COMMUNITY DETECTION WITH APPLICATIONS TO MULTIREFERENCE ALIGNMENT

Project Number: 20.1.1.2135

Final report

Written by

Alexey Kiryushkin 324439876

Tomer Matityahu 312116668

Under the instruction of

Mr. Noam Janco Tel-Aviv University

Dr. Tamir Bendory Tel-Aviv University

Project was done from home

TABLE OF CONTENTS

ABSTRACT

Single-particle reconstruction in Cryogenic Electron Microscopy (cryo-EM)[2] is a tool for constructing a 3D model of a biological macromolecule using 2D projections of the macromolecules taken by an electron microscope. An unsupervised classification of the 2D images is required in order to separate macromolecular projections of different conformations. Due to high noise levels and data heterogeneity, sophisticated clustering methods are needed.

In our project we employ Community Detection (CD) algorithms to cluster data generated from the Multireference Alignment (MRA) statistical model, the model abstracts away much of the intricacy of cryo-EM while retaining some of its essential features. **Fig. 1** shows a diagram of the process.

In addition to the implementation of CD algorithms on the MRA data, we use a modified K-means clustering algorithm to perform the same task, and compare the performance of both methods. In the last section of this document we discuss the differences between the methods and the advantages and disadvantages of the method we propose.

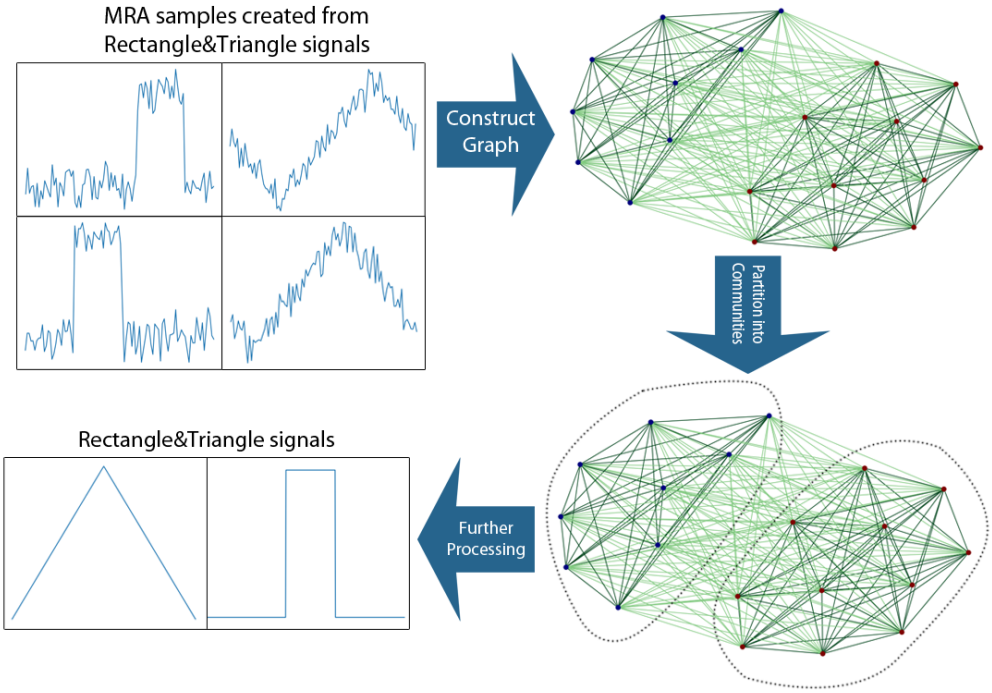


Figure 1: Project process. Further processing stage presents the idea behind clustering the data and is outside of the scope of the project.

List of Figures

List of Tables

List of Abbreviations

C

CD Community Detection. 3, 4, 7, 9, 11

cryo-EM Cryogenic Electron Microscopy. 3, 7, 8

F

FFT Fast Fourier Transform. 8

M

MI Mutual Information. 11

MRA Multireference Alignment. 1, 3, 4, 7–9

N

NMI Normalized Mutual Information. 11

S

SNR Signal To Noise Ratio. 4, 7, 8

1 Introduction

Single-particle reconstruction in cryo-EM is a powerful image-processing tool used to determine the 3D structure of biological macromolecular complexes. 2D images (micrographs) of a macromolecule are taken by an electron microscope and particle picking is performed to extract 2D images of the macromolecules. Essentially, the set of all the 2D images for a given macromolecule spans a 3D model of the macromolecule. Thus, single-particle reconstruction is using the 2D images to build a 3D model of the macromolecule.
Due to high sensitivity of the biological macromolecules to radiation damage, electron microscope provides limited electron doses when producing micrographs. This and other factors like low contrast of micrographs and digitalization of the images result in cryo-EM data having very low Signal To Noise Ratio (SNR)[2].
The cryo-EM technology has the potential to offer the ability to analyze different functional and conformational states of macromolecules, an important ability for the field of molecular biology. Practically, it entails the classification of heterogeneous cryo-EM data.
Many different approaches for cryo-EM data classification have been developed. Typically, likelihood optimization algorithms and Bayesian inference frameworks are used to deal with data heterogeneity[23, 22, 21, 24, 6].
In our project we propose a different approach for cryo-EM data classification using Community Detection (CD) algorithms that are common in the field of complex networks. According to our approach, data will be classified following the steps:

- Converting data into a weighted graph, where each node corresponds to a sample and the edges between nodes represent the degree of similarity between samples.
- Applying Community Detection algorithms to partition the graph into distinct communities
- Each community represent a single conformation of a sampled macromolecule.

For the sake of an abstraction of the cryo-EM data we use the Heterogeneous Multireference Alignment (MRA) statistical model, throughout our project we use the simplified 1D version of the model.
In our project we introduce the modified KMeans algorithm that takes into account the characteristics of the MRA data. The modified KMeans is used as a reference point against which Community Detection (CD) algorithms are evaluated.

