Minimal-Cost Robust Representation of a Graph Via Reconstruction of the Set of Its Communities

Lukas Mericle & Andrea Ramazzina

March 15, 2018

1 Introduction

We seek to determine the optimal linear combination of a set of subgraphs that together form the community structure of a graph, such that the information loss when moving from one representation of the graph to the other is minimal.

Then we want to examine a number of community detection algorithms to determine which one results in the community structure that, when recombined, results in the least information loss.

Let us define $i = 1 \dots n$ as the index for the nodes in the complete graph and $g = 0 \dots m$ as the index of the graphs. $g = 0$ corresponds to the complete unpartitioned graph, whose name shall be G, and the remaining indices refer to the m subgraphs of G.

Then we seek to reconstruct an analog graph G^* from all subgraphs g which most closely resembles G as determined by the minimum of the information loss between them. This reconstruction is defined by the coefficients associated with each subgraph which are organized into an m-length vector $\vec{\lambda}$.

This reconstruction can then be compared to the original complete graph, such that the information loss corresponds to the information contained in the removed edges. Different community structures will result in different measurements of the information loss, so theoretically there is at least one optimal community structure for a given G.

Our task is to implement the process of determining the maximal-information reconstruction G^* of the graph G and compare how optimally various community detection algorithms define the communities in a graph as measured by the information loss when moving from G to G^*.

We submit code that performs the essential calculations and provides plots for analyzing the output of various community detection algorithms under our framework[1].

2 Theoretical support

2.1 Community detection and graph partitioning

Community detection is the process of determining subsets of a graph G which are densely connected within each subset and sparsely connected between subsets. How these communi-

ties are precisely determined varies between the algorithms implemented to detect them.

Graph partitioning is a specific form of community detection whereby the set of subsets of G forms a partition of G. While in this section we treat both community detection and graph partitioning algorithms, on the experimental part the focus will be exclusively on the latter.

2.2 Converting graphs to Markovian form

Let us assume a directed multigraph with arbitrary non-negative weights on the edges. To convert such a graph to a Markov process, we simply normalize the outgoing edges of each node. In other words, we convert the outgoing edges into transition probabilities $q_{ij} = e_{ij}/\sum_i e_{ij}$, where e_{ij} is the weight on the edge from node j to node i. Assuming each node represents a unique state of a system described by such a Markov process, we arrive at a Markov model, which will have some stationary probability distribution over the nodes.

The adjacency matrix A is the matrix that contains the entries e_{ij}.

2.3 Determining the stationary distribution of a Markovian graph

Modeled as a Markov process, we can recast the adjacency matrix describing a graph as a Markov transition matrix $\widetilde{P} = (A^\star)^T$, where $A^\star$ is the adjacency matrix of the graph after each column has been normalized such that $1 = \sum_{i=1}^{n} e_{ij}\ \forall j$.

Assuming a unique stationary distribution exists, the stationary distribution is then the row vector $\vec{\pi}$ that is the leading left eigenvector of the transition matrix $\widetilde{P}$: we want $\vec{\pi}$ such that it satisfies

$$\vec{\pi}\widetilde{P} = \vec{\pi} \quad \Rightarrow \quad \widetilde{P}^T\vec{\pi}^T = A^\star\vec{u} = \vec{u}, \tag{1}$$

where $\vec{u} = \vec{\pi}^T$. So the leading right eigenvector of $A^\star$ is the stationary distribution of the Markov process as a column vector. This can be computed efficiently for a sparse matrix using the power method.

2.4 Relating a graph to its subgraphs

To compare a complete graph to a combination of its subgraphs, we must consider the probability of occupying node i in each graph when it is interpreted as a Markov process. Explicitly, let us define $P_G = \{p_{0i}\}_i$ and $P_{G^*} = \{\sum_{g=1}^{m} \lambda_g p_{gi}\}_i = \{p_{*i}\}_i$. That is, the probability of occupying node i in G^* is a linear combination of the probabilities of occupying node i in all of the subgraphs of G. Note that $p_{gi} \neq 0$ for exactly 1 subgraph of G when the set of subgraphs is a partition (*i.e.*, $g_a \cap g_b = \varnothing\ \forall a \neq b, a = 1 \ldots m, b = 1 \ldots m$).

