Scaling Density-Based Clustering to Large Collections of Sets

Daniel Kocher[*], Nikolaus Augsten[*], and Willi Mann[+]

[*]University of Salzburg [+]Celonis SE

EDBT 2021

March 25, 2021

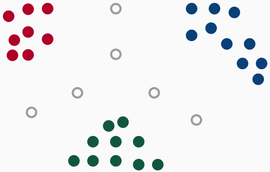
database research group

PARIS LODRON UNIVERSITY SALZBURG

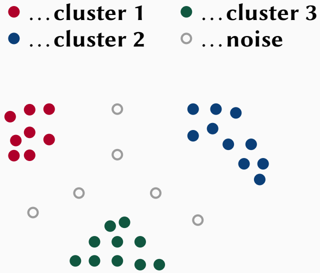
Density-Based Clustering:

Density-Based Clustering:

● …cluster 1 ● …cluster 3
● …cluster 2 ○ …noise



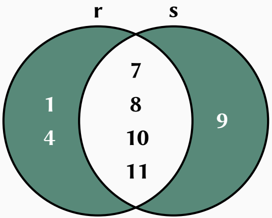
Density-Based Clustering:



Sets and the Hamming Distance:

Each data point represents a set:

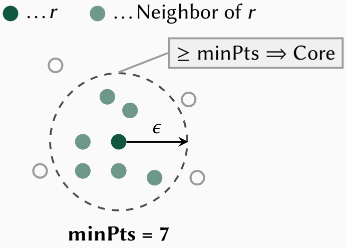
r: | 1 | 4 | 7 | 8 | 10 | 11 | **s:** | 7 | 8 | 9 | 10 | 11 |



$$H(r, s) = 3$$

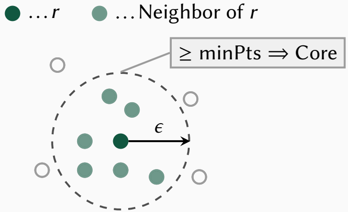
The DBSCAN Algorithm[1]

Two Parameters: ϵ (similarity threshold), minPts (density threshold)

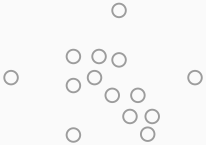


[1] Ester et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. SIGKDD 1996.

Two Parameters: ϵ (similarity threshold), minPts (density threshold)



minPts = 7

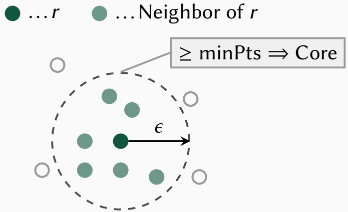


DBSCAN or **neighbor-by-neighbor order**

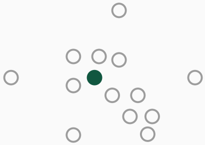
[1] Ester et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. SIGKDD 1996.

The DBSCAN Algorithm[1]

Two Parameters: ϵ (similarity threshold), minPts (density threshold)



minPts = 7

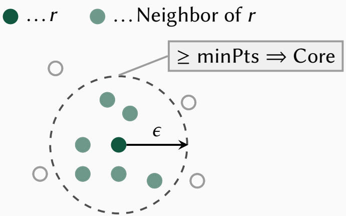


DBSCAN or **neighbor-by-neighbor order**

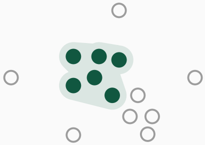
[1] Ester et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. SIGKDD 1996.

The DBSCAN Algorithm[1]

Two Parameters: ϵ (similarity threshold), minPts (density threshold)



minPts = 7

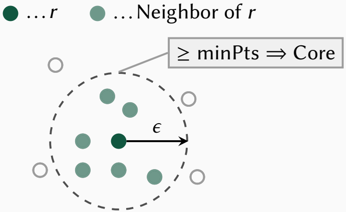


DBSCAN or **neighbor-by-neighbor order**

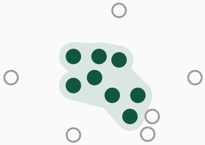
[1] Ester et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. SIGKDD 1996.

The DBSCAN Algorithm[1]

Two Parameters: ϵ (similarity threshold), minPts (density threshold)



minPts = 7

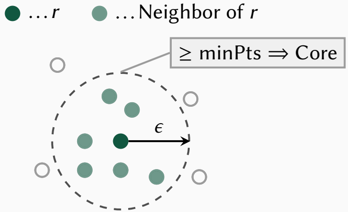


DBSCAN or **neighbor-by-neighbor order**

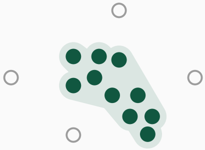
[1] Ester et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. SIGKDD 1996.

