CHAPTER 7. PAPER 3:

EFFICIENT HIERARCHICAL CLUSTERING OF LARGE DATA SETS USING P-TREES

7.1. Abstract

Hierarchical clustering methods have attracted much attention by giving the user a maximum amount of flexibility. Rather than requiring parameter choices to be predetermined, the result represents all possible levels of granularity. In this paper, a hierarchical method is introduced that is fundamentally related to partitioning methods, such as k-medoids and k-means, as well as to a density based method, center-defined DENCLUE. It is superior to both k-means and k-medoids in its reduction of outlier influence. Nevertheless, it avoids both the time complexity of some partition-based algorithms and the storage requirements of density-based ones. An implementation that is particularly suited to spatial, stream, and multimedia data using P-trees for efficient data storage and access is presented.

7.2. Introduction

Many clustering algorithms require choosing parameters that will determine the granularity of the result. Partitioning methods such as the k-means and k-medoids [1] algorithms require that the number of clusters, k, be specified. Density-based methods, e.g., DENCLUE [2] and DBScan [3], use input parameters that relate directly to cluster size rather than the number of clusters. Hierarchical methods [1,4] avoid the need to specify either type of parameter and instead produce results in the form of tree structures that include all levels of granularity. When generalizing partitioning-based methods to hierarchical ones, the biggest challenge is performance. With the ever-increasing data-set sizes in most data mining applications, this performance is one of the main issues we have to address.

Determining a k-medoids based clustering for only one value of k has already got a prohibitively high time complexity for moderately sized data sets. Many solutions have been proposed [5-7], such as CLARA [1] and CLARANS [5], but an important fundamental issue remains: the algorithm inherently depends on the combined choice of cluster centers. Its complexity, thereby, must scale essentially as the square of the number of investigated sites. In this paper, we analyze the origins of this unfavorable scaling and see how it can be eliminated at a fundamental level. We note that our proposed solution is related to the density-based clustering algorithm DENCLUE [2], albeit with a different justification.

Based on our definition of a cluster center, we define a hierarchy that naturally structures clusters at different levels. Moving up the hierarchy uniquely identifies each low-level cluster as part of a particular higher-level one. Low-level cluster centers provide starting points for the efficient evaluation of higher-level cluster centers. Birch [4] is another hierarchical algorithm that uses k-medoids related ideas without incurring the high time complexity. In Birch, data are broken up into local clustering features and then combined into CF Trees. In contrast to Birch, we determine cluster centers that are defined globally. We represent data in the form called P-trees, which are efficient both in their storage requirements and the time-complexity of computing global counts.

The paper is organized as follows. In Section 7.3, we analyze the reasons for the high time complexity of k-medoids-based algorithms and show how this complexity can be avoided using a modified concept of a cost-function. In Section 7.4, we introduce an algorithm to construct a hierarchy of cluster centers. In Section 7.5, we discuss our implementation using P-trees.

7.3. Taking a Fresh Look at Established Algorithms

Partition-based and density-based algorithms are commonly seen as fundamentally and technically distinct. Work on combining both has focused on an applied rather than a fundamental level [8]. We will present three of the most popular algorithms from the two categories in a context that allows us to extract the essence of both.

The existing algorithms we consider in detail are k-medoids [1] and k-means as partitioning techniques, and the center-defined version of DENCLUE [2] as a density-based one. The goal of these algorithms is to group data items with cluster centers that represent their properties well. The clustering process has two parts that are strongly related to each other in the algorithms we review but will be separated in our clustering algorithm. The first part is to find cluster centers while the second is to specify boundaries of the clusters. We first look at strategies that are used to determine cluster centers. Since the k-medoids algorithm is commonly seen as producing a useful clustering, we start by reviewing its definition.

