Clustering Algorithms

Work 3
Introduction to Machine Learning

Aina Llaneras, Angélica Jaimes, Bernat Comas, Xavier Querol

Contents

1 Introduction

Clustering is a widely used unsupervised learning algorithm that aims to group data with its underlying information. In this report, we are going to implement different clustering algorithms and see how well they organize data in different types of datasets.

Five clustering techniques will be applied: Ordering Points To Identify the Clustering Structure (from now on, referred to as OPTICS), Spectral Clustering, K-Means, two improved K-Means and the Generalized-Suppressed Fuzzy C-Means (GS-FCM). All algorithms will be tested with different parameter configurations, to see which ones work better on each dataset.

The main objective of the report is to compare how different algorithms (and its different parameter configuration) perform on a clustering task in datasets with distinct characteristics.

2 Methodology

In this chapter, the methodology followed to obtain the results will be explained. All techniques used throughout the analysis will be described.

2.1 Preprocessing

In this project, we are using three different datasets. Two of these datasets include both numerical and categorical features (sick and vowel) and one that includes only numerical features (TAO-grid).

For all three datasets (TAO-grid, Sick, and Vowel), preprocessing involves cleaning the data, handling missing values, transforming categorical variables, and scaling numerical values. First, the data is loaded from its source file and converted into a DataFrame. Then, we have constructed a tailor-made preprocessing process for each one of the datasets.

For the TAO-grid dataset, from now on "grid", the steps are straightforward. We simply use a binary encoding for the class, which only takes two values. Then, the numerical features are scaled to a range between 0 and 1 using a min-max scaler.

For the Sick dataset, preprocessing is a bit more complicated. The data is loaded and cleaned, UTF-8 strings decoded and columns with too many missing values identified. Because there is only one row that has age null, we remove it. We remove the rows where the age is over 120 as well because there is only one, and we assume it was an error. We also drop TBG_measured and TBG because TBG only contains NaNs, and TBG_measured contains the value "false" for all columns, indicating that it has not been measured. The rest of the columns that contain NaNs are filled with a Linear Regressor. Numerical columns are then scaled using a MinMaxScaler, categorical variables are transformed using one-hot and binary encoding. Finally, the data is reorganized before being saved as a CSV file.

For the Vowel dataset, the process starts by separating the target variable from the features. The Train_or_Test column is dropped because we are not going to use a train-test split for this project. Categorical features are transformed using one-hot encoding, and binary encoding is applied where needed. The numerical data is scaled

with a min-max scaler to ensure all values fall within the same range. Finally, we save the target column at the end of the dataset.

2.2 Clustering Methods

2.2.1 OPTICS

Optics, or Ordering Points To Identify Cluster Structure, is a Density-Based Cluster-Ordering algorithm. Density based clustering finds clusters by identifying dense regions of data. OPTICS works very similarly to DBSCAN, another popular density-based clustering, however it solves its major weakness, OPTICS can find clusters of varying density and shapes. OPTICS can prove very useful, seeing as it is different from other clustering methods, because it does not have the value of k (number of clusters) as an input required [1].

To apply this method to the data, the scikit learn library will be used. In this implementation of the OPTICS algorithm, clusters are extracted using a DBSCAN-like method. It deviates from the original approach by first performing K-nn searches on all points to identify core sizes, then computing only the distances to unprocessed points when constructing the cluster order [2].

Two different algorithms to compute the K-nn will be tested, the Ball Tree and brute force. The Ball Tree algorithm may be less accurate but more efficient than brute force, and we are interested in seeing if the Ball Tree algorithm can perform as well as brute force while being more efficient. Three different distance metrics will be used and compared, the Euclidean distance and the Manhattan distance, which are widely used metrics and known to work well for numerical data, and the Hamming distance, which is known to perform better for categorical data. The parameter min_samples will be fixed to 15, because in early stages of testing it has proven to work better than its preset value, 5.

2.2.2 Spectral Clustering

Spectral Clustering represents data as an undirected, weighted graph, where vertices are data points and edge weights reflect their similarity. Traditional partitioning methods, like minimum cut or normalized cut, optimize specific objectives but face NP-hard challenges on large datasets. Spectral Clustering goes one step further by using spectral decomposition to relax the problem into a continuous domain, improving computational efficiency [3].

A key step in Spectral Clustering is constructing the similarity and Laplacian matrices, which model relationships and structural properties of the graph, respectively. The Laplacian matrix's eigenvectors, corresponding to the smallest eigenvalues, project the data into a lower-dimensional space where clustering, often via k-means, is performed [3].

In scikit-learn's implementation of Spectral Clustering [6], key hyperparameters include n_clusters, defining the number of clusters, and affinity, which determines how the similarity matrix is computed (rbf or nearest_neighbors). For affinity = nearest_neighbors, n_neighbors specifies the number of neighbors in graph construction. The assign_labels parameter selects the labeling method (kmeans or discretize), and eigen_solver determines how eigenvectors are computed (arpack or lobpcg).

2.2.3 K-Means

K-means, introduced by MacQueen (1967), is a common clustering algorithm that divides a dataset into k clusters by minimizing the variance within each cluster [5]. The number of clusters, k, must be defined in advance. While K-means usually uses Euclidean distance, it can also work with other distance metrics, like Manhattan distance or cosine similarity, depending on the data. The K-means algorithm is as follows:

1. **Initialization:** Choose k initial cluster centers, either randomly or using a method like K-means++.
2. **Assignment Step:** Assign each data point to the nearest cluster center using a chosen distance metric (e.g., Euclidean, Manhattan, or Cosine).
3. **Update Step:** Calculate the new cluster centers as the average (for Euclidean distance) or another method based on the assigned points.
4. **Repeat:** Repeat the assignment and update steps until the cluster assignments no longer change, or a stopping criterion is met (e.g., maximum iterations).

In this project, we will clusterize using the different mentioned distances as the only parameter.

2.2.4 G-Means

The G-means algorithm starts with a small number of K-means centers, usually one, and increases the number of clusters as needed. At each step, K-means is run to optimize the clustering, and a statistical test checks if the data assigned to each center follow a Gaussian distribution. If the data are Gaussian, the center remains unchanged. If not, the cluster is split into two, and the process repeats until no more splits are required. This allows G-means to adaptively determine the number of clusters based on the data.

