Significant DBSCAN towards Statistically Robust Clustering

Yiqun Xie
xiexx347@umn.edu
Dept. of Computer Science and Engineering
University of Minnesota – Twin Cities

Shashi Shekhar
shekhar@umn.edu
Dept. of Computer Science and Engineering
University of Minnesota – Twin Cities

ABSTRACT

Given a collection of geo-distributed points, we aim to detect statistically significant clusters of varying shapes and densities. Spatial clustering has been widely used many important societal applications, including public health and safety, transportation, environment, etc. The problem is challenging because many application domains have low-tolerance to false positives (e.g., falsely claiming a crime cluster in a community can have serious negative impacts on the residents) and clusters often have irregular shapes. In related work, the spatial scan statistic is a popular technique that can detect significant clusters but it requires clusters to have certain predefined shapes (e.g., circles, rings). In contrast, density-based methods (e.g., DBSCAN) can find clusters of arbitrary shape efficiently but do not consider statistical significance, making them susceptible to spurious patterns. To address these limitations, we first propose a modeling of statistical significance in DBSCAN based clustering. Then, we propose a baseline Monte Carlo method to estimate the significance of clusters and a Dual-Convergence algorithm to accelerate the computation. Experiment results show that significant DBSCAN is very effective in removing chance patterns and the Dual-Convergence algorithm can greatly reduce execution time.

CCS CONCEPTS

• **Information systems** → **Clustering**.

KEYWORDS

statistical significance, DBSCAN, robust, clustering, spatial

ACM Reference Format:
Yiqun Xie and Shashi Shekhar. 2019. Significant DBSCAN towards Statistically Robust Clustering. In *16th International Symposium on Spatial and Temporal Databases (SSTD '19), August 19–21, 2019, Vienna, Austria.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340964.3340968

1 INTRODUCTION

Given a collection of N points in a spatial domain, we aim to detect statistically significant clusters with varying shapes and densities. The points are instances of certain events (e.g., disease, crime).

Detection of significant spatial clusters has been widely applied in important societal domains such as public health, public safety, transportation and environmental science. In public health, epidemiologists use significant clusters (a.k.a, hotspots) to monitor and alert the public about disease outbreaks (e.g., legionnaires' disease, leukemia) [9, 11]. The Research Surveillance Program at the National Cancer Institute has included significant clustering (e.g., SaTScan) as an important methodology and tool [1]. In public safety, police officers use clusters of crime cases to identify neighborhoods with abnormally high crime rates or locate serial criminals [7]. In transportation, many local governments (e.g., US states) have launched "Zero Death" initiatives to save lives from traffic-related accidents. With significant clustering, planners can find roads with significantly high rates of car accidents or pedestrian fatalities, which indicate potentially unsafe driving conditions (e.g., damaged side walks, pot holes). In forestry, forest managers use significant clusters to locate high-risk or fragile forest regions and identify potential forest health problems. In environmental science, significant clusters of pollution or contamination can be used to alert administrators of unusual regional phenomenon or problems (e.g., well water contamination by E. coli bacteria). These are just a few of the many applications using significant clustering.

In these societal applications, decisions often have big impacts, making them have low-tolerance to false positives. For example, false alarms of disease outbreaks may lead to a huge waste of limited public resources (e.g., budgets for sanitation, medication, research) and cause unnecessary social stress or anxiety among lots of people. Similarly, identifying a region as a crime cluster by mistake can greatly reduce the number of people visiting the region, lowering property values and hurting local businesses. For this reason, significance testing is very important and used in many critical societal applications to greatly reduce the risk of false positives.

In related work, the spatial scan statistic [9] is a widely used method for detecting significant clusters. The major strength of the spatial scan statistic is its inclusion of statistical significance. The method introduced a likelihood ratio based framework for testing the significance of spatial clusters and can effectively remove cluster candidates that are likely to be chance patterns. The major limitation of spatial scan statistic based methods is that they require one pre-defined geometric shape (e.g., circle [9, 13], ring [7], square [11], linear), or require a pre-defined irregular partitioning of the spatial domain (e.g., county boundaries in a state) which are not available for many applications. In real-world scenarios, the shapes of clusters are affected by many factors and may change through time, making it difficult to represent them well with pre-defined shapes or partitioning. In addition, the modeling of likelihood ratio in spatial scan statistics did not consider spatial nondeterminism, making it prone to detect very small clusters with just a few points [12, 13]. In the data mining community, DBSCAN [8] is one of the most popular methods for cluster detection, which won the 2014

SSTD '19, August 19–21, 2019, Vienna, Austria
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6280-1/19/08…$15.00
https://doi.org/10.1145/3340964.3340968

"Test of Time" Award from ACM SIGKDD. DBSCAN and its variations (e.g., OPTICS [4], H-DBSCAN [6]) are well-known for their ability to detect clusters of varying geometric shape in the presence of noise. It does not require any user input on cluster shapes and can automatically capture them from the data. One major limitation of DBSCAN is that it does not consider the statistical significance of detected clusters [10], leaving space for spurious and chance patterns in the output. While DBSCAN has a default modeling of noise, the type of noise it considers is hard-thresholded and can only handle simple un-clustered points at the per-point level. In a complete random point distribution (i.e., no clustered regions), points that appear near each other just by chance may be treated as valid detections in DBSCAN. This issue was also realized by the creators of the original DBSCAN and OPTICS [6]: "small clusters of objects that may be highly similar to each other just by chance". However, there is still no general modeling of statistical significance for DBSCAN to resolve this problem. A remedy used by some users is to use a threshold of minimum cluster size (e.g., 5 points as defaulted in [2] and many others). While this is an encouraging start, we will show that the absolute definition or value of "small" cannot accurately or appropriately reflect the statistical randomness or significance in the DBSCAN setup (Sec. 2.2).

To address these limitations, we aim to join the strengths of statistics and computer science to improve the flexibility and robustness of spatial clustering. Our contributions are: (1) we explore, propose and compare several modelings of statistical significance in DBSCAN; (2) we focus on a specific modeling and propose a baseline Monte Carlo method to compute the statistical significance; (3) we propose a Dual-Convergence algorithm to improve the computational efficiency of significance testing; and (4) based on the proposed significant DBSCAN (sig-DBSCAN), we present a heuristic search method to describe the detection of clusters with varying densities in the context of significance testing.

Experiments show that the proposed sig-DBSCAN can effectively eliminate spurious patterns with significance testing and the Dual-Convergence algorithm can greatly reduce the computational cost.

2 PROBLEM DEFINITION

2.1 Key Concepts

Point distribution: A collection of N geo-distributed points of certain events (e.g., disease) in a spatial domain.

Point Process: A statistical process that governs the generation of a point distribution. It determines the probability of having a point at each location within the spatial domain. A homogeneous point process (e.g., complete spatial randomness) has identical probability across all locations (i.e., no true cluster). In contrast, a biased/clustered point process has higher probabilities for locations inside the clustered regions and lower outside. Point process is used to define the null and alternative hypotheses in significance testing.

DBSCAN: Density-Based Spatial Clustering of Applications with Noise [8]. It takes two inputs: (1) search radius ϵ, and (2) minimum number of points $minPts$. With $(\epsilon, minPts)$, DBSCAN classifies a point as a core point if its ϵ neighborhood contains at least $minPts$ points. In short, once a core point c is found, DBSCAN initializes a cluster point set $P(\epsilon)$ of all points with its ϵ neighborhood. For any point $c' \in P(\epsilon)$ that is also a core point, it expands $P(\epsilon)$ by adding all points (without duplication) in the ϵ neighborhood of c' to $P(\epsilon)$. The expansion continues until there is no unexpanded core point left in $P(\epsilon)$, including any newly added ones in the process. All the points in $P(\epsilon)$ form a single cluster. Then DBSCAN continues to find another cluster if it exists. Finally, all points that do not belong to any $P(\epsilon)$ are labeled as noise.