For this relation to be relevant for G and its subgraphs, $\sum_{g=1}^{m} \lambda_g = 1$, since the probability of occupying all nodes in each of G, G^*, and every subgraph g must be unity.

2.5 Deriving an optimization scheme for $\vec{\lambda}$

When we cast a graph as a Markov process, we can apply information-theoretic measures to the probability distributions of occupying the nodes of the resulting Markov model. In partic-

ular, since we are concerned with information loss, we apply the Kullback-Leibler divergence to the two probability distributions to quantify the loss.

This is a constrained optimization problem, so a natural choice for the optimization scheme is the method of Lagrange multipliers.

There are two ways to apply the Kullback-Leibler divergence to two probability distributions, and they are not symmetric with respect to one another, although as we will see later they tend to be highly correlated for this application. One way, $K[P_G; P_{G^*}]$, addresses the information introduced by specifying the community structure, and can be seen as quantifying the amount of information added about the relationships between nodes that belong to the same community. The other way, $K[P_{G^*}; P_G]$, addresses the information introduced by adding the edges that bring us from the community structure of the graph to the complete graph, and can be seen as quantifying the information contained in the connections that, when removed, develop the community structure.

2.5.1 From complete graph to reconstructed graph

Let us define the Lagrangian of our problem

$$\begin{aligned} L &= K\left[P_G; P_{G^*}\right] + (\mu+1)\left(1 - \sum_{g=1}^{m} \lambda_g\right) \\ &= \sum_{i=1}^{n} p_{*i} \log \frac{p_{*i}}{p_{0i}} + (\mu+1)\left(1 - \sum_{g=1}^{m} \lambda_g\right), \end{aligned} \tag{2}$$

where $p_{*i} = \sum_{g=1}^{m} \lambda_g p_{gi}$.

The solution to our problem then is the choice of $\vec{\lambda}$ that minimizes the information loss $K\left[P_G; P_{G^*}\right]$ subject to the normalization constraint. Applying the method of Lagrange multipliers (namely solving the set of $m+1$ equations consisting of $\partial L/\partial\lambda_g = 0,\ g = 1 \ldots m$ and $\partial L/\partial(\mu+1) = 0$),

$$\begin{aligned} \frac{\partial L}{\partial \lambda_g} &= \left[\sum_{i=1}^{n} p_{gi} \log \frac{p_{*i}}{p_{0i}} + p_{*i} \frac{p_{0i}}{p_{*i}} \frac{p_{gi}}{p_{0i}}\right] - (\mu+1) \\ &= \sum_{i=1}^{n} p_{gi} \log \frac{p_{*i}}{p_{0i}} + \sum_{i=1}^{n} p_{gi} - (\mu+1) \\ &= \sum_{i=1}^{n} p_{gi} \log \frac{p_{*i}}{p_{0i}} - \mu = 0. \end{aligned} \tag{3}$$

2.5.1.1 Optimizing under graph partitioning constraints

Under graph partitioning, $p_{gi} \neq 0$ for only one g, so the first term of Equation (3) is zero except when we are considering the nodes comprising subgraph g. Then we are only interested in summing over the nodes in g. We can thus reduce p_{*i} to the single term relevant for

subgraph g, giving

$$\frac{\partial L}{\partial \lambda_g} = \sum_{i \in g} p_{gi} \log \frac{\lambda_g p_{gi}}{p_{0i}} - \mu = 0. \tag{4}$$

