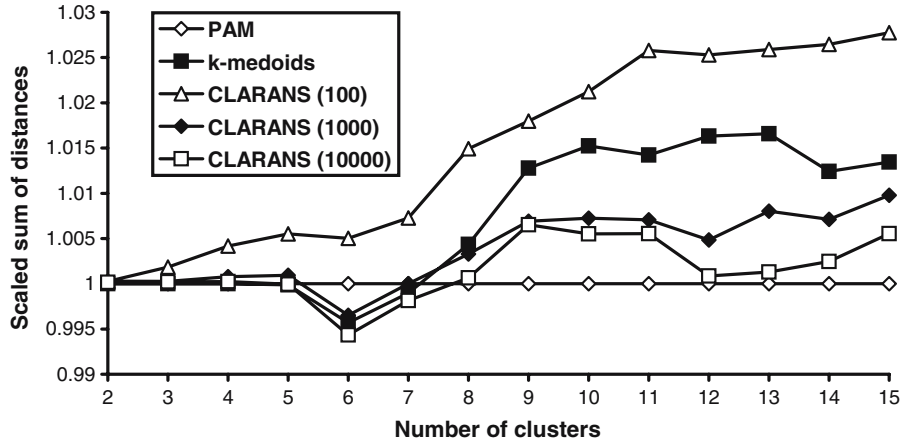


Figure 11. A comparison of CLARANS with  $k$ -medoids and PAM.

10 times for each combination of  $k$  and  $maxneighbour$  and the average results are plotted.

When  $maxneighbour$  has values of 1,000 or 10,000, CLARANS easily outperforms  $k$ -medoids. However, CLARANS only outperforms PAM when  $k$  is six or seven.

Part of the motivation for the development of CLARANS is the fact that PAM can take a long time to finish, especially on large databases and for large values of  $k$ . A worthwhile question to ask is whether CLARANS outperforms PAM on shorter runs. Considering the case where  $k$  is 15, the graph of Figure 12 shows how the quality of the solutions improves over time for both PAM and CLARANS. During the first third of the run, both versions of CLARANS

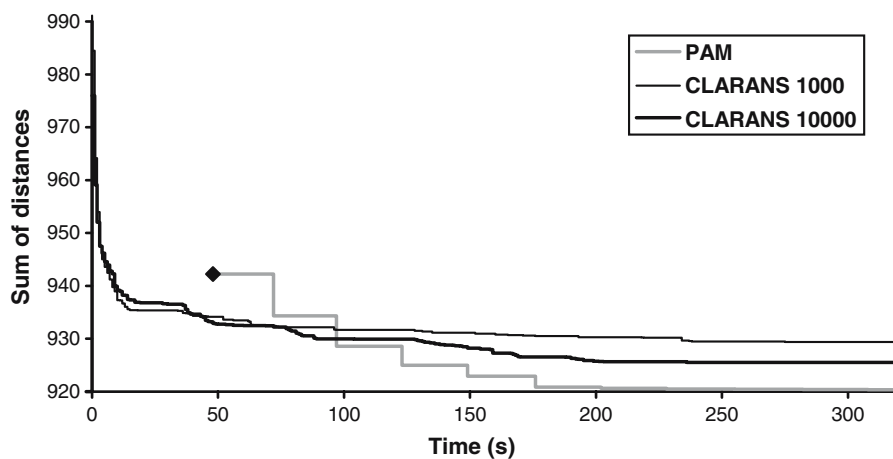


Figure 12. Solution improvement graphs for  $k = 15$ .

outperform PAM, but PAM produces improved solutions over the remainder of the run.

PAM takes a long while to reach reasonable solutions, because each step of PAM requires the evaluation of the entire neighbourhood. However, it does not halt until a true local optimum is found. If *maxneighbour* is small compared to the neighbourhood size, CLARANS will declare a solution to be a “local optimum” after evaluating only a fraction of the neighbourhood, and then restart from a random solution. CLARANS finished with better results in the reported experiments for higher values of *maxneighbour*. However, with higher values of *maxneighbour* CLARANS may evaluate the same, non-improving neighbour several times before either finding a better solution or declaring that a “local optimum” is reached.

We use these facts as motivation for the examination of variants of CLARANS, so that algorithms can be found that perform well in early stages of the search through the use of a ‘first-found’ strategy but also have the persistence of PAM.

The first variant, deterministic CLARANS (d-CLARANS), uses random starting solutions but evaluates possible moves in a set sequence. Given an array of medoids and an array of unselected rules, a move is represented by two numbers representing positions in these arrays. The sequence of moves starts with those that replace the first medoid, then those that replace the second, and so on.

After a move to a better solution is found and taken, the algorithm does not return to the first move in the sequence, but continues from where it left off. This prevents the algorithm from concentrating solely on improving just the first few medoids. Once the end of the sequence of moves is reached, the algorithm continues from the start of the sequence. If the entire sequence is examined without finding an improving move, the search has reached a local optimum. Note that experiments have also been performed where the first move considered at each solution is selected at random, though results are not of the quality obtained using d-CLARANS.

