2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS): SmartCity17: The 3rd IEEE INFOCOM Workshop on Smart Cities and Urban Computing

Automated Density-Based Clustering of Spatial Urban Data for Interactive Data Exploration

Erica Rosalina
Computer Science and Information Technology
School of Science
RMIT University
Melbourne, Australia
Email: s3340587@student.rmit.edu.au

Flora D. Salim
Computer Science and Information Technology
School of Science
RMIT University
Melbourne, Australia
Email: flora.salim@rmit.edu.au

Timos Sellis
School of Software and Electrical Engineering
Swinburne University
Melbourne, Australia
Email: tsellis@swin.edu.au

***Abstract*—This paper presents a method to automatically estimate parameters for density-based clustering based on data distribution. It also includes several techniques for visualizing the clusters over a map, useful for interactive data exploration. The proposed method enables parameter estimation to automatically adapt to multiple resolutions, allowing the clusters to be recomputed and visualized interactively at query time with the changes of zoom levels and panning of the map. We apply a voting scheme with existing cluster indices to rank the clustering results. The framework of multi-resolution density-based clustering and visualization is implemented and evaluated using a real-world road crash datasets.**

I. Introduction

Interactive data exploration enables users to discover interesting patterns [1] and extract relationships hidden in large datasets visually [2] in a highly ad-hoc and interactive process [3]. In order to enable interactive data exploration over a big volume of spatial data, there is a need to summarize and discover patterns from the data, and one method is to use clustering.

Not all existing clustering algorithms proposed so far are applicable for clustering spatio-temporal data from cities or road networks. One method to discover interesting patterns from data is by **computing the density of data points in space**. However, existing density-based clustering algorithms, such as *DBSCAN* [4], the most widely-used density-based algorithm, and *HDBSCAN* [5], one of the most recent DBSCAN enhancements, require input parameters. In a dynamic map-based application, it is not efficient to require users to input the parameters whenever the clustering process needs to be executed, especially when it is performed intermittently. In addition, the parameters are often hard to determine and greatly affect the cluster quality, and users might not have prior knowledge about the data to help them decide the parameter values. Therefore, this is one of the research challenges addressed in this paper - how one can automatically estimate the required parameters of the clustering algorithms without having to rely on user's assistance (e.g. DBSCAN depends on how well one can estimate Eps (ε), which defines how close points should be to each other to be considered part of a cluster; and m_{pts}, which represents the number of neighbours each point should have to be considered part of a cluster).

Second, in order to correctly **visualize the results** of clustering during the interactive data exploration process on a map, one needs to accommodate situations when a user zooms in and out (i.e. resolution changes). In a higher resolution scenario (i.e. when the map is zoomed out), only a few of the major roads are shown, while smaller roads only appear in low resolution (i.e. when the map is zoomed in). Ideally, different resolutions should lead to the different partitioning of clusters as well. Hence, the computation of the clustering parameters is presented in this paper in order to adjust to various resolutions and multiple visualizations of the clustering results. As a result, the contribution that is brought upon our proposed method can be used to facilitate interactive visualization applications [6], especially when dynamic granularity is crucial at different zoom level.

These problems are addressed in this paper with the use of historical road crash data and road safety analytics as a case study and experimental evaluation. One of the outcomes of this research is an interactive map of road crashes that can assist road authorities in finding areas of high risk.

II. Background and Related Work

A. Density-Based Clustering Algorithms

Clustering refers to the process of grouping together similar objects while keeping dissimilar objects in different groups.

Density-based method [7] treats clusters as dense regions (high density) in the data space that are separated by sparse regions (low density). DBSCAN (Density-based spatial clustering of applications with noise) [4] is one of the most popular density-based methods. It does not need to know the number of clusters beforehand. However, it requires two input parameters: ε (Eps) and m_{pts}. The general idea of DBSCAN is that each point of a cluster should contain at least a minimum number of m_{pts} points in its ε-neighbourhood. However, DBSCAN is very sensitive to the selection of its parameters: ε and m_{pts}. It also does not take into account varying density in the datasets, which is caused by the use of ε as its global density threshold.

