# OPTICS

M.M. Pedram
pedram@khu.ac.ir
Kharazmi University

1

## OPTICS: A Cluster-Ordering Method (1999)

- **OPTICS**: **O**rdering **P**oints **T**o **I**dentify the **C**lustering **S**tructure
  - Ankerst, Breunig, Kriegel, and Sander (SIGMOD'99)
  - Produces a special order of the database w.r.t. its density-based clustering structure,
  - This cluster-ordering contains info equivalent to the density-based clusterings corresponding to a broad range of parameter settings,
  - Cluster ordering can be used to extract basic clustering information,
  - Good for both automatic and interactive cluster analysis, including finding intrinsic clustering structure,
  - Can be represented graphically or using visualization techniques,



2

## OPTICS

- It addresses one of DBSCAN's major weaknesses: the problem of detecting meaningful clusters in data of varying density.

- The similarity between OPTICS and DBSCAN:
  - two parameters are required, i.e.: ε and *MinPts*.
  - A point *p* is a *core point* if at least *MinPts* points are found within its ε-neighborhood.

- The Difference between OPTICS and DBSCAN:
  - Contrary to DBSCAN, OPTICS also considers points that are part of a more densely packed cluster, so each point is assigned a core distance that basically describes the distance to its *MinPts*-th point.
  - The *reachability-distance* of a point *p* from another point *r* is the distance between *p* and *o*, or the core distance of *o*.



## OPTICS - Main idea

- In DBSCAN, for constant *MinPts*, clusters with high density (lower ε) are completely contained in **density-connected** sets obtained with lower density.
- in order to produce a set or ordering of density-based clusters, DBSCAN is extended to process a set of distance parameter ε at the same time.
- in order to produce a set or ordering of density-based clusters, the objects need to be processed in a specific order.
- This order selects an object that is density reachable w.r.t. lowest ε so that clusters of higher density (lower ε) will be finished first.

## OPTICS - Main idea

- Based on this idea, 2 values need to be stored for each object:
  - *Core distance*
  - *Reachability distance*



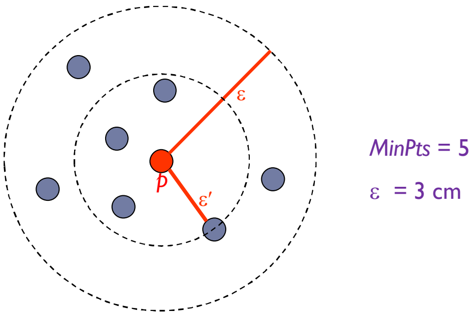


## OPTICS - Main idea

- *Core distance*: smallest radius (ε) that makes it a core object. If $p$ is not core, it is undefined.

$$\text{core-distance}_{\varepsilon, MinPts}(p) = \begin{cases} \text{distance to the } (MinPts\text{-}1)th\ NN & \text{otherwise} \\ \text{undefined} & \text{if } |N_\varepsilon(p)| < MinPts \end{cases}$$

  - Core Distance of $p$ or $\varepsilon'$ : distance between $p$ and its 4-thNN.



## OPTICS - Main idea

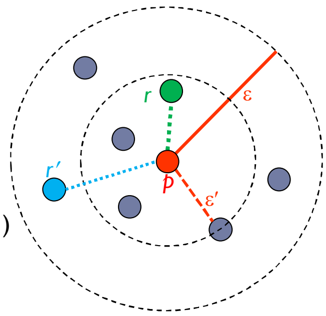
- *Reachability distance* of *r* w.r.t. *p* is the greater value of the core distance of *p* and the Euclidean distance between *p* & *r*. If *p* is not a core object, distance reachability between *p* & *q* is undefined.

$$reachability\text{-}distance_{\varepsilon,MinPts}(p,r) = \begin{cases} \max\left(core\text{-}distance_{\varepsilon,MinPts}(p), dist(p,r)\right) & otherwise \\ undefined & if\ |N_{\varepsilon}(p)| < MinPts \end{cases}$$

$core\text{-}distance_{\varepsilon,MinPts}(p) = \varepsilon'$

$reachability\text{-}distance_{\varepsilon,MinPts}(p, r) = \varepsilon'$

$reachability\text{-}distance_{\varepsilon,MinPts}(p, r') = d(p, r')$




*MinPts* = 5

ε = 3 cm

- Note: ε is not necessary.



7

## OPTICS - Main idea

- Intuitively, the reachability-distance of an object *r* w.r.t. another object *p* is the smallest distance such that *r* is *directly density-reachable* (*DDR*) from *p* if *p* is a core object.

- Basically, if *r* and *p* are nearest neighbors, this is the $\varepsilon' < \varepsilon$ we need to assume in order to have *r* and *p* belong to the same cluster.



8

# OPTICS - Main idea

- Both the core-distance and the reachability-distance are undefined if no sufficiently dense cluster (w.r.t. ε) is available.
  - Given a sufficiently large ε, this will never happen, but then every ε-neighborhood query will return the entire database, resulting in an untractable runtime cost. Hence, the ε parameter is required to cut off the density of clusters that is no longer considered to be interesting.

