

PARALLELISATION OF DBSCAN ALGORITHM : AN ALGORITHM FOR DATA CLUSTERING

NAME : ASHISH CHOUDHARY

ROLL NUMBER : CED18I061

COURSE : HIGH PERFORMANCE COMPUTING

COURSE INSTRUCTOR : DR NOOR MAHAMMAD

OVERVIEW:

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996. It is a density-based clustering non-parametric algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN is one of the most common clustering algorithms and also most cited in scientific literature.

Unlike the most well known K-mean, DBSCAN does not need to specify the number of clusters. It can automatically detect the number of clusters based on your input data and parameters. More importantly, DBSCAN can find arbitrary shape clusters that k-means are not able to find. For example, a cluster surrounded by a different cluster.

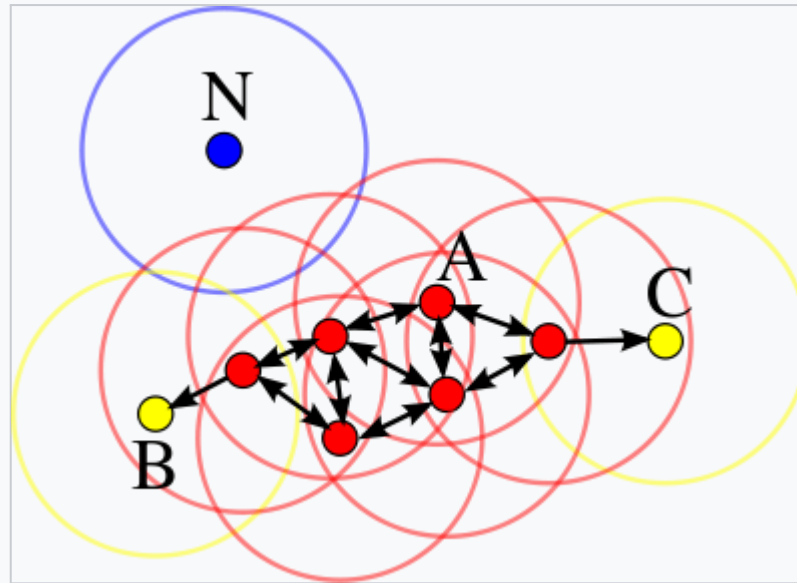


In 2014, the algorithm was awarded the test of time award (an award given to algorithms which have received substantial attention in theory and practice) at the leading data mining conference, ACM SIGKDD. As of July 2020, the follow-up paper "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN" appears in the list of the 8 most downloaded articles of the prestigious ACM Transactions on Database Systems (TODS) journal.

Consider a set of points in some space to be clustered. Let ε be a parameter specifying the radius of a neighborhood with respect to some point. For the purpose of DBSCAN clustering, the points are classified as *core points*, (*density-*) *reachable points* and *outliers*, as follows:

- A point p is a *core point* if at least minPts points are within distance ε of it (including p).
- A point q is *directly reachable* from p if point q is within distance ε from core point p . Points are only said to be directly reachable from core points.
- A point q is *reachable* from p if there is a path p_1, \dots, p_n with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i . Note that this implies that the initial point and all points on the path must be core points, with the possible exception of q .
- All points not reachable from any other point are *outliers* or *noise points*.

Now if p is a core point, then it forms a *cluster* together with all points (core or non-core) that are reachable from it. Each cluster contains at least one core point; non-core points can be part of a cluster, but they form its "edge", since they cannot be used to reach more points.



In this diagram, $\text{minPts} = 4$. Point A and the other red points are core points, because the area surrounding these points in an ϵ radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable.

Reachability is not a symmetric relation: by definition, only core points can reach non-core points. The opposite is not true, so a non-core point may be reachable, but nothing can be reached from it. Therefore, a further notion of *connectedness* is needed to formally define the extent of the clusters found by DBSCAN. Two points p and q are density-connected if there is a point o such that both p and q are reachable from o . Density-connectedness *is* symmetric.

A cluster then satisfies two properties:

1. All points within the cluster are mutually density-connected.
2. If a point is density-reachable from some point of the cluster, it is part of the cluster as well.

Algorithm

Original query-based algorithm

DBSCAN requires two parameters: ϵ (eps) and the minimum number of points required to form a dense region (minPts). It starts with an arbitrary starting point that has not been visited. This point's ϵ -neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise. Note that this point might later be found in a sufficiently sized ϵ -environment of a different point and hence be made part of a cluster.

If a point is found to be a dense part of a cluster, its ϵ -neighborhood is also part of that cluster. Hence, all points that are found within the ϵ -neighborhood are added, as is their own ϵ -neighborhood when they are also dense. This process continues until the density-connected cluster is completely found. Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

DBSCAN will also require a distance function (as well as similarity functions or other predicates). The distance function (dist) can therefore be seen as an additional parameter.

