Efficient Density Clustering Method for Spatial Data

Fei Pan, Baoying Wang, Yi Zhang, Dongmei Ren, Xin Hu, and William Perrizo

Computer Science Department
North Dakota State University
Fargo, ND 58105
{fei.pan,baoying.wang,yi.zhang,dongmei.ren,xin.hu,
William.perrizo}@ndsu.nodak.edu
Tel: (701) 231-6403/6257
Fax: (701) 231-8255

Abstract. Data mining for spatial data has become increasingly important as more and more organizations are exposed to spatial data from sources such as remote sensing, geographical information systems, astronomy, computer cartography, environmental assessment and planning, etc. Recently, density based clustering methods, such as DENCLUE, DBSCAN, OPTICS, have been published and recognized as powerful clustering methods for data mining. These approaches have run time complexity of $O(n\log n)$ when using spatial index techniques, R^* tree and grid cell. However, these methods are known to lack scalability with respect to dimensionality. In this paper, a unique approach to efficient neighborhood search and a new efficient density based clustering algorithm using EIN-rings are developed. Our approach exploits compressed vertical data structures, Peano Trees (P-trees[1]), and fast P-tree logical operations to accelerate the calculation of the density function within EIN-rings. This approach stands in contrast to the ubiquitous approach of vertically scanning horizontal data structures (records). The average run time complexity of our algorithm for spatial data in d-dimension is $O(dn\sqrt{n})$. Our proposed method has comparable cardinality scalability with other density methods for small and medium size of data, but superior speed and dimensional scalability.

1 Introduction

With the rapid growth of large quantities of spatial data collected in various application areas, such as remote sensing, geographical information systems, astronomy, computer cartography, environmental assessment and planning, efficient spatial data mining methods are in great demand. Density based cluster algorithms have been widely used in the mining of large spatial data. Density based cluster algorithms group the attribute objects into a set of connected dense components separated by regions of low density. A cluster is regarded as a connected dense region of objects, which grows in any direction that density leads. Density based cluster algorithms have

[1] Patents are pending on the P-tree technology. This work is partially supported by GSA Grant ACT#: K96130308.

N. Lavrač et al. (Eds.): PKDD 2003, LNAI 2838, pp. 375–386, 2003.

been recognized as a powerful clustering approach capable of discovering arbitrary shape of clusters as well as dealing with noise and outliers for spatial data mining.

There are two major approaches for density-based methods. The first approach is represented by DENCLUE [3]. It exploits a density function, e.g., step function or Gaussian function to measure the density in attribute metric space. Clusters are identified by determining corresponding density attractors. Thus, clusters of arbitrary shape can be easily determined by overall density functions. This algorithm scales well with run time complexity $O(n\log n)$ by means of grid cells techniques. However, it requires careful selection of the density parameter σ and noise threshold ξ, which may significantly influence the quality of the clustering results [10].

The second approach calculates the density of all data points and groups them based on density connectivity. Typical algorithms in this approach include DBSCAN [6] and OPTICS [8]. DBSCAN first defines a core object as a set of neighbor points consisting of more than a specified number of data points. All the data points reachable within a chain of overlapping core objects define a cluster. The run time complexity of DBSCAN is $O(n\log n)$ for spatial data when using a spatial index. Otherwise, it is $O(n^2)$ [10]. OPTICS can be considered as an extension of DBSCAN without providing global density. It assumes each cluster has its own density parameter and uses a random variable to learn its probability distribution. It has the same run time complexity as DBSCAN, that is, $O(n\log n)$ if a spatial index is used and $O(n^2)$ otherwise.

However, the spatial index techniques, such as R tree, R^+ tree, and grid cell, are known to be suitable for low dimensional data sets. They perform well in 2-3 dimensions. In high dimensional spaces they exhibit poor behavior in the worst case and in typical cases as well [0]. The reason is that the data space becomes sparse at high dimensionalities causing the bounding regions to become large. In this paper, a unique approach to efficient neighborhood search using EIN-rings, and a new efficient density based clustering algorithm are developed. The center idea is to make use of P-trees and EIN-rings to calculate the density function in $O(\sqrt{n})$ time, on the average. Our approach exploits compressed vertical data structures, Peano Trees (P-trees), and fast P-tree logical operations to accelerate the calculation of the density function within EIN-rings. This approach stands in contrast to the ubiquitous approach of vertically scanning horizontal data structures (records). Furthermore, we adopt a ***look around*** pruning method to combine the density calculation and a hill climbing technique. The overall run time complexity is $O(dn\sqrt{n})$ for a d-dimensional data set, on the average. Experimental results show that the algorithm works efficiently on large-scale, high-dimensional spatial data, outperforming other density methods significantly.