Test statistic: A random variable used to summarize the sample data (e.g., a cluster in a point distribution) and test the hypotheses. In this context, it can be considered as a score calculated from the data (e.g., density of a cluster in a point distribution). The significance of the score determines whether to reject the null hypothesis.

2.2 Modeling of Statistical Significance

We explore and compare several different modelings of statistical significance for DBSCAN clustering. To be specific, the significance of DBSCAN we are modeling here is in the context of a single $(\epsilon, minPts)$ combination given by a user. The use of this modeling in the context of varying $(\epsilon, minPts)$ will be discussed in Sec. 3.3.

The null hypothesis states that a cluster is generated by a homogeneous point process. In contrast, the alternative hypothesis states that it is generated by a clustered/biased point process (Sec. 2.1). Note that for significant cluster detection, it is insufficient to just check whether the entire point distribution belongs to a homogeneous point process or not (e.g., using Ripley's K function). For example, in a point distribution generated by a clustered point process, a clustering method such as DBSCAN often identifies a mixture of significant clusters and chance patterns. As a result, just knowing that the overall point distribution is clustered cannot help us filter out the chance patterns. This is why the hypotheses need to be fine-grained to cluster levels.

We explore several test statistics for the significance or hypothesis testing as shown in Table 1. For a given cluster from a point distribution, its test statistic value will be used to determine if it can be generated by the null hypothesis.

According to Table 1, both density and likelihood ratio requires calculation of the cluster's area in Euclidean space. In the framework of spatial scan statistics, cluster regions are pre-defined (e.g., all circular regions of certain areas) so it is trivial to calculate the areas (e.g., πr^2). However, area calculation is not well-defined in the DBSCAN framework whose output clusters are represented by "maximal point sets". While conceptually (or visually in low-dimension space) it is easy to sense the region covered by a point-set cluster, mathematically it becomes tricky to model such an area. For example, a convex hull is a popular model to depict the region covered by a point set. However, since DBSCAN clusters can have arbitrary shapes (e.g., concave), a convex hull modeling will introduce large errors into the estimation. An alternative is to use the ϵ neighborhoods of all core points in a cluster to approximate its area. However, this requires computation of the union of all these ϵ neighborhoods. Even for the two dimensional spatial case, just one area calculation will take $O(|c|^2 \log |c|)$ time where $|c|$ is the number of core points (or ϵ neighborhoods). It will require higher complexity and huge amount of time in higher-dimensional space.

Each test statistic involves normalization to make different clusters comparable [13]. For example, density ($d = n/a$) uses cluster area a as a normalization so it can be used to rank clusters with different areas a and number of points n, when areas are computable

Table 1: Example candidates of test statistics for DBSCAN

Test statistic	Area of cluster	Normalization	Bias towards small clusters	Computation
Density d	Required	Area	Yes [11, 13]	Area dependent
Likelihood ratio lr	Required	Area + null hypothesis	Yes (less) [12, 13]	Area dependent
Cluster size n	N/A	Search context dependent, e.g., fixed radius [13], $(\epsilon, minPts)$ in DBSCAN	No	$O(1)$ for a given cluster

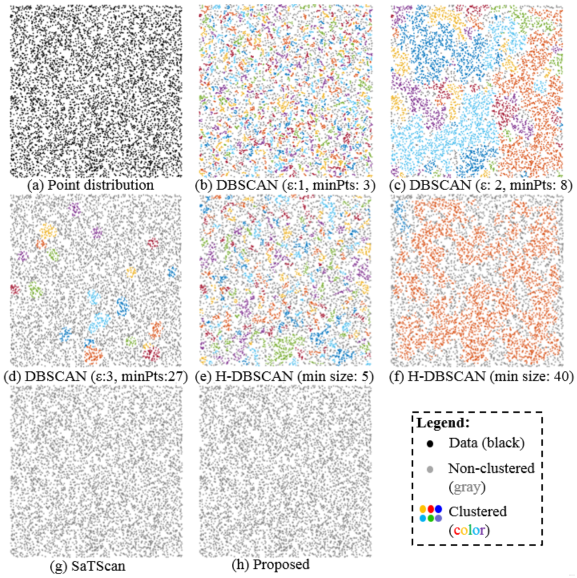

(a) Point distribution (b) DBSCAN (ε:1, minPts: 3) (c) DBSCAN (ε: 2, minPts: 8)
(d) DBSCAN (ε:3, minPts:27) (e) H-DBSCAN (min size: 5) (f) H-DBSCAN (min size: 40)
(g) SaTScan (h) Proposed

Figure 1: 2000-point data by a homogeneous point process.

or available. One major disadvantage of density is its strong bias towards small clusters. Given a cluster of density d_0, there always exists a sub-region of it that has a higher density d_1. This means that a cluster of highest density will always have the smallest area, which is not a desired property [11, 13]. To reduce the effect, likelihood ratio incorporates the null hypothesis into the normalization and is able to reduce the bias. However, it ignores the phenomenon of spatial nondeterminism, making it still susceptible to the bias and leading to incorrect rankings of candidates [12, 13] (e.g., a well-known issue is that it includes tiny patterns in its results).

Cluster size n (i.e., number of points in the cluster) is another measure being used in scan statistics methods [13] that does not require computation of the area. To make clusters comparable, it does require some normalizing or constraining conditions to be enforced into the cluster search process; otherwise a bigger set of points is always superior. For DBSCAN, the normalizing conditions come naturally through the required parameters $(\epsilon, minPts)$. The search radius ϵ and minimum number of points $minPts$ clearly define the conditions that valid cluster points must satisfy, so the size of valid clusters (point sets) n are constrained by the conditions.

In the present study, our significance modeling uses cluster size n as the test statistic because of the computational benefits (e.g., not requiring area computation).

In [6] (joint work by authors of DBSCAN and OPTICS), one issue discussed is the existence of "small" clusters in the results, which are likely to happen by chance. Here we extend this good start by formalizing the definition of "small" using statistical significance. Previously, to mitigate the "small" cluster issue, a remedy used by some users is to enforce a hard-threshold on minimum cluster size (e.g., default "5" in [2]). While intuitively small clusters (e.g., with only 3 points) are likely to be chance patterns, chance patterns are not necessarily small. Fig. 1 shows the results of DBSCAN and HDBSCAN on a random point distribution generated by a homogeneous point process. In this point distribution, all clusters detected are chance patterns. Although the chance patterns are indeed small in a few results (e.g., Fig. 1 (b)(c)(e)), they turn out to be quite large (e.g., thousands of points) in others. Thus, the exact definition of "small" has to depend on many factors, such as the input data, the DBSCAN (or HDBSCAN) parameters, the desired significance level and the null hypothesis.

In this paper we use significance testing to identify the exact threshold of "small" (i.e., minimum cluster size n_{min}) under all these factors to remove chance patterns (e.g., Fig. 1(h)).

2.3 Formal Problem Formulation

Inputs:

(1) A distribution of N geo-located points;
(2) DBSCAN parameters: $(\epsilon, minPts)$;
(3) A test statistic;
(4) A significance level α.

Output: Significant DBSCAN clusters (if they exist).
Objectives: Solution quality and computational efficiency.
Constraints: Correctness and completeness.

This formulation shows the main scope of the paper, which is to enable significant DBSCAN clustering. While the formulation is defined using a single pair of $(\epsilon, minPts)$, later in Sec. 3.3 we will show how this formulation can be used as a sub-routine to detect clusters of varying densities in the context of significance testing (i.e., requiring enumeration of multiple ϵ and $minPts$). The test statistic we use is the cluster size n.