Many DBSCAN-based algorithms tried to improve the problem of clusters with very different densities in DBSCAN. For example, VDBSCAN [8] and DMDBSCAN [9] both attempt to tackle this problem by using the k-dist plot to find multiple

978-1-5386-2784-6/17/$31.00 ©2017 IEEE

ε values. Sharp changes in the k-dist plot correspond to a suitable value for ε. However, both algorithms require the user to choose the ε values from the plot manually. HDBSCAN [5] is quite different compared to the other mentioned algorithms because it combines DBSCAN with a hierarchical approach. Instead of using a single global density threshold like in DBSCAN, HDBSCAN able to find nested clusters by generating a complete density-based clustering hierarchy. Although it requires 2 parameters, $MinPts$ and m_{clSize}, the latter one is often set to be the same with $MinPts$ which turns $MinPts$ into a single parameter that acts as both a smoothing factor and a threshold for the cluster size.

In conclusion, none of the DBSCAN enhancements is totally parameter-free and most of them require some prior knowledge about the dataset that is used or heavily rely on human assistance. Our work will focus on how to estimate DBSCAN & HDBSCAN parameters in order to get better clustering results for linear constrained data like the road accident dataset.

B. Cluster Indices

Since clustering is an unsupervised learning technique, the clusters are not known apriori and different algorithms partition the dataset differently. The next issue that comes up is how to evaluate the clustering results to find partitions that best fit the underlying data. Most algorithms used 2-D datasets since it can be easily visualized and the validity of the clusters often can be verified simply from the visualization.

There are three approaches in examining the cluster validity based on the following criteria respectively [10]: 1) *external criteria*: evaluate based on a pre-specified structure imposed on a dataset, i.e. external information that is not contained in the dataset [11]; 2) *internal criteria*: evaluate the clustering algorithm results in terms of quantities that involve the vectors of the dataset themselves (e.g. proximity matrix), i.e. information is intrinsic to the dataset alone and no external information provided; 3) *relative criteria*: compare the results to other clustering schemes, resulting by the same algorithm but with different parameter values. Each of the above criteria also has its corresponding indices used as the numerical measurement. We will only discuss 6 internal indices that will be used in our evaluation, which include C index [12], Calinski-Harabasz [13], Davies-Bouldin [14], Dunn [15], Silhouette [16] and Xie-Beni [17].

C. Related Work

A number of validation techniques for density-based clustering are reviewed and evaluated in [18]. However, the method uses datasets with ground truth information (direct observation to compare with the information provided by the inferences), whereas the dataset in our research does not have ground truth. Several of the internal evaluation measures in this paper are also used in our research. A survey of algorithms and evaluation techniques for density-based data stream clustering is presented in [19], which are not directly applicable to our research, as the reviewed techniques are for data streams. A framework for computing hierarchical estimates for density-based cluster trees, outliers, and visualizing the clusters is proposed in [20]. Our paper follows a similar intuition of computing density estimates based on the data distribution, however, [20] presents the method for computing the level sets of density including the local and global outliers. Whereas, in this paper, the focus is on the automatic computation of the density at every level of user query in the interactive data exploration process, regardless of the level of data resolution.

III. Clustering Methodology: Automated Parameter Selection for Spatial Clustering

A. Automated Parameter Selection with DBSCAN

Since DBSCAN requires 2 parameters, Eps and m_{pts}. It is important to choose sensible values for these parameters since the resulting clusters are highly dependent on the choice of parameters. The proposed technique from [4] for determining Eps is to plot a *sorted k-dist graph* of which the first "valley" or the "knee" is to become the Eps value. However, they only discussed an approach to determine Eps without providing enough information on how to choose a good m_{pts} value. Therefore for the m_{pts} value, we are going to adopt the simple heuristic applied by [21] which suggests

$$m_{pts} = ln(N) \tag{1}$$

where N is the size of the dataset. However, we use the *number of visible points* to replace N. In this case, the number of visible points changes with respect to the resolution and can be less than the whole dataset size.

For a given value k which is often set to be equal to m_{pts} (i.e. $k = m_{pts}$), let k-distance be the distance from a point p to its k^{th}-nearest neighbor. If we find the k-distance of every point in the database, sort the k-distances, and then plot the sorted k-distances as a graph, that graph is what we referred to as the sorted k-dist graph.

The first "valley" or the "knee" of the sorted k-dist graph is called the *threshold point*. The terms "valley", "knee" and "threshold point" will be used interchangeably from now on. The knee corresponds to a sharp change in the density distribution amongst points. For this reason, Eps is often set to be equal to the threshold point value.