7.3.1. K-medoids Clustering as a Search for Equilibrium

A good clustering in k-medoids is defined through the minimization of a cost function. The most common choice of the cost function is the sum of squared Euclidean distances between each data item and its closest cluster center. An alternative way of looking at this definition borrows ideas from physics: we can look at cluster centers as particles that are attracted to the data points. The potential that describes the attraction for each data item is taken to be a quadratic function in the Euclidean distance as defined in the *d*-dimensional space of all attributes. The energy landscape surrounding a cluster center with position $X^{(m)}$ is the sum of the individual potentials of data items at locations $X^{(i)}$:

$$E(X^{(m)}) = \sum_{i=1}^{N} \sum_{j=1}^{d} (x_j^{(i)} - x_j^{(m)})^2 ,$$

where *N* is the number of data items that are assumed to influence the cluster center. It can be seen that the potential that arises from more than one data point will continue to be quadratic since the sum of quadratic functions is again a quadratic function. We can calculate the location of its minimum as follows:

$$\frac{\partial}{\partial x_j^{(m)}} E(X^{(m)}) = -2\sum_{i=1}^{N} (x_j^{(i)} - x_j^{(m)}) = 0$$

The minimum of the potential is, therefore, the mean of coordinates of the data points to which it is attracted.

$$x_j^{(m)} = \frac{1}{N} \sum_{i=1}^{N} x_j^{(i)}$$

This result illustrates a fundamental similarity between the k-medoids and the k-means algorithm. Similar to k-means, the k-medoids algorithm produces clusters where cluster centers are data points close to the mean. The main difference between both algorithms lies in the degree to which they explore the configuration space of cluster centers. Cluster membership of any one data point is determined by the cluster boundaries that are, in turn, determined by neighboring cluster centers. Finding a global minimum of the cost function requires exploring the space of combined choices for cluster centers. Whereas k-medoids related techniques extensively search that space, k-means corresponds to a simple hill-climbing approach.

We can, therefore, see that the fundamental challenge of avoiding the complexity in k-medoids related techniques consists of eliminating the dependency of the definition of one cluster center on the position of all others. Going back to our analogy with a physical system, we can now look for modifications that will remove this dependency. A physical system that has the properties of the k-medoids algorithm would show a quadratic attraction of cluster centers to data points within the cluster and no attraction to data points outside the cluster. The key to making cluster centers independent from each other lies in defining an interaction that is independent of cluster boundaries. One may have the idea to achieve this goal by eliminating the restriction that only data points within the cluster interact. A simple consideration reveals that any superposition of quadratic functions will always result in a quadratic function that corresponds to the mean of the entire data set, which is hardly a useful result. It is therefore necessary to impose a range limit to the "interaction." We look for a potential that is quadratic for small distances and approaches a constant for large distances since constant potentials are irrelevant to the calculation of forces and can be ignored. A natural choice for such a potential is a Gaussian function; see Figure 7.1.

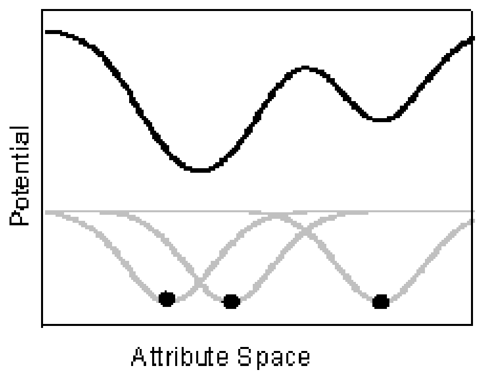


Figure 7.1. Energy landscape (black) and potential of individual data items (gray) for a Gaussian influence function.

Cluster centers are now determined as local minima in the potential landscape that is defined by a superposition of Gaussian functions for each data point. This approach is highly related to the density-based algorithm DENCLUE [2]. In DENCLUE, the Gaussian function is one possible choice for a so-called influence function that is used to model the overall density. We will take over the term "influence function" used in DENCLUE, noting that our influence function is the negative of DENCLUE’s. We would not expect the cluster centers in this approach to be identical to the k-medoid ones because a slightly different problem is solved, but both can be well motivated from the analogy with a physics problem. A particular benefit of using an influence function results from the fact that outliers have a less significant influence on the cluster center location. In partition-based clustering, any cluster member affects the cluster center location equally. Data points to the boundaries have equal weight to ones close to the centers. Using an influence function always determines influence based on the distance alone and, thereby, limits the influence of outliers and noise.