The second variant, persistent CLARANS (p-CLARANS), performs a stochastic local search in the same way as basic CLARANS, except that neighbours of a solution are selected without replacement. Again, the search is not declared to be at a local minimum until all the moves have been examined without finding one that improves the clustering.

Both of these approaches prevent CLARANS from evaluating the same neighbour twice, ensure that the entire neighbourhood is evaluated before the algorithm declares that it has reached a local optimum and remove the need for the parameter *maxneighbour*.

When run for the same amount of time as one run of PAM, both d-CLARANS, p-CLARANS and PAM produce results that are of approximately the same quality, as shown in Figure 13. Comparing the algorithms over

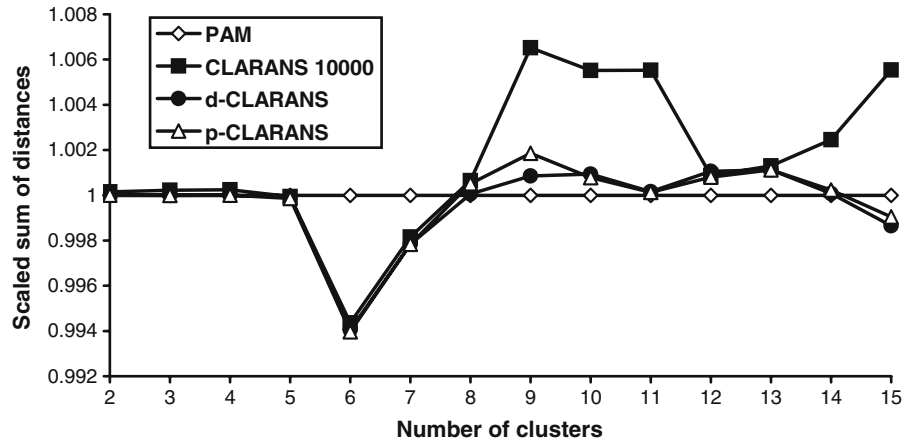


Figure 13. A comparison of p-CLARANS and d-CLARANS with standard CLARANS and PAM.

shorter time periods illustrates the true advantage of the two variants over both basic CLARANS and PAM. Figure 14 shows how the quality of the solutions produced by the four algorithms improves over time. Note how p-CLARANS in particular combines speed in obtaining good solutions and high final solution quality.

#### 8.6. THE CLUSTERS OBTAINED

As mentioned in Section 8.4, when applied to the 4,785 rules produced by the all-rules algorithm from the adult database, PAM obtains the best clusterings

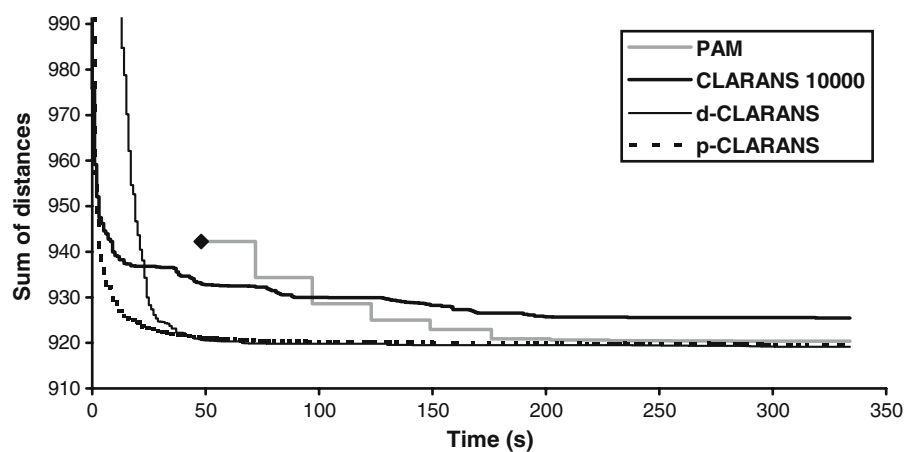


Figure 14. Solution improvement graphs for  $k = 15$ .

