Fusion Based Methodology for Spatial Clustering

Pavani Kuntala and Vijay V. Raghavan
The Center for Advanced Computer Studies
University of Louisiana at Lafayette, LA 70504-4330
pavani@louisiana.edu, raghavan@cacs.louisiana.edu

Abstract

In this paper, a novel clustering algorithm is proposed to address the clustering problem within both spatial and non-spatial domains by employing a fusion-based approach. The motivation for this work is to overcome the limitations of the existing spatial clustering methods. In most conventional spatial clustering algorithms, the similarity measurement mainly takes the geometric attributes into consideration. However, in many real applications, there is a need to fuse the information from both the spatial and the non-spatial attributes. The goal of our approach is to create and optimize clusters, such that the data objects satisfy both spatial and non-spatial similarity constraints. The proposed algorithm first captures the spatial cores having the highest structure and then employs an iterative, heuristic mechanism to determine the optimal number of spatial cores and non-spatial clusters that exist in the data. Such a fusion-based framework allows for comparing clusters in spatial and non-spatial contexts. The correctness and efficiency of the proposed clustering algorithm is demonstrated on real world data sets.

1. Introduction and Motivation for Spatial Clustering

Clustering is one of the prominent data mining tasks, which has been studied in detail[14]. Spatial data mining or knowledge discovery in spatial databases refers to the extraction, from spatial databases, of implicit knowledge, spatial relations, or other patterns that are not explicitly stored [6]. The large size and high dimensionality of spatial data make the complex patterns that lurk in the data hard to find. It is expected that the coming years will witness very large number of objects that are location-enabled to varying degrees. Spatial clustering has been used as an important process in the areas such as geographic analysis, exploring data from sensor networks, traffic control, and environmental studies. Spatial data clustering has been identified as an important technique for many applications and several techniques have been proposed over the past decade based on density-based strategies, random walks, grid based strategies, and brute-force exhaustive searching methods[8]. This paper deals with spatial clustering using a fusion-based approach.

Spatial data is about instances located in a physical space. Spatial clustering aims to group similar objects into the same group considering spatial attributes of the object. The existing spatial clustering algorithms in literature focus exclusively either on the spatial distances or minimizing the distance of object attributes pairs. i.e., the locations are considered as another attribute or the non-spatial attribute distances are ignored. Much activity in spatial clustering focuses on clustering objects based on the location nearness to each other[12]. Finding clusters in spatial data is an active research area, and the current non-spatial clustering algorithms are applied to spatial domain, with recent application and results reported on the effectiveness and scalability of algorithms [9, 12]. One of the earliest and most prominent clustering algorithms applied for spatial data mining are partition based approaches. Partitioning algorithms are best suited to such problems where minimization of a distance function is required and a common measure used in such algorithms is the Euclidian distance. Recently new set of spatial clustering algorithms has been proposed, which represents faster method to find clusters with overlapping densities. DBSCAN, GDBDCAN and DBRS are density-based spatial clustering algorithms, but they each perform best only on particular types of datasets[12]. However, these algorithms also ignore the non-spatial attribute participation and require user defined parameters. For large-scale spatial databases, the current density based cluster algorithms can be found to be expensive as they require large volume of

----
Proceedings of the 3rd Workshop on STDBM
Seoul, Korea, September 11, 2006

memory support due to its operations over the entire database. Another disadvantage is the input parameters required by these algorithms are based on experimental evaluations. There is a large interest in addressing the automation of the general purpose clustering approach without user intervention[13]. However, it is difficult to adapt these algorithms to spatial data.

Spatial dimensions (e.g., latitude and longitude) cannot simply be treated as two additional non-spatial dimensions because of several important reasons. Spatial data are generally multi-dimensional, requires the adoption of real-world dissimilarity measures, and are autocorrelated. Spatial data includes topological relationships. These distinctions put spatial and non-spatial data into different categories with far-reaching implications for conceptual, processing, and storage issues. General-purpose clustering methods mainly deal with non-spatial feature spaces and have very limited power in recognizing spatial patterns that involve identification of dense spatial neighborhoods.