7.4. Hierarchical Algorithm

We have now motivated the following procedure for determining cluster centers. The clustering process is viewed as a search for equilibrium of cluster centers in an energy landscape that is given by the sum of the Gaussian influences of all data points $X^{(i)}$:

$$E(X) = -\sum_{i} e^{-\frac{(d(X,X^{(i)}))^2}{2\sigma^2}} ,$$

where the distance, d, is taken to be the Euclidean distance calculated in the d-dimensional space of all attributes

$$d(X, X^{(i)}) = \sqrt{\sum_{j=1}^{d} (x_j - x_j^{(i)})^2}$$

It is important to note that the minima of "potential energy" depend on the effective range of the interaction. This range is parameterized by the width of the Gaussian influence function, σ. The width of the Gaussian function, σ, specifies the range for which the potential approximates a quadratic function. It directly determines the minimum cluster size that can be found: two Gaussian functions have to be at least 2σ apart to be separated by a minimum; therefore, we cannot get clusters with a diameter smaller than 2σ. For areas in which data points are more widely spaced than 2σ, each data point would be considered an individual cluster. Such small clusters are undesirable except possibly at the leaf-level of the equilibrium tree since widely spaced points are likely to be due to noise. We will exclude such points from being cluster centers by limiting the search to areas with energy lower than a given threshold.

Rather than treating σ as a constant that has to be chosen by the user, we iterate through a sequence of values. This strategy is computationally inexpensive since we can use the configuration at iteration i, σ_i, as starting point for the next minimization. Choosing 2σ smaller than the smallest distance between data points would ensure that every data point forms its own cluster. This setup is the common starting point for hierarchical agglomerative methods such as Agnes [1]. In practice, it is sufficient to choose 2σ such that it is smaller than the smallest cluster size in which the user is likely to be interested. The cluster centers at this lowest level are considered the leaves of a tree while branches will join at higher levels for increasing σ. We will call the resulting structure the equilibrium tree of the data. We will now look at the details of our algorithm.

7.4.1. Defining Cluster Boundaries

So far, we have discussed how to find cluster centers but not yet cluster boundaries. A natural choice for cluster boundaries could be the ridges that separate minima. A corresponding approach is taken by center-defined DENCLUE. We decide against this strategy for practical and theoretical reasons. The practical reasons include that finding ridges would increase the computational complexity significantly. To understand our theoretical reasons, we have to look at the purpose of partitioning methods. Contrary to density-based methods, partition-based methods are commonly used to determine objects that represent a group–i.e., a cluster–of data items. For this purpose, it may be hard to argue that a data point should be considered as belonging to a cluster other than the one defined by the cluster center to which it is closest. If one cluster has many members in the data set that is used while clustering, it will appear larger than its neighbors with few members. Placing the cluster boundary according to the attraction regions would only be appropriate if we could be sure that the distribution will remain the same for any future data set. This assumption is a stronger than the general clustering hypothesis that the location of the cluster centers will be the same.

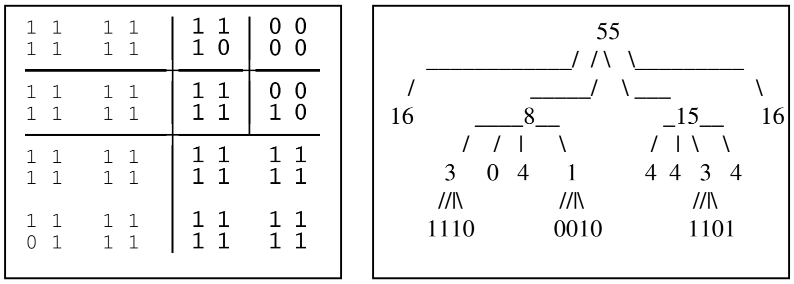
We will, therefore, use the k-means/k-medoids definition of a cluster boundary, which always places a data point with the cluster center to which it is closest. If the distance from a data point to the nearest cluster center exceeds a predefined threshold, the data point is considered an outlier. This approach avoids the expensive step of precisely mapping out the shape of clusters. Furthermore, it simplifies the process of building the hierarchy of equilibrium cluster centers. Cluster membership of a data point can be determined by a simple distance calculation without the need of referring to an extensive map.

7.5. Algorithm

To treat clustering as a search for equilibrium cluster centers requires us to have a fast method of finding data points based on their feature attribute values. Density-based algorithms such as DENCLUE achieve this goal by saving data in a special data structure that allows referring to neighbors. We use a data structure, Peano Count Tree (or P-tree) [9-13], that allows fast calculation of counts of data points based on their attribute values.

