LUISS

Department of Business and Management

Bachelor's Degree in Management and Computer Science

Course of Artificial Intelligence and Machine Learning

Graduation thesis - developed in collaboration with Poste Italiane

AI-fostered Graph Analysis:

Network Clustering and Representation Learning

Implementation in a business context

Prof. Giuseppe Francesco Italiano
SUPERVISOR

Dott. Lorenzo Gangemi
COMPANY TUTOR

Matteo Spadaccia
CANDIDATE (ID 277141)

Academic year 2023/2024

Abstract

Nowadays, information sets of various types are increasingly often conceivable in graph form: for implementations in disparate fields, data mining is substantially based on social networks, Internet of Things, and similar technologies characterized by a highly spread interconnectivity. Moreover, even when information sources are not intrinsically scattered, the graph-encryption of correlation features gets useful in many contexts to efficiently analyze huge data amounts.

Another consistent and renowned characteristic of the digital era is indeed the increasing trend in data quantities. The latter are physiologically produced as always more daily actions are accomplished at least indirectly exploiting some electronic device; however, the same information volumes must also be properly stored and – most importantly despite less intuitively – very wisely processed in order to derive any type of related benefit.

Since the wide network paradigm is pressingly invading the data-driven environment we technically live in, Machine Learning is eccentrically definable as the complementary element integrating a hopefully virtuous circle. In fact, AI algorithms are both fundamental tools for interconnected metadata creation (e.g., enabling text and speech mining) and principal solutions to obtain value from overwhelming information amounts. Nevertheless, ML's specific adaptability to graph analysis tasks is still not a fully examined issue at all.

Therefore, the present paper aims to summarize the current state of the art on the subject of networks study via Artificial Intelligence techniques, with particular reference to graph clustering and representation learning algorithms. Illustrative strategies of the main different types are discussed and compared through academic papers review; subsequently, the same methods are put at cooperation in analyzing a delivery-logistic sub-graph granted by Poste Italiane, to better consider AI instruments' mastery in examining process-related networks.

Table of contents

0 – Introduction

0.1 – Context

In this section, an extremely concise presentation of the two main topics at issue is proposed.

0.1.1 – Artificial Intelligence

Defined as the set of programming techniques enabling computers to mimic human intelligence, AI includes two main subsets of methods: Machine Learning – implying a statistics-fostered and experience-based performance improvement – and Deep Learning – gathering the most advanced ML strategies which expose multilayer Neural Networks to vast data amounts. These kinds of algorithms typically present many powerful capabilities, among which high adaptability and comparably fast solving skills over huge datasets and complex problems, sometimes even not faceable at all through traditional programming.

A fundamental taxonomy in the field is based on such systems' source of data used for optimization: supervised learning involves training models on labeled information and is effective for classification and regression tasks in particular; conversely, unsupervised learning deals with unlabeled data to find hidden patterns based on less explicit instructions. Key algorithms in the latter domain include clustering ones, aiming to group together similar objects while separating them from dissimilar ones without any a-priori known set of categories. Other nuances are represented by semi-supervised methods – utilizing a small amount of labeled input along with unlabeled data – and Reinforcement Learning – involving a decision-making agent's interaction with an environment to autonomously find a reward-maximizing decisional strategy.

0.1.2 – Network structures

In the fields of systems modeling and data science, graphs are fundamental structures composed of elements (i.e. nodes) interconnected by links (i.e., edges) and representing a relationship diagram of whichever determined type between a set of entities. Networks may either include a single kind of elements and links, or display nodes and edges of disparate sort in one interconnection framework only. In the latter case, a so-called heterogeneous graph is built; networks of this type might be applied in the structured computationally-navigable representation of very complex realities (a.k.a. knowledge graph), integrating information from diverse sources and enabling the inference of novel awareness by prevision of not yet detected connections.

Graph analysis precisely involves studying networks to uncover the patterns, properties, and structures they retain, encompassing disparate techniques – including community detection – ultimately aimed at understanding the organization and dynamics of the underlying interconnected systems. In this domain, networks can be categorized based on direction, weight, and edges' multiplicity: undirected graphs have bidirectional connections, while directed ones distinguish links' direction; weighted networks assign non-binary values representing strength, capacity, or cost to the edges, instead of simply recording their presence; multigraphs permit multiple independent edges between the same node couples.

Basic theoretical definitions

$\delta(S^1, S^2)$: Kronecker delta. It equals 1 if set S^1 corresponds to set S^2, otherwise it equals 0.