The completeness and correctness require that the clusters detected must satisfy the DBSCAN conditions and all cluster candidates that satisfy the conditions should be enumerated. This will be guaranteed by using exact DBSCAN as a sub-routine during significance testing. Note that while the formulation is for DBSCAN, it can be applied generally to model significance in its variations (e.g., HDBSCAN with the same test statistic), assuming that the outputs are also clustered point sets.

3 SIGNIFICANT DBSCAN CLUSTERING

In this section, we first propose a baseline Monte Carlo method to evaluate the statistical significance of clusters detected by DBSCAN with a single pair of $(\epsilon, minPts)$ in Sec. 3.1. Then we propose a

Dual-Convergence algorithm to accelerate the significance testing in Sec. 3.2. Finally, we present a heuristic search strategy which uses the single pair version as a sub-routine to detect significant clusters of varying densities in Sec. 3.3.

3.1 Significance testing for a single (ϵ, $minPts$)

Here we discuss significance testing for a given pair of ($\epsilon, minPts$) in DBSCAN. Denote the significance level as α (e.g., 0.01, 0.05), the size of a detected cluster C as n_C, the total number of points in the point distribution as N, and the spatial domain of the point distribution as S. Denote $p_{null}(n_C, N, S, \epsilon, minPts)$ as the probability of having a cluster of size n_C or greater in a N-point distribution in S generated by a homogeneous point process. p_{null} is also the p-value.

Definition 3.1. Cluster C is statistically significant if its p-value $p_{null}(n_C, N, S, \epsilon, minPts) < \alpha$.

Currently there still does not exist a known statistical model that can calculate the probability p_{null} in closed-form, because the calculation of the probability needs to consider the search and expansion process of DBSCAN as well as the randomness associated with distributing N points in a spatial domain S which can have irregular shape (e.g., a sub-space of a city in Sec. 4.1.4). Thus, we use a Monte Carlo method to estimate p_{null}.

3.1.1 *A Baseline Algorithm with Monte Carlo Estimation.*

In Monte Carlo estimation (Alg. 1), we generate M simulation trials to approximate the distribution of cluster size n (i.e., the test statistic) in point distributions generated by a homogeneous point process.

In each trial, we first generate a random N point distribution using homogeneous point process, and then run DBSCAN with the same input ($\epsilon, minPts$) to get the best or maximum cluster size $\hat{n}$ in the trial. After M trials, we will have M best $\hat{n}$ values from the trials. By sorting the M values in a descending order, we can estimate the p-value p_{null} of a cluster C detected from the real data by checking its rank r in the sorted list: $p_{null}(n_C, N, S, \epsilon, minPts) = r/M$. Note that M has to be at least $1/\alpha$ to evaluate the significance.

We reject the null hypothesis and mark cluster C as significant if $p_{null} < \alpha$ (or $r < \alpha M$). Equivalently, we just need to find the $(\alpha M)^{th}$ largest value in the sorted list and use that as a threshold (denoted as n_α) of cluster size to filter out non-significant clusters.

Algorithm 1: Monte Carlo estimation of n_α

```
Require:
    • total number of points N and spatial domain S
    • DBSCAN parameters ($\epsilon$, $minPts$)
    • significance level $\alpha$ and number of Monte Carlo trials M
1: assert($M \geq 1/\alpha$)
2: $nList$ = new List($M$)
3: for $i$ = 1 to $M$ do
4:     $data_r$ = getRandomPointDistribution($N$, $S$)
5:     $clusters$ = DBSCAN($data_r$, $\epsilon$, $minPts$)
6:     $nList(i)$ = max($clusters$.getSizes())
7: end for
8: $nlist$ = $nlist$.sort('DESC')
9: return $n_\alpha = nList(ceil(\alpha \cdot M))$
```

3.2 A Dual-Convergence Algorithm

The output of DBSCAN often contains multiple clusters with different sizes (i.e., number of points) and p-values. The baseline algorithm finds a threshold for cluster size n_α using the significance level α to classify the detections into significant ($n > n_\alpha$) and non-significant ($n \leq n_\alpha$) groups.

Finding the exact value of n_α in the baseline algorithm requires executing the exact DBSCAN algorithm across all M trials. Our idea is to reduce the number of exact DBSCAN runs in the Monte Carlo trials by generating bounds on the size n_{max} of the largest cluster (line 6, Alg. 1) in a simulated data (line 4, Alg. 1).

Since DBSCAN detections in real data may likely be a mixture of significant and non-significant clusters, an acceleration has to consider both cases to be truly effective. The proposed Dual-Convergence algorithm achieves this with: (1) an upper bound of n_{max} to accelerate the validation of significant clusters (Sec. 3.2.1); (2) a lower bound of n_{max} and an early-termination technique with a probabilistic performance guarantee to accelerate the filtering of non-significant clusters (Sec. 3.2.1 and 3.2.2); and (3) a dual-convergence framework which makes the above techniques work together to maximize the speed-up (Sec. 3.2.3).

3.2.1 *Upper and Lower Bounds of n_{max} with a Discrete Scan.*

To simplify the illustration, here we first consider the case of testing the significance of a single cluster with size n_C, which is detected from real-data. The general case will be detailed in Sec. 3.2.3.

In the baseline algorithm, in each round it finds the exact n_{max}. After M trials, if the total number of trials with $n_{max} \geq n_C$ is smaller than αM (equivalently $n_\alpha < n_C$), then n_C is significant; otherwise, non-significant.

Here our goal is to design an efficiently-computable upper bound $UB(n_{max})$ and lower bound $LB(n_{max})$ of n_{max} to avoid the need of exact DBSCAN if $n_C > UB(n_{max})$ or $n_C \leq LB(n_{max})$.

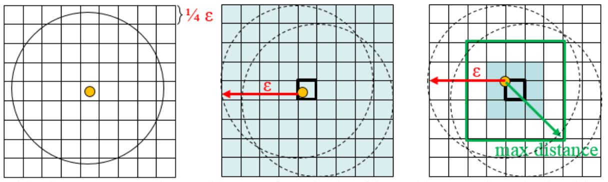
DBSCAN uses a circular ϵ-neighborhood to find core points and merge them into clusters through range queries. Denote ALG_{scan} as a more general version of DBSCAN, which may use other neighborhood definitions for finding core points and performing range queries for merging. Denote n'_{max} as the size of the largest cluster returned by ALG_{scan}. Lemmas 3.2 and 3.3 show the sufficient conditions (not necessary conditions) for an ALG_{scan} to make n'_{max} an upper or a lower bound of n_{max} from DBSCAN, respectively.

Lemma 3.2. $n'_{max} \geq n_{max}$ *if the neighborhoods defined in an* ALG_{scan} *(may vary from point to point) always fully* ***contain*** *the* ϵ *neighborhoods of DBSCAN as subspaces.*

Proof. The proof is straightforward. The core point set of ALG_{scan} must be a superset of that of DBSCAN, and any two core points merged by DBSCAN's range query must also be merged by ALG_{scan}'s because any ϵ-neighborhood of DBSCAN is always fully contained by the corresponding search neighborhood from ALG_{scan}. □

Lemma 3.3. $n'_{max} \leq n_{max}$ *if the neighborhoods defined in an* ALG_{scan} *(may vary from point to point) are always fully* ***contained by*** *the* ϵ*-neighborhoods of DBSCAN.*

Proof. The proof is symmetric to that of Lemma 3.2 by replacing supersets by subsets. Details skipped to avoid redundancy. □



(a) Discrete space and original ε neighborhood (b) Upper-bound sub-grid on ε neighborhood (blue) (c) Lower-bound sub-grid on ε neighborhood (blue)

Figure 2: Sub-grids of upper and lower bounds.