2 Background

2.1 Heterogeneous 1D Multireference Alignment model

In our project we use the Heterogeneous 1D Multireference Alignment (MRA) statistical model[5] for the sake of cryo-EM data abstraction. Below is the definition of the model.
Let $x_1, ..., x_K \in \mathbb{R}^L$ be K unknown normalized signals (distinct even up to shift) and let R_s be the cyclic shift operator: $(R_s x)[n] = x[\langle n - s \rangle_L]$. We are given N observations:

$$y_j = R_{s_j} x_{k_j} + \varepsilon_j, \quad j = 1, ..., N \tag{1}$$

where $s_j \sim U[0, L-1], k_j \sim U[0, K-1]$ and $\varepsilon_j \sim \mathcal{N}(0, \sigma^2 I)$ is i.i.d white Gaussian noise. Our goal is to estimate the signals $x_1, ..., x_K$ from the observations.
Simply speaking, MRA observation is a randomly chosen signal x_{k_j}, shifted randomly by s_j and distorted using white noise. **Fig. 2** shows an example of two MRA observations at different noise levels.

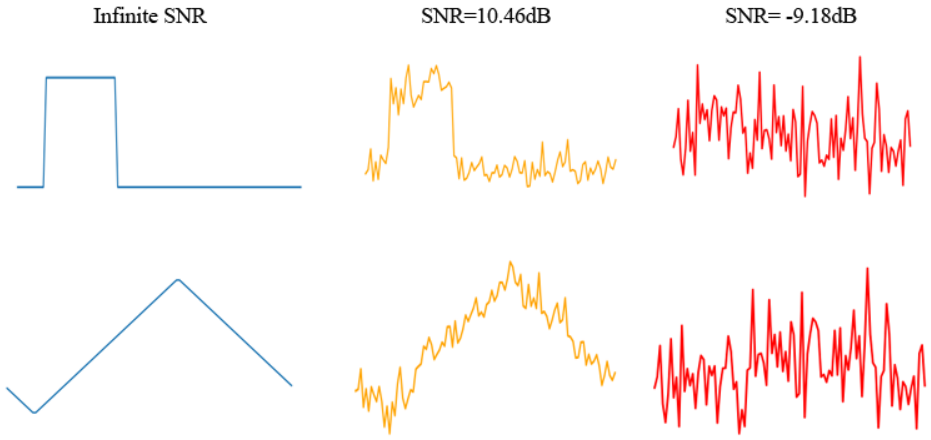


Figure 2: Example of MRA observations at different SNR levels. Each column shows a shifted distinct signal at different noise levels. We can see that for low SNR the task of signal estimation is quite challenging.

For the sake of data classification by the methods we suggest in our project, we define a similarity measure between observations. We use the cross-correlation, as its invariance to shift holds great value:

$$(x \star y)_n \triangleq \sum_{l=\infty}^{\infty} x_l^*(y)_{n+l} \tag{2}$$

The convolution theorem states that the convolution of two signals equals to the inverse Fourier transform of the product of the Fourier transforms of each signal. Thus we can write the equation above in terms of Fourier transforms and later exploit FFT in our simulations.

$$(x \star y)_n = \mathcal{F}^{-1}\{X^* \cdot Y\}_n \tag{3}$$

2.2 Community detection

As part of our suggested solution to the heterogeneity problem, we convert the MRA data into a weighted graph. Each node in the graph represents a different MRA observation, as was defined in **Eq.1**. Each edge connecting a pair of nodes has a weight that represents a similarity measure between the nodes, as was defined in **Eq. 3**.
Our objective is the partition of all MRA observations into K non overlapping classes, where each class represents a distinct signal before it was distorted by **Eq. 1**. Our classification process is carried out using Community Detection (CD) algorithms applied on the produced weighted graph.
Community, in a broad sense, is a set of nodes, which are similar to each other and dissimilar from the rest of the network. Community Detection (CD) algorithms aim to find these communities in a graph. **Fig. 3** shows the result of a CD algorithm.

Finding communities in a graph carries a great value to revealing hidden patterns in data and has many use-cases, such as social behaviour prediction [27] or medical diagnosis[12]. Evidently, Community Detection algorithms are well studied and widely used. In our project we study a group of state-of-the-art CD algorithms and apply them on MRA data.
It is important to note that CD problem is known to be NP-hard[10], meaning that the most efficient CD algorithm that partitions a graph correctly has an exponential time complexity (that is, unless P = NP), making it infeasible to run on large graphs. For this reason, it is common to use *heuristic* algorithms that provide approximate solution to the CD problem, with the advantage of polynomial time complexity.

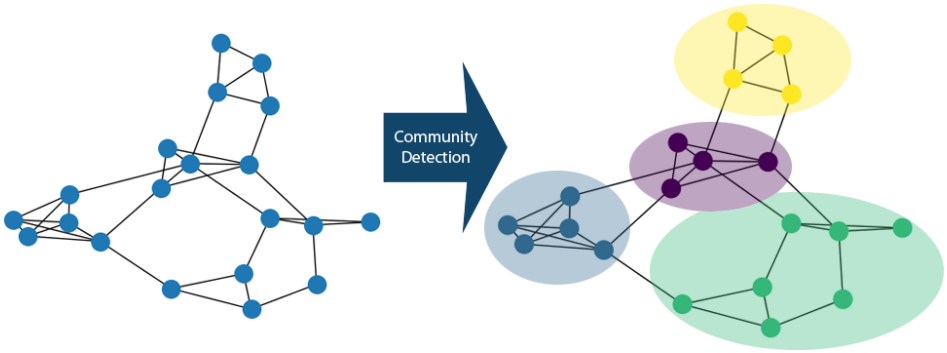


Figure 3: Graph partition by Community Detection. In this example for the sake of simplicity an unweighed graph is used. Note that in our project we used weighted graphs.

Below is a summary of the algorithms we studied.

- *Edge Betweenness*[11] uses the "edge betweenness" as its optimization parameter, which is defined as the number of shortest paths that go through an edge in a graph. Following this definition, edges with higher "betweenness" can be interpreted as connections between different communities. The algorithm computes the "edge betweenness" score for each edge in the graph and iteratively removes the edges with the highest score, while computing the score for each iteration. The end product is a disconnected graph separated into separate communities.

- *Fast Greedy*[15] is based on the idea of *modularity*. Let e_{ij} be the fraction of edges in a graph that connect nodes in group i to those in group j, and let $a_i = \sum_j e_{ij}$. Then modularity is defined as:

$$Q = \sum_i (e_{ii} - a_i^2) \tag{4}$$

  Simply speaking, modularity is the fraction of edges that fall within communities minus the expected value of the same quantity if edges fall at random. Thus, low modularity values indicate a random structure of the graph, while higher values point to deviation from randomness and more defined communities within a graph. The algorithm operates by initializing a community for each node in a graph. Then it repeatedly joins nodes in pairs such that the modularity of the graph is maximized.