The algorithm can be expressed in pseudocode as follows:

```
DBSCAN(DB, distFunc, eps, minPts) {  
    C := 0                                /* Cluster counter */  
    for each point P in database DB {  
        if label(P) ≠ undefined then continue /* Previously processed in inner loop */  
        Neighbors N := RangeQuery(DB, distFunc, P, eps) /* Find neighbors */  
        if |N| < minPts then {              /* Density check */  
            label(P) := Noise                /* Label as Noise */  
            continue  
        }  
        C := C + 1                          /* next cluster label */  
        label(P) := C                       /* Label initial point */  
        SeedSet S := N \ {P}                /* Neighbors to expand */  
        for each point Q in S {             /* Process every seed point Q */  
            if label(Q) = Noise then label(Q) := C /* Change Noise to border point */  
            if label(Q) ≠ undefined then continue /* Previously processed (e.g., border point) */  
            label(Q) := C                   /* Label neighbor */  
            Neighbors N := RangeQuery(DB, distFunc, Q, eps) /* Find neighbors */  
            if |N| ≥ minPts then {           /* Density check (if Q is a core point) */  
                S := S ∪ N                  /* Add new neighbors to seed set */  
            }  
        }  
    }  
}
```

where RangeQuery can be implemented using a database index for better performance, or using a slow linear scan:

```
RangeQuery(DB, distFunc, Q, eps) {  
  Neighbors N := empty list  
  for each point P in database DB {  
    if distFunc(Q, P) ≤ eps then {  
      N := N ∪ {P}  
    }  
  }  
  return N  
}
```

/ Scan all points in the database */*
/ Compute distance and check epsilon */*
/ Add to result */*

Abstract algorithm

The DBSCAN algorithm can be abstracted into the following steps:

1. Find the points in the ϵ (eps) neighborhood of every point, and identify the core points with more than minPts neighbors.
2. Find the connected components of *core* points on the neighbor graph, ignoring all non-core points.
3. Assign each non-core point to a nearby cluster if the cluster is an ϵ (eps) neighbor, otherwise assign it to noise.

A naive implementation of this requires storing the neighborhoods in step 1, thus requiring substantial memory. The original DBSCAN algorithm does not require this by performing these steps for one point at a time.