Note that the first term in Equation (4) is equivalent to $K[\frac{P_G}{\lambda_g}; P_{G^*}]_{i \in g} \geq 0$, thus $\mu \geq 0$. Another way to consider Equation (4) is

$$\begin{aligned} \frac{\partial L}{\partial \lambda_g} &= \sum_{i \in g} p_{gi} \left(\log \lambda_g + \log \frac{p_{gi}}{p_{0i}} \right) - \mu \\ &= \log \lambda_g \sum_{i \in g} p_{gi} + \sum_{i \in g} p_{gi} \log \frac{p_{gi}}{p_{0i}} - \mu \\ &= \log \lambda_g + K\left[P_G; P_g\right] - \mu \\ &= \log \lambda_g + \Delta I_g - \mu = 0, \end{aligned} \tag{5}$$

where ΔI_g can be understood as the difference in information content between G and only the single subgraph g. ΔI_g is well-defined given all p_{gi}. Then all that remains is to determine the value of μ to exactly specify $\vec{\lambda}$.

We rearrange the m constraints into the form

$$\lambda_g = e^{\mu} e^{-\Delta I_g} = e^{\mu} e^{-K[P_G; P_g]} = e^{\mu} Q_g \tag{6}$$

Then we apply the normalization constraint $\sum_{g=1}^{m} \lambda_g = 1$ to get

$$1 = e^{\mu} \sum_{g=1}^{m} Q_g \quad \Rightarrow \quad \mu = -\log \sum_{g=1}^{m} Q_g. \tag{7}$$

Putting it all together, we find the explicit formula

$$\lambda_g = e^{\mu} Q_g = \frac{e^{-K[P_G; P_g]}}{\sum_{l=1}^{m} e^{-K[P_G; P_g]}}, \tag{8}$$

which looks a lot like the Gibbs distribution, suggesting that the formulation we are using preserves the maximum entropy assumption.

Therefore we can conclude that, given a full specification of the stationary probability distribution over the nodes for G and each subgraph g, Equation (8) provides the means to exactly compute the weight of each subgraph in the reconstructed graph G^* that minimizes the information loss.

The total information loss is

$$\begin{aligned} \Delta I = K\left[P_G; P_{G^*}\right] &= \sum_{i=1}^{n} p_{*i} \log \frac{p_{*i}}{p_{0i}} \\ &= e^{\mu} \sum_{i=1}^{n} \left(\sum_{g=1}^{m} Q_g p_{gi} \right) \log \frac{e^{\mu} \sum_{g=1}^{m} Q_g p_{gi}}{p_{0i}} \\ &= e^{\mu} \sum_{i=1}^{n} \left(\sum_{g=1}^{m} Q_g p_{gi} \right) \left(\mu + \log \frac{\sum_{g=1}^{m} Q_g p_{gi}}{p_{0i}} \right). \end{aligned} \tag{9}$$

2.5.1.2 Optimizing under community detection constraints

When deriving a relation for λ_g under community detection, we can no longer assume that $p_{gi} \neq 0$ for exactly one g. Letting $p_{*i} = \sum_{l=1}^{m} \lambda_l p_{li}$ and starting from Equation (3),

$$\begin{aligned} \frac{\partial L}{\partial \lambda_g} &= \sum_{i=1}^{n} p_{gi} \log \frac{p_{*i}}{p_{0i}} - \mu = 0 \\ \rightarrow \mu &= \sum_{i \in g} p_{gi} \log \frac{p_{*i}}{p_{0i}} = \sum_{j \in h} p_{hj} \log \frac{p_{*j}}{p_{0j}} \quad \forall g, h. \end{aligned} \tag{10}$$

Then for a given realization of p_* that is not the optimal one, there will be some residual inequality between the two calculated quantities for μ using subgraphs g and h. Let us define a cost function

$$H\left(\vec{\lambda}\right) = \frac{1}{4} \sum_{g,h} \left(\mu_g - \mu_h\right)^2 = \frac{1}{2} \sum_{g<h} \left(\mu_g - \mu_h\right)^2 \tag{11}$$

to minimize this residual, where $\mu_g = \sum_{i \in g} p_{gi} \log \frac{p_{*i}}{p_{0i}}$ is the value of μ calculated using only the probabilities from subgraph g.

