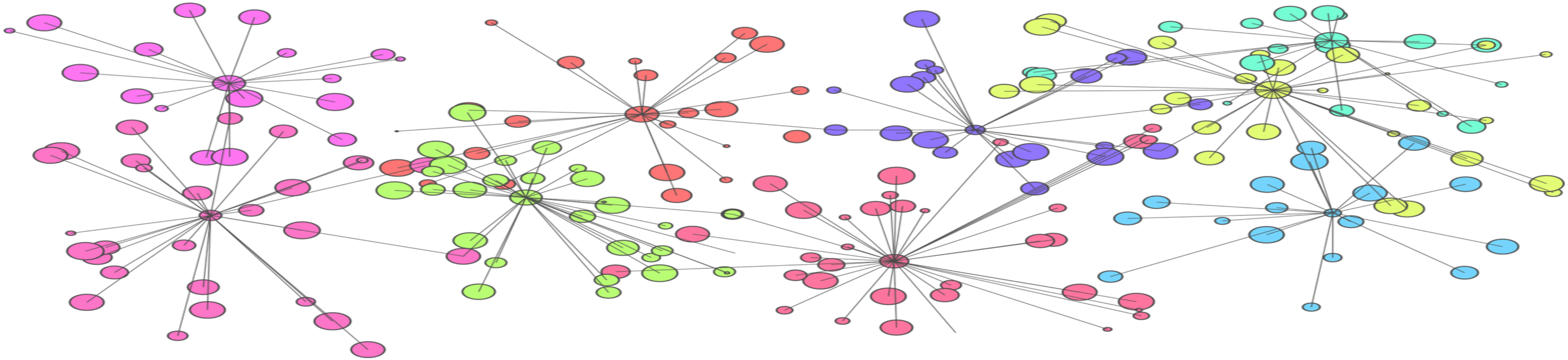


Python Implementation of the SUBSCALE algorithm

based on Java reference

Speaker:
Stanislav Ramin

Supervisor:
Prof. Dr. rer. nat. Tobias Lauer
M. Sc. Jürgen Prinzbach

Hochschule Offenburg
offenburg.university

Introduction

- First implemented
  - University of Western Australia in 2014
  - Ph.D. Amardeep Kaur
  - Java
- M.Sc. Nicolas Kiefer
  - C++
  - CUDA for GPU support
  - In 2020

- Outline
  - Why Python?
  - Theory
    - Clustering
    - Problems
    - Solutions
  - Subscale Algorithm
  - Performance and Comparison
    - Trade offs SUBSCALE
    - Other Clustering Algorithms

Why Python?

Hochschule Offenburg
offenburg.university

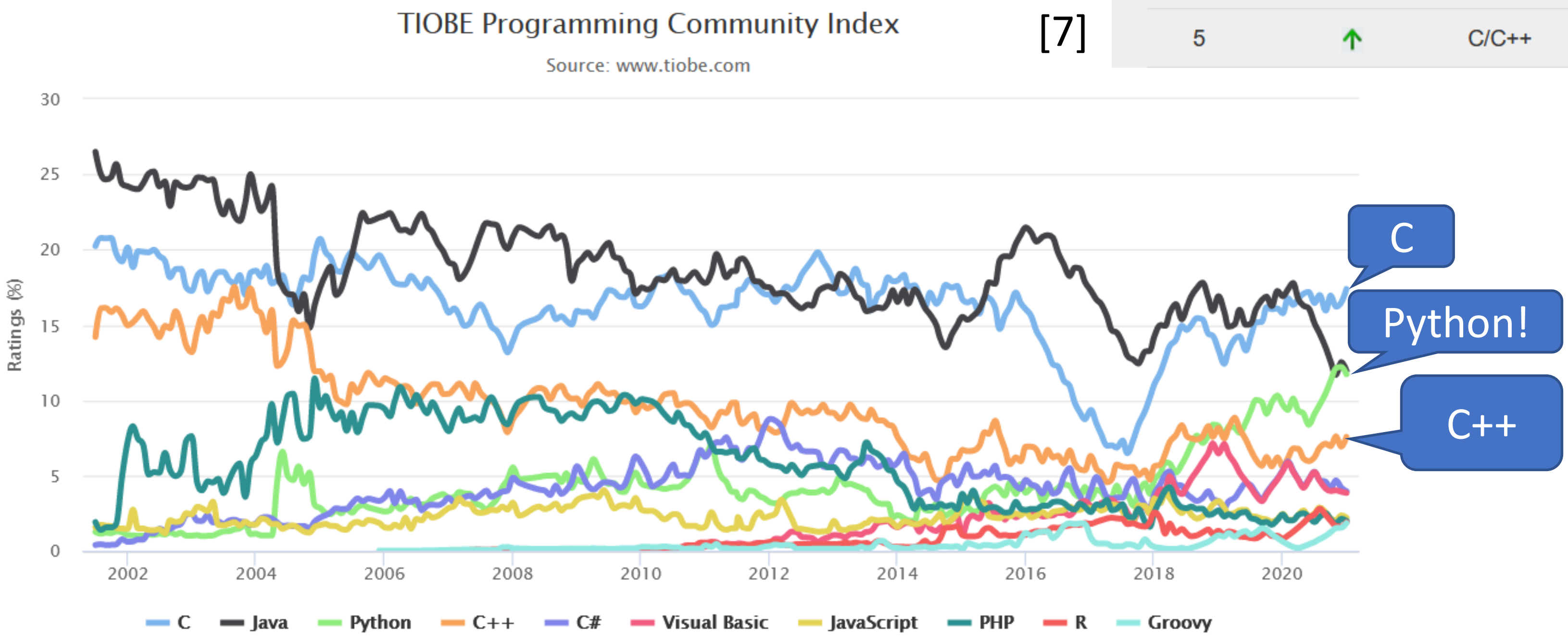
- Usability, System integration
  - State of the art for Machine learning
  - Front end for C and C++ programs
  - In Docker container