This paper is organized as follows. In section 2, we first briefly review the basic P-trees, and then present a variation of P-tree, range predicate tree. In section 3, we define a unique equal interval neighborhood rings, EIN-rings, and then present the new efficient density clustering method using EIN-rings. Finally, we compare our method with other density methods experimentally in section 4 and conclude the paper in section 5.

2 Extended Peano Trees

A new tree structure, the Peano tree (P-tree), was developed to facilitate efficient data mining [1][2]. In this section, we first briefly review the basic P-trees, and then develop a new calculation method of a variation of P-tree, range predicate trees. In this paper, we use $\wedge$, $\vee$ and prime (') to denote P-tree operations AND, OR and NOT, respectively.

2.1 Review of Basic Peano Trees

A basic P-tree is a lossless, bitwise, vertical quadrant-based compressed tree, which can be 1-dimensional, 2-dimensional, 3-dimensional, etc. For a data set with d feature attributes, $X = (A_1, A_2 \ldots A_d)$, and the binary representation of j^{th} feature attribute A_j as $b_{j,m}b_{j,m-1}...b_{j,i}\ldots b_{j,1}b_{j,0}$, we strip each feature attribute into several files, one file for each bit position. Such files are called bit files. A bit file is then recursively partitioned into quadrants and each quadrant into sub-quadrants until the sub-quadrant is pure (entirely 1-bits or entirely 0-bits). The recursive raster ordering is called the Peano or Z-ordering in the literature – therefore, the name Peano tree.

We illustrate the detailed construction of P-trees using an example shown in Fig.1. The spatial data is the red reflective value of a 2-dimensional spatial data, which is shown in a). We represent the reflectance as binary values, e.g., $(7)_{10} = (111)_2$. Then strip them into three separate bit files, one file for each bit, as shown in b), c), and d). The corresponding basic P-trees, P_1, P_2 and P_3, are constructed by recursive partition, which are shown in e), f) and g).

As shown in e) of Fig.1, the root of P_1 tree is 36, which is the 1-bit count of the entire bit file-1. The second level of P_1 contains the 1-bit counts of the four quadrants, 16, 7, 13, and 0. Since quadrant 0 and quadrant 3 are pure, there is no need to partition these quadrants. Quadrant 1 and 2 are further partitioned recursively. We note here that we identify quadrants using a Quadrant identifier, Qid - the string of successive sub-quadrant numbers (01,2 or 3 in Z or Peano order, separated by "." (as in IP addresses). Thus, the Qid of the bolded and underlined quadrant in Fig.1 is 2.2.

AND, OR and NOT logic operations are the most frequently used P-tree operations. The P-tree logical operations are performed level-by-level starting from the root level. They are commutative and distributive, since they are simply pruned bit-by-bit operations. For instance, ANDing a pure-0 node with anything results in a pure-0 node, ORing a pure-1 node with anything results in a pure-1 node.

2.2 Range Predicate Trees

Range predicate tree, $P_{x \prec y}$, is a basic P-tree that satisfies predicate $x \prec y$, where y is a boundary value, and $\prec$ is the comparison operator, i.e., $<$, $>$, $\geq$, and $\leq$. Without loss of generality, we only present the calculation of range predicate $P_{A>c}$, $P_{A\leq c}$ and their proof as follows.

Lemma 1. Complement Rule of P-tree Let P_1, P_2 be basic P-trees, and P_1' is the complement P-tree of P_1, then $P_1 \vee (P_1' \wedge P_2) = P_1 \vee P_2$.