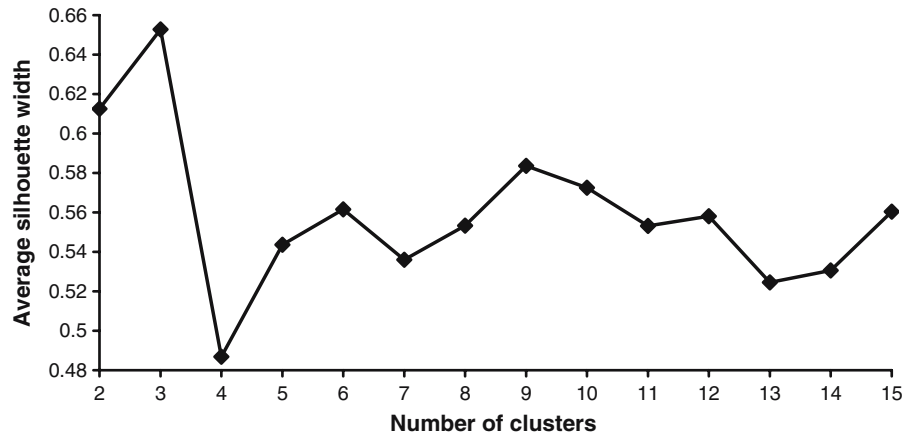


Figure 15. Silhouette widths of clusterings obtained by using PAM on the rules generated by NSGA-II.

according to silhouette width when  $k$  equals two or seven. In both cases, the strongest cluster contains all 1,433 rules with a clause of the form “Cap\_gain  $\geq x$ ”. This cluster has an average silhouette width of 0.74 in both cases and has “IF Age  $\geq 32$  AND Cap\_gain  $\geq 3,103$  THEN \*Salary = > 50 K” as its medoid. In the two cluster case, the other 3,352 rules appear in the second cluster, which has an average silhouette width of 0.27. In the seven cluster case, these rules are split into six clusters, with average silhouette widths from 0.07 to 0.37.

The clusters produced can be examined on a confidence–coverage plot. Rules that are close according to the distance measures described will also have similar

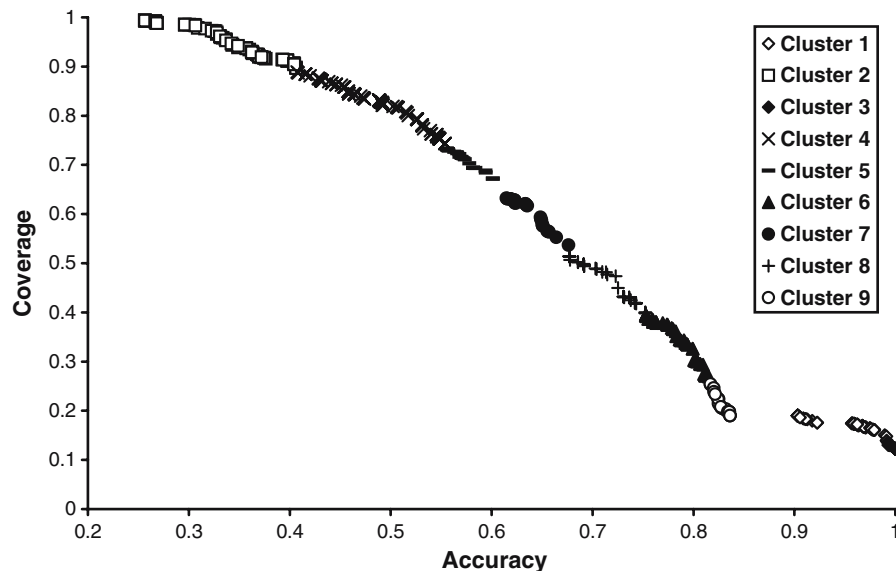


Figure 16. Clusters obtained using PAM, with  $k = 9$ , on the rules produced by NSGA-II.

values for confidence and coverage. However, two rules with similar confidence and coverage need not be close according to the distance measure. Therefore, one would expect the clusters to overlap on such plots. However, when PAM is applied to the rules generated by NSGA-II, the Pareto-front is grouped into clusters with little or no overlap on the confidence coverage plot.

Figure 15 shows the average silhouette widths obtained for each value of  $k$  from two to fifteen. The highest silhouette width occurs at  $k = 3$ , with another peak occurring at  $k = 9$ . Figure 16 shows the confidence coverage plot for  $k$  equal to nine. The clusters obtained with  $k = 3$  can almost be obtained by merging clusters in Figure 16. The first cluster is the result of merging clusters one and three in Figure 16. The second is obtained by merging clusters two, four, five and seven and adding the closest member of cluster eight. The third cluster contains the remainder.

## 9. Hierarchical Clustering Algorithms

### 9.1. THE EFFECT OF NUMERICAL FIELDS AND NEAR IDENTICAL RULES

The all-rules algorithm ensures that the only numerical values that are permitted in an attribute test are those that appear in the training data. Suppose, for example, that the training data does not include a record with age equal to 19. Both of the rules

```
if age ≤ 18 then claim = yes
```

and

```
if age ≤ 19 then claim = yes
```

match the same set of records and have the same confidence and coverage. The all-rules algorithm produces only the first of these rules, despite there being equal evidence for the second.

Although the all-rules algorithm prevents the production of some near identical rules that match identical sets of records, it does not eliminate all such duplicates. For example, suppose the training data includes a record with age equal to fifty, but no rule with both age equal to 50 and sex equal to male. Then both of the rules

```
if sex = male and age ≤ 49 then claim = yes
```

and

```
if sex = male and age ≤ 50 then claim = yes
```

match the same set of records and have the same confidence and coverage. However, in this case, both rules are produced by the all-rules algorithm.

