Methods for assessing clustering stability in single-cell expression datasets [1]

Andi Munteanu [1]

Supervisors
Dr. Liviu Ciortuz [1]
Dr. Irina Mohorianu [2]

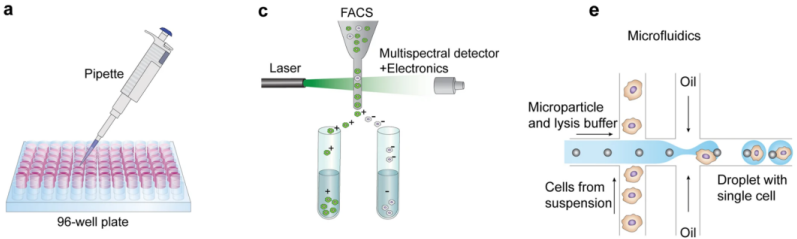
[1]Faculty of Computer Science, Alexandru Ioan Cuza University, Iași
[2]Wellcome-MRC Cambridge Stem Cell Institute, University of Cambridge

June 2022

Outline

Table of Contents

Single-cell data processing



	gene #1	gene #2	gene #3	gene #4	gene #5	gene #6	gene #7	gene #8	gene #9
cell #1	0	15	20	3	0	0	0	5	7
cell #2	7	4	0	0	25	21	4	0	0
cell #3	0	0	23	5	15	0	0	0	0

Figure: **Single-cell data preprocessing** Image source: Single-cell RNA sequencing technologies and bioinformatics pipelines, Hwang, Lee & Bang, 2018

Table of Contents

PhenoGraph pipeline

PhenoGraph: pipeline proposed by Levine et al. [2] that applies graph clustering on gene-expression datasets.

Consists of three steps:

- dimensionality reduction

PhenoGraph pipeline

PhenoGraph: pipeline proposed by Levine et al. [2] that applies graph clustering on gene-expression datasets.

Consists of three steps:

- dimensionality reduction
- graph construction

PhenoGraph pipeline

PhenoGraph: pipeline proposed by Levine et al. [2] that applies graph clustering on gene-expression datasets.

Consists of three steps:

- dimensionality reduction
- graph construction
- graph clustering

Divergent results

We compared the partitions inferred by two frequently used R packages, Seurat [3] and Monocle [4], that implement the PhenoGraph pipeline to process single-cell data.

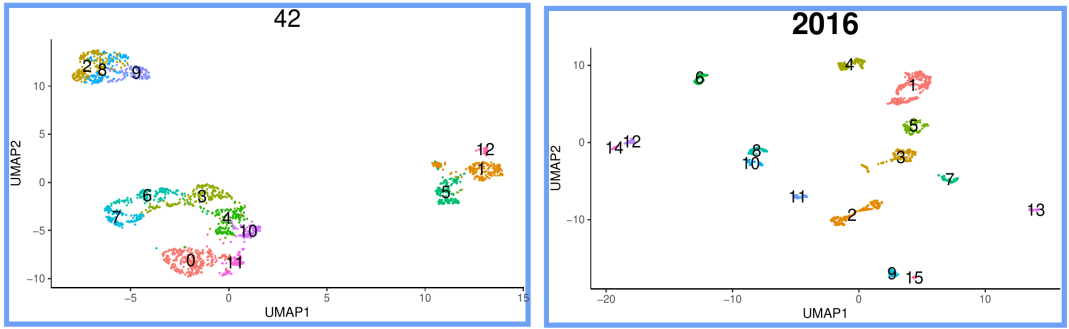


Figure: **Partitions obtained on default values of the parameters** Clustering with Monocle (right) and Seurat (left). The title indicates the default random seed that was used.

Aligning the results

Parameter name	Step	Monocle value	Seurat value
seed	-	2016	42
feature set	Dim reduction	all genes	HV genes
UMAP min dist	Dim reduction	0.1	0.3
UMAP n neighbours	Dim reduction	15	30
base embedding	Graph building	UMAP	PCA
SNN implementation	Graph building	no self-neighbours only direct neighbours	with self-neighbours direct and indirect neighbours
graph type	Graph building	unweighted	weighted
clustering method	Graph clustering	Leiden	Louvain
quality function	Graph clustering	CPM	RBConfiguration
resolution	Graph clustering	1e-4	0.8
#iterations	Graph clustering	2	10

Table: **Parameters used for aligning the results of Monocle and Seurat**

Table of Contents

Table of Contents

What is ECS?