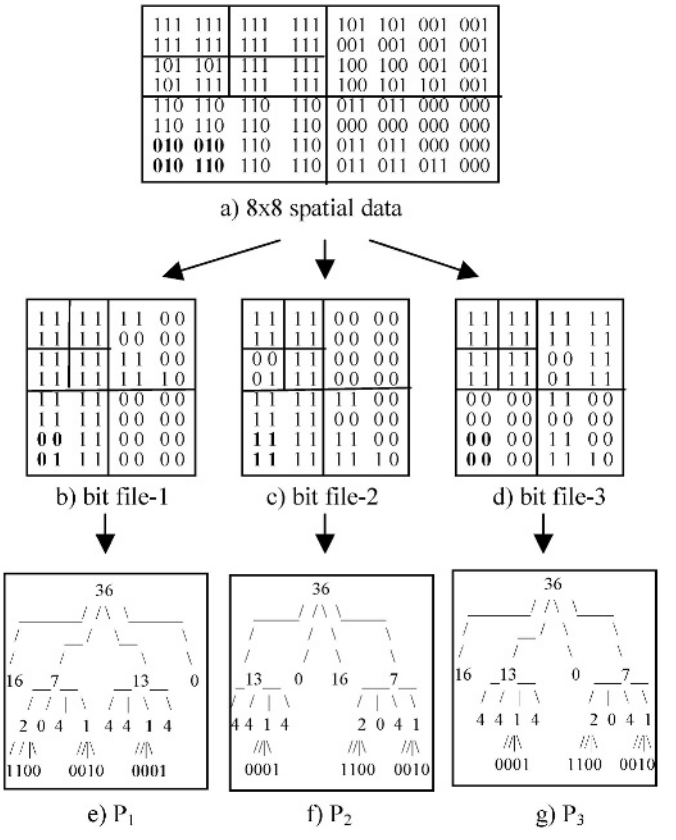


Fig. 1. Construction of 2-D Basic P-trees for Spatial Data.

Proof:

$P_1 \vee (P_1' \wedge P_2)$

(according to the distribution property of P-tree operations)

$= (P_1 \vee P_1') \wedge (P_1 \vee P_2)$

$= \text{True} \wedge (P_1 \vee P_2)$

$= P_1 \vee P_2$

Proposition 1. Let A be j^{th} attribute of data set X, m be its bit-width, and P_m, P_{m-1}, … P_0 be the basic P-trees for the vertical bit files of A. Let $c=b_m \ldots b_i \ldots b_0$, where b_i is i^{th} binary bit value of c, and $P_{A>c}$ be the predicate tree for the predicate A>c, then

$$P_{A>c} = P_m \, op_m \ldots P_i \, op_i \, P_{i-1} \ldots op_{k+1} \, P_k, \quad k \le i \le m, \tag{1}$$

where 1) op_i is $\wedge$ if $b_i=1$, op_i is $\vee$ otherwise, 2) k is the rightmost bit position with value of "0", i.e., $b_k=0$, $b_j=1, \forall\, j<k$, and 3) the operators are right binding. Here the

right binding means operators are associated from right to left, e.g., $P_2\ op_2\ P_1\ op_1\ P_0$ is equivalent to $(P_2\ op_2\ (P_1\ op_1\ P_0))$.

Proof (by induction on number of bits):

Base case: without loss of generality, assume b_0=1, then need show $P_{A>c} = P_1\ op_1\ P_0$ holds. If b_1=1, obviously the predicate tree for $A>(11)_2$ is $P_{A>c} = P_1 \wedge P_0$. If b_1=0, the predicate tree for $A>(01)_2$ is $P_{A>c} = P_1 \vee (P_1' \wedge P_0)$. According to Lemma 1, we get $P_{A>c} = P_1 \vee P_0$ holds.

Inductive step: assume $P_{A>c} = P_n\ op_n \ldots P_k$, we need to show $P_{A>c} = P_{n+1} op_{n+1} P_n\ op_n \ldots P_k$ holds. Let $P_{right} = P_n\ op_n \ldots P_k$, if b_{n+1}=1, then obviously $P_{A>c} = P_{n+1} \wedge P_{right}$. If b_{n+1}= 0, then $P_{A>c} = P_{n+1} \vee (P'_{n+1} \wedge P_{right})$. According to Lemma 1, we get $P_{A>c} = P_{n+1} \vee P_{right}$ holds.

Proposition 2. Let A be j^{th} attribute of data set X, m be its bit-width, and $P_m, P_{m-1}, \ldots P_0$ be the basic P-trees for the vertical bit files of A. Let $c=b_m \ldots b_i \ldots b_0$, where b_i is i^{th} binary bit value of c, and $P_{A\le r}$ be the predicate tree for A≤c, then

$$P_{A\le c} = P'_m op_m \ldots P'_i\ op_i\ P'_{i-1} \ldots op_{k+1} P'_k, \quad k \le i \le m, \tag{2}$$

where 1). op_i is $\wedge$ if b_i=0, op_i is $\vee$ otherwise, 2) k is the rightmost bit position with value of "0", i.e., b_k=0, b_j=1, $\forall$ j<k, and 3) the operators are right binding.