Because an analytical expression for $\vec{\lambda}$ is difficult to find, we use an iterative optimization method instead. So using Equation (11), we can derive a formula for performing gradient descent to find the optimal $\vec{\lambda}$. We propose a simple gradient descent algorithm with two steps: a traversal step followed by a normalization step. We find the gradient of the cost function,

$$\frac{\partial H}{\partial \lambda_k} = \sum_{g<h} \left(\mu_g - \mu_h\right) \left(\rho_{gk} - \rho_{hk}\right), \tag{12}$$

$$\rho_{gk} = \frac{\partial \mu_g}{\partial \lambda_k} = \sum_{i \in (g \cap k)} p_{gi} \frac{p_{0i}}{p_{*i}} \frac{p_{ki}}{p_{0i}} = \sum_{i \in (g \cap k)} \frac{p_{gi} p_{ki}}{p_{*i}} \tag{13}$$

Using this definition, we can apply Algorithm 1 until convergence to produce an approximation for the optimal $\vec{\lambda}$. It is unknown whether $H(\vec{\lambda})$ is convex on the surface defined by the normalization constraint, so it is possible that multiple trials with random initial conditions may be needed to find the global optimum. In our limited tests, we found that the algorithm usually converges to the analytical result for graph partitions.

Algorithm 1 Gradient descent for minimizing information loss by finding an optimal $\vec{\lambda}$.

while $\|\vec{\lambda}^{(t)} - \vec{\lambda}^{(t-1)}\| > T$ **do**
  for $k = 1 \ldots m$ **do** ▷ Traversal
    $\lambda_k^{(t+1)} \leftarrow \lambda_k^{(t)} - \eta \frac{\partial H}{\partial \lambda_k}$
  $Q \leftarrow \sum_{g=1}^{m} \lambda_g^{(t+1)}$
  for $k = 1 \ldots m$ **do** ▷ Normalization
    $\lambda_k^{(t+1)} \leftarrow \lambda_k^{(t+1)} / Q$
  $t \leftarrow t + 1$

The convergence condition of this algorithm implies that the surface defined by the constraint will be approximately perpendicular to the gradient of the cost function when the algorithm terminates (assuming a reasonable parameter value for T, the tolerance of error). The method of Lagrange multipliers solves for the same condition, albeit by different means. Indeed the structure of the problem is partially encoded by our Lagrangian defined in Equation (2).

The stability of this algorithm, while not rigorously confirmed, will nonetheless be reasonable as long as η is kept small.

2.5.2 From reconstructed graph to complete graph

Following the logic of the previous sections, we define the Lagrangian

$$\begin{aligned} L &= K\left[P_{G^*}; P_G\right] - \mu\left(1 - \sum_{g=1}^{m} \lambda_g\right) \\ &= \sum_{i=1}^{n} p_{0i} \log \frac{p_{0i}}{p_{*i}} - \mu\left(1 - \sum_{g=1}^{m} \lambda_g\right). \end{aligned} \tag{14}$$

Then

$$\begin{aligned} \frac{\partial L}{\partial \lambda_g} &= \left[\sum_{i=1}^{n} p_{0i}\left(-\frac{1}{p_{*i}} p_{gi}\right)\right] + \mu \\ &= -\sum_{i=1}^{n} \frac{p_{0i} p_{gi}}{p_{*i}} + \mu = 0. \end{aligned} \tag{15}$$

2.5.2.1 Optimizing under graph partitioning constraints

Under graph partitioning, $p_{gi} \neq 0$ for exactly one g, so the first term of Equation (15) is zero except when we are considering the nodes comprising subgraph g. Then we are only interested in summing over the nodes in g,

$$\begin{aligned} \frac{\partial L}{\partial \lambda_g} &= -\sum_{i \in g} \frac{p_{0i} p_{gi}}{\lambda_g p_{gi}} + \mu \\ &= \mu - \sum_{i \in g} \frac{p_{0i}}{\lambda_g} = 0 \\ \lambda_g \mu &- \sum_{i \in g} p_{0i} = 0 \end{aligned} \tag{16}$$

