Efficient and Effective Clustering Methods for Spatial Data Mining

Raymond T. Ng*
Department of Computer Science
University of British Columbia
Vancouver, B.C., V6T 1Z4,
Canada.

Jiawei Han†
School of Computing Sciences
Simon Fraser University
Burnaby, B.C., V5A 1S6,
Canada.

Abstract

Spatial data mining is the discovery of interesting relationships and characteristics that may exist implicitly in spatial databases. In this paper, we explore whether clustering methods have a role to play in spatial data mining. To this end, we develop a new clustering method called CLARANS which is based on randomized search. We also develop two spatial data mining algorithms that use CLARANS. Our analysis and experiments show that with the assistance of CLARANS, these two algorithms are very effective and can lead to discoveries that are difficult to find with current spatial data mining algorithms. Furthermore, experiments conducted to compare the performance of CLARANS with that of existing clustering methods show that CLARANS is the most efficient.

keywords: spatial data mining, clustering algorithms, randomized search

1 Introduction

Data mining in general is the search for hidden patterns that may exist in large databases. Spatial data mining in particular is the discovery of interesting relationships and characteristics that may exist implicitly in spatial databases. Because of the huge amounts (usually, tera-bytes) of spatial data that may be obtained from satellite images, medical equipments, video cameras, etc., it is costly and often unrealistic for users to examine spatial data in detail. Spatial data mining aims to automate such a knowledge discovery process. Thus, it plays an important role in a) extracting interesting spatial patterns and features; b) capturing intrinsic relationships between spatial and non-spatial data; c) presenting data regularity concisely and at higher conceptual levels; and d) helping to reorganize spatial databases to accommodate data semantics, as well as to achieve better performance.

*Research partially sponsored by NSERC Grants OGP0138055 and STR0134419.
†Research partially supported by NSERC Grant OGP03723 and the Centre for Systems Science of Simon Fraser University.

Many excellent studies on data mining have been conducted, such as those reported in [1, 2, 4, 7, 11, 13, 15]. [1] considers the problem of inferring classification functions from samples; [2] studies the problem of mining association rules between sets of data items; [7] proposes an attribute-oriented approach to knowledge discovery; [11] develops a visual feedback querying system to support data mining; and [15] includes many interesting studies on various issues in knowledge discovery such as finding functional dependencies between attributes. However, most of these studies are concerned with knowledge discovery on non-spatial data, and the study most relevant to our focus here is [13] which studies spatial data mining. More specifically, [13] proposes a spatial data-dominant knowledge-extraction algorithm and a non-spatial data-dominant one, both of which aim to extract high-level relationships between spatial and non-spatial data. However, both algorithms suffer from the following problems. First, the user or an expert must provide the algorithms with spatial concept hierarchies, which may not be available in many applications. Second, both algorithms conduct their spatial exploration primarily by merging regions at a certain level of the hierarchy to a larger region at a higher level. Thus, the quality of the results produced by both algorithms relies quite crucially on the appropriateness of the hierarchy to the given data. The problem for most applications is that it is very difficult to know *a priori* which hierarchy will be the most appropriate. Discovering this hierarchy may itself be one of the reasons to apply spatial data mining.

To deal with these problems, we explore whether cluster analysis techniques are applicable. Cluster Analysis is a branch of statistics that in the past three decades has been intensely studied and successfully applied to many applications. To the spatial data mining task at hand, the attractiveness of cluster analysis is its ability to find structures or clusters directly from the given data, without relying on any hierarchies. However, cluster analysis has been applied rather unsuccessfully in the past to general data mining and machine learning. The complaints are that cluster analysis algorithms are ineffective and inefficient. Indeed, for cluster analysis algorithms to work effectively, there need to be a natural notion of similarities among the "objects" to be clustered. And traditional cluster analysis algorithms are not designed for large data sets, say more than 2000 objects.

For spatial data mining, our approach here is to apply cluster analysis only on the spatial attributes, for which natural notions of similarities exist (e.g. Euclidean or Manhattan distances). As will be shown in this paper, in this way, cluster analysis techniques are effective for spatial data mining. As for the efficiency concern, we develop our own cluster analysis algorithm, called CLARANS, which is designed for large data sets. More specifically, we will report in this paper:

- the development of CLARANS, which is based on randomized search and is partly motivated by two existing algorithms well-known in cluster analysis, called PAM and CLARA; and
- the development of two spatial mining algorithms SD(CLARANS) and NSD(CLARANS).

Given the nature of spatial data mining, and the fact that CLARANS is based on randomized search, the methodology we have adopted here is one based on experimentation. In particular, we will present:

- experimental results showing that CLARANS is more efficient than the existing algorithms PAM and CLARA; and
- experimental evidence and analysis demonstrating the effectiveness of SD(CLARANS) and NSD(CLARANS) for spatial data mining.

The paper is organized as follows. Section 2 introduces PAM and CLARA. Section 3 presents our clustering algorithm CLARANS, as well as experimental results comparing the performance of CLARANS, PAM and CLARA. Section 4 studies spatial data mining and presents two spatial data mining algorithms, SD(CLARANS) and NSD(CLARANS). Section 5 gives an experimental evaluation on the effectiveness of SD(CLARANS) and NSD(CLARANS) for spatial data mining. Section 6 discusses how SD(CLARANS) and NSD(CLARANS) can assist in further spatial discoveries, and how they can contribute towards the building of a general-purpose and powerful spatial data mining package in the future.

2 Clustering Algorithms based on Partitioning

2.1 Overview

In the past 30 years, cluster analysis has been widely applied to many areas such as medicine (classification of diseases), chemistry (grouping of compounds), social studies (classification of statistical findings), and so on. Its main goal is to identify structures or *clusters* present in the data. While there is no general definition of a cluster, algorithms have been developed to find several kinds of clusters: spherical, linear, drawn-out, etc. Motivated by different kinds of applications, techniques have also been developed to deal with data of various types: binary, nominal and other kinds of discrete variables, continuous variables, similarities, and dissimilarities. See [10, 17] for more detailed discussions and analyses of these issues.

Existing clustering algorithms can be classified into two main categories: *hierarchical* methods and *partitioning* methods. Hierarchical methods are either agglomerative or divisive. Given n objects to be clustered, agglomerative methods begin with n clusters (i.e. all objects are apart). In each step, two clusters are chosen and merged. This process continues until all objects are clustered into one group. On the other hand, divisive methods begin by putting all objects in one cluster. In each step, a cluster is chosen and split up into two. This process continues until n clusters are produced. While hierarchical methods have been successfully applied to many biological applications (e.g. for producing taxonomies of animals and plants [10]), they are well known to suffer from the weakness that they can never undo what was done previously. Once an agglomerative method merges two objects, these objects will always be in one cluster. And once a divisive method separates two objects, these objects will never be re-grouped into the same cluster.

In contrast, given the number k of partitions to be found, a partitioning method tries to find the best k partitions [1] of the n objects. It is very often the case that the k clusters found by a partitioning method are of higher quality (i.e. more similar) than the k clusters produced by a hierarchical method. Because of this property, developing partitioning methods has been

[1] Partitions here are defined in the usual way: each object is assigned to exactly one group.

one of the main focuses of cluster analysis research. Indeed, many partitioning methods have been developed, some based on k-means, some on k-medoid, some on fuzzy analysis, etc. Among them, we have chosen the k-medoid methods as the basis of our algorithm for the following reasons. First, unlike many other partitioning methods, the k-medoid methods are very robust to the existence of outliers (i.e. data points that are very far away from the rest of the data points). Second, clusters found by k-medoid methods do not depend on the order in which the objects are examined. Furthermore, they are invariant with respect to translations and orthogonal transformations of data points. Last but not least, experiments have shown that the k-medoid methods described below can handle very large data sets quite efficiently. See [10] for a more detailed comparison of k-medoid methods with other partitioning methods. In the remainder of this section, we present the two best-known k-medoid methods on which our algorithm is based.