PYPL PopularitY:

Worldwide, Jan 2021 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	30.44 %	+1.2 %
2		Java	16.76 %	-2.0 %
3		JavaScript	8.44 %	+0.3 %
4		C#	6.53 %	-0.7 %
5	↑	C/C++	6.33 %	+0.3 %

[7]



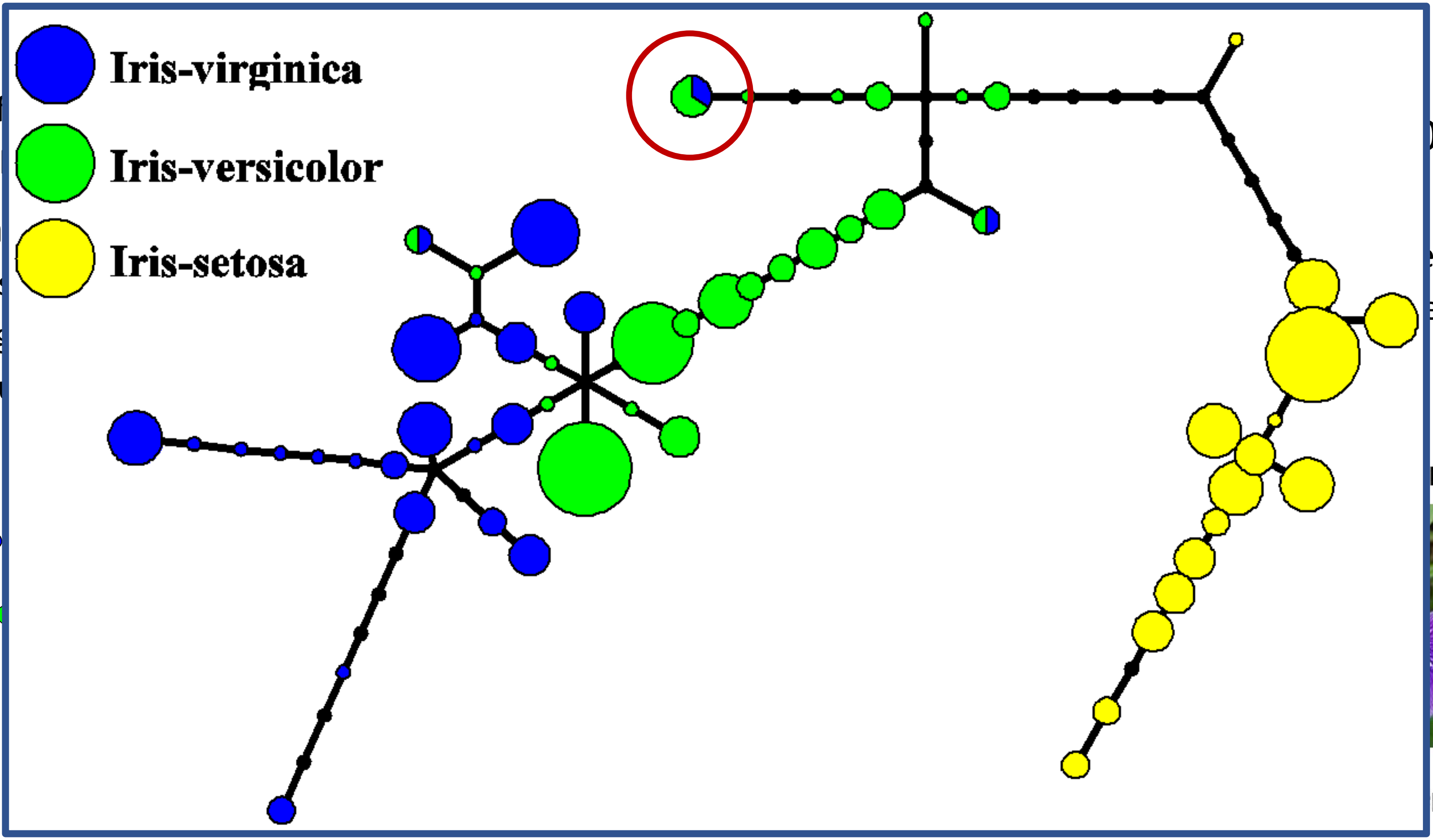
[8]

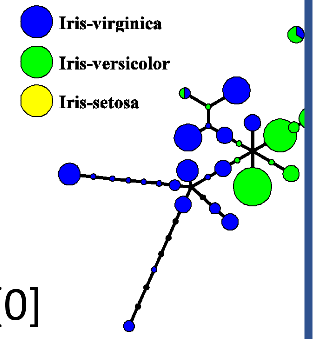
What is clustering and why use it?
Example Iris Dataset → Number of species?

