Calculation of Density-Based Clustering Parameters Supported with Distributed Processing

Marcin Gorawski and Rafal Malczok

Silesian University of Technology,
Institute of Computer Science,
Akademicka 16,
44-100 Gliwice, Poland
{Marcin.Gorawski, Rafal.Malczok}@polsl.pl

Abstract. In today's world of data-mining applications there is a strong need for processing spatial data. Spatial objects clustering is often a crucial operation in applications such as traffic-tracking systems or telemetry-oriented systems. Our current research is focused on providing an efficient caching structure for a telemetric data warehouse. We perform spatial objects clustering for every level of the structure. For this purpose we employ a density-based clustering algorithm. However efficient and scalable, the algorithm requires an user-defined parameter *Eps*. As we cannot get the *Eps* from user for every level of the structure we propose a heuristic approach for calculating the *Eps* parameter. Automatic *Eps* Calculation (AEC) algorithm analyzes pairs of points defining two quantities: distance between the points and density of the stripe between the points. In this paper we describe in detail the algorithm operation and interpretation of the results. The AEC algorithm was implemented in both centralized and distributed version. Included test results compare the two versions and verify the AEC algorithm correctness against various datasets.

1 Introduction

In today's world of computer science and computer applications there is a strong need for processing spatial data. There are on-line services providing very precise and high-quality maps created from satellite images [2]. Another example is traffic tracking in big cities, which results are later used to support decisions of building new bypasses, highways and introducing other rationalizations. More very interesting details can be found in [1].

One very important branch of spatial systems is telemetry. We are working on a telemetric system of integrated meter readings. The system consist of utility meters, collecting nodes and telemetric servers. The meters are located in blocks of flats, housing developments etc. They meter water, natural gas and energy usage and send the readings to the collecting nodes via radio. The collecting nodes

A Min Tjoa and J. Trujillo (Eds.): DaWaK 2006, LNCS 4081, pp. 417–426, 2006.

collect the readings and send them to the telemetric servers through the Ethernet network. Apart from meter readings, the data warehouse database stores information about meters' geographical location, and their attributes (e.g. meter type, installation date etc).

The most typical use for the described data warehouse is to investigate the consumption of the utilities. Our current research is focused on providing fast and accurate answers to spatial aggregate queries. We are in the process of designing and implementing a hierarchical caching structure dedicated for telemetry-specific data. We named the structure a Clustered Hybrid aR-Tree (CHR-Tree) because we intend to use clustering to create the structure nodes, and, like in the aR-Tree [5], the structure nodes store aggregates.

We already have a solution to a problem of storing and processing the aggregates in the CHR-Tree nodes [4]. Currently we are trying to construct the structure of the CHR-Tree. To create the intermediate level nodes we employ density-based clustering algorithm. We decided to use the DBRS algorithm [6]. Although efficient and scalable, the algorithm requires an user-defined parameter *Eps*. *Eps* is a parameter defining a half of the range query square side. The side length is used by the clustering algorithm to evaluate range queries when searching for neighboring points. As we cannot get the *Eps* parameter from the user for every level of the structure, we propose a heuristic approach for calculating the *Eps* parameter. We named the algorithm an Automatic *Eps* Calculation (AEC) algorithm. The algorithm is not limited to the telemetry-specific data and can be applied to any set of two-dimensional points. In the following sections we provide an extensive description of the AEC algorithm and its operation and implementation versions. We present also tests results proving that the AEC algorithm is applicable to sets of two-dimensional points of a wide variety and cardinality.

2 AEC Algorithm

As mentioned above, to apply the DBRS algorithm in our research we must be able to define the *Eps* parameter for proper clustering process. To the best of our knowledge, there is no automatic method for calculating or even estimating the *Eps* parameter for the density-based clustering. Authors of the DBScan algorithm proposed in [3] a simple heuristics to determine the *Eps* and $MinPts$ parameters. However, the heuristics cannot be considered automatic as it requires user interaction. In this section we present an Automatic *Eps* Calculation (AEC) algorithm which, basing on the points distribution characteristics, is able to calculate the *Eps* parameter value. The sets of points analyzed by the algorithm may be large, hence the amount of processed data must be limited. A random sampling approach alllows obtaining good results in acceptable time.