The DBSCAN Algorithm[1]

Two Parameters: ϵ (similarity threshold), minPts (density threshold)



minPts = 7



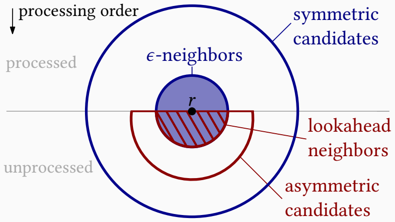
DBSCAN or **neighbor-by-neighbor order**

- **Indexes** accelerate neighborhood queries.
- **Symmetric** indexes **return all neighbors** for a given query point.

[1] Ester et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. SIGKDD 1996.

Effective Indexes for Sets

- Optimized set indexes are **asymmetric** and generate fewer candidates.
- **Asymmetric indexes**
  - rely on a specific **processing order** and
  - return only a **specific part** of the **neighborhood**, the **lookahead neighbors**.
- **Problem:** Asymmetric indexes are not compatible with DBSCAN order.



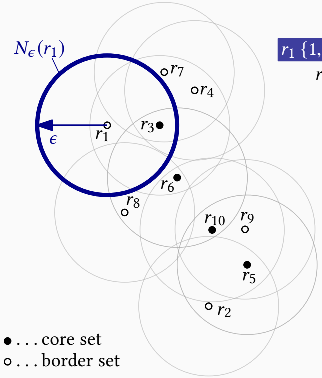
The Spread Algorithm

Spread integrates asymmetric indexes into DBSCAN in linear space.

Algorithm Outline:

- Impose a **processing order** that is compatible with asymmetric indexes.
- Retrieve each pair of neighbors once $\Rightarrow$ **lookahead neighbors are sufficient.**
- Lookahead neighbors are not enough to deduce clusters $\Rightarrow$ **backlinks to fix it.**
- Multiple subclusters may grow independently $\Rightarrow$ **Spanning tree of subclusters.**
- Propagate information forward, i.e., **spread the information.**

Find All (Border) Points of a Cluster



$\epsilon = 3$, minPts = 4.

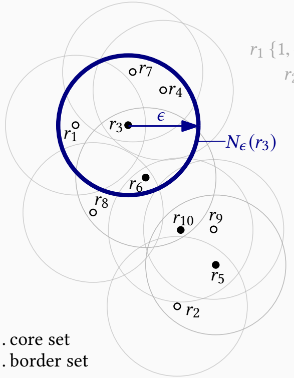
Collection R:

r_1	$\{1, 4, 7, 8, 10, 11, 12, 13, 14\}$
r_2	$\{1, 3, 4, 5, 6, 12, 13, 14\}$
r_3	$\{1, 4, 7, 8, 10, 11\}$
r_4	$\{7, 8, 9, 10, 11\}$
r_5	$\{1, 3, 4, 5, 6\}$
r_6	$\{1, 2, 4, 7, 8\}$
r_7	$\{7, 8, 10, 11\}$
r_8	$\{3, 4, 7, 8\}$
r_9	$\{2, 3, 4, 5\}$
r_{10}	$\{1, 2, 3, 4\}$

Problem:

- r_1 sees r_3 as lookahead neighbor.
- **But: r_3 does not see r_1.**

Find All (Border) Points of a Cluster



● . . . core set
○ . . . border set

$\epsilon = 3$, minPts $= 4$.

Collection R:

r_1 $\{1, 4, 7, 8, 10, 11, 12, 13, 14\}$
r_2 $\{1, 3, 4, 5, 6, 12, 13, 14\}$
r_3 $\{1, 4, 7, 8, 10, 11\}$
r_4 $\{7, 8, 9, 10, 11\}$
r_5 $\{1, 3, 4, 5, 6\}$
r_6 $\{1, 2, 4, 7, 8\}$
r_7 $\{7, 8, 10, 11\}$
r_8 $\{3, 4, 7, 8\}$
r_9 $\{2, 3, 4, 5\}$
r_{10} $\{1, 2, 3, 4\}$

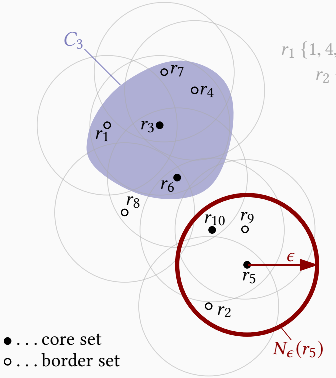
Problem:

- r_1 sees r_3 as lookahead neighbor.
- **But: r_3 does not see r_1.**

Solution:

- r_3 stores a link back to r_1.
- **r_3 sees r_1 retrospectively.**
- Max. minPts backlinks per set $\Rightarrow$ **$O(n)$ space**, with $n = |R|$.