- An *affinity matrix* of a partition summaries the co-occurrences of elements by calculating the number of different paths between them.
- For a disjoint partition, the affinity matrix is calculated using the formula:

$$p_{ij} = \begin{cases} 0, \text{if } i \text{ and } j \text{ don't belong to the same cluster} \\ \frac{\alpha}{|C_\beta|}, \text{if } i \text{ and } j \text{ belong to the cluster } \beta \\ 1 - \alpha + \frac{\alpha}{|C_\beta|}, \text{if } i = j \end{cases}$$

- Element-Centric Similarity (ECS) [5] is a clustering comparison score built on the L1 distance between the affinity matrices: $S_i(\mathcal{A}, \mathcal{B}) = 1 - \frac{1}{2\alpha} \sum_{j=1}^{N} |p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}}|$

ECS properties

Optimising the calculation of the ECS score for disjoint partitions

- For disjoint partitions, the ECS has the same value for points from the same clusters.
- The number of ECS calculations drops from $N \times N$ to $n \times m$, where n is the number of clusters of the first partition and m is the number of clusters of the second.
- This optimization removes the dependency on the affinity matrix and enhances the time and memory efficiency.

▸ Optimization proof

Element-Centric Consistency

- Gates et al. also proposed Element-Centric Consistency (or ECC; also named *frustration*) score, used on a list of multiple partitions.
- The ECC score is calculated as the average of the sum of the ECSs obtained for every unique pair of partitions from the list:

$$\frac{2}{T(T-1)} \sum_{j=1}^{T} \sum_{k=1}^{j-1} S_i(\mathcal{R}_k, \mathcal{R}_j)$$

Weighted ECC

- For multiple occurrences of the same partition, the ECC requires redundant calculations of the ECS of equivalent pairs.
- To alleviate the computational burden, we proposed a weighted version of ECC, where for each unique partition we retain a weight that is the number of duplicates.

Weighted ECC pseudocode

Merging identical partitions

The weights are calculated by performing an automatic merge of identical partitions.

To determine if two partitions are identical, we use their contingency table.

Figure: **Pair-wise contingency table**. The rows are associated with the clusters from one partition, and the columns with the clusters from the other partitions. In each $[i, j]$ cell we present the number of cells in clusters i and j, respectively

Almost identical partitions - ECS threshold

There are cases where the difference between two partitions is negligible (see below). To allow the merging of almost identical partitions, we introduced the *ECS threhshold* i.e. a lower bound for the ECS for determining partitions which can be considered quasi identical and merge-able.

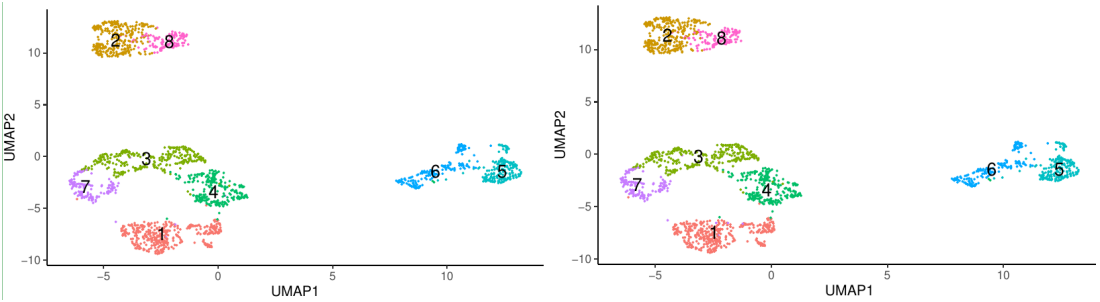


Figure: **Two almost identical partitions** Each panel represents the distribution of a partition on a low-dimensional (UMAP) topology.