2.2 PAM

PAM (Partitioning Around Medoids) was developed by Kaufman and Rousseeuw [10]. To find k clusters, PAM's approach is to determine a representative object for each cluster. This representative object, called a *medoid*, is meant to be the most centrally located object within the cluster. Once the medoids have been selected, each non-selected object is grouped with the medoid to which it is the most similar. More precisely, if O_j is a non-selected object, and O_i is a (selected) medoid, we say that O_j belongs to the cluster represented by O_i, if $d(O_j, O_i) = min_{O_e} d(O_j, O_e)$, where the notation min_{O_e} denotes the minimum over all medoids O_e, and the notation $d(O_a, O_b)$ denotes the dissimilarity or distance between objects O_a and O_b. All the dissimilarity values are given as inputs to PAM. Finally, the *quality of a clustering* (i.e. the combined quality of the chosen medoids) is measured by the average dissimilarity between an object and the medoid of its cluster.

To find the k medoids, PAM begins with an arbitrary selection of k objects. Then in each step, a swap between a selected object O_i and a non-selected object O_h is made, as long as such a swap would result in an improvement of the quality of the clustering. In particular, to calculate the effect of such a swap between O_i and O_h, PAM computes costs C_{jih} for all non-selected objects O_j. Depending on which of the following cases O_j is in, C_{jih} is defined by one of the equations below.

First Case: suppose O_j currently belongs to the cluster represented by O_i. Furthermore, let O_j be more similar to $O_{j,2}$ than O_h, i.e. $d(O_j, O_h) \geq d(O_j, O_{j,2})$, where $O_{j,2}$ is the second most similar medoid to O_j. Thus, if O_i is replaced by O_h as a medoid, O_j would belong to the cluster represented by $O_{j,2}$. Hence, the cost of the swap as far as O_j is concerned is:

$$C_{jih} = d(O_j, O_{j,2}) - d(O_j, O_i). \quad (1)$$

This equation always gives a non-negative C_{jih}, indicating that there is a non-negative cost incurred in replacing O_i with O_h.

Second Case: O_j currently belongs to the cluster represented by O_i. But this time, O_j is less similar to $O_{j,2}$ than O_h, i.e. $d(O_j, O_h) < d(O_j, O_{j,2})$. Then, if O_i is replaced by O_h, O_j would belong to the cluster represented by O_h. Thus, the cost for O_j is given by:

$$C_{jih} = d(O_j, O_h) - d(O_j, O_i). \quad (2)$$

Unlike in Equation (1), C_{jih} here can be positive or negative, depending on whether O_j is more similar to O_i or to O_h.

Third Case: suppose that O_j currently belongs to a cluster other than the one represented by O_i. Let $O_{j,2}$ be the representative object of that cluster. Furthermore, let O_j be more similar to $O_{j,2}$ than O_h. Then even if O_i is replaced by O_h, O_j would stay in the cluster represented by $O_{j,2}$. Thus, the cost is:

$$C_{jih} \quad = \quad 0. \tag{3}$$

Fourth Case: O_j currently belongs to the cluster represented by $O_{j,2}$. But O_j is less similar to $O_{j,2}$ than O_h. Then replacing O_i with O_h would cause O_j to jump to the cluster of O_h from that of $O_{j,2}$. Thus, the cost is:

$$C_{jih} \quad = \quad d(O_j, O_h) - d(O_j, O_{j,2}), \tag{4}$$

and is always negative. Combining the four cases above, the total cost of replacing O_i with O_h is given by:

$$TC_{ih} \quad = \quad \sum_j C_{jih} \tag{5}$$

We now present Algorithm PAM.

Algorithm PAM

1. Select k representative objects arbitrarily.

2. Compute TC_{ih} for *all* pairs of objects O_i, O_h where O_i is currently selected, and O_h is not.

3. Select the pair O_i, O_h which corresponds to $min_{O_i,O_h}\ TC_{ih}$. If the minimum TC_{ih} is negative, replace O_i with O_h, and go back to Step (2).

4. Otherwise, for each non-selected object, find the most similar representative object. Halt. □

Experimental results show that PAM works satisfactorily for small data sets (e.g. 100 objects in 5 clusters [10]). But it is not efficient in dealing with medium and large data sets. This is not too surprising if we perform a complexity analysis on PAM. In Steps (2) and (3), there are altogether $k(n-k)$ pairs of O_i, O_h. For each pair, computing TC_{ih} requires the examination of $(n-k)$ non-selected objects. Thus, Steps (2) and (3) combined is of $O(k(n-k)^2)$. And this is the complexity of only one iteration. Thus, it is obvious that PAM becomes too costly for large values of n and k. This analysis motivates the development of CLARA.

2.3 CLARA

Designed by Kaufman and Rousseeuw to handle large data sets, CLARA (Clustering LARge Applications) relies on sampling [10]. Instead of finding representative objects for the entire data set, CLARA draws a sample of the data set, applies PAM on the sample, and finds the medoids of the sample. The point is that if the sample is drawn in a sufficiently random way, the medoids of the sample would approximate the medoids of the entire data set. To come up with better approximations, CLARA draws multiple samples and gives the best clustering as the output. Here, for accuracy, the quality of a clustering is measured based on the average dissimilarity of all objects in the *entire* data set, and not only of those objects in the samples. Experiments reported in [10] indicate that 5 samples of size $40 + 2k$ give satisfactory results.

Algorithm CLARA

1. For $i = 1$ to 5, repeat the following steps:
2. Draw a sample of $40+2k$ objects randomly from the entire data set [2], and call Algorithm PAM to find k medoids of the sample.
3. For each object O_j in the entire data set, determine which of the k medoids is the most similar to O_j.
4. Calculate the average dissimilarity of the clustering obtained in the previous step. If this value is less than the current minimum, use this value as the current minimum, and retain the k medoids found in Step (2) as the best set of medoids obtained so far.
5. Return to Step (1) to start the next iteration. □

Complementary to PAM, CLARA performs satisfactorily for large data sets (e.g. 1000 objects in 10 clusters). Recall from Section 2.2 that each iteration of PAM is of $O(k(n-k)^2)$. But for CLARA, by applying PAM just to the samples, each iteration is of $O(k(40+k)^2 + k(n-k))$. This explains why CLARA is more efficient than PAM for large values of n.

3 A Clustering Algorithm based on Randomized Search

In this section, we will present our clustering algorithm – CLARANS (Clustering Large Applications based on RANdomized Search). We will first introduce CLARANS by giving a graph abstraction of it. Then after describing the details of the algorithm, we will present experimental results showing that CLARANS outperforms CLARA and PAM in terms of both efficiency and effectiveness. In the next section, we will show how CLARANS can be used to provide effective spatial data mining.

3.1 Motivation of CLARANS: a Graph Abstraction

Given n objects, the process described above of finding k medoids can be viewed abstractly as searching through a certain graph. In this graph, denoted by $G_{n,k}$, a node is represented by a set of k objects $\{O_{m_1}, \ldots, O_{m_k}\}$, intuitively indicating that $O_{m_1}, \ldots, O_{m_k}$ are the selected medoids. The set of nodes in the graph is the set $\{ \{O_{m_1}, \ldots, O_{m_k}\} \mid O_{m_1}, \ldots, O_{m_k}$ are objects in the data set$\}$.

[2] [10] reports a useful heuristic to draw samples. Apart from the first sample, subsequent samples include the best set of medoids found so far. In other words, apart from the first iteration, subsequent iterations draw $40 + k$ objects to add on to the best k medoids.

Two nodes are neighbors (i.e. connected by an arc) if their sets differ by only one object. More formally, two nodes $S_1 = \{O_{m_1}, \ldots, O_{m_k}\}$ and $S_2 = \{O_{w_1}, \ldots, O_{w_k}\}$ are neighbors if and only if the cardinality of the intersection of S_1, S_2 is $k-1$, i.e. $|S_1 \cap S_2| = k-1$. It is easy to see that each node has $k(n-k)$ neighbors. Since a node represents a collection of k medoids, each node corresponds to a clustering. Thus, each node can be assigned a cost that is defined to be the total dissimilarity between every object and the medoid of its cluster. It is not difficult to see that if objects O_i, O_h are the differences between neighbors S_1 and S_2 (i.e. $O_i, O_h \notin S_1 \cap S_2$, but $O_i \in S_1$ and $O_h \in S_2$), the cost differential between the two neighbors is exactly given by T_{ih} defined in Equation (5).

