Incremental constrained clustering by minimal weighted modification

Aymeric Beauchamp, Thi-Bich-Hanh Dao,
Samir Loudni, Christel Vrain

May 6th, 2024

CAVIAR doctoral programme

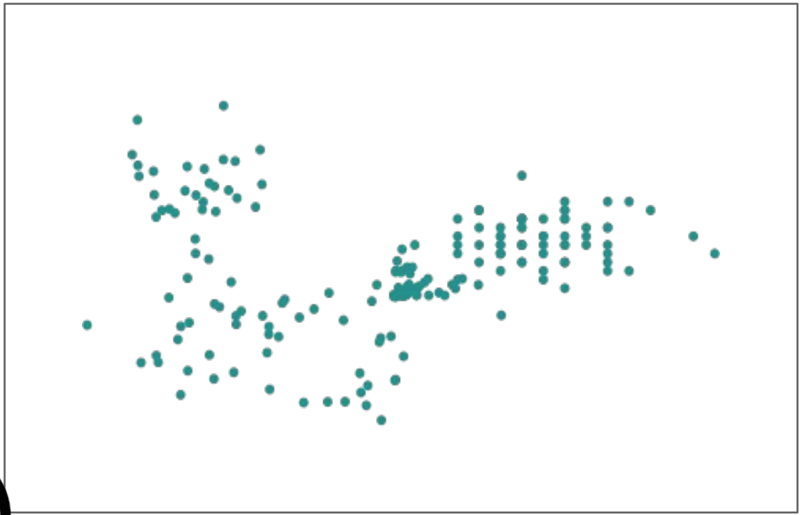
Article published at CP'23

Problem

Clustering is grouping data into clusters as homogeneous and separated as possible

In constrained clustering, users are often expected to provide useful constraints from scratch

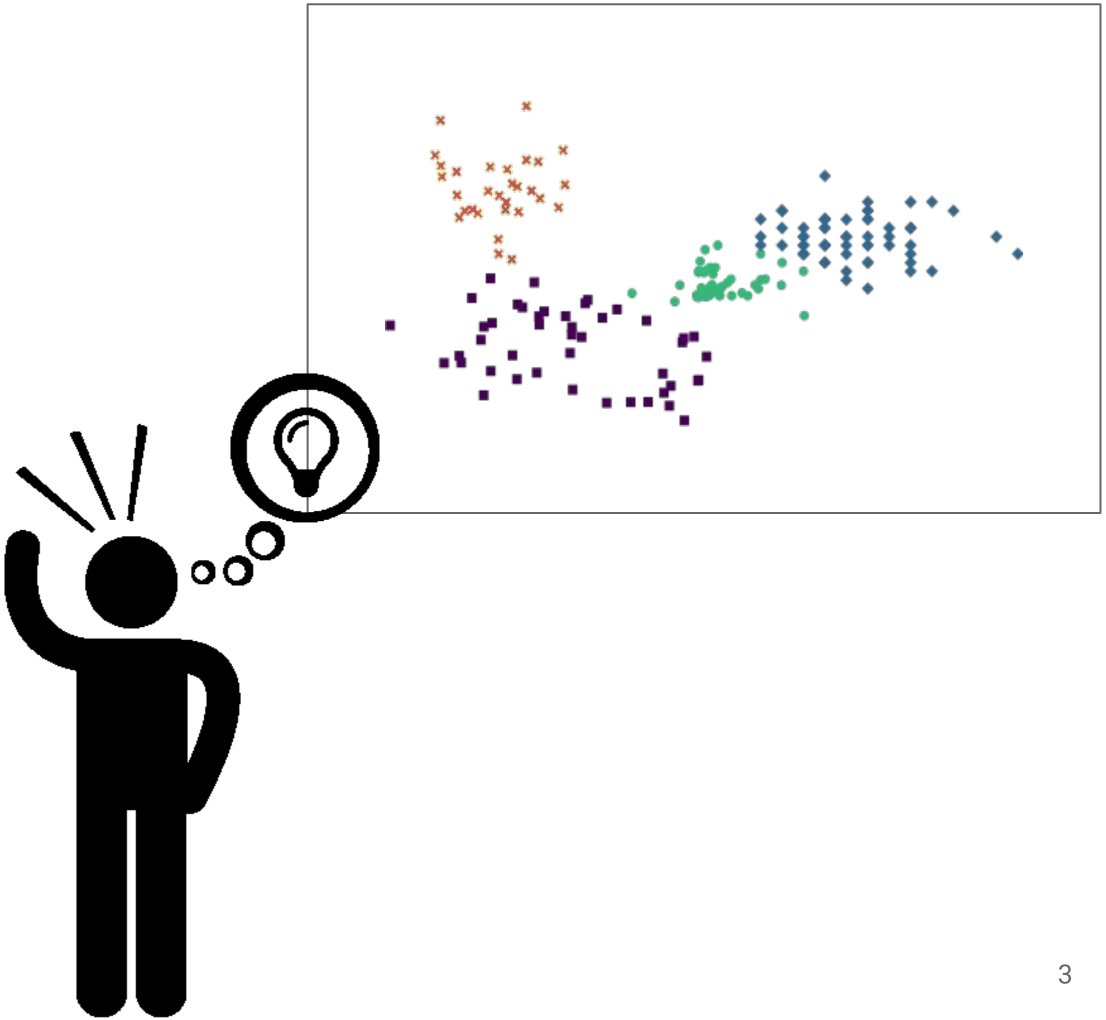
It is not applicable in practice, especially in complex domains of application



Problem

Intuition : making the user react on an existing partition yields better feedback

Iterative approach : the user critiques the partition to improve it step-by-step by adding constraints