2.1 Algorithm Coefficients

A dataset containing all points is marked P. The AEC algorithm creates two sets of points. The first set N contains points randomly chosen from the set P.

A function creating the N set takes one optional parameter r, that defines the region from which the points are being picked. When the r parameter is present during the N set generation, we mark the set with an appropriate subscript: N_r. The second set H also contains points randomly picked from the P set. The function creating the set, next to the r optional parameter, whose meaning is identical as for the N set, takes another parameter defining the point that is skipped during random points drawing. The H sets are created for points from the N set. The notation H_{r,n_i} means that the H set was created for the point $n_i \in N$; the point n_i was skipped during random points drawing and the points in H are located in a region r.

The cardinalities of N and H sets are the AEC algorithm parameters. Thanks to the parameterization of those values we can easily control the algorithm precision and operation time. The cardinality of the N set is defined as the percent of the whole P set. The cardinality of the H set is defined directly by the number of points creating the set.

The AEC algorithm coefficients are calculated using the N and H sets. The algorithm picks random points from the set P creating the set N. In the next step, for each point $n_i \in N$ the algorithm creates set H_{n_i}.

Distance Between Points. The first AEC algorithm coefficient is a distance between two points. The distance analysis is based on calculating the Euclidean distances between the point n_i and all the points from the related H_{n_i} set. The distances are calculated for all points in the N set and all related H sets.

Points In Stripe and Stripe Density To evaluate the *Eps* parameter the AEC algorithm requires the knowledge about the neighborhood of the analyzed points; actually about the points in the region between the investigated points p_i and p_j. We decided to introduce a coefficient PIS (Points In Stripe). The value of $PIS(p_i, p_j)$ is the number of points located in a *stripe* connecting the points p_i and p_j. To evaluate the PIS coefficient value for a pair of points we use one spatial query and four straight lines equations. Having the p_i and p_j points coordinates we can easily calculate the parameters a and b of the straight line L equation $y = ax + b$. The line L contains the points p_i and p_j. In the next step we calculate equations of the lines perpendicular to L in points p_i and p_j, respectively L_{p_i} and L_{p_j} (we do not include the equations because of the complicated notation and straightforward calculations). The final step is to calculate two lines parallel to L, the first above line $L - L_a$ and the second below line $L - L_b$. The distance between the parallel lines and the L line (the difference in the b line equation coefficient) is defined as a fraction of the distance between points p_i and p_j. The fraction is the AEC algorithm parameter named $stripeWidth$; $stripeWidth \in (0, 1)$. The lines create a *stripe* between the points, and the *stripe* encompasses some number of points (fig. 1).

Having the lines equations we can easily calculate, whether an arbitrary point from the set P is located inside the stripe between points p_i and p_j or not. In order to reduce the number of points being analyzed we evaluate a rectangle encompassing the whole stripe. The rectangle vertexes coordinates are set by

calculating the coordinates of the points where the stripe-constructing lines (L_a, L_b, L_{p_i} and L_{p_j}) cross, and then choosing the extreme crossing points coordinates. Using the stripe-encompassing rectangle we execute the range query, to choose the points which can possibly be located within the stripe between p_i and p_j. In the next step, only the points chosen by the range query are examined whether they are located within the stripe.

Basing on the distance between points: $dist(p_i, p_j)$ and the number of points in a stripe between points $PIS(p_i, p_j)$ we calculate another coefficient which is a density of the stripe between p_i and p_j: $dens(p_i, p_j) = \frac{PIS(p_i,p_j)}{dist(p_i,p_j)^2 \cdot stripeWidth}$.