7.5.1. A Summary of P-tree Features

Many types of data show continuity in dimensions that are not used as data mining attributes. Spatial data that are mined independently of location will consist of large areas of similar attribute values. Data streams and many types of multimedia data, such as videos, show a similar continuity in their temporal dimension. Peano Count Trees are constructed from the sequences of individual bits; i.e., 8 P-trees are constructed for byte-valued data. Each node in a P-tree corresponds to one quadrant, with children representing sub-quadrants. Compression is achieved by eliminating quadrants that consist entirely of 0 or 1 values. Two and more dimensional data are traversed in Peano order, or recursive raster order, which ensures that continuity in all dimensions benefits compression equally. Counts are maintained for every quadrant. The P-tree for an 8-row-8-column bit-band is shown in Figure 7.2. In this example, the root count is 55, and the counts at the next level, 16, 8, 15, and 16, are the 1-bit counts for the 4 major quadrants. Since the first and last quadrant is made up of entirely 1-bits, we do not need sub-trees for these two quadrants.



```
1 1  1 1 | 1 1 | 0 0
1 1  1 1 | 1 0 | 0 0
---------------------
1 1  1 1 | 1 1 | 0 0
1 1  1 1 | 1 1 | 1 0
1 1  1 1 | 1 1   1 1
1 1  1 1 | 1 1   1 1

1 1  1 1 | 1 1   1 1
0 1  1 1 | 1 1   1 1
```

```
                         55
           ___________/ /\ \________
   /                 _____/  \___         \
16            ____8__              _15__     16
           /  /  |    \           /  | \  \
         3    0  4    1          4  4  3  4
        //|\          //|\            //|\
       1110          0010           1101
```

Figure 7.2. 8-by-8 image and its P-tree.

A major benefit of the P-tree structure lies in the fact that calculations can be done in the compressed form, which allows efficiently evaluating counts of attribute values or attribute ranges through an “AND” operation on all relevant P-trees.

7.5.2. Energy Evaluation Using the HOBBit Intervals

The first step in our algorithm consists of constructing P-trees from the original data. It is important to note that P-trees cannot only be used for clustering, but also for classification [10] and association rule mining [11]. Starting points for the minimization are selected randomly, although other strategies would be possible. As a minimization step, we evaluate neighboring points, with a distance (step size) *s*, in the energy landscape. This step requires the fast evaluation of a superposition of Gaussian functions. We intervalize distances and then calculate counts within the respective intervals. We use equality in the High Order Basic Bit, or HOBBit, distance [10] to define intervals. The HOBBit distance between two integer coordinates is equal to the number of digits by which they have to be right-shifted to make them identical. The HOBBit distance of the binary numbers 11001 and 11011, for example, would be 2. The number of intervals that have distinct HOBBit distances is equal to the number of bits of the represented numbers. For more than one attribute, the HOBBit distance is defined as the maximum of the individual HOBBit distances of the attributes. The range of all data items with a HOBBit distance smaller than or equal to d_H corresponds to a *d*-dimensional hyper cube. Counts within these hyper cubes can be efficiently calculated by an "AND" operation on P-trees.

We start with 2σ being smaller than any cluster center in which we are interested. We then pick a potential cluster center and minimize its energy. The minimization is done in standard valley decent fashion: we calculate the energy for its location in attribute space and compare it with the energies for neighboring points in each dimension (distance *s*). We replace the central point with the point that has the lowest energy. If no point has lower energy, step size *s* is reduced. If the step size is already at its minimum, we consider the point as a leaf in the hierarchy of cluster centers. If the minimum we find is already listed as a leaf in the hierarchy, we ignore it. If we repeatedly rediscover old cluster centers, we accept the leaf level of the hierarchy as complete.

Further steps in the hierarchy are done analogously. We chose a new, larger σ. Each of the lower-level cluster centers is considered a starting point for the minimization. If cluster centers move to the same minimum, they are considered a single node of the hierarchy.

7.6. Performance Analysis

We tested speed and effectiveness of our algorithm by comparing with the result of using k-means clustering. For storage requirements of P-trees we refer to [13]. We used synthetic data with 10% noise. The data were generated with no assumptions on continuity in the structural dimension (e.g., location for spatial data or time for multimedia data). Such continuity would significantly benefit the use of P-tree methods. The speed demonstrated in this section can, therefore, be seen as an upper bound to the time complexity. Speed comparison was done on data with 3 attributes for a range of data set sizes. Figure 7.3 shows the time that is required to find the equilibrium location of one cluster center with an arbitrary starting location (leaf node) and with a starting location that is an equilibrium point for a smaller σ (internal node). We compare with the time that is required to find a k-means clustering for a particular k (k=2 in Figure 7.3) divided by k. It can be seen that our algorithm shows the same or better scaling and approximately the same speed as k-means. Since k-means is known to be significantly faster than k-medoids-based algorithms, this performance can be considered highly competitive with partition-based methods.