We use a discrete-scan to build and compute the upper and lower bounds. In the discrete scan, we discretize the continuous spatial domain S into a grid where the value of each grid cell is the number of points inside (no need to store the actual points, just a numeric value). The length of a grid cell is a fraction of the neighborhood size ϵ from DBSCAN (e.g., 1/4).

Fig. 2(b) illustrates the definition of the neighborhoods in the discrete-scan to construct the upper bounds (Lemma 3.2). Denote G as a grid with with I rows and J columns generated from the discretization, $G(i,j)$ as a cell at the i^{th} row and the j^{th} column (row and column IDs start with 1), and $G(i_0 : i_1, j_0 : j_1)$ as a sub-grid containing all the cells with row $i \in [i_0, i_1]$ and column $j \in [j_0, j_1]$. Thm. 3.4 shows the neighborhood definition in discrete-scan that satisfies the sufficient condition for upper-bounding.

Theorem 3.4. *For any point within $G(i,j)$, its circular ϵ neighborhood is always fully contained by the square neighborhood covered by $G(i-\Delta i : i+\Delta i', j-\Delta j : j+\Delta j')$, where $\Delta i = \min(\lceil \frac{\epsilon}{L} \rceil, i-1)$, $\Delta i' = \min(\lceil \frac{\epsilon}{L} \rceil, I-i)$, $\Delta j = \min(\lceil \frac{\epsilon}{L} \rceil, j-1)$, and $\Delta j' = \min(\lceil \frac{\epsilon}{L} \rceil, J-j)$.*

Proof. Every point in the original space must be contained by a grid cell (e.g., the center cell of Fig. 2(b)(a)). Note that in order to make sure there is strictly no overlaps between cells, each cell only contains its top and left boundaries, except for those at the right-most columns or bottom rows. As shown in Fig. 2(b), a point in a cell lies at most on or infinitely next to the boundaries of a cell. Thus, its ϵ search distance in the continuous space is at most $\lceil \frac{\epsilon}{L} \rceil$ cells in the discrete space. As a result, the circular ϵ neighborhood must be fully contained by a sub-grid whose length is $(2 \cdot \lceil \frac{\epsilon}{L} \rceil + 1)$ cells centered at $G(i,j)$. The min function is used to constrain the sub-grid to be within G. □

Similarly, Thm. 3.5 shows the neighborhood definition that satisfies the sufficient condition for lower bounding.

Theorem 3.5. *For any point within $G(i,j)$, its circular ϵ neighborhood is always fully contained by the square neighborhood covered by $G(i-\Delta i : i+\Delta i', j-\Delta j : j+\Delta j')$, where $L \leq \frac{\epsilon}{\sqrt{2}}$, $\Delta i = \min(\lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \rfloor, i-1)$, $\Delta i' = \min(\lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \rfloor, I-i)$, $\Delta j = \min(\lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \rfloor, j-1)$, and $\Delta j' = \min(\lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \rfloor, J-j)$.*

Proof. Here we need to guarantee that the sub-grid neighborhood is fully contained by the ϵ neighborhood of any point within $G(i,j)$. In other words, the furthest distance from a point in $G(i,j)$ to any location within the sub-grid neighborhood must be smaller than ϵ. As shown in Fig. 2(c), the maximum distance is achieved between a point in $G(i,j)$ located at its corner and another location at the sub-grid's furthest corner. This distance is at most $\frac{k-1}{2} \cdot \sqrt{2}L$, where k is an odd number representing the side length (unit: cell) of the sub-grid neighborhood. Since $(\frac{k-1}{2} + 1) \cdot \sqrt{2}L \leq \epsilon$, we have k to be at most $\lfloor \frac{\sqrt{2}\epsilon}{L} - 1 \rfloor$. Thus, the search distance outside $G(i,j)$ is at most $\frac{k-1}{2} = \lfloor \frac{\epsilon}{\sqrt{2}L} - 1 \rfloor$ cells. □

According to Theorems 3.4 and 3.5, we can construct the upper and lower bounds of n_{max} by using the grid-based neighborhoods. Note that the only difference in the neighborhood definitions for the upper and lower bounds lies in the values of Δi, $\Delta i'$, Δj and $\Delta j'$. Other than the side-lengths of the sub-grid based neighborhoods, the steps in the discrete-scan are exactly the **same** for the upper and lower bound calculation. Thus, in the following we will use $G(i-\Delta i : i+\Delta i', j-\Delta j : j+\Delta j')$ in a general manner (i.e., defined by either the upper or lower bound) to illustrate the key steps in the discrete-scan.

In the discrete-scan, we consider all the points within $G(i,j)$ as a single super-point whose (1) location is represented by the spatial extent of $G(i,j)$; (2) search neighborhood is covered by $G(i-\Delta i : i+\Delta i', j-\Delta j : j+\Delta j')$; and (3) cardinality is equal to the number of original points within $G(i,j)$. The cardinality will be used only when determining the core points and computing the sizes of the clusters. Again, the core points here will be a superset of the core points in DBSCAN.

We use super-points and grid-based neighborhoods to perform the discrete-scan. Since super-points can be represented by grid cells $\{G(i,j)\}$, we only need to enumerate through grid cells instead of the actual points after the grid is constructed. Then, for each $G(i,j)$, the discrete scan uses the sub-grid $G(i-\Delta i : i+\Delta i', j-\Delta j : j+\Delta j')$ (Thm. 3.4 and 3.5) as the search neighborhood to determine if $G(i,j)$ is a core super-point:

$$G(i,j) \text{ is } \begin{cases} \text{core}, & \text{if } \sum_{G(i,j)\in g} |G(i,j)| \geq minPts \\ \text{not core}, & \text{otherwise} \end{cases} \tag{1}$$

where $g = G(i-\Delta i : i+\Delta i', j-\Delta j : j+\Delta j')$ and $|G(i,j)|$ is the cardinality of the super-point.

The merging or expansion process in the discrete scan also uses the sub-grid based neighborhoods as illustrated in Alg. 2. Note that Alg. 2 describes the expansion for a single super-point $G(i,j)$, and its output n_C is the total number of points in the cluster containing $G(i,j)$. In line 11 of Alg. 2, the Hadamard Product ($cluster \circ G$) is the matrix multiplication performed in element-wise fashion. As a reminder, the value of a cell in G is the number of points in it.

After calculating the cluster sizes for all clusters, we can easily compute the bound on n_{max}. Note that we need to run the discrete-scan twice with different neighborhood definitions (i.e., Thm. 3.4 and 3.5) to compute both $UB(n_{max})$ and $LB(n_{max})$. Given the size of a cluster n_C from real data, we can avoid running the exact DBSCAN if $n_C > UB(n_{max})$ or $n_C \leq LB(n_{max})$.

3.2.2 *Early Termination with Theoretical Guarantee.*

Early termination has been widely used to stop Monte Carlo estimation with $M' < M$ trials. For example, if there are already (αM) trials with $n_{max} \geq n_C$, then the cluster of size n_C must not be significant and we can directly terminate it without performing the rest trials.

Algorithm 2: Expansion of a single $G(i, j)$ in a discrete-scan

```
Require:
    • Grid $G$ and a core super-point $G(i, j)$
    • Helper binary grid (visited: 1, non-visited: 0) $V$
    • DBSCAN's minimum number of points $minPts$
$queue$ = Queue().enqueue($G(i, j)$)
{# Mark the cells belonging to this cluster}
$cluster$ = new Grid(size=$G$.size; value=0)
while !empty($queue$) do
    $core$ = $queue$.dequeue()
    $g$ = getGridNeighborhood($G$, $core$)
    $cores_{new}$ = findNewCoresInNeighbor($core$, $g$, $minPts$, $V$)
    $queue$.enqueue($cores_{new}$)
    updateVisitedCells($V$, $g$)
    markClusterCells($cluster$, $g$, newValue=1)
end while
{# Get cluster size.}
$n_C$ = computeHadamardProduct($cluster$, $G$).sum()
return $n_C$, $V$
```

In related work, early termination is mostly used as a simple heuristic as its effectiveness is not well understood in theory.