$\delta_x(S^1, S^2)$: Inverse Kronecker delta. It equals 0 if set S^1 corresponds to set S^2, otherwise it equals 1.

$|S|$: Set's cardinality. It returns the number of elements contained in set S.

S^c: Complementary set. It contains all the relevant elements not included in set S.

$M_{n\times m}$: nxm-shaped matrix. M^T: Matrix transposition. It reflects M along its main diagonal.

$\boldsymbol{G} \equiv (V, E)$: Graph. It includes a set of nodes (a.k.a. vertices) V and a set of edges E representing a network.

A: Adjacency matrix. It contains the interconnection data between an unweighted graph's node couples.

A_{ij}: Unweighted edge. The value representing a connection (1) or its absence (0) from node i to node j.

X: Traffic matrix. It contains the interconnection data between a weighted graph's node couples.

X_{ij}: Weighted edge. The intensity of the connection from node i to node j.

n: Number of nodes. $\equiv |V|$ m: Number of edges. $\equiv |E|$ T: Total traffic of the graph. $\equiv \sum_{i,j} X_{ij}$

N_i: Set of neighbors of node i. N_i': Set of neighbors of node i and node i itself. $\equiv N_i \cup \{i\}$

k_i^{in}: In-degree of node i. It corresponds to the number of edges arriving at node i.

k_i^{out}: Out-degree of node i. It corresponds to the number of edges departing from node i.

k_i: Degree of node i. It corresponds to the total number of edges related to node i. $\equiv |N_i| = k_i^{in} + k_i^{out}$

t_i: Traffic output of node i, corresponding to the total traffic from node i to other nodes. $\equiv \sum_{j\in V} X_{ij}$

$p(S)$: probability that an arbitrarily selected node belongs to set S.

$p(S^1, S^2)$: probability that an arbitrarily selected node belongs to both set S^1 and set S^2.

0.2 – AI-fostered graph analysis

Many Artificial Intelligence concepts are suitably applicable in studying networks; the two methods of main relevance which will be furtherly discussed are hereafter introduced.

0.2.1 – Network clustering

(Salcedo-Sanz, Naldi and Carro-Calvo)

Comprising a wide set of unsupervised Machine Learning techniques, network clustering consists in the partition of a graph's nodes into cohesive groups (i.e., clusters) with the aim of favoring intra-cluster relationships and minimizing the strength of inter-cluster ones. These vertices' affinities are conceivable both as common characteristics – reducing the clustering process to a plain analysis of nodes' attributes in samples-format – and as implicitly-computed relationships deriving from the connectivity structure itself.

The identification of a significant clustering model allows deeper understanding of networks at whole-framework, community, and single-entities levels. Therefore, algorithms with this aim find applications in disparate domains of the data analysis realm, emerging as powerful pattern-uncovering tools to gain consistent and decision-making valuable insights on intricated real-world interconnection scenarios.

For instance, network clustering is used on social networks to pinpoint communities – thus influential individuals and opinion leaders through critical node detection; these methods are extremely useful in social dynamics understanding and marketing prognostic analysis. Similar algorithms are applied to train the recommendation systems which daily enhance near everyone's consumer experience (e.g., via discovery and relevant-ads features) by grouping similar people, items or services based on user behavior.

Moreover, network clustering strategies are widely exploited in scientific studies too, especially through knowledge graphs; in particular, chemistry and biology have accomplished incredibly rapid advances due to AI-aided analyses. Protein-protein interaction and gene expression networks have indeed been computationally represented and their functional modules identified via proper algorithms, leading researchers to understand the relevant functions and interactions in an unprecedentedly complete and efficient way.

Another clustering's application is network security: in this field, strategies to detect patterns are helpful in identifying fraudulent activities (e.g., money laundering in financial webs) as anomalies and malicious actors as outliers, providing data to assist and automate safety maintenance and crime prevention processes. However, graph clustering also finds a role in many other fields, as long as the deduction of valuable conclusions from a significantly huge dataset of interconnected entities is implicated.

Basic theoretical definitions

C_i': Cluster to which node i is assigned. It includes the nodes inside the model-obtained cluster of i, i included.