Proof (by induction on number of bits):

Base case: without loss of generality, assume b_0=0, then need show $P_{A\le c} = P'_1\ op_1\ P'_0$ holds. If b_1=0, obviously the predicate tree for $A\le(00)_2$ is $P_{A\le c} = P'_1 \wedge P'_0$. If b_1=1, the predicate tree for $A\le(10)_2$ is $P_{A\le c} = P'_1 \vee (P_1 \wedge P'_0)$. According to Lemma 1, we get $P_{A\le c} = P'_1 \vee P'_0$ holds.

Inductive step: assume $P_{A\le c} = P'_n\ op_n \ldots P'_k$, we need to show $P_{A\le c} = P'_{n+1} op_{n+1} P'_n\ op_n \ldots P'_k$ holds. Let $P_{right} = P'_n\ op_n \ldots P'_k$, if b_{n+1}=0, then obviously $P_{A\le c} = P'_{n+1} \wedge P_{right}$. If b_{n+1}= 1, then $P_{A\le c} = P'_{n+1} \vee (P_{n+1} \wedge P_{right})$. According to Lemma 1, we get $P_{A\le c} = P'_{n+1} \vee P_{right}$ holds.

Theorem 1. **Complement Rule** Let A be j^{th} attribute of data set X, $P_{A\le c}$ and $P_{A>c}$ are the predicate tree for A≤c and A>c, where c is a boundary value, then $P_{A\le c} = P'_{A>c}$.

Proof: It is obvious. This theorem can be exploited to reduce computation time of predicate trees.

3 The EIN-Ring Based Density Clustering Algorithm

In this section, we present an EIN-ring based Density Clustering approach (EDC). We first define neighborhood rings and equal interval neighborhood ring (EIN-ring), and then describe the approach of calculation of EIN-ring using P-trees. In section 3.2, we describe calculation of the density function using EIN-rings. In section 3.3, the algorithm for finding density attractors is developed. Finally, the efficiency of our algorithm is analyzed in terms of time complexity.

3.1 EIN-Ring Based Neighborhood Search

Definition 1. The Neighborhood Ring of data point c with radii r_1 and r_2 is defined as the set $R(c, r_1, r_2) = \{x \in X \mid r_1 < |c-x| \leq r_2\}$, where $|c-x|$ is the distance between x and c.

Definition 2. The Equal Interval Neighborhood Ring of data point c with radii r and fixed interval λ is defined as the neighborhood ring $R(c, r, r+\lambda) = \{x \in X \mid r < |c-x| \leq r+\lambda\}$, where $|c-x|$ is the distance between x and c.

The interval λ is a user-defined parameter based on accuracy requirements. The higher the accuracy requirement, the smaller the interval. For $r = k\lambda$, k=1,2,…, the rings called the ***k*th *EIN-rings***. Fig.2 shows 2-D EIN-rings with k = 1, 2, and 3.

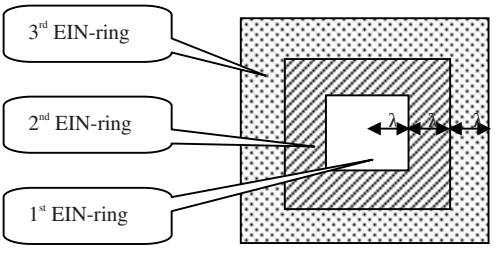


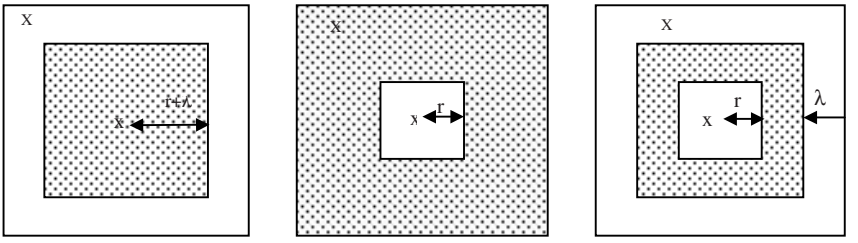
Fig. 2. Diagram of EIN-rings.