To test for Gaussian fit, the algorithm uses the Anderson-Darling test. Data points are projected onto the line connecting two candidate child centers, reducing them to one dimension. This simplifies the normality test. If the test confirms the data follow a Gaussian distribution, the cluster is kept as is. If not, the cluster is split into two. A Bonferroni correction is applied to control errors across multiple tests, ensuring reliable results.

There are two ways to initialize child centers. The first uses random small shifts around the original center. The second, a deterministic method, calculates the principal component of the data to place the new centers along the direction of maximum variance. We choose the deterministic approach because it is the one they apply.

2.2.5 Global K-Means

Global k-means is a clustering algorithm designed to overcome some of the limitations of traditional k-means by improving cluster initialization. In this project, we implemented the Global k-means algorithm, as described in the paper by Likas et al. (2003)[4]. This algorithm improves on standard k-means by incrementally adding clusters one at a time, optimizing their placement to minimize the overall clustering error, and avoiding random initialization.

This algorithm requires setting the number of clusters, k, as a parameter. Additionally, the stopping criterion is based on minimizing the total within-cluster sum of squares (WCSS), ensuring compact and well-separated clusters.

Our implementation adapts the process to include the computational improvements of the Fast variant. The algorithm begins by initializing a single cluster center at the mean of all data points. It then incrementally adds new clusters by testing only a subset of data points as potential cluster centers, leveraging the reduced search space to improve efficiency. At each step, the cluster configuration that minimizes WCSS is selected, and the process repeats until the desired number of clusters is achieved.

2.2.6 Fuzzy C-Means

Fuzzy C-Means allows data points to belong to multiple clusters with varying membership degrees. In this project, we implemented the "Generalized-Suppressed Fuzzy C-Means" algorithm, as described in the 2014 paper Generalization Rules for the Suppressed Fuzzy C-Means Clustering Algorithm[7]. This algorithm, improves on the traditional Fuzzy C-Means algorithm by incorporating a suppression factor that reduces the influence of noise and outliers in the clustering process.

This algorithm requires two parameters, m and η, to be manually adjusted. Our implementation also includes α, but we will not tweak it because the generalized version updates it automatically at each iteration. The m parameter controls the level of fuzziness: higher values allow more even membership across clusters, while lower values create more definitive assignments.

Our implementation follows the algorithm in the paper, using the first generalization rule with the η parameter. The process begins by setting the number of clusters, the fuzzification parameter, and initializing cluster prototypes randomly. It then computes fuzzy memberships with the standard FCM formula, identifies the winner cluster for each point, and applies generalized suppression on the membership matrix (updating α and suppression rules). Finally, suppressed memberships are calculated, and prototypes are updated. This repeats until the change in prototypes is below a predefined threshold.

We measure this change through the norm difference between the current cluster centers and the previous ones. If the sum of these differences is below the predefined tolerance, we conclude that the cluster centers no longer change significantly. This way we make sure to give the same importance to all clusters. This tolerance is a parameter of the algorithm, but we are going to consider that it has converged when our tolerance is below 0.00001.

2.3 Evaluation methodology

To evaluate how the different clustering methods and their hyperparameters perform on the different dataset, different validation methods will be used, both internal criteria and external. As internal indexes, the Silhouette Coefficient, the Davis-Bouldin Index and the Calinski-Harabasz Index will be used. The Silhouette Coefficient measures how well samples are clustered taking into account samples that are similar to them, it ranges from -1 to 1 and higher values mean a better data separation has been achieved. The Davis-Bouldin Index measures how further apart are clusters and how dispersed they are, when the index takes a value of 0 it means the clusters are perfectly separated,

and pretty compact, and higher values mean they are not very further from each other and may be a more dispersed. The Calinski-Harabasz score uses the variance ratio to evaluate the cluster between and within. Higher values mean better separated clusters with a small intra-cluster variance. As external indexes, the Adjusted Rand Index (ARI), the F-measure and when necessary the Purity will be used. ARI measures how well true labels are classified between clusters, it ranges from -1 to 1, where 1 means the true labels and the cluster labels are identical. F-measures uses precision and recall to evaluate the true and predicted labels, it ranges from 0 to 1, where 1 is a perfect classification. The purity is the ratio between the dominant class in the cluster, and the size of the cluster, it ranges from 0 to 1 where 1 indicates that the true labels are the predicted labels. On top of all these metrics, the solving time will also be taken into account.

Regarding the GS Fuzzy C Means clustering model, we will also be implementing and using a fuzzy-specific metric, the Xie-Beni, that measures the quality of fuzzy clustering by considering the compactness of data points within clusters (based on their membership degrees) and its separation.

This metrics will be implemented from the scikit learn library, except for the Xie-Beni, Purity and F-measure, which will be manually implemented. The combination of both external and internal indexes will give us a thorough view of how the different clustering methods perform.

3 Results

3.1 Individual Models

3.1.1 OPTICS

The hyperparameters tested in this study are the K-nn algorithm applied at the beginning of the OPTICS algorithm, which are the Ball Tree algorithm and brute force. Different distance metrics will also be tested, in this case, the Euclidean, Manhattan and Hamming distance. The model has been trained with all three datasets proposed, the Grid, Sick and Vowel datasets. In the tables displayed in this section, the F-measure will be referred to as F-m., the Davis-Bouldin index as DB, the Silhouette Coefficient as Silho., and the Calinski-Harabasz as CH.

Grid

k	Dist	Alg	ARI	Purity	F-m.	DB	Silho.	CH	Time
2	Eucl.	Brute	-4.95e-4	0.500	0.557	0.0	-4.07e-4	2.55e-27	3.329
2	Eucl.	B.Tree	-4.95e-4	0.500	0.557	0.0	-4.07e-4	2.55e-27	1.741
2	Manh.	Brute	-4.95e-4	0.500	0.557	0.0	-4.07e-4	2.55e-27	1.954
2	Manh.	B.Tree	-4.95e-4	0.500	0.557	0.0	-4.07e-4	2.55e-27	1.767
1	Hamm.	Brute	NaN	NaN	NaN	NaN	NaN	NaN	2.159
1	Hamm.	B.Tree	NaN	NaN	NaN	NaN	NaN	NaN	1.513

Table 1: Metrics of the OPTICS algorithm for Grid with Solving Time

For the grid dataset (1), the OPTICS algorithm does not seem to work. As we can see, when using the Hamming distance, the algorithm only finds one cluster, therefore no metrics are calculated. For the other two distances, regardless of the algorithm, two

clusters are found. However, the values of all metrics are very bad. This means that the OPTICS algorithm does not work for the Grid dataset, which makes sense. The OPTICS algorithm is a density based algorithm, and the GRID dataset, which only has two features, when plotted represents a perfect full circle of data points. Seeing as all points are close, the OPTICS algorithm fails.