- *Label Propagation*[18] exploits the basic notion of a community as a set of similar neighbouring nodes. Each node starts with a unique label, and after each iteration, a node updates its label to the most common label of

its neighbours. The algorithm stops when each node has a label that the maximum number of its neighbours have.

- *Leading Eigenvector*[16] is based on the *spectral clustering*. The basic idea behind spectral clustering is to convert the graph into a *similarity matrix* and find the eigenvectors that correspond to the (second) smallest eigenvalue of the matrix. These eigenvectors define the *minimum cut size* of the graph (cut size is defined as the number of edges running between two groups of nodes). The second smallest eigenvalue is chosen because the smallest correspond to minimum cut size of 0, where the whole graph is a single community.
  Let $\mathbf{A}$ be the *adjacency matrix*:

$$A_{ij} = \begin{cases} 1 & \text{if nodes } (i,j) \text{ are connected} \\ 0 & otherwise \end{cases} \tag{5}$$

  and $\mathbf{P}$, such that P_{ij} is the expected number of edges between nodes i, j given a random graph under the constraint that the expected *degree* (number of edges connected to a node) of each node equal to the degree of each corresponding node in the input graph. *Modularity matrix* is then defined as:

$$\mathbf{B} = \mathbf{A} - \mathbf{P} \tag{6}$$

  Leading eigenvector algorithm uses the modularity matrix as the similarity matrix and proceeds to perform a spectral clustering of the input graph.

- *Walktrap*[17] is based on the idea that short random walks on a graph tend to stay in the same densely connected area. Each node starts in its own community and a random walk process is run on each node. Based on the random walk, distances between nodes are calculated, and nodes with the smallest distances are lumped together. The process then repeats until all nodes are in the same community. At each step modularity (**Eq.** (4)) of the partition is computed, and in the end the partition with the maximum modularity is chosen.

- *Infomap*[20] uses *Huffman code* as a way to compress the information about a path of a random walk in a graph. Each node in a graph is given a unique Id. A random walk that explores the entire graph is initialized, note that random walk tends to stay longer in a densely connected areas. This allows to combine nodes Ids into Huffman codes, that will ultimately label the communities in a graph. The algorithm then optimizes the number of Huffman codes such that the encoded information about the path of the random walk in a graph is maximally compressed.

- *Louvain*[4] maximizes modularity (**Eq.** (4)) in a hierarchical fashion. In the first phase of the algorithm, each node starts in its own community. Each node is then placed in a community with its neighbour such that a maximum modularity score is obtained for the graph. In the second phase a new graph is created, whose nodes are now the communities found during the first phase, such that the weight of the edges between these nodes equal to the edge density between the communities from the first phase. After the second phase is complete, the new graph is passed through the first phase again. The process is repeated until no modularity gain is obtained.

- *Leiden*[25] is an improvement of the Louvain algorithm. Louvain algorithm, in some cases, has proven to produce bad partitions in the form of disconnected communities (a community that contains disconnected nodes). Leiden algorithm introduces a refinement stage in the first phase (see Louvain algorithm). At the second phase, the refined partition is aggregated and initially partitioned based on the unrefined partition. By creating the aggregated graph based on the refined partition, Leiden algorithm has more room for identifying high-quality partitions.
  Simply speaking, when a community is aggregated to a node at the second phase in Louvain algorithm, no further changes inside the community are possible. Leiden, on the other hand, gives more flexibility by refining the graph in the first phase.
  Leiden also uses a faster method for implementing the nodes transitions between communities in the first phase.

- *SpinGlass*[19] adopts ideas from the field of statistical mechanics for the clustering of the graph, and is based on the Potts model [26]. The algorithm regards to the graph as a system with at most of k *spin states* (which are equivalent to the number of communities we wish to find), and tries to minimize the collective *energy* of the system by arranging its *particles* (which are equivalent to the nodes of the graph) into said spin states. The energy of the system is quantified by the *Hamiltonian* function, which is defined for a group of spin states σ_i as:

$$-\sum_{i\neq j} a_{ij}\cdot \underbrace{A_{ij}\delta\sigma_i,\sigma_j)}_{InternalEdges} + \sum_{i\neq j} b_{ij}\cdot \underbrace{(1-A_{ij})\delta\sigma_i,\sigma_j}_{Non-existingInternalEdges}$$
$$+\sum_{i\neq j} c_{ij}\cdot \underbrace{A_{ij}(1-\delta\sigma_i,\sigma_j)}_{ExternalEdges}\sum i\neq j d_{ij}\cdot \underbrace{(1-A_{ij})(1-\delta\sigma_i,\sigma_j)}_{Non-existingExternalEdges}$$

where **A** is the adjacency matrix (**Eq. 5**) of the graph, a_{ij},b_{ij},c_{ij} and d_{ij} are hyper parameters which affect the contribution of each factor to the hamiltonian and can be changed and $\delta(a,b)$ symbolizes Kronecker delta. In order to minimize the hamiltonian, the algorithm also uses simulated annealing[14].

Table 1 summarizes time complexities of each algorithm.

Algorithm	Complexity	Notes
Edge Betweenness	$O(\|E\|\|V\|(\|E\|+\|V\|))$	
Fast Greedy	$O(\|V\|(\|E\|+\|V\|))$	
Label Propagation	$O(\|E\|)$	
Leading Eigenvector	$O(\|V\|(\|E\|+\|V\|))$	
Walktrap	$O(D\|E\|\|V\|^2)$	D the size of the dendrogram
Infomap	$O(t^2\|E\|)$	t maximal number of iterations
Louvain	$O(\|E\|)$	
Leiden	$O(\|E\|)$	Considered to be superior to Louvain, i.e its constant factor is smaller
SpinGlass	$O(\|V\|^{3.2})$	For sparse graphs

Table 1: CD algorithms time complexities. $|E|$ represents the total number of edges in the graph and $|V|$ the total number of nodes (vertices).