Hochschule Offenburg
offenburg.university

- Cluster
  - Set of
  - Each
- Field of In
  - Data
  - Biolog

  Data fra

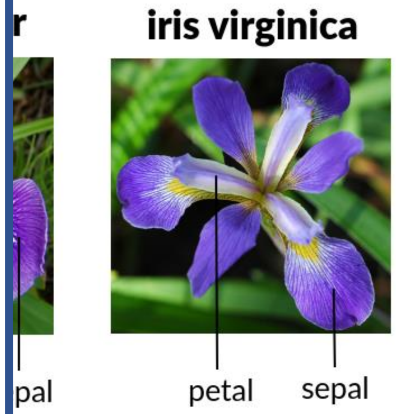




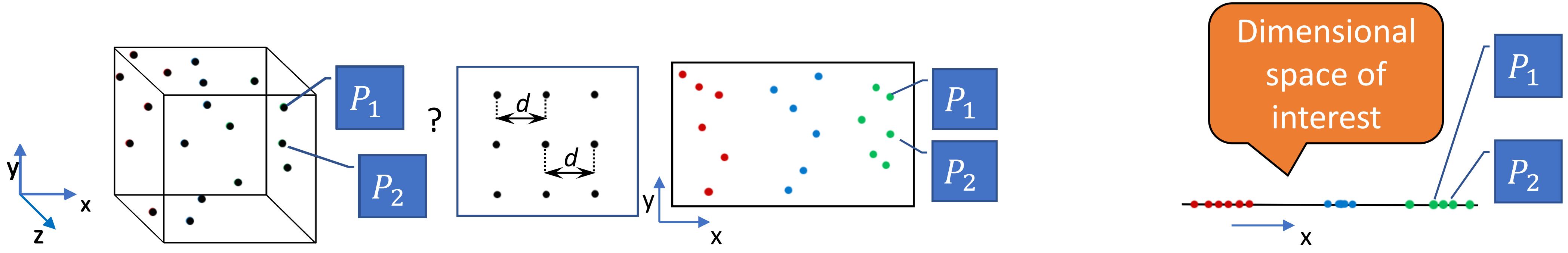
[0]