C_i: Set of nodes in the model-obtained cluster of i, excluding node i itself. $\equiv\ C_i' \setminus \{i\}$

$\boldsymbol{C}$: Clustering. It is the set of nodes' clusters. $\equiv\ \{C_a, \dots, C_z\} \mid \forall C,\ \bigcup_{C_i \in \boldsymbol{C}} C_i = V \wedge C_i \cap C_j = \emptyset, \forall C_i, C_j \in \boldsymbol{C}$

Y_i': Set of all nodes labeled like node i. It includes the nodes in the ground-truth class of i, i included.

Y_i: Set of nodes in the ground-truth class of i, excluding node i itself.

$\boldsymbol{Y}$: Ground-truth set of nodes' classes. $\equiv\ \{Y_a, \dots, Y_z\} \mid \forall Y,\ \bigcup_{Y_i \in \boldsymbol{Y}} Y_i = V \wedge Y_i \cap Y_j = \emptyset, \forall Y_i, Y_j \in \boldsymbol{Y}$

SS: Couples of nodes clustered together both by model and ground-truth. $\equiv\ \sum_{i,j \in V, i \neq j} \left[\delta\left(C_i', C_j'\right)\delta\left(Y_i', Y_j'\right)\right]$

DD: Couples of nodes clustered aside both by model and ground-truth. $\equiv\ \sum_{i,j \in V, i \neq j} \left[\delta_x\left(C_i', C_j'\right)\delta_x\left(Y_i', Y_j'\right)\right]$

SD: Couples of nodes clustered together by model but not by ground-truth. $\equiv \sum_{i,j \in V, i \neq j} \left[\delta\left(C_i', C_j'\right)\delta_x\left(Y_i', Y_j'\right)\right]$

DS: Couples of nodes clustered together by ground-truth but not by model. $\equiv \sum_{i,j \in V, i \neq j} \left[\delta_x\left(C_i', C_j'\right)\delta\left(Y_i', Y_j'\right)\right]$

$map(C_i')$: Best permutation function from cluster C_i' to the most comparable label-based set Y_i';
it might be found through the Kuhn-Munkres algorithm. *(Plummer and Lovász)*

0.2.2 – **Representation learning**

(Mavromatis and Karypis)

Graph representation learning algorithms focus on encoding non-Euclidean graph-structure information – typically high-dimensional data – and eventually nodes' or edges' attributes into a low dimensional space. Through these methods, a set of embeddings is learnt which should represent the original graph's main characteristics and thus might be useful in various subsequent tasks (e.g., nodes classification, link prediction).

An unsurprisingly primary sequel opportunity is precisely clustering: the aforementioned pre-elaborated information might be precious as already sifted input for typical grouping algorithms. The latter may indeed benefit from a prior identification of data distributions' principal components, particularly when the original sample is wide or intricately represented and labels are not yet defined.

1 – State of the art

In this section, various currently valid techniques for network clustering and representation learning are defined; their theoretical grounds are thus discussed coherently with the respective academic description.

1.1 – Clustering metrics

A series of indices are below described which might be useful in assessing clusterings' quality and characteristics, always applying a traffic-aware approach to describe algorithms' performance both in simply topology-based contexts and when dealing with real non-negative edges' weights.

1.1.1 – Graph-based

Measures studying clusterings' quality with respect to the relevant network structure are firstly presented.

(Cov) – **Coverage**
(Khanfor, Nammouchi and Ghazzai)

Coverage compares the weighted fraction of all intra-cluster edges to the total number of edges in the graph. In particular, it is defined as:

$$Cov(\boldsymbol{C}) = \frac{\sum_{i,j \in \boldsymbol{V}} X_{ij}\, \delta(C_i', C_j')}{\sum_{i,j \in \boldsymbol{V}} X_{ij}} \quad \Rightarrow \quad 0 \leq Cov(\boldsymbol{C}) \leq 1$$

Therefore, the higher the coverage, the better the graph topology is clustered.

(SCM) – **Scaled Coverage Measure**
(Laura, Naldi and Italiano)

For each node i, the local Performance (a.k.a. Scaled Coverage Measure) is computed as:

$$SCM_i(\boldsymbol{C}) = 1 - \frac{|N_i'^c \cap C_i| + |N_i \cap C_i'^c|}{|N_i \cup C_i|} = 1 - \frac{|NNBR_i| + |XNBR_i|}{|N_i \cup C_i|}$$

($NNBR_i$: i-non-neighbors in the same cluster as i. $XNBR_i$: i-neighbors not in the same cluster as i.)