Table of Contents

- Random seed is a factor that affects the clustering output.
- We developed a stability pipeline that offers a set of visual summaries for the assessment of the robustness of a parameter configuration in the context of variable random seeds.
- The robustness is determined by running the clustering pipeline multiple times on different random seeds. The Element-Centric Consistency of the obtained list of partitions is an indicator of stability.
- The stability pipeline follows the steps presented in the PhenoGraph algorithm.

Feature stability

The feature set has an important role in obtaining the low-dimensional topology.

We assess the stability of different feature sets with varying sizes.

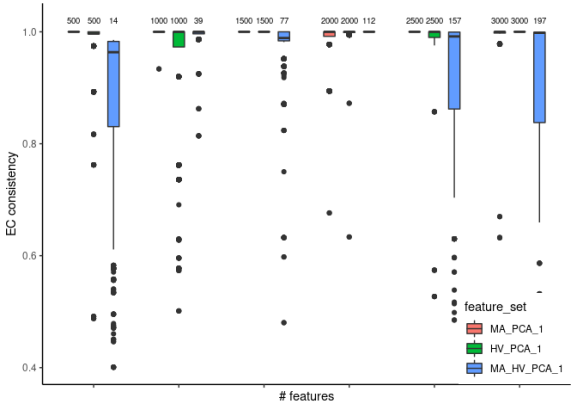


Figure: **Incremental stability of feature sets**. Each colour corresponds to a different feature set. The size of the set is displayed above each boxplot. Each boxplot represents the ECS distribution across partitions.

The impact of the number of nearest neighbours on the graph connectivity

The number of nearest neighbours affects the graph connectivity.

The number of connected components decreases as the number of nearest neighbours increases.

The number of connected components is a lower bound for the possible number of clusters.

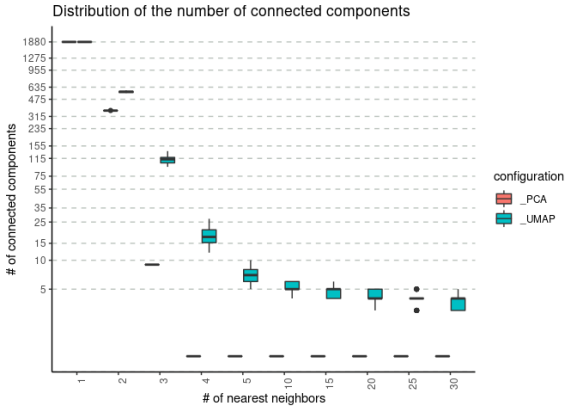


Figure: **Graph connectivity assessment**. The colour indicates the reduced space used for graph building. On the X-axis we vary the number of nearest neighbours. The Y-axis indicates the number of connected components.

Resolution - number of clusters stability

For graph clustering methods, the resolution parameter controls the number of clusters.

The stability can be assessed on different parameter configurations.

The colour gradient describes the statistical reliability of the assessment.

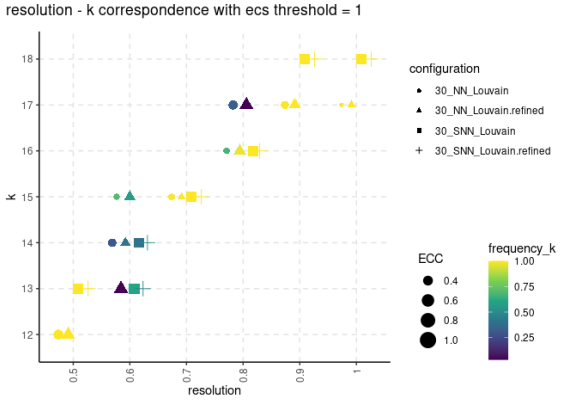


Figure: **Stability link between the Resolution and the number of clusters**. Each point-shape indicates a parameter configuration. The point size quantifies the stability of the resolution - number of clusters pair. The colour gradient illustrates the frequency of the number of clusters.

Table of Contents

Table of Contents

- To benchmark the performance of the `ClustAssess` package, we compare it to `Clusim`.