- The parameter ε is strictly speaking not necessary. It can be set to a maximum value. It often claimed that OPTICS abstract from DBSCAN by removing this parameter. It does however play a practical role when it comes to **complexity** (i.e. time complexity).



# Extracting the Clusters

- The ordering information produced by OPTICS, is sufficient for the extraction of all density-based clusterings w.r.t. any distance $\varepsilon' < \varepsilon$ used in generating the order.

- Using a *reachability-plot* (a special kind of dendrogram, i.e. from Greek *dendron* "tree", -*gramma* "drawing", or a *tree diagram*), the hierarchical structure of the clusters can be obtained easily.

## Extracting the Clusters

- ***Reachability-plot***: a 2D plot, with the ordering of the points on the x-axis and the reachability distance on the y-axis.
  - Since points belonging to a cluster have a low reachability distance to their nearest neighbor, the clusters show up as valleys in the reachability plot. The deeper the valley, the denser the cluster.




## Pseudocode

- OPTICS hence outputs the points in a particular ordering, annotated with their *smallest reachability distance* (in the original algorithm, the core distance is also exported, but this is not required for further processing).

# Pseudocode

```
OPTICS (SetOfObjects, ε, MinPts, OrderedFile)
    OrderedFile.open();
    for i =1 to SetOfObjects.size do
        Object := SetOfObjects.get(i); // get an object from database
        if NOT Object.Processed then
            ExpandClusterOrder(SetOfObjects, Object, ε, MinPts, OrderedFile)
    OrderedFile.close();
end; // OPTICS
```





# Pseudocode

```
ExpandClusterOrder (SetOfObjects, Object, ε, MinPts, OrderedFile);
    neighbors := SetOfObjects.neighbors(Object, ε); // retrieve the ε-neighborhood of Object
    Object.Processed := TRUE;
    Object.reachability_distance := UNDEFINED;
    Object.setCoreDistance(neighbors, ε, MinPts); // determine the core distance for Object
    OrderedFile.write(Object); // write Object into the OrderFile with its c.d. and r.d.
    if Object.core_distance != UNDEFINED then // if Object is core, then collect its DDR to expand
        OrderSeeds.update(neighbors, Object); // sort objects by their r.d. to the closet core ★
        while NOT OrderSeeds.empty() do
            currentObject := OrderSeeds.next(); // get the object with the smallest r.d.
            neighbors := SetOfObjects.neighbors(currentObject, ε);
            currentObject.Processed := TRUE;
            currentObject.setCoreDistance(neighbors, ε, MinPts);
            OrderedFile.write(currentObject); // write current Object into the OrderFile with its …
            if currentObject.core_distance != UNDEFINED then
                OrderSeeds.update(neighbors, currentObject);
end; // ExpandClusterOrder
```

## Pseudocode

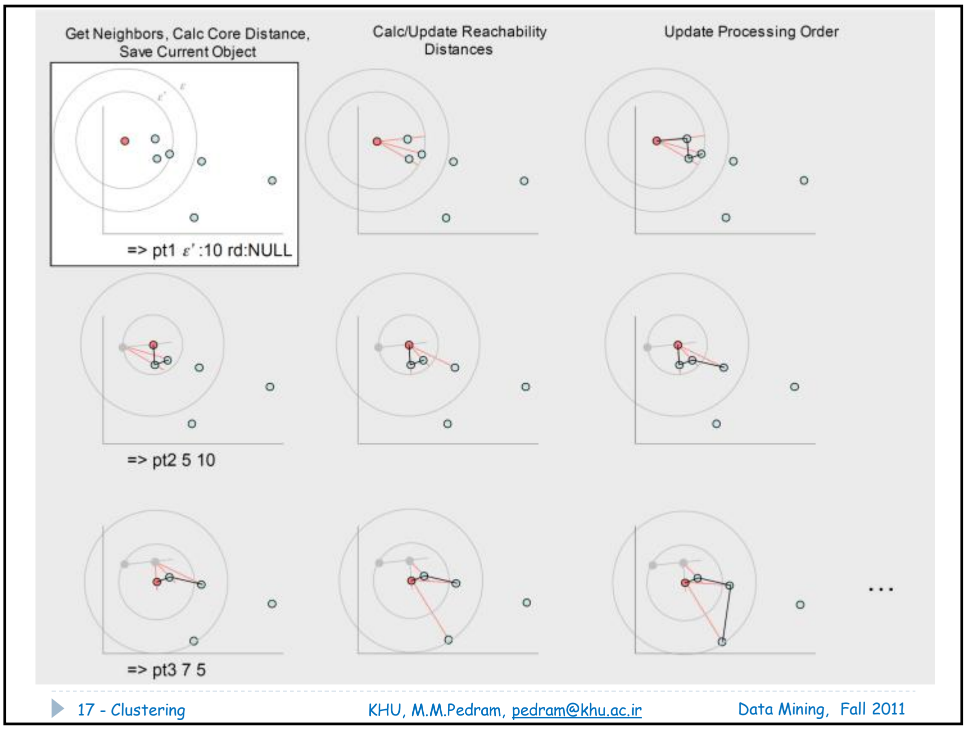
```
OrderSeeds::update(neighbors, centerObj):
    d = centerObj.coreDistance
    for each unprocessed obj in neighbors:
        newRdist = max(d, dist(obj, centerObj))
        if obj.reachability == NULL then
                obj.reachability = newRdist
                insert(obj, newRdist)
        elseif newRdist < obj.reachability then
                obj.reachability = newRdist
                decrease(obj, newRdist)
```