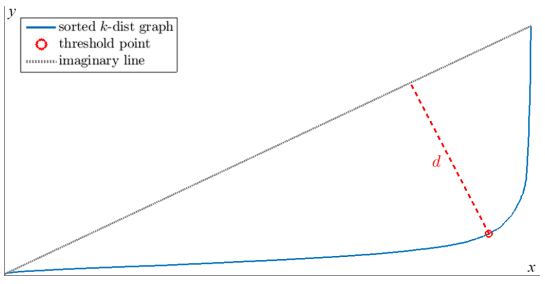


Fig. 1: Finding threshold point of a sorted k-dist graph using geometrical approach. d illustrates the distance from the threshold point to the imaginary line.

Our study employs a *geometric approach* to automatically find the threshold point from a sorted k-dist graph. This geometric approach is an existing method used for finding the *knee of a curve*.The general idea of this method is to draw an *imaginary* straight line from the first point to the last point of the curve and retrieve the point in the curve that is furthest from the imaginary line, i.e. the point which has *maximum distance* d from the imaginary line. In our case, basically the curve is the sorted k-dist graph (example shown in Figure 1).

ALGORITHM 1: Find knee point of a curve: FindKneePoint(K)

Input: K = a sorted array of curve points
$maxIndex \leftarrow 0$;
$maxDist \leftarrow -1$;
$N \leftarrow$ size of K;
$x_1 \leftarrow 0$ ▷ x_1 = first index of array K
$y_1 \leftarrow K[x_1]$ ▷ y_1 = first element of array K
$x_2 \leftarrow N-1$ ▷ x_2 = last index of array K
$y_2 \leftarrow K[x_2]$ ▷ y_2 = last element of array K
for $i \leftarrow 0$ *to* N **do**
  $currDist \leftarrow \frac{|(y_2-y_1)x_0-(x_2-x_1)y_0+x_2y_1-y_2x_1|}{\sqrt{(y_2-y_1)^2+(x_2-x_1)^2}}$ ▷ Equation 2
  if $maxIndex = 0$ *or* $currDist > maxDist$ **then**
    $maxDist \leftarrow currDist$;
    $maxIndex \leftarrow i$;
  end
end
return $K[maxIndex]$;

The general algorithm of the geometric approach is described in Algorithm 1. In the algorithm, the curve is represented as an array K in which the curve's x-axis represents the array indexes and the corresponding y-axis represents the array values. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be the curve's starting point and ending point respectively. These two points form the imaginary line. Then, we calculate the distances from every point/element in the array to the imaginary line and take the point with the largest distance as the knee point. The distance D from the imaginary line to point (x_0, y_0) is:

$$D(P_1, P_2, (x_0, y_0)) = \frac{|(y_2-y_1)x_0-(x_2-x_1)y_0+x_2y_1-y_2x_1|}{\sqrt{(y_2-y_1)^2+(x_2-x_1)^2}} \quad (2)$$

In order to calculate the threshold point, the sorted k-dist graph should be represented as an array of sorted k-distances (k-distance = distance from a point to its k^{th}-nearest neighbor). Let K represents the array & N be the size of array K, which should also be the same with the dataset size. The starting point P_1 corresponds to the first element of the array, while the ending point P_2 corresponds to the last element of the array. Assuming a zero-based array indexing is used, x_1 equals to the index of the first element (i.e. $x_1 = 0$) and x_2 equals to the index of the last element (i.e. $x_2 = N-1$). Correspondingly, $y_1 = K[x_1]$ and $y_2 = K[x_2]$. Thus, by passing array K as the parameter to Algorithm 1, we will get the threshold point as the return value which is basically our Eps value.

To summarize, in order to automatically determine DBSCAN parameters, we set $m_{pts} = ln(N)$ where N is the dataset size, then work out the "threshold point" of the sorted k-distance plot ($k = m_{pts}$) by applying the geometric approach and assign the threshold point to Eps.

We evaluate the choice of DBSCAN parameters with regards to the quality of clusters by incorporating the automated parameter selection techniques to determine the parameters. We use Euclidean distance as the distance measure. The clusters quality is assessed from the visualization and by the six cluster validity indices previously reviewed: C index [12] (denoted as **C**), Calinski-Harabasz [13] (denoted as **CH**), Davies-Bouldin [14] (denoted as **DB**), Dunn [15] (denoted as **D**), Silhouette [16] (denoted as **S**) and Xie-Beni [17] (denoted as **XB**). A voting system is applied to choose the best clusters based on the number of best indices.