)
eight
eight

r

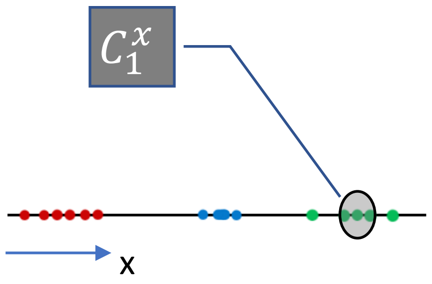


Curse of Dimensionality



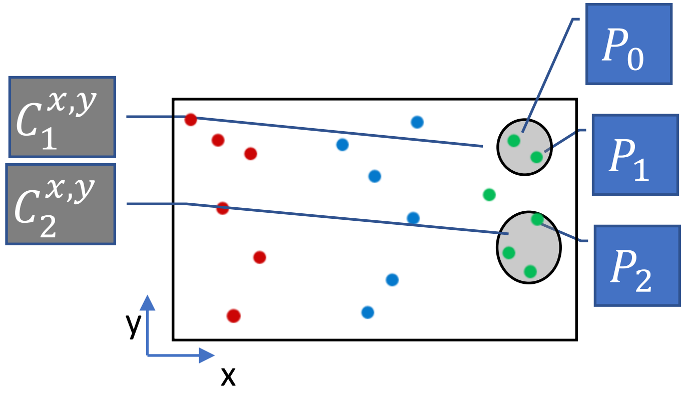
Bottom-up clustering

Cluster in 1d space



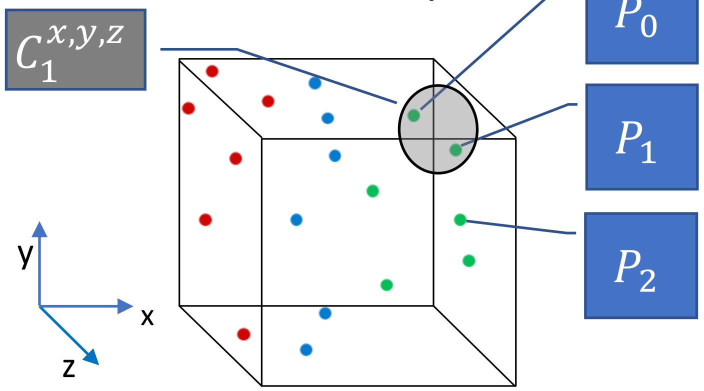
P_0, P_1, P_2
clustered in C_1^x 😊

Cluster in 2d space



$P_1 \in C_1^{x,y}$ and $P_2 \notin C_1^{x,y}$ 😒

Cluster in 3d space

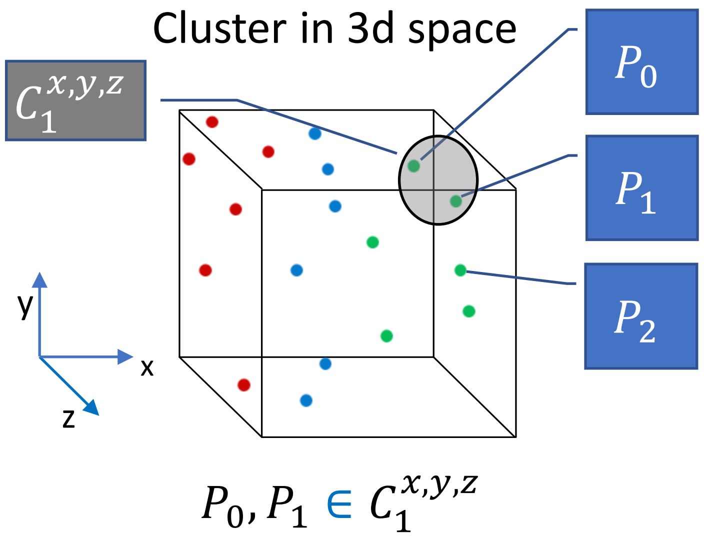


$P_0, P_1 \in C_1^{x,y,z}$

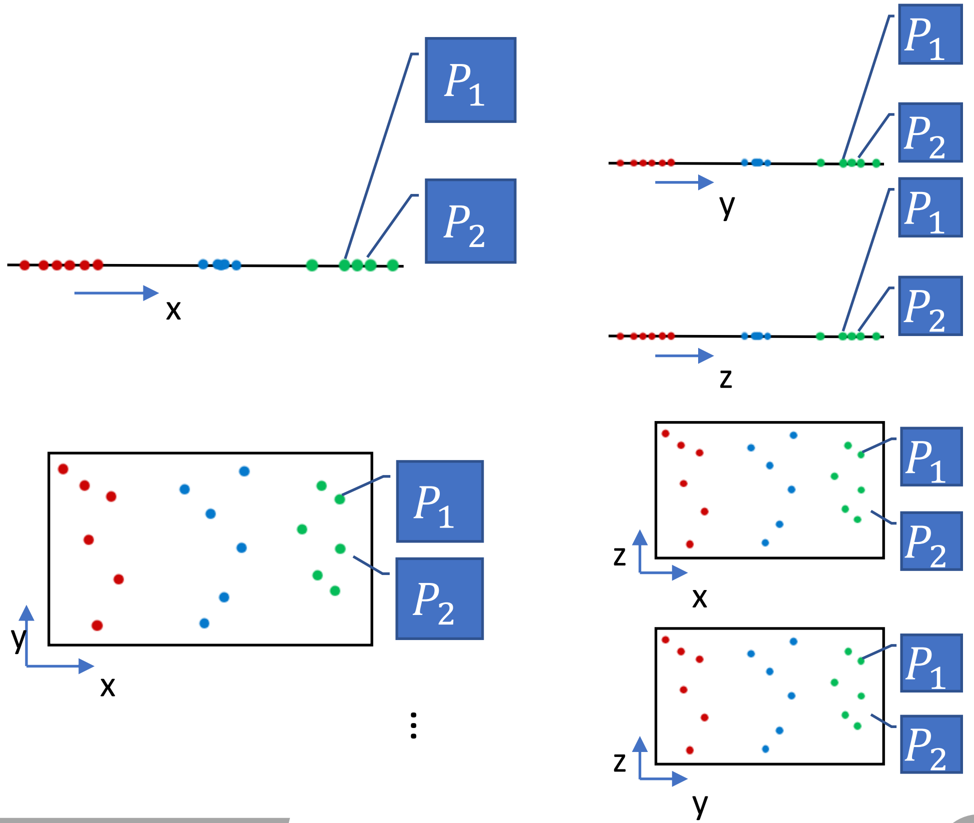
Hochschule Offenburg
offenburg.university

Solved Problems (I)

- Problem 1
  - Only Points in full dimensional space are discovered

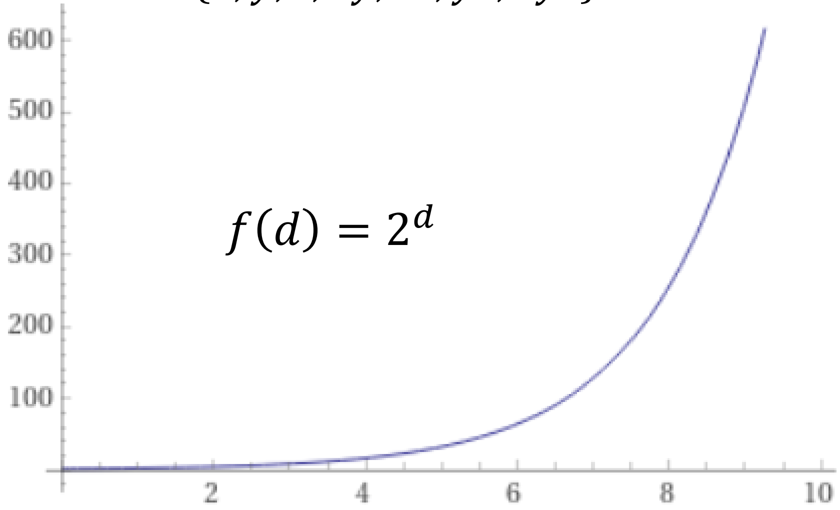


- Solution 1
  - Search in all Subspaces of the features

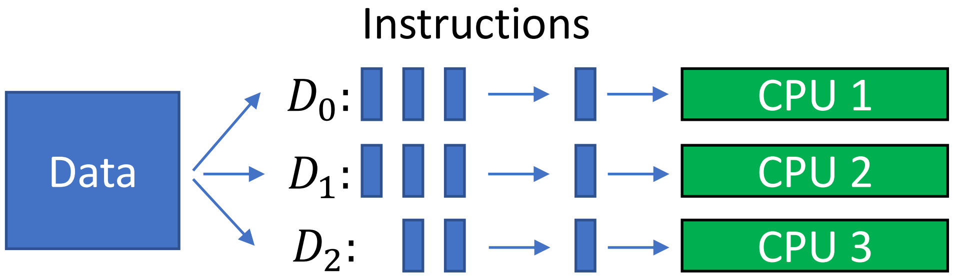


Solved Problems (II)