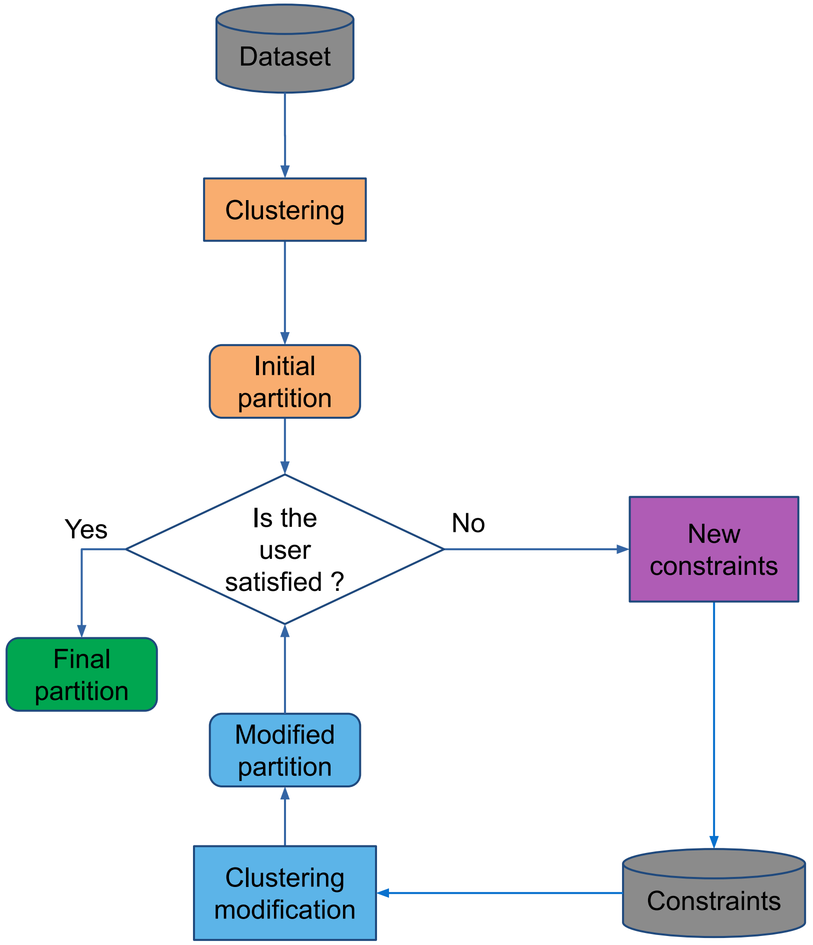
Incremental constrained clustering

1 : make a first clustering

2 : ask for user feedback

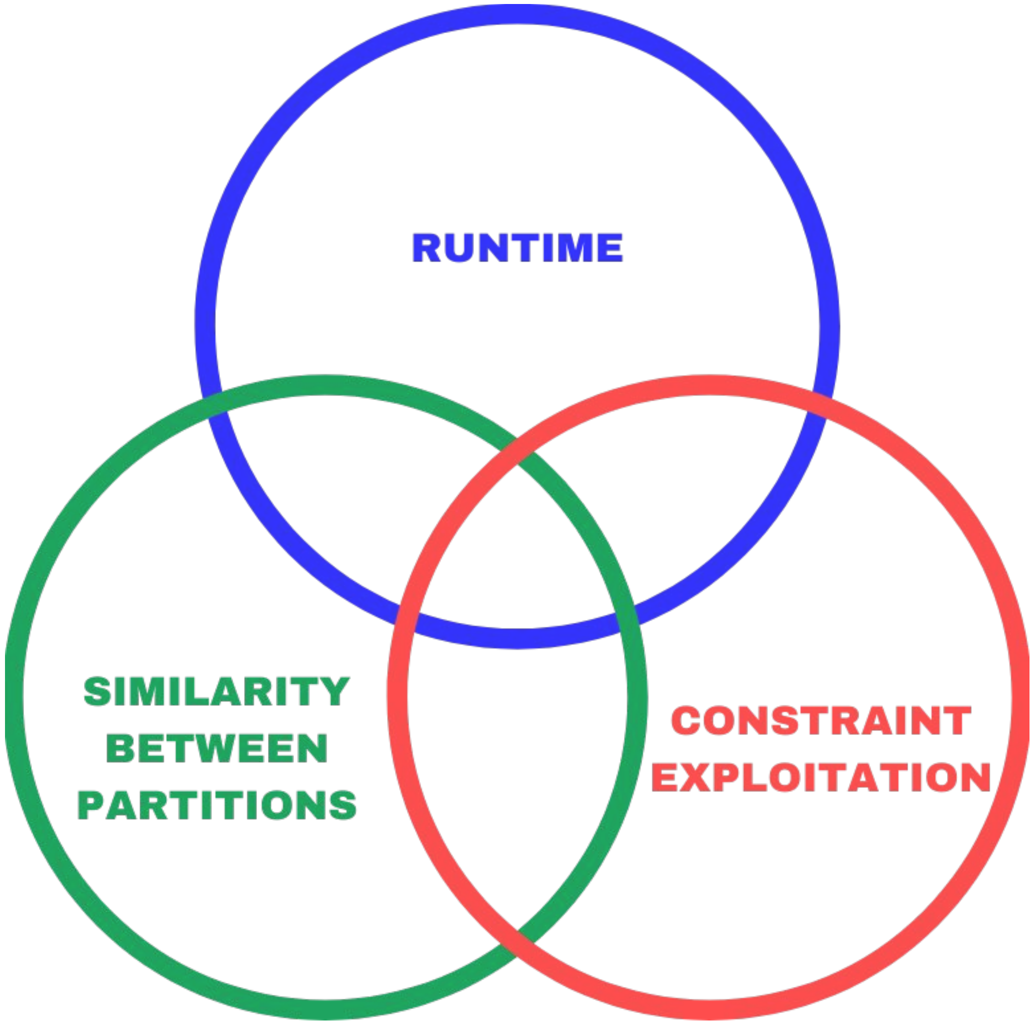
3 : collect user constraints

4 : change the partition



Challenges

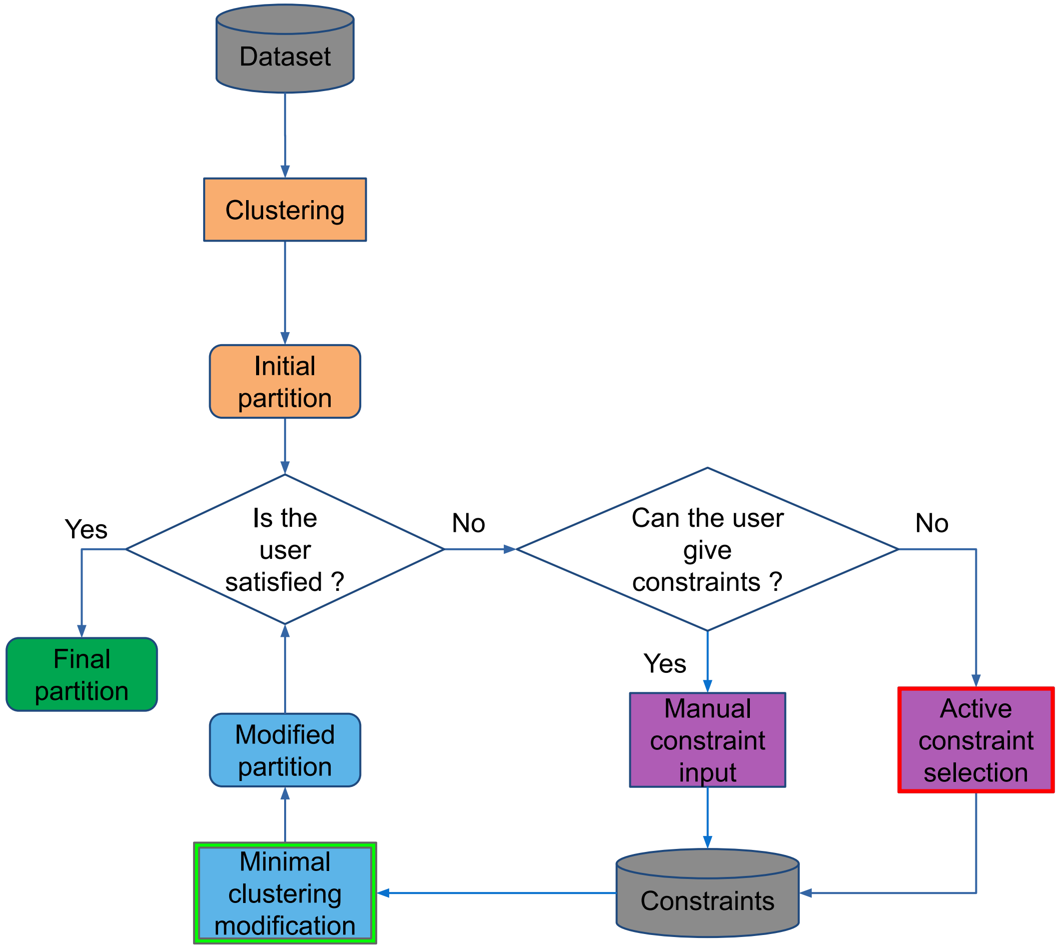
- Runtime to avoid waiting too long
- Constraint exploitation to avoid iterating too much
- Similarity to avoid user confusion



Incremental and active clustering framework (IAC)

Improving similarity by minimal clustering modification

Improving quality by selecting informative constraints



Minimal Weighted Clustering Modification

Input

- a dataset
- a partition of the dataset computed from a classic algorithm (or a result from a previous iteration)
- a set of user constraints

Output

- a partition satisfying all user constraints while being as similar as possible to the input partition

The model only **works on constrained points** instead of reclustering everything, making the model fast

Constraint Programming model