B. Automated Parameter Selection with HDBSCAN

There are two parameters affecting the results of HDBSCAN, m_{pts} and m_{clSize}. Since HDBSCAN's m_{pts} is equivalent to DBSCAN's m_{pts}, the same approach for calculating DBSCAN's m_{pts} (Equation 1) is applied in here:

$$m_{pts} = ln(N) \quad (3)$$

where N is the size of the dataset. We incorporate two ways to calculate m_{clSize} in our system. The first one is the *normal* approach where $m_{clSize} = m_{pts}$ (referred as **HDBSCAN Normal**). This approach is often used to simplify HDBSCAN and create the impression that HDBSCAN is a single parameter algorithm (i.e. it only requires 1 parameter m_{pts}). The second one is the *mode* approach which is also our proposed method (referred as **HDBSCAN Mode**).

The mode approach estimates m_{clSize}, which represents the minimum cluster size, based on the most frequent number of neighbours that are within knee of core distances. The idea comes from the fact that the density thresholds of HDBSCAN are defined from the core distances. Consequently, we can make use of the most significant core distance to be the radius distance for finding the number of neighbours (neighbour count) that each object has; and it is reasonable to set the majority value of neighbour counts as the minimum cluster size. The algorithm works as the following:

ALGORITHM 2: m_{clSize}'s mode approach

$d_{core} \leftarrow$ core distances of every objects in dataset D;
Sort d_{core};
$d_{kneeCore} \leftarrow FindKneePoint(d_{core})$;
▷ Find knee of core distances using Algorithm 1
$C \leftarrow NeighbourCounts(D, d_{kneeCore})$ $d_{kneeCore}$ from the object. ▷ Refers to Algorithm 3
$m_{clSize} \leftarrow$ mode of set C;

ALGORITHM 3: Compute the set of neighbour counts that are within distance d

$C \leftarrow \emptyset$;
for *each object* $p \in D$ **do**
  Add $|N_d(p)|$ to C
end
return C

The algorithm requires the first stage of HDBSCAN, which is computing the core distances, to be run first. Then, the core distances need to be sorted in order to get the knee core distance. We adopt the idea of finding ε in DBSCAN to identify the significant core distance (knee of the core distances). The knee of the core distances is then used as the radius for getting the neighbour counts. The neighbour count of object p within radius d is defined as $nc_d(p) = |N_d(p)|$ with $\varepsilon = d$. Thus, the neighbour counts of every object in the dataset that are within radius d can be represented as $C = \{nc_d(p_1), nc_d(p_2), \cdots, nc_d(p_n)\}$ where $D = \{p_1, p_2, \cdots, p_n\}$ is the dataset of n objects and d is set to be the knee core distance. Subsequently, **mode** function is applied to C to get the most frequently occurred neighbour count value, which is then set to be the value for parameter m_{clSize}.

We evaluate the parameter selection of HDBSCAN and its two approaches for calculating m_{clSize} (*normal* approach & *mode* approach) in the same manner we evaluate DBSCAN. In other words, it is run in two different resolutions with various combinations of parameters, measured by the cluster validity indices, and then the voting system is employed to choose the best run.

IV. Experiment

A. Dataset

Our study uses historical road crashes data in Victoria, Australia from the period of 1 January 2006 through to 30 June 2013 and Victoria's road network data. The crash dataset is provided by VicRoads and is available from Victorian Government Data Directory website[1]. The dataset originally contains 73101 accident nodes, but it was filtered to 72176 nodes due to missing location details in some records. Our study area is limited to 63 localities in South Eastern part of Victoria, with the total number of accident nodes 7864. There are 5361 affected road segments with the total length of 1425.2 km. The total study area is approximately 1909.3 km^2.