In section 2 we present the proposed fusion based spatial clustering. The results of the application of proposed clustering on real-world problems are examined in section 3. The conclusions are presented in section 4.

2. Proposed Fusion-Based Spatial Clustering

Data fusion is a process in which the available data is combined to find representations of higher quality. The U.S. Department of Defense [2] stated that “data fusion is a multilevel, multifaceted process dealing with the automatic detection, association, correlation, estimation, and combination of data and information from multiple sources.” More recent concepts of data fusion are discussed in [11]. Data Fusion is used in different context in different fields. For example in fields of sensors and image analysis it is viewed as target clustering problem [4]. We illustrate in our proposed approach, that effective data fusion can be achieved through clustering and vice versa.

The method proposed for the fusion is a two step procedure comprising the following steps: (i) an initial phase for locating the initial spatial cores based on the spatial densities and (ii) an optimization phase for finding the optimal spatial and non-spatial fusion. In the initialization phase, we find the initial data tessellation, called spatial cores, based on spatial densities. The second phase is the fusion of spatial and non-spatial constraints. The goal of the optimization phase is to iteratively provide an intuitive grouping of the data objects, such that these groups satisfy both the spatial and non-spatial constraints. In every iteration of the optimization phase, we check if the algorithm is improved. The algorithm is considered improved, when the spatial distance is minimized and the non-spatial dissimilarity is also minimal in each spatial partition. Next, we describe the details of the initialization and optimization phases of our proposed algorithm.

2.1 Initialization

We apply a density-based approach to get reasonable neighborhood for the spatial core initialization process. These approaches hold that, for given radius each object within a cluster, the neighborhood of a give cluster must exceed a specified threshold. Density clustering methods depend on the proper selection of the neighborhood size and the density size. Without prior knowledge of the structure of the input data set, proper parameter selection is cumbersome.

The initialization step of our proposed algorithm differs from the existing approaches mainly in the following two aspects.

Instead of choosing a fixed radius, we employ a heuristic approach to find a sequence of varying neighborhoods to catch the cluster structures of the input data.

The clusters are grown in the direction of maximum movement of objects. Here we assume the object is attracted to the strongest neighbor, which is the neighbor with highest densities.

We assume the input data set X is $X = \{\vec{X}_1, \vec{X}_2, ..., \vec{X}_i, ..., \vec{X}_n\}$, where $\vec{X_i}$ is an object represented as a vector of non-spatial features, and n is the total number of objects. $\vec{X}_i = (X_{i1}, X_{i2}, .., X_{ij,..}, .., X_{id}) \in \Re^d$, where d is the total number of non-spatial features, and X_{ij} represents the value of attribute j of object $\vec{X_i}$.

Let $L = \{L_1, L_2, ... L_\lambda\}$ be the set of spatial locations in the data space. The set of objects at location s, L_s *are* $\{L_p = \{\overrightarrow{l(X_{i'})}, ..., \overrightarrow{l(X)_{i''}}\} \mid 1 \le i', i'' \le n, |X_s| \le |X|\}$. If we assume each location as a representation of singleton set of objects, i.e., one object per location, the object information at any location can be represented as $\overrightarrow{l(X_i)}$. That is, the cardinality of L is the same as the cardinality of X, *i.e.,* $\lambda = n$. We scan the input data set X once to find the spatial densities of each object, within a neighbourhood of radius φ. Let z be the number of spatial cores, and set it to a small value initially. Initially we find the area of the minimum bounding rectangle of the objects, and tessellate the data into z cores. For each object $\vec{X_i}$ we find the density ξ_i of that object as the number of the other objects that are within a geometric neighbourhood of radius φ.

Initially, we assume that the objects are uniformly distributed among all initial spatial cores, and the spatial clusters have a uniform neighborhood area. This initial assumption makes the algorithm independent of the input order of the data, and provides a better approach, than partitioning the data using random seeds. Note that in later stages of our algorithm, the number of cores varies based on the optimization stage. At this stage, since we are finding dense partitions based only on spatial dimensions, a Minimum Bounding Rectangle, MBR provides a good approximation for the total area of the objects.