Sick

k	Dist	Alg	ARI	Purity	F-m.	DB	Silho.	CH	Time
38	Eucl.	Brute	0.002	0.962	0.494	1.313	0.278	126.094	6.736
38	Eucl.	B.Tree	0.002	0.962	0.494	1.313	0.278	126.321	8.498
38	Manh.	Brute	0.001	0.962	0.523	1.274	0.241	120.525	5.021
38	Manh.	B.Tree	0.001	0.962	0.523	1.274	0.241	120.525	5.380
4	Hamm.	Brute	-0.003	0.961	0.797	2.770	0.147	226.693	6.064
4	Hamm.	B.Tree	-0.003	0.961	0.797	2.770	0.147	226.693	5.517

Table 2: Metrics of the OPTICS algorithm for Sick

In the sick dataset (2), OPTICS performance has not been affected by the choice of algorithm, which implies that the Ball Tree algorithm does not lose performance to the brute force. The distance metrics however do affect the results. The Euclidean and Manhattan distance produce similar results, identifying 38 clusters, while the Hamming distance identified 4. While the Hamming distance performs better in terms of F-measure and Calinski-Harabasz Index, overall the other two seem to produce more consistent results. Seeing that both the Euclidean and the Manhattan distance seem to perform better or worse depending on the choice of metric, we base our choice of best model according to the efficiency, therefore the Manhattan distance, with brute force, seems to be the best combination.

Vowel

k	Dist	Alg	ARI	Purity	F-m.	DB	Silho.	CH	Time
21	Eucl.	Brute	0.033	0.176	0.224	1.323	0.218	76.321	2.110
21	Eucl.	B.Tree	0.033	0.176	0.225	1.323	0.218	76.003	0.997
18	Manh.	Brute	0.040	0.168	0.233	1.347	0.142	68.332	0.717
18	Manh.	B.Tree	0.040	0.168	0.233	1.347	0.142	68.332	0.769
14	Hamm.	Brute	-0.011	0.091	0.108	0.934	0.536	271.772	0.950
14	Hamm.	B.Tree	-0.011	0.091	0.108	0.934	0.536	271.772	0.796

Table 3: Metrics of the OPTICS algorithm for Vowel

For the vowel dataset (3), we find that the choice of algorihtm does not affect the performance in terms of the metrics evaluated, however it does affect the solving time, the Ball Tree seems to be faster. The number of clusters found varies depending on the distance used, the Euclidean finds k = 21, the Manhattan k = 18 and the Hamming k = 14. Overall, the external indexes show poor performance across all models, but when looking at the internal indexes we see that the best distance metric seems to be the Hamming, followed by the Euclidean and then the Manhattan across all of them. This said, the best hyperparameter combination seems to be using the Hamming distance alongside the Ball Tree algorithm.

3.1.2 Spectral Clustering

The hyperparameters considered in this study include the number of clusters, the type of affinity, the label assignment method, the eigenvalue solver, and, when applicable, the number of nearest neighbors. The spectral clustering model is trained iteratively for each combination of hyperparameters. In each iteration, the model is fitted to the data, and the corresponding cluster labels are generated. Subsequently, objective metrics are calculated to evaluate the performance of each configuration. This systematic approach ensures a comprehensive exploration of the parameter space, allowing for the identification of the optimal settings for each dataset.

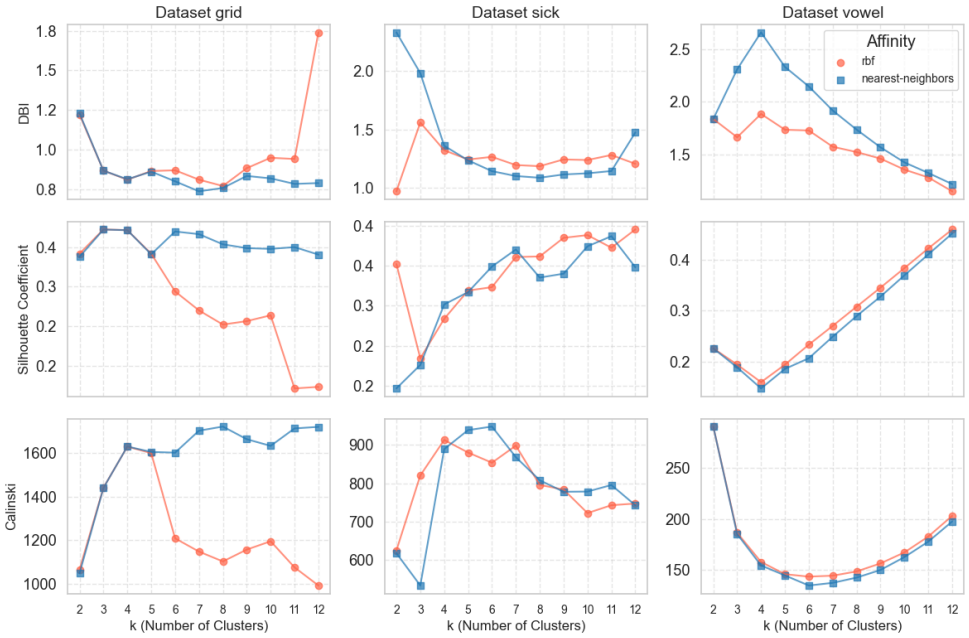


Figure 1: Spectral results

Since the affinity parameter defines how the similarity between points is calculated and has a critical impact on clustering quality, the models have been filtered based on this criterion. The best models for each k value are determined by identifying the parameter combinations that optimize clustering quality metrics using a ranking system. Subsequently, the configurations matching these optimal parameters are selected. This approach enables the comparison and visualization of results for each dataset and affinity type, as illustrated in Figure 1 presents a comparison of clustering evaluation metrics across the different datasets, considering two affinity (RBF and nearest-neighbors). The analysis examines variations in the number of clusters (k) to determine the optimal configuration for each dataset.