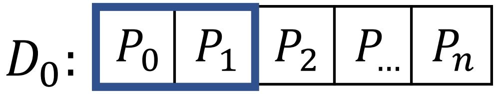
- Problem 2
  - d: number of dimensions or features
  - 2^d-1 possible subsets (subspaces)
  - Example:
    - For $d = 3$
    - Subspaces $= 7$ $\{x, y, z, xy, xz, yz, xyz\}$



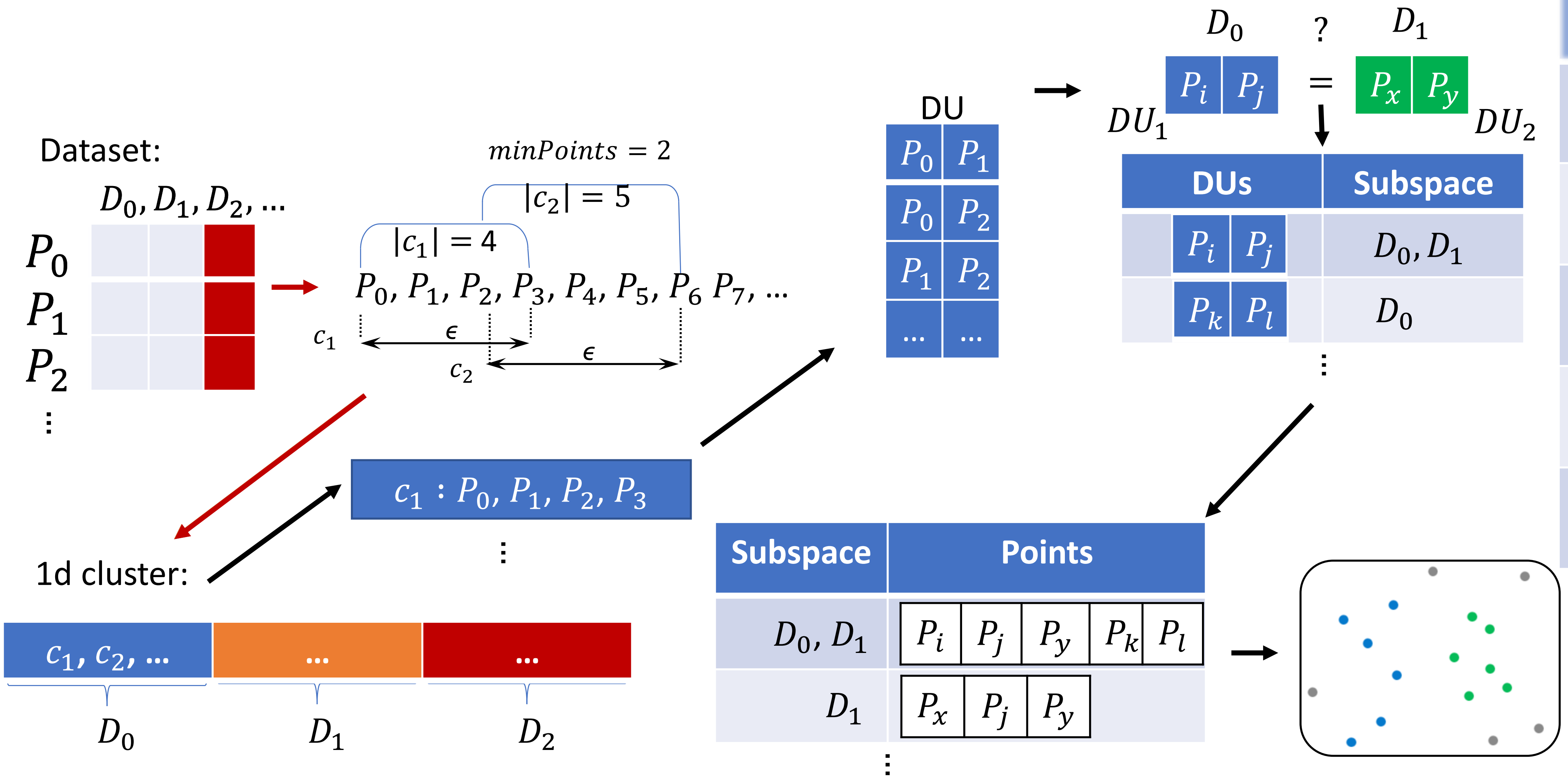
- Solution: SUBSPACE
  - Each dimension: processed 1 x
  - Good parallelization when data has a lot of dimensions



  - Only 1 scan per dimension



SUBSCALE Overview



1. Data Projection
2. CoreSet creation
3. Calc. of Dense Units
4. Collision of Dense Units
5. Transfor-mation
6. Final Clustering