Minimum Bounding Rectangles: The initial tessellation of data is based on Minimum Bounding Rectangles. MBRs have been used extensively to approximate objects in spatial data structures and spatial reasoning, because they need only two objects for their representation; in particular, for a set of objects X', where $X' \subseteq X$, X' is represented as an ordered pair $\left(l_1(X'), l_2(X')\right)$. Such approximations naturally extended to finding topological relations[7]. $l_1(X'), l_2(X')$ correspond to the lower left, and the upper right object of the MBR that covers all objects X'. $l_1(X')$ and $l_2(X')$ are called the edge objects.

Definition 1. *Given a set of objects X', let $l_\alpha(X') = \left(l_{1\alpha}(X'), l_{2\alpha}(X')\right)$ and $l_\beta(X') = \left(l_{1\beta}(X'), l_{2\beta}(X')\right)$ be the intervals of X', created by projecting all the objects on spatial dimensions α and β, respectively. The MBR of the set of objects X' is defined as the domain $\left(l_1(X'), l_2(X'), X'\right)$, where $l_1(X') = \left(l_{1\alpha}(X'), l_{1\beta}(X')\right)$, and $l_2(X') = \left(l_{2\alpha}(X'), l_{2\beta}(X')\right)$, and $l_{1\alpha}(X') <= \alpha <= l_{2\alpha}(X') \Lambda\ l_{1\beta}(X') <= \beta <= l_{2\beta}(X')$.*

Initial Tessellation of Data Space: We define dense region as a set of data objects, clustered using only spatial information. The density cluster as defined by Ester et. al in[3], requires both surrounding radius of an object φ, within which at least ξ neighboring objects should be found. The main differences between our proposed spatial core approach and the current density clustering approaches are: 1) Our proposed algorithm heuristically calculates the radius φ, and, 2) the dense regions are evolved only in the direction in which the objects naturally grow.

We propose a heuristic approach to compute φ. Assume that objects are uniformly distributed in the clusters are equally separated in the MBR. Then assuming the clusters are circular, $\varphi = f(MBR,z) = \sqrt{(1/\pi z)A(X')}$ where $A(X')$ is the area of the MBR $\left(l_1(X'), l_2(X'), X'\right)$. By employing such a heuristic approach, we can find clusters where the data points are not densely packed and hence might have a low ξ.

Since the number of spatial clusters evolve as the algorithm evolves, and the shapes of the clusters change, these assumptions are good to estimate the initial cores. The densities of all the objects within radius φ are calculated and sorted. Let $\overrightarrow{X_i}$ be the object with the maximum density. We assign all the objects within radius of φ to one spatial cluster. For the other $z-1$ clusters we grow the cluster in the direction of its movement. This allows for outlier detection and identifying a dense region that is surrounded by another uniformly distributed sparse region.

Any objects belonging to X', and not part of initial dense regions are considered as unclassified objects. In the optimization stage detailed in next section, the objects in the above initial cores and unclassified objects are optimized to arrive at the clusters satisfying both the spatial and non-spatial constraints.

2.2 Optimization

The optimization stage aims at dynamically combining the best features of both spatial and non-spatial distances in the manner of alternating, fusing spatial and non-spatial clusters, and arriving at heuristically computed number of spatial cores and non-spatial clusters. We introduce some more notations and definitions to explain the concepts of cores and $\wp$-clusters. In previous section we mentioned that each data object $\overrightarrow{X_i}$ is a vector of d non-spatial features. We define $\wp$-clusters are clusters satisfying non-spatial optimization criterion, and core as a representation of a set of objects which satisfy the spatial constraints. i.e., all objects in a core are geometrically proximate.

Definition 2. *A $\wp$- cluster C_f is a set of objects which are homogenous only in non-spatial dimensions and are well-separated from the other $\wp$-clusters only in non-spatial dimensions.*

The non-spatial distances are dependent on the application domain and types of features. Assuming $d_{ns}\left(X_{ij}, X_{i'j}\right)$ is the non-spatial distance function, and Opt_{ns} is the non-spatial optimization function, the objective function for $\wp$- clusters is to minimize

$$Opt_{ns}(X,K) = \sum_{f=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{d} d_{ns}\left(X_{ij}, Z_{fj}\right),$$

where X represents the object matrix, K is the cluster membership matrix, and k is the number of $\wp$-clusters.

