CLARANS: A method for clustering objects for spatial data mining

2 authors:

Raymond T. Ng
University of British Columbia - Vancouver
417 PUBLICATIONS   **15,663** CITATIONS

Jiawei Han
University of Illinois, Urbana-Champaign
664 PUBLICATIONS   **65,621** CITATIONS

Some of the authors of this publication are also working on these related projects:

Payload Extraction from Web Documents View project

Heart Failure View project

CLARANS: A Method for Clustering Objects 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. To this end, this paper has three main contributions. First, we propose a new clustering method called CLARANS, whose aim is to identify spatial structures that may be present in the data. Experimental results indicate that when compared with existing clustering methods, CLARANS is very efficient and effective. Second, we investigate how CLARANS can handle not only points objects, but also polygon objects efficiently. One of the methods considered, called the IR-approximation, is very efficient in clustering convex and non-convex polygon objects. Third, building on top of CLARANS, we develop two spatial data mining algorithms that aim to discover relationships between spatial and non-spatial attributes. Both algorithms can discover knowledge that is difficult to find with existing spatial data mining algorithms.

keywords: spatial data mining, clustering algorithms, randomized search, computational geometry

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.

Many excellent studies on data mining have been conducted, such as those reported in [2, 3, 6, 14, 20, 23, 26]. Agrawal et al. consider the problem of inferring classification functions from samples [2], and study the problem of mining association rules between sets of data items [3]. Han, Cai and Cercone propose an attribute-oriented approach to knowledge discovery [14]. And the book edited by Shapiro and Frawley includes many interesting studies on various issues in knowledge discovery such as finding functional dependencies between attributes [26]. However, most of these studies are concerned with knowledge discovery on non-spatial data, and the work most relevant to our focus here is the one reported in [23]. More specifically, Lu, Han and Ooi propose one spatial dominant and one non-spatial dominant algorithm 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.

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 to work effectively, there are the following key issues:

- Whether there exists a natural notion of similarities among the "objects" to be clustered. For spatial data mining, our approach here is to apply cluster analysis only on the spatial attributes. If these attributes correspond to point objects, natural notions of similarities exist (e.g., Euclidean or Manhattan distances). However, if the attributes correspond to polygon objects, the situation is more complicated. More specifically, the similarity (or distance) between two polygon objects may be defined in many ways, some better than others. But more accurate distance measurements may require more effort to compute. The main question then is for the kind of spatial clustering under consideration, which measurement achieves the best balance.

- Whether clustering a large number of objects can be efficiently carried out. Traditional cluster analysis algorithms are not designed for large data sets, with say more than 1000 objects.

In addressing these issues, we report in this paper:

- the development of CLARANS, which aims to use randomized search to facilitate the clustering of a large number of objects; and

- a study on the efficiency and effectiveness of three different approaches to calculate the similarities between polygon objects. They are the approach that calculates the exact separation distance between two polygons, the approach that over-estimates the exact distance by using the minimum distance between vertices, and the approach that under-estimates the exact distance by using the separation distance between the isothetic rectangles of the polygons.

To evaluate our ideas and algorithms, we present results – more often experimental than analytic – showing that:

- CLARANS is more efficient than the existing algorithms PAM and CLARA, both of which motivate the development of CLARANS; and

- calculating the similarity between two polygons by using the separation distance between the isothetic rectangles of the polygons is the most efficient and effective approach.

In [25], we present a preliminary study of CLARANS and the two spatial data mining algorithms. But this paper extends [25] in two major ways. First, CLARANS and the data mining algorithms are generalized to support polygon objects. As motivated above, clustering polygon objects effectively and efficiently is not straightforward at all. Second, this paper presents more detailed analysis and experimental results on the behavior of CLARANS and on the ways to fine tune CLARANS for specific applications.

Since the publication of [25], many clustering methods have been developed, which can be broadly categorized into partitioning methods [7], hierarchical methods [33, 12, 4, 18], density-based methods [11, 15], and grid-based methods [31, 29, 1]. In [7], Bradley et al. proposes an algorithm that follows the basic framework of the K-means algorithm, but that provides scalability by intelligently compressing some regions of the data space. In [33, 12, 4, 18], the proposed hierarchical methods try to detect nested clustering structures, which are prevalent in some applications. In [11, 15], the proposed density-based methods attempt to provide better clustering for elongated clusters; partitioning methods are typically much better suited for spherical clusters. In [31, 29, 1], the developed grid-based methods superimpose a grid structure onto the data space to facilitate clustering.

To compare CLARANS with these works, we make the following general observations:

- Many of the aforementioned techniques require some tree or grid structures to facilitate the clustering. Consequently, these techniques do not scale up well with increasing dimensionality of the datasets. While it is true that the material discussed in this paper is predominantly 2-D, the CLARANS algorithm works the same way for higher dimensional datasets. Because CLARANS is based on randomized search, and does not use any auxiliary structure, CLARANS is much less affected by increasing dimensionality.

- Many of the aforementioned techniques assume that the distance function is Euclidean. CLARANS, being a local search technique, makes no requirement on the nature of the distance function.

- Many of the aforementioned techniques deal with point objects; CLARANS is more general and supports polygonal objects. A considerable portion of this paper is dedicated to handling polygonal objects effectively.

3

- CLARANS is a main-memory clustering technique, while many of the aforementioned techniques are designed for out-of-core clustering applications. We concede that whenever extensive I/O operations are involved, CLARANS is not as efficient as the others. However, we argue that CLARANS still has considerable applicability. Consider the 2-D objects to be discussed in this paper. Each object is represented by 2 real numbers, occupying a total of 16 bytes. Clustering 1,000,000 objects would require slightly more than 16 Mbytes of main memory. This is an amount easily affordable by a personal computer, let alone computers for data mining. The point here is that given the very low cost of RAM, main-memory clustering algorithms, such as CLARANS, are not completely dominated by out-of-core algorithms for many applications. Finally, on a similar note, although some newly developed clustering methods may find clusters "natural" to the human eye and good for certain applications [4], there are still many applications, such as delivery services, to which partitioning-based clustering, such as CLARANS, is more appropriate.

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 and evaluates experimentally the three different approaches that compute the similarities between polygon objects. Section 5 concludes the paper with a discussion on ongoing works.