3. Calculation of DUs

$c : P_0, P_1, \dots, P_5$ →

DU

P_0	P_1	P_2
P_0	P_1	P_3
P_0	P_2	P_3
…	…	…

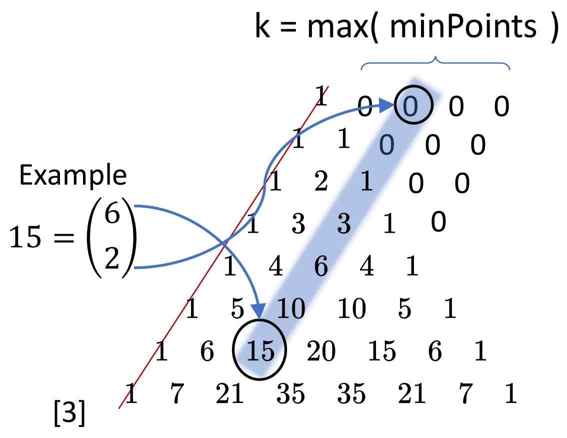
- Number of calculations
  - $C = \{0, 1, 2, 3, 4, 5\}, |C| = 6$
  - $\binom{n}{k}$: # of k-sized subsets of n
  - $k = minPoint = |DU| = 3$
  - $n = |C|$
  - E.g. $\binom{6}{3} = 20$ with $n = |C|$
- Co-Lexicographic Order
  - Sequential Calculation:
    0: {0, 1, 2}, 1: {0, 1, 3}, 2: {0, 2, 3},
    3: {1, 2, 3}, 4: {0, 1, 4}, 5: {0, 2, 4},
    … , 18: {2,4,5}, 19: {3,4,5}
- Runtime
  - $O(1)$

- Direct calculation of i-th subset
  - Knowledge of predecessor

    DU not required
  - useful for parallelization
  - $N_i = \binom{c_k}{k} + \binom{c_{k-1}}{k-1} + \dots + \binom{c_1}{1}$
  - $k = |DU|$
  - $n > c_k > c_{k-1} > \dots > c_1$
- *Example: 18. subset*: {2,4,5}
  - $18 = \binom{5}{3} + \binom{4}{2} + \binom{2}{1}$

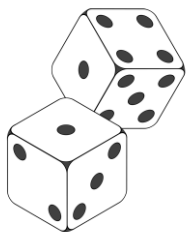
    $= 10 + 6 + 2$
- Runtime:
  - With lookup-table: $O(k \cdot log_2 n)$
  - $0 \leq N_i \leq \binom{n}{k}$

- Optimizing: binominal coefficient
  - Pascal Triangle as lookup-table



4. Collisions

- Label each point with random value
  - E.g. $Label(P_i) = 615233464694684$



[4]

Signature Calculation

$$\sum_{i=0}^{(minPoints)} Label(P_i)\,,\ P_i \in DU$$

D_0: P_1 | P_2 DU_1 $\overset{?}{=}$ D_1: P_x | P_y DU_2

$$Sig(DU_1) \overset{!}{=} Sig(DU_2)$$

Already in table

Collision table:

Signature	DUs	Subspace
$Sig(DU_1)$	P_1 P_2	D_0 D_1
$Sig(DU_x)$	P_x P_y	D_1

$\sum$ P_x P_y DU_x

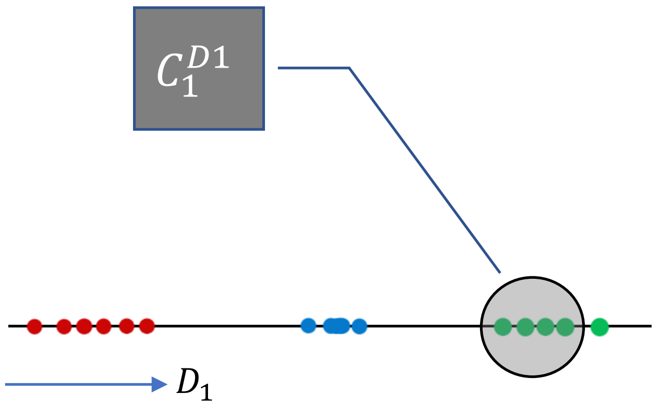
[illegible]

5. Mapping

Signature	DUs	Subspace
$Sig(DU_1)$	P_1 P_2 P_3	D_0 D_1
$Sig(DU_2)$	P_3 P_5 P_7	D_0
$Sig(DU_3)$	P_3 P_5 P_9	D_0
$Sig(DU_4)$	P_3 P_4 P_5	D_0 D_1

Subspace	Points
D_0, D_1	P_1 P_2 P_3 P_4 P_5
D_1	P_3 P_5 P_7 P_9

Potential cluster in 1d space



P_3, P_5, P_7 , P_9:
maybe clustered in C_1^{D1}

1. Data Projection
2. CoreSet creation
3. Calc. of Dense Units
4. Collision of Dense Units
5. Transfor-mation
6. Final Clustering