Identifying Core Points Correctly



Collection R:

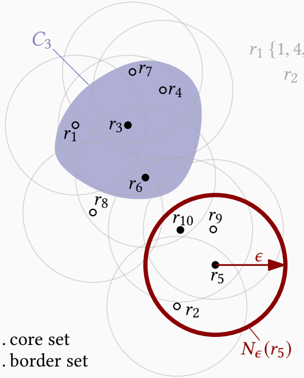
r_1 {1, 4, 7, 8, 10, 11, 12, 13, 14}
r_2 {1, 3, 4, 5, 6, 12, 13, 14}
r_3 {1, 4, 7, 8, 10, 11}
r_4 {7, 8, 9, 10, 11}
r_5 {1, 3, 4, 5, 6}
r_6 {1, 2, 4, 7, 8}
r_7 {7, 8, 10, 11}
r_8 {3, 4, 7, 8}
r_9 {2, 3, 4, 5}
r_{10} {1, 2, 3, 4}

$\epsilon = 3$, minPts = 4.

Problem:

- r_5 sees only r_9 and r_{10} as neighbors.
- **But: r_2 sees r_5 as neighbor**
  $\Rightarrow$ r_5 wrongly classified as non-core.

Identifying Core Points Correctly



● . . . core set
○ . . . border set

$\epsilon = 3$, minPts $= 4$.

Collection R:

r_1 $\{1, 4, 7, 8, 10, 11, 12, 13, 14\}$
r_2 $\{1, 3, 4, 5, 6, 12, 13, 14\}$
r_3 $\{1, 4, 7, 8, 10, 11\}$
r_4 $\{7, 8, 9, 10, 11\}$
r_5 $\{1, 3, 4, 5, 6\}$
r_6 $\{1, 2, 4, 7, 8\}$
r_7 $\{7, 8, 10, 11\}$
r_8 $\{3, 4, 7, 8\}$
r_9 $\{2, 3, 4, 5\}$
r_{10} $\{1, 2, 3, 4\}$

+1

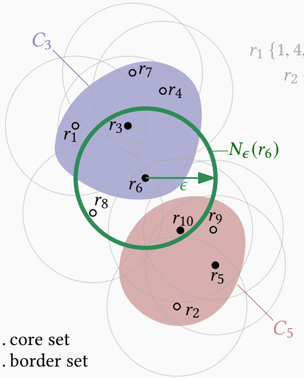
Problem:

- r_5 sees only r_9 and r_{10} as neighbors.
- **But: r_2 sees r_5 as neighbor**
  $\Rightarrow$ r_5 wrongly classified as non-core.

Solution:

- Maintain a density counter per set.
- r_2 increments counter of r_5.
- Use **counter to classify r_5** $\Rightarrow$ core.

Subcluster Merging



● . . . core set
○ . . . border set

Collection R:

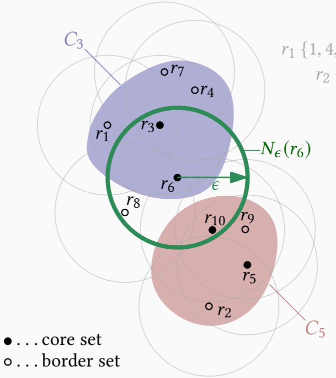
r_1 {1, 4, 7, 8, 10, 11, 12, 13, 14}
r_2 {1, 3, 4, 5, 6, 12, 13, 14}
r_3 {1, 4, 7, 8, 10, 11}
r_4 {7, 8, 9, 10, 11}
r_5 {1, 3, 4, 5, 6}
r_6 {1, 2, 4, 7, 8}
r_7 {7, 8, 10, 11}
r_8 {3, 4, 7, 8}
r_9 {2, 3, 4, 5}
r_{10} {1, 2, 3, 4}

$\epsilon = 3$, minPts = 4.