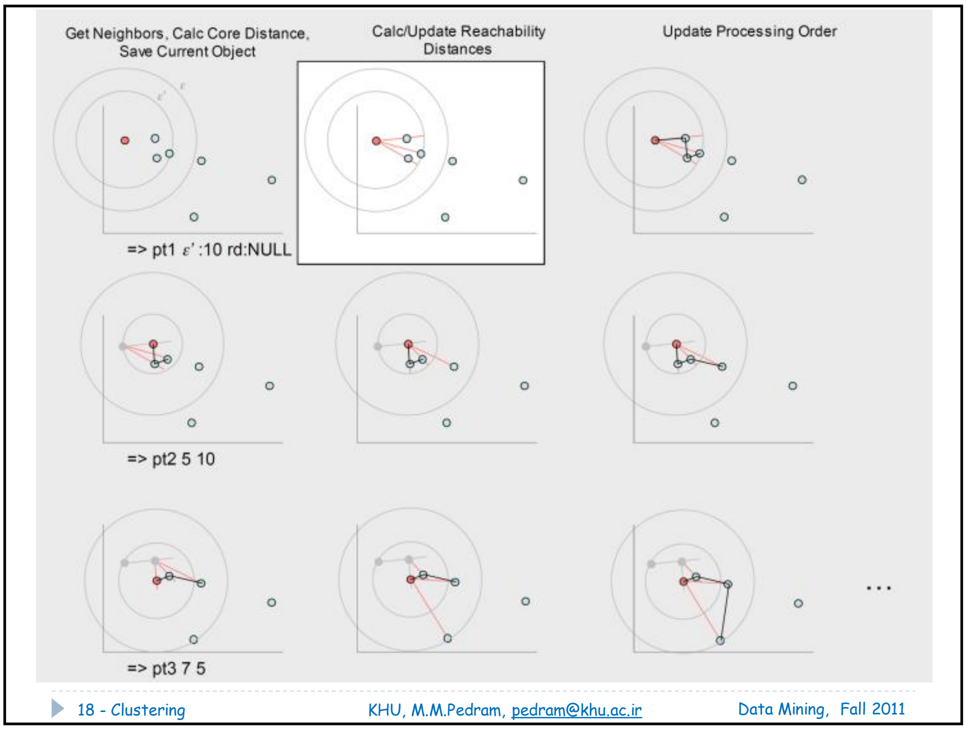


## Pseudocode

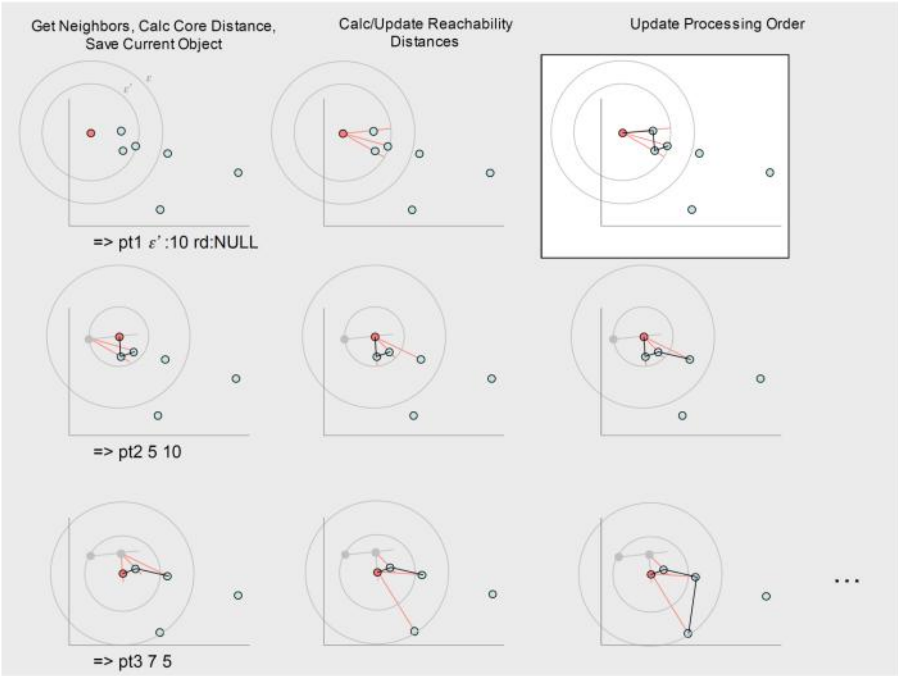
```
ExtractDBSCAN-Clustering (ClusterOrderedObjs, ε' , MinPts)
    // Precondition: ε' ≤ generating dist ε for ClusterOrderedObjs
    ClusterId := NOISE;
    for i=1 to ClusterOrderedObjs.size do
        Object := ClusterOrderedObjs.get(i);
        if Object.reachability_distance > ε' then
            // UNDEFINED > ε
            if Object.core_distance ≤ ε' then
                ClusterId := nextId(ClusterId);
                Object.clusterId := ClusterId;
            else
                Object.clusterId := NOISE;
        else // Object.reachability_distance ≤ ε'
            Object.clusterId := ClusterId;
end; // ExtractDBSCAN-Clustering
```