In order to find an explicit expression of μ, we now sum Equation (15) over all the graph partitions g and apply the normalization and probability-unity constraints,

$$\begin{aligned} \sum_{g=1}^{m}\left(\lambda_g \mu - \sum_{i \in g} p_{0i}\right) = \mu\left(\sum_{g=1}^{m} \lambda_g\right) - \sum_{g=1}^{m}\left(\sum_{i \in g} p_{0i}\right) = \mu - 1 = 0 \\ \Rightarrow \mu = 1 \end{aligned} \tag{17}$$

It is now possible to derive an explicit formulation for any λ_g substituting such result in Equation (17)

$$\lambda_g = \sum_{i \in g} p_{0i} \tag{18}$$

Interestingly, λ_g for any partition of the graph is simply the cumulative stationary probability of all the nodes belonging to that partition in the original graph. This makes intuitive sense but is gratifying to derive analytically.

2.5.2.2 Optimizing under community detection constraints

The optimization routine is identical to that in the above section except

$$\mu_g = \sum_{i \in g} \frac{p_{0i} p_{gi}}{p_{*i}} \tag{19}$$

$$\rho_{gk} = - \sum_{i \in (g \cap k)} \frac{p_{0i} p_{gi} p_{ki}}{p_{*i}^2} \tag{20}$$

3 Methods

The two graph reconstruction approaches will now be examined. Within this scope, different partition algorithms will be used on several complex networks, both synthetic and real. What we would expect is different approaches to behave either qualitatively or quantitatively different depending on the partitions and the structure of the graph.

3.1 Community Detection Algorithms

In order to split the graphs under analysis into their partitions, three community detection algorithms have been employed. The choice not to restrict ourselves to only one approach comes from our initial suspicion that in fact different algorithms would cluster the network in slightly different ways and such differences can comport a different loss of information.

Therefore, the main question in analysis now is: Is there a correlation between the choice of the community detection algorithm and the quantity of lost information? And if so, how can such result be explained?

3.1.1 Louvain Method

The Louvain Method is one of the most well-known and used community detection algorithms [2]. It is based on the optimization of the modularity, a network measure that quantifies the division of the graph into clusters. In particular, the modularity of a (weighted) network is

$$Q = \sum_{i,j \in \{1,..,N\}^2} \left(\frac{A_{i,j}}{2m} - k_i k_j \right) \delta_{c_i, c_j}, \tag{21}$$

where

- A is the adjacency matrix of the graph,
- m is the total number of edges present in the graph,
- k_i is equal to the sum of the weights of all the edges linked to the node i,
- δ_{c_i,c_j} is the Kronecker delta function that is 1 if the two nodes i & j are in the same cluster and 0 otherwise.

Such quantity gets maximized through an iterative approach, in which for each node of the network the increase of modularity ΔQ is computed by moving the node in analysis from its community to a neighbouring one. This process is repeated until there is no more possible increase in the modularity.

3.1.2 Map equation framework

In the map equation approach, a completely different approach is adopted. The main idea relies on the concept of constructing a 'codebook' with the aim of minimizing the description length of the movement of a random walker traveling through the network [3]. The principle changing variable defining such a codebook is the partition of the graph into any number of communities; in fact, it is expected that once entered in, a random walker would remain in a region for several following steps. More specifically, instead of attributing a unique index for each node of the graph, dividing it into modules it can be possible to reuse shorter indexes for multiple nodes, thus lowering the mean description length of the movement of a random walker.

The map equation is able to express such (optimal) value as a function of the partition without actually having to derive such a codebook. For a general case, it can be written as

$$L(M) = q_e H(Q) + \sum_{i=1}^{m} p_i H\left(P^i\right) \tag{22}$$

where

- $H(Q)$ is the expected value of the codeword length in the index codebook,
- q_e represents the rate at which the index codebook is used and it is given by the sum of the probabilities of exiting the modules,
- $H\left(P^i\right)$is the expected value of the codeword length in the ith module's codebook,
- p_i is the rate at which the ith module's codebook is used and it is given by the sum of the probabilities of visiting the nodes of the module plus the probabilities of exiting this module.