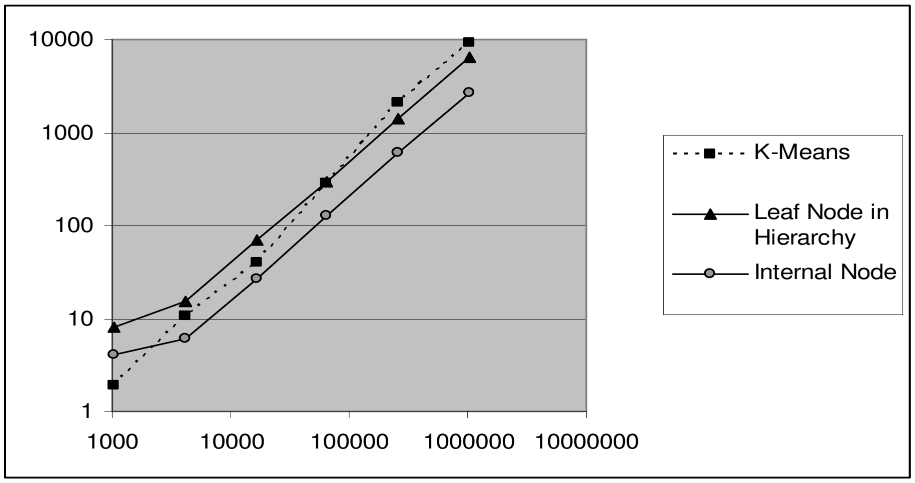


Figure 7.3. Speed comparison between our approach and k-means.

To determine clustering effectiveness, we used a small data set with two attributes and visually compared results with the histogram in Figure 7.4.

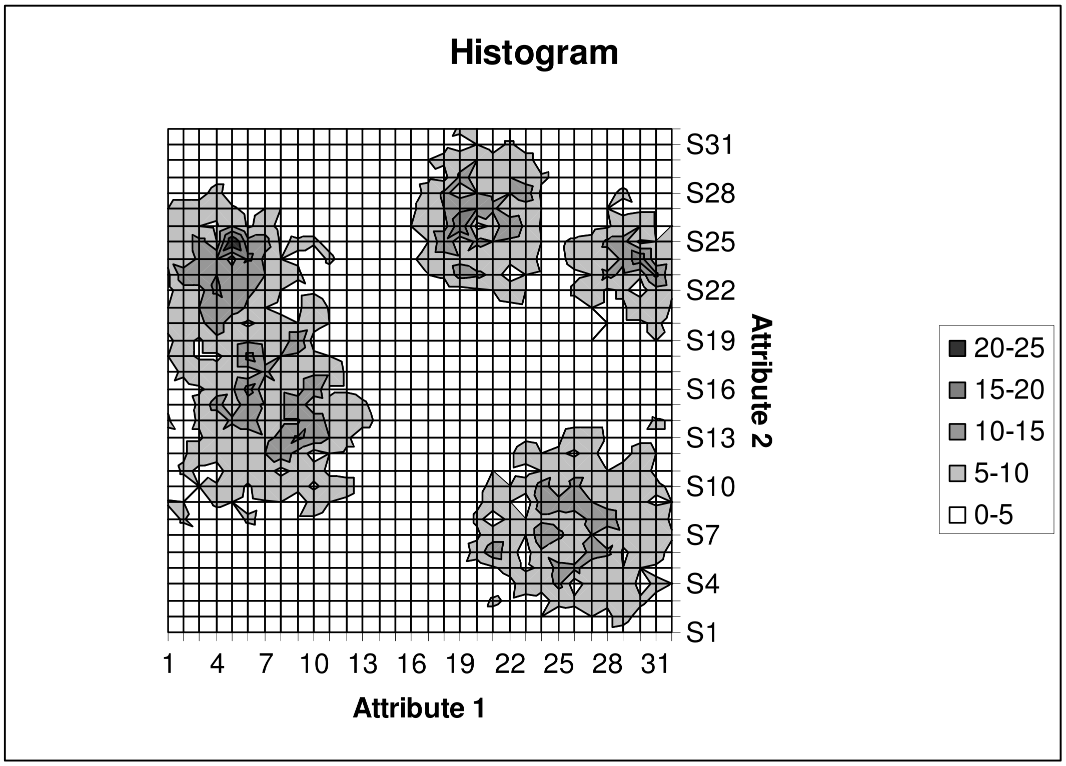


Figure 7.4. Histogram of values used to compare our approach with k-means.