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. 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 [19]), 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

---

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

methods has been 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 [19] 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 [19]. 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_m is a (selected) medoid, we say that O_j belongs to the cluster represented by O_m, if $d(O_j, O_m) = min_{O_e} d(O_j, O_e)$, where the notation min_{O_e} denotes the minimum over all medoids O_e, and the notation $d(O_1, O_2)$ denotes the dissimilarity or distance between objects O_1 and O_2. 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_m and a non-selected object O_p is made, as long as such a swap would result in an improvement of the quality of the clustering.

Before we embark on a formal analysis, let us consider a simple example. Suppose there are 2 medoids: A and B. And we consider replacing A with a new medoid M. Then for all the objects Y that are originally in the cluster represented by A, we need to find the nearest medoid in light of the replacement. There are two cases. In the first case, Y moves to the cluster represented by B, but not to the new one represented by M. In the second case, Y moves to the new cluster represented by M, and the cluster represented by B is not affected. Apart from re-considering all the objects Y that are originally in A's cluster, we also need to consider all the objects Z that are originally in B's cluster. In light of the replacement, Z either stays with B, or moves to the new cluster represented by M. Figure 1 illustrates the four cases.

In the remainder of this paper, we use:

- O_m to denote a current medoid that is to be replaced (e.g., A in Figure 1);

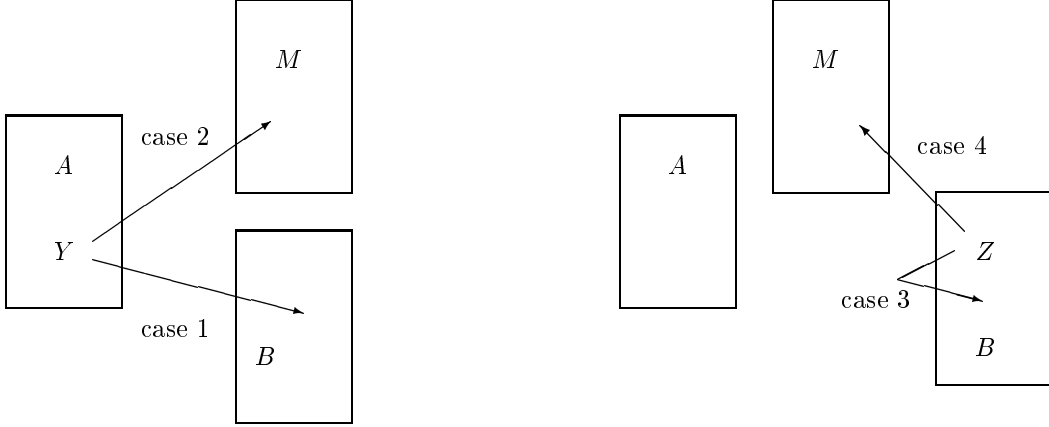- O_p to denote the new medoid to replace O_m (e.g., M in Figure 1);

Figure 1: Four cases for Replacing A with M

- O_j to denote other non-medoid objects that may or may not need to be moved (e.g., Y and Z in Figure 1); and

- $O_{j,2}$ to denote a current medoid that is nearest to O_j without A and M (e.g., B in Figure 1).

Now to formalize the effect of a swap between O_m and O_p, PAM computes costs C_{jmp} for all non-medoid objects O_j. Depending on which of the following cases O_j is in, C_{jmp} is defined differently.

Case 1: suppose O_j currently belongs to the cluster represented by O_m. Furthermore, let O_j be more similar to $O_{j,2}$ than to O_p, i.e., $d(O_j, O_p) \geq d(O_j, O_{j,2})$, where $O_{j,2}$ is the second most similar medoid to O_j. Thus, if O_m is replaced by O_p as a medoid, O_j would belong to the cluster represented by $O_{j,2}$ (cf: Case 1 in Figure 1). Hence, the cost of the swap as far as O_j is concerned is:

$$C_{jmp} \;\; = \;\; d(O_j, O_{j,2}) - d(O_j, O_m). \tag{1}$$

This equation always gives a non-negative C_{jmp}, indicating that there is a non-negative cost incurred in replacing O_m with O_p.

Case 2: O_j currently belongs to the cluster represented by O_m. But this time, O_j is less similar to $O_{j,2}$ than to O_p, i.e., $d(O_j, O_p) < d(O_j, O_{j,2})$. Then, if O_m is replaced by O_p, O_j would belong to the cluster represented by O_p (cf: Figure 1). Thus, the cost for O_j is given by:

$$C_{jmp} \;\; = \;\; d(O_j, O_p) - d(O_j, O_m). \tag{2}$$

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

Case 3: suppose that O_j currently belongs to a cluster other than the one represented by O_m. Let $O_{j,2}$ be the representative object of that cluster. Furthermore, let O_j be more

similar to $O_{j,2}$ than to O_p. Then even if O_m is replaced by O_p, O_j would stay in the cluster represented by $O_{j,2}$. Thus, the cost is:

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

Case 4: O_j currently belongs to the cluster represented by $O_{j,2}$. But O_j is less similar to $O_{j,2}$ than to O_p. Then replacing O_m with O_p would cause O_j to jump to the cluster of O_p from that of $O_{j,2}$ Thus, the cost is:

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

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

$$TC_{mp} \quad = \quad \sum_j C_{jmp} \tag{5}$$

We now present Algorithm PAM.

Algorithm PAM

1. Select k representative objects arbitrarily.

2. Compute TC_{mp} for *all* pairs of objects O_m, O_p where O_m is currently selected, and O_p is not.

3. Select the pair O_m, O_p which corresponds to $min_{O_m, O_p} TC_{mp}$. If the minimum TC_{mp} is negative, replace O_m with O_p, 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 [19]). 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_m, O_p. For each pair, computing TC_{mp} 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 [19]. 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 [19] 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 give a graph-theoretic framework within which we can compare PAM and CLARA, and motivate the development of CLARANS. Then after describing the details of the algorithm, we will present experimental results showing how to fine tune CLARANS, and that CLARANS outperforms CLARA and PAM in terms of both efficiency and effectiveness.

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$\}$.

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

---

[2] [19] 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.