Get Neighbors, Calc Core Distance, Save Current Object
Calc/Update Reachability Distances
Update Processing Order
=> pt1 ε' :10 rd:NULL
=> pt2 5 10
=> pt3 7 5
...
17 - Clustering
KHU, M.M.Pedram, pedram@khu.ac.ir
Data Mining, Fall 2011

17

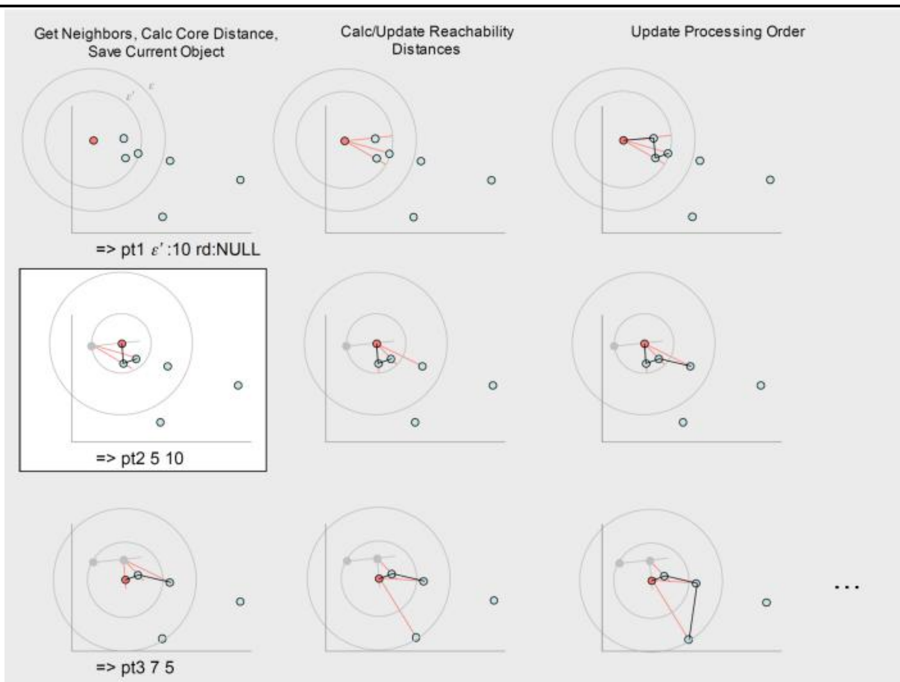

Get Neighbors, Calc Core Distance, Save Current Object
Calc/Update Reachability Distances
Update Processing Order
=> pt1 ε' :10 rd:NULL
=> pt2 5 10
=> pt3 7 5
...
18 - Clustering
KHU, M.M.Pedram, pedram@khu.ac.ir
Data Mining, Fall 2011

18


Get Neighbors, Calc Core Distance, Save Current Object
Calc/Update Reachability Distances
Update Processing Order
=> pt1 ε' :10 rd:NULL
=> pt2 5 10
=> pt3 7 5
...