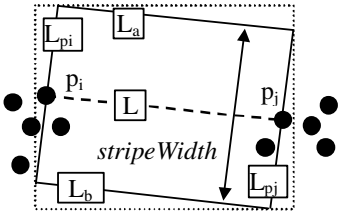

Fig. 1. Stripe between p_i and p_j points containing two points

Equipped with the distance and stripe density coefficients we are able to ascertain whether two points are relatively close to each other, and whether they are located in dense neighborhood. Our approach is not to search for a distance between points in clusters or for the thinnest cluster diameter, but rather for a minimal distance between clusters. The distance, or at least a value based on the distance, can be used as the *Eps* parameter in the density-based clustering algorithm. Using a minimal distance between clusters as the *Eps* parameter should result in grouping all the points whose distances to their closest neighbors are shorter than the minimal distance between clusters (they are in one cluster) and not grouping points when the distance between them is greater than the minimal distance between clusters.

2.2 Algorithm Operation

The AEC algorithm applies an iterative approach. In every iteration the algorithm tries to minimize the possible minimal distance between clusters. In the first step the algorithm sets the initial average distance between clusters $dist_{init}$ and related initial density $dens_{init}$. The values must be set in a way that they reduce the number of iterations to minimum, but on the other hand does not narrow down the set of possible solutions. After many experiments we decided to use an average distance between randomly selected points, and average density related to the distance. The calculation of the initial values uses the N and H sets.

The iterative section of the algorithm performs the following steps:

1. If the current iteration is the first iteration, assume that the current minimum distance between clusters $dist_{cur} = dist_{init}$, and the respective current density is $dens_{cur} = dens_{init}$.
2. Create a new N set, in a way described in subsection 2.1.
3. For every point $n_i \in N$ create a rectangle r_{n_i}, which vertexes coordinates are given by the following equation:
   $r_{n_i}(left, top, right, bottom) = r_{n_i}(n_i.x - dist_{cur}, n_i.y + dist_{cur}, n_i.x + dist_{cur}, n_i.y - dist_{cur})$.
4. For every point $n_i \in N$ create a set $H_{r_{n_i}, n_i}$ skipping the point n_i.
5. Evaluate an average density of the r_{n_i} rectangle.
6. For every point n_i, and points from the related H_{n_i} set calculate a set of quantities: distance, PIS and density of the stripe between points.
7. From all the results choose the shortest distance, for which the $PIS > 0$ and the density is less than the average density of the r_{n_i}. If there is no such result, do not return anything.
8. Compare the result obtained for the point n_i with the current values of $dist_{cur}$ and $dens_{cur}$. If $dist_i < dist_{cur}$ and $dens_i <= dens_{cur}$ then update the current values of minimal distance and minimal density between clusters: $dist_{cur} := dist_i$ and $dens_{cur} := dens_i$. If only the first part of the condition holds ($dist_i < dist_{cur}$), then check a *suspected region* defined by using the coordinates of points for which the $dist_i$ was calculated. Details of this operation are described below. The operation of checking a *suspected region* can possibly return a pair of results: the distance $dist_s$ and related density $dens_s$. The returned pair is compared with the iteration results and if $dist_s < dist_{cur}$ and $dens_s <= dens_{cur}$ then the results of the iteration are updated: $dist_{cur} := dist_s$ and $dens_{cur} := dens_s$.
9. Check the iteration breaking condition. The iterations can be broken in two cases: (1) the number of performed iterations is greater than the declared number of iterations (which is another AEC algorithm parameter), and (2) if the result returned by consecutive iterations was repeated a fixed number of times. Breaking the iteration caused by the second condition is more desirable, because we can expect that the algorithm found a minimal distance between clusters that cannot be replaced by any other distance.

Suspected Regions Analysis. The case of a *suspected region* is considered for points p_i, p_j when only the distance condition ($dist(p_i, p_j) < dist_{cur}$)) holds, the density condition ($dens(p_i, p_j) <= dens_{cur}$)) does not. Our experiments show that there are two possible scenarios resulting in examining the *suspected region*:

1. the points p_i, p_j are located close to each other inside a cluster. Then the distance is short, but the density of the stripe between the points is high.
2. the points p_i, p_j are located in separate clusters but they are not border points (according to the definition presented in [6]). The density of the stripe between the points is increased by the presence of the border points of both clusters.