The presence of numerical fields can cause the proliferation of near identical rules, e.g.,

```
if age  $\geq$  16 and income  $\geq$  12,000 then claim = yes,
if age  $\geq$  16 and income  $\geq$  12,500 then claim = yes,
if age  $\geq$  17 and income  $\geq$  12,000 then claim = yes,
if age  $\geq$  17 and income  $\geq$  12,500 then claim = yes
```

Notice that none of these rules is considered to be a sub-rule of any of the others, so the all-rules algorithm will produce all of these rules, without regard to the ‘improvement’ constraint. The proliferation of such rules creates a bias, in the clustering algorithms considered thus far, towards clusters that are centred around such rules. This is unfortunate, since once one of these rules is known, each subsequent rule provides little extra understanding of the data.

The rest of this section describes how agglomerative hierarchical clustering algorithms can be used that:

- produce essentially the same clustering, regardless of how the all-rules algorithm handles near identical rules that match identical sets of records and
- do not suffer the aforementioned bias when the data includes numerical fields.

Section 9.2 describes agglomerative nesting (AGNES) [13]. Then Section 9.3 describes variants of AGNES that are blind to the number of times the same rule or similar rules occur once such rules have been grouped together. Finally, Section 9.4 shows some of the results of applying these algorithms to the clustering of rules.

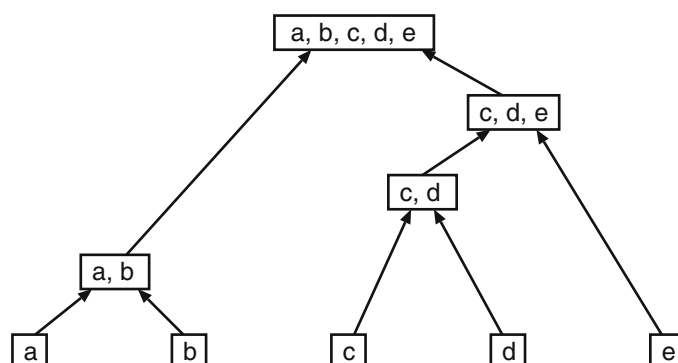


Figure 17. Agglomerative clustering.

### 9.2. AGGLOMERATIVE NESTING

Given  $n$  items to cluster, agglomerative nesting starts with  $n$  clusters, each containing one item. At each step, the closest pair of clusters are agglomerated, to form a new larger cluster. This is repeated until there is just one cluster containing all  $n$  items. The result is a tree-like hierarchy, with the individual items in the leaf nodes and all other nodes representing clusters of two or more items.

Figure 17 shows the hierarchy produced from a simple case with just five items. Starting from the bottom, each item is considered to be in its own cluster. Then, the two closest clusters – those containing  $a$  and  $b$  – are joined to make a new cluster. Now the two closest clusters are those containing  $c$  and  $d$ , so these two clusters are joined. This process continues until only one cluster is left, containing all five clusters. The hierarchy in Figure 17 is displayed so that clusters that are formed earlier in the search are displayed lower down than those that are formed later.

Partitions for different values of  $k$  are found by taking a snapshot of the algorithm before completion. In the example, taking a snapshot just after the cluster containing  $c$  and  $d$  is created produces three clusters,  $\{a, b\}$ ,  $\{c, d\}$  and  $\{e\}$ . As care has been taken to ensure that clusters appear lower if they are created earlier, such snapshots can be obtained by cutting across the tree in Figure 17 with a horizontal line.

The framework described permits a number of clustering algorithms that differ solely on the method for calculating the dissimilarity between clusters of rules. The algorithm AGNES, as described by Kaufman and Rousseeuw [13] uses the group average method of Sokal and Michener [18]. The dissimilarity between clusters  $R$  and  $S$  is defined as the average of all dissimilarities  $d(r, s)$ , where  $r \in R$  and  $s \in S$ . When merging clusters  $A$  and  $B$  to create cluster  $R$ , the distance of the new cluster from some other cluster  $Q$  can be calculated as

$$d(R, Q) = \frac{|A|}{|R|}d(A, Q) + \frac{|B|}{|R|}d(B, Q).$$

### 9.3. UNBIASED VARIANTS OF AGNES

As described above, the group average method calculates distances between a newly created cluster,  $R$ , and another cluster,  $Q$ , by using a weighted average of the distances of the constituent clusters,  $A$  and  $B$ , from  $Q$ . These weights are proportional to the number of items in clusters  $A$  and  $B$ . As a result, if one of these clusters, for example  $A$ , consists of a large group of identical or near identical rules, produced via minor changes to bounds on numerical fields, the distances calculated for the new cluster are biased towards the distances for cluster  $A$ .