Encoding the new partition :

N variables $X_1, \ldots, X_N$ with domain $\{1, \ldots, k\}$

$X_i = c \Rightarrow$ point i is part of cluster c

Objective function :

$$\arg\min \overbrace{\sum_{i=0}^{N}}^{\text{for each point}} \overbrace{\mathbb{I}(\mathcal{P}[i] \neq X_i)}^{\text{if it's modified}} \overbrace{\mathcal{D}[i, X_i]}^{\text{add this distance}}$$

$D[i,X_i]$ is a N x k distance matrix

The optimal partition is the most similar to the input in the sense that cluster assignments were changed:

- only when necessary
- to the closest acceptable cluster

User constraints

- Must-link/cannot-link : "points *i* and *j* must be grouped together/apart"

$$X_i = X_j \;/\; X_i \neq X_j$$

- Triplet : "*a* is more similar to *p* than to *n*"

$$X_a = X_n \Rightarrow X_a = X_p$$

$$X_a \neq X_p \Rightarrow X_a \neq X_n$$

- Span-limited constraints
- logical combinations of pairwise constraints

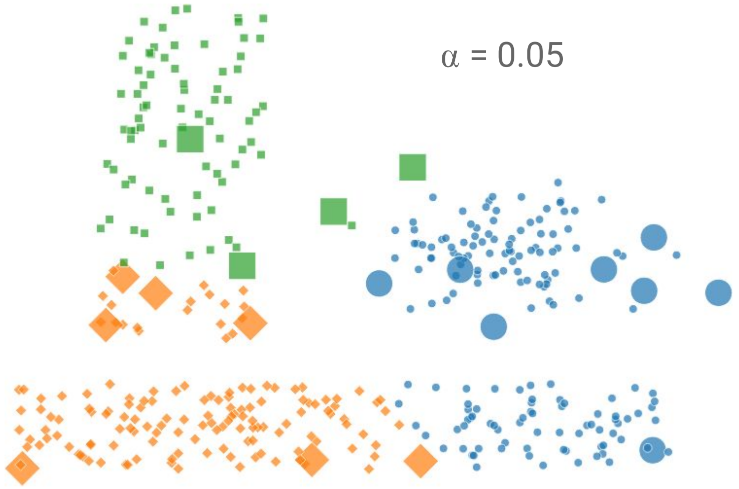
Constraints can be relaxed through reification