2.3 Performance evaluation

In order to compare performance between the different CD algorithms, several criteria of performance evaluation need to be defined:

- **The (asymptotic) running time complexity of each algorithm**, or for the sake of simplicity, 'how many steps the algorithm should do on given inputs until it converges, depending on the input's size'. For this criterion, we used the traditional big-O notation.
- **A scoring function to measure the accuracy of the partition an algorithm yielded**, by measuring the degree of similarity between the partition produced by an algorithm and the correct partition of the graph, i.e. the *ground truth*.
  As was mentioned previously, CD algorithms are heuristic for solving the NP-hard graph partition problem, therefore cannot always cluster the graph perfectly, assuming that P $\neq$ NP. Thus a scoring function can be introduced to provide us a way to quantify the quality of each partition. In our project we chose the Normalized Mutual Information (NMI) as the accuracy score function, as it is considered a good method to measure CD algorithms partition accuracy[8].
  We start by introducing the Mutual Information (MI) scoring function, which incorporates ideas from the information theory. Let $G=(V,E)$ be an unweighed graph and let P be a partition of the graph to M communities $C_1, C_2, \ldots, C_M$. *Shannon entropy* is then defined as

$$H(P) = -\sum_i \frac{|C_i|}{V} log(\frac{|C_i|}{V}) \tag{7}$$

  where $|C_i|$ is the number of nodes in C_i. It is also convenient to define the *conditional Shannon entropy*

$$H(P_1|P_2) = -\sum_{i,j} \frac{|C_i \cap C_j|}{V} log(\frac{|C_i \cap C_j|}{V}) \tag{8}$$

where P_1,P_1 different partitions of G, and $|C_i \cap C_j|$ number of nodes that belong to both C_i,C_j. MI is then defined as:

$$MI(P_1, P_2) = H(P_1) - H(P_1|P_2) \tag{9}$$

One can note that the range of the MI quality function is $[0, \infty)$ making it impractical for comparing many cases. Thus it is useful to define the NMI function:

$$NMI(P_1, P_2) = \frac{2MI(P_1, P_2)}{H(P_1) + H(P_2)} \tag{10}$$

Note that the range of the NMI quality function is $[0, 1]$, where NMI=1 if the partitions are identical and NMI=0 if the partitions are completely different.
A very useful property of the NMI is its independency from the number of communities in each partition, providing us a convenient tool to compare two partitions with different number of communities.

A few important remarks about performance measurement should be noted:

1. **There is no absolute performance metric**, meaning an algorithm's performance may receive a 'good' score by some of the criteria but a 'bad' score by other criteria.
2. **An algorithm performance may depend on the input itself**, that is, an algorithm may perform well on some inputs, but badly on others. In order to reduce this dependability, we should analyze an algorithm's performance by considering the *expectation* of its performance according to each criterion on several random inputs.
3. **Comparison between algorithms should be done on the same inputs exactly**. In other words, it is not enough to run each algorithm on several random inputs separately, and then to compare the results.

2.4 Modified K-means algorithm

In order to examine if CD is beneficial compared to other clustering methods, we compare the CD algorithms performance against a different method of clustering. We present a modified K-means algorithm that takes into account the fact that the MRA data is shifted. We start by introducing the traditional KMeans algorithm[13], a well known clustering algorithm used widely for pattern recognition applications. Let $X = x_1, ..., x_n \in \mathbb{R}^L$ be the set of L-dimensional points to be clustered into K clusters, $C = c_1, ..., c_K$. Each cluster is represented by a centroid μ_k, $k = 1, ..., K$, which is the mean of all points in the cluster. K-means finds a partition of X into K clusters such that the squared error of the distances between the centroids of the clusters and the points in the corresponding clusters is minimized. Therefore, the goal of K-means is to minimize the function in **Eq. 11**.

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - \mu_k||^2 \tag{11}$$

Minimizing the given function is an NP-hard problem [9], thus can only converge to a local minimum.
The algorithm's operation can be summarized in the following steps:

1. Select an initial partition of the data set into K partitions (K is a parameter of the algorithm). In our project we used the Kmeans++ implementation[1]:
   (a) Choose a random point from X to be the first centroid.
   (b) Choose the next centroid μ_k, selecting $\mu_k = x' \in X$ with probability $\frac{D(x')^2}{\sum_{x \in X} D(x)^2}$, where D is distance from the centroid.
   (c) Repeat steps (b) until K centroids were chosen.
2. Assign each point in the data set to its closest centroid, i.e assign x_i to cluster c_k such that $||x_i - \mu_k||^2$ is minimal for $k = 1, ..., K$.
3. Recalculate the centroids of the new clusters by computing the mean of each new cluster.
4. Repeat steps (2) and (3) until cluster membership stabilizes.

Our modified algorithm introduces two adjustments to the algorithm presented above:

1. The distance metric used is the Euclidean distance between pairs of centroids and data points shifted, such that the cross correlation between each data point and centroid is maximal. Simply, data points were first shifted, then the distance was calculated.
2. Centroids were calculated using *template matching*, i.e first sample of each cluster was chosen as the reference point (template) and every other point in the cluster was shifted such that the cross-correlation between each point and the first point is maximal. After the shifting mean is calculated.

For a fixed maximal number of iterations t the time complexity of K-means is $O(ndKt)$ where n is the number of data points, d the dimension of each point and K number of specified clusters.

3 Results

To validate the solution we propose, we performed a number of simulations with the aim of testing CD algorithms on the MRA problem. We used three types of signals from which MRA samples were generated:

- Rectangular and triangle pulses, as illustrated in **Fig. 2**, $K = 2$.
- Normally distributed i.i.d signals, i.e $x_j \sim \mathcal{N}(0, \sigma^2 I)$.
- Correlated normally distributed signals, i.e $x_j \sim \mathcal{N}(0, \Sigma)$, Σ the covariance matrix.

All signals have a length of $L = 50$ and are normalized. Simulations were executed on an Ubuntu 18.04 machine, Intel i5 2.5GHz CPU, 4GB RAM using Python. All CD algorithms were implemented using CDlib and NetworkX libraries.

3.1 Process overview

Fig. 4 shows the full process required to partition a graph generated from MRA data using Community Detection. Thus a key requirement is an efficient implementation of each step in the process to minimize the overhead caused by the graph creation. In our project we used the Numpy python package to implement vectorization methods for cross-correlation calculation and graph generation to ensure graph creation will not be the bottleneck of the solution.