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_m, O_p are the differences between neighbors S_1 and S_2 (i.e., $O_m, O_p \notin S_1 \cap S_2$, but $O_m \in S_1$ and $O_p \in S_2$), the cost differential between the two neighbors is exactly given by T_{mp} 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 [16, 17], 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 2 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 $maxneighbor$, where $minmaxneighbor$ is a user-defined minimum value for $maxneighhbor$:

  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 2 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 2(a) shows that the lower the value of p, the smaller the amount of runtime CLARANS requires. And as expected, Figure 2(b) shows that a lower value of p produces a lower quality clustering (i.e., higher (relative) average distance). But the very amazing

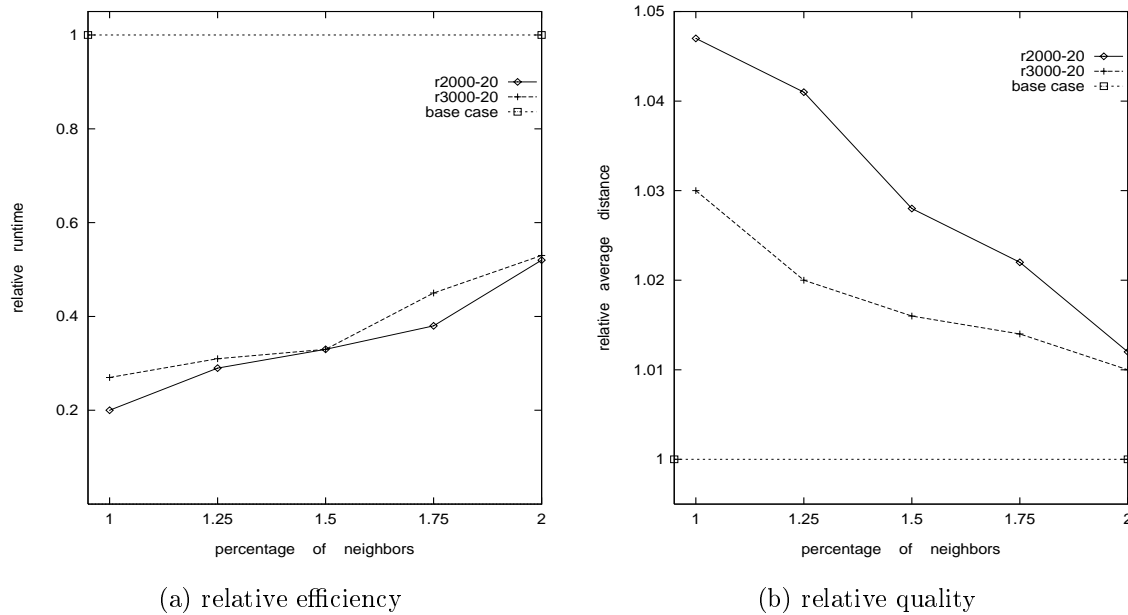11

| (a) relative efficiency | (b) relative quality |

Figure 2: Determining the Maximum Number of Neighbors

feature shown in Figure 2(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 of experiments, we ran CLARANS with $numlocal = 1, \ldots, 5$ on data sets rn-k and tn-k for
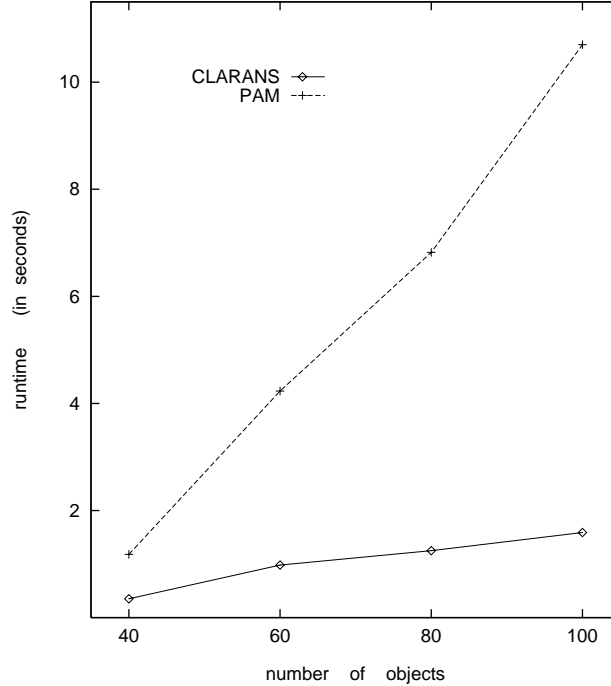
Figure 3: Efficiency: CLARANS vs PAM

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 data sets with 40, 60, 80 and 100 points in 5 clusters. Figure 3 shows the runtime taken by
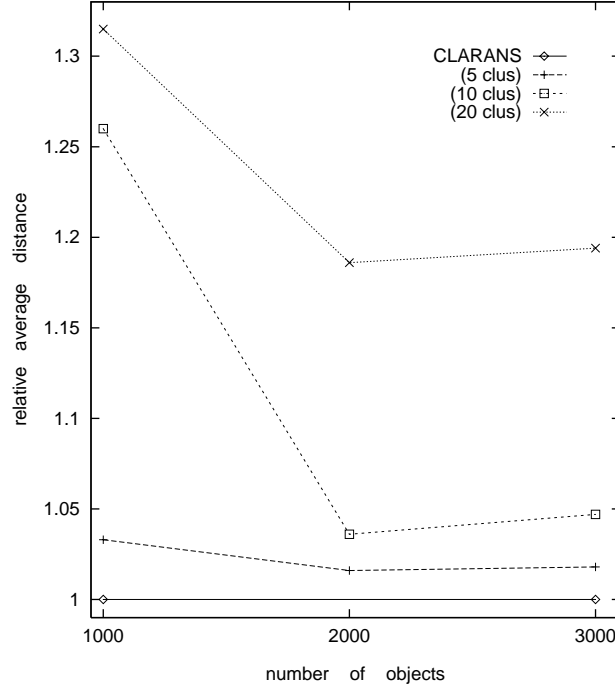
13

Figure 4: Relative Quality: Same Time for CLARANS and CLARA

both algorithms. Note that for all those data sets, the clusterings produced by both algo-rithms 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 3 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 Sec-tion 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 pre-sented 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 4 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 4 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 Clustering Convex Polygon Objects

4.1 Motivation

As described in Section 3.3, all the experiments presented so far assume that each object is represented as a point, in which case standard distance metrics such as the Manhattan distance, and the Euclidean distance can be used to calculate the distance between two objects/points. However, in practice, numerous spatial objects that we may want to cluster are polygonal in nature, e.g., shopping malls, parks. The central question then is how to calculate the distance between two polygon objects efficiently and effectively for clustering purposes. One obvious way to approximate polygon objects is to represent each object by a representative point, such as the centroid of the object. However, in general, the objects being clustered may have widely varied sizes and shapes. For instance, a typical house in Vancouver may have a lot size of 200 square meters and a rectangular shape, whereas Stanley Park in Vancouver has a size of about 500,000 square meters and an irregular shape that hugs the shoreline. Simply representing each of these objects by its centroid, or any single point, would easily produce clusterings of poor quality.