The global SCM is then defined as the average of the local Performances:

$$SCM(\boldsymbol{C}) = \frac{1}{n} \sum_{i \in \boldsymbol{V}} SCM_i(\boldsymbol{C}) \quad \Rightarrow \quad 0 \leq SCM(\boldsymbol{C}) \leq 1$$

To maximize the SCM, each node should be clustered only with its neighbors, minimizing the number of intra-cluster non-neighboring nodes as well as of extra-cluster neighboring nodes. The standard Scaled Coverage Measure is an established method for topology-based clustering, but not meant for traffic-based contexts.

(TS) – **Traffic-Aware SCM**

(Laura, Naldi and Italiano)

Bypassing the binary concept of neighboring nodes, the Traffic-aware Scaled Coverage Measure considers nodes' closeness as a real-valued variable in the $[0,1)$ range. In particular, V_{ij} represents the degree of closeness of node i to node j (thus, in general, $V_{ij} \neq V_{ji}$, since traffic flows are typically not symmetric) and is derived from a logistic transformation of the traffic intensity X_{ij}:

$$V_{ij} = \frac{2}{1 + e^{-\alpha X_{ij}}} - 1 \quad | \quad V_{ij} \in [0,1) \ \wedge \ V_{ij} = 0 \Leftrightarrow X_{ij} = 0 \ \wedge \ \lim_{X_{ij} \to \infty} V_{ij} = 1$$

The coefficient α may be determined to set the correspondence of a specific $X_{\bar{\imath}\bar{\jmath}}$ towards its $V_{\bar{\imath}\bar{\jmath}}$:

$$\alpha = \frac{1}{X_{\bar{\imath}\bar{\jmath}}} \ln\left(\frac{1 + V_{\bar{\imath}\bar{\jmath}}}{1 - V_{\bar{\imath}\bar{\jmath}}}\right)$$

For example, based on the traffic distribution's quantiles to assess an α respecting the above properties of V's elements, $V_{\bar{\imath}\bar{\jmath}}$ might be set to 0.9 at $X_{\bar{\imath}\bar{\jmath}}$ being the 90-percentile value among the traffic intensities leaving i.

Since the objective is favoring clusterings which group closer nodes while avoiding to group nodes that are not close, TS employs two error indicators revealing respectively the quantity of inclusions in a cluster of non-close nodes and of exclusions from a cluster of close nodes. In particular, for each node i:

$$W_i(\boldsymbol{C}) = \sum_{j \in C_i} \left(1 - V_{ij}\right) \ \wedge \ Z_i(\boldsymbol{C}) = \sum_{j \notin C_i'} V_{ij} \quad \Rightarrow \quad W_i(\boldsymbol{C}) \leq |C_i'| - 1 \ \wedge \ Z_i(\boldsymbol{C}) < n - |C_i'|$$

Thus, for node i, the local and global Traffic-aware Scaled Coverage Measure are defined as:

$$TS_i(\boldsymbol{C}) = 1 - \frac{W_i(\boldsymbol{C}) + Z_i(\boldsymbol{C})}{n - 1} \quad \Rightarrow \quad 0 < TS_i(\boldsymbol{C}) < 1 \quad \Leftarrow \quad W_i(\boldsymbol{C}) + Z_i(\boldsymbol{C}) < n - 1$$

$$TS(\boldsymbol{C}) = \frac{1}{n} \sum_{i \in \boldsymbol{V}} TS_i(\boldsymbol{C}) \quad \Rightarrow \quad 0 < TS(\boldsymbol{C}) < 1$$

(Q) – **Modularity**

(Newman, Analysis of weighted networks)

With A-represented unweighted directed networks, Modularity analyzes the presence of each intra-cluster edge of the graph by considering the probability for that edge to exist in a random graph; it is defined as:

$$Q(\boldsymbol{C}) = \frac{1}{2m} \sum_{i,j \in \boldsymbol{V}} \left[\left(A_{ij} - \frac{k_i k_j}{2m}\right) \delta\left(C_i', C_j'\right)\right] \quad \Rightarrow \quad -1 \leq Q(\boldsymbol{C}) \leq 1$$

In the case of X-represented weighted directed networks, Modularity compares instead the weight of each intra-cluster edge of the graph and the one expected from the random distribution of traffic over all possible origin-destination pairs. Thus, it is defined as:

$$Q(\boldsymbol{C}) = \frac{1}{2T} \sum_{i,j \in \boldsymbol{V}} \left[\left(X_{ij} - \frac{t_i t_j}{2T}\right) \delta\left(C_i', C_j'\right)\right] \quad \Rightarrow \quad -1 \leq Q(\boldsymbol{C}) \leq 1$$