- To benchmark the performance of the `ClustAssess` package, we compare it to `Clusim`.
- `Clusim` is a Python package that contains the official implementation from the authors of the ECS paper [6].

- To benchmark the performance of the `ClustAssess` package, we compare it to `Clusim`.
- `Clusim` is a Python package that contains the official implementation from the authors of the ECS paper [6].
- In `Clusim`, the ECS between two disjoint partitions is determined based on the affinity matrices.

- To benchmark the performance of the `ClustAssess` package, we compare it to `Clusim`.
- `Clusim` is a Python package that contains the official implementation from the authors of the ECS paper [6].
- In `Clusim`, the ECS between two disjoint partitions is determined based on the affinity matrices.
- The benchmark relies on the ECS between two fixed partitions using the implementations from `ClustAssess` and `Clusim`. The size of the partitions varies from 50 to 90 000. The number of clusters is set to 20 for both partitions. The assessment is repeated 30 times.

Runtime comparison

`ClustAssess` requires less than a second to calculate the ECS, while the execution time of `Clusim` increases exponentially.

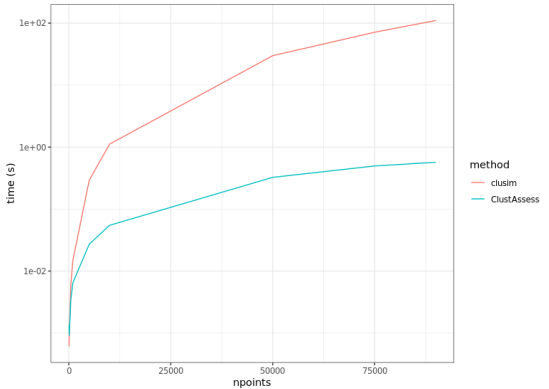


Figure: **Execution time benchmark** The size of the partitions used for the benchmark are displayed on the X-axis. The time (in seconds) required to calculate the ECS using ClustAssess (blue) and Clusim (red) is displayed on the Y-axis. The results are displayed on a logarithmic scale.

Comparison of memory usage

For the optimized version from `ClustAssess` the data storage scales proportionally to the number of clusters.

Memory allocation of $\leq$ 100 MiBs.

The storage of the whole affinity matrix in `Clusim` is inefficient, reaching a memory allocation of 241 GiBs for the largest dataset.

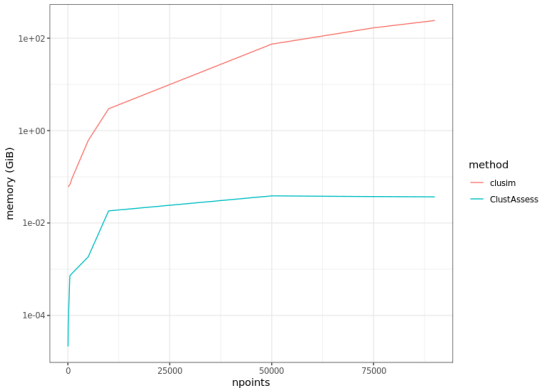


Figure: **Memory usage benchmark** The size of the partitions used for the benchmark are displayed on the X-axis. The amount of memory (in GiBs) required to calculate the ECS using ClustAssess (blue) and Clusim (red) is displayed on the Y-axis. The results are displayed on a logarithmic scale.

Performance with varying number of clusters

- The time complexity for calculating the ECS in `ClustAssess` is $O(nm)$, where n and m represent the number of clusters of both partitions.
- As the number of clusters increases, the `ClustAssess` package will require more time and space to calculate the ECS.
- `Clusim` calculates the whole affinity matrix, so its performance is not influenced by the number of clusters.