In this work, early termination will be a key building block in the dual-convergence process (Sec. 3.2.3) to filter out non-significant clusters. Thus, we provide a detailed theoretical analysis of early termination and shows the probabilistic performance estimation of this technique for non-significant clusters.

Denote M as the number of Monte Carlo trials and α as the significance level ($M \geq 1/\alpha$). Denote n as the cluster size of a non-significant cluster and $p(n)$ as the probability of having $n_{max} \geq n$ in a random point distribution generated by a homogeneous point process. We first develop the following Lemma 3.6.

Lemma 3.6. *For cluster size n with $p(n)$, the probability of terminating within the x trials is:*

$$1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot [p(n)^i \cdot (1 - p(n)^{x-i})] \tag{2}$$

Proof. First, having exactly i trials with $n_{max} \geq n$ in x trials follows a binomial distribution, so we have its probability as $f_{bin}(i, x, p(n)) = \binom{x}{i} \cdot [p(n)^i \cdot (1 - p(n)^{x-i})]$. Then, the probability of terminating within x trials is equivalent to having at least αM trials with $n_{max} \geq n$ in x trials. Thus, this probability follows the cumulative binomial distribution. □

Lemma 3.6 is applied for a specific non-significant cluster size n and requires knowing $p(n)$. To make it useful in practice, we remove the $p(n)$ requirement by integrating Eq. (2) across all possible values of $p(n)$ and calculating the expectation:

Theorem 3.7. *For a non-significant cluster, the probability of terminating within the x trials is lower bounded by:*

$$P_{early}(\alpha M, x) = 1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot \frac{i!(x-i)!}{(x+1)!} \tag{3}$$

Proof. By integrating Eq. (2) across all $p(n)$ we have:

$$\begin{aligned} P_{early}(\alpha M, x) &= \int_0^1 \left\{ 1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot [p^i \cdot (1-p)^{x-i})] \right\} dp \\ &= 1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot Beta(i+1, x-i+1) \\ &= 1 - \sum_{i=0}^{\lceil \alpha M - 1 \rceil} \binom{x}{i} \cdot \frac{i!(x-i)!}{(x+1)!} \end{aligned}$$

where $Beta(i+1, x-i+1)$ is the Beta function with two inputs. □

Suppose we have $\alpha = 0.01$ and $M = 100$. The probability of terminating after the 2^{nd} trial is 0.67, 3^{rd} is 0.75 and 10^{th} is 0.91. Similarly, for $\alpha = 0.01$ and $M = 1,000$, then the probability of terminating after the 20^{th} trial is 0.52, 30^{th} is 0.68 and 100^{th} is 0.90. This shows that early-termination can be very effective.

3.2.3 *Dual Convergence towards the Significance Boundary.*

To summarize, the purpose of the upper bound is to accelerate the validation of significant clusters and the purpose of the lower bound and early-termination is to speed-up the filtering of non-significant ones.

Our previous discussion mainly concerned acceleration of significance testing for a single cluster size. In a real-world scenario, DBSCAN often returns a list of clusters with different sizes. Denote D as a list of the sizes of detected clusters from a real dataset. D is sorted in a descending order. There are three possible scenarios of cluster composition in D: (1) D contains only significant clusters; (2) D contains only non-significant clusters; and (3) D contains a mixture of significant and non-significant clusters. The first two scenarios are relatively easier because they can achieve acceleration from a single technique (Sections 3.2.1 and 3.2.2).

For the third scenario, the techniques need to share information and work together to really reduce computational cost. For example, the upper bound avoids having to run the exact DBSCAN if $n_C > UB(n_{max})$, where n_C is the size of a detected cluster from real data. Since we have multiple sizes in D, the condition becomes $\min(D) > UB(n_{max})$ in order to avoid the exact DBSCAN. When D contains non-significant clusters, this condition may be rarely satisfied, making the upper bound ineffective. Similarly, early termination is not expected to be effective when there is at least one significant cluster in D because this means we have to perform at least $(1 - \alpha)M$ trials (e.g., 99% of the trials for $\alpha = 0.01$).

To address this issue, we present a dual-convergence framework to coordinate the techniques and make them increasingly efficient as the trials progress.

Fig. 3 shows the progression of the trials and the information sharing among the techniques. Here early-termination controls a pointer et^* on D to mark the current boundary of non-significant clusters. Accordingly, when the upper bound operates in a trial, it can skip the exact DBSCAN if the size at et^* is greater than $UB(n_{max})$, i.e., $D(et^*) > n_{max}$ instead of $min(D) > n_{max}$. As more trials complete (e.g., 5% of M according to the analysis in Sec. 3.2.2), et^* will gradually cover all non-significant clusters and the

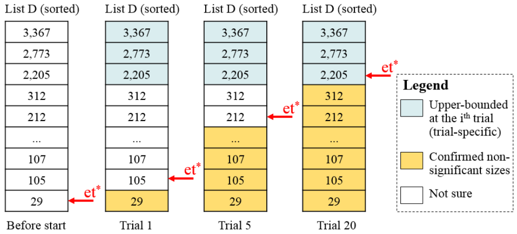


Figure 3: Progression along trials in dual convergence.

upper bound will approach its maximal effectiveness. Note that the upper-bounded sizes are trial-specific.

Algorithm 3 shows the execution order of the three techniques, i.e., upper bound, lower bound and early termination.

Algorithm 3: Dual convergence

```
Require:
    • List of cluster sizes $D$
    • Number of Monte Carlo trials $M$
    • Significant level $\alpha$ {#Other detailed inputs are skiped}
$et$ = $D$.length
$D_{geq}$ = new List(length=$D$.length) {# Tracking the number of trials with $n_{max} \geq D(i)$}
for $t$ = 1 to $M$ do
    $UB$ = getUB() {#upper bound}
    if $D(et) \leq UB$ then
        $unbounded\_id$ = getMinZeroID($D > UB$)
        $LB$ = getLB() {#lower bound}
        if $D(unbounded\_id) \geq LB$ then
            $n_{max}$ = exact_DBSCAN()
            $ids$ = getAllNonzeroID($D \leq n_{max}$)
            $D_{geq}(ids)$ += 1
        else
            $D_{geq}(unbounded\_id : end)$ +=1
        end if
        $et$ = getMaxNonzeroID($D_{geq} < \alpha M$){#early term.}
        if $et$==None, then break, end if
    end if
end for
return $D_{sig}$ = ($D_{geq} < \alpha M$)
```

3.2.4 Time complexity. Due to the space limit, we show the time complexity of the significant DBSCAN without detailed proofs. Denote N as total number of points, M as number of trials, and $|G|$ as number of grid cells in the discrete scan. The time complexity of the baseline algorithm is $O(f_{DB}(N) \times M) = O(MN \log N)$ in a 2D spatial space, where $f_{DB}(N)$ is the complexity of DBSCAN that can differ by data dimensions. This paper focuses on the spatial case. The time complexity of the Dual-Convergence algorithm is $O(\rho MN \log N + (1-\rho)M \cdot \max(|G|, N))$, where $\rho \in (0, 1]$ is the portion of trials that requires the exact DBSCAN. Note that in a 2D spatial case, as N increases we normally have $|G| << N$. The discrete scan still requires at least linear time because we need to count the number of points in each grid cell. This needs a single and simple pass through the actual points (the cell a point belongs to is trivially computable in closed-form). After that, it is just a quick linear scan through the cells (i.e., $O(|G|)$ time), which no longer involves any computation with the actual points or other special data structures (can be used with integral image for acceleration). In experiments (Sec. 4.2) we will see that the discrete scan works very well (overhead is almost negligible). For non-clustered data, we also expect the early termination to be very effective according to the probabilistic analysis (Sec. 3.2.2) and experiment results (Sec. 4.2). In a high-dimensional space, the overall computation is still challenging because grid-based methods normally have time complexity exponential to the variable dimension d (unless the complexity treats dimension d as a constant no matter how large it is, e.g., $d = 100$). Our current scope is the 2D spatial case.