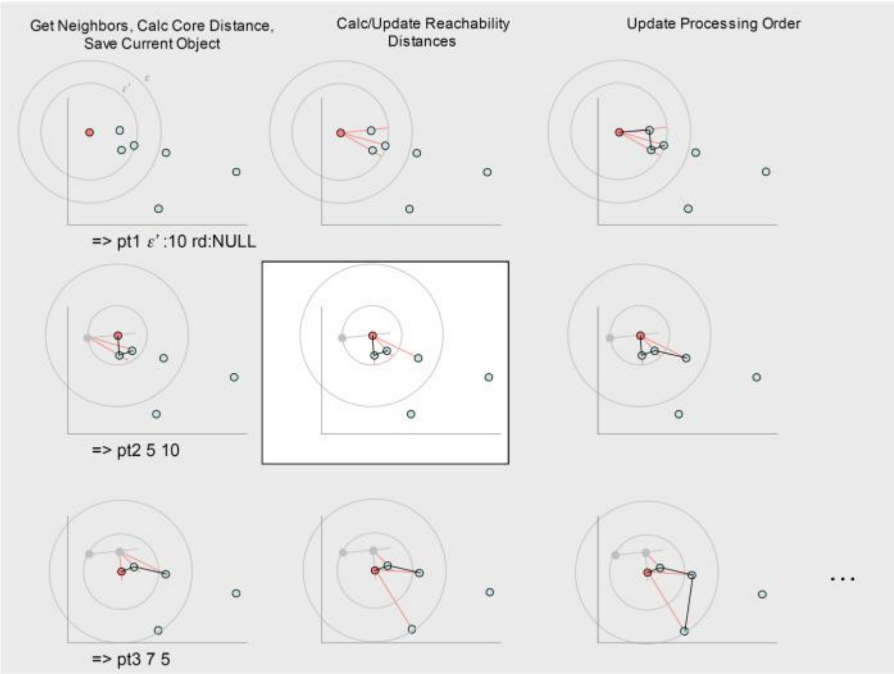


19


Get Neighbors, Calc Core Distance, Save Current Object
Calc/Update Reachability Distances
Update Processing Order
=> pt1 ε' :10 rd:NULL
=> pt2 5 10
=> pt3 7 5
...



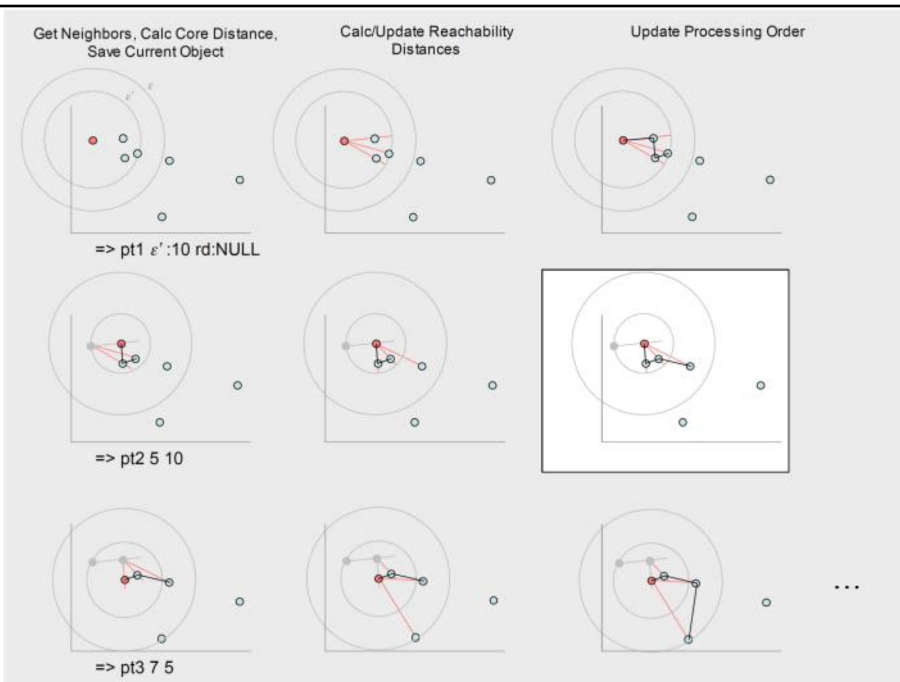
20



Get Neighbors, Calc Core Distance, Save Current Object
Calc/Update Reachability Distances
Update Processing Order
=> pt1 ε' :10 rd:NULL
=> pt2 5 10
=> pt3 7 5
...



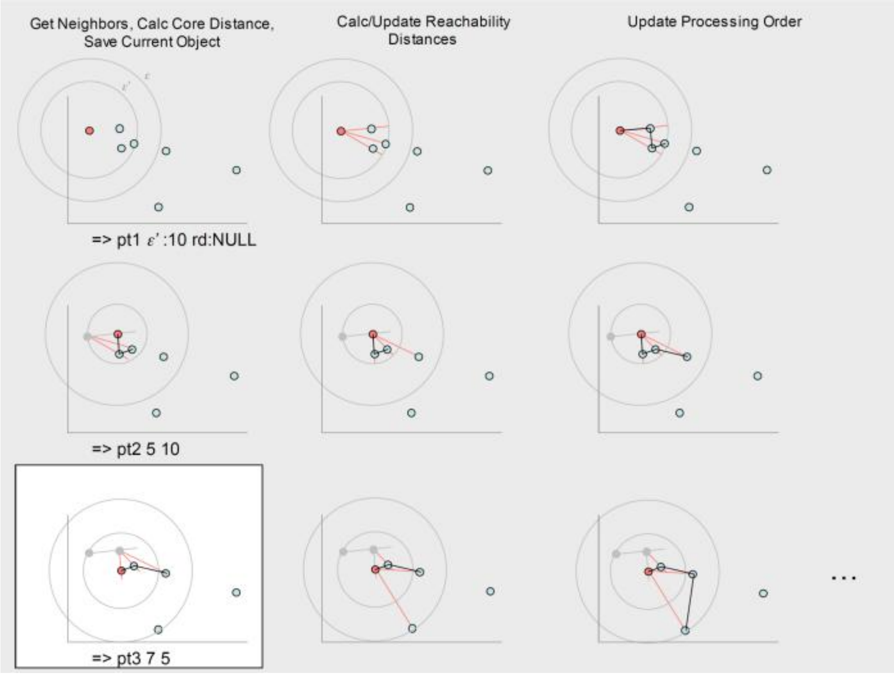

21



Get Neighbors, Calc Core Distance, Save Current Object
Calc/Update Reachability Distances
Update Processing Order
=> pt1 ε' :10 rd:NULL
=> pt2 5 10
=> pt3 7 5
...