Again, the problem of partitioning the graph into communities has been reduced to an iterative optimization problem. In this study a stochastic and recursive search algorithm is used as provided by the `Infomap` package, and mainly revolves around the idea of agglomerative clustering and repeating this process for different hierarchies.

3.1.3 Semi-synchronous label propagation method

The label propagation method is a novel community detection algorithm based on the broader Label Propagation algorithm which uses the spreading of information in the network to identify the communities[4].

First, each node is assigned to a different label, *i.e.*, a different community. Subsequently, through an iterative method each node's label can be changed depending on the label of the nodes in its neighbourhood. More specifically, given a node a it will update its label l_a according to the rule

$$l_a = argmax_l \sum_{u \in N(a)} [l == l_u] \tag{23}$$

where $N(a)$ is the neighbourhood of the node a and the notation [] is the Iverson bracket. In case of more than one maximum (for example at the beginning of the algorithm), one of them is randomly chosen.

This process is repeated until a stop condition is reached, such as stability of the communities. The updating of the labels can be executed in different ways, such as changing the labels synchronously or asynchronously. The algorithm here uses a semi-synchronous approach, in which a color is assigned to each node so that nodes with the same color do not have any influence on each other, and the label updating is executed one color at a time.

3.2 Test networks

Such algorithms have been employed on a variety of networks, different both in topology, size and origin. More specifically, there are three of the most famous and common archetype of graphs, i.e. random graphs (Erdős-Rényi), small-world and scale-free networks (obtained through the preferential attachment procedure). Note that some of them do not present at all a communities-like structure and this will strongly affect our result, nonetheless it has been decided to include such examples to have a more holistic view of the problem in analysis.

Furthermore, three real-world networks are taken,

- US power grid: High-voltage power grid in the Western States of the United States of America, where the nodes are transformers, generators and substations while the edges are high-voltage transmission lines,
- Jazz musicians network: Network of jazz musicians, where the nodes are the single musicians,
- Rome99: Directed network describing the road network of Rome, Italy, part of the 2006 Uniroma challenge

3.3 Graph measures

In order to observe how the structure of a graph or the way in which it is partitioned affects the resulting loss of information, the main approach carried is to visualize such quantities against different characteristic measures,

- The number of clusters per node,

- The coverage of a partition, that is, the number of intra-community edges divided by the total number of edges present in the graph [5]; it is used to assess the quality of a partition,
- The density of a graph (independently from the partition). Such measure for directed graphs is $d = \frac{m}{n(n-1)}$, where n is the number of nodes and m is the number of edges in G.

Note also that in all the plots the Kullback–Leibler divergence is also normalized by the number of the nodes, such that the quantity is the information lost *per symbol*. This is due to the fact that such quantities appear to steadily grow as the network size increases. Such observation is coherent with the underlying functioning of the community detection algorithms, which tend to subdivide the network in bigger partitions as this grows larger, thus making it more likely to incur a loss of information.

4 Results

The three presented community detection algorithms are now used on a set of different graphs, both synthetic (random, small-world and preferential-growth) and real. Subsequently, for each community partition the stationary probabilities of the nodes are reconstructed using the two information loss minimization approaches (from complete graph to reconstructed graph and vice versa). The results of our analysis are found in Figure 1.

First, the results of these two information calculation methods are compared. Plotting the Kullback–Leibler divergences corresponding to the complete-to-reconstruction approach against the reconstructed-to-complete one, it is clear how there is actually a strong linear correlation between the two results, with the information loss of the latter only slightly larger for bigger values.

Interestingly, although the two methods differ on how the stationary probabilities are computed, the results are often very similar and this similarity brings unequivocally to almost identical Kullback–Leibler divergences.

Thanks to such strong correlation, in the further analysis only the results from the reconstruction-to-complete approach are taken, in order to simplify the comparisons. The choice of this measure over the other one is because it is more explicative of how much information we get from knowing the totality of the graph from its partition. The second question in analysis is whether the choice of the community detection algorithm affects the amount of information loss in the partitioning of the graphs. As can be seen in the lower panel, although the information loss might noticeably vary, it is not possible to clearly identify a better-performing algorithm or detecting the change of magnitude of the results.