Given the above argument, one may wonder whether it is sufficient to represent a polygon object by multiple points in the object, e.g., points on the boundary of the object. But for large objects like Stanley Park, two of its representative points may be 5,000 meters apart

---

[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.

from each other. If these two representative points are fed to CLARANS as *individual* points/objects, there is no guarantee that they will be in the same cluster. This would result in Stanley Park being assigned to more than one cluster, thus violating the partitioning requirement of the clustering algorithms. [4]

This motivates why in this section, we study how CLARANS (and for that matter, CLARA and PAM) can be augmented to allow convex polygon objects – in their entireties – to be clustered. The key question is how to compute efficiently the distance between two polygons. To answer this question, we study three different approaches. The first one is based on computing the exact separation distance between two convex polygon objects. The second approach uses the minimum distance between vertices to approximate the exact separation distance. The third approach approximates by using the separation distance between isothetic rectangles. We analyze the pros and cons, and complexities of these approaches. Last but not least, we propose a performance optimization that is based on memoizing the computed distances. At the end of this section, we will give experimental results evaluating the usefulness of these ideas.

4.2 Calculating the Exact Separation Distance

In coordinate geometry, the distance between a point P and a line L is defined as the minimum distance, in this case the perpendicular distance, between the point and the line, i.e., $min \{d(P,Q) \mid Q$ is a point on $L\}$. Thus, given two polygons A, B, it is natural for us to define the distance between these two polygons to be the minimum distance between any pair of points in A, B, i.e., $min \{d(P,Q) \mid P, Q$ are points in A, B respectively $\}$. This distance is exactly the same as the minimum distance between any pair of points on the boundaries of A, B. This is called the *separation* distance between the two polygons [9, 27].

We need two key steps to compute the separation distance between two convex polygons A, B. First, we need to determine whether A and B have any intersection. This can be done in $O(log\, n\, +\, log\, m)$ time where n, m denotes the number of vertices A and B have [27]. (A vertix of a polygon is the intersection point between two edges of the polygon.) If the two polygons intersect, then the separation distance is zero. Otherwise, we compute the separation distance between the two boundaries. This again can be done in $O(log\, n + log\, m)$ time [9, 21].

While there exist different algorithms that compute the separation distance and that have the same $O(log\, n\, +\, log\, m)$ complexity, we have chosen and implemented one of the most efficient such algorithms that is reported in [21]. Instead of trying all vertices and edges on the boundaries of the two polygons, this algorithm first identifies two chains of vertices and consecutive line segments (one chain from each polygon) that "face" each other in the sense that a vertex in either chain can "see" at least one vertex in the other chain. If P is a vertex in A, and Q a vertex in B, we say that P and Q "see" each other if the line segment joining P and Q does not intersect the interior of either polygon [5]. Thus, by definition, the

---

[4]Note that there exist clustering algorithms that allow clusters to overlap. However, here we are only concerned with the most standard kind of clusterings, generally known as *crisp* clustering, where each object is assigned to exactly one cluster.

[5]This definition ignores any object that may lie between A and B.

(a) MV-approximation vs separation distance
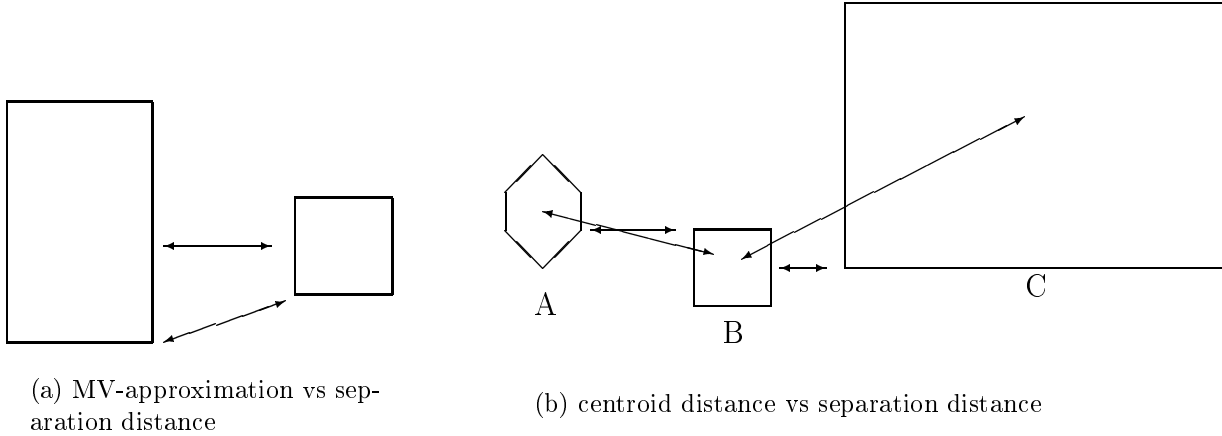
(b) centroid distance vs separation distance

Figure 5: MV-approximation vs Separation Distance vs Centroid Distance

separation distance of the two polygons must be the minimum distance between any pair of points (not necessarily vertices) on the two chains. By taking a binary search strategy, the algorithm finds the latter distance.

4.3   Approximating by the Minimum Distance between Vertices

One way to approximate the exact separation distance between two polygons is to find the minimum distance between the vertices of the polygons, i.e., $min\ \{d(P,Q)\,|\,P,Q$ are vertices of A, B respectively $\}$. Hereafter, we refer to this approximation as the *MV-approximation*. Obviously, the MV-approximation requires a time complexity of $O(n*m)$. Even though from a complexity point of view, this approximation is inferior to the algorithm described above that computes the exact separation distance, in practice it usually outperforms the exact algorithm, unless when the values of n and m are moderately high, e.g., exceeding 20. For many spatial data mining applications, it is sufficient to represent most spatial objects with fewer than 20 vertices and edges. This justifies using the MV-approximation to optimize performance. Section 4.6.2 will give experimental results on the efficiency of the MV-approximation.