Quantitative measurements, such as total squared distance of data points from the cluster center, cannot be compared between our algorithm and k-means because outliers are treated differently. Table 7.1 compares the cluster centers of k-means for k = 5 with those found by our algorithm. It can clearly be seen that the k-means results are influenced more by the noise between the identifiable clusters than the results of our algorithm.

Table 7.1. Comparison of cluster centers for the data set of Figure 7.3.

k-means (k=5)	<11.11>	<4, 12>	<25, 6>	<4, 22>	<23, 23>
our algorithm	<9, 11>	<27, 22>	<24, 6>	<4, 21>	<18, 25>

7.7. Conclusions

We have developed a hierarchical clustering algorithm that is based on some of the same premises as well-known partition- and density-based techniques. The time-complexity of k-medoids related algorithms is avoided in a systematic way and the influence of outliers reduced. The hierarchical organization of data represents information at any desired level of granularity and relieves the user from the necessity of selecting parameters prior to clustering. Different levels in the hierarchy are efficiently calculated by using lower-level solutions as starting points for the computation of higher-level cluster centers. We use the P-tree data structure for efficient storage and access of data. Comparison with k-means shows that we can achieve the benefits of improved outlier handling without sacrificing performance.

7.8. References

[1] L. Kaufman and P.J. Rousseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis," John Wiley & Sons, New York, 1990.

[2] A. Hinneburg and D. A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," Int. Conf. Knowledge Discovery and Data Mining (KDD'98), pp. 58-65, New York, Aug. 1998.

[3] M. Ester, H.-P. Kriegel, and J.Sander, "Spatial Data Mining: A Database Approach," Int. Symp. Large Spatial Databases (SSD'97), pp. 47-66, Berlin, Germany, July 1997.

[4] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," Int. Conf. Management of Data (SIGMOD'96), pp. 103-114, Montreal, Canada, June 1996.

[5] R. Ng and J. Han, "Efficient and Effective Clustering Method for Spatial Data Mining," 1994 Int. Conf. Very Large Data Bases (VLDB'94), pp. 144-155, Santiago, Chile, Sept. 1994.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification," 4th Int. Symp. Large Spatial Databases (SSD'95), pp. 67-82, Portland, ME, Aug. 1995.

[7] P. Bradley, U. Fayyard, and C. Reina, "Scaling Clustering Algorithms to Large Databases," 1998 Int. Conf. Knowledge Discovery and Data Mining (KDD'98), pp. 9-15, New York, Aug. 1998.

[8] M. Dash, H. Liu, and X. Xu, "1+1>2: Merging Distance and Density Based Clustering," Conf. on Database Systems for Advanced Applications, pp. 32-39, 2001.

[9] Q. Ding, M. Khan, A. Roy, and W. Perrizo, "P-tree Algebra," ACM Symposium on Applied Computing (SAC'02), Madrid, Spain, 2002.

[10] M. Khan, Q. Ding, and W. Perrizo, "K-nearest Neighbor Classification of Spatial Data Streams Using P-trees," PAKDD-2002, Taipei, Taiwan, May 2002.

[11] Q. Ding, Q. Ding, and W. Perrizo, "Association Rule Mining on Remotely Sensed Images Using P-trees," Pacific Asian Conference on Knowledge Discovery and Data Mining (PAKDD-2002), Taipei, Taiwan, 2002.

[12] Q. Ding, W. Perrizo, and Q. Ding, "On Mining Satellite and other Remotely Sensed Images," Workshop on Data Mining and Knowledge Discovery (DMKD-2001), pp. 33-40, Santa Barbara, CA, 2001.

[13] A. Roy, "Implementation of Peano Count Tree and Fast P-tree Algebra," M.S. thesis, North Dakota State University, Fargo, North Dakota, 2001.