Problem:

- r_6 belongs to subcluster C_3 (of r_3).
- r_6 sees r_{10}, which is core and belongs to subcluster C_5 (of r_5).
- **Subclusters must be merged.**

Subcluster Merging



Collection R:

r_1 $\{1, 4, 7, 8, 10, 11, 12, 13, 14\}$
r_2 $\{1, 3, 4, 5, 6, 12, 13, 14\}$
r_3 $\{1, 4, 7, 8, 10, 11\}$
r_4 $\{7, 8, 9, 10, 11\}$
r_5 $\{1, 3, 4, 5, 6\}$
r_6 $\{1, 2, 4, 7, 8\}$
r_7 $\{7, 8, 10, 11\}$
r_8 $\{3, 4, 7, 8\}$ $\oplus$
r_9 $\{2, 3, 4, 5\}$
r_{10} $\{1, 2, 3, 4\}$

$\epsilon = 3$, minPts $= 4$.

Problem:

- r_6 belongs to subcluster C_3 (of r_3).
- r_6 sees r_{10}, which is core and belongs to subcluster C_5 (of r_5).
- **Subclusters must be merged.**

Solution:

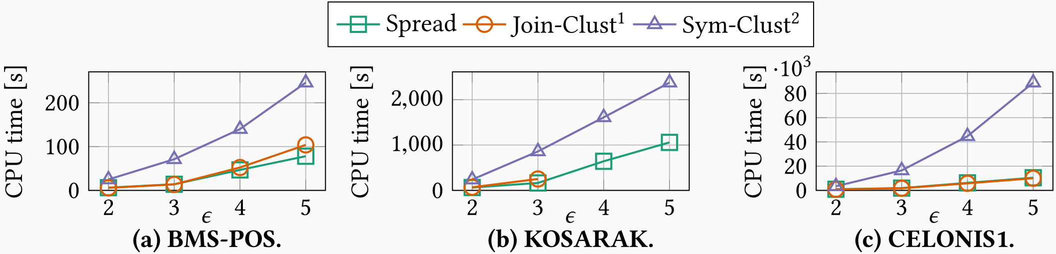
- **Link subclusters** in spanning tree.
- Disjoint-set data structure $\Rightarrow$ $\boldsymbol{O(n)}$ **space**, with $n = |R|$.

Experimental Results – Runtime

BMS-POS:
$3.2 \cdot 10^5$ sets (avg. size: 9.3)

KOSARAK:
$6.1 \cdot 10^5$ sets (avg. size: 11.9)

CELONIS1:
$8.2 \cdot 10^6$ sets (avg. size: 20.3)



Runtime over ϵ, minPts = 16.

[1] **Join-Clust:** DBSCAN is executed after a set similarity join that materializes neighborhoods.

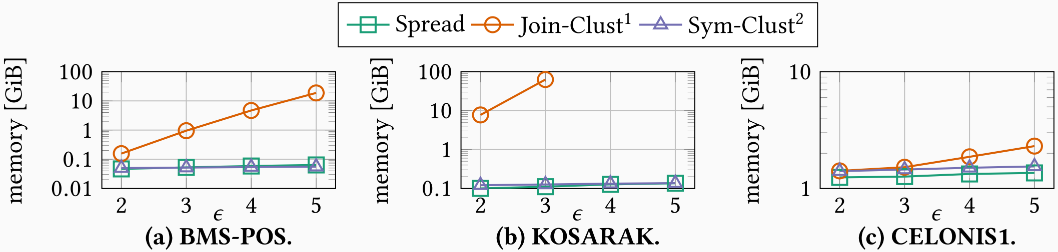
[2] **Sym-Clust:** DBSCAN is executed with a symmetric index to query the neighborhoods on the fly.

Experimental Results – Memory

BMS-POS:
$3.2 \cdot 10^5$ sets (avg. size: 9.3)

KOSARAK:
$6.1 \cdot 10^5$ sets (avg. size: 11.9)

CELONIS1:
$8.2 \cdot 10^6$ sets (avg. size: 20.3)



Heap peak over ϵ, minPts = 16.

[1] **Join-Clust:** DBSCAN is executed after a set similarity join that materializes neighborhoods.
[2] **Sym-Clust:** DBSCAN is executed with a symmetric index to query the neighborhoods on the fly.

Concluding Remarks

	Fast Runtime	Low Memory	Effective Set Indexes
DBSCAN	×	✓ (linear)	×
Join-Based	✓	× (quadratic)	✓
Spread	✓	✓ (linear)	✓

Conclusion:

- **Asymmetric set indexes:** more effective but **not compatible with DBSCAN.**
- **Materialization-based solutions** have a **large memory footprint.**
- **Spread** combines the **best of the two worlds** and is **DBSCAN-compliant**.

Scaling Density-Based Clustering to Large Collections of Sets

Daniel Kocher*, Nikolaus Augsten*, and Willi Mann+

*University of Salzburg +Celonis SE

EDBT 2021

March 25, 2021

database research group

PARIS LODRON UNIVERSITY SALZBURG