By now, it is obvious that PAM can be viewed as a search for a minimum on the graph $G_{n,k}$. At each step, all the neighbors of the current node are examined. The current node is then replaced by the neighbor with the deepest descent in costs. And the search continues until a minimum is obtained. For large values of n and k (like $n = 1000$ and $k = 10$), examining all $k(n-k)$ neighbors of a node is time consuming. This accounts for the inefficiency of PAM for large data sets.

On the other hand, CLARA tries to examine fewer neighbors and restricts the search on subgraphs that are much smaller in size than the original graph $G_{n,k}$. However, the problem is that the subgraphs examined are defined entirely by the objects in the samples. Let Sa be the set of objects in a sample. The subgraph $G_{Sa,k}$ consists of all the nodes that are subsets (of cardinalities k) of Sa. Even though CLARA thoroughly examines $G_{Sa,k}$ via PAM, the trouble is that the search is fully confined within $G_{Sa,k}$. If M is the minimum node in the original graph $G_{n,k}$, and if M is not included in $G_{Sa,k}$, M will never be found in the search of $G_{Sa,k}$, regardless of how thorough the search is. To atone for this deficiency, many, many samples would need to be collected and processed.

Like CLARA, our algorithm CLARANS does not check every neighbor of a node. But unlike CLARA, it does not restrict its search to a particular subgraph. In fact, it searches the original graph $G_{n,k}$. One key difference between CLARANS and PAM is that the former only checks a sample of the neighbors of a node. But unlike CLARA, each sample is drawn dynamically in the sense that no nodes corresponding to particular objects are eliminated outright. In other words, while CLARA draws a sample of *nodes* at the beginning of a search, CLARANS draws a sample of *neighbors* in each step of a search. This has the benefit of not confining a search to a localized area. As will be shown in Section 3.3, a search by CLARANS gives higher quality clusterings than CLARA, and CLARANS requires a very small number of searches. We now present the details of Algorithm CLARANS.

3.2 CLARANS

Algorithm CLARANS

1. Input parameters *numlocal* and *maxneighbor*. Initialize i to 1, and *mincost* to a large number.

2. Set *current* to an arbitrary node in $G_{n,k}$.

3. Set j to 1.

4. Consider a random neighbor S of *current*, and based on Equation (5), calculate the cost differential of the two nodes.

5. If S has a lower cost, set *current* to S, and go to Step (3).

6. Otherwise, increment j by 1. If $j \leq maxneighbor$, go to Step (4).

7. Otherwise, when $j > maxneighbor$, compare the cost of *current* with *mincost*. If the former is less than *mincost*, set *mincost* to the cost of *current*, and set *bestnode* to *current*.

8. Increment i by 1. If $i > numlocal$, output *bestnode* and halt. Otherwise, go to Step (2). □

Steps (3) to (6) above search for nodes with progressively lower costs. But if the current node has already been compared with the maximum number of the neighbors of the node (specified by *maxneighbor*) and is still of the lowest cost, the current node is declared to be a "local" minimum. Then in Step (7), the cost of this local minimum is compared with the lowest cost obtained so far. The lower of the two costs above is stored in *mincost*. Algorithm CLARANS then repeats to search for other local minima, until *numlocal* of them have been found.

As shown above, CLARANS has two parameters: the maximum number of neighbors examined (*maxneighbor*), and the number of local minima obtained (*numlocal*). The higher the value of *maxneighbor*, the closer is CLARANS to PAM, and the longer is each search of a local minima. But the quality of such a local minima is higher, and fewer local minima needs to be obtained. Like many applications of randomized search [8, 9], we rely on experiments to determine the appropriate values of these parameters.

3.3 Experimental Results: Tuning CLARANS

3.3.1 Details of Experiments

To observe the behavior and efficiency of CLARANS, we ran CLARANS with generated data sets whose clusters are known. For better generality, we used two kinds of clusters with quite opposite characteristics. The first kind of clusters is rectangular, and the objects within each cluster are randomly generated. More specifically, if such a data set of say 3000 objects in 20 clusters is needed, we first generated 20 "bounding boxes" of the same size. To make the clusters less clear-cut, the north-east corner of the i-th box and the south-west corner of $(i+1)$-th box touch. Since for our application of spatial data mining, CLARANS is used to cluster spatial coordinates, objects in our experiments here are pairs of $x-, y-$ coordinates. For each bounding box, we then randomly generated 150 pairs of coordinates that fall within the box. Similarly, we generated data sets of the same kind but of varying numbers of objects and clusters. In the figures below, the symbol rn-k (e.g. r3000-20) represents a data set of this kind with n points in k clusters.

Unlike the first kind, the second kind of clusters we experimented with does not contain random points. Rather, points within a cluster are ordered in a triangle. For example, the points with coordinates (0,0), (1,0), (0,1), (2,0), (1,1), and (0,2) form such a triangular cluster of size 6. To produce a cluster next to the previous one, we used a translation of the origin (e.g. the points (10,10), (11,10), (10,11), (12,10), (11,11), and (10,12)). In the figures below, the symbol tn-k (e.g. t3000-20) represents a data set organized in this way with n points in k clusters.

All the experiments reported here were carried out in a time-sharing SPARC-LX workstation. Because of the random nature of CLARANS, all the figures concerning CLARANS are average figures obtained by running the same experiment 10 times (with different seeds of the random number generator).

3.3.2 Determining the Maximum Number of Neighbors

In the first series of experiments, we applied CLARANS with the parameter $maxneighbor =$ 250, 500, 750, 1000, and 10000 on the data sets rn-k and tn-k, where n varies from 100 to 3000 and k varies from 5 to 20. To save space, we only summarize the two major findings that lead to further experiments:

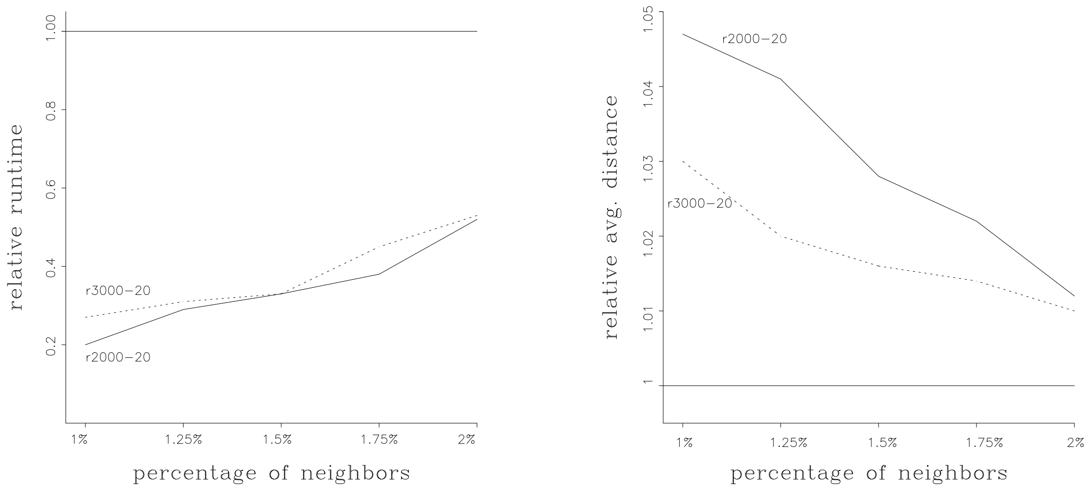
- When the maximum number of neighbors $maxneighbor$ is set to 10000, the quality of the clustering produced by CLARANS is effectively the same as the quality of the clustering produced by PAM (i.e. $maxneighbor = k(n-k)$). While we will explain this phenomenon very shortly, we use the results for $maxneighbor = 10000$ as a yardstick for evaluating other (smaller) values of $maxneighbor$. More specifically, the runtime values of the first graph and the average distance values (i.e. quality of a clustering) of the second graph in Figure 1 below are normalized by those produced by setting $maxneighbor = 10000$. This explains the two horizontal lines at $y-$ value $= 1$ in both graphs.

- As expected, a lower value of $maxneighbor$ produces a lower quality clustering. A question we ask is then how small can the value of $maxneighbor$ be before the quality of the clustering becomes unacceptable. From the first series of experiments, we find out that these critical values seem to be proportional to the value $k(n-k)$. This motivates us to conduct another series of experiments with the following enhanced formula for determining the value of $maxneighbor$:

  if $k(n-k) \leq minmaxneighbor$ then $maxneighbor = k(n-k)$; otherwise, $maxneighbor$ equals the the larger value between $p\%$ of $k(n-k)$ and $minmaxneighbor$.