In the case of the Grid dataset, the optimal number of clusters is determined to be $k = 4$. This conclusion is supported by all three evaluation metrics. The DBI achieves its lowest value at $k = 4$, indicating strong compactness and separation between clusters. Similarly, the SC reaches its maximum value at this point, demonstrating that the clusters are both well-defined and distinct. Lastly, the CH also attains its highest value at $k = 4$, confirming that this configuration provides the most effective clustering structure for the dataset. Notably, the nearest-neighbors affinity consistently

outperforms RBF across all metrics for this dataset.

For the Sick dataset, the analysis also suggests $k = 7$ as the optimal number of clusters. At this point, the DBI exhibits a low value, reflecting good separation between clusters. The SC shows a marked increase and stabilizes around $k = 7$, which highlights the clustering quality in terms of intra-cluster cohesion and inter-cluster separation. The calinski further supports this conclusion, peaking at $k = 7$, which indicates a well-balanced relationship between intra-cluster compactness and inter-cluster dispersion. Similar to the Grid dataset, the nearest-neighbors affinity demonstrates superior performance compared to RBF.

In the case of the Vowel dataset, the metrics indicate that the best clustering structure is achieved with $k = 12$. The DBI consistently decreases as k increases, reaching its minimum value at $k = 12$, which suggests improved separation between clusters. The SC exhibits a gradual increase, stabilizing at $k = 12$, signifying that this configuration produces compact and well-separated clusters. The CH metric also peaks at $k = 12$, further reinforcing that this number of clusters captures the dataset's inherent complexity. While the nearest-neighbors affinity slightly outperforms RBF in the DBI and CH metrics, both affinities yield comparable results in terms of the Silhouette Coefficient.

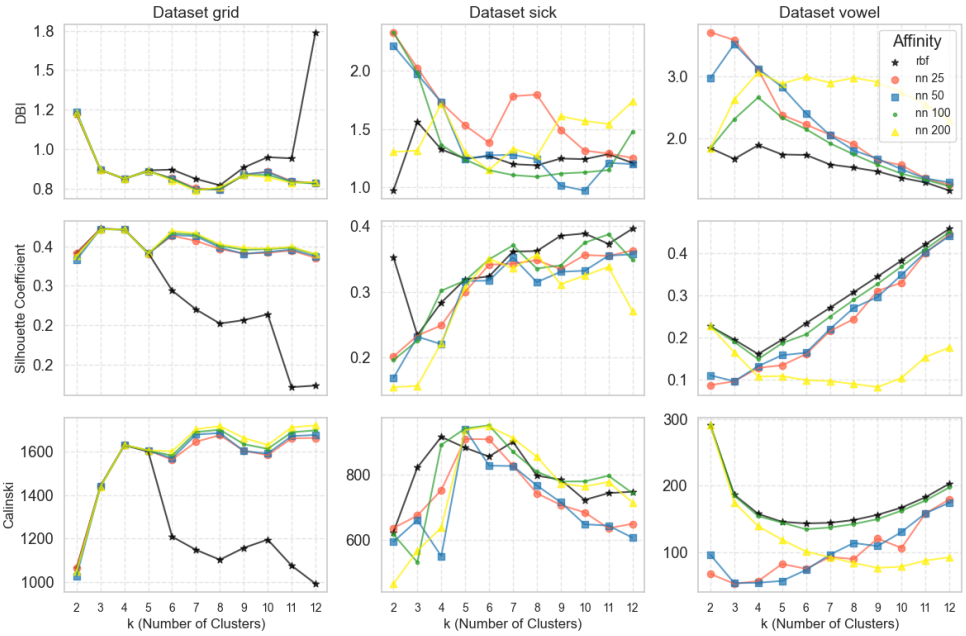


Figure 2: Spectral results

Overall, the nearest-neighbors affinity generally outperforms rbf across the evaluated metrics. In the figure 2, it can be observed that affinities based on nearest neighbors consistently outperform RBF, particularly when the number of neighbors is moderate (25, 50, or 100). This suggests that these configurations are better suited for capturing local relationships and generating more coherent partitions in the analyzed datasets. The configuration with n=200 demonstrates that considering such a high number of neighbors introduces an overgeneralization effect, making it harder to identify essential local patterns for effective partitioning.

3.1.3 K-Means

The algorithm is not deterministic, meaning it can produce different results each time it runs. To deal with this, we run the algorithm N times for every combination of dataset, k, and distance metric.

In Figure 3, we show the Silhouette Coefficients for all combinations of datasets, k, and distance metrics. The results clearly show that the algorithm gives different outputs depending on how it starts, even with the same dataset and parameters. This highlights the importance of running the algorithm multiple times to reduce errors caused by random initialization.

For example, when using the sick dataset with k=2 and the distance metric set to 'euclidean', the Silhouette Scores vary a lot. In some runs, the score is as low as 0.05, while in others, it reaches as high as 0.4. This wide range of results shows why running the algorithm more than once is necessary.

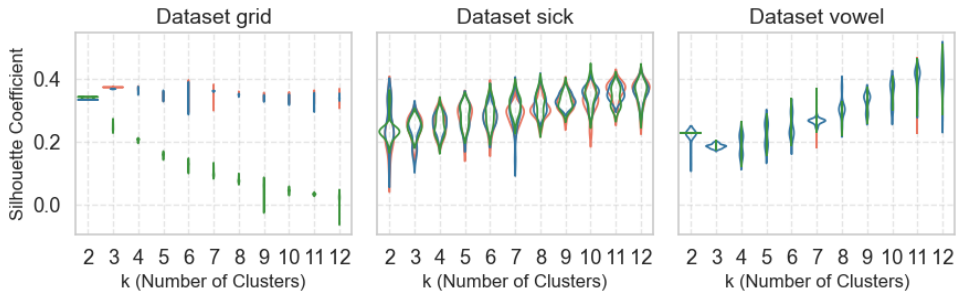


Figure 3: Running KMeans multiple iterations

For each combination, we select the result that has the highest score. This ensures that we are picking the best outcome from all the attempts. As we have three unsupervised metrics to evaluate the clustering, we use them all to choose the best result. All the runs for each combination will be ranked based on the three metrics, and the one with the lowest mean value will be considered.

With this new subset of runs, we build Figure 4 to understand better all the clustering options.