Performance with varying number of clusters

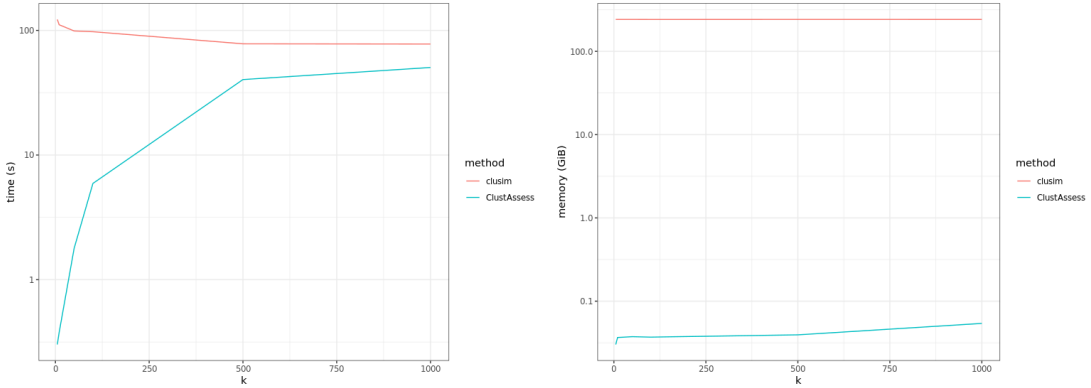


Figure: **Performance comparison for different number of clusters** Left panel - the median time of execution (in seconds) measured after 30 runs for ClustAssess (blue) and Clusim (red). Right panel - the median memory usage (in GiB) measured after 30 runs for ClustAssess (blue) and Clusim (red). The results are displayed on a linear scale.

Table of Contents

How the stability pipeline will be benchmarked

- We evaluated the performance of the five independent components: `feature_stability`, `nn_n_conn_comps`, `nn_importance`, `clustering_importance`, `resolution_importance`.

How the stability pipeline will be benchmarked

- We evaluated the performance of the five independent components: `feature_stability`, `nn_n_conn_comps`, `nn_importance`, `clustering_importance`, `resolution_importance`.
- The stability pipeline was evaluated on the *Mende et al* data [7] with subsamples ranging from 1000 to 13000 cells.

How the stability pipeline will be benchmarked

- We evaluated the performance of the five independent components: `feature_stability`, `nn_n_conn_comps`, `nn_importance`, `clustering_importance`, `resolution_importance`.
- The stability pipeline was evaluated on the *Mende et al* data [7] with subsamples ranging from 1000 to 13000 cells.
- The pipeline allows support for parallelisation, i.e. we measured the pipeline's performance by varying the number of cores.

How the stability pipeline will be benchmarked

- We evaluated the performance of the five independent components: `feature_stability`, `nn_n_conn_comps`, `nn_importance`, `clustering_importance`, `resolution_importance`.
- The stability pipeline was evaluated on the *Mende et al* data [7] with subsamples ranging from 1000 to 13000 cells.
- The pipeline allows support for parallelisation, i.e. we measured the pipeline's performance by varying the number of cores.
- We also evaluated the impact of using different ECS threshold values: 1, 0.99 and 0.95.

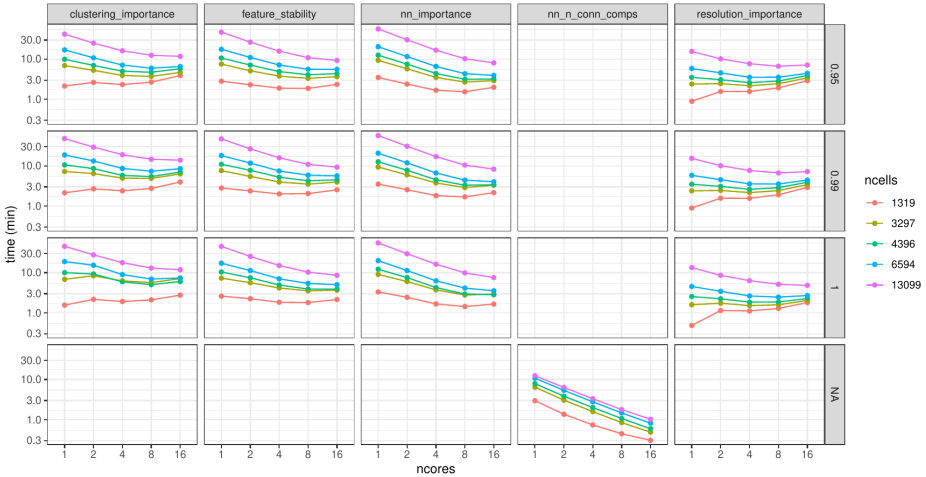


Figure: **Stability pipeline runtime benchmark** The columns correspond to different components of the pipeline. The rows indicate different ECS threshold values. On the X-axis we represent different number of cores. The Y-axis contains the runtime (in minutes). The colour indicates the size of the dataset.