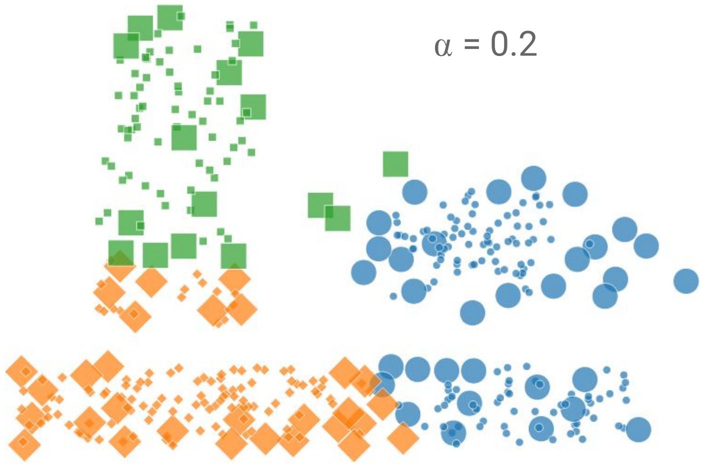
Anchor computation

Anchor set : subset of a cluster denoting its structure

In the objective function, D[i,X_i] is the distance of instance i to the closest anchor of cluster X_i

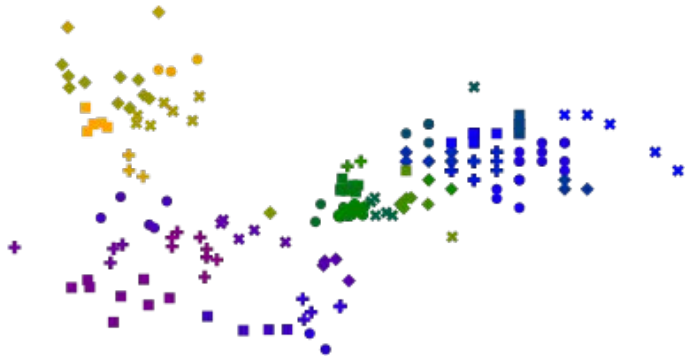
Parameter α : proportion of anchors per cluster (0% means medoids are used)





Propagating modifications

Modifications can be generalized by computing virtual super-points representing multiple real data points

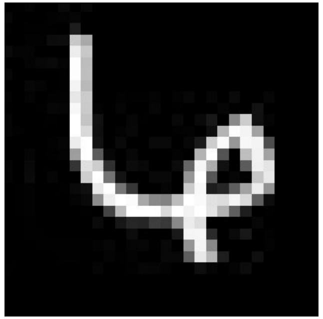


Super-instances are obtained by dividing all clusters into smaller groups using hierarchical clustering and computing the centroids of the groups

Parameter β : proportion of super-instances per cluster (100% means no propagation)

If a super-instance is modified, all points in the super-instance change too

Span-limited constraint



What is this number ?

The group of instances *S* must be affected to a subset of clusters C (1) or to a maximum number γ of clusters (2)

“We don’t know exactly where these instances should belong, but we can rule out some clusters”

$$\begin{array}{ll} count(c, [X_i \mid i \in S], \geq, 0) & \forall\ c \in C \\ count(c, [X_i \mid i \in S], =, 0) & \forall\ c \notin C \end{array} \qquad (1)$$

$$atmost\_nvalue(\gamma, [X_i \mid i \in S]) \qquad (2)$$

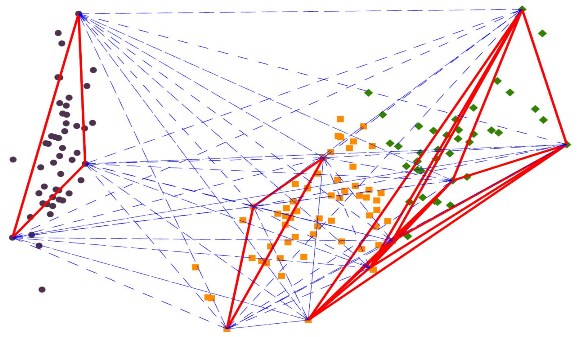
Active constraint selection

Input

- a dataset
- a partition (optional)

Output

- a set of informative constraints



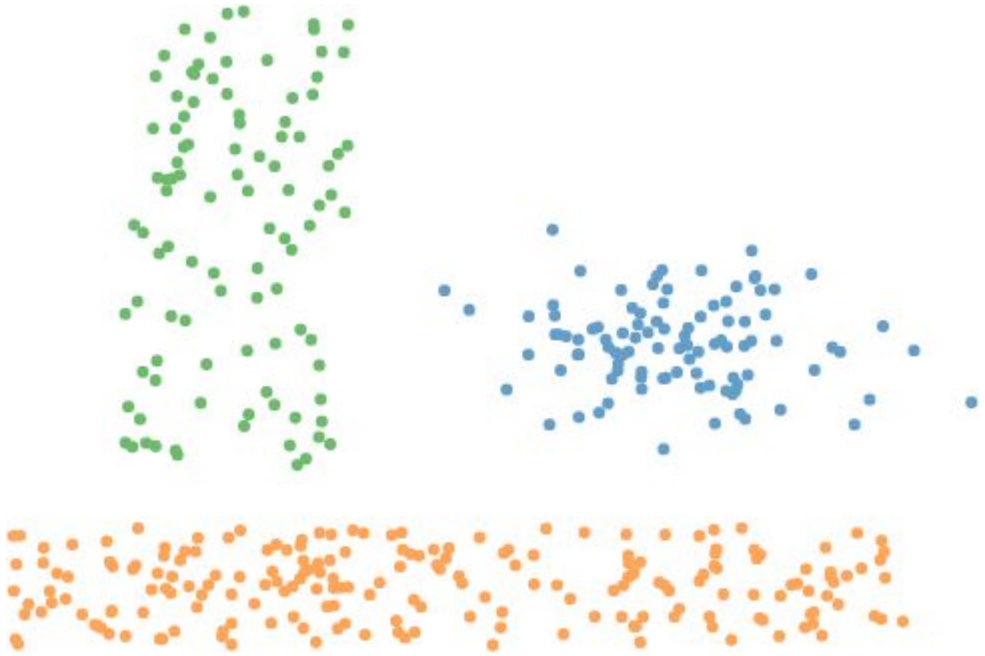
In IAC, active selection is done by a neighborhood-based algorithm called NPU

Outline :