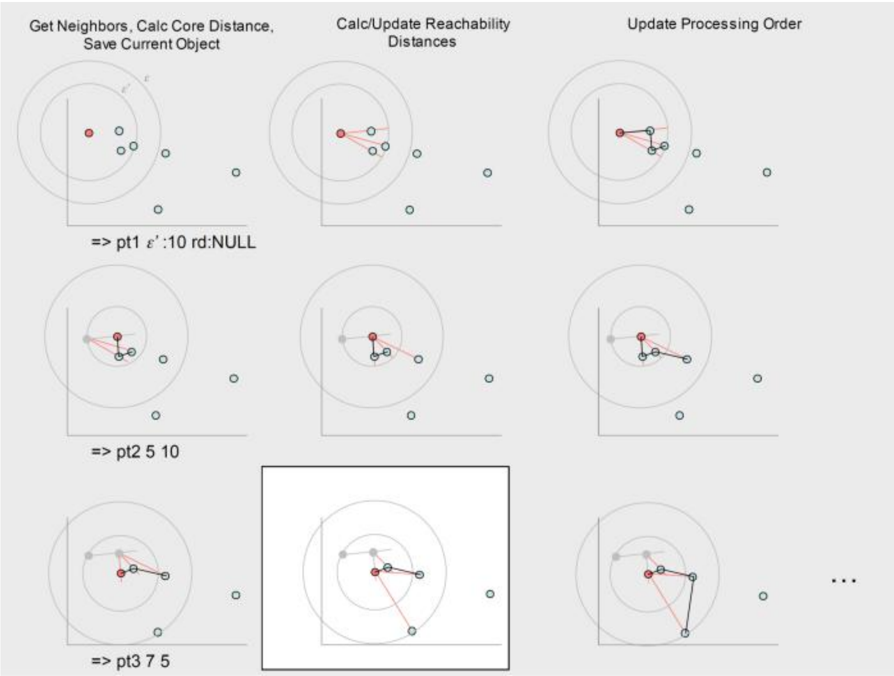

22



Get Neighbors, Calc Core Distance, Save Current Object
Calc/Update Reachability Distances
Update Processing Order
=> pt1 ε' :10 rd:NULL
=> pt2 5 10
=> pt3 7 5
...



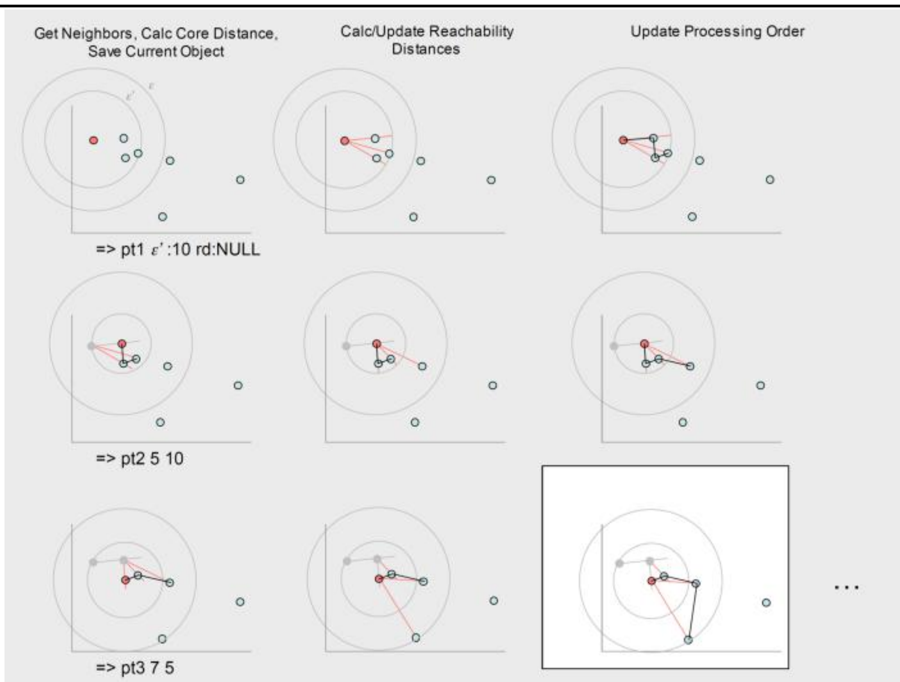

23



Get Neighbors, Calc Core Distance, Save Current Object
Calc/Update Reachability Distances
Update Processing Order
=> pt1 ε' :10 rd:NULL
=> pt2 5 10
=> pt3 7 5
...