Of considerable interest is the second case. The AEC algorithm does not analyze distances with the zero PIS coefficient. There are many cases when clusters' shapes make it difficult to randomly pick two points so that one of them is a border point of the first cluster and the second is located near the border of the second cluster. The analysis of *suspected region* is performed as follows:

1. define the *suspected region*. The rectangle r_s for the *suspected region* has its center directly between the points p_i and p_j. In the next step calculate the density $dens_{r_s}$ of the r_s.
2. create a set of points N_{r_s}.
3. for each point $n_i \in N_{r_s}$ create a set H_{n_i,r_s}, then calculate distances and densities of the stripe between points n_i and the related points $h_j \in H_{n_i,r_s}$. As the result choose the minimal distance with the minimal density.

In the event the calculated result density is less than the average density of the r_s region, the *suspected region* analysis results are compared with the results of the analysis in the iterative section of the AEC algorithm. For a pair of points located inside a cluster the *suspected region* analysis does not influence the results because the density condition is not satisfied (the density is high inside a cluster). But for the points located in two different clusters the analysis often gives important results.

The amount of points checked during *suspected regions* analysis depends on the number of points in the r_s rectangle. If the number is less than the N set cardinality, then all the points are checked. But if the number is greater, the cardinality of the N_{r_s} set equals the cardinality of the N set created in the iterative section of the algorithm. The situation is identical for the H sets.

Clustering with AEC-calculated *Eps* Parameter. Application of the calculated *Eps* parameter to density-based clustering results in creating clusters which number and cardinalities depend on the points distribution characteristics. If the density of all clusters is similar (the distances between neighboring points in all clusters are always less than the distances between border points of the closest clusters), then the result of the AEC algorithm is the distance between a pair of closest clusters. Having the estimated distance between the closest clusters we can define the *Eps* parameter for the density-based clustering as 85% – 90% of the obtained distance. Decreasing the value of the distance we prevent merging of the closest clusters during clustering process.

If the points are grouped in clusters of significantly different density then the AEC algorithm outcome depends on the density of the sparse clusters. If the distance between dense clusters is lower than the distance between neighboring points in the most sparse cluster, then the AEC algorithm outcome is the distance between the dense clusters. Performing the clustering results in creating the dense clusters and, in sparse clusters, merging points located close to each other. But if distances between neighboring points inside all kinds of clusters (both dense and sparse) are less than the minimal distance between border points of two closest clusters, then the AEC algorithm outcome is the minimal distance between clusters. Performing the clustering results in proper creating both dense and sparse clusters.

2.3 Implementation

The process of calculating the *Eps* parameter by means of the AEC algorithm is time-intensive. In order to improve the efficiency we used distributed processing. The architecture of the distributed implementation is based on the client-server standard. Every server stores the set of all points P and every server performs the same operations but for different subsets of points. Each server is assigned a set of points from which it creates the N sets. The sets are disjoint for all servers. Thus we minimize the possibility that some servers examine the same pair of points. The H sets are created from the whole P set, without limitations. We implemented two different distributed versions of the AEC algorithm.

1. The first version named *at once* (AO) assumes, that the client and servers do not communicate during the process of *Eps* evaluation. The servers calculate the minimum distance between clusters with the lowest density and return the results to the client which selects the best result (the shortest distance with the lowest related density). Disadvantage of this approach is that the servers calculations are less precise because they use N sets which cardinalities are only $\frac{1}{K}$ cardinality of the sets used in the centralized version (where K is the number of servers).
2. The second version named iterative (IT) assumes that the client requests the servers to perform the i iteration of the whole process. The servers return results of the i iteration to the client. The client selects the best result from all the answers. In the next step, the client transfers the chosen result to all the servers. The servers use the result as the initial distance and initial density for the next $i + 1$ iteration. The number of performed iterations and the number of repeated consecutive results are controlled by the client. Operation of the servers is *synchronized* by setting the initial distance and initial density. In this approach client and servers communicate more often, but the obtained results are more precise.