1. pick a random point to start a first neighborhood
2. find the most informative point in the data (one that has equal probability of being in all clusters)
3. query the user about the point to add it to a neighborhood or create a new one
4. repeat steps 2-3 until query budget runs out

Neighborhoods are consolidated over the iterations

Example



Experiments

- CPMpy implementation, using ortools CP-SAT solver

- 16 datasets from 150 to 60000 points
- Oracle : ground truth labeling of the data

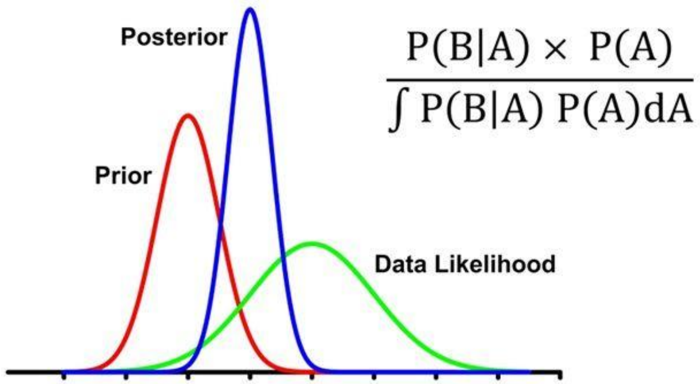
- Metrics : Adjusted Rand Index (ARI), Adjusted Mutual Information (AMI), Folkes-Mallows Index (FMI)
  - Quality of the partition (proximity to ground truth)
  - Similarity between partitions
  - Runtime
- Average over 90 runs

Bayesian analysis

Experimental results are validated using a Bayesian hierarchical inference model to make pairwise comparisons of methods :

- Prior belief is the null hypothesis : “Results are only due to random effects”, i.e. “There is no significant difference in performance between the methods”
- Data likelihood is the experimental results
- Posterior belief is the updated prior when taking into account the experiments

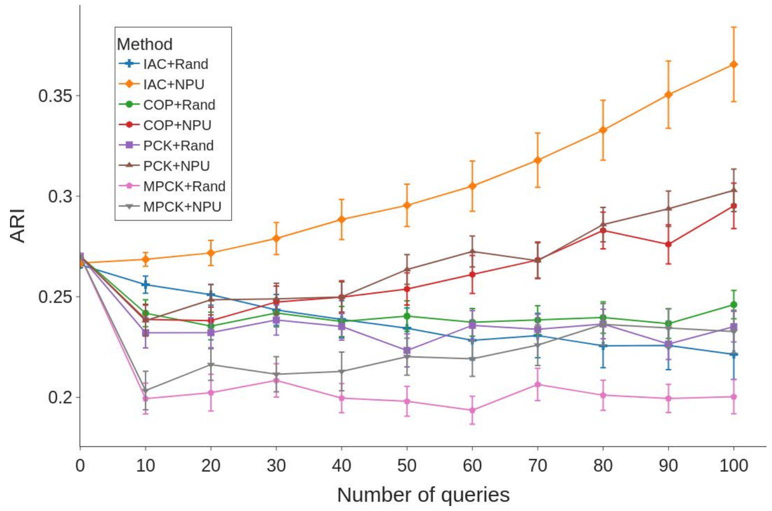
The probability that method A outperforms method B can be computed by sampling the posterior distribution with the Monte Carlo method



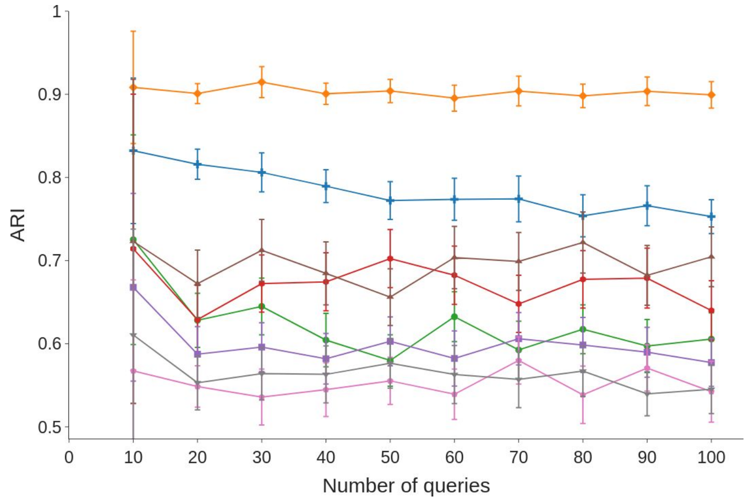
$$\frac{P(B|A) \times P(A)}{\int P(B|A)\,P(A)dA}$$

Area under the budget curve

Task : improve a KMeans partition over 10 iterations, making 10 user queries with active selection and modifying the partition (or reclustering)



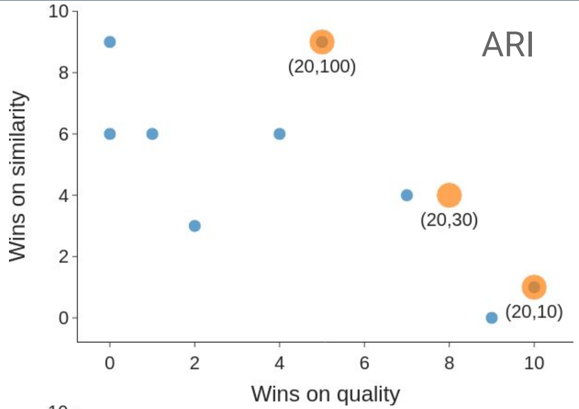
Quality : comparing current partition to ground truth