3.3 Significant Clusters with Varying Densities

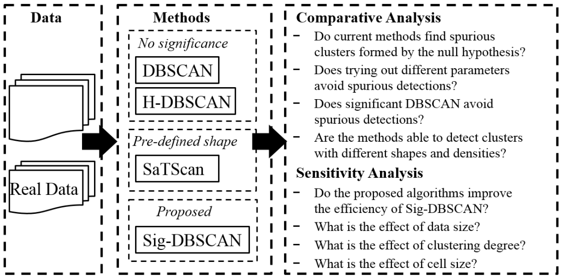
In this section we discuss a heuristic search strategy which uses significant DBSCAN for a single $(\epsilon, minPts)$ as a subroutine to detect clusters of different densities in the context of significance testing. The **main focus** here is to illustrate the benefits of significance testing and the way to perform it when considering various densities. The scope of this discussion is **not** to further mature related work in terms of detecting clusters of varying densities. Nonetheless, the comparisons to related work are shown through experiments.

In DBSCAN, in order to detect clusters of varying densities we need to consider a range of *eps* and *minPts* rather than a fixed pair. A key issue that rises when we aggregate DBSCAN results with multiple pairs of $(\epsilon, minPts)$ is that it may greatly increase the number of spurious detections (i.e., chance patterns from different $(\epsilon, minPts)$). This is where significance testing becomes more important by playing a key role in eliminating the spurious patterns.

In the following, we describe significance testing in this scenario with an example heuristic search strategy for multi-densities.

3.3.1 An Example Heuristic Search Strategy. The search strategy basically enumerates through a list of $(\epsilon, minPts)$ for various densities (i.e., $minPts/(\pi\epsilon^2)$). Since density calculation requires both the number of points and area, considering only a single scale (i.e., one ϵ) for each density may not be appropriate. To mitigate this, the search strategy additionally considers a range of ϵ for each density.

Denote U as a matrix where each row corresponds to a single density and each column represents a specific scale (i.e., ϵ). The value of each entry $U(i, j)$ is the $minPts$ value (rounded) of $\epsilon(j)$ for $density(i)$. Lower row IDs in U represent smaller densities and lower column IDs represent smaller scales. Starting at the highest density, the heuristic search follows two key steps in each search-round. **Step-1:** It fixes the density at the current row i and sequentially executes original DBSCAN on the same data but through different scales from low to high. It stops at the scale $U(i, j)$ when the increase in average cluster size from $U(i, j)$ to $U(i, j + 1)$ is smaller than $\lambda\%$ (e.g., 10%); **Step-2:** It fixes the scale at the j^{th} column (from step-1) and sequentially executes original DBSCAN towards lower densities. It stops at $U(i', j)$ when the average cluster size from $U(i', j)$ to $U(i' + 1, j)$ is smaller than $\lambda\%$ (e.g., 10%). The DBSCAN result at $U(i', j)$ is returned for this search-round (corresponding to a density-level). This two-step process is used to avoid inappropriate densities (e.g., too high) or scales (e.g., too small) which may cause a single cluster to be shattered into pieces.



Comparative Analysis

- Do current methods find spurious clusters formed by the null hypothesis?
- Does trying out different parameters avoid spurious detections?
- Does significant DBSCAN avoid spurious detections?
- Are the methods able to detect clusters with different shapes and densities?

Sensitivity Analysis

- Do the proposed algorithms improve the efficiency of Sig-DBSCAN?
- What is the effect of data size?
- What is the effect of clustering degree?
- What is the effect of cell size?

Figure 4: Overall validation framework.

Using this example two-step (per round) search, we show how significance testing should be performed in the next section.

3.3.2 Significance testing. Significance testing is performed after each round on the returned result. To be consistent, the significance testing uses the same $(\epsilon, minPts)$ parameter pair used for the returned result. Note that here we only have a single pair of parameters and can directly perform the significance testing described in Sec. 3.2. In addition, the search strategy is only needed for real data and not the significance testing (single $(\epsilon, minPts)$ based).

After finding significant clusters, a **critical step** is to remove them from the data before the next round of search. Since the next thing we need to test is whether the rest of the data is generated by a homogeneous point process or it contains other significant clusters (e.g., those with lower densities), the analysis should be independent from any already-confirmed significant clusters. Thus, we need to remove both the points of the significant clusters and their spatial coverage. This will result in a smaller amount of points and a smaller spatial domain for future significance testing. The challenge in this step is the removal of the spatial coverage of significant clusters. Since the exact coverage of the clusters (point sets) is not defined in DBSCAN, we rasterize the continuous spatial domain into pixels and approximate the coverage of the clusters by marking the pixels within the ϵ neighborhoods of the core points. In subsequent significance testing, the points will only be randomly distributed to the unmarked sub-spaces.

After the removal, the heuristic search continues with the next (lower) density.

4 VALIDATION

Fig. 4 shows the overall validation framework and questions.

4.1 Comparative Analysis

The candidate methods include DBSCAN with varying parameters, H-DBSCAN with varying parameters, SaTScan, and the proposed significant DBSCAN with significance level set to 0.01. Note that the significant DBSCAN used the example heuristic search strategy described in Sec. 3.3, and any parameters associated with it remain the same throughout the experiments for data with different number of points, cluster shapes, cluster densities, point processes, etc.

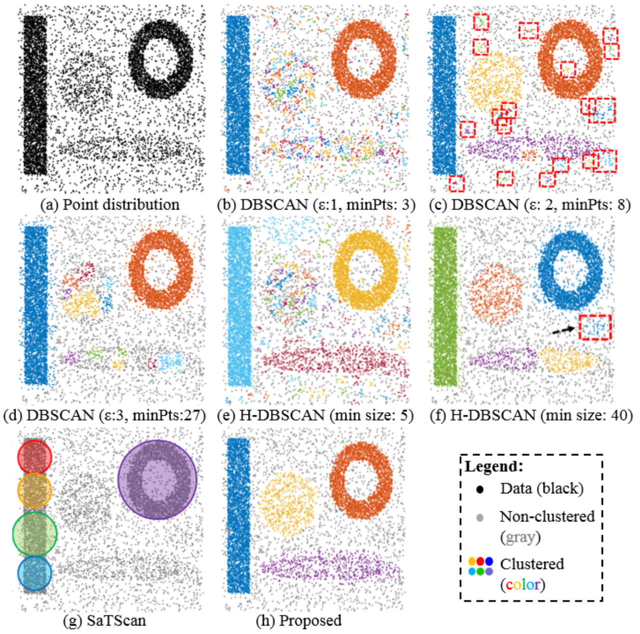
4.1.1 Performance on Data Generated by a Homogeneous Point Process. Fig. 1 (2,000 points, in Sec. 2.2) shows the results of the candidates methods on two datasets generated by a homogeneous point process (i.e., no true clusters). The results of DBSCAN were evaluated on three sets of parameters $(\epsilon, minPts)$. Note that we could have possibly used parameters corresponding to very high densities and made sure there would be no clusters detected. However, this choice is not appropriate for two main reasons: (1) we cannot be sure whether or not a given data is truly clustered (i.e., it could possibly contain real clusters) without testing so it is not proper to assume it beforehand; and (2) the same parameters were tried on clustered data with 2,000 and 10,000 points in the same spatial domain in Sec. 4.1.2. We can see that some of these parameters already represented a density that is too high to correctly detect true clusters (e.g., Fig. 5(b)(d) and Fig. 6(b)(c)). Thus, the choice of parameters already covered sufficiently high densities.