3 Test Results

In this section we present tests results obtained for eight various sets of points. The sets were marked from A to H; they vary with cardinality, points distribution and clusters shapes (fig. 2). The A set contains about 650 points grouped in 10 dense clusters; density of all clusters is very similar. The next set, B, contains about 200 points grouped in three relatively sparse clusters; density of all clusters is similar. The C set contains only about 120 points grouped in eight small clusters. In the D set 400 points are grouped in three dense clusters, one less dense, and one sparse cluster. The E and F sets contain over 400 points. The G and H sets contain respectively 1000 and 1500 points. In all four sets, clusters have similar density but significantly differ in shapes. Small clusters located inside the big ones were intended to disrupt the AEC algorithm when calculating

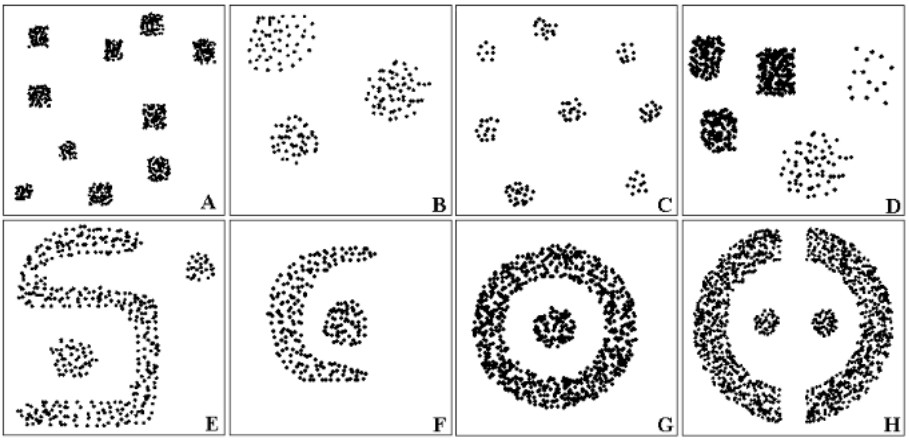

Fig. 2. Sets of points used for testing

the PIS coefficient. For each dataset we performed a set of experiments with the following parameters:

- the cardinality of the N set was 5, 15, 25 and 35% of the input dataset cardinality,
- the cardinality of the H set was 5, 15, 25 and 35 points for each value of the N set cardinality,
- the number of iterations was set to 10, 20 and 30 for each combination of N and H sets cardinality.

As can be easily calculated, a single test set contained $4 \times 4 \times 3 = 48$ tests. In our tests the iterations were broken if the result of the consecutive iterations was repeated more than 5 times. The iteration breaking was always caused by the number of repeated consecutive results. Thus we can treat the tests for identical cardinality of N and H sets as three repeated tests, which is useful in the presence of the random factor. We performed the tests for a centralized version of the algorithm, and for two distributed versions AO and IT.

The AEC algorithm is written in Java. All the experiments were run on machines equipped with Pentium IV 2.8 GHz and 512 MB RAM. The software environment was Windows XP Professional, Java Sun 1.5 and Oracle 10g. The distributed environment consisted of four machines connected with Ethernet 100Mbit network. The communication was based on Java RMI.

The main purpose of the experiments was to verify the AEC algorithm correctness and efficiency against various datasets. The AEC algorithm was run with a given set of parameters. The calculated *Eps* parameter was passed to the DBRS algorithm, which was returning the number of created clusters. If the number of clusters declared for a given dataset equaled the number of clusters found by the DBRS, we marked the experiment as success. If the number of clusters was not equal, we marked the experiment as failure.