Figure 5(a) shows a simple example demonstrating that the separation distance between two polygons need not be equal to the minimum distance between vertices. However, it is easy to see that the separation distance cannot exceed the minimum distance between vertices. Thus, the MV-approximation always over-estimates the actual separation distance. From the point of view of clustering objects by their MV-approximations, the key question is whether such over-estimations would affect the (quality of the) clusterings.

Recall from Section 4.1 that we argue that using just the centroid to represent an object could produce clusterings of poor quality. And the distance between the centroids of A and B always over-estimates the actual separation distance. At this point, one may wonder whether the over-estimation by centroid distance and the MV-approximation have similar effects. The key difference is that the former, depending largely on the sizes and shapes of the polygons, gives "non-uniform" approximations, whereas the latter is more consistent in

its approximations. In Figure 5(b), B is closer to C than to A based on their exact separation distances. Whereas the centroid distance between A and B is fairly close to the separation distance it approximates, the centroid distance between B and C is many times higher than the actual separation distance between B and C. In fact, by their centroid distances, B is closer to A than to C, reversing the ordering induced by their separation distances. In general, if a collection of objects have widely varied sizes and shapes, the centroid distance approximation would produce poor clusterings (relative to the clusterings produced by using the separation distances). On the other hand, for the example shown in Figure 5(b), the MV-approximation preserves the ordering that B is closer to C than to A. Certainly, an approximation is an approximation, and it is not hard to construct situations where the MV-approximation can be inaccurate. However, by trying the approximation on numerous polygon objects that can be found in real maps, we have verified that the MV-approximation is reasonable, and is much less susceptible to variations in sizes and shapes than the centroid distance approximation is. Section 4.6.4 will give experimental results on the quality of the clusterings produced by the MV-approximation.

4.4 Approximating by the Separation Distance between Isothetic Rectangles

Another way to approximate the exact separation distance between two polygons A, B is to 1) compute isothetic rectangles I_A, I_B, and 2) calculate the separation distance between I_A and I_B. Given a polygon A, the isothetic rectangle I_A is the smallest rectangle that contains A, and whose edges are parallel to either the x- or the y- axes. Hereafter, we refer to this approximation to the exact separation distance as the *IR-approximation.*

For any given polygon A, a minimum bounding rectangle of A is defined to be the smallest rectangle – in area – that contains A. As such, a minimum bounding rectangle need not have its edges parallel to either the x- or the y- axes. Precisely because of this, it is relatively costly to compute a minimum bounding rectangle. In contrast, the isothetic rectangle, while possibly having an area larger than that of a minimum bounding rectangle, can be easily obtained by finding the minimum and maximum of the sets $\{v \mid v$ is the x-coordinate of a vertex of the polygon $\}$, and $\{w \mid w$ is the y-coordinate of a vertex of the polygon $\}$. Thus, Step 1 of the IR-approximation takes a trivial amount of time to compute.

Like computing the exact separation distance between two polygons, as described in Section 4.2, calculating the exact separation distance between two isothetic rectangles requires two steps. In the first step where possible intersection is checked, it only takes constant time for isothetic rectangles, but time logarithmic to the number of vertices for polygons. Similarly, in the next step where the actual separation distance is computed (necessary when the two rectangles or polygons do not intersect), it is constant time for isothetic rectangles, but logarithmic time for polygons. In particular, for isothetic rectangles, it suffices to call repeatedly a procedure that computes the distance between a point and a line segment. Thus, Step 2 of the IR-approximation can be done efficiently. And the IR-approximation can be used to reduce the time otherwise needed to compute the exact separation distance.

From efficiency to effectiveness, just as we evaluate in the previous section how the MV-approximation affects the quality of the clusterings, here we should ask a similar question.

On one hand, the IR-approximation is different from the MV-approximation in that the former always under-estimates the actual separation distance between the original polygons. This is because the isothetic rectangle of a polygon contains the polygon. On the other hand, like the MV-approximation, so long as the IR-approximation under-estimates all polygons being clustered fairly uniformly, the clusterings produced would be of comparable quality to those produced by using the exact separation distances. Section 4.6.4 will give experimental results comparing the quality of the clusterings produced by these two approaches.

So far we have portrayed the IR-approximation (and the MV-approximation) as a performance optimization to computing the exact separation distance. Actually there is another advantage being offered by the IR-approximation (and the MV-approximation). That is, it does not require the original polygon to be convex. As described above, the definition of the isothetic rectangle of a polygon applies equally well to convex and non-convex polygons. Thus, when integrated with the IR-approximation, CLARANS can be used to cluster any polygons. In contrast, the method described in Section 4.2 only works for convex polygons, thereby also restricting CLARANS to convex polygons.

4.5 Memoization of Exact and Approximated Distances

Recall from previous sections that for any pair of objects O_m, O_j, the distance $d(O_m, O_j)$ between the two objects may be referenced numerous times in an execution of CLARANS. When the objects are merely points, this distance can be computed dynamically each time it is needed. This is sufficient because calculating the Manhattan or the Euclidean distance is a simple operation that takes microseconds to complete.

The situation is, however, very different when the objects being clustered are polygons. As will be shown experimentally in Section 4.6.2, calculating the similarity between two polygons, even when the IR-approximation or the MV-approximation is used, still takes milliseconds to complete. Since the distance $d(O_m, O_j)$ may be needed repeatedly, it makes sense to, once computed, memoize the distance. This would ensure that the distance between each pair of polygon objects is computed at most once. Clearly, this memoization strategy trades off space for time efficiency. Section 4.6.5 will give experimental results evaluating whether memoization is valuable, and whether this tradeoff is worthwhile.