The following statements hold when we refer to two kinds of centers, *gravity* and *centroid* as defined in definition 4. The geometrical centers of the spatial cores or $\wp$-clusters will be referred to as the gravity. The centers based on non-spatial features, of spatial cores or $\wp$-clusters, will be referred to as centroid. Z_f is the centroid of the $\wp$-cluster f, and Z_{fj} is the centroid value of non-spatial feature j for $\wp$-cluster f. The centroid Z_f is the average of the non-spatial characteristics.

Definition 3. *A core C^r is a combination of $\wp$- clusters. Two distinct cores can have the $\wp$- clusters that are non-spatially similar. Each $\wp$- cluster of a core indicates the objects in a core that are non-spatially homogenous and well-separated from the other $\wp$- clusters in the core. C_f^r represents a $\wp$- cluster in core C^r.*

The following criterion holds for cores and $\wp$- clusters.

1. $C_f{}^r \neq \phi, r = 1 \ldots z; f = 1 \ldots k$
2. $\left| \bigcup_{r=1}^{z} \bigcup_{f=1}^{k} C_f^r \right| <= z \times k$
3. $C^r \bigcap C^{r'} = \phi, 1 \leq r, r' \leq z, r \neq r'$
4. $C_f{}^r \bigcap C_{f'}{}^r = \phi, 1 \leq f, f' \leq k, f \neq f'$
5. $\left| \bigcup_{r=1}^{z} C_f^r \right| \leq k$

Let object $\overrightarrow{X_i}$ belonging to core C^r be represented as X_i^r, and object $\overrightarrow{X_i}$ belonging to $\wp$-cluster C_f^r be represented as X_i^{rf}.

Definition 4. *The gravity Z^r for core C^r (or Z_f^r for $\wp$-cluster C_f^r) , is defined as the geometrical center for the set of objects in the core (or $\wp$-clusters) . It is computed as the mean of the locations of all objects in the core.*

$$Z^r = \frac{\sum_{i=1}^{p} l(X_i^r)}{p_r} ,$$

where p_r is the number of objects in C^r.

The objective function for core C^r is to minimize the spatial distances among the $\wp$-clusters belonging to that core. Depending on spatial dimensions, any spatial distance metric, for example rectilinear distance function[1] for two dimensional spatial features can be used to find the spatial proximity. Assuming $d_{sp}\left(X_i, X_{i''}\right)$ is the spatial distance function, and Opt_{sp} is the spatial optimization function, the objective function for core C^r is to minimize

$$Opt_{sp}(C_f^r) = \sum_{f=1}^{q_r} \sum_{i=1}^{p_f} d_{ns}\left(X_i^{rf}, Z_f^r\right) ,$$

where q_r is the number of $\wp$-clusters in a core C^r and p_f is the number of objects in *$\wp$-cluster* C_f^r.

The intuition of the fusion approach is to use the information derived from spatial proximity distance to merge-split the non-spatial clusters and vice-versa. We define the distance function for the fusion based approach as

$$d_{fusion} = (\sum_{1}^{z} d_{ns}(Z^{r_t}, Z^{r_{t'}}) + \sum_{1}^{k} d_{sp}(Z_{f_a}, Z_{f_{a'}}))^{1/2},$$

where $t, t' \in \{1..z; t \neq t'\}$ and $a, a' \in \{1..z; a \neq a'\}$. After, the initial clusters have been created, the optimization phase of the algorithm iteratively tries to improve the object distribution among the cores and the non-spatial distribution of the $\wp$-clusters. i.e., $Opt_{fusion}(X) = \min(d_{fusion})$.