B. Clustering Evaluation

In this section, we compare the best results of DBSCAN and the best results of HDBSCAN's two approaches (HDBSCAN Normal & HDBSCAN Mode), and identify the best out of the three approaches by using the cluster indices and visualization. We are going to make use of the existing results from previous sections by combining them together and extract only the significant ones. Those approaches will be analyzed at two resolutions: *high* resolution and *low* resolution. For each resolution, the following comparisons will be performed:

1) *overall* comparison: all parameters combination from each approach are considered for selecting the best outcome of every index; only the best ones are further analyzed
2) *indices best* comparison: only the best run(s) (according to the indices results) of each approach is/are considered
3) *default* comparison: only the default run (i.e. with default m_{pts}) from each approach is considered, so the comparison is always done on 3 rows (1 for each approach) with identical m_{pts} value

Each approach has 8 runs (i.e. rows), thus there are a total of 24 runs [8 (runs per approach) $\times$ 3 (approaches)], all with different parameter combinations ranging from $m_{pts} = 8$ to $m_{pts} = 15$. The results of the comparisons (Table I) clearly shows that HDBSCAN is much more favored than DBSCAN.

In the overall comparison (Table Ia), DBSCAN has no run (or row) simply because none of its runs is selected as the best by the indices when compared to the HDBSCAN approaches. Meanwhile, there are three runs tie in votes where the two of them belong to HDBCSAN Mode. However, the visualization shows that HDBSCAN Mode is better than HDBSCAN Normal in high resolution due to the clusters in the latter approach being too fine-grained, thus not suitable for high resolution.

Nevertheless, HDBSCAN Mode is the best out of the three approaches in high resolution from the visual perspective. Although the clusters are not too small like in HDBSCAN Normal, different densities of the clusters can still be detected unlike in DBSCAN approach.

TABLE I: Comparisons of all approaches in high resolution [Best (Highlighted)]

(a) *Overall* Comparison

	m_{pts}	ε	m_{clSize}	C	CH	DB	D	S	XB	
DBSCAN	-	-	-	-	-	-	-	-	-	
HDBSCAN Normal	9	-	9	0.0068	14234.6935	**0.1285**	0.0107	**0.6981**	50.9123	Fig. 3b
HDBSCAN Mode	10	-	42	**0.0062**	**22919.2289**	0.2213	0.0272	0.6374	21.5041	Fig. 4b
	13	-	55	0.0199	11660.6816	0.4520	**0.0444**	0.6404	11.3414	
	15	-	60	0.0101	17637.9780	0.2205	**0.0444**	0.6529	**8.0308**	Fig. 4c

(b) *Indices Best* Comparison

	m_{pts}	ε	m_{clSize}	C	CH	DB	D	S	XB	
DBSCAN	14	2028.24	-	0.0704	3898.0643	0.2088	0.0414	0.6640	18.3471	
HDBSCAN Normal	9	-	9	0.0068	14234.6935	**0.1285**	0.0107	**0.6981**	50.9123	Fig. 3b
	13	-	13	0.0064	17200.2566	0.1705	0.0268	0.6956	19.5653	
HDBSCAN Mode	10	-	42	**0.0062**	**22919.2289**	0.2213	0.0272	0.6374	21.5041	Fig. 4b
	15	-	60	0.0101	17637.9780	0.2205	**0.0444**	0.6529	**8.0308**	Fig. 4c

(c) *Default* Comparison

	m_{pts}	ε	m_{clSize}	C	CH	DB	D	S	XB	
DBSCAN	8	1092.41	-	0.0208	7302.8705	0.2957	0.0124	0.6396	137.5297	
HDBSCAN Normal	8	-	8	**0.0085**	12674.4550	**0.2010**	0.0187	**0.6889**	44.4553	Fig. 3a
HDBSCAN Mode	8	-	35	0.0143	**15247.0290**	0.3151	**0.0246**	0.6367	**28.6544**	Fig. 4a

Each approach evaluated on low resolutions has 9 runs (m_{pts} between 7 to 15), so there are 27 runs in total. The results of the comparisons are presented in Table II. Similar to the high resolution results, DBSCAN also does not receive a single vote in low resolution. In this case, at least the indices and the visualization both agree with each other.

In overall comparison (Table IIa), there are 4 best runs because some of the indices best values happen to be the same in several runs, like those in indices D & XB. In fact, there are only 3 distinct best runs because when $m_{pts} = 14$, the generated m_{clSize} from both HDBSCAN approaches are very

[1]Source: https://www.data.vic.gov.au/data/dataset/crash-stats-data-extract

similar, i.e. one is 14 while the other is 15. Hence, the resulting clusters between those two runs are identical (Fig. 5b & 2b) and so does the indices values. You might also notice that some values in the table appear to be the same but while one is considered as the best (i.e. bold text), the others are not. For example the S values in (a) HDBSCAN Normal $m_{pts} = 14$, (b) HDBSCAN Mode $m_{pts} = 13$ and (c) HDBSCAN Mode $m_{pts} = 14$ (Table IIa). Only (b) is treated as the best value for S because the values in the tables are actually rounded to the nearest decimal, the real values of (a) & (c) are smaller than (b).