Final Clustering

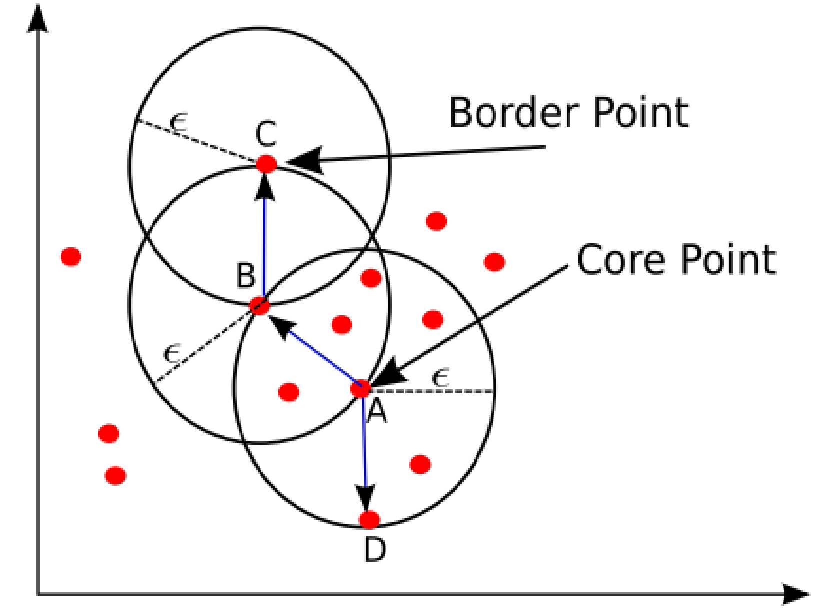
- Input for DBSCAN:
  - [Dims, Points]

Subspace	Points				
D_0, D_1	P_i	P_j	P_y	P_k	P_l
D_1	P_x	P_j	P_y		

- Output

Point	ClusterID
P_i	1
P_j	1
P_x	3
…	…

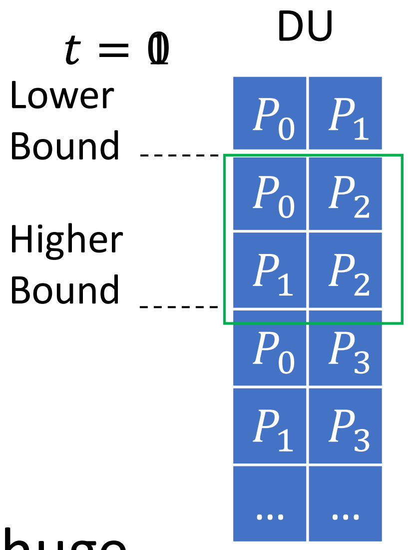
DBSCAN



[5]

Performance Trade Offs

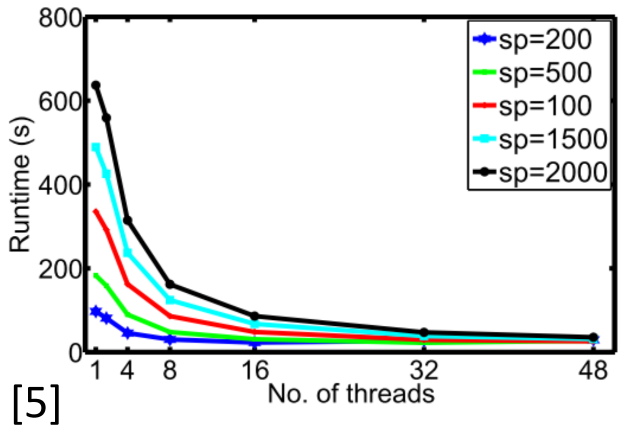
- Keep in mind
  - Computer memory
  - # Cores



- Bottleneck: memory
  - Because $\binom{n}{k}$ becomes huge
  - Partwise processing of DUs
  - Keep only DUs of current bounds in Signatures-Datastructure
    - $LowerBound \leq Sig(DU) < HigherBound$

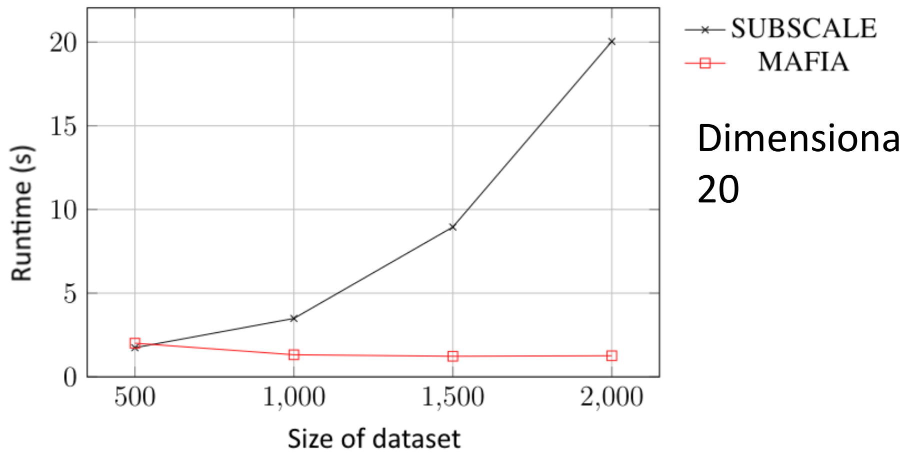
- Parallelization approaches with multicores
  - DUs
  - Bounds
  - Dimensions