However, excluding the particular cases of the random and the preferential growth networks, overall it appears that the map equation approach brings usually to a relatively low amount of information lost. This observation is coherent with the underlying functioning of this information-theoretic approach. Nonetheless, it has to be noted that for certain networks, the finer partitions obtained through this method are actually dividing in really small partitions (as noticeable in the top right graph) and this over-partitioning obviously lowers the information lost, losing at the same time any meaning, since each node belongs to its

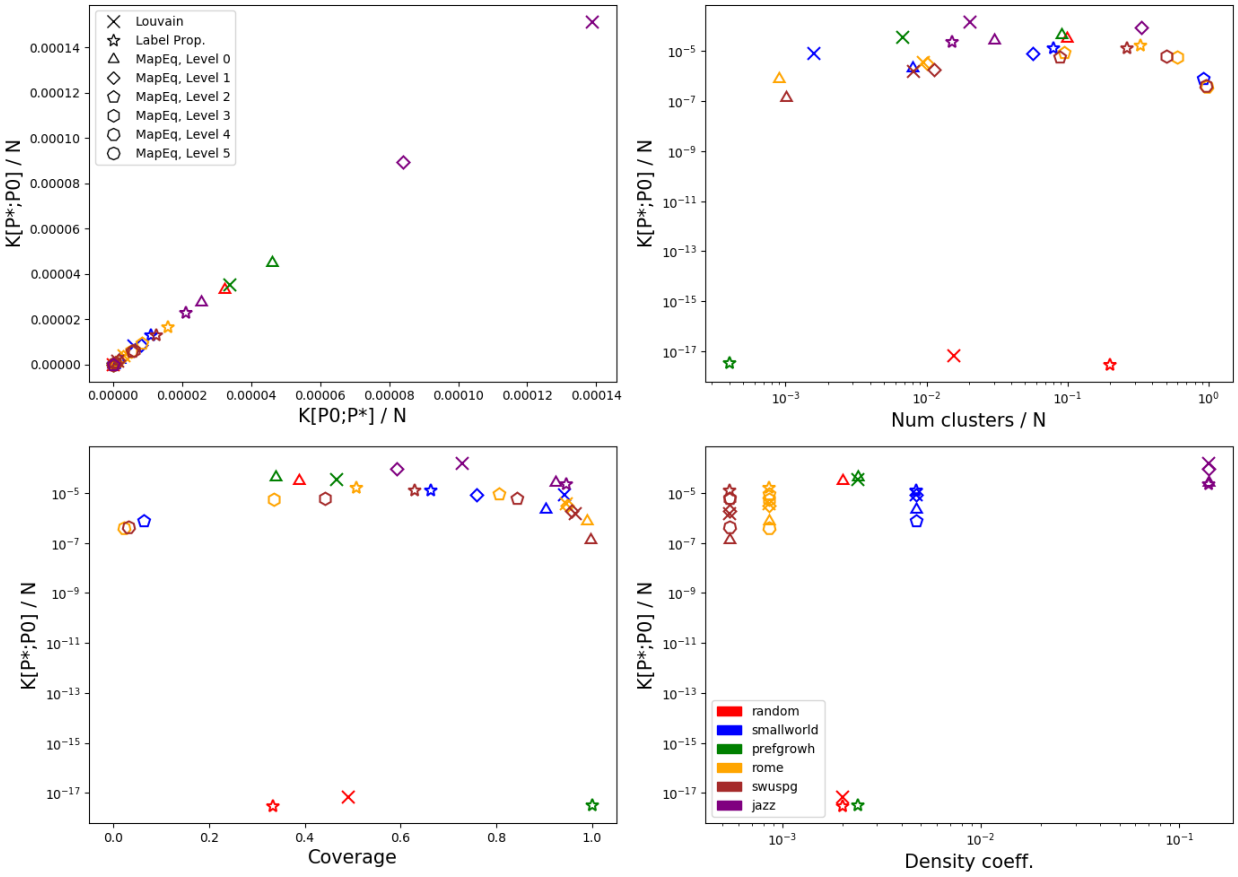

Figure 1: Top left: the Kullback–Leibler divergences per node corresponding to the complete-to-reconstruction approach against the reconstructed-to-complete. Top right: the latter Kullback–Leibler divergence per node as function of the number of clusters per node. Bottom left: the Kullback–Leibler divergence per node against the coverage measure of the relative partition covergence measure. Bottom right: Kullback–Leibler divergence per node and the density of the graphs.