The pseudo code of clustering based on co-learning and fusion of spatial and non-spatial algorithm is depicted as shown in Fig. 1. In the beginning of the optimization phase, we consider each dense region formed in the initialization phase as a core. A representative non-spatial $\wp$-cluster is found for each core. All the objects belonging to a representative cluster should satisfy the following criteria.

$$rep(C_f^r) = \{X_i : i = 1 \ldots p_r \mid X_i \in C^r \wedge X_i \in C_f\} ,$$

where

$$X_i \in C^r \quad iff \quad d_{sp}\left(X_i, Z^r\right) < d_{sp}\left(X_i, Z^{r'}\right); 1 \leq r, r' \leq z; r \neq r'$$

$$X_i \in C_f \quad iff \quad d_{ns}\left(X_i, Z^{r,f}\right) < d_{ns}\left(X_i, Z^{r'f'}\right); 1 \leq f, f' \leq k; f \neq f'$$

i.e., find the objects that have the non-spatial characteristics similar to the centroid of the current core. In the iterative phase of the optimization algorithm, new cores and $\wp$-clusters are formed based on defined merge-split rules. Each object $\overrightarrow{X_i}$ is categorized as follows: *C-Similar, $\wp$-similar*, and *joint*. *C-Similar* implies that the object is spatially proximate to the same core but not to any of the existing $\wp$-clusters, i.e., it is non-spatially similar to the object closest to the gravity of the unclassified objects. *$\wp$-similar* implies that the object is similar to a $\wp$-cluster but not geometrically proximate to existing cores, i.e., it is geometrically proximate to the object closest to the centroid of unclassified objects. An object is considered *joint* if and only if it is similar to an existing $\wp$-cluster of a core. All the unclassified objects after each iteration, which satisfy the *joint* criteria, are merged with the existing $\wp$-cluster of a core. All the objects of a core which are *C-similar,* are considered as tentative $\wp$-clusters of the core that they are geometrically proximate. Similarly all objects, which are *$\wp$-similar*, form new tentative cores. Based on the MBRs, if the tentative cores are non-overlapping they are considered singleton cores, and are merged with a core whose gravity is closer than the gravity of the unclassified objects. If not singleton, and the MBRs are overlapping, a new core is formed. After each iteration of the optimization phase, we compute the new fusion distance, and if the rate of decrease d_{fusion} is greater than the percentage of unclassified objects, we terminate the algorithm.

The algorithm is guaranteed to converge, because the d_{fusion} ensures that the points are geometrically proximate to only one core, and non-spatially proximate to only one ℘-cluster of a core, at any given time. An added benefit of the above fusion based clustering process is that the cores can be explored to retrieve region specific ℘-clusters. Further we can compare two cores, by comparing the ℘-clusters of any two given cores.

```
Algorithm Optimize
Input :  Initial cores of objects
Output: Spatial Cores and ℘-clusters in each core
Step 1:
  Initial z dense regions;
  Find the gravities and centroids of each core;
Step 2:
  $d_{f\_prev} = d_{fusion}$;
  Find the rep ℘-clusters, $rep(C_f^r)$ , of each core;
  Compute C-similar, ℘-similar, and joint;
Step 3:
  For all unclassified objects {
    Add the joint objects to existing ℘-cluster of a core;
    In each core, create ℘-clusters with all objects c- similar;
    Create tentative cores with all objects ℘-similar;
}
Step 4:
  if non-overlapping singleton cores {
    merge with an existing core;}
 compute $d_{fusion}$;
Step 5:
  Find representative ℘- clusters $rep(C_f^r)$
  Objects that do not belong to core or ℘-cluster
      considered unclassified;
  if [($d_{f\_prev}$ -$d_{fusion}$)/ $d_{f\_prev}$ -$d_{fusion}$]> [ unclassified / n)]
   go to step 2
  else exit;
}
```

Figure 1. Algorithm Optimize

3. Experimental Evaluation and Results

Comprehensive experiments were conducted to assess the accuracy and efficiency of the proposed approach. By comparing to existing algorithms, such as purely spatial clustering (GraviClust[5]), Non-Spatial(k-means), and spatial clustering algorithm with partial capability to support non-spatial attributes(GDBSCAN), we demonstrate accuracy of the proposed approach. Our choice of comparative algorithms is based on their effectiveness and popularity in literature. Since our algorithm emphasizes on the need for looking at both the spatial distances and attribute values, we evaluate our algorithm in the three evaluation setups, as shown in section 3.1.