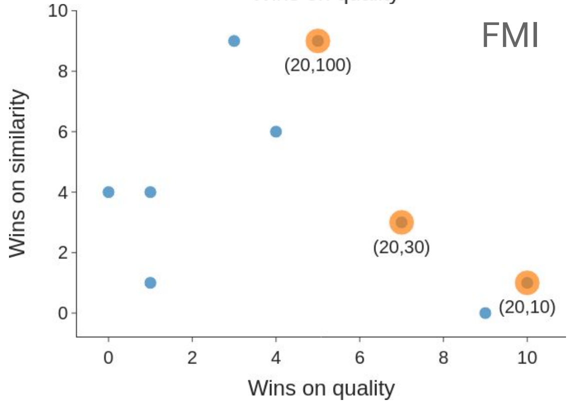


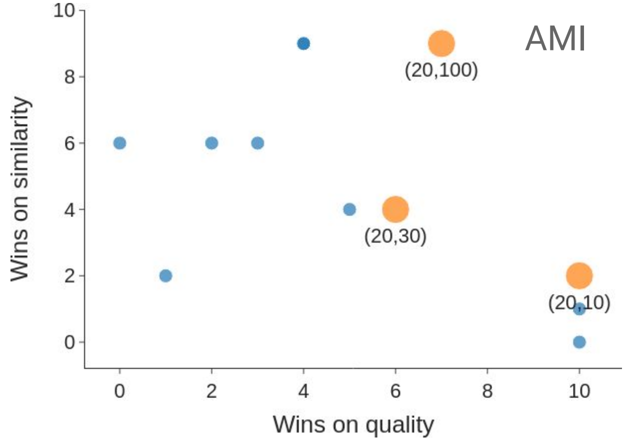
Similarity : comparing current partition to the previous one

Two values for each metric : $AUBC_{quality}$ and $AUBC_{similarity}$ to quantify the whole process

Parameters of IAC



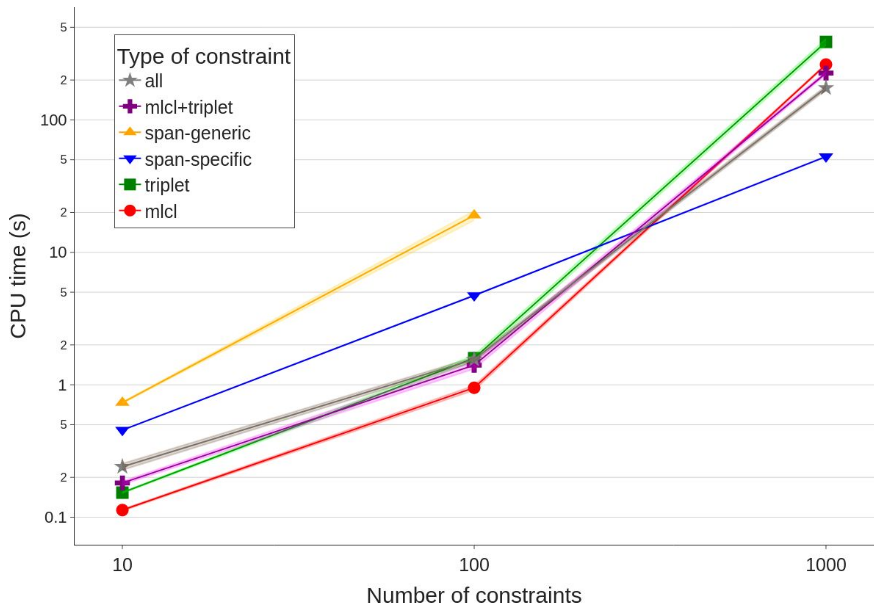




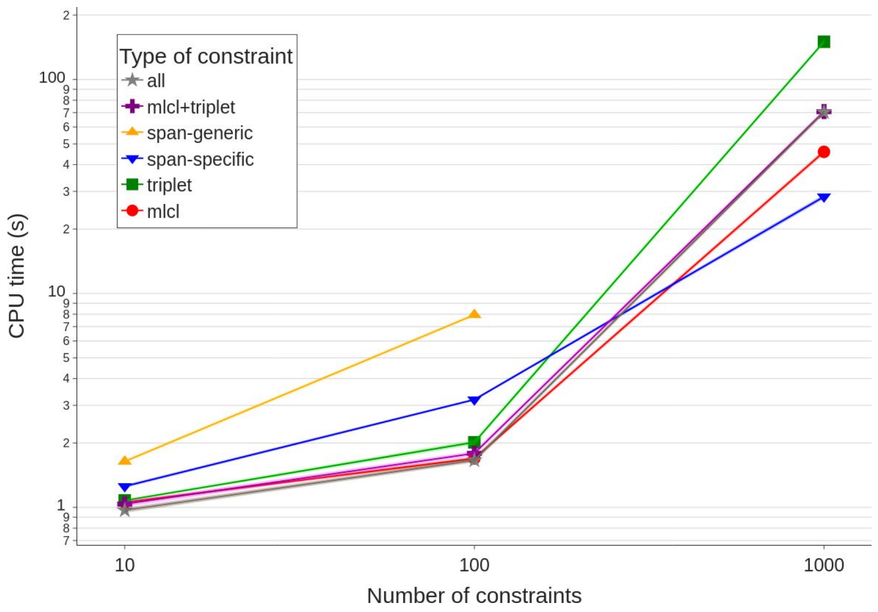
Parameter α : proportion of anchors per cluster (0% means medoids are used) ranging in [0%, 5%, 20%]

Parameter β : proportion of super-instances per cluster (100% means no propagation) ranging in [10%, 30%, 50%, 100%]

Model scaling



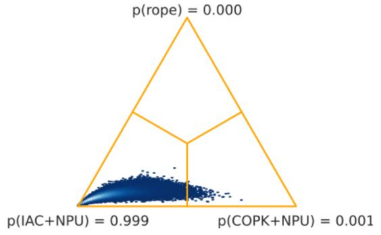
Letters (20k points, 26 clusters)



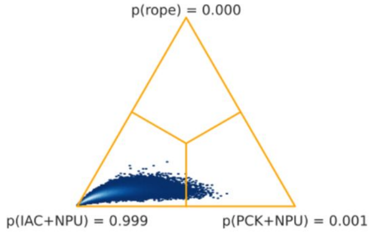
MNIST (60k points, 10 clusters)

Our model finds a solution **within seconds** when given **tens to hundreds** of constraints at once