The visual aspects of the overall comparison best runs have clearly shown that the actual best run is only one and that is when HDBSCAN Mode $m_{pts} = 10$ (Fig. 6b). The other three supposedly best runs (Fig. 5b, 2a, and 2b) are not useful for low resolution for the same reason with DBSCAN (i.e. the clusters are so large that they do not take into account that the accident data is distributed along the road network).

(a) $m_{pts} = 13$ (b) $m_{pts} = 14$

Fig. 2: HDBSCAN *mode* in low resolution for *overall* comparison (Table IIa)

In the indices best comparison (Table IIb), again DBSCAN does not get a single vote from the indices and the outcome is a tie between HDBSCAN Normal & HDBSCAN Mode with 3 votes each. On the other hand, the default comparison (Fig. IIc) shows a clear win for HDBSCAN Mode. This implies that out of the three approaches, HDBSCAN Mode is the most appropriate to run if the users need to get some reasonable partitioning on the accident data without any background knowledge required. Although it should be noted that in low resolution there is only a slight difference between the default HDBSCAN Mode (Fig. 6a) & the default HDBSCAN Normal (Fig. 5a), mainly due to similar m_{clSize} values.

In summary, HDBSCAN outperforms DBSCAN both in high and low resolutions. Although the performance of DBSCAN is average in the high resolution scenario, it may not be suitable for a low resolution application due to the fact that the accident data is linearly constrained by the road network, which eventually resulted in poor performance.

Between the two HDBSCAN approaches, HDBSCAN Mode gives better partitioning compared to HDBSCAN Normal in high resolution. On the other hand, both approaches are head to head in low resolution.

V. Conclusion

In this paper, we have proposed an adaptive clustering method to enable an interactive data exploration of road accidents. Our system can automatically calculate the parameters

TABLE II: Comparisons of all approaches in low resolution [Best (Highlighted)]

(a) *Overall* Comparison

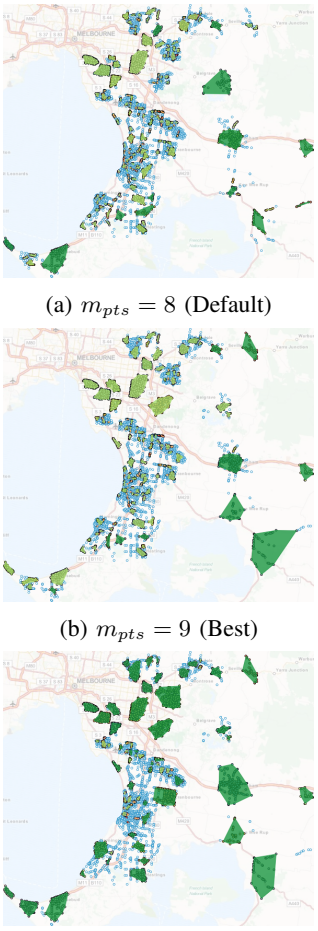
	m_{pts}	ε	m_{clSize}	C	CH	DB	D	S	XB	
DBSCAN	-	-	-	-	-	-	-	-	-	
HDBSCAN Normal	13	-	13	0.0349	2329.5368	**0.1580**	0.0818	0.7074	5.5712	
	14	-	14	0.0179	3073.5154	0.4101	**0.1573**	0.7086	**2.8608**	Fig. 5b
HDBSCAN Mode	10	-	12	**0.0109**	**3303.1032**	0.1728	0.0913	0.6294	3.8294	Fig. 6b
	13	-	38	0.0180	3059.3662	0.4101	**0.1573**	**0.7086**	2.8609	Fig. 2a
	14	-	15	0.0179	3073.5154	0.4101	**0.1573**	0.7086	**2.8608**	Fig. 2b