We evaluated both the effectiveness and efficiency of our approach using two real world datasets. In the spatial clustering literature, there are no fixed benchmarking datasets for evaluation of the algorithms. Our choice of datasets is based on the number of non-spatial dimensions, and the distribution of classes in both spatial and non-spatial dimensions. To demonstrate the functionality of the fusion based approach, we will details the experiments conducted using a data set with two spatial dimensions. Next, we show our experimental results on high-dimensional data sets to show the scalability of our approach.

The first data set is US census tract housing data. The observations contain 4 non-spatial attributes land area, population, median per capita income, median year built, as well as the latitude and longitude of the centroid of the tract from the 1990 Census. This resulted in 57,647 observations with complete data. The associated class information is the log of the median price of housing. After looking at the Gaussian distribution of the class values, we binned the log of median price of housing into 7 classes. The class distribution for this dataset is equally influenced by the spatial and non-spatial features. The second data set is a multivariate GIS data set. The actual forest cover type for a given observation (30 x 30 meter cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Independent variables were derived from data originally obtained from US Geological Survey (USGS) and USFS data. The total number of instances (observations) are around 500,000. The attributes are distributed as 54 columns of data. The data represents 7 types of forest cover types. We found for this data set the class distribution is spatially biased.

3.1 Evaluation Metrics

Since the final results of an algorithm are clusters based on the fusion of different distance metrics, our evaluation metric should be independent of any particular distance metric. We identified three evaluation criterion to measure the quality of our clusters.

1. Cluster Accuracy: r-value (purity). The r-value as defined in [10] is

$$r = (1/n)\sum_{i=1}^{c} a_i \quad ,$$

where, c stands for the number of classes , a_i denotes the number of objects of the cluster i with the largest representation in class *i*, and *n* is the total number of objects in the database. This criterion measures the degree of correspondence between the resulting clusters and the classes assigned a priori to each object.

2. Total number of cluster assignment errors: Number of objects that co-occur in a class versus that appear in different clusters.

3. Number of clusters found: Total number of cores or clusters discovered versus the actual number of classes in the data.

3.2 Effectiveness

In this section, we demonstrate the effectiveness of our algorithm, in comparison to the KM, GDBSCAN and GraviClust, and for varying data set sizes. We chose the census dataset to demonstrate the effectiveness our algorithm, since the natural grouping of the data is equally influenced by the spatial and non-spatial features.
We varied the dataset sizes by gradually increasing the size of a region. We incrementally increased the longitude by one degree, thus obtaining varying data set sizes (and number of classes). This provides a natural spatial partition of the data, without losing the inherent spatial and non-spatial information. Instead if we have chosen random samples of the objects in census data, the spatial information would become meaningless. Table 1 shows the data set sizes and the number of classes of each sample.

Table 1. Sample Data Set Sizes and Number Of Classes

Degrees of Longitude	Number of Objects	Number of Classes
4	694	5
4.2	1191	5
5	1931	5
6	2750	6
7	5984	6
8	7953	6
10	11841	7
13	15615	7

Table 2 illustrates the effectiveness of our algorithm in comparison with traditional k-means(KM) and the GDBSCAN algorithm. Higher r-value indicates better accuracy. Since the competitor algorithms are input order and parameter dependent, we heuristically chose the settings that resulted in maximum accuracy for the competitors.

Table 2. Accuracy of the clusters based on r-value

DataSet Size	KM	GDBSCAN	Gravi-Clust	Proposed Fusion based Clustering (Cores)	Proposed Fusion based Clustering ($\wp$-Clusters)
694	0.49424	0.20749	0.56052	0.69308	0.67147
1191	0.39463	0.59446	0.69353	0.82284	0.70109
1931	0.37856	0.56499	0.4811	0.90005	0.6753
2750	0.31927	0.39673	0.41127	0.80655	0.51709
5984	0.30348	0.52741	0.39205	0.8735	0.54596
7953	0.26619	0.58758	0.4382	0.87476	0.55363
1184	0.31771	0.50452	0.45182	0.91031	0.56397

The last two columns of Table 2, are the accuracy results separately for the cores and $\wp$-clusters. The values in these columns illustrate that the cores we obtained are effective, considering either the spatial or non-spatial constraints. The high accuracy values of cores in comparison with the $\wp$-clusters illustrate that the way the classes (in this case the range of house prices) are grouped, are based more on their spatial coordinates than the non-spatial features.
The next effectiveness measure is cluster assignment errors. Table 3 and Fig. 2 illustrate the number of misclassified objects using varying approaches. From Table 3 and Fig. 2 we observe by using fusion based approach the number of objects misclassified are significantly lower than the competitor algorithms. Note that even though k-means used the spatial attributes it performed very poorly in comparison to the proposed fusion based approach.