Increasing performance

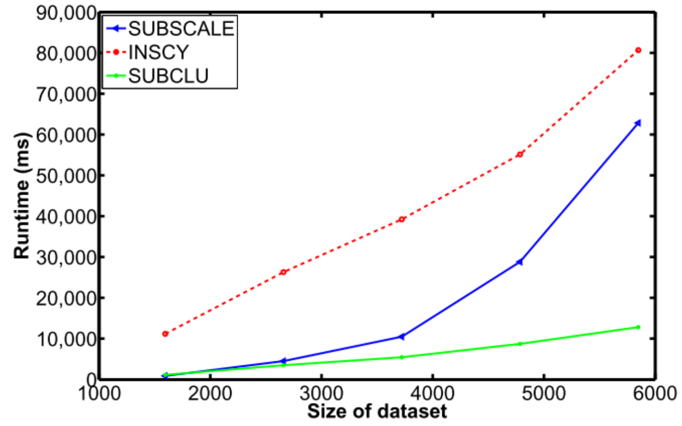


[5]

Comparison

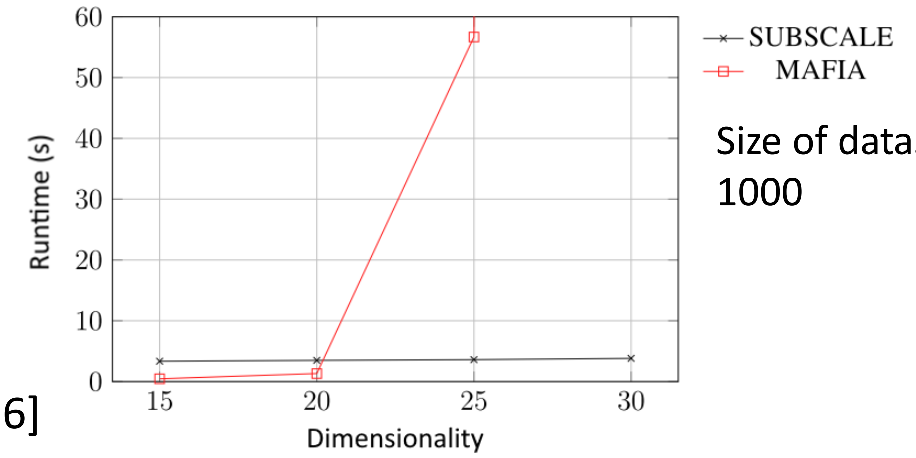


Dimensionality: 20



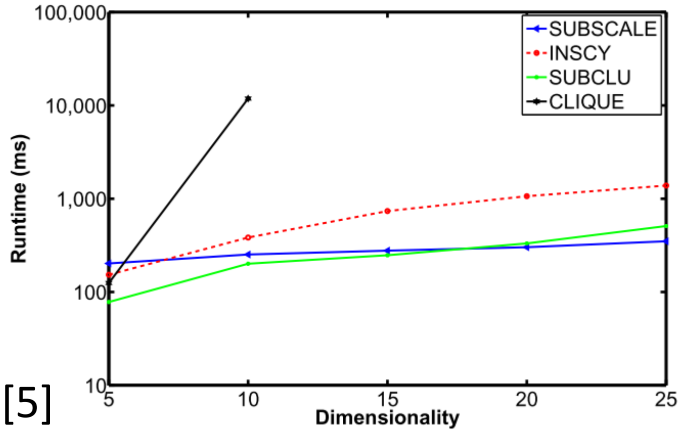
Dimensionality: 20

[6]



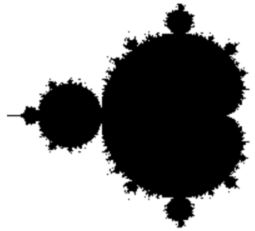
Size of dataset: 1000

[5]



Size of dataset: 1600

Python - The Slowpoke



- Runtime
  - Dataset: 100x500 (rows x dimensions)
- Java – “Subscale Extended”
  - 4,6 [s]
- Python
  - Default data structure for a hash map
    - ca. 19 [s]
    - 4 x slower than Java
  - Shared memory data structure for a hash map
    - ca. 11 [Min]
    - 34 x slower than normal dictionary

→ Use Python as front end

mandelbrot

source	secs	mem
Python 3	172.58	12,216
Java	4.11	68,204

vs.

regex-redux

source	secs	mem
Python 3	**1.36**	112,052
Java	5.70	656,328

Python: my Outlook list

- Multi- / Manycore support
- Performance improvement with specialized libraires
  - Numpy
- Use Python as front end suite for C++ implementation

?

Link to this document:
https://bit.ly/2KXjdnw

Sources

- [0]: https://en.wikipedia.org/wiki/Iris_flower_data_set#/media/File:Principal_tree_for_Iris_data_set.png
- [1]: https://de.wikipedia.org/wiki/Bl%C3%BCte#/media/Datei:Bluete-Schema.svg
- [2]: https://morioh.com/p/eafb28ccf4e3
- [3]: https://wikimedia.org/api/rest_v1/media/math/render/svg/23050fcb53d6083d9e42043bebf2863fa9746043
- [4]: https://de.wikipedia.org/wiki/Datei:2-Dice-Icon.svg
- [5]: Amardeep Kaur (2016). “Fast and Scalable Subspace Clustering of High Dimensional Data”, Perth, Australia, The University of Western Australia
- [6]: Nicolas Kiefer (2020) „Datenparalleles Subspace Clustering mit Grafikprozessoren“, Offenburg, Deutschland, Hochschule Offenburg
- [7]: https://pypl.github.io/PYPL.html
- [8]: https://www.tiobe.com/tiobe-index/