(b) *Indices Best* Comparison

	m_{pts}	ε	m_{clSize}	C	CH	DB	D	S	XB	
DBSCAN	9	710.20	-	0.0310	1294.5558	0.4337	0.1098	0.6317	5.5642	
HDBSCAN Normal	14	-	14	0.0179	3073.5154	0.4101	**0.1573**	**0.7086**	**2.8608**	Fig. 5b
HDBSCAN Mode	10	-	12	**0.0109**	**3303.1032**	**0.1728**	0.0913	0.6294	3.8294	Fig. 6b

(c) *Default* Comparison

	m_{pts}	ε	m_{clSize}	C	CH	DB	D	S	XB	
DBSCAN	7	586.15	-	0.0362	627.2269	0.6294	0.0291	0.5974	78.3715	
HDBSCAN Normal	7	-	7	0.0140	2692.1316	0.2387	0.0574	0.6720	6.3584	
HDBSCAN Mode	7	-	6	**0.0124**	**2968.1770**	**0.2198**	**0.0745**	**0.6771**	**5.2215**	Fig. 6a

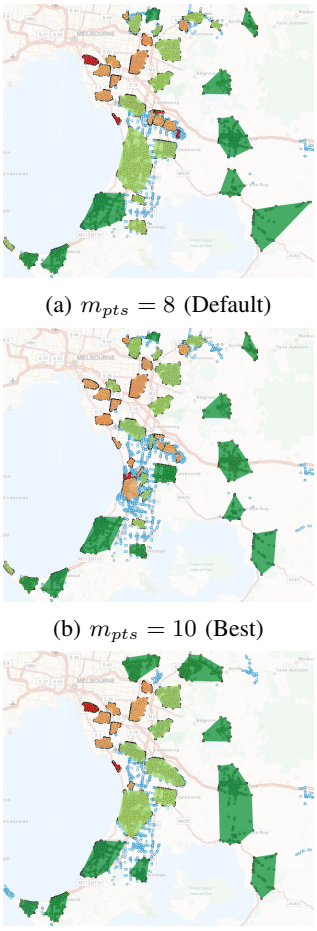


(a) $m_{pts} = 8$ (Default)

(b) $m_{pts} = 9$ (Best)

(c) $m_{pts} = 13$ (Best)

Fig. 3: HDBSCAN *normal* in high resolution

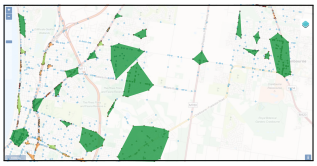


(a) $m_{pts} = 8$ (Default)

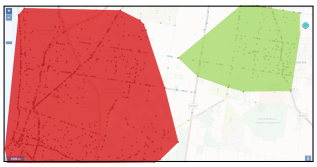
(b) $m_{pts} = 10$ (Best)

(c) $m_{pts} = 15$ (Best)

Fig. 4: HDBSCAN *mode* in high resolution

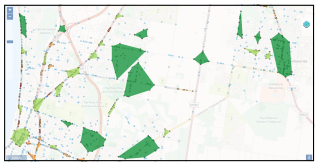


(a) $m_{pts} = 7$ (Default)

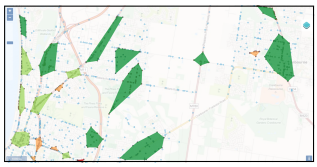


(b) $m_{pts} = 14$ (Best)

Fig. 5: HDBSCAN *normal* in low resolution



(a) $m_{pts} = 7$ (Default)



(b) $m_{pts} = 10$ (Best)

Fig. 6: HDBSCAN *mode* in low resolution

required to generate the clusters so that user does not need to have prior knowledge on the dataset to run the clustering. However, the user control for visualization is not limited to the calculated parameters as they can freely adjust the parameters as required to allow for interactive data exploration. In addition, the clustering process adjusts well to various resolutions of the map. In order to enable interactive data exploration for various zoom levels and panning interactions, our system utilizes the visible points to be used as the dataset in the clustering process to adapt to different zoom levels or resolutions. In this case, the visible points will change accordingly according to the given resolutions. We make use of two existing clustering algorithms: DBSCAN & HDBSCAN, and identify the one that can take into account for the fact that the accidents happen on roads. This condition is particularly crucial in lower resolution when the road network is visible.