For the grid dataset, the optimal number of clusters is considered to be 4, where the Calinski-Harabasz index and Silhouette coefficient reach their peaks, and the Davies-Bouldin Index (DBI) is sufficiently low. Among the evaluated distances used with k=4, the Euclidean distance yields the best performance.

For the sick dataset, the optimal number of clusters is identified as 7, where the Calinski-Harabasz index, Silhouette coefficient, and DBI are in harmony. With k=7 fixed, different distances result in varying performance. To determine the best metric, the three distances were ranked across the different metrics. As shown in Table 4, Manhattan distance emerges as the best-performing one.

For the vowel dataset, the optimal number of clusters is determined to be 12, where the Calinski-Harabasz index and Silhouette coefficient continue to increase while the DBI decreases. Although a greater number of clusters could also be considered, this falls outside the scope of the current study. With k=12 fixed, different distance metrics

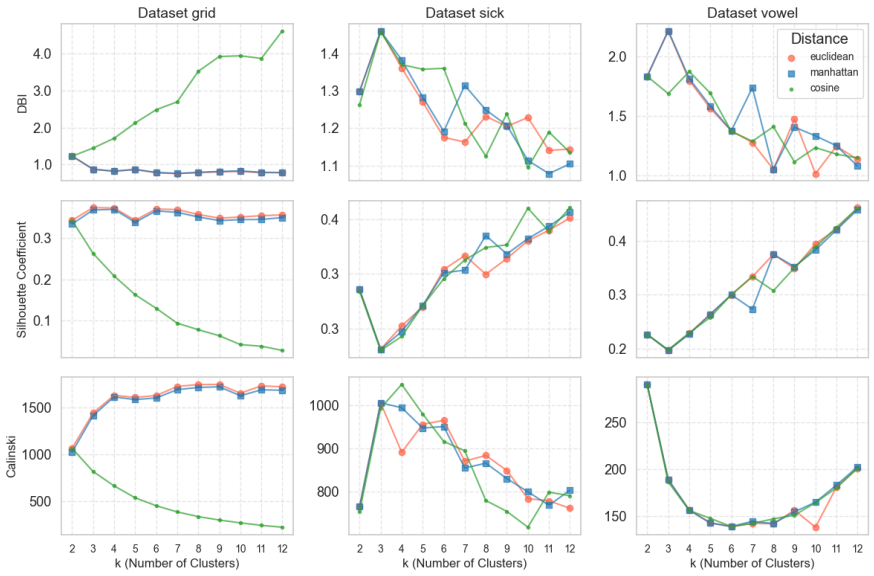


Figure 4: KMeans results

	Davies-Bouldin Index	Calinski	Silhouette Coefficient	Mean
Manhattan	1.0	2.0	1.0	1.33
Euclidean	2.0	1.0	2.0	1.66
Cosine	3.0	3.0	3.0	3.0

Table 4: Metric Ranks in sick dataset for KMeans

result in varying performance. To decide the best-performing distance, the three distances were ranked across the different metrics. As shown in Table 5, cosine distance is identified as the best-performing distance.

	Davies-Bouldin Index	Calinski	Silhouette Coefficient	Mean
Cosine	1.0	1.0	3.0	1.66
Euclidean	2.0	2.0	1.0	1.66
Manhattan	3.0	2.0	3.0	2.66

Table 5: Metric Ranks in vowel dataset for KMeans

3.1.4 G-Means

The algorithm is deterministic so we will only run the model one time for each combination of (dataset, k and distance).

For the grid dataset, the optimal number of clusters is considered to be 9, where the Calinski-Harabasz index and Davies-Bouldin Index (DBI) reach their peaks, and the Silhouette coefficient attains its second-highest value. Among the evaluated distances for k=9, the Euclidean distance yields the best performance.

For the sick dataset, the optimal number of clusters is identified as 4, where the Calinski-Harabasz index reaches its peak, and the Silhouette coefficient and DBI ex-

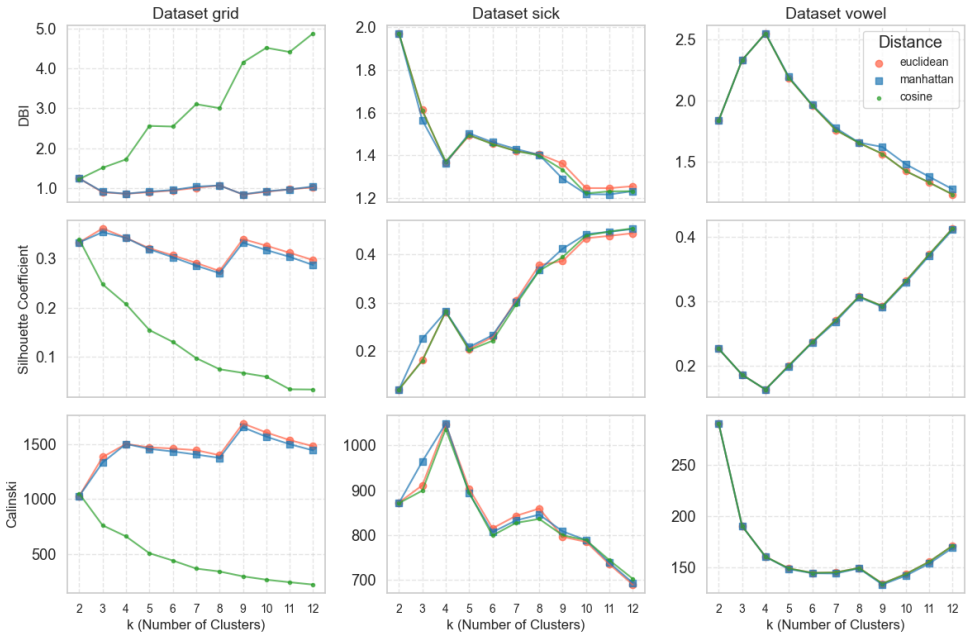


Figure 5: GMeans results

hibit satisfactory values. With k=4 fixed, the Manhattan distance delivers the best performance.

For the vowel dataset, the optimal number of clusters is determined to be 12, where Silhouete Cofficient and DBI reach their peaks, and Calinski the 3rd highest value. Using the ranking method, a tie is observed between Euclidean and Cosine distances.