Table of Contents

Conclusions

- The time complexity of ECS calculation between two disjoint partitions has been improved from $\mathcal{O}(N^2)$ to $\mathcal{O}(nm)$.
- The calculation of ECC is also improved by merging (almost) identical partitions and by using weights.
- The output of the clustering pipeline is affected by the change of random seed.
- We proposed a stability pipeline which provides visual tools to assess the robustness at the change of random seed of different configurations.

Table of Contents

Limitation and Future work

Limitations:

- Our analysis was performed on data processed using only two sequencing techniques, SMART seq and 10x.
- The results were obtained on relatively small datasets with up to 13000 cells.
- The benchmarks were performed only on crisp partitions.

Future work:

- Further improvement of the ECS calculation.
- Optimization of ECS calculation for the overlapping and hierarchical cases.
- Test the scalability of the stability pipeline on bigger datasets.
- Optimization of the merging of almost identical partitions.
- Finding a correlation between the maximum number of disagreements and the ECS value.

Table of Contents

Acknowledgements

The thesis is based on the ClustAssess paper to which I contributed as co-author.
The paper was presented at CSCI 2022 annual retreat and will be presented at ECCB 2022.

References I

[1] Arash Shahsavari, Andi Munteanu, and Irina Mohorianu. Clustassess: tools for assessing the robustness of single-cell clustering. *bioRxiv*, 2022.

[2] Jacob H. Levine, Erin F. Simonds, Sean C. Bendall, Kara L. Davis, El-ad D. Amir, Michelle D. Tadmor, Oren Litvin, Harris G. Fienberg, Astraea Jager, Eli R. Zunder, Rachel Finck, Amanda L. Gedman, Ina Radtke, James R. Downing, Dana Pe'er, and Garry P. Nolan. Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis. *Cell*, 162(1):184–197, July 2015.

[3] Yuhan Hao, Stephanie Hao, Erica Andersen-Nissen, William M. Mauck, Shiwei Zheng, Andrew Butler, Maddie J. Lee, Aaron J. Wilk, Charlotte Darby, Michael Zager, Paul Hoffman, Marlon Stoeckius, Efthymia Papalexi, Eleni P. Mimitou, Jaison Jain, Avi Srivastava, Tim Stuart, Lamar M. Fleming, Bertrand Yeung, Angela J. Rogers, Juliana M. McElrath, Catherine A. Blish, Raphael Gottardo, Peter Smibert, and Rahul Satija. Integrated analysis of multimodal single-cell data. *Cell*, 184(13):3573–3587.e29, June 2021.

References II

[4] Junyue Cao, Malte Spielmann, Xiaojie Qiu, Xingfan Huang, Daniel M. Ibrahim, Andrew J. Hill, Fan Zhang, Stefan Mundlos, Lena Christiansen, Frank J. Steemers, Cole Trapnell, and Jay Shendure. The single-cell transcriptional landscape of mammalian organogenesis. *Nature*, 566(7745):496–502, February 2019.

[5] Alexander J. Gates, Ian B. Wood, William P. Hetrick, and Yong-Yeol Ahn. Element-centric clustering comparison unifies overlaps and hierarchy. *Scientific Reports*, 9(1):8574, December 2019.

[6] Alexander J. Gates and Yong-Yeol Ahn. Clusim: a python package for calculating clustering similarity. *Journal of Open Source Software*, 4(35):1264, 2019.