The calculation of neighbors within EIN-ring $R(x, r, r+\lambda)$ is as follows. Let $P_{r,\lambda}$ be the P-tree representing data points within EIN-ring $R(x, r, r+\lambda)$. We note $P_{r,\lambda}$ is just the predicate tree corresponding to the predicate $x-r-\lambda<X\leq x-r$ or $x+r<X\leq x+r+\lambda$. We first calculate the data points within neighborhood ring R(x, 0, r) and $R(x, 0, r+\lambda)$ by $P_{x-r<X\leq x+r}$ and $P'_{x-r-\lambda<X\leq x+r+\lambda}$ respectively. $P_{x-r<X\leq x+r}$ is shown as the shadow area of a) and $P'_{x-r-\lambda<X\leq x+r+\lambda}$ is the shadow area of b) in Fig.3. The data points within the EIN-ring $R(x, r, r+\lambda)$ are those that are in $R(x, 0, r+\lambda)$ but not in R(x, 0, r). Therefore $P_{r,\lambda}$ is calculated by the following formula, which is the shadow area shown in c) of Fig.3.

$$P_{r,\lambda} = P_{x-r-\lambda<X\leq x+r+\lambda} \wedge P'_{x-r<X\leq x+r} \tag{3}$$

3.2 Calculation of the Density Function Using EIN-Ring

Density based clustering algorithm is a clustering method based on a set of density distribute function, called an influence function, which describes the impact of a data point within its neighborhood. Our algorithm employs a special EIN-ring based influence function. The overall density of the data space is then modeled as the sum of the influence functions of all data points. Clusters are determined by identifying density attractors, where density attractors are local maxima of the overall density function.



a) $P_{x-r-\lambda<X\leq x+r+\lambda}$ b) $P'_{x-r<X\leq x+r}$ c) $P_{x-r-\lambda<X\leq x+r+\lambda} \wedge P'_{x-r<X\leq x+r}$

Fig. 3. Calculation of Data Points within EIN-ring R(x, r, r+λ).

Let x and y be data points in F^d, a d-dimensional feature space. The influence function of the data point y on x is a function $f_H^y: F^d \rightarrow R_0^+$, which is defined based on EIN-ring:

$$\begin{aligned} f_{r1,r2}^{\ y}(x)&=1 \qquad \text{if } y\in R(c, r_1, r_2) \\ &=0 \qquad \text{if } y\notin R(c, r_1, r_2)\ . \end{aligned} \tag{4}$$

The EIN-ring based density function of x is defined as the weighted summation of RC(x,r), which is calculated as follows

$$\begin{aligned} f_h^D(x) &= \sum_{r=1}^{m} w_r * f_{r1,r2}^{\ y}(x) \\ &= \sum_{r=1}^{m} w_r * RC(x,r)\ . \end{aligned} \tag{5}$$

where $f_h^D(x)$ denotes the EIN-ring based density of data point x, with respect to weights, w_r. The selection of this weight is based on a RBF kernel function of the radius of EIN-ring, such as Gaussian function, step function, etc.

3.3 Finding Density Attractors Using the *Look Around* Pruning Technique

Once the density of each data point is defined, the next step is to define *density attractors*, i.e., local maxima of the overall density function. Having a high density doesn't necessarily make a point a density attractor - it must have the highest density among its neighbors. Instead of using formal hill climbing as is done in DENCLUE [3], we adopt a simpler heuristic ***look around*** technique.

Algorithm 1. Look Around Pruning We first define a neighborhood as a ball of some chosen radius r. The number r can range from 0 to the maximal bit length of the attributes. After finding the density function D_x of a point, x, we compare that density with that of data points within its neighborhood. If it is greater than the density of all

its neighbors, it is labeled as a new density attractor. Any old density attractor in that neighborhood is de-labeled as a density attractor.

After all the data points have gone through the process above, we have a set of *intermediate* density attractors. We compare each intermediate attractor's density with that of its nearest neighbor data point. If the former is less than the latter, the attractor is de-labeled. Otherwise, it is a final density attractor. This step finds attractors that are isolated and therefore should be removed as noise.

Definition 3. Density Attractor Set Given a sequence of points $x_1, x_2, \ldots x_n$, the Density Attractor Set DAS $(x_1, x_2, \ldots x_n)$ is a set of attractors produced by the look around algorithm applied to the data points in the order, $x_1, x_2, \ldots x_n$.

Definition 4. A data point x is ***reachable*** from data point y if $x \in R(y, 0, r)$, where r is the user-defined radius for the density clustering. If x is reachable from y, y is also reachable from x.