	Davies-Bouldin Index	Calinski	Silhouette Coefficient	Mean
Euclidean	1.5	1.5	1.5	1.5
Cosine	1.5	1.5	1.5	1.5
Manhattan	3.0	3.0	3.0	3.0

Table 6: Metric Ranks in vowel dataset for Gmeans

3.1.5 Global K-Means

The algorithm is deterministic so we will only run the model one time for each combination of (dataset, k and distance).

For the grid dataset, the optimal number of clusters is determined to be 4. At this value, the three evaluation metrics show notably good results, and the cluster count remains manageable. Among the evaluated distance measures for k=4, the Euclidean distance delivers the best performance.

For the sick dataset, the Euclidean distance emerges as the most effective. With this distance measure fixed, the optimal number of clusters is found to be 12, where the Silhouette Coefficient and DBI achieve their highest values, and the Calinski-Harabasz Index shows a satisfactory result.

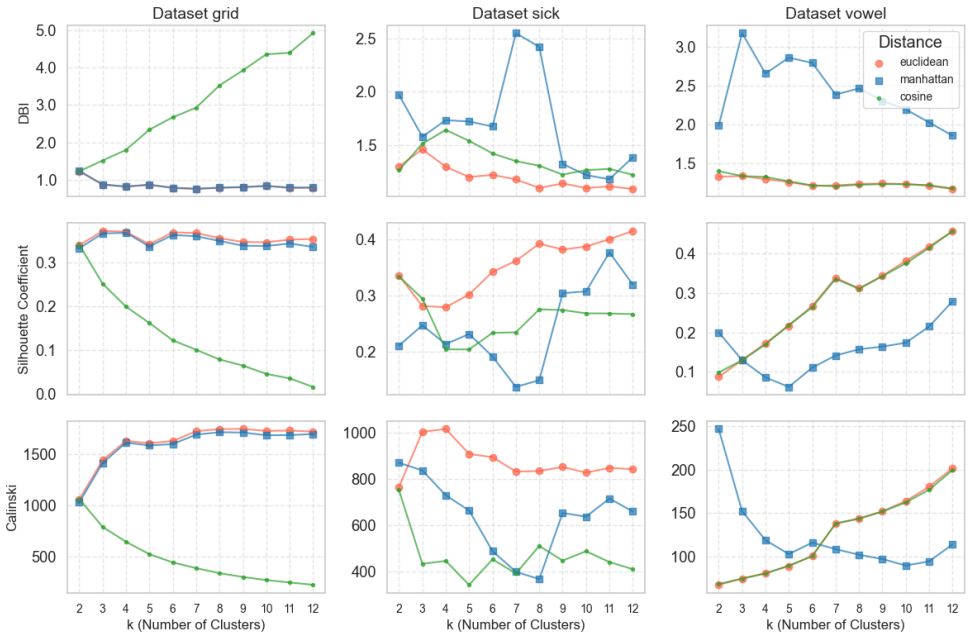


Figure 6: Global KMeans results

For the vowel dataset, the optimal number of clusters is also identified as 12. At this point, the Silhouette Coefficient and DBI reach their peaks, while the Calinski-Harabasz Index records its second-highest value. Based on the ranking method, the Euclidean distance is confirmed as the best-performing one.

	Davies-Bouldin Index	Calinski	Silhouette Coefficient	Mean
Euclidean	1.0	1.0	2.0	1.33
Cosine	2.0	2.0	1.0	1.66
Manhattan	3.0	3.0	3.0	3.0

Table 7: Metric Ranks in vowel dataset for GlobalKmeans

3.1.6 Fuzzy C-Means

To find the best hyperparameter combination, we will test m values between 1.05 and 2, as these are standard and effective for the formula. For the generalization parameter η, we will use 0.1, 0.5, and 0.9, as it must remain between 0 and 1. To account for the variability caused by random cluster initialization, each parameter combination will be run 10 times, with the best configuration selected based on the Xie-Beni metric.

The "eta" parameter plays an important role in balancing how compact and flexible clusters are, which affects performance across different metrics. However, the results for different values of m show only minor variations. For the sick dataset, η=0.5 consistently gives strong results across metrics like the Davies-Bouldin Index (DBI), Xie-Beni, and Silhouette Coefficient. In the grid dataset, the best η value depends heavily on m, except for Xie-Beni, which consistently performs best with η=0.1. For the vowel dataset, the first three metrics show little difference across η values, but

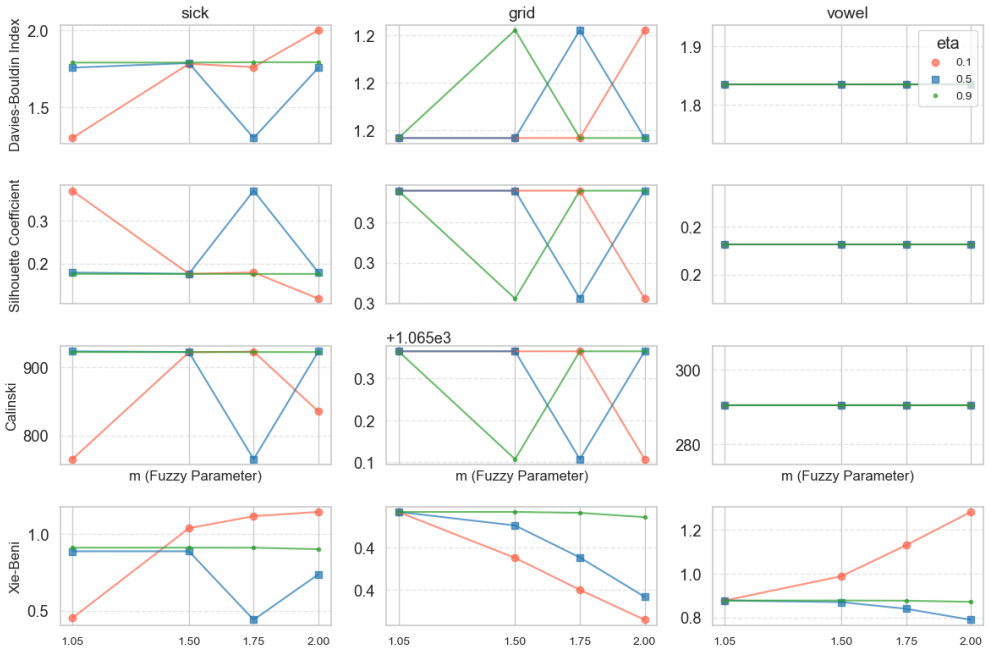


Figure 7: Plotting of a preselection of metrics for every dataset and value for m and eta, getting the mean rank of the model for all k.