4.6 Experimental Evaluation

4.6.1 Details of the Experiments

Recall that both the MV-approximation and the IR-approximation are applicable to calculating the separation distance between non-convex polygons. But because the method described in Section 4.2 only works for convex polygons, all our experiments are restricted to polygon objects that are convex. To generate random polygons of different shapes, sizes, and orientations, we used the convex polygon generator developed by Snoeyink for testing some of the ideas reported in [21]. The generator computes and outputs n evenly-spaced points on an ellipse with center $(ctrx, ctry)$, starting with angle $offset$, and having major and minor axes parallel to the x- and y- axes with radii r_1, r_2 respectively, where $n, ctrx, ctry, offset, r_1, r_2$ are all inputs to the generator. We used a random number generator to create all these
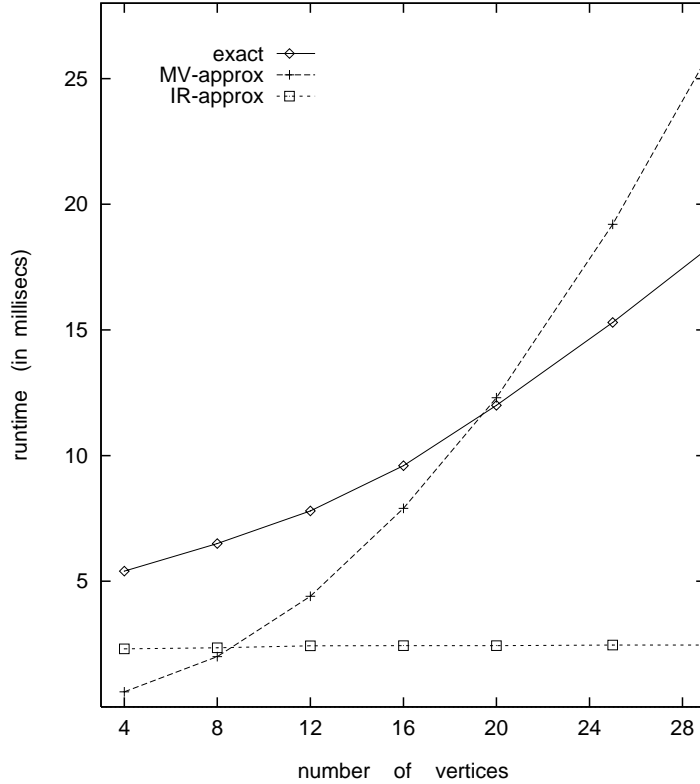
Figure 6: Efficiency of the Approximations

inputs, in such a way that all generated polygons lie in a rectangular region with length l and width w. By varying the values of l and w, we can generate sets of polygons with different densities, and thus with different proportions of polygons that overlap with some other polygons in the set.

All the experiments reported here were carried out in a time-sharing SPARC-LX workstation. Whenever CLARANS was used, all figures concerning CLARANS, due to its random nature, were average figures obtained by running the same experiment 10 times.

4.6.2 Efficiency: Exact Distance vs IR-approximation vs MV-approximation

In this series of experiments, we used polygons with different numbers of vertices, and recorded the average amount of runtime needed to calculate the exact separation distance, its IR-approximation, and its MV-approximation for one pair of polygons. Figure 6 shows the results with the runtimes recorded in milliseconds.

The IR-approximation approach is the clear winner in that it always outperforms the exact separation distance approach, that it beats the MV-approximation approach by a wide margin when the number of vertices is high, and that even when the number of vertices is small, it delivers a performance competitive with that of the MV-approximation approach. Recall from Section 4.4 that two steps are needed to compute the IR-approximation. The first step is to compute the isothetic rectangles, which has a complexity of $O(n)$. The second step is to compute the separation distance between the isothetic rectangles, which has a complexity of $O(1)$. The flatness of the curve for the IR-approximation in Figure 6 clearly

20

shows that the second step dominates the first one. Thus, as a whole, the IR-approximation does not vary as the number of vertices increases.

In contrast, both the exact separation distance and the MV-approximation require higher runtimes as the number of vertices increases. When the number of vertices is less than 20, calculating the exact separation distance takes more time than the MV-approximation does. But the reverse is true when the number of vertices exceeds 20. In other words, the runtime for the MV-approximation grows faster than that for computing the exact separation distance. This is consistent with the complexity results presented in Sections 4.2 and 4.3.

4.6.3 Clustering Effectiveness: Exact Distance vs IR-approximation vs MV-approximation

While the previous series of experiments do not involve clustering, in this series of experiments, CLARANS was integrated with the three different ways to calculate distances between polygons. Because the number of vertices of polygons is a parameter that affects the performance of the three approaches, we ran CLARANS with sets of n-sided polygons where n varies from 4 to 20, and with a random mix of polygons, each of which has between 4 to 20 edges. For this series of experiments, we focus on both clustering effectiveness and efficiency.

Regarding the effectiveness of the two approximations relative to the exact distance approach, recall that the IR-approximation always underestimates the exact distance, while the MV-approximation always overestimates it. Thus, it is not appropriate to simply measure the effectiveness based on the average approximated distance between a polygon and the medoid of its cluster. Instead, we compare the clusters produced by the exact distance approach and the clusters produced by the two approximations. More precisely, for a particular polygon A, we calculate the ratio:

the exact distance between A and M'_A divided by the exact distance between A and M_A,

where M'_A is the medoid of the cluster A is assigned to using the approximation, and M_A is the corresponding medoid using exact distance. The closer to 1 the ratio is, the more accurate the approximation.

As it turns out for almost every set of polygons we experimented with, over 90% of the polygons satisfy $M'_A = M_A$. That is to say, despite the approximations, the clustering structures are very much preserved. Furthermore, the following table shows the aforementioned ratio average over all polygons A.