Table 3. Total Number of Misclassified objects

#Objects	KM	GDBSCAN	GraviClust	Proposed Fusion Based Clustering
694	351	550	305	213
1191	721	483	365	211
1931	1200	840	1002	193
2750	1872	1659	1619	532
5984	4168	2828	3638	757
7953	5836	3280	4468	996
11841	8079	5867	6491	1062

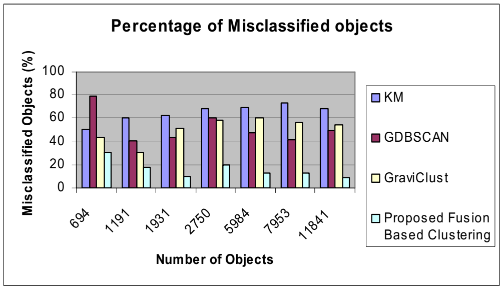


Figure 2. The percentage of misclassified objects

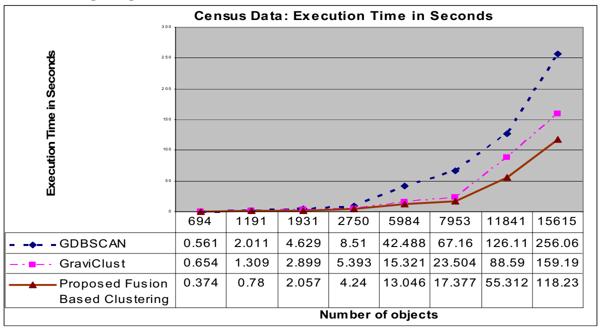
The final evaluation of effectiveness is based on the number of discovered clusters. Table 4 demonstrates the number of spatial partitions we found, are closer to the actual number of classes in the original data. In each core there are varying numbers of $\wp$-clusters, since all the regions might not have same number of classes (in this case household price range).

Table 4. The Number of Actual Classes Compared to The Number of Discovered Cores/ρ-Clusters

DataSize	Actual Classes	Number of discovered Cores	Non-Spatial Clusters /Core	Number of Discovered Clusters
108	4	3	(3,1,2)	3
694	5	5	(4,3,1,1)	4
1191	5	6	(5,4,4,3,1)	6
1931	5	8	(5,4,3,1,1)	6
2750	6	8	(2,3,2,1,1)	6
5984	6	5	(3,2,2)	5

3.3 Efficiency

As we iterate for an optimal solution, the number of data objects that need to be clustered decreases significantly. Hence our algorithm is more efficient, because the algorithm works on progressively reduced problem space. The efficiency can be further improved by using spatial indexing for the core clustering. Fig. 3 illustrates the efficiency of our algorithm in comparison to the other spatial clustering algorithms.



	694	1191	1931	2750	5984	7953	11841	15615
GDBSCAN	0.561	2.011	4.629	8.51	42.488	67.16	126.11	256.06
GraviClust	0.654	1.309	2.899	5.393	15.321	23.504	88.59	159.19
Proposed Fusion Based Clustering	0.374	0.78	2.057	4.24	13.046	17.377	55.312	118.23

Figure 3. Execution Time in Seconds on Dataset1 with Varying Number of Objects

The efficiency results as applied on forest dataset are illustrated Table 5. Here we show the iterations as opposed to time in secs, because we keep the dataset size constant to 5000 points, and vary the number of classes. Here it can be seen that the proposed approach converges much faster than KM. The iterations do not apply for GDBSCAN as it is a single pass algorithm.