Kaufman and Rousseeuw [13] also describe variants of AGNES that use different methods for calculating the dissimilarity between clusters. A number of these are blind to the number of items in each of the clusters to be agglomerated. If one of these variants is used to cluster rule sets that contain copies of identical rules, then any subset of identical rules is rapidly clustered. From then on, these clusters of identical rules are considered in essentially the same way as a single rule. Therefore, these variants do not suffer from a bias towards clusters centred on rules involving numerical fields. Furthermore, if rules are duplicated, these variants will produce the same clustering as when they are not.

The variant considered here uses the weighted average linkage [19] to calculate the distance between a newly created cluster and the remaining clusters. When merging clusters  $A$  and  $B$  to create cluster  $R$ , the distance of the new cluster from some other cluster  $Q$  can be calculated as

$$d(R, Q) = \frac{1}{2}d(A, Q) + \frac{1}{2}d(B, Q).$$

Results using this variant of AGNES are given in Section 9.4.

An alternative approach would be to consider that each cluster of rules represents a rule itself. The antecedent of this rule is simply the disjunction of the antecedent of the rules in the cluster. Considering each cluster as a rule, it is possible to calculate the distance between clusters in exactly the same way as that between simple rules. A potential disadvantage in this approach is that calculating such distances is likely to be computationally expensive. All other clustering algorithms described in this paper can work from a pre-prepared dissimilarity matrix. Exploring this alternative approach is a matter for further research.

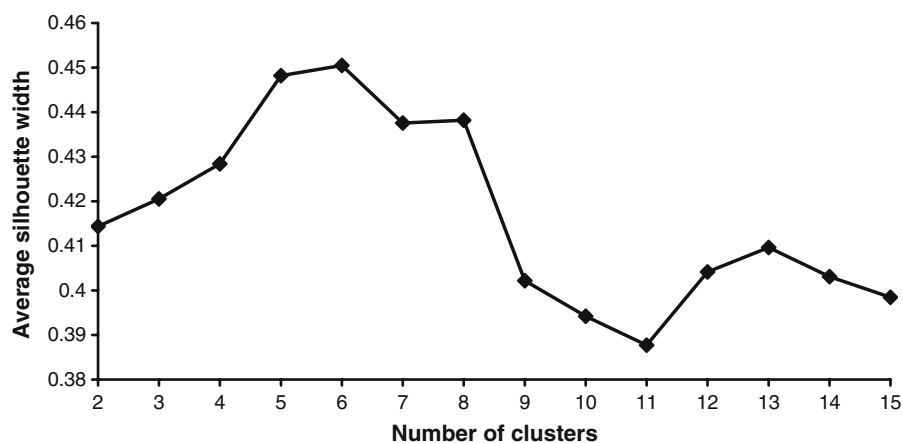


Figure 18. Silhouette widths of clusterings extracted from the AGNES hierarchy for rules produced by the all-rules algorithm.



## 9.4. RESULTS

Agglomerative clustering was applied to the rules produced by the all-rules algorithm, using the weighted average linkage to calculate inter-cluster distance. Partitions were then extracted from the hierarchy produced for values of  $k$  from two to fifteen. The mean silhouette widths were calculated and plotted against  $k$  to produce the graph in Figure 18. The best clustering extracted, according to silhouette widths, is that containing six clusters.

Extracting just two clusters produces exactly the same results as running PAM with  $k$  equal to two. However, comparing Figures 18 and 7 shows that, according to silhouette widths, the clustering extracted from the AGNES hierarchy with  $k$  equal to six is better than any clustering obtained by PAM with small values of  $k$ .

One of these six clusters is the same strong cluster of 1,433 rules obtained by all the algorithms, with a silhouette width of 0.74. The other five clusters have silhouette widths varying between 0.27 and 0.72. The clusters obtained by AGNES in this way vary considerably in size, varying from 49 to 2,093 items. The cluster of 49 items has a high average silhouette width (0.72). PAM and the related partition-based algorithms tend to produce clusters of more equal sizes,