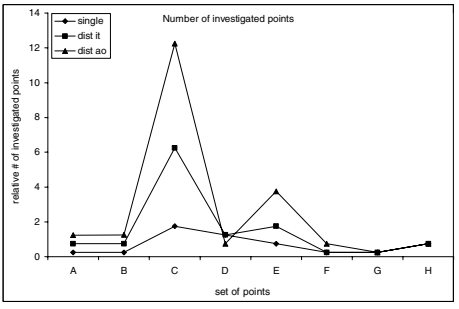

Fig. 3. Relative number of points checked for correct *Eps* calculation

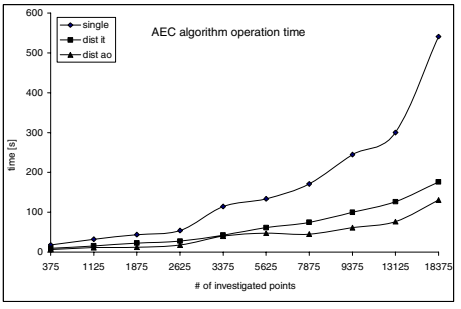

Fig. 4. AEC algorithm operation times as function of investigated points number

The graph in figure 3 illustrates the relative number of investigated points for various set of points. The number of investigated points calculated as $|N| \cdot |H|$ was related to the cardinality of the set P, hence we can compare the results for sets of different cardinality on a single plot. In figure 4 we present a graph comparing AEC algorithm operation times for the three implementation versions. The x axis shows the number of investigated points. The y axis shows AEC algorithm operation times in seconds. We considered only the cases when the algorithm gave the correct results. As expected, the centralized version execution consumes much more time when compared to the distributed versions. For small cardinalities of investigated points sets (less than 3000) the differences in operation times are not significant. But for greater cardinalities the distributed versions operate much more efficiently. For cardinalities exceeding 10000 points we observe nearly linear speed-up.

Summarizing the tests results we notice that for all tested sets of points the AEC algorithm gives proper results. There are more and less *difficult* sets of points but the algorithm is able to correctly analyze all of them. The most difficult to analyze are sets of points with a big number of small clusters. The algorithm operation is not disturbed by the differences in densities and/or shapes of the clusters. Also the presence of small clusters inside big ones does not negatively affect the algorithm operation. The accuracy of the AEC algorithm is determined by the algorithm parameters. The bigger the N and H sets cardinalities (the more pairs of points the algorithm investigates) and the more iterations performed, the more accurate the results. However, every investigated pair of points has its influence on the algorithm operation time. The parameters should be set according to the tested dataset. If the dataset characteristics are not known in advance (as with the presented test scenario) the obtained results show that investigating 25% of a dataset and setting the maximal number of iterations to 10 always gives accurate results.

The centralized version of the AEC algorithm gives the most accurate results. For all tested sets of points the centralized version always required the smallest

N and H sets. This version also needed the smallest number of iterations for obtaining the correct results. The AO distributed version operates most efficiently (is able to examine the biggest number of pairs of points in the shortest time), but on the other hand, the AO version always requires the biggest N and H sets, and the biggest number of iterations. Therefore, the best choice is the iterative distributed version (IT). It is faster than the centralized version and gives better results than the distributed AO version.

4 Conclusions and Future Work

This paper presents an empirical approach to a problem of automatic calculation of *Eps* parameter applicable in density-based clustering algorithms such as DBRS and DBScan. In our approach the AEC algorithm, working iteratively, chooses randomly a fixed number of sets of points and calculates three coefficients: distance between the points, number of points located in a stripe between the points and density of the stripe. Then the algorithm chooses the best possible result, which is the minimal distance between clusters. The calculated result has an influence on the sets of points created in the next iteration.

We implemented the AEC algorithm in one centralized and two distributed versions. We presented test results for a collection of eight different sets of points. With appropriately high number of examined points the algorithm was able to calculate the proper *Eps* parameter for all tested sets of points. Our future work includes further improving the AEC algorithm efficiency. We want to eliminate the most time-intensive part of the algorithm which is calculating the value of the PIS coefficient. We are currently searching for conditions allowing us to skip the PIS coefficient calculation.

References

1. Barclay T., Slutz D.R., Gray J.: TerraServer: A Spatial Data Warehouse, Proc. ACM SIGMOD 2000, pp: 307-318, June 2000
2. http://maps.google.com
3. Ester M., Kriegel H.-P., Sander J., Wimmer M.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In proc. of 2^{nd} International Conference on Knowledge Discovery and Data Mining, 1996
4. Gorawski, M., Malczok, R.: On Efficient Storing and Processing of Long Aggregate Lists. DaWaK, Copenhagen, Denmark 2005.
5. Papadias D., Kalnis P., Zhang J., Tao Y.: Effcient OLAP Operations in Spatial Data Warehouses. Spinger Verlag, LNCS 2001
6. Wang X., Hamilton H.J.: DBRS: A Density-Based Spatial Clustering Method with Random Sampling. In proceedings of the 7^{th} PAKDD, Seoul, Korea, 2003