The above formula allows CLARANS to examine all the neighbors as long as the total number of neighbors is below the threshold $minmaxneighbor$. Beyond the threshold, the percentage of neighbors examined gradually drops from 100% to a minimum of $p\%$. The two graphs in Figure 1 show the relative runtime and quality of CLARANS with $minmaxneighbor = 250$ and p varying from 1% to 2%. While the graphs only show the results of the rectangular data sets with 2000 and 3000 points in 20 clusters, these graphs are representative, as the appearances of the graphs for small and medium data sets, and for the triangular data sets are very similar.

Figure 1(a) shows that the lower the value of p, the smaller the amount of runtime CLARANS requires. And as expected, Figure 1(b) shows that a lower value of p produces a lower quality clustering (i.e. higher (relative) average distance). But the very amazing



(a) relative efficiency

(b) relative quality

Figure 1: Determining the Maximum Number of Neighbors

feature shown in Figure 1(b) is that the quality is still within 5% from that produced by setting $maxneighbor = 10000$ (or by PAM). As an example, if a maximum of $p = 1.5\%$ of neighbors are examined, the quality is within 3%, while the runtime is only 40%. What that means is that examining 98.5% more neighbors, while taking much longer, only produces marginally better results. This is consistent with our earlier statement that CLARANS with $maxneigh = 10000$ gives the same quality as PAM, which is effectively the same as setting $maxneighbor = k(n-k) = 20(3000\text{-}20) = 59600$.

The reason why so few neighbors need to be examined to get good quality clusterings can be best illustrated by the graph abstraction presented in Section 3.1. Recall that each node has $k(n-k)$ neighbors, making the graph very highly connected. Consider two neighbors S_1, S_2 of the current node, and assume that S_1 constitutes a path leading to a certain minimum node S. Even if S_1 is missed by not being examined, and S_2 becomes the current node, there are still numerous paths that connect S_2 to S. Of course, if all such paths are not strictly downward (in cost) paths, and may include "hills" along the way, S will never be reached from S_2. But our experiments seem to indicate that the chance that a hill exists on *every* path is very small.

To keep a good balance between runtime and quality, we believe that a p value between 1.25% and 1.5% is very reasonable. For all our later experiments with CLARANS, we chose the value $p = 1.25\%$.

3.3.3 Determining the Number of Local Minima

Recall that Algorithm CLARANS has two parameters: *maxneighbor* and *numlocal*. Having dealt with the former, here we focus on determining the value of *numlocal*. In this series

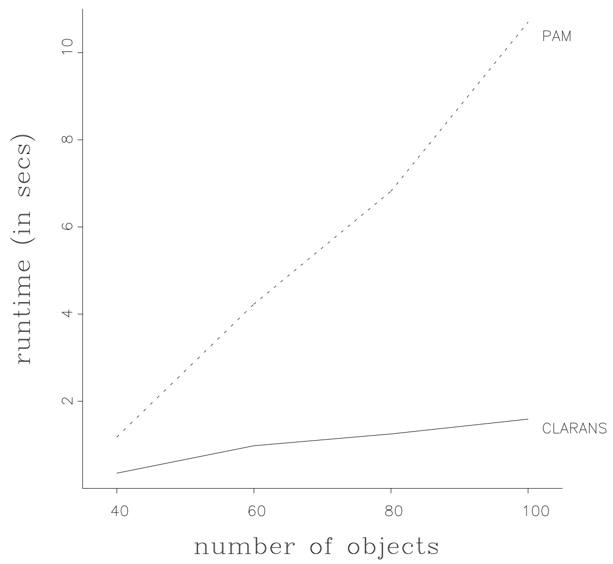


Figure 2: Efficiency: CLARANS vs PAM

of experiments, we ran CLARANS with $numlocal = 1, \ldots, 5$ on data sets rn-k and tn-k for small, medium and large values of n and k. For each run, we recorded the runtime and the quality of the clustering. The following table (which is typical of all data sets) shows the relative runtime and quality for the data set r2000-20. Here all the values are normalized by those with $numlocal = 5$.

$numlocal$	1	2	3	4	5
relative runtime	0.19	0.38	0.6	0.78	1
relative average distance	1.029	1.009	1	1	1

As expected, the runtimes are proportional to the number of local minima obtained. As for the relative quality, there is an improvement from $numlocal = 1$ to $numlocal = 2$. Performing a second search for a local minimum seems to reduce the impact of "unlucky" randomness that may occur in just one search. However, setting $numlocal$ larger than 2 is not cost-effective, as there is little increase in quality. This is an indication that a typical local minimum is of very high quality. We believe that this phenomenon is largely due to, as discussed previously, the peculiar nature of the abstract graph representing the operations of CLARANS. For all our later experiments with CLARANS, we used the version that finds two local minima.

3.4 Experimental Results: CLARANS vs PAM

In this series of experiments, we compared CLARANS with PAM. As discussed in Section 3.3.2, for large and medium data sets, it is obvious that CLARANS, while producing clusterings of very comparable quality, is much more efficient than PAM. Thus, our focus here was to compare the two algorithms on small data sets. We applied both algorithms to

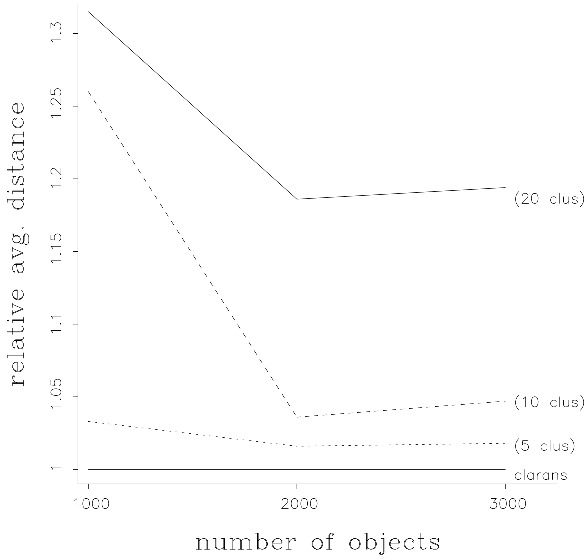


Figure 3: Relative Quality: Same Time for CLARANS and CLARA

data sets with 40, 60, 80 and 100 points in 5 clusters. Figure 2 shows the runtime taken by both algorithms. Note that for all those data sets, the clusterings produced by both algorithms are of the same quality (i.e. same average distance). Thus, the difference between the two algorithms is determined by their efficiency. It is evident from Figure 2 that even for small data sets, CLARANS outperforms PAM significantly. As expected, the performance gap between the two algorithms grows, as the data set increases in size.

3.5 Experimental Results: CLARANS vs CLARA

In this series of experiments, we compared CLARANS with CLARA. As discussed in Section 2.3, CLARA is not designed for small data sets. Thus, we ran this set of experiments on data sets whose number of objects exceeds 100. And the objects were organized in different number of clusters, as well as in the two types of clusters described in Section 3.3.1.

When we conducted this series of experiments running CLARA and CLARANS as presented earlier, CLARANS is always able to find clusterings of better quality than those found by CLARA. However, in some cases, CLARA may take much less time than CLARANS. Thus, we wondered whether CLARA would produce clusterings of the same quality, if it was given the same amount of time. This leads to the next series of experiments in which we gave both CLARANS and CLARA the same amount of time. Figure 3 shows the quality of the clusterings produced by CLARA, normalized by the corresponding value produced by CLARANS.

Given the same amount of time, CLARANS clearly outperforms CLARA in all cases. The gap between CLARANS and CLARA increases from 4% when k, the number of clusters, is 5 to 20% when k is 20. This widening of the gap as k increases can be best explained by looking at the complexity analyses of CLARA and CLARANS. Recall from Section 2.3 that each iteration of CLARA is of $O(k^3+nk)$. On the other hand, recall from Section 3.3.2 that the cost of CLARANS is basically linearly proportional to the number of objects [3]. Thus, an increase in k imposes a much larger cost on CLARA than on CLARANS.