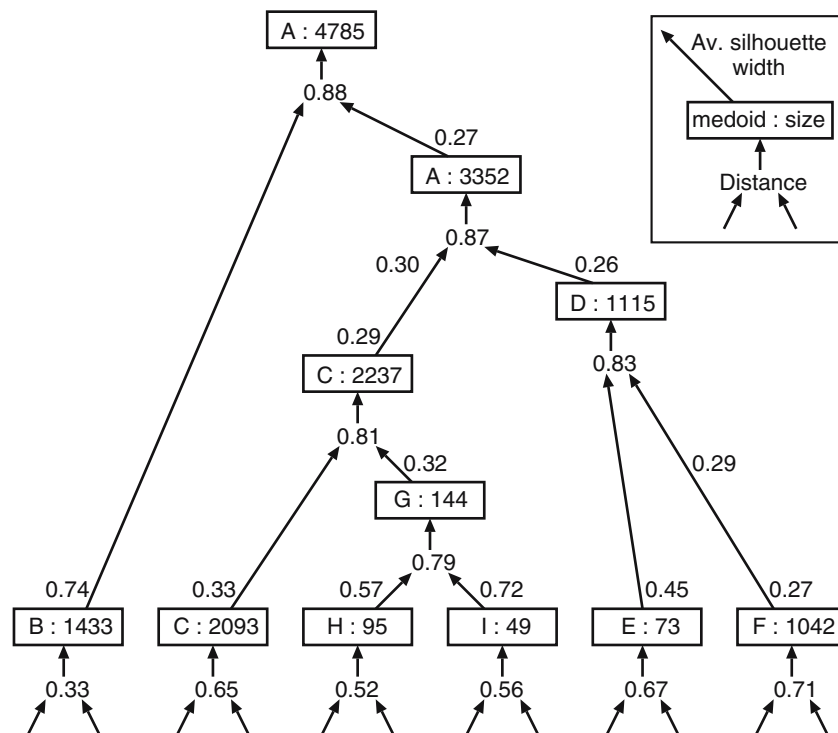


Figure 19. The clustering hierarchy produced by AGNES on rules produced by the all-rules algorithm, down to  $k = 6$ .

Table I. Medoid rules from the top of the AGNES hierarchy.

Medoid	Rule
A	IF Age $\geq$ 36 AND Edu-yrs $\geq$ 11 AND Sex = Male AND Hours-per-week $\geq$ 43 THEN Salary > 50 K
B	IF Age $\geq$ 32 AND Cap-gain $\geq$ 3103 THEN Salary = > 50 K
C	IF Age $\geq$ 36 AND Edu-yrs $\geq$ 11 AND Sex = Male AND Hours-per-week $\geq$ 44 THEN Salary > 50 K
D	IF Age $\geq$ 45 AND Edu-yrs $\geq$ 11 AND Sex = Male AND Hours-per-week $\geq$ 36 THEN Salary > 50 K
E	IF Age $\geq$ 31 AND Edu-yrs $\geq$ 11 AND Work = Prof-specialty AND Sex = Male THEN Salary > 50 K
F	IF Age $\geq$ 45 AND Edu-yrs $\geq$ 11 AND Sex = Male AND Hours-per-week $\geq$ 37 THEN Salary > 50 K
G	IF Age $\geq$ 32 AND Edu-yrs $\geq$ 10 AND Work = Exec-managerial AND Sex = Male THEN Salary > 50 K
H	IF Age $\geq$ 31 AND Edu-yrs $\geq$ 10 AND Work = Exec-managerial AND Sex = Male THEN Salary > 50 K
I	IF Age $\geq$ 36 AND Education = Bachelors AND Sex = Male AND Hours-per-week $\geq$ 38 THEN Salary > 50 K

which mean that such strong but small clusters end up grouped with fairly unrelated items to make a larger cluster.

Figure 19 shows the top of the clustering hierarchy produced by AGNES, down to  $k = 6$ . Inside each box is a letter that represents the medoid of the cluster and the number of rules. The medoid rules are given in Table I. AGNES does not require that the medoids be found, but they are calculated here to represent each cluster. Beneath each box is the distance between the component sub-clusters. The other numbers on each of the arrows give the silhouette widths of each

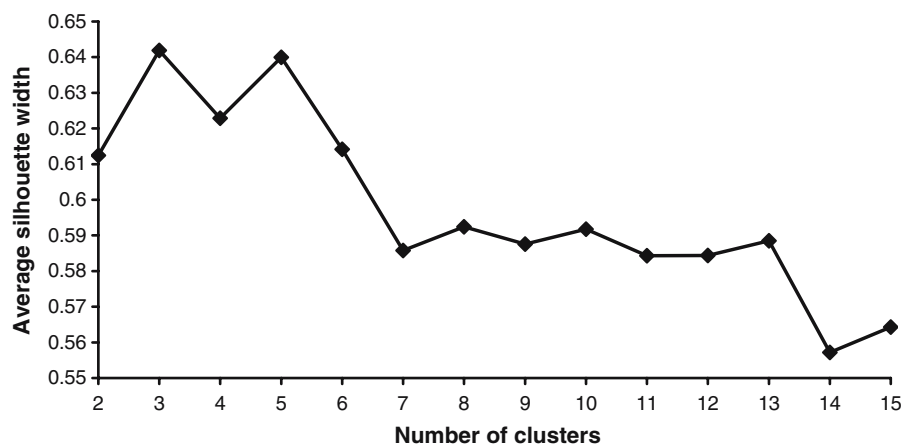


Figure 20. Silhouette widths of clusterings extracted from the AGNES hierarchy for rules produced by NSGA-II.