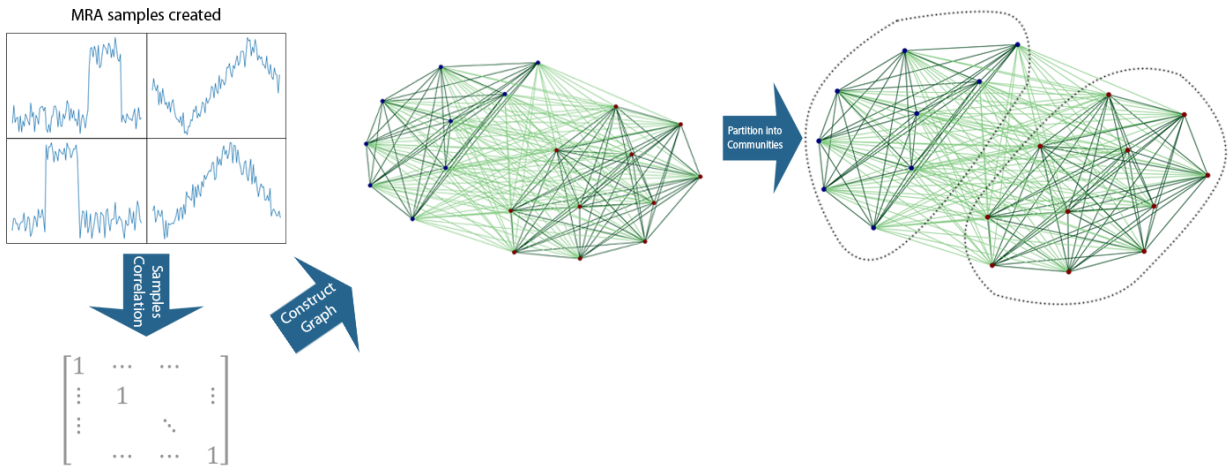
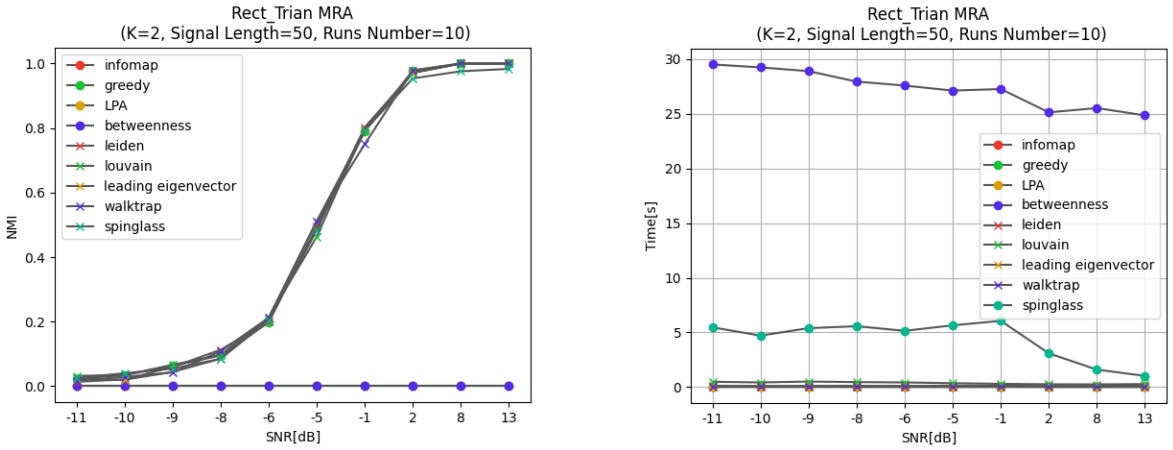


Figure 4: Full process. Full process of graph creation and partition by CD.

3.2 Community Detection best performers selection

Different CD algorithms have different methods to find communities within a graph, some methods are more suitable to the MRA problem than others. We perform a basic simulation on 100 MRA samples generated from rectangle and triangle signals to measure the accuracy of the partitions made against a true partition of the graph. Execution time was also measured to gain an idea about the different algorithms efficiency. **Fig. 5** shows the results of the simulations. Upon observing the clustering quality simulation we conclude that there is a set of algorithms that perform similarly well, while other algorithms fail to partition the data even for high SNR. These algorithms (Fast Greedy, Leiden, Louvain, Leading Eigenvector, Walktrap) are selected as the best performers to be evaluated against KMeans. Spinglass algorithm also partitioned the data quite accurately, but the execution time simulation shows that the algorithm is slower to other algorithms, thus should not be used on large datasets.



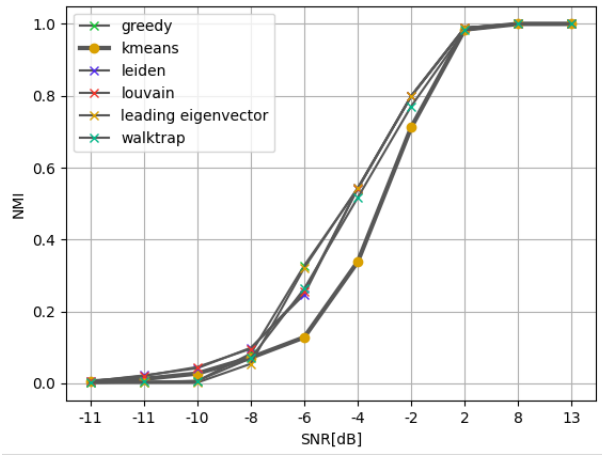
(a) Clustering quality simulation. NMI score of the partitions created by each CD algorithm as a function of the samples SNR.

(b) Execution time simulation. Average execution time of each CD algorithm as a function of the samples SNR.

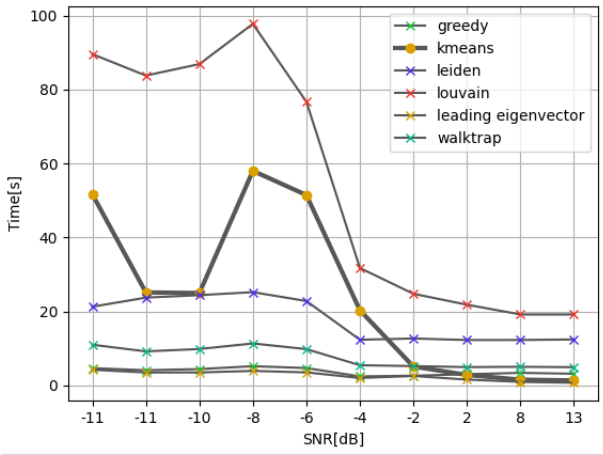
Figure 5: Community Detection algorithms performance evaluation. In both graphs, for each SNR value ten different random graphs were evaluated and the results were averaged.