The above complexity comparison also explains why for a fixed number of clusters, the higher the number of objects, the narrower the gap between CLARANS and CLARA is. For example, when the number of objects is 1000, the gap is as high as 30%. The gap drops to around 20% as the number of object increases to 2000. Since each iteration of CLARA is of $O(k^3+nk)$, the first term k^3 dominates the second term. Thus, for a fixed k, CLARA is relatively less sensitive to an increase in n. On the other hand, since the cost of CLARANS is roughly linearly proportional to n, an increase in n imposes a larger cost on CLARANS than on CLARA. This explains why for a fixed k, the gap narrows as the number of objects increases. Nonetheless, the bottom-line shown in Figure 3 is that CLARANS beats CLARA in all cases.

In sum, we have presented experimental evidence showing that CLARANS is more efficient than PAM and CLARA for small and large data sets. Our experimental results for medium data sets (not included here) lead to the same conclusion.

4 Spatial Data Mining based on Clustering Algorithms

In this section, we will present two spatial data mining algorithms that use clustering methods. In the next section, we will show experimental results on the effectiveness of these algorithms.

4.1 Spatial Dominant Approach: SD(CLARANS)

There are different approaches to spatial data mining. The kind of spatial data mining considered in this paper assumes that a spatial database consists of both spatial and non-spatial attributes, and that non-spatial attributes are stored in relations [3, 12, 16]. The general approach here is to use clustering algorithms to deal with the spatial attributes, and use other learning tools to take care of the non-spatial counterparts.

DBLEARN is the tool we have chosen for mining non-spatial attributes [7]. It takes as inputs relational data, generalization hierarchies for attributes, and a learning query specifying the focus of the mining task to be carried out. From a learning request, DBLEARN first extracts a set of relevant tuples via SQL queries. Then based on the generalization hierarchies of attributes, it iteratively generalizes the tuples. For example, suppose the tuples relevant to a certain learning query have attributes $\langle major, ethnicgroup\rangle$. Further assume

[3] There is a random aspect and a non-random aspect to the execution of CLARANS. The non-random aspect corresponds to the part that finds the cost differential between the current node and its neighbor. This part, as defined in Equation (5) is linearly proportional to the number of objects in the data set. On the other hand, the random aspect corresponds to the part that searches for a local minimum. As the values to plot the graphs are average values of 10 runs, which have the effect of reducing the influence of the random aspect, the runtimes of CLARANS used in our graphs are largely dominated by the non-random aspect of CLARANS.

that the generalization hierarchy for *ethnicgroup* has Indian and Chinese generalized to Asians. Then a generalization operation on the attribute *ethnicgroup* causes all tuples of the form $\langle m, Indian \rangle$ and $\langle m, Chinese \rangle$ to be merged to the tuple $\langle m, Asians \rangle$. This merging has the effect of reducing the number of remaining (generalized) tuples. As described in [7], each tuple has a system-defined attribute called *count* which keeps track of the number of original tuples (as stored in the relational database) that are represented by the current (generalized) tuple. This attribute enables DBLEARN to output such statistical statements as 8% of all students majoring in Sociology are Asians. In general, a generalization hierarchy may have multiple levels (e.g. Asians further generalized to non-Canadians), and a learning query may require more than one generalization operation before the final number of generalized tuples drops below a certain threshold [4]. At the end, statements such as 90% of all Arts students are Canadians may be returned as the findings of the learning query.

Having outlined what DBLEARN does, the specific issue we address here is how to extend DBLEARN to deal with spatial attributes. In particular, we will present two ways to combine clustering algorithms with DBLEARN. The algorithm below, called SD(CLARANS), combines CLARANS and DBLEARN in a *spatial dominant* fashion. That is, spatial clustering is performed first, followed by non-spatial generalization of every cluster.

Algorithm SD(CLARANS)

1. Given a learning request, find the initial set of relevant tuples by the appropriate SQL queries.

2. Apply CLARANS to the spatial attributes and find the most natural number k_{nat} of clusters.

3. For each of the k_{nat} clusters obtained above,

   (a) collect the non-spatial components of the tuples included in the current cluster.

   (b) Apply DBLEARN to this collection of the non-spatial components. □

Similarly, Algorithms SD(PAM) and SD(CLARA) can be obtained. But as shown in the last section that CLARANS is more efficient than PAM and CLARA, the experimental evaluation to be reported in Section 5 only considers SD(CLARANS).

4.2 Determining k_{nat} for CLARANS

Step (2) of Algorithm SD(CLARANS) tries to find k_{nat} clusters, where k_{nat} is the most natural number of clusters for the given data set. However, recall that CLARANS and all partitioning algorithms require the number k of clusters to be given as input. Thus, an immediate question to ask is whether SD(CLARANS) knows beforehand what k_{nat} is and can then simply pass the value of k_{nat} to CLARANS. The unfortunate answer is *no*. In fact, determining k_{nat} is one of the most difficult problems in cluster analysis, for which no unique solution exists. For SD(CLARANS), we adopt the heuristics of computing the *silhouette*

[4] Apart from generalization operations (also known as hierarchy ascension operations), DBLEARN, in its full form, may sometimes choose to drop an attribute, if generalizing such an attribute would produce uninteresting results (e.g. generalizing names of students).

coefficients, first developed by Kaufman and Rousseeuw [10]. (For a survey of alternative criteria, see [14].) For space considerations, we do not include the formulas for computing silhouettes, and will only concentrate on how we use silhouettes in our algorithms.

Intuitively, the silhouette of an object O_j, a dimensionless quantity varying between -1 and 1, indicates how much O_j truly belongs to the cluster to which O_j is classified. The closer the value is to 1, the higher the degree O_j belongs to its cluster. The silhouette width of a cluster is the average silhouette of all objects in the cluster. Based on extensive experimentation, [10] proposes the following interpretation of the silhouette width of a cluster:

silhouette width	interpretation
0.71 – 1	the cluster is strong
0.51 – 0.7	the cluster is reasonable
0.26 – 0.5	the cluster is weak and could be artificial
≤ 0.25	no cluster can be found

For a given number $k \geq 2$ of clusters, the silhouette coefficient for k is the average silhouette widths of the k clusters. Notice that the silhouette coefficient does not necessarily decrease monotonically as k increases [5]. If the value k is too small, some distinct clusters are incorrectly grouped together, leading to a small silhouette width. On the other hand, if k is too large, some natural clusters may be artificially split, again leading to a small silhouette width. Thus, the most natural k is the one whose silhouette coefficient is the highest. However, our experiments on spatial data mining show that just using the highest silhouette coefficient may not lead to intuitive results. For example, some clusters may not have reasonable structures, i.e. widths ≤ 0.5. Thus, we use the following heuristics to determine the value k_{nat} for SD(CLARANS).

Heuristics for Determining k_{nat}

1. Find the value k with the highest silhouette coefficient.

2. If all the k clusters have silhouette widths ≥ 0.51, $k_{nat} = k$, and halt.

3. Otherwise, remove the objects in those clusters whose silhouette widths are below 0.5, provided that the total number of objects removed so far is less than a threshold (e.g. 25% of the total number of objects). The objects removed are considered to be outliers or noises. Go back to Step (1) for the new data set without the outliers.

4. If in Step (3), the number of outliers to be removed exceeds the threshold, simply set $k_{nat} = 1$, indicating in effect that no clustering is reasonable. □

In Section 5, we will see the usefulness of the heuristics.

As we have completed the description of SD(CLARANS), it is a good time to compare SD(CLARANS) with an earlier approach reported in [13] whose goal is to enhance

[5] However, this is not the case for the average dissimilarity of an object from its medoid. The larger the value of k, the smaller the average dissimilarity is. This explains why average dissimilarity is only suitable as a measurement criterion for fixed k, but is otherwise not suitable to be used to compare the quality of clusterings produced by different k values.