Table 5. DataSet 2: Number of iterations for convergence for 3,5& 7 clusters

#Classes	KM	GraviClust	Proposed Fusion Based Clustering
3	14	12	11
5	21	20	19
7	46	17	15

In Table 6, we summarize the characteristics of our proposed approach to some of the other classes of clustering algorithms supporting spatial features.

Table 6. Advantages of the Proposed Fusion based Spatial Clustering

	Distance Metrics	K-required	Detects Noise	Requires parameters/thresholds	Arbitary shape clusters	Support for Incremental
Partition	Spatial ,or, Non-Spatial	Y	N	Y	N	Y
Density	Non-Spatial Filter	N	Y	Y	Y	N
Gravity	Spatial	Y	N	Y	N	N
Proposed Fusion Clustering	Fusion Based	N	Y	N	Y	Y

4. Conclusions

In this paper, we proposed an iterative and automated spatial clustering algorithm based on the spatial and non-spatial attribute fusion. The experimental evaluation shows that our approach is more effective than the existing approaches, which either consider location attribute as if it is another non-spatial feature, or ignore non-spatial attribute information altogether. The proposed clustering is automated because it does not require any user input, and employs a heuristic approach to find the optimal number of spatial core clusters. We also proposed a fusion based metric to find the optimal clusters and a stopping criterion for algorithm convergence. In summary, the proposed algorithm has following characteristics : i*nsensitive* to input order; a*utomatic*, that is the number of clusters need not be specified as input; iteratively clusters the data based on *fusion* of spatial and non-spatial attributes; i*ntuitive* spatial cluster comparison; permits the use of separate *distance metrics* for spatial and non-spatial features.

5. References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Introduction to Algorithms: MIT Press, 1990.

[2] U. S. D. o. Defence, "Data fusion lexicon," 1991.

[3] M. Ester, H. P. Kriegel, J. Sander, and X. X, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," Proc. 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226-231, 1996.

[4] P. S. Huang and T.-M. Tu, "A Target Fusion-Based Approach for Classifying High Spatial Resolution Imagery," Proc. 2003 IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data, pp. 175 - 181, 2003.

[5] M. Indulska and M. E. Orlowska, "Data Clustering: Gravity Based Spatial Clustering," Proc. 10th ACM International Symposium on Advances in Geographic Information Systems, pp. 125 - 130, 2002.

[6] K. Koperski and J. Han, "Discovery of Spatial Association Rules in Geographic Information Databases," Proc. 4th International Symposium on Large Spatial Databases, pp. 47-66, 1995.

[7] D. Papadias, Y. Theodoridis, T. Sellis, and M. Egenhofer, "Topological Relations in the World of Minimum Bounding Rectangles: A study with R-trees," Proc. ACM SIGMOD International Conf. on Management of Data, pp. 92-103, 1995.

[8] J. Roddick and B. G. Lees, "Paradigms for Spatial and Spatio-Temporal Data Mining," in Geographic Data Mining and Knowledge Discovery, H. Miller and J. Han, Eds.: Taylor & Francis, 2001.

[9] Y. Tao, J. Zhang, D. Papadias, and N. Mamoulis, "An Efficient Cost Model for Optimization of Nearest Neighbor Search in Low and Medium Dimensional Spaces," IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 10, pp. 1169-1184, 2004.

[10] D. K. Tasoulis and M. N. Vrahatis, "Generalizing the K-Windows Clustering Algorithm in Metric Spaces," Mathematical and Computer Modelling (accepted for publication), 2005.

[11] L. Wald, "Some Terms of Reference in Data Fusion," IEEE Transactions on Geoscientific Remote Sensing, vol. 37, no. 3, pp. 1190-1193, 1999.

[12] X. Wang and H. J. Hamilton, "Clustering Spatial Data in the Presence of Obstacles," International Journal on Artificial Intelligence Tools, vol. 14, no. 1-2, pp. 177-198, 2005.

[13] Y. Xie, V. V. Raghavan, P. Dhatric, and X. Zhao, "A New Fuzzy Clustering Algorithm For Optimally Finding Granular Prototypes," International Journal of Approximate Reasoning, vol. 40, no. 1-2, pp. 109-124, 2005.

[14] R. Xu and D. Wunsch, II, "Survey of clustering algorithms," IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 645- 678, 2005.