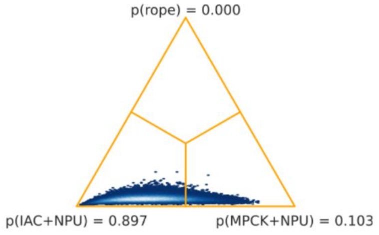
Comparison with state of the art



(a) IAC vs COPK-Means

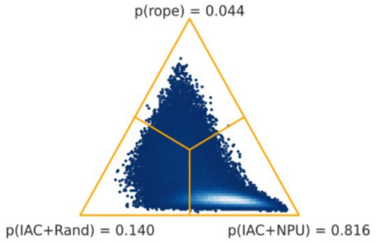


(b) IAC vs. PCK-Means

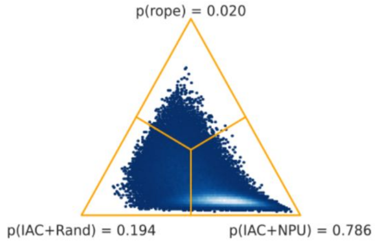


(c) IAC vs. MPCK-Means

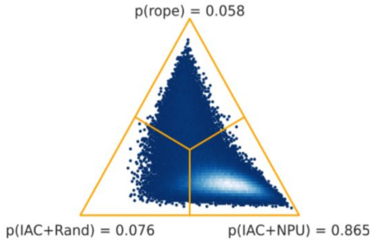
Figure 8 Bayesian comparisons with IAC using NPU w.r.t. $AUBC_{quality}$ values for ARI.



(a) ARI



(b) AMI



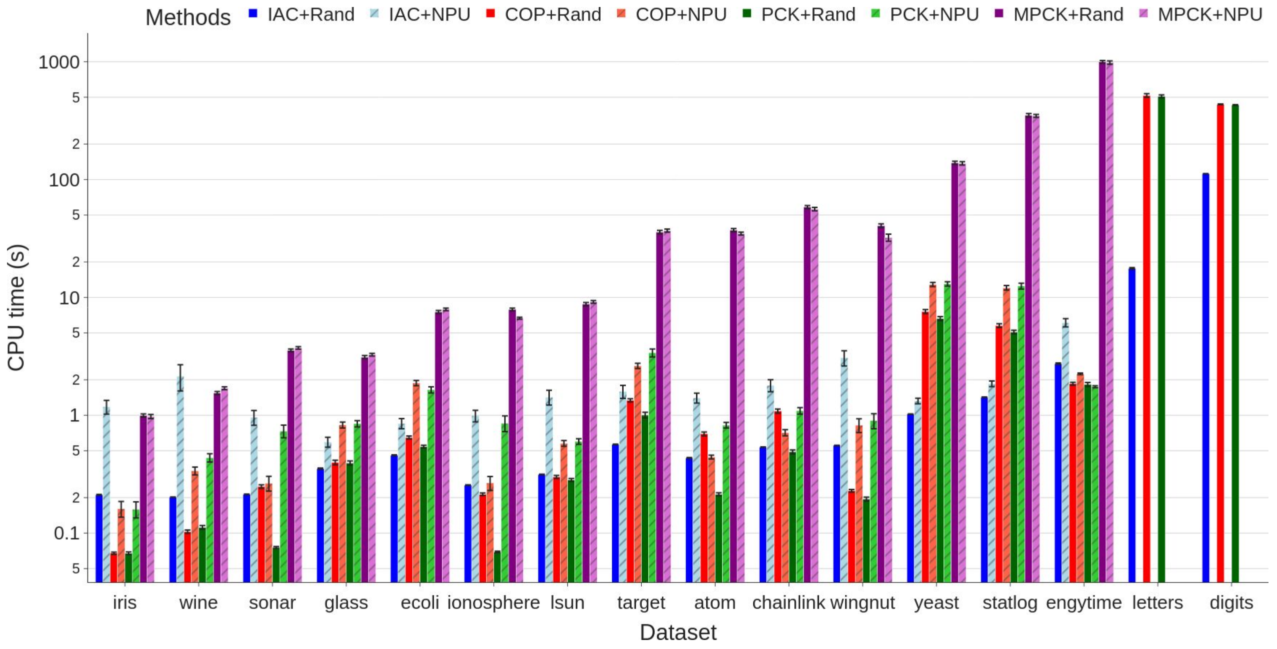
(c) FMI

Figure 9 Bayesian comparison of IAC with or without NPU w.r.t $AUBC_{similarity}$ for all metrics.

Bayesian analysis shows high probability (89% to 99%) that IAC outperforms other methods on both quality and similarity

Using NPU has a positive effect on similarity on top of improving quality

Comparison of runtime with state of the art



IAC has comparable runtime and better scaling on the largest datasets used
Active clustering is a limiting factor

Application : remote sensing

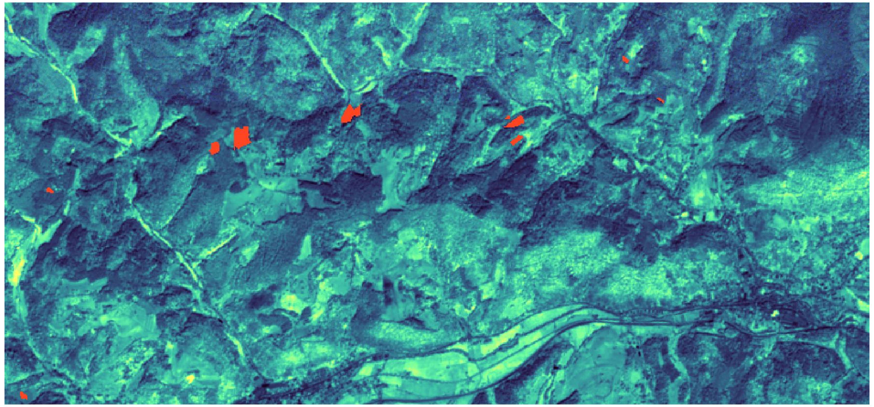
- Application within an ANR research project
- Assisting experts in detecting temporal phenomena in satellite image time series
- Relevant evolution patterns are hard to find without help from a domain expert