DBLEARN with spatial learning capabilities. One of the two proposed approaches there is to first perform spatial generalizations, and then to use DBLEARN to conduct non-spatial generalizations. The fundamental difference between SD(CLARANS) and that algorithm in [13] is that a user of the latter must give a priori as input generalization hierarchies for spatial attributes. The problem is that without prior analysis, it is almost impossible to guarantee that the given hierarchies are suitable for the given data set. (This may in fact be one of the discoveries to be found out by the spatial data mining task!) For example, suppose a spatial data mining request is to be performed on all the expensive houses in Greater Vancouver. A default spatial hierarchy to use may be the one that generalizes streets to communities and then to cities. However, if some of the expensive houses are spatially located along something (such as a river, the bottom of a range of mountains, etc.) that runs through many communities and cities, then the default spatial hierarchy would be very ineffective, generating such general statements as that the expensive houses are more or less scattered in all the cities in Greater Vancouver.

Far extending the capability of the algorithm in [13], SD(CLARANS) finds the clusters *directly* from the given data. To a certain extent, the clustering algorithm, CLARANS in this case, can be viewed as computing the spatial generalization hierarchy dynamically. The result of such computation, combined with the above heuristics to find k_{nat}, precisely finds the clusters (if indeed exist in the data set) in terms of the $x-$ and $y-$ coordinates of the points, and not confined by any hierarchies specified a priori. For the expensive houses example discussed above, SD(CLARANS) could directly identify clusters along the river or the bottom of the mountain range, and could lead to such statements as 80% of all mansions have either a mountain or a river view. In Section 5, we will see how well our spatial data mining algorithms can handle a data set arguably more complex than the example discussed here.

4.3 Non-Spatial Dominant Approach: NSD(CLARANS)

To a large extent, spatial dominant algorithms, such as SD(CLARANS), can be viewed as focusing asymmetrically on discovering non-spatial characterizations of spatial clusters. Non-spatial dominant algorithms, on the other hand, focus on discovering spatial clusters existing in groups of non-spatial data items. For example, these algorithms may find interesting discoveries based on the spatial clustering or distribution of a certain type of houses. More specifically, unlike spatial dominant algorithms, non-spatial dominant algorithms first apply non-spatial generalizations, followed by spatial clustering. The following algorithm, NSD(CLARANS), uses DBLEARN and CLARANS to perform data mining on non-spatial and spatial attributes respectively.

Algorithm NSD(CLARANS)

1. Given a learning request, find the initial set of relevant tuples by the appropriate SQL queries.

2. Apply DBLEARN to the non-spatial attributes, until the final number of generalized tuples fall below a certain threshold (cf. Section 4.1).

3. For each generalized tuple obtained above,

(a) collect the spatial components of the tuples represented by the current generalized tuple.

(b) Apply CLARANS and the heuristics presented above to find the most natural number k_{nat} of clusters.

4. For all the clusters obtained above, check if there are clusters that intersect or overlap. If exist, such clusters can be merged. This in turn causes the corresponding generalized tuples to be combined. □

Recall from the previous section on clustering algorithms that for a given data set, clusters do not overlap or intersect. This is why SD(CLARANS) does not include a step analogous to Step (4) above. However, for NSD(CLARANS) (and other non-spatial dominant algorithms such as NSD(PAM)), clusters obtained for different generalized tuples can overlap or intersect. In that case, opportunities arise for further generalization of spatial and non-spatial data. This is the purpose of Step (4) above. In the following, we present experimental results evaluating the effectiveness of NSD(CLARANS), as well as SD(CLARANS).

5 Evaluation of SD(CLARANS) and NSD(CLARANS)

5.1 A Real Estate Data Set

One way to evaluate the effectiveness of a data mining algorithm is to apply it to a real data set and see what it finds. But sometimes it may be difficult to judge the quality of the findings, without knowing a priori what the algorithm is supposed to find. Thus, to evaluate our algorithms, we generated a data set that honors several rules applicable to the 2500 expensive housing units in Vancouver. These rules, very close to reality to the best of our knowledge, are as follows:

A. house type, price and size:

1. If the house type is mansion, the price falls within the range [1500K,3500K], and the size within the range [6000,10000] square feet.
2. If the house type is single-house, the price and size ranges are [800K,1500K] and [3000,7000].
3. If the house type is condo, the price and size ranges are [300K,800K] and [1000,2500]. For simplicity, we assumed uniform distributions within all the ranges.

B. distribution:

1. There are 1200 condos uniformly distributed in the Vancouver downtown area – the rectangular region at the top of Figure 4. From now on, this region will be referred to as Area B1.
2. Along Marine Drive, there are about 320 mansions and about 80 single-houses – the stripe at the bottom left-hand corner of Figure 4. This area will be referred to as Area B2.

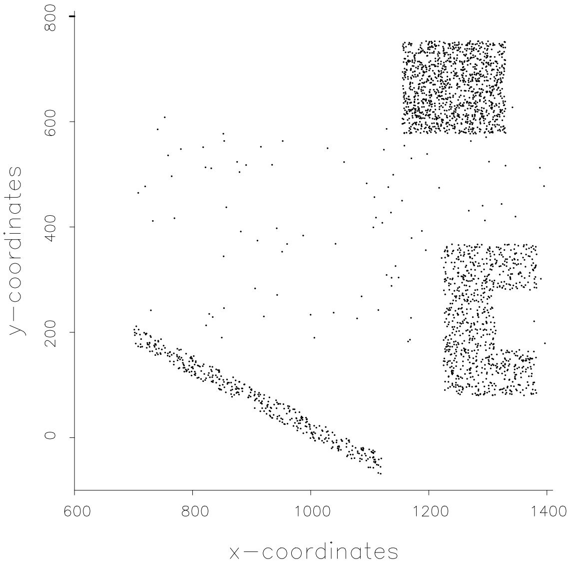


Figure 4: Spatial Distribution of the 2500 Housing Units

3. Around Queen Elizabeth Park, there are 800 single-houses – the polygonal area at the bottom right-hand corner of Figure 4. This area will be referred to as Area B3.
4. Finally, to complicate the situation, there are 100 single-houses uniformly distributed in the rest of Vancouver.

5.2 Effectiveness of SD(CLARANS)

Based on the heuristics presented in Section 4.2, Step (2) of SD(CLARANS) appropriately sets the value of k_{nat} to 3. The silhouette coefficient for $k_{nat} = 3$ is 0.7, indicating that all 3 clusters are quite strong. Thus, Steps (3) and (4) of the heuristics are not needed in this case. After computing k_{nat}, it takes CLARANS about 25 seconds to identify the 3 clusters (in a time-sharing SPARC-LX workstation environment). The first cluster contains 832 units all single-houses, 800 of which are those in Area B3 defined in Section 5.1. For this cluster, DBLEARN in Step (3) of SD(CLARANS) correctly finds the price and size ranges to be [800K,1500K] and [3000,7000]. It also reveals that the prices and sizes are more or less uniformly distributed.

The second cluster contains 1235 units, 1200 of which are condos, and the remainders single-houses. It contains all the units in Area B1 introduced in Section 5.1. For this cluster, DBLEARN finds the condo prices and sizes uniformly distributed within the ranges [300K,800K] and [1000,2500] respectively. It also discovers that the single-house prices and sizes fall within [800K,1500K] and [3000,7000].

The third cluster contains 431 units, 320 of which are mansions, and the remainders single-houses. This cluster includes all the units along the stripe Area B2. For this cluster, DBLEARN finds the mansion prices and sizes uniformly distributed within the ranges [1500K,3500K] and [6000,10000]. As for the single-houses in the cluster, DBLEARN again finds the right ranges.

In sum, SD(CLARANS) is very effective. This is due primarily to the clusters found by CLARANS, even in the presence of outliers (cf. B.4 of Section 5.1). Once the appropriate clusters are found, DBLEARN easily identifies the non-spatial patterns. Thus, CLARANS and DBLEARN together enable SD(CLARANS) to successfully discover all the rules described in Section 5.1 that it is supposed to find.