Approach	Average Ratio
MV-approximation	1.02
IR-approximation	1.03

The quality of the clusterings produced by the IR-approximation and the MV-approximation is almost identical to the quality of the clustering produced by using the exact separation distance, differing by about 2-3%. This clearly shows that the two approximations are very effective, as they under-estimate or over-estimate the actual distances so consistently that the clusters are preserved. Thus, using the IR-approximation and the MV-approximation as ways to optimize performance is justified.
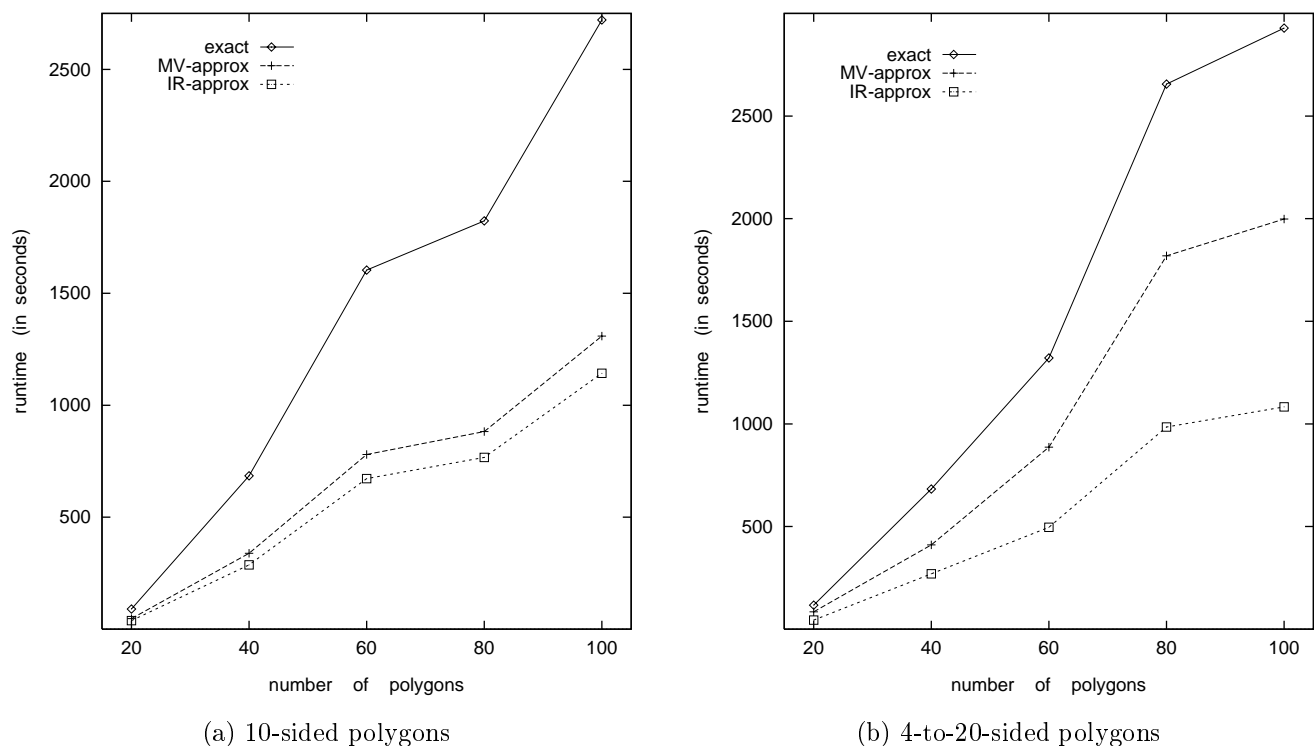
(a) 10-sided polygons   (b) 4-to-20-sided polygons

Figure 7: Clustering Efficiency of the Approximations

4.6.4 Clustering Efficiency: Exact Distance vs IR-approximation vs MV-approximation

Next we consider the efficiency of the two approximations relative to the exact distance approach. Figure 7 shows the times needed by CLARANS to cluster varying number of polygons that have 10 edges, and varying number of polygons that have between 4 to 20 edges. In both cases, the IR-approximation and the MV-approximation considerably outperform the exact separation distance approach. In particular, the IR-approximation is always the most efficient, requiring only about 30% to 40% the time needed by the exact distance approach.

Other sets of polygons that we experimented with, including some that have varying densities, also give the same conclusion. Thus, given the fact that the IR-approximation is capable of delivering clusterings that are of almost identical quality to those produced by the exact approach, the IR-approximation is the definite choice for CLARANS. Other experiments we conducted indicate that same conclusion can be drawn if PAM and CLARA were to use to cluster polygon objects.

4.6.5 Effectiveness of Memoizing Computed Distances

While the graphs in Figure 7 identify the IR-approximation approach as the clear winner, the performance results of the approximation is disappointing. For example, it takes about 1000 seconds to cluster 100 polygons. Recall from Figure 6 that the IR-approximation for one pair of polygons takes 2 to 3 milliseconds. For 100 polygons, there are 100*100/2 = 5000 pairs of polygons. These 5000 distances take a total of 10 to 15 seconds to compute. Thus, what
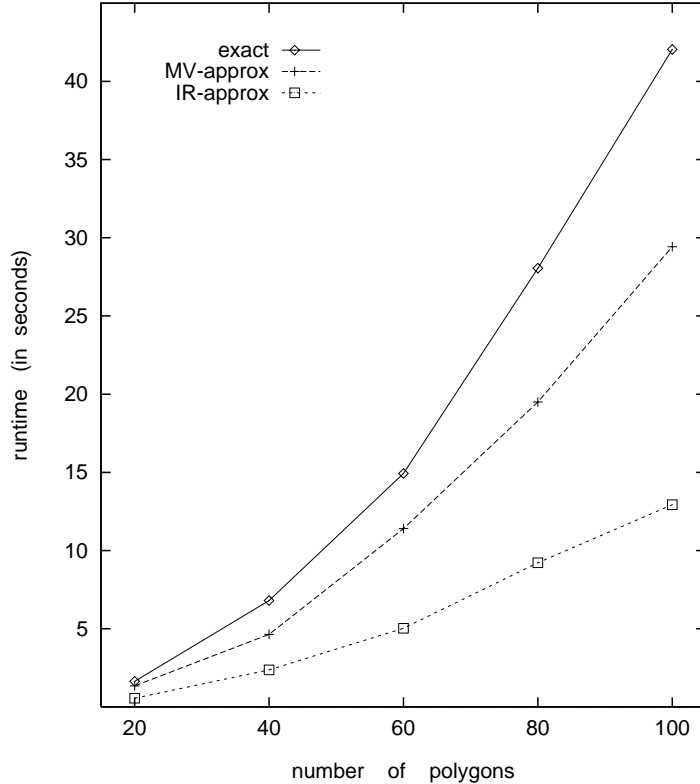
Figure 8: Clustering Efficiency with Distance Memoization

is happening is that the distance between each pair of polygons is computed on the average 60 to 100 times. This argues very strongly why computed distances should be memoized as a performance optimization. Figure 8 shows the results of applying memoization to the same set of polygons used in Figure 7(b). Indeed, with memoization, the time taken to cluster 100 polygons with the IR-approximation drops to about 10 seconds, as estimated above. Similar performance gains are obtained if either the exact separation distance or the MV-approximation are used.