H-DBSCAN [6] is considered as an improved cluster detection method that combines the strengths of DBSCAN and OPTICS[4] (made by the authors of the papers). By default, H-DBSCAN does not require input parameters and can automatically find the best clusters based on its criteria. If needed, a minimum cluster size can be set to refine the criteria. A common value for the minimum cluster size is 5 (e.g., defaulted in its standard Python library [2]). In our experiments, we tried both 5 and 40.

The results of both DBSCAN and H-DBSCAN contain spurious patterns for different parameters. For DBSCAN, in general the spurious patterns were smaller when the $(\epsilon, minPts)$ corresponds to a higher density (also affected by the scale). Some detections are quite large in cluster size so it is difficult to remove them with a preset threshold of minimum cluster size. We can see a similar trend for H-DBSCAN results. When a larger minimum cluster size is used, the detected spurious clusters also becomes larger, making them difficult to remove without significance testing. This also mirrors our earlier analysis in Sec. 2.2. In Fig. 1(g)(h) we can see that both SaTScan and the proposed significant DBSCAN can successfully avoid chance patterns with significance testing.

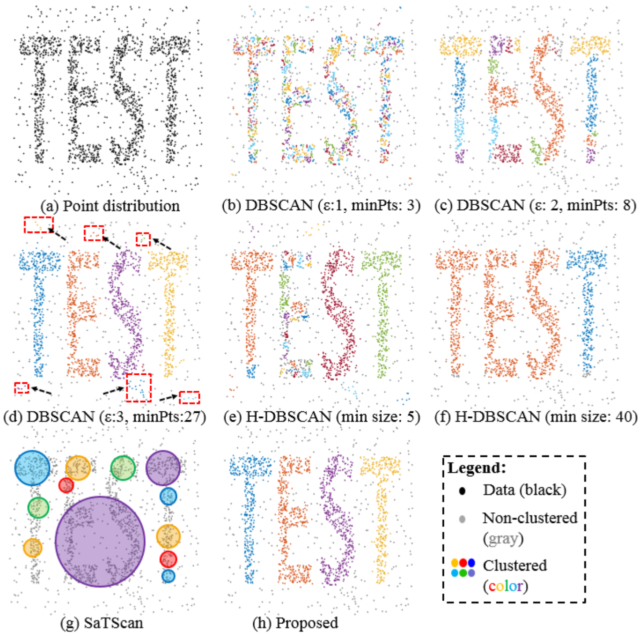
4.1.2 Performance on Data with Clusters of Varying Shapes. Fig. 5 (10,000 points) and Fig. 6 (2,000 points) show the results on two datasets generated by biased/clustered point processes. Each has four clusters of different shapes. As we can see, SaTScan is overall robust against the noises with significance testing, but it is limited in detecting clusters of general shapes. As a spatial scan statistic approach, SaTScan operates in a top-down fashion. In other words, it enumerates candidate regions of different sizes with a pre-defined geometric shape and checks their statistical significance. In addition, the enumeration space for a single shape is often already quite-large (e.g., circles of different sizes across all locations) and results in large computational cost. In contrast, DBSCAN operates in a bottom-up fashion, which allows it to trace arbitrary shapes automatically without predefined shapes. Note that it does assume the density within a cluster is generally homogeneous without drastic changes.

As we can see, in general DBSCAN based methods show stronger capacity than SaTScan in finding clusters with varying shapes, although some of their results still contain many spurious patterns due to the lack of significance testing. In addition, H-DBSCAN methods also seem to be sensitive to the minimum cluster size threshold. For example, in Fig. 6(f), the first three clusters were detected as a single one. In Fig. 5(h) and Fig. 6(h), the proposed significant DBSCAN is able to detect the significant clusters of arbitrary shapes while successfully avoiding chance patterns.



(a) Point distribution (b) DBSCAN (ε:1, minPts: 3) (c) DBSCAN (ε: 2, minPts: 8)
(d) DBSCAN (ε:3, minPts:27) (e) H-DBSCAN (min size: 5) (f) H-DBSCAN (min size: 40)
(g) SaTScan (h) Proposed

Figure 5: Clustered 10000-point shape data.

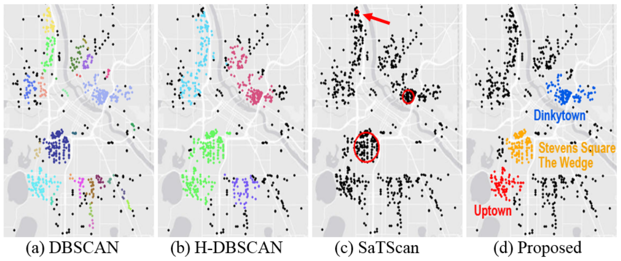


(a) Point distribution (b) DBSCAN (ε:1, minPts: 3) (c) DBSCAN (ε: 2, minPts: 8)
(d) DBSCAN (ε:3, minPts:27) (e) H-DBSCAN (min size: 5) (f) H-DBSCAN (min size: 40)
(g) SaTScan (h) Proposed

Figure 6: Clustered 2000-point test data.

4.1.3 Performance on Data with Clusters of Varying Densities. The four clusters in Fig. 5 have different densities. The probability density of having a point in the rectangle and ring is twice as high as that in the circle and ellipse.

As we can see, DBSCAN methods can detect true clusters of different densities by varying its parameters. However, the number of chances patterns also greatly increases if we merge their results. The same trend can be seen in the results of H-DBSCAN. With significance testing, the proposed sig-DBSCAN method was able to filter out the chance patterns and keep only the significant clusters of varying densities (i.e., with the heuristic search in Sec. 3.3). Again, any parameter associated with the heuristic search remained the same with no change throughout the experiments on different data.



(a) DBSCAN (b) H-DBSCAN (c) SaTScan (d) Proposed

Figure 7: A real-world example: snow emergency tows.

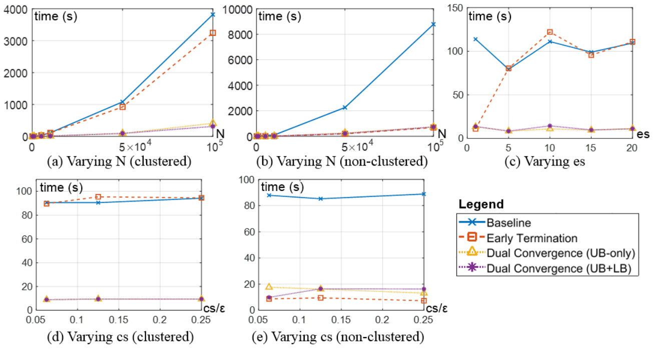
4.1.4 Real-world Example: Minneapolis Snow Emergency Tows. Fig. 7 shows the results of the candidate methods on an example real-world data of snow-emergence vehicle tows (948 points) in Minneapolis, US, 2019. The city is located at the north side of the US near the Lake Superior, and receives heavy snows in the winter. To plow the thick snow from the streets (i.e., curb to curb), a snow emergency requires all cars parked in the declared zones to be moved to other places; otherwise they will be tagged or towed.

Given that there are limited tow-trucks and time for towing, towing priority (e.g., more tow trucks, higher frequency) is given to neighborhoods that have narrower streets, which will be very difficult to use without complete snow shoveling. According to a transportation officer in Minneapolis (2019)[3], the neighborhoods with the most pressing needs in the snow emergency were: **Stevens Square, the Wedge**, **Dinkytown** and **Uptown**. We used this real-world data and information to test if the methods can identify these priority neighborhoods in the snow emergency.