The ***look around*** pruning algorithm is robust, which means the clustering results are independent of data point treating order. The proof is given as follows.

Lemma 2. (Density Characterization Lemma) Data point y is a density attractor iff $Dy \geq Dz$, $\forall z \in R(y,0,r)$. If y is not a density attractor, $\exists z \in R(y, 0, r)$ ∋ : $Dz > Dy$.

Lemma 3. Given a data point y, and $Dy \geq Dz$, $\forall z \in R(y, 0, r)$, y is the density attractor independent of the order in which y and z are treated in the ***look around*** process.

Proof (Proof by contradiction):

Assume the statement is not true, i.e. y is an attractor, but $\exists z \in R(y, 0, r)$, ∋ : $Dz > Dy$. If z is treated first and z is an attractor, then when y gets treated, y would not be an attractor (Lemma 3.2.1). If y is treated before z, y could be designated an attractor at that time. But when z gets treated, y will be de-labeled according to ***look around*** pruning algorithm 3.2.1. Therefore y is not an attractor. Contradiction!

Theorem 2. Given data set X in two different sequences: $\{x_{i1}, x_{i2}, \ldots x_{in}\}$ and $\{ x_{j1}, x_{j2}, \ldots, x_{jn}\}$, then $DAS(x_{i1}, x_{i2}, \ldots x_{in}) = DAS(x_{j1}, x_{j2}, \ldots x_{jn})$.

Proof (Proof by contradiction):

Assume the statement is not true, i.e. $DAS(x_{i1}, x_{i2}, \ldots x_{in}) \neq DAS(x_{j1}, x_{j2}, \ldots x_{jn})$. That means $\exists x \in DAS(x_{i1}, x_{i2}, \ldots x_{in})$ but $x \notin DAS(x_{j1}, x_{j2}, \ldots x_{jn})$. According to $x \in DAS(x_{i1}, x_{i2}, \ldots x_{in})$ and Lemma 3.2.1, $\forall z \in R(x,0,r)\ Dx \geq Dz$. Also according to $x \notin DAS(x_{j1}, x_{j2}, \ldots x_{jn})$ and Lemma 3.2.1, $\exists z \in R(x,0,r)$ ∋ : $Dz > Dx$. Contradiction!

We illustrate the finding of density attractors using look around pruning algorithm as follows. Suppose Qid of data point X is 0.3.2 and $D_x = 250$. We need compare D_x with the neighbor's density. From the Px,σ, x has four neighbors with Qids of 0.0.2, 0.3.1, 2.3.0 and 2.3.3. If densities of these points are respectively 300, 0, 220 and 0, and 0.0.2 and 2.3.0 are labeled as density attractors. By comparing Dx with the maximal density of 0.0.2 and 2.3.0, 250 < max(300, 220), therefore we determine that x is not a density attractor. Otherwise if $D_x = 350$, 350 > max (300, 220), x is labeled as the new density attractor. The old density attractors 0.0.2 and 2.3.0 are de-labeled and will not be considered later. Finally, clusters are determined by the density attractors. The pseudo code of overall algorithm is shown in Fig. 4.

```
INPUT: P-tree Set P_ij for bit j and attribute i, HOBBit ring R(i, 0, σ)
OUTPUT: Density attractors
// P_i,j – P-tree for attribute i and bit j;
// P_i – Neighborhood P-tree;
// N - # of data points; n - # of attributes;
// flag[i] – label array of cluster center of data point i.
BEGIN
    FOR i=1 to N DO
        flag[i]← 0
        P_i ← Pure1 P-tree, DENS[i] ← 0, PrevRC ← 0
        FOR h = 1 TO m - 1 DO
            P_v ← Pure1 P-tree
            FOR j = 1 TO n DO
                GET b_jh[i]
                IF b_jh[i] = 1
                    PX_jh ← P_j,h
                ELSE
                    PX_jh ← P`_j,h
                P_v[h] ← P_v [h] & PX_jh

            END FOR
            P_i ← P_i & P_v [h]
            w [i] = h * 2^(h*n)
            DENS[i]← DENS[i]+ w * (RootCount(P_i )- PrevRC);

            PrevRC ← RootCount(P_i);

            IF h = m - σ
                P_σ ← P_i
        END FOR
        IF DENS[i] > the density of attractors within neighborhood ,
            flag[i] ← 1, clear the flags of its neighbors.

    END FOR
    // Final look around pruning to intermediate attractors
    FOR i = 1 to N DO
    IF flag[i] = 1DENS[i] < The density of the closest neighbor
        Clear its flag
    END FOR
```

Fig. 4. EIN-ring base density Clustering algorithm

3.4 Time Complexity Analysis

Let f be the fan-out of a P-tree and let n be the number of data points it represents. We first present some Lemmas on P-trees, and then derive the average run time complexity to be $O(n\sqrt{n})$.