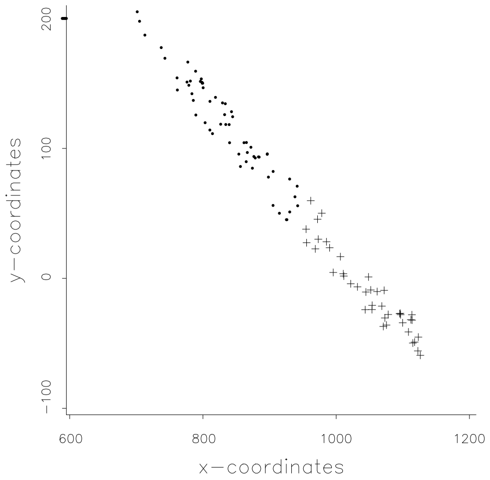
5.3 Effectiveness of NSD(CLARANS)

In Step (2) of NSD(CLARANS), DBLEARN finds 12 generalized tuples, 4 for each type of housing units. Let us first consider the 4 generalized tuples for mansions. The 4 tuples represent respectively mansions in the following categories: a) price in [1500K,2600K], size in [6000,8500]; b) price in [1500K,2600K], size in [8500,10000]; c) price in [2600K,3500K], size in [6000,8500]; and d) price in [2600K,3500K], size in [8500,10000]. The 4 graphs in Figure 5 show the spatial distributions of the mansions in the four categories. When CLARANS is applied to the points shown in each of the 4 graphs, 2 clusters are found in each case. The silhouette coefficients for $k_{nat} = 2$ vary from 0.62 to 0.65. In each graph, points in the two clusters are represented by either dots or +. As shown quite obviously in Figure 5, when Step (4) of NSD(CLARANS) is executed, all 4 clusters represented by dots overlap. These clusters are merged into one larger region. Similarly, all 4 clusters represented by + are merged into another region. Furthermore, these two regions intersect, and are merged into an even bigger region, which is now identical to the stripe Area B2 in Figure 4. Last but not least, these merges of clusters and regions cause the 4 generalized tuples to be combined as well. As a result, NSD(CLARANS) finds out that all mansions are located in the stripe area, and have prices and sizes in the ranges [1500K,3500K] and [6000,10000].

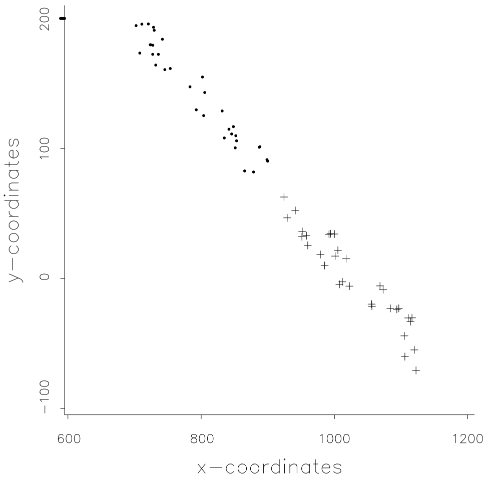
The 4 tuples for condos correspond respectively to the following categories: a) price in [300K,600K], size in [1000,1800]; b) price in [300K,600K], size in [1800,2500]; c) price in [600K,800K], size in [1000,1800]; and d) price in [600K,800K], size in [1800,2500]. The processing of these tuples is very similar to the processing of those for mansions above. The only difference is that for all 4 tuples, no cluster is found [6], i.e. k_{nat} set to 1 in Step (4) of the heuristics in Section 4.2. Thus, in the final step of NSD(CLARANS), all 4 regions/clusters, which overlap, are merged into an area that coincides precisely with Area B1 Figure 4. Consequently, NSD(CLARANS) discovers that all (expensive) condos are located in the Vancouver downtown area, and have prices and sizes in the ranges [300K,800K] and [1000,2500].

The processing of single-houses is the most complicated. The 4 tuples correspond to the categories: a) price in [1200K,1500K], size in [3000,5500]; b) price in [1200K,1500K], size in [5500,7000]; c) price in [800K,1200K], size in [3000,5500]; and d) price in [800K,1200K], size in [5500,7000]. When CLARANS is applied to the houses in the category a) (as shown in Figure 6(a)), the highest silhouette coefficient is found when the number of clusters is 4.

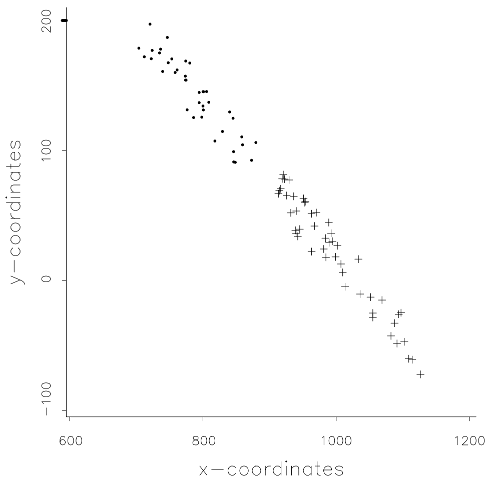
[6] 25% is the threshold used in Step (3) of the heuristics in Section 4.2.



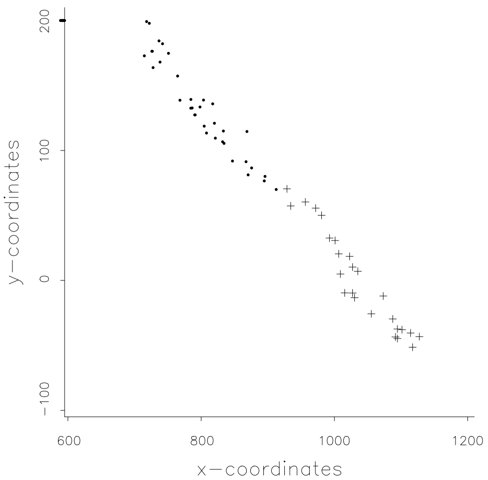
(a) tuple 1



(b) tuple 2

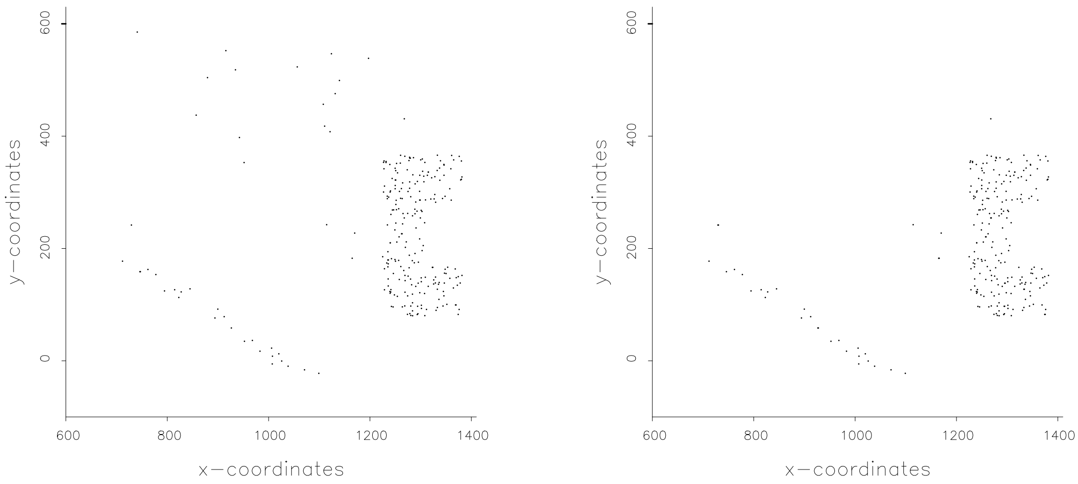


(c) tuple 3



(d) tuple 4

Figure 5: Clusters for the 4 Generalized tuples for Mansions



(a) before removing outliers

(b) after removing outliers

Figure 6: Spatial Distributions for Category a) of Single-houses

However, even though the silhouette coefficient is above 0.5, the silhouette widths of two of the clusters are below 0.5. Thus, Step (3) of the heuristics in Section 4.2 is invoked. As a result, 15 out of the original 253 points are removed. Figure 6(b) shows the spatial distribution of this new collection of points after the outliers are removed. For this new collection, two clusters are identified: i) along the stripe Area B2 in Figure 4, and ii) around Area B3 in Figure 4.

The clustering for category d) of single-houses is very similar to the one described above. Again, outliers need to be removed. At the end, 2 clusters are found, which are identical to the ones listed i) and ii) above.