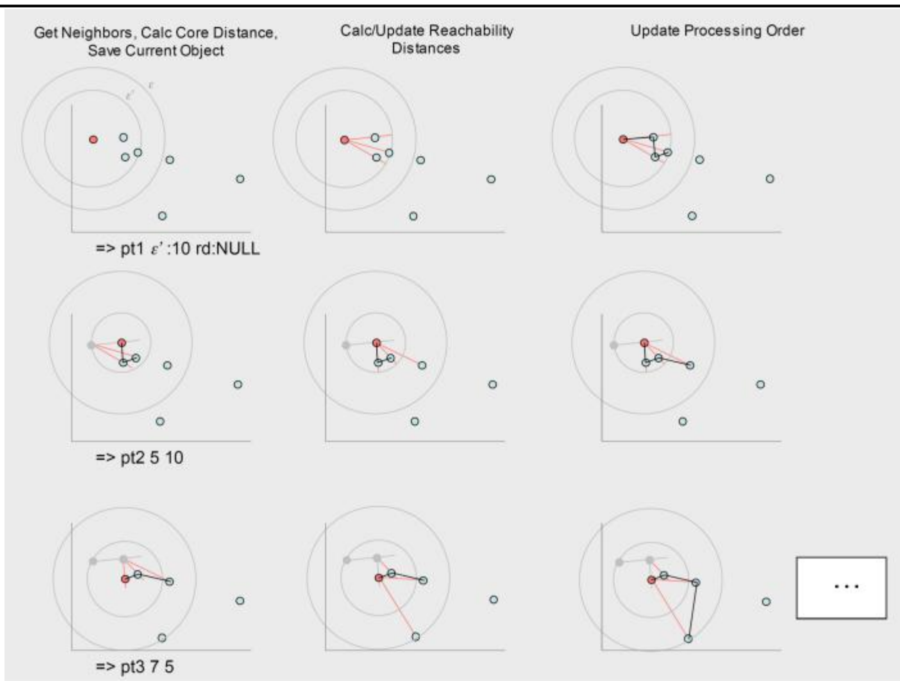



24




25 - Clustering KHU, M.M.Pedram, pedram@khu.ac.ir Data Mining, Fall 2011

25

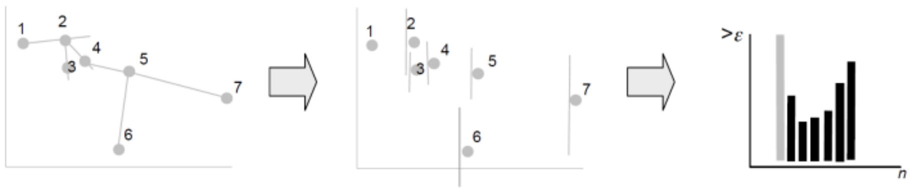



26 - Clustering KHU, M.M.Pedram, pedram@khu.ac.ir Data Mining, Fall 2011

26

# Reachability Plots

- A **reachability plot** is a bar chart that shows each object's reachability distance in the order the object was processed.

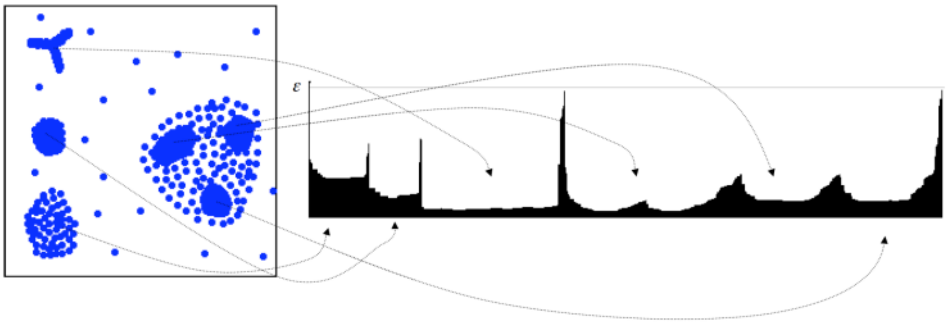


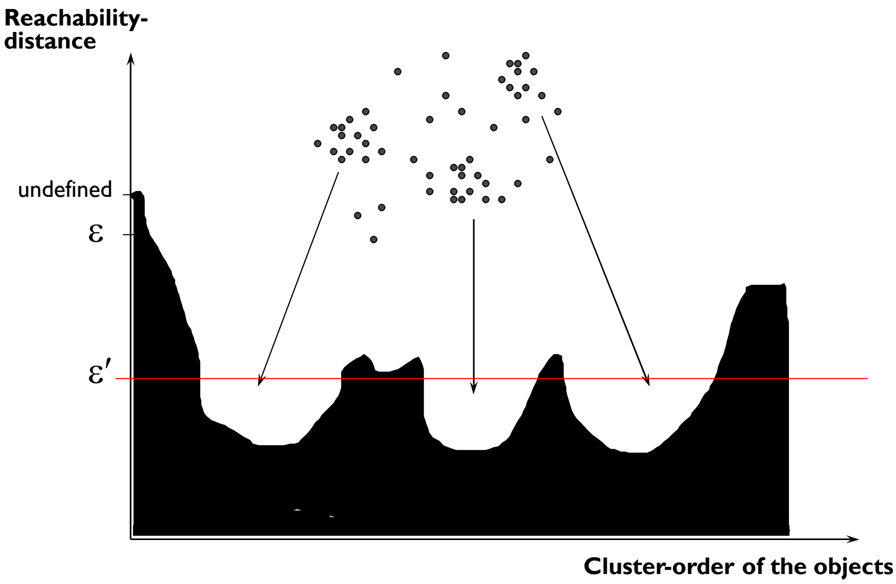





# Reachability Plots

- Reachability plots clearly show the cluster structure of the data.