Lemma 4. The number of level of P-tree k = log(f) n

Proof: The numbers of nodes in each level of P-trees are: 1, f, f^2, f^3, … f^k. Obviously the leaf level k is n bits long, i.e. f^k = n. Thus k = log(f) n.

Lemma 5. The maximum number of nodes in P-tree in the worst case $\eta = (n - 1) / (f - 1)$.

Proof: Without compression, the total number of nodes is $\eta = 1 + f + f^2 + f^3 + \ldots f^{k-1} = (f^k - 1) / (f - 1)$. According to Lemma 3.3.1, $f^k = n$, we get
$\eta = (n - 1) / (f - 1)$

Lemma 6. Total number of nodes in a P-tree with a compression ratio of ρ ($\rho<1$) is $\eta = 1 + (\rho^k * n - f) / (f * \rho - 1)$, where k is the number of levels of P-tree.

Proof: The numbers of nodes in each level of a P-tree with compression ratio ρ at level i is $f^i * \rho^{i-1}$., where i ranges from 1 to k.. For example, at level 2, there are $(f * \rho)* f = f^2 * \rho$ nodes. We get the total number of nodes in the case that the P-tree has a compression ratio of ρ as

$$\begin{aligned}\eta &= 1 + f + f^2 * \rho + f^3 * \rho^2 + \ldots + f^{k-1} * \rho^{k-2} \\ &= 1 + f * (f^{k-1} * \rho^{k-1} - 1) / (f * \rho - 1) \\ &= 1 + (f^k * \rho^k - f) / (f * \rho - 1) \\ &= 1 + (\rho^k * n - f) / (f * \rho - 1)\end{aligned}$$

Corollary 1. When $\rho = 0$, the total number of nodes in the P-tree is 1; when $\rho = 1$, the total number of nodes in the P-tree is $(n - f) (f - 1) + 1$. When $\rho = 0.5$ and $f = 4$, the total number of nodes in a P-tree with compression ratio η is

$$\begin{aligned}\eta &= 1 + (4^k/2^k - 2 * 4) / (4 - 2) \\ &= 1 + (4^{k/2} - 8) /2 \\ &= 1 + (\sqrt{n} - 8) /2\end{aligned}$$

Theorem 3. The average run time complexity of EDC with compression ratio 0.5 and fan-out 4 is O (d*n * $\sqrt{n}$), where d is the number of dimensions.

Proof: The P-tree ANDing operation is executed node by node when calculating the density. Each node ANDing is counted as one operation. For n data points in d-dimension, there are d*m basic P-trees, here m is the maximal bit size of each dimension. The total run time to get density P-trees is d*m*n*η, where η is the total number of nodes of a P-tree.

For data sets with fan-out $f = 4$ and average compress rate $\rho = 0.5$, according to Corollary 3.4.1, the total number of nodes of a P-tree $\eta = 1 + (\sqrt{n} - 8) /2$. Therefore, the total time to get the density for n data points in d-dimension is d*m*n * (1 + ($\sqrt{n}$ - 8) /2). Thus, the average time complexity of density based clustering using P-tree with compression ratio 0.5 and fan-out of 4 is O (d*n * $\sqrt{n}$).

4 Experiment Evaluation

Our experiments were implemented in the C++ language on a 1GHz Pentium PC machine with 1GB main memory, running on Debian Linux 4.0. The test data includes the aerial TIFF image (with Red, Green and Blue band reflectance values), moisture, and nitrate map of the Oakes Irrigation Test Area in North Dakota. The data is prepared in five sizes, that is, 128x128, 128x256, 256x256, 256x512, 512x512. The data sets are available at [4]. We evaluate our proposed EIN-ring base density Clustering algorithm (EDC) with respect to scalability, which is tested by increasing number of data records and number of attributes.

In this experiment, we compare our proposed EDC with Density Function based Clustering method using Euclidian distance (DFC). The experiment was performed on the five different sizes of data sets. The average CPU run time of 30 runs is shown in Fig.5.