η=0.5 provides the best Xie-Beni scores. Based on these observations, η=0.5 will be chosen for the optimal model on the sick and vowel datasets, and η=0.1 will be chosen for the grid dataset.

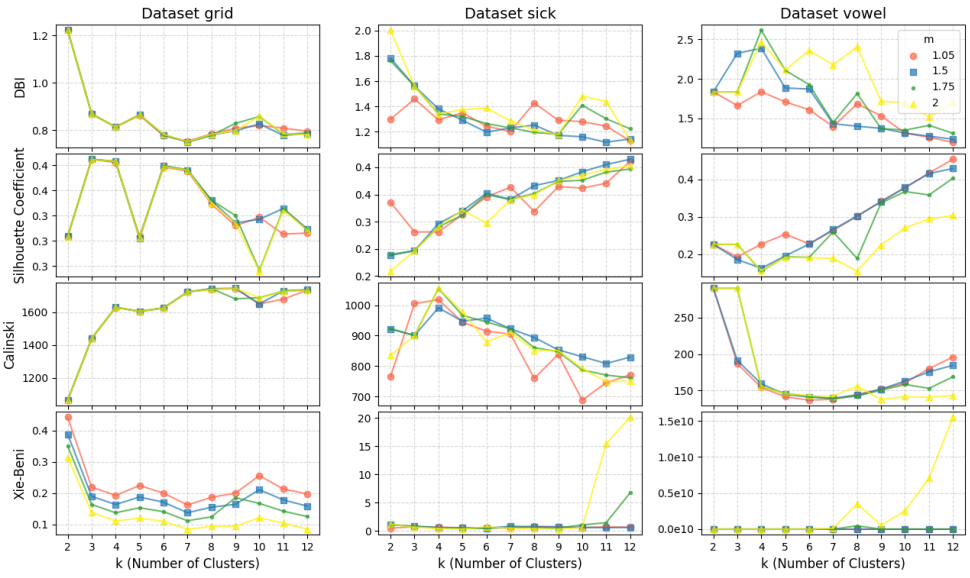


Figure 8: Plotting of a preselection of metrics for every dataset and value for m

Regarding the sick dataset, the Davies-Bouldin Index appears to show a clear elbow on k=6. On the other hand, the Silhouette Coefficient increases with k, peaking around

k=6, but it continues increasing with k. This might mean that the optimal is higher than k=12. The Calinski-Harabasz score also favors a k between 3 and 6, showing a strong peak at this value before declining. The Xie-Beni index minimizes around k=6, confirming this as the optimal number of clusters. The best value for m seems to be m=1.5 or m=1.75, as they consistently yield stable and better-performing metrics across most values of k. For instance, the Davies-Bouldin and Xie-Beni indices are more stable and lower for these values of m. This dataset seems to benefit most from clustering with k=6 and a moderate value of m (1.5 or 1.75). Larger k values introduce instability, and overly high m (e.g., m=2) leads to a deterioration in performance for certain metrics. So we will choose the optimal values for this dataset to be k=6, m=1.5 and eta=0.5.

The grid dataset consistently shows good performance for lower values of k. The Davies-Bouldin Index and Xie-Beni both suggest k=7 as optimal. However, the Silhouette Coefficient does not show a lot of variation, indicating that the dataset may cluster well across a range of different k. The Calinski-Harabasz score slightly favors k=7 to k=8, with diminishing returns for higher k. In this case higher m values (m=1.75 and m=2) tend to perform better across all metrics, being clearer in the Davies-Bouldin and Xie-Beni indices. In conclusion, k=7 and m=2 and eta=0.1 offer the most stable and optimal clustering. Higher k values do not significantly improve the clustering metrics, and lower m values show clearly worse performance in certain metrics like Xie-Beni.

For the vowel dataset, the behavior is more confuse. The Davies-Bouldin Index favors larger k, while the Silhouette Coefficient and Calinski improves steadily with increasing k, suggesting more clusters might better fit the data. The Xie-Beni shows that lower m works better with high k. Overall, despite indications of smaller k being sufficient by Calinski, DBI and SC lead us to select k=12, m=1.05, and η=0.5 as the optimal configuration.

3.2 Global Comparison

DS	Method	K	ARI	F-Measure	DBI	SC	Calinski	ST
Grid	GlobalK	4	0.26	0.60	0.81	0.37	1629.66	18.53
	Gmeans	9	0.10	0.34	0.83	0.34	1688.81	0.03
	Kmeans	4	0.25	0.59	0.81	0.37	1630.56	0.00
	Spectral	8	0.11	0.37	0.76	0.35	1721.32	0.65
	GS-FCM	7	0.12	0.42	0.75	0.37	1724.07	0.99
	Optics	2	-4.95e-4	0.557	0.0	-4.07e-4	2.55e-27	1.741
Sick	GlobalK	12	0.00	0.35	1.09	0.42	843.32	149.64
	Gmeans	4	0.00	0.55	1.36	0.29	1048.49	0.06
	Kmeans	7	0.00	0.55	1.16	0.37	870.34	0.05
	Spectral	7	0.00	0.57	1.10	0.37	868.26	1.38
	GS-FCM	6	0.00	0.55	1.19	0.35	952.39	0.23
	Optics	38	0.01	0.523	1.274	0.241	120.525	5.41
Vowel	GlobalK	12	-0.01	0.13	1.17	0.46	201.81	13.28
	Gmeans	12	-0.01	0.13	1.23	0.41	171.21	0.03
	Kmeans	12	-0.01	0.11	1.08	0.46	202.55	0.02
	Spectral	12	-0.01	0.12	1.22	0.45	197.49	0.20
	GS-FCM	12	-0.01	0.13	1.17	0.46	198.69	0.05
	Optics	14	-0.011	0.108	0.934	0.536	271.772	0.796

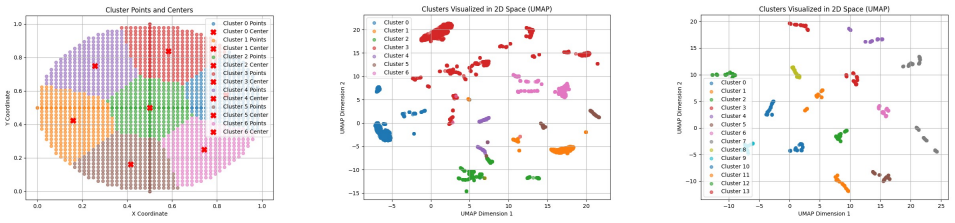
For the Grid dataset, Table 9 shows that GS-FCM is arguably the best method, with the best Calinski and an F-Measure of 0.42. While the ARI value is not high, it outperforms OPTICS, Gmeans and Spectral, indicating that it finds a better structure in the data. GlobalK is a close second. Optics, Gmeans, and Kmeans struggle the most, the later one showing good silhouette score but a worse DBI. That being said, the performance of all the models is quite bad and very similar, obtaining close silhouette scores and DBI indices as well as Calinski metrics, highlighting the lack of clear structure of the grid dataset.