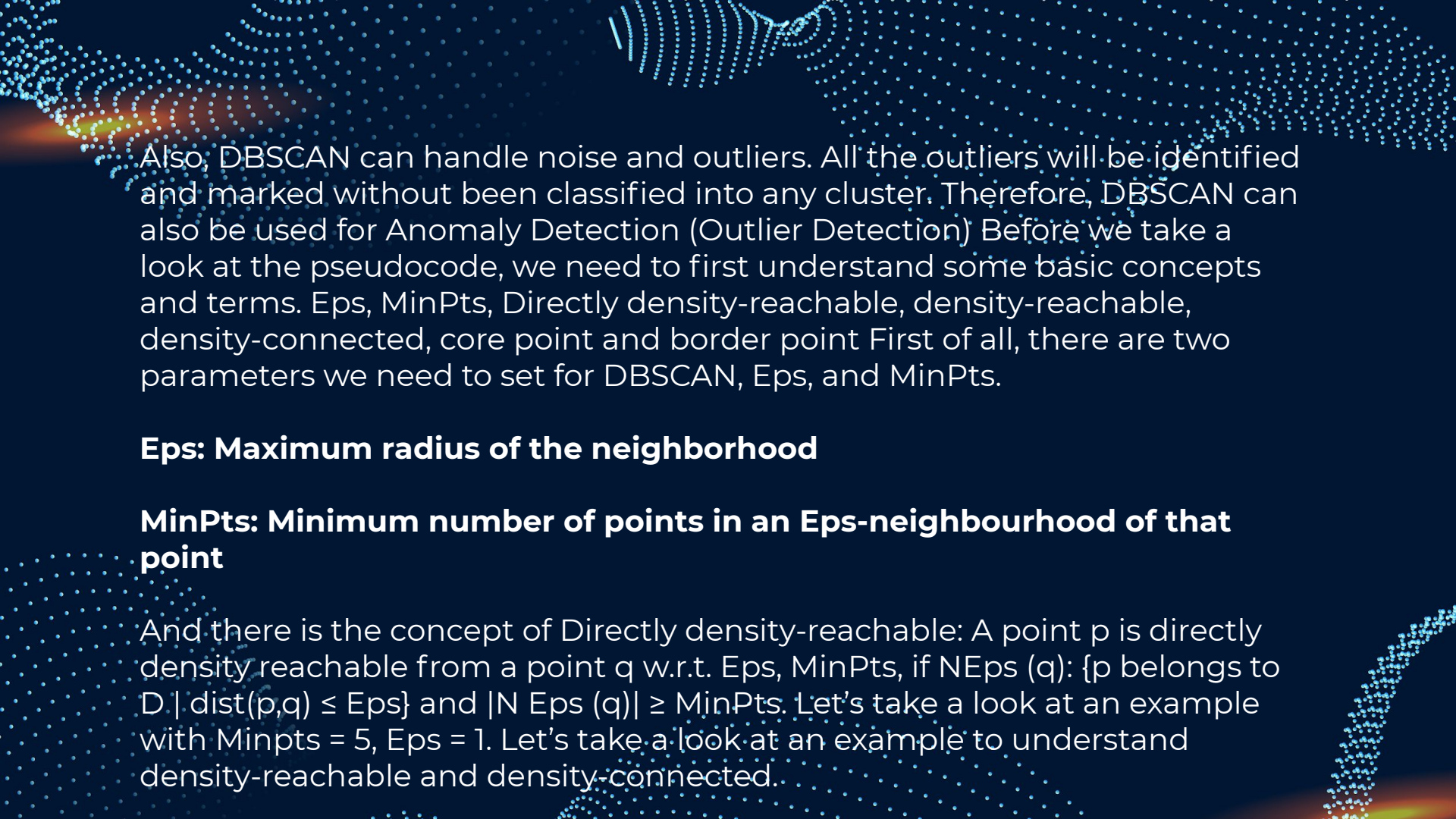
PARALLELISATION OF DBSCAN ALGORITHM : AN ALGORITHM FOR DATA CLUSTERING

**ASHISH CHOUDHARY
CED18I061**

What is DBSCAN?

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a commonly used unsupervised clustering algorithm proposed in 1996. Unlike the most well known K-means, DBSCAN does not need to specify the number of clusters. It can automatically detect the number of clusters based on your input data and parameters. More importantly, DBSCAN can find arbitrary shape clusters that k-means are not able to find. For example, a cluster surrounded by a different cluster.





Also, DBSCAN can handle noise and outliers. All the outliers will be identified and marked without been classified into any cluster. Therefore, DBSCAN can also be used for Anomaly Detection (Outlier Detection) Before we take a look at the pseudocode, we need to first understand some basic concepts and terms. Eps, MinPts, Directly density-reachable, density-reachable, density-connected, core point and border point First of all, there are two parameters we need to set for DBSCAN, Eps, and MinPts.

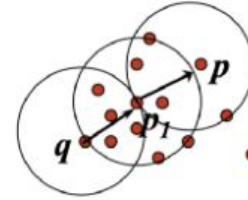
Eps: Maximum radius of the neighborhood

MinPts: Minimum number of points in an Eps-neighbourhood of that point

And there is the concept of Directly density-reachable: A point p is directly density reachable from a point q w.r.t. Eps, MinPts, if $N_{Eps}(q) = \{p \text{ belongs to } D \mid \text{dist}(p,q) \leq Eps\}$ and $|N_{Eps}(q)| \geq MinPts$. Let's take a look at an example with Minpts = 5, Eps = 1. Let's take a look at an example to understand density-reachable and density-connected.

- Density-reachable:

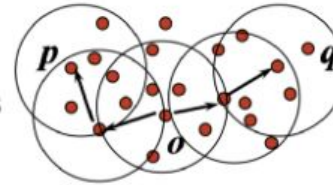
- A point p is **density-reachable** from a point q w.r.t. Eps , $MinPts$ if there is a chain of points p_1, \dots, p_n , $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i



Density-reachable example

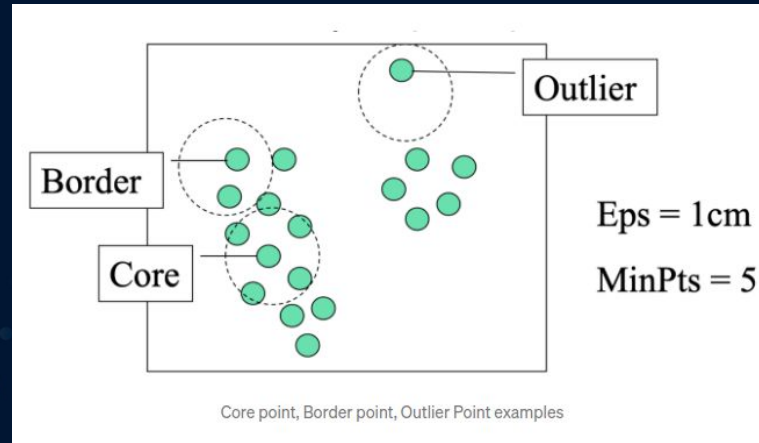
- Density-connected

- A point p is **density-connected** to a point q w.r.t. Eps , $MinPts$ if there is a point o such that both, p and q are density-reachable from o w.r.t. Eps and $MinPts$



Density-connected example

Finally, a point is a core point if it has more than a specified number of points (MinPts) within Eps. These are points that are at the interior of a cluster A. And a border point has fewer than MinPts within Eps, but is in the neighborhood of a core point. We can also define the outlier(noise) point, which is the points that are neither core nor border points.



DBSCAN algorithm

1. Arbitrary select a point p
2. Retrieve all points density-reachable from p based on Eps and $MinPts$
3. If p is a core point, a cluster is formed
4. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database
5. Continue the process until all of the points have been processed

If a spatial index is used, the computational complexity of DBSCAN is $O(n \log n)$, where n is the number of database objects. Otherwise, the complexity is $O(n^2)$

OBJECTIVE

To take more advantage of multi-core architecture of the parallel computing system and Disjoint Set Union to reduce the computing time of this algorithm for a graph with large dataset

- (i) shared memory paradigm
- (ii) distributed memory paradigm



THANK YOU