3.3 Community Detection and KMeans comparison

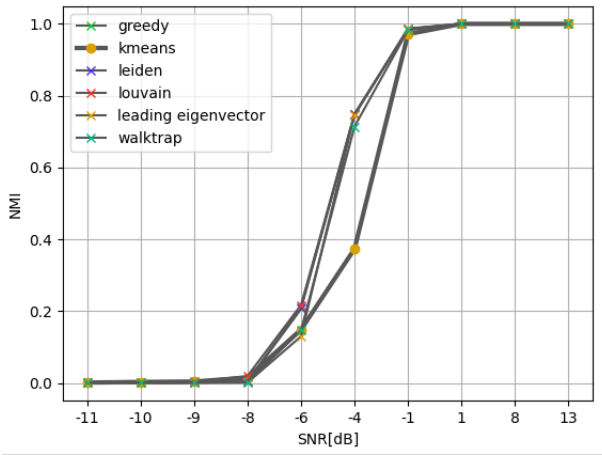
In order to evaluate the effectiveness of CD against other clustering methods we used the modified KMeans algorithm. In the simulations below 1000 MRA samples were generated for different values of K. For each SNR value ten different sets of 1000 samples were generated, and the performances of the algorithms were averaged over the sets. **Fig. 6** shows clustering quality and execution time for $K = 2$, i.e MRA samples were generated from two distinct signals. **Fig. 7** and **Fig. 8** show the clustering quality for higher values of K. Execution times trends didn't change much for higher K, the simulation results can be found in the Appendix.



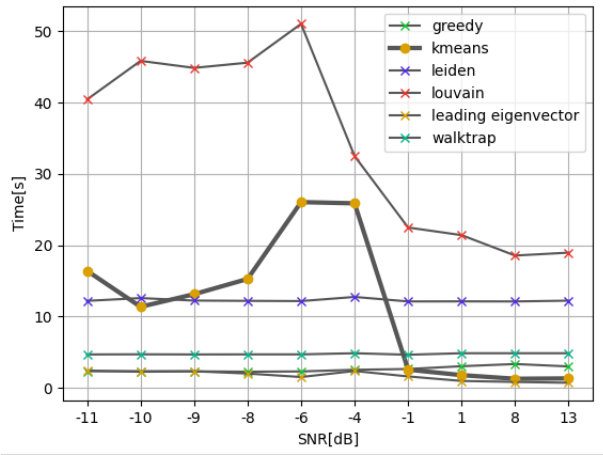
(a) Clustering quality simulation. Rectangle and Triangle.



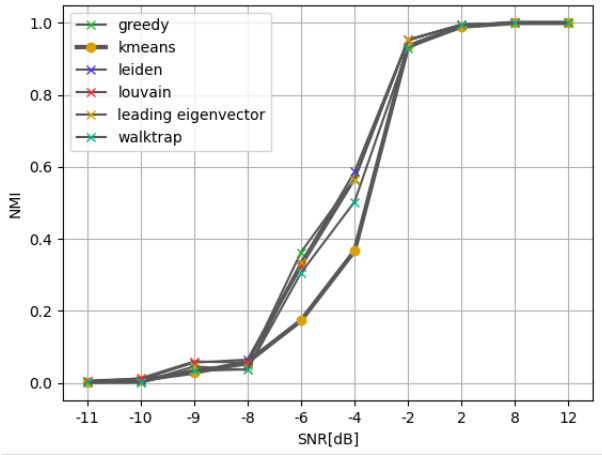
(b) Execution time simulation. Rectangle and Triangle.



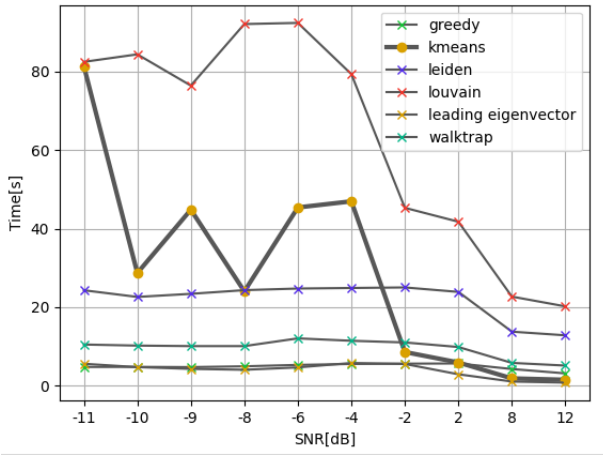
(c) Clustering quality simulation. Normal i.i.d.



(d) Execution time simulation. Normal i.i.d.

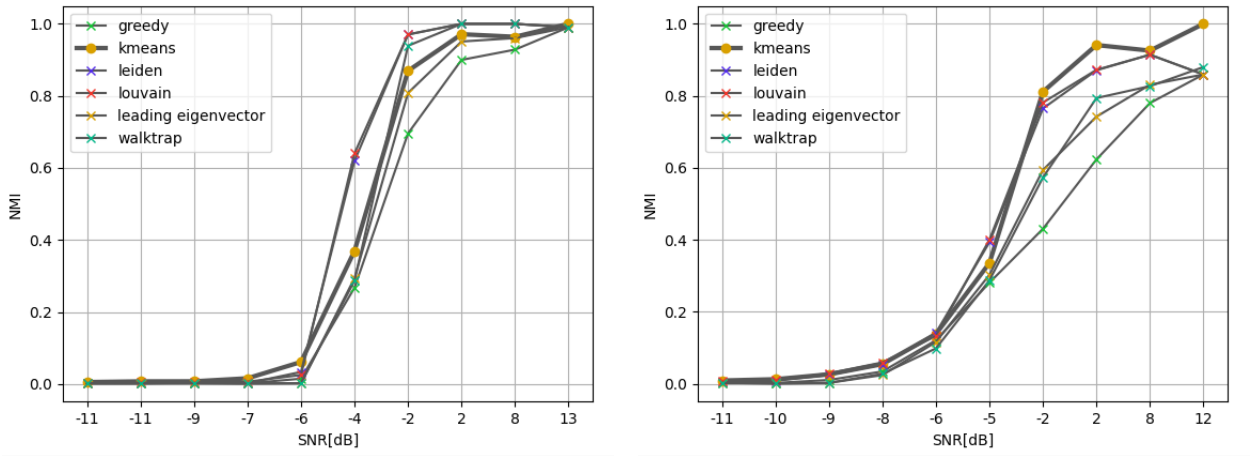


(e) Clustering quality simulation. Correlated normal.