References III

[7] Nicole Mende, Hugo P Bastos, Antonella Santoro, Krishnaa T Mahbubani, Valerio Ciaurro, Emily Francesca Calderbank, Mariana Quiroga Londoño, Kendig Sham, Giovanna Mantica, Tatsuya Morishima, Emily Mitchell, Maria Rosa Lidonnici, Fabienne Meier-Abt, Daniel Hayler, Laura Jardine, Abbie Curd, Muzlifah Haniffa, Giuliana Ferrari, Hitoshi Takizawa, Nicola K Wilson, Bertie Gottgens, Kourosh Saeb-Parsy, Mattia Frontini, and Elisa Laurenti, PhD. Unique molecular and functional features of extramedullary hematopoietic stem and progenitor cell reservoirs in humans. *Blood*, January 2022.

ECS properties

ECS has multiple properties:

- it has no bias towards randomized membership, cluster size or number of clusters
- ECS can be calculated for disjoint, overlapping and hierarchical partitions
- compared to most metrics, ECS provides a similarity score for each element; this allows both global and local assessment of the similarity.

Jump to ECS slide

Proof for ECS optimization

Remark

$$\mathcal{S}_i(\mathcal{A},\mathcal{B}) = 1 - \frac{1}{2}\left(|C_a^{\mathcal{A}} \cap C_b^{\mathcal{B}}| \cdot \left|\frac{1}{c_a} - \frac{1}{c_b}\right| + |C_a^{\mathcal{A}} \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})| \cdot \frac{1}{c_a} + |(\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap C_b^{\mathcal{B}}| \cdot \frac{1}{c_b}\right),$$

for any $i \in C_a^{\mathcal{A}} \cap C_b^{\mathcal{B}}$.

Proof

In order to calculate the Element-Centric Similarity between partitions $\mathcal{A}$ and $\mathcal{B}$ in the point i, we must use the simplified formula for defining the affinity matrix.
To define $p_{ij}^{\mathcal{A}}$ and $p_{ij}^{\mathcal{B}}$, we must explore all four cases of labeling the point j.

Back to Optimization slide

Proof (Cont)

Case 1: $j \in C_a^{\mathcal{A}} \cap C_b^{\mathcal{B}}$ (j belongs to the same cluster as i in both partitions.)

Using simplified formula, we have $\begin{cases} p_{ij}^{\mathcal{A}} & = \frac{\alpha}{c_a} \\ p_{ij}^{\mathcal{B}} & = \frac{\alpha}{c_b} \end{cases}$. Therefore,

$$p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}} = \alpha \left(\frac{1}{c_a} - \frac{1}{c_b} \right).$$

This equation also holds when $j = i$.

Back to Optimization slide

Proof (Cont)

Case 2: $j \in C_a^{\mathcal{A}} \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})$ (j belongs to the same cluster as i only in the first partition.)

Using simplified formula, we have $\begin{cases} p_{ij}^{\mathcal{A}} & = \frac{\alpha}{c_a} \\ p_{ij}^{\mathcal{B}} & = 0 \end{cases}$. Therefore,

$$p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}} = \alpha \frac{1}{c_a}.$$

Back to Optimization slide

Proof (Cont)

Case 3: $j \in (\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap C_b^{\mathcal{B}}$ (j belongs to the same cluster as i only in the second partition.)

Using simplified formula, we have $\begin{cases} p_{ij}^{\mathcal{A}} & = 0 \\ p_{ij}^{\mathcal{B}} & = \frac{\alpha}{c_b} \end{cases}$. Therefore,

$$p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}} = \alpha \frac{1}{c_b}.$$

Back to Optimization slide

Proof (Cont)

Case 4: $j \in (\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})$ (j belongs to a different cluster than i both partitions.)