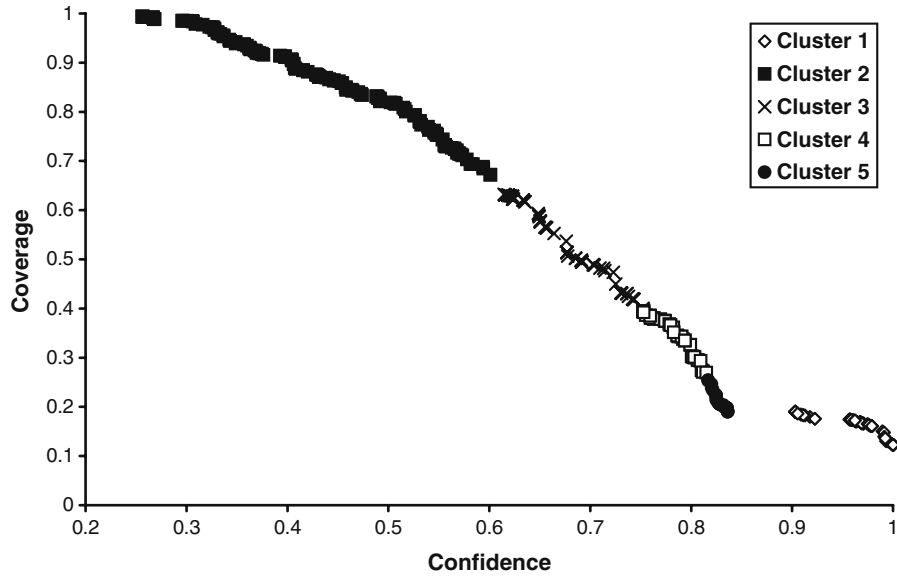


Figure 21. Clusters obtained using AGNES, with  $k = 5$ , on the rules produced by NSGA-II.

cluster. Note that the silhouette width of a cluster may change as other clusters are merged.

AGNES was also applied to the rules produced from NSGA-II. Clusterings with  $k$  varying from two to 15 were extracted from the hierarchy produced and the mean silhouette widths were calculated. The results are shown in Figure 20.

The best clusterings, according to silhouette widths, are obtained with  $k$  equal to three and five. Plotting the five clusters on a confidence-coverage plot produces Figure 21. The same plot for  $k$  equal to three can be created by merging clusters two and three and clusters four and five. Notice again that there is no overlap between the clusters. There are two possible explanations for this. The first is that it may be the case that rules on or near the Pareto-optimal front that have similar confidence and coverage are also close with regards to the rule dissimilarity measures. The second is that very different rules with similar accuracy and coverage may exist near the Pareto-optimal front, but that NSGA-II, which uses a crowding measure based solely on confidence and coverage, fails to find them.

## 10. Conclusions and Further Research

The  $k$ -medoids algorithm, PAM and CLARANS have been applied to the clustering of rules obtained from both the all-rules algorithm and from a multi-objective genetic algorithm. Furthermore, a modification to CLARANS has been made that allows it to perform as well as PAM in long runs and significantly outperform all the partition-based algorithms tested in short runs.

The clusters obtained by applying the partition-based algorithms to the rules produced by the all-rules algorithm seem to be reasonable, especially the cluster of all rules containing a clause of the form ‘Cap\_gain  $\geq x$ ,’ which is the only strong cluster produced by PAM when  $k$  is two or seven. When the rules produced by the multi-objective genetic algorithm are clustered, the Pareto-front is partitioned very neatly, suggesting that, in this case, clustering using rule distances produces a similar result to clustering based solely on the difference in the objective values.

In addition to partition-based algorithms, a variant of a hierarchical clustering algorithm known as AGNES has been applied to these sets of rules. By choosing a suitable method for calculating the distance between clusters, it was possible to produce an algorithm that does not have a bias towards clusters centred groups of near identical rules involving numerical fields. Taking cuts through the cluster hierarchy produced by AGNES resulted in different partitions from those obtained using the partition-base algorithms, with a greater variation in cluster size and improved silhouette widths. Note, however, that further work is required to determine whether this improvement in average silhouette widths is due to an significant advantage in the algorithm when using this measure of clustering quality. In order to test this hypothesis, the algorithms could be compared on rule sets generated from a range of different data sets.

There are a number of other potential areas for further research:

- Better methods for summarizing clusters that give more information than just the cluster medoid are required if we are to improve our understanding of the data via clustering of rules. The confidence–coverage plots, while providing some information, do not help us to describe the clusters.
- Our difference measures are based purely on the sets of supporting records. An alternative strategy would be ‘syntax-based’, i.e., based on the actual expressions used in the rules. A comparison of the clusters obtained using the two strategies is a matter for further research.
- Clusters of rules can be considered as rules themselves, which means that distances between clusters can be calculated in the same way as between rules. An agglomerative clustering algorithm can use this method for calculating distances between clusters.
- Rule clusters produced from the rules found by NSGA-II form tidy regions with no overlap on a confidence–coverage plot. Since rules that are close on this plot need not be close with regards to the dissimilarity measures, some overlap between the clusters might be expected. Further research might explain this phenomenon.
- The application of metaheuristics to the problem of clustering rules is an obvious extension to this work. Some work has been reported on the modification of clustering algorithms to incorporate metaheuristics, for example the development, by Chu et al. [4], of a simulated annealing version of

CLARANS known as CLASA. The combination of our improvements to CLARANS with suitably selected metaheuristic techniques, and the application of the algorithms produced to the clustering of rules is a matter for further research.