(f) Execution time simulation. Correlated normal.

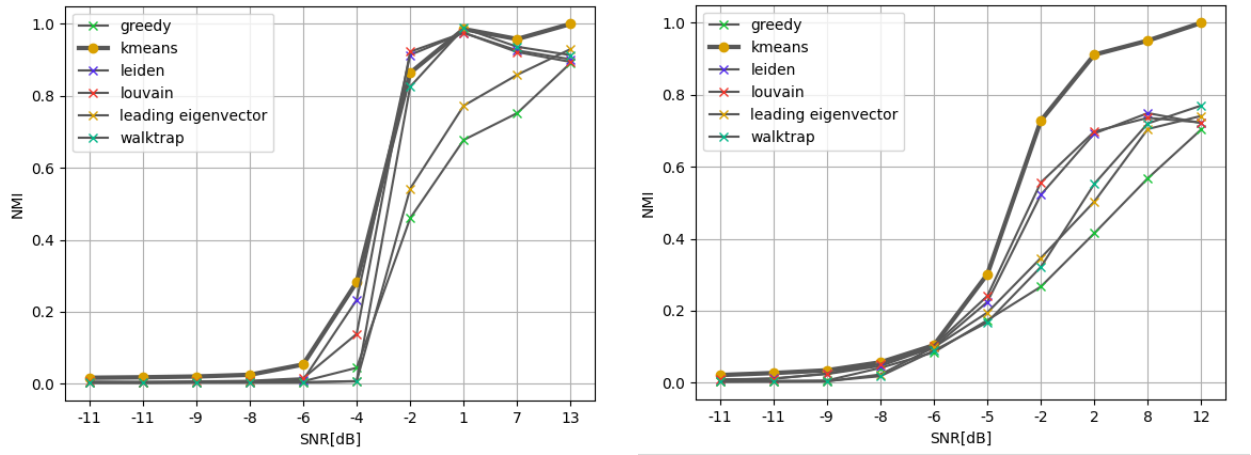
Figure 6: Low K CD and KMeans comparison. $K = 2$



(a) Clustering quality simulation. Normal i.i.d.

(b) Clustering quality simulation. Correlated normal.

Figure 7: Medium K CD and KMeans comparison. $K = 5$



(a) Clustering quality simulation. Normal i.i.d.

(b) Clustering quality simulation. Correlated normal.

Figure 8: High K CD and KMeans comparison. $K = 10$

4 Discussion

The simulations in the previous section presented the clustering quality and execution time performances of both the CD and the modified KMeans algorithms.
In the first part, the best CD algorithms were selected based on their clustering ability and runtime. Even though the algorithms that were chosen use different strategies to find communities within a graph and have different runtime, the communities that were found by the different algorithms were very similar. Infomap, edge betweenness and label propagation have failed to partition the graph even for high SNR. Most likely the edge weights that were used are incompatible with the algorithms operation, further investigation is required.
In the second part, the best CD algorithms were evaluated against the modified KMeans algorithm. For MRA data with low to medium K CD and KMeans algorithms performed comparably well, CD performed even better for low and medium values of K for uncorrelated data, and for low K for correlated data. For high values of K KMeans outperforms the CD algorithms, but not by a high margin for uncorrelated data. Better performance of KMeans was expected, since KMeans receives the number K as a parameter, whereas CD algorithms do not.
In regard to the runtime of the algorithms, KMeans execution time was generally worse than CD, except for Louvain, but Louvain can be eliminated from the selected algorithms and Leiden, which is the improved version, can be used. Generally Leiden was the best performing CD algorithm, with the best overall clustering quality and an adequate runtime.

5 Conclusion

In our project we presented a method of classifying cryo-EM micrographs by using CD to separate data into clusters that represent different molecular conformations. The proposed approach was tested on the simplified 1D MRA model. State-of-the-art CD algorithms were evaluated for clusters quality and runtime against the KMeans algorithm with modifications we made, that account for the characteristics of the MRA data, particularly the fact that the data is shifted. In our simulations we observed that even though KMeans had the advantage of an extra parameter K that defines the number of real clusters, CD algorithms performed comparably well for MRA data generated from uncorrelated signals, and even better for lower values of K. Furthermore, unlike the KMeans algorithm that needed to be modified to account for the input data characteristics, CD algorithms were implemented in a naive fashion with no regard to the type of input that is presented. In summary, even for relatively high values of K, i.e high data heterogeneity, CD algorithms perform well even for relatively low levels of SNR, though when the signals are correlated, performance of CD is poor for high level of heterogeneity and is outperformed by our modified KMeans algorithm.

5.1 Further work

KMeans algorithm modifications we composed use template matching for means computation, which is known to fail even at moderate values of SNR [3]. This can explain the fact that CD algorithms occasionally outperform KMeans, even with the disadvantage of not knowing the correct number of graph partitions. Bispectrum inversion[3] can be used instead of template matching to improve our modified KMeans algorithm performance. It can also be used to improve the performance of CD by extracting a similarity measure from the *invariant features* estimated by the bispectrum inversion method, and use it instead of, or in addition to, the cross-correlation.
Aside from improvements that can be made to partition 1D MRA data, our solution needs to be tested on actual cryo-EM data to conclude if it presents improvement upon the existing methods.

6 References

[1] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[2] Tamir Bendory, Alberto Bartesaghi, and Amit Singer. Single-particle cryo-electron microscopy: Mathematical theory, computational challenges, and opportunities. *IEEE Signal Processing Magazine*, 37(2):58–76, 2020.

[3] Tamir Bendory, Nicolas Boumal, Chao Ma, Zhizhen Zhao, and Amit Singer. Bispectrum inversion with application to multireference alignment. *IEEE Transactions on Signal Processing*, 66(4):1037–1050, Feb 2018.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, Oct 2008.

[5] Nicolas Boumal, Tamir Bendory, Roy R. Lederman, and Amit Singer. Heterogeneous multireference alignment: a single pass approach, 2018.

[6] Saikat Chowdhury, Stephanie A Ketcham, Trina A Schroer, and Gabriel C Lander. Structural organization of the dynein–dynactin complex bound to microtubules. *Nature structural & molecular biology*, 22(4):345–347, 2015.

[7] B. A. Cordier. Ohsu latex dissertation template, 2021. Available on Overleaf.