Using simplified formula, we have $\begin{cases} p_{ij}^{\mathcal{A}} & = 0 \\ p_{ij}^{\mathcal{B}} & = 0 \end{cases}$. Therefore,

$$p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}} = 0.$$

Back to Optimization slide

Proof (Cont)

These four cases are exhaustive so $\sum_{j=1}^{N} |p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}}|$ will be equivalent to

$$\sum_{j \in C_a^{\mathcal{A}} \cap C_b^{\mathcal{B}}} |p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}}| + \sum_{j \in C_a^{\mathcal{A}} \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})} |p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}}| + \sum_{j \in (\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap C_b^{\mathcal{B}}} |p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}}| + \sum_{j \in (\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})} |p_{ij}^{\mathcal{A}}$$

$$= \sum_{j \in C_a^{\mathcal{A}} \cap C_b^{\mathcal{B}}} \left| \alpha \left(\frac{1}{c_a} - \frac{1}{c_b} \right) \right| + \sum_{j \in C_a^{\mathcal{A}} \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})} \left| \alpha \frac{1}{c_a} \right| + \sum_{j \in (\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap C_b^{\mathcal{B}}} \left| \alpha \frac{1}{c_b} \right| + \sum_{j \in (\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap (\mathcal{P} \setminus C_b^{\mathcal{B}}} |0| =$$

$$= \alpha \sum_{j \in C_a^{\mathcal{A}} \cap C_b^{\mathcal{B}}} \left| \frac{1}{c_a} - \frac{1}{c_b} \right| + \alpha \sum_{j \in C_a^{\mathcal{A}} \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})} \left| \frac{1}{c_a} \right| + \alpha \sum_{j \in (\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap C_b^{\mathcal{B}}} \left| \frac{1}{c_b} \right| =$$

$$= \alpha \left(|C_a^{\mathcal{A}} \cap C_b^{\mathcal{B}}| \cdot \left| \frac{1}{c_a} - \frac{1}{c_b} \right| + |C_a^{\mathcal{A}} \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})| \cdot \frac{1}{c_a} + |(\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap C_b^{\mathcal{B}}| \cdot \frac{1}{c_b} \right)$$

Back to Optimization slide

Proof (Cont)

Replacing this result in the ECS formula will lead to

$$
\begin{aligned}
S_i(\mathcal{A},\mathcal{B}) &= 1 - \frac{1}{2\alpha}\sum_{j=1}^{N} |p_{ij}^{\mathcal{A}} - p_{ij}^{\mathcal{B}}| \\
&= 1 - \frac{1}{2\alpha}\alpha\left(|C_a^{\mathcal{A}} \cap C_b^{\mathcal{B}}| \cdot \left|\frac{1}{c_a} - \frac{1}{c_b}\right| + |C_a^{\mathcal{A}} \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})| \cdot \frac{1}{c_a} + |(\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap C_b^{\mathcal{B}}| \cdot \frac{1}{c_b}\right), \\
&= 1 - \frac{1}{2}\left(|C_a^{\mathcal{A}} \cap C_b^{\mathcal{B}}| \cdot \left|\frac{1}{c_a} - \frac{1}{c_b}\right| + |C_a^{\mathcal{A}} \cap (\mathcal{P} \setminus C_b^{\mathcal{B}})| \cdot \frac{1}{c_a} + |(\mathcal{P} \setminus C_a^{\mathcal{A}}) \cap C_b^{\mathcal{B}}| \cdot \frac{1}{c_b}\right)
\end{aligned}
$$

which is the desired output.

□

Back to Optimization slide

input : *partList*: the list of disjoint partitions
weights: the weight array that contains the number of duplicates
output : the Element-Centric Consistency of the list

$nPartitions \leftarrow \text{size}(partList)$;
$nTotalPartitions \leftarrow \text{sum}(weights)$; ; // includes duplicates
$ecc \leftarrow \mathbf{0}$;
for $i = 1$ **to** $nPartitions - 1$ **do**
  for $j = i + 1$ **to** $nPartitions$ **do**
    $ecc \leftarrow ecc + \text{ecs}(partList[i], partList[j]) * weights[i] * weights[j]$;
  end
end
$nDuplicatesECS \leftarrow 0$;
for $i = 1$ **to** $nPartitions$ **do**
  $nDuplicatesECS \leftarrow nDuplicatesECS + weights[i] * (weights[i] - 1)/2$;
end
$ecc \leftarrow ecc + nDuplicatesECS * \mathbf{1}$;
$ecc \leftarrow ecc/(nTotalPartitions * (nTotalPartitions - 1)/2)$;
return *ecc*

Jump to Weighted ECC slide