As for categories b) and c) of single-houses, the result is slightly different. In both cases, because single-houses are quite sparsely located along the stripe area (i.e. the cluster listed i) above), so as to obtain acceptable k_{nat} values, some of the houses along that area are removed as outliers. Consequently, for both categories, 2 clusters are identified: again along Area B2 and around Area B3. The only difference between here and the situation for categories a) and d) is that the clusters along the stripe area are smaller in sizes than expected, because of outliers removal.

After applying CLARANS to all four categories/tuples of single-houses, NSD(CLARANS) in Step (4) merges overlapping or intersecting clusters. As a result, NSD(CLARANS) discovers 2 clusters of single-houses, identical to the ones listed i) and ii) above. While the total number of units in cluster ii) is as expected, the total number of units in cluster i) is less. Again, this is due to the removal of outliers. Furthermore, NSD(CLARANS) correctly identifies the price and size ranges for single-houses to be [800K,1500K] and [3000,7000].

5.4 Summary

With respect to the rules listed in Section 5.1, both SD(CLARANS) and NSD(CLARANS) find most of what they are supposed to find. In terms of performance and effectiveness, SD(CLARANS) has the edge. As discussed earlier, this is due to CLARANS' success in identifying the clusters right away. On the other hand, in NSD(CLARANS), performing non-spatial generalizations divides the entire set of points into different groups/tuples. This may have the effect of breaking down the tightness of some clusters. Outliers removal may then be needed to extract reasonable clusters from each group. This procedure, as we have seen, may weaken the eventual findings and takes more time. Finally, merging overlapping and intersecting clusters can also be costly.

However, to be fair with NSD(CLARANS), the rules described in Section 5.1 are more favorable to SD(CLARANS). There is a strong emphasis on finding out non-spatial characterizations of spatial clusters, which is the focus of spatial dominant algorithms. In contrast, a non-spatial dominant algorithm focuses more on finding spatial clusters within groups of data items that have been generalized non-spatially. For example, if the spatial distribution of single-houses is primarily determined by their price and size categories, then NSD(CLARANS) could be more effective than SD(CLARANS).

6 Discussions

6.1 Exploring Spatial Relationships

Thus far, we have shown that clustering algorithms, such as CLARANS, are very promising and effective for spatial data mining. But we believe that there is an extra dimension a clustering algorithm can provide. As discussed in Section 4.2, a clustering algorithm does not require any spatial generalization hierarchy to be given, and directly discovers the groups/clusters that are the most appropriate to the given data. In other words, clustering can provide very tight spatial characterizations of the groups. The tightness and specificity of the characterizations provide opportunities for exploring spatial relationships that may exist between the clusters and other interesting objects.

For example, as shown in Section 5.2, SD(CLARANS) finds 3 clusters of expensive housing units (cf. Figure 4). Those 3 clusters can then be overlaid with Vancouver maps of various kinds (e.g. parks, highways, lakes, etc.) The following findings can be obtained:

- About 96% of the houses in the first cluster (as described in Section 5.2) are within 0.6km from Queen Elizabeth Park.
- About 97% of the housing units in the second cluster are located in the Vancouver downtown area which is adjacent to Stanley Park [7].
- About 92% of houses in the third cluster are within 0.4km from the western coast line of Vancouver.

[7]During the summit meeting between Russia and the US in 1993, Clinton dined in Queen Elizabeth Park and jogged in Stanley Park!

The point here is that while SD(CLARANS) or NSD(CLARANS) do not directly find the above features (which is the job of another package that can provide such spatial operations as map overlays), they do produce structures or clusters that can lead to further discoveries.

6.2 Towards Building a More General and Efficient Spatial Data Mining Framework

A natural extension to SD(CLARANS) and NSD(CLARANS) will be the integration of the two algorithms by performing neither spatial dominant nor non-spatial dominant generalizations, but interleaved or balanced generalizations between spatial and non-spatial components. At each step, the data mining algorithm may select either a spatial or a non-spatial component to generalize. For example, if a clustering method can detect some high quality clusters, clustering may be performed first. These clusters may trigger generalization on non-spatial components in the next step if such a generalization may group objects into interesting groups. It is an interesting research issue to study how to compare the quality of spatial and non-spatial generalizations.

A spatial database may be associated with several thematic maps, each of which may represent one kind of spatial data. For example, in a city geographic database, one thematic map may represent the layout of streets and highways, another may outline the emergency service network, and the third one may describe the distribution of educational and recreational services. To many applications, it will be very useful if data mining on multiple thematic maps can be conducted simultaneously. This would involve not only clustering, but also other spatial operations such as spatial region growing, overlays and spatial joins. Thus, it is an interesting research issue to study how to provide an effective framework that integrates all these operations together for simultaneous mining of multiple maps.

There are many kinds of spatial data types, such as regions, points and lines, in spatial databases. Clustering methods, as presented here, are most suitable for points or small regions scattered in a relatively large background. However, it remains an open question as to how they can be effectively applied to deal with line-typed spatial data, such as to examine how highways are located in cities.

Furthermore, due to the nature of spatial data, noise or irrelevant information is prevalent in spatial databases. The development of a general framework for removing noises and filtering out irrelevant data is important to the effectiveness of spatial data mining. It is also interesting to find out what roles approximation and aggregation can play in the framework.

7 Conclusions

In this paper, we have presented a clustering algorithm called CLARANS which is based on randomized search. We have also developed two spatial data mining algorithms SD(CLARANS) and NSD(CLARANS). Experimental results and analysis indicate that both algorithms are effective, and can lead to discoveries that are difficult to obtain with existing spatial data mining algorithms. Finally, we have presented experimental results showing that CLARANS itself is more efficient than existing clustering methods. Hence, CLARANS has established itself as a very promising tool for efficient and effective spatial data mining.

References

[1] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. (1992) *An Interval Classifier for Database Mining Applications*, Proc. 18th VLDB, pp 560–573.

[2] R. Agrawal, T. Imielinski, and A. Swami. (1993) *Mining Association Rules between Sets of Items in Large Databases*, Proc. 1993 SIGMOD, pp 207–216.

[3] W. G. Aref and H. Samet. (1991) *Optimization Strategies for Spatial Query Processing*, Proc. 17th VLDB, pp. 81-90.

[4] A. Borgida and R. J. Brachman. (1993) *Loading Data into Description Reasoners*, Proc. 1993 SIGMOD, pp 217–226.

[5] T. Brinkhoff and H.-P. Kriegel and B. Seeger. (1993) *Efficient Processing of Spatial Joins Using R-trees*, Proc. 1993 SIGMOD, pp 237-246.

[6] O. Günther. (1993) *Efficient Computation of Spatial Joins*, Proc. 9th Data Engineering, pp 50-60.

[7] J. Han, Y. Cai and N. Cercone. (1992) *Knowledge Discovery in Databases: an Attribute-Oriented Approach,* Proc. 18th VLDB, pp. 547–559.

[8] Y. Ioannidis and Y. Kang. (1990) *Randomized Algorithms for Optimizing Large Join Queries*, Proc. 1990 SIGMOD, pp. 312–321.

[9] Y. Ioannidis and E. Wong. (1987) *Query Optimization by Simulated Annealing*, Proc. 1987 SIGMOD, pp. 9–22.

[10] L. Kaufman and P.J. Rousseeuw. (1990) *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley & Sons.

[11] D. Keim and H. Kriegel and T. Seidl. (1994) *Supporting Data Mining of Large Databases by Visual Feedback Queries*, to appear in Proc. 10th Data Engineering, Houston, TX.

[12] R. Laurini and D. Thompson. (1992) *Fundamentals of Spatial Information Systems*, Academic Press.

[13] W. Lu, J. Han and B. Ooi. (1993) *Discovery of General Knowledge in Large Spatial Databases,* Proc. Far East Workshop on Geographic Information Systems, Singapore, pp. 275-289.

[14] G. Milligan and M. Cooper. (1985) *An Examination of Procedures for Determining the Number of Clusters in a Data Set,* Psychometrika, 50, pp. 159–179.

[15] G. Piatetsky-Shapiro and W. J. Frawley. (1991) *Knowledge Discovery in Databases*, AAAI/MIT Press.

[16] H. Samet. (1990) *The Design and Analysis of Spatial Data Structures*, Addison-Wesley.

[17] H. Spath. (1985) *Cluster Dissection and Analysis: Theory, FORTRAN programs, Examples*, Ellis Horwood Ltd.