own partition or inhabits one with only a couple other nodes. This particular behaviour is in fact pointed out in the bottom left graph, where these cases have a low coverage measure. This brings us to the final point of our analysis, that is the research of particular influences of the structure of the network or of the partitions on the amount of information loss.

As previously mentioned, a first distinction between the random and preferential growth graphs and the rest has to be made. In fact, these two networks are inherently characterized by a total absence of clusters, thus different community detection algorithms will have a great difference in outputs with a consequent divergence of results regarding the information loss; however due to the nature of these graphs such partitions are not meaningful and neither are the subsequent results.

Looking at the bottom right graph, it appears that as the average density increases, the different partition algorithms tend to bring to a more similar loss of information, although

as it can be seen in the top right and bottom left graphs they visibly differ in functioning. Furthermore, it seems that for low values of density the Louvain algorithm is the worst performing algorithm.

Another interesting observation is that the better results are obtained when the number of clusters and coverage are either very low or very high. The fine-partitioning case has already been discussed, while the case of big communities might be explainable by the fact that only splitting the network into a few partitions might not greatly affect the stationary probability of the nodes.

5 Conclusions

The scope of this project was to derive an algorithm to determine the optimal linear combination of a set of partitions of a graph such that the information loss when moving from the complete graph to the combination of subgraphs is minimal. Furthermore, it was in the interest of this work to determine how different community detection algorithms would perform in such a framework and if the underlying structure of the graph would affect the total amount of information loss in the partitioning. Overall, we managed to derive with a rigorous mathematical method two formulations of such information-loss minimization graph reconstruction, and we showed how actually these two methods produce similar results. Furthermore, analyzing the results of different community detection algorithms on graphs of different natures, certain aspects of these algorithms and of the inherent characteristics of the networks have been observed as affecting the resulting loss of information.

6 Further Investigations

6.1 Emphasis on overlapping community structure

First of all, a further study might focus on the case where communities overlap and how this eventuality would change the results of information loss, as the graph reconstruction algorithm gets more complicated and one could expect more information loss. Furthermore, it might be interesting to investigate the reasons behind the different results obtained through the partitioning algorithms in an information-theoretic framework.

6.2 Extension to clustering of arbitrary data

One can extend such a community structure quality assessment to clustering of arbitrary N-dimensional data by recasting the dataset as a graph. Specifically we would find an undirected weighted complete graph whose edge weights correspond to the inverse of the distance between observations in the dataset according to some distance metric. Then the above analysis can be run without modification and the quality of that particular cluster structure can be assessed.

7 Contribution Report

Overall, we have been often working together, mainly in the first parts of the work in which we focused on the analytical derivations.

Lukas then focused on the theory underlying our work to extract neat and correct solutions and on coding and integrating parts for the simulations.

Andrea delineated the analysis procedure and the interpretations of the output, along with the choice of the community detection algorithms.

References

[1] A. Ramazzina and L. Mericle. Minimal-cost robust representation of a graph via reconstruction of its partition. `https://github.com/andrearama/graph-partition-reconstruction`, 2018.

[2] V. D. Blondel, J. L. Guillaume, and R. Lambiotte. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10:12, 2008.

[3] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *Eur. Phys. J. Special Topics*, 178:13–23, 2010.

[4] G. Cordasco and L. Gargano. Community detection via semi-synchronous label propagation algorithms. *Business Applications of Social Network Analysis (BASNA)*, pages 1–8, 2010.

[5] S. Fortunato. Community detection in graphs. *Physical Reports*, 486:75–174, 2009.