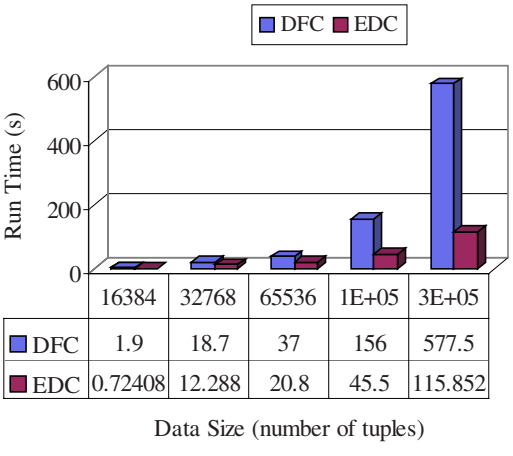


Fig. 5. Running Time Comparison of EDC with other Density Clustering

From Fig. 5, we see that EDC method is much faster than all of them on these five data sets. Especially when the data set size increases, the time of EDC method increases at a much lower rate than other methods. The experiment results show that EDC method is fast and scalable for large spatial data set.

5 Conclusion

In this paper, a unique approach to efficient neighborhood search using EIN-rings, and a new efficient density based clustering algorithm are developed. Our approach exploits compressed vertical data structures, Peano Trees (P-trees), and fast P-tree logical operations to accelerate the calculation of the density function within EIN-rings. This approach stands in contrast to the ubiquitous approach of vertically scanning horizontal data structures (records). The overall run time complexity is $O(dn\sqrt{n})$

for a d-dimensional data set, on the average. Experimental results show that the algorithm works efficiently on large-scale, high-dimensional spatial data, outperforming other density methods significantly.

Our method is particularly useful for data streams. In data streams, such as large sets of transactions, remotely sensed images, multimedia video, etc., new data keeps on arrival continually. Therefore both speed and accuracy are critical issues. Achieving high speed using P-tree, and high accuracy using the weighted EIN-rings provides a density based clustering method that is well suited to the clustering of steam data. Besides spatial data, our method also has potential applications in other areas, such as DNA micro array and medical image analysis.

Reference

1. Perrizo, W.: Peano Count Tree Technology. Technical Report NDSU-CSOR-TR-01-1 (2001)
2. Khan, M., Ding, Q., & Perrizo, W.: *K*-Nearest Neighbor Classification on Spatial Data Streams Using P-Trees. PAKDD 2002, Spriger-Verlag, LNAI 2336 (2002) 517-528
3. Hinneburg, A., & Keim, D. A.: An Efficient Approach to Clustering in Large Multimedia Databases with Noise. Proceeding 4th Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press (1998)
4. TIFF image data sets. Available at <http://midas-10cs.ndsu.nodak.edu/data/images/>.
5. Ester, M., Kriegel, H.P., Sander, J., & Xu, X.: Density-Connected Sets and their Application for Trend Detection in Spatial Databases. Proceeding 3rd Int. Conf. On Knowledge Discovery and Data Mining, AAAI Press (1997)
6. ESTER, M., KRIEGEL, H-P., SANDER, J. & XU, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd ACM SIGKDD, Portland, Oregon (1996) 226-231
7. SANDER, J., ESTER, M., KRIEGEL, H.-P., & XU, X.: Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. In Data Mining and Knowledge Discovery, 2 (1998) 169-194.
8. ANKERST, M., BREUNIG, M., KRIEGEL, H.-P., & SANDER, J.: OPTICS: Ordering points to identify clustering structure. In Proceedings of the ACM SIGMOD Conference, Philadelphia, PA (1999) 49-60
9. XU, X., ESTER, M., KRIEGEL, H.-P., & SANDER, J.: A distribution-based clustering algorithm for mining in large spatial databases. In Proceedings of the 14th ICDE, Orlando, FL (1998) 324-331
10. HAN, J. & KAMBER, M.: Data Mining. Morgan Kaufmann Publishers. San Francisco, CA (2001)
11. HAN, J., KAMBER, M., & TUNG, A. K. H.: Spatial clustering methods in data mining: A survey. In Miller, H. and Han, J. (Eds.) Geographic Data Mining and Knowledge Discovery, Taylor and Francis (2001)
12. Arya, S., Mount, D. M. & Narayan, O.: Accounting for boundary effects in nearest-neighbor searching. Discrete and Computational Gemetry (1996) 155-176