Note that the invalid regions (e.g., rivers, lakes, parks, urban forests, highway etc.) in the figure were excluded from the input spatial domain before running these methods. The study area used for the snow emergency data was approximated by the general coverage of the points. In the experiment results, we can see that the four neighborhoods are reasonably well captured by the significant DBSCAN (Fig. 7(d)). The clusters that contain the neighborhoods (sometimes together with a few adjacent ones) used the same colors as the text. In the DBSCAN result (Fig. 7(b)), we can still see these significant clusters, but there are many non-significant ones covering most of the points. In H-DBSCAN, the clusters are more contiguous with some being merged together but non-significant ones still persist in most of the study area. SaTScan (Fig. 7(c)) was able to find roughly two of the neighborhoods. It missed the one at Uptown, potentially due to the shape of the clusters (i.e., a large empty space towards the right-side when a circle is used to cover it). In the other two large clusters it identified, there is not much empty space left in the circles. In addition, SaTScan detected a very tiny cluster near the top-left, which is also a well-known issue (i.e., tiny clusters tend to have a very high likelihood ratio).

4.2 Sensitivity Analysis

We evaluated the computational performance of the baseline and the Dual-Convergence algorithms on various data sizes N, clustering degrees (or effect sizes) es and cell-sizes cs in discrete scan. For



(a) Varying N (clustered)

(b) Varying N (non-clustered)

(c) Varying es

(d) Varying cs (clustered)

(e) Varying cs (non-clustered)

Figure 8: Execution time analysis.

clustered data, the clustered regions are the same as those in Fig. 5. To add some density variations in the clustered data, the probability density of having points (modeled by es) in the rectangle and ring was set twice as high as that in the circle and ellipse. The default parameter values used were $N = 10,000$, $es = (10, 20)$ and $cs = \epsilon/4$.

4.2.1 Effect of Data Size. We evaluated the execution on varying data sizes N on both clustered (Fig. 8(a)) and non-clustered data (Fig. 8(b)). For both data types, the general trend is that the proposed Dual-Convergence method can greatly reduce the computational cost and the savings becomes greater as N increases. For early-termination, it is not very effective by itself when data is clustered because at least $(1 - \alpha)M$ trials are needed. Comparing the upper-bound and lower-bound methods, we can see that the execution time savings are mostly achieved by the upper-bound alone while not much more is contributed by the lower bound. The reason might be that the lower bound is not as tight as the upper bound with the cell size used (Fig. 2 (b) and (c)).

4.2.2 Effect of Clustering Degree. Clustering degree (or effective size) es shows how much more likely an individual location within a clustered region will have a point compared to a location outside. Note that if the spatial coverage of a cluster is too small, even a large effect size can be hard to observe and confirm due to the small cumulative probability. We used two different es for each clustered data: two clusters will have es_0 and the other two will have $es_1 = 2 \cdot es_0$. The X-axis in Fig. 8(c) shows es_0. Note that for non-clustered data all the locations have the same $es = 1$.

The trend is that the Dual-Convergence algorithm consistently achieved great time reduction compared to the baseline throughout the experiment. The sharp increase of early-termination shows the switch from non-clustered $es = 1$ to clustered data $es > 1$.

4.2.3 Effect of Cell Size in Discrete Scan. Fig. 8(d) and (e) show the execution time under varying cell sizes cs in the discrete scan. Note that cs we used is a fraction of ϵ. Our expectation is that a finer cs can tighten the upper and lower bounds at the cost of increased number of cells in the discretization. We can see that overall the cost associated with the increase of number of cells is secondary (i.e., very small overhead) in terms of computation. In addition, there was a small improvement of lower bound from $cs/\epsilon = 1/8$ to 1/16 while the improvement in the upper bound is not obvious. The reason could be that the upper bound is much tighter than the lower bound so it is less affected within this range of cs variation, whereas the lower bound can benefit a lot more with a smaller cs.

5 CONCLUSIONS AND FUTURE WORK

We introduced, discussed and proposed a framework for incorporating statistical significance in DBSCAN clustering. To perform the significance testing, we proposed a baseline Monte-Carlo method as well as a Dual-Convergence algorithm for acceleration. In addition, we discussed cluster detection of varying densities (i.e., multiple $(\epsilon, minPts)$) in the context of significance testing. Our experiment results show that the proposed significant DBSCAN can greatly improve solution quality by robustly eliminating chance patterns. The Dual-Convergence algorithm also can greatly improve the computational efficiency.

Future work: This work is just a start of improving the robustness of widely-used clustering methods by modeling statistical significance. We expect the results to encourage more studies to explore significance testing of other methods (e.g., k-means). Also, many other opportunities exist to further improve this work, including different modeling strategies (e.g., advanced test statistics, hypotheses, underlying population), different DBSCAN methods (e.g., H-DBSCAN), higher dimensions beyond spatial, better strategies for multi-density search, analysis of significance and clustering quality (e.g., scale invariance, consistency, richness) [5], computational enhancements (e.g., distributed computing), etc.

ACKNOWLEDGMENTS

This work is supported by the US NSF under Grants No. 1737633, 1029711, IIS-1320580, 0940818 and IIS-1218168, the USDOD under Grants HM0210-13-1-0005, ARPA-E under Grant No. DE-AR0000795, USDA under Grant No. 2017-51181-27222, NIH under Grant No. UL1 TR002494, KL2 TR002492 and TL1 TR002493 and the OVPR U-Spatial and MSI at the U. of Minnesota. We also thank Dr. Hans-Peter Kriegel for his encouragement on carrying out this research.

REFERENCES

[1] 2017. National Cancer Institute. https://surveillance.cancer.gov/satscan/.
[2] 2019. HDBSCAN. https://hdbscan.readthedocs.io/en/latest/api.html.
[3] 2019. Minneapolis sets winter parking restrictions until April 1. http://www.startribune.com/minneapolis-sets-winter-parking-restrictions-until-april-1/506388832/.
[4] Mihael Ankerst et al. 1999. OPTICS: ordering points to identify the clustering structure. In *ACM Sigmod record*, Vol. 28. ACM, 49–60.
[5] Shai Ben-David et al. 2009. Measures of clustering quality: A working set of axioms for clustering. In *Adv. in neural information processing systems*. 121–128.
[6] Ricardo Campello et al. 2015. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10, 1 (2015), 5.
[7] Emre Eftelioglu et al. 2014. Ring-shaped hotspot detection: a summary of results. In *Data Mining (ICDM), 2014 IEEE Intl. Conf. on*. IEEE, 815–820.
[8] Martin Ester et al. 1996. A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Intl. Conf. on Knowledge Discovery and Data Mining (KDD'96)*. 6.
[9] Martin Kulldorff. 1997. A spatial scan statistic. *Comm. in Statistics-Theory and methods* 26, 6 (1997), 1481–1496.
[10] Daniel B Neill. 2006. Detection of spatial and spatio-temporal clusters. In *Tech Rep CMU-CS-06-142, PhD thesis*. Carnegie Mellon University.
[11] Daniel B Neill et al. 2004. Rapid detection of significant spatial clusters. In *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*. 256–265.
[12] Yiqun Xie et al. 2017. Transdisciplinary Foundations of Geospatial Data Science. *ISPRS International Journal of Geo-Information* 6, 12 (2017), 395.
[13] Yiqun Xie et al. 2019. A nondeterministic normalization based scan statistic (NN-scan) towards robust hotspot detection: a summary of results. In *SIAM Intl. Conf. on Data Mining (SDM'19)*.