## References

1. Bayardo, Jr., R. J. and Agrawal, R.: Mining the most interesting rules, in S. Chaudhuri and D. Madigan (eds.), *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, California, United States, 1999, pp. 145–154.
2. Bayardo, Jr., R. J., Agrawal, R. and Gunopulos, D.: Constraint-based rule mining in large, dense databases, in *Proceedings of the 15th International Conference on Data Engineering*, Sydney, Australia, 1999, pp. 188–197.
3. Blake, C. and Merz, C.: ‘UCI Repository of machine learning databases,’ (1998), <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
4. Chu, S. C., Roddick, J. F. and Pan, J. S.: A comparative study and extensions to  $k$ -medoids algorithms, in *Fifth International Conference on Optimization*, Hong Kong, China, 2001, pp. 1708–1717.
5. de la Iglesia, B., Philpott, M. S., Bagnall, A. J. and Rayward-Smith, V. J.: Data mining rules using multi-objective evolutionary algorithms, in R. Sarker, R. Reynolds, H. Abbass, K. C. Tan, B. McKay, D. Essam, and T. Gedeon (eds.), *Proceedings of 2003 IEEE Congress on Evolutionary Computation*, Canberra, Australia, 2003, pp. 1552–1559.
6. de la Iglesia, B., Reynolds, A. and Rayward-Smith, V. J.: Developments on a Multi-Objective Metaheuristic (MOMH) algorithm for finding interesting sets of classification rules, in C. A. Coello Coello, A. H. Aguirre and E. Zitzler (eds.), *Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005*, Guanajuato, Mexico, 2005, pp. 826–840.
7. Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms*, Chichester, Wiley, England, 2001.
8. Deb, K., Agrawal, S., Pratab, A. and Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II,’ in Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo, Hans-Paul Schwefel (eds.), *Proceedings of the Parallel Problem Solving from Nature VI Conference. Lecture Notes in Computer Science No. 1917*, Paris, France, 2000, pp. 849–858.
9. Gower, J. C. and Legendre, P.: Metric and Euclidean properties of dissimilarity coefficients, *J. Classif.* **3** (1986), 5–48.
10. Handl, J. and Knowles, J.: Evolutionary multiobjective clustering, in X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervs, J. Bullinaria, J. Rowe, P. Tino, A. Kabn, and H.-P. Schwefel (eds.), *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, Birmingham, UK, 2004, pp. 1081–1091.
11. Handl, J. and Knowles, J.: Exploiting the trade-off – the benefits of multiple objectives in data clustering, in C. A. Coello Coello, A. H. Aguirre and E. Zitzler (eds.), *Evolutionary Multi-Criterion Optimization: Third International Conference, EMO 2005*, Guanajuato, Mexico, 2005, pp. 547–560.
12. Jaccard, P.: Étude comparative de la distribution florale dans une portion des Alpes et des Jura, *Bull. Soc. Vaud. Sci. Nat.* **37** (1901), 547–579.
13. Kaufman, L. and Rousseeuw, P. J.: *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley series in probability and mathematical statistics, Wiley, 1990.

14. MacQueen, J. B.: Some methods for classification and analysis of multivariate observations, in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1, 1967, pp. 281–297.
15. Ng, R. T. and Han, J.: CLARANS: A method for clustering objects for spatial data mining, *IEEE Trans. Knowl. Data Eng.* **14**(5) (2002), 1003–1016.
16. Reynolds, A. P., Richards, G. and Rayward-Smith, V. J.: The application of  $K$ -medoids and PAM to the clustering of rules, in Z. R. Yang, H. Yin, and R. Everson (eds.), in *Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'04)*, 2004, pp. 173–178.
17. Richards, G. and Rayward-Smith, V. J.: Discovery of association rules in tabular data, in N. Cercone, T. Y. Lin and X. Wu (eds.), in *Proceedings of IEEE First International Conference on Data Mining, San Jose, California, USA*, San Jose, California, 2001, pp. 465–473.
18. Sokal, R. R. and Michener, C. D.: A statistical method for evaluating systematic relationships, *Univ. Kans. Sci. Bull.* **38** (1958), 1409–1438.
19. Sokal, R. R. and Sneath, P. H. A.: *Principles of Numerical Taxonomy*, Freeman, San Francisco, 1963.