In the Sick dataset, the clustering methods show better performances. Optics has the highest DBI score, indicating that it found fewer distinct clusters and potentially struggled with noise or overlapping data. GlobalK achieved the highest Silhouette Coefficient (0.42), suggesting it performed reasonably well in terms of cluster cohesion and separation. That being said, as the best one for this dataset we will pick Spectral, as it shows the best F-measure and Calinski metric, while having a much more reduced solving time than GlobalK.

In the Vowel dataset, Optics performed the best among all clustering methods, as it showed slightly higher scores in all the internal metrics: Silhouette Score, Calinski and smallest DBI. That being said, because OPTICS is a model that finds its optimal K, we see that it proposes a k of 14, and this might mean that if we had tried bigger values for K in the rest of the datasets we might even have outperformed it.

That being said, not even in this case have we correctly identified the true labels as clusters. In the Grid dataset, GlobalK and Kmeans showed some structure, but still fell short of perfectly identifying the true labels, with ARI values around 0.25–0.26. For the Sick and Vowel datasets, the methods failed to find any meaningful clusters, with most ARI values at or near 0. The low ARI scores across all methods indicate that these labels are not easy to derive from the data without having previously seen them. This was expected, especially in the case of the Grid dataset, where the points are defined by their coordinates, and the labels appear arbitrary from the perspective of clustering algorithms, as they cannot derive the labeling logic purely from the features without supervision.

Finally, Figure 9 shows the plotted clusters for each dataset using umap when the number of features was greater than 2.



(a) Grid dataset. (b) Sick dataset. (c) Vowel dataset.

Figure 9: Execution results for each dataset with the best model respectively.

4 Conclusions

Taking all the results into account, what we have observed throughout the algorithms is that the choice of dataset affects the best combination of hyperparameters. Starting with the OPTICS, we have clearly seen the impact of its parameters. In the Spectral clustering, the optimal number of clusters changed across the datasets, however it has been found that the best affinity is the nearest neighbors in all three datasets. When applying the K-Means algorithm, all three datasets have different optimal number of clusters and distances, which are repeated when applying G-Means. For the Global K-Means algorithm, the sick and vowel datasets have the same optimal k and distance, while the grid dataset only opposes in the optimal number of clusters. Finally, for the Fuzzy C-Means, the different datasets have different optimal values of k, m and η. All this leads us to believe that when performing clustering, different values of the hyperparameters need to be tested in order to reach the best possible combination to achieve a good outcome.

What is interesting to mention, is that when performing the algorithms that have to improve the K-Means algorithm, G-Means and Global K-Means, we have not observed an improvement. The Global K-Means removes the randomness of the initialization, and the G-Means algorithm does not assume circular clusters, and it also finds its own value of k, it does not need to be an input. Even if in theory these algorithms improve the performance, what we have observed in all three of our datasets, when comparing them with the different metrics, is that they do not necessarily perform a better clustering. Global K-Means even worsens the performance when efficiency is taken into account. This might be due to the fact that we are running K-means ten times and selecting the result that we consider to be the best, whilst this can't be done for the improved versions because they are deterministic.

All in all, when comparing the best models of each algorithm, we have observed that the dataset has a big impact. In case of the Grid dataset, with only numerical attributes, we have seen quite a bad performance from all models, however if we had to choose the best one, the Fuzzy C-Means algorithm would stand out. The worst one by far is the OPTICS algorithm, because it is a density-based clustering and the dataset contains equally separated points as features. On the Sick dataset, which has mixed attributes, a better performance was achieved, but picking the best algorithm overall depends on which metric one chooses to evaluate the algorithms. We have chosen the Spectral Clustering as the best, because it consistently performs well across all metrics. Finally, for the Vowel dataset, which has mixed attributes as well, the OPTICS algorithm seems to perform better than the others overall. This could be, however, due to the other algorithms being evaluated with 12 as the highest number of clusters. The OPTICS algorithm assesses k = 14 as the optimal k, and the other datasets if fitted with k = 14 might perform similarly to it, seeing that all of them have deemed k = 12 as the best number of clusters.

In conclusion, when performing clustering, different algorithms with different hyperparameter combinations should be tested to achieve the best results possible.

References

[1] Mihael Ankerst et al. "OPTICS: Ordering points to identify the clustering structure". In: *ACM SIGMOD Record* 28.2 (1999), pp. 49–60.

[2] scikit-learn developers. *OPTICS Clustering Algorithm — scikit-learn 1.3.1 Documentation*. https://scikit-learn.org/dev/modules/generated/sklearn.cluster.OPTICS.html. Accessed: 2024-12-05. 2024.

[3] Hongjie Jia et al. "The latest research progress on spectral clustering". In: *Neural Computing and Applications* 24 (2014), pp. 1477–1486.

[4] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. "The global k-means clustering algorithm". In: *Pattern Recognition* 36.2 (2003). Biometrics, pp. 451–461. ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(02)00060-2. URL: https://www.sciencedirect.com/science/article/pii/S0031320302000602.

[5] J MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*. 1967.

[6] F. Pedregosa et al. *sklearn.cluster.SpectralClustering*. https://scikit-learn.org/1.5/modules/generated/sklearn.cluster.SpectralClustering.html. Accessed: 2024-11-30. 2024.

[7] László Szilágyi and Sándor M. Szilágyi. "Generalization rules for the suppressed fuzzy c-means clustering algorithm". In: *Neurocomputing* 139 (2014), pp. 298–309. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2014.02.027. URL: https://www.sciencedirect.com/science/article/pii/S0925231214004263.