[8] Leon Danon, Albert Díaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008–P09008, Sep 2005.

[9] Petros Drineas, Alan Frieze, Ravindran Kannan, Santosh Vempala, and V. Vinay. Clustering in large graphs and matrices. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 291–299, 10 2000.

[10] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, Feb 2010.

[11] Michelle Girvan and Mark Newman. Girvan, m. and newman, m. e. j. community structure in social and biological networks. proc. natl acad. sci. usa 99, 7821-7826. *Proceedings of the National Academy of Sciences of the United States of America*, 99:7821–6, 07 2002.

[12] Roger Guimerà and Luís Amaral. Functional cartography of complex metabolic networks. *Nature*, 23:22–231, 01 2005.

[13] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., USA, 1988.

[14] Scott Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science (New York, N.Y.)*, 220:671–80, 06 1983.

[15] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6), Jun 2004.

[16] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3), Sep 2006.

[17] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. *J. Graph Algorithms Appl.*, 10:191–218, 01 2006.

[18] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3), Sep 2007.

[19] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1), Jul 2006.

[20] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, Nov 2009.

[21] Sjors HW Scheres. Beam-induced motion correction for sub-megadalton cryo-em particles. *elife*, 3:e03665, 2014.

[22] Sjors HW Scheres, Mikel Valle, and José-María Carazo. Fast maximum-likelihood refinement of electron microscopy images. *Bioinformatics*, 21(suppl_2):ii243–ii244, 2005.

[23] Fred J Sigworth. A maximum-likelihood approach to single-particle image refinement. *Journal of structural biology*, 122(3):328–339, 1998.

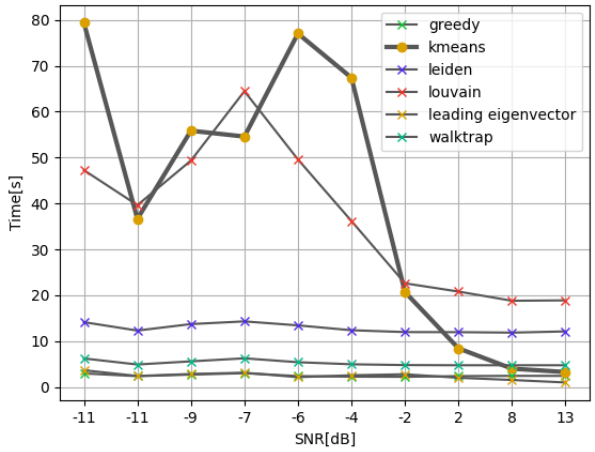
[24] Chun Feng Song, Kostas Papachristos, Shaun Rawson, Markus Huss, Helmut Wieczorek, Emanuele Paci, John Trinick, Michael A Harrison, and Stephen P Muench. Flexibility within the rotor and stators of the vacuolar h+-atpase. *PLoS One*, 8(12):e82207, 2013.

[25] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), Mar 2019.

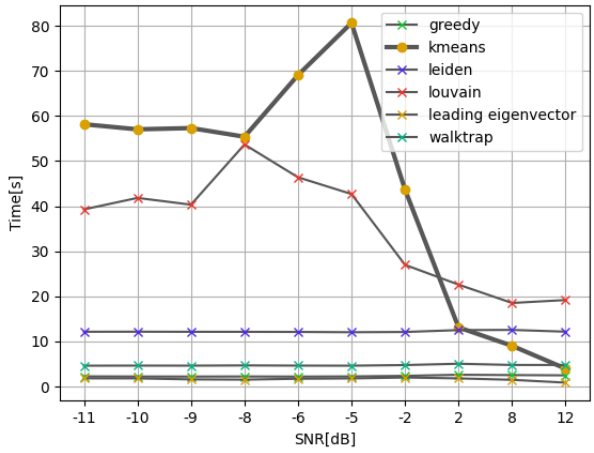
[26] F. Y. Wu. The potts model. *Rev. Mod. Phys.*, 54:235–268, Jan 1982.

[27] Wayne Zachary. An information flow model for conflict and fission in small groups1. *Journal of anthropological research*, 33, 11 1976.

7 Appendix

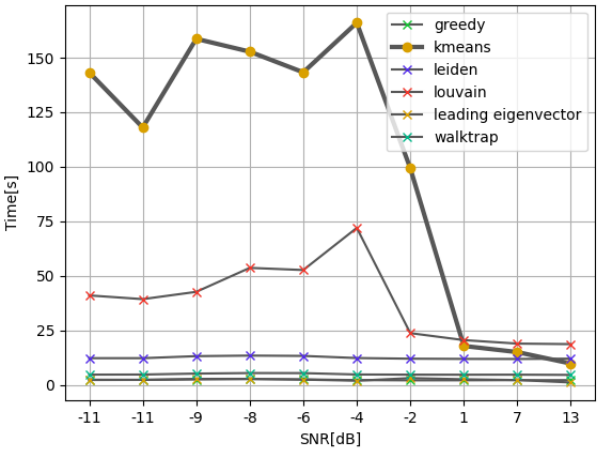


(a) Execution time simulation. Normal i.i.d.

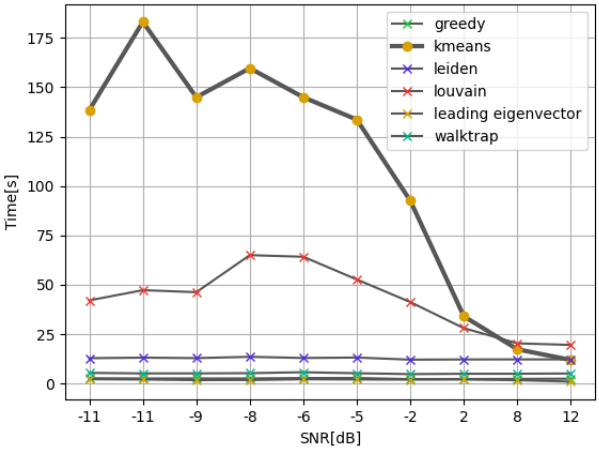


(b) Execution time simulation. Correlated normal.

Figure 9: Medium K CD and KMeans time performances. $K = 5$



(a) Execution time simulation. Normal i.i.d.



(b) Execution time simulation. Correlated normal.

Figure 10: High K CD and KMeans time performances. $K = 10$

The template that was used to make this document was provided by the courtesy of Benjamin Cordier[7].