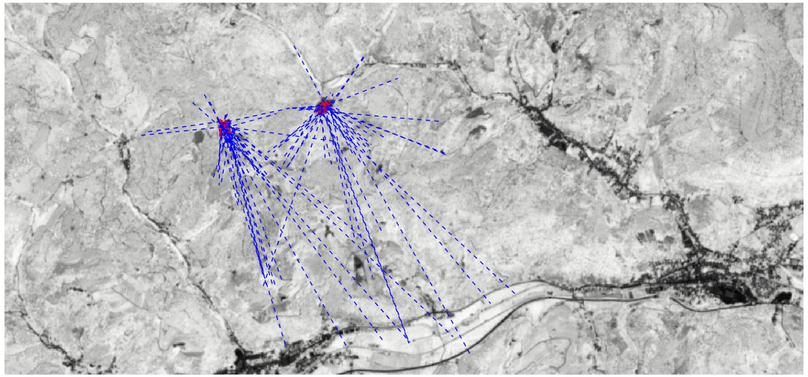
Hérelles

Un cadre collaboratif unifié pour l’analyse interactive de données temporelles

Application : remote sensing

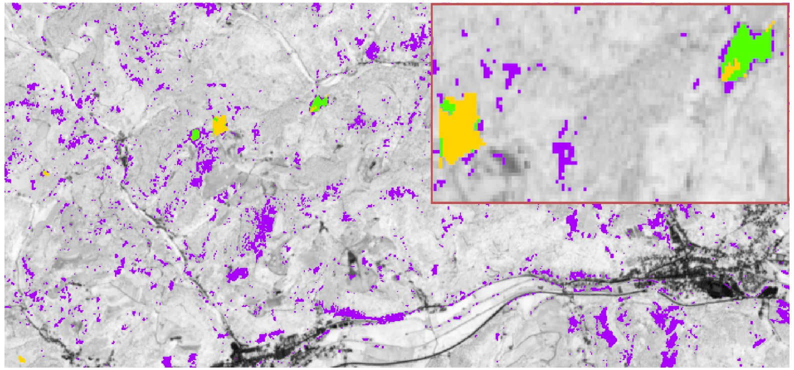
- SITS with 11 timestamps
- 243 000 time series
- Tree cut cluster in red



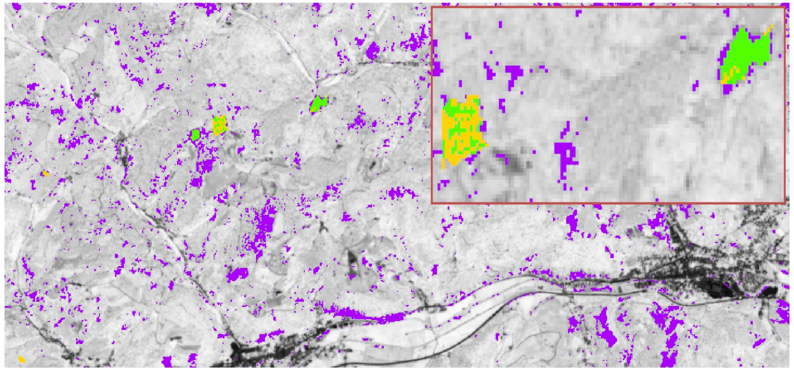


Pairwise constraints from human annotators (must-link and cannot-link)

Application : remote sensing



Objective : recover all tree cut pixels in the same cluster



- Half of the left zone recovered within 20 seconds
- Unable to generalize changes due to runtime

Conclusion

- A framework for incremental constrained clustering
- A model for minimal clustering modification
- Can handle various instance-level and group-level constraints
- Better scaling, faster convergence to ground truth and better similarity than classical constrained clustering

Perspectives :

- Improve generalization
- Develop an active query strategy suited for the incremental clustering setting
- Integrate high-level thematic knowledge

Thank you for your attention !

NPU

Algorithm 2 Incremental NPU

```
Input : Dataset $\mathcal{D}$, partition $\mathcal{P}$, oracle
Output : constraint set $\mathcal{C}$
1: $\mathcal{C} \leftarrow \emptyset$ ; $l \leftarrow 1$ ; $\mathcal{N} \leftarrow N_1 \mid N_1 = \{random(\mathcal{D})\}$
2: $x^* \leftarrow MostInformative(\mathcal{D}, \mathcal{P}, \mathcal{N})$
3: for each $N_i \in \mathcal{N}$ in decreasing order of $P(x^* \in N_i)$ do
4:     Query $x^*$ against any $x_i \in N_i$ to the oracle
5:     if $(x^*, x_i, ML)$ then
6:         $\mathcal{C} \leftarrow (x^*, x_i, ML)$
7:         $N_i = N_i \cup x^*$
8:         break
9:     else
10:        $\mathcal{C} \leftarrow (x^*, x_i, CL)$
11: if no ML is returned then
12:     $l++$ ; $N_l = x^*$ ; $\mathcal{N} \leftarrow \mathcal{N} \cup N_l$
return $\mathcal{C}$
```

Algorithm 3 MostInformative

```
Input : Dataset $\mathcal{D}$, partition $\mathcal{P}$, set of neighborhoods $\mathcal{N}$
Output : most informative data point $x^*$
1: Learn a random forest classifier using $\mathcal{P}$ as labels
2: Compute the similarity matrix $M$ s.t. $M[i, j]$ is the number of leaves where $i$ and $j$ are
   together normalized by the number of trees of the RF
3: for each $x \in \mathcal{U} = \mathcal{D} \setminus \mathcal{N}$ do
4:     for $i = 1$ to $l$ do
5:         $p(x \in N_i) = \frac{\frac{1}{|N_i|} \sum_{x_j \in N_i} M(x, x_j)}{\sum_{p=1}^{l} \frac{1}{|N_p|} \sum_{x_j \in N_p} M(x, x_j)}$
6:     $H(\mathcal{N}|x) = -\sum_{i=1}^{l} p(x \in N_i) \log_2 p(x \in N_i)$
7:     $E(x) = \sum_{i=1}^{l} i * p(x \in N_i)$
return $\arg\max_{x \in \mathcal{U}} \frac{H(\mathcal{N}|x)}{E(x)}$
```