We propose a new approach to estimate HDBSCAN's parameter called *HDBSCAN Mode* that can produce clusters applicable for both high and low resolution. The results show that HDBSCAN Mode works well in any resolutions and can take into account linear-constrained data. Our evaluation shows that DBSCAN can produce quite reasonable clusters in higher resolution but works poorly in lower resolution. Thus, it cannot take into consideration that the data is constrained by the road networks. On the other hand, the normal approach of HDBSCAN works very well in lower resolution. However, it produces clusters that are too fine grained in higher resolution.

This approach can be adapted to multiple different domains in smart city and urban computing where the proliferation of spatial and temporal data makes data summarization, visualization, and exploration challenging.

Acknowledgment

This research is partly supported by ARC Discovery Project (DP160102114) and RMIT Sustainable Urban Precint Project 'iCommunity'.

References

[1] M. v. Leeuwen, "Interactive Data Exploration Using Pattern Mining," in *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*, ser. Lecture Notes in Computer Science, A. Holzinger and I. Jurisica, Eds. Springer Berlin Heidelberg, 2014, no. 8401, pp. 169–182.

[2] "A Framework for Knowledge-based, Interactive Data Exploration," *Journal of Visual Languages and Computing*, vol. 5, pp. 339–363, 1994.

[3] K. Dimitriadou, O. Papaemmanouil, and Y. Diao, "Explore-by-example: An Automatic Query Steering Framework for Interactive Data Exploration," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14. New York, NY, USA: ACM, 2014, pp. 517–528.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, vol. 96, no. 34, 1996, pp. 226–231.

[5] R. J. Campello, D. Moulavi, and J. Sander, "Density-Based Clustering Based on Hierarchical Density Estimates," in *Advances in Knowledge Discovery and Data Mining*, ser. Lecture Notes in Computer Science, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Springer Berlin Heidelberg, 2013, vol. 7819, pp. 160–172.

[6] J. Liono, F. D. Salim, and I. F. Subastian, "Visualization oriented spatiotemporal urban data management and retrieval," in *Proceedings of the ACM First International Workshop on Understanding the City with Urban Informatics*. ACM, 2015, pp. 21–26.

[7] J. Han, M. Kamber, and J. Pei, *Data Mining : Concepts and Techniques*, 3rd ed., ser. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011.

[8] P. Liu, D. Zhou, and N. Wu, "VDBSCAN: Varied Density Based Spatial Clustering of Applications with Noise," in *Service Systems and Service Management, 2007 International Conference on*, June 2007, pp. 1–4.

[9] M. T. Elbatta and W. M. Ashour, "A Dynamic Method for Discovering Density Varied Clusters." *International Journal of Signal Processing, Image Processing & Pattern Recognition*, vol. 6, no. 1, 2013.

[10] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 1999.

[11] E. Rendón, I. M. Abundez, C. Gutierrez, S. D. Zagal, A. Arizmendi, E. M. Quiroz, and H. E. Arzate, "A comparison of internal and external cluster validation indexes," in *Proceedings of the 2011 American Conference, San Francisco, CA, USA*, vol. 29, 2011.

[12] L. Hubert and J. Schultz, "QUADRATIC ASSIGNMENT AS A GENERAL DATA ANALYSIS STRATEGY," *British Journal of Mathematical and Statistical Psychology*, vol. 29, no. 2, pp. 190–241, 1976.

[13] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974.

[14] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-1, no. 2, pp. 224–227, April 1979.

[15] J. C. Dunn, "Well-Separated Clusters and Optimal Fuzzy Partitions," *Journal of Cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.

[16] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, no. 0, pp. 53 – 65, 1987.

[17] X. Xie and G. Beni, "A Validity Measure for Fuzzy Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 841–847, 1991.

[18] D. Moulavi, P. Jaskowiak, R. Campello, A. Zimek, and J. Sander, "Density-Based Clustering Validation," in *Proceedings of the 2014 SIAM International Conference on Data Mining*, ser. Proceedings. Society for Industrial and Applied Mathematics, Apr. 2014, pp. 839–847.

[19] A. Amini, T. Y. Wah, and H. Saboohi, "On Density-Based Data Streams Clustering Algorithms: A Survey," *Journal of Computer Science and Technology*, vol. 29, no. 1, pp. 116–141, Jan. 2014.

[20] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 1, pp. 5:1–5:51, Jul. 2015.

[21] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial–temporal data," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 208–221, 2007.