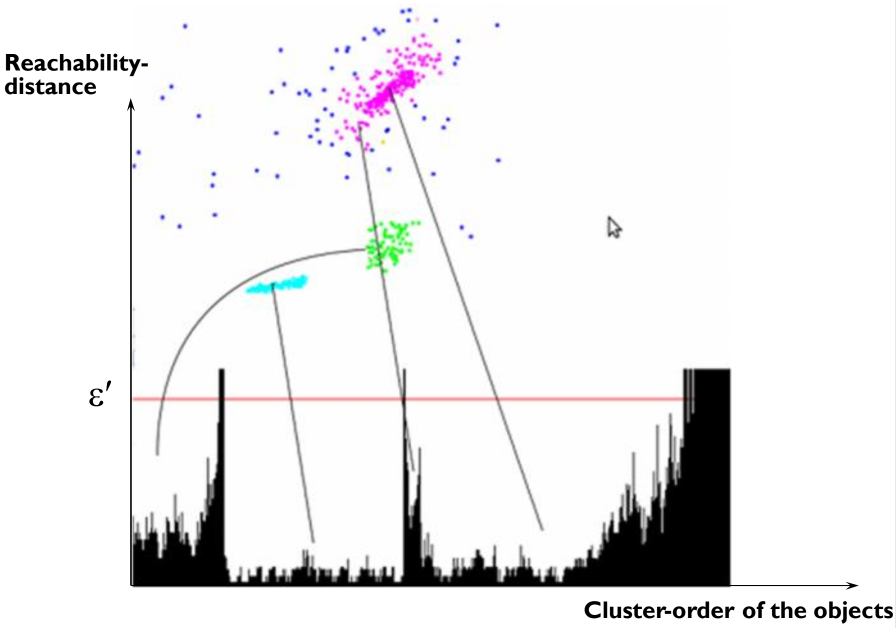




Reachability-distance
undefined
ε
ε′
Cluster-order of the objects



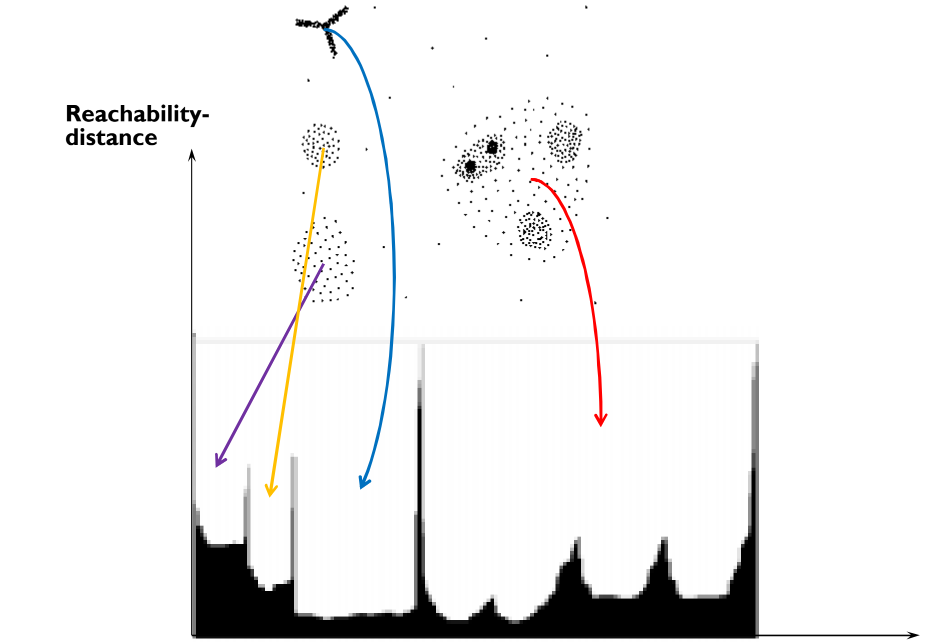

29



Reachability-distance
ε′
Cluster-order of the objects




30








# Conclusion

- It addresses one of DBSCAN's major weaknesses: the problem of detecting meaningful clusters in data of varying density. In order to do so:
  - The points of the database are (linearly) ordered such that points which are spatially closest become neighbors in the ordering.
  - Additionally, a special distance is stored for each point that represents the density that needs to be accepted for a cluster in order to have both points belong to the same cluster. This is represented as a dendrogram.
- Because of the structural equivalence of the OPTICS algorithm to DBSCAN, the OPTICS algorithm has the same runtime complexity as that of DBSCAN, that is, $O(n.\log n)$ if a spatial index is used, where $n$ is the number of objects.