4.7   Summary

Regarding how to compute the similarities between objects, our experimental results clearly indicate that the IR-approximation approach is the choice. While only approximating the exact separation distance, the IR-approximation approach can deliver clusterings of almost identical quality to the clusterings produced by the exact distance approach. The decisive factor is then the efficiency of the IR-approximation approach. It has the desirable property that its computation time does not vary with the number of vertices of the polygon objects, and that it outperforms the exact distance approach typically by 3 to 4 times. Furthermore, it can be used for both convex and non-convex polygon objects.

While the material in this section focuses on collections of objects that are all polygons, the results can be easily generalized to heterogeneous collections of objects, where some objects are polygons and the remaining ones are points. For such collections, the similarity between a point and a polygon can be defined as the distance between the point and the

isothetic rectangle of the polygon. This distance can be computed in constant time.

5   Conclusions

In this paper, we have presented a clustering algorithm called CLARANS which is based on randomized search. For small data sets, CLARANS is a few times faster than PAM; the performance gap for larger data sets is even larger. When compared with CLARA, CLARANS has the advantage that the search space is not localized to a specific subgraph chosen a priori, as in the case of CLARA. Consequently, when given the same amount of run-time, CLARANS can produce clusterings that are of much better quality than those generated by CLARA.

We have also studied how polygon objects can be clustered by CLARANS. We have proposed three different ways to compute the distance between two polygons. Complexity and experimental results indicate that the IR-approximation is a few times faster than the method that computes the exact separation distance. Furthermore, experimental results show that despite the much smaller run-time, the IR-approximation is able to find clusterings that are of quality almost as good as those produced by using the exact separation distances. In other words, the IR-approximation can give significant efficiency gain – but without loss of effectiveness.

In ongoing work, we are developing a package that uses clustering as a basis for providing many operations for mining spatial data. The spatial data mining algorithms SD(CLARANS) and NSD(CLARANS) described in [25] are two examples. There are also operations for mining with maps [32].

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan. (1998) *Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications*, Proc. 1998 ACM-SIGMOD, pp. 94–105.

[2] 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.

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

[4] M. Ankerst, M. Breunig, H.-P. Kriegel and J. Sander. (1999) *OPTICS: Ordering Points To Identify the Clustering Structure*, Proc. 1999 ACM-SIGMOD, pp. 49–60.

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

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

[7] P. Bradley, U. Fayyad and C. Reina. (1998) *Scaling Clustering Algorithms to Large Databases*, Proc. 4th KDD, pp. 9–15.

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

[9] D. Dobkin and D. Kirkpatrick. (1985) *A Linear Algorithm for Determining the Separation of Convex Polyhedra*, Journal of Algorithms, 6, 3, pp 381–392.

[10] M. Ester, H. Kriegel and X. Xu. (1995) *Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification*, Proc. 4th Int'l Symp. on Large Spatial Databases (SSD'95), pp 67-82.

[11] M. Ester, H. Kriegel, J. Sander and X. Xu. (1996) *A Density-based Algorithm for Discovering Large Clusters in Large Spatial Databases with Noise*, Proc: 2nd KDD, 1996.

[12] S. Guha, R. Rastogi and K. Shim. (1998) *CURE: An Efficient Clustering Algorithm for Large Databases*, Proc. 1998 ACM-SIGMOD, pp. 73–84.

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

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

[15] A. Hinneburg and D. A. Keim. (1998) *An Efficient Approach to Clustering in Large Multimedia Databases with Noise*, Proc. 1998 KDD, pp. 58–65.

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

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

[18] G. Karypis, E.-H. Han and V. Kumar. (1999) *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling*, IEEE Computer, 32, 8, pp. 68–75.

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

[20] 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.

[21] D. Kirkpatrick and J. Snoeyink. (1993) *Tentative Prune-and-search for Computing Fixed-Points with Applications to Geometric Computation*, to appear in Fundamenta Informaticae. Prelim. version in Proc. 9th ACM Symposium on Computational Geometry, pp 133–142.

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

[23] 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.

[24] 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.

[25] R. Ng and J. Han. (1994) *Efficient and Effective Clustering Methods for Spatial Data Mining*, Proc. 20th VLDB, pp. 144-155.

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

[27] F. Preparata and M. Shamos. (1985) *Computational Geometry*, New York, Springer-Verlag.

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

[29] G. Sheikholeslami, S. Chatterjee and A. Zhang. (1998) *WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases*, Proc. 1998 VLDB, pp. 428–439.

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

[31] W. Wang, J. Yang and R. Muntz. (1997) *STING: a Statistical Information Grid Approach to Spatial Data Mining*, Proc. 23rd VLDB, pp. 186–195.

[32] Y. Yu. (1996) *Finding Strong, Common and Discriminating Characteristics of Clusters from Thematic Maps*, MSc Thesis, Department of Computer Science, University of British Columbia.

[33] T. Zhang, R. Ramakrishnan and M. Livny. (1996) *BIRCH: an Efficient Data Clustering Method for Very Large Databases*, Proc. '96 SIGMOD, pp. 103–114.

Biographies

Raymond Ng received his PhD degree in computer science from the University of Maryland, College Park in 1992. Since then, he has been an assistant and associate professor at the University of British Columbia. His current research interests include data mining, bioinformatics and multimedia database systems. He has published over 80 journal and conference papers, and has served in many program committees of ACM-SIGMOD, VLDB and SIG-KDD.

Jiawei Han received the PhD degree in computer sciences from the University of Wisconsin at Madison in 1985. He is currently the director of Intelligent Database Systems Research Laboratory and professor in the School of Computing Science, Simon Fraser University, Canada.

He has conducted research in the areas of database systems, data mining, data warehousing, geo-spatial data mining, web mining, deductive and object-oriented database systems, and artificial intelligence, with more than 150 journal or conference publications. He is currently a project leader of the Canada Networks of Centres of Excellence (NCE)/IRIS-3 project "Building, querying, analyzing and mining data warehouses on the internet" (1998–2002), and the chief architect of the DBMiner system. He has served or is currently serving in the program committees for more than 50 international conferences and workshops. He has also been serving on the editorial boards of IEEE Transactions on Knowledge and Data Engineering, Data Mining and Knowledge Discovery, and the Journal of Intelligent Information Systems. He is a member of the IEEE Computer Society, the ACM, the ACM-SIGMOD, ACM-SIGKDD, and AAAI.