(AC) – **Average Conductance**

(Yang and Leskovec)

Clusters' Average Conductance considers the internal and outgoing traffic amounts of the clusters as follows:

$$AC(\boldsymbol{C}) = \frac{1}{|\boldsymbol{C}|}\sum_{C\in\boldsymbol{C}}\frac{\sum_{i,j\in\boldsymbol{V}|i\in C,j\notin C} X_{ij}}{2\sum_{i,j\in\boldsymbol{V}|i,j\in C} X_{ij} + \sum_{i,j\in\boldsymbol{V}|i\in C,j\notin C} X_{ij}} \quad \Rightarrow \quad 0 \le AC(C) \le 1$$

Note that a high AC value implies poor clustering quality: sensible partitioning leads to a low Conductance.

1.1.2 – Label-based

A second set of metrics is discussed, useful to evaluate clusterings by comparison with another partitioning.

(Acc) – **Accuracy**

(Liu, Zhu and Li)

Expressing the exactness of a clustering with respect to some labeling, Accuracy is defined as:

$$Acc(\boldsymbol{C},\boldsymbol{Y}) = \frac{1}{n}\sum_{i\in\boldsymbol{V}}\delta(map(C_i'), Y_i') \quad \Rightarrow \quad 0 \le Acc(\boldsymbol{C},\boldsymbol{Y}) \le 1$$

(J) – **Jaccard coefficient**

(Amigó, Gonzalo and Artiles)

Considering all the nodes pairs' label attribution, J indirectly conveys the similarity of $\boldsymbol{C}$ to $\boldsymbol{Y}$:

$$J(\boldsymbol{C},\boldsymbol{Y}) = \frac{SS}{SS + SD + DS} \quad \Rightarrow \quad 0 \le J(\boldsymbol{C},\boldsymbol{Y}) \le 1$$

(FM) – **Folkes and Mallows score**

(Amigó, Gonzalo and Artiles)

Similarly, FM might be applied to assess a clustering exactness with respect to a given ground-truth:

$$FM(\boldsymbol{C},\boldsymbol{Y}) = \sqrt{\frac{SS}{SS\ +\ SD}\cdot\frac{SS}{SS\ +\ DS}} \quad \Rightarrow \quad 0 \le FM(C,Y) \le 1$$

(ARI) – **Adjusted Rand Index**

(Manning)

Being a basic measure of similarity between two partitions, the Rand Index is defined as:

$$RI(\boldsymbol{C},\boldsymbol{Y}) = \frac{\sum_{i,j\in\boldsymbol{V},i\neq j}\left[\delta(C_i',C_j')\delta(Y_i',Y_j') + \delta_x(C_i',C_j')\delta_x(Y_i',Y_j')\right]}{2\binom{n}{2}} \quad \Rightarrow \quad 0 \le RI(\boldsymbol{C},\boldsymbol{Y}) \le 1$$

$$\Downarrow$$

$$E[RI(\boldsymbol{C},\boldsymbol{Y})] = \left(\sum_{C\in\boldsymbol{C}}\binom{|C|}{2}\sum_{Y\in\boldsymbol{Y}}\binom{|Y|}{2}\right)/2\binom{n}{2}$$

Interpretable as label-based score, the Adjusted Rand Index compensates RI's sensitivity to chance:

$$ARI(\boldsymbol{C},\boldsymbol{Y}) = \frac{\sum_{C\in\boldsymbol{C},Y\in\boldsymbol{Y}}\binom{|C\cap Y|}{2} - 2E[RI(\boldsymbol{C},\boldsymbol{Y})]}{\frac{1}{2}\left(\sum_{C\in\boldsymbol{C}}\binom{|C|}{2} + \sum_{Y\in\boldsymbol{Y}}\binom{|Y|}{2}\right) - 2E[RI(\boldsymbol{C},\boldsymbol{Y})]}$$

(NMI) – **Normalized Mutual Information**

(Liu, Zhu and Li)

When comparing two nodes' partitions, Mutual Information is defined as:

$$MI(\boldsymbol{C},\boldsymbol{Y}) = \sum_{C\in\boldsymbol{C},Y\in\boldsymbol{Y}} \left[\frac{|C\cap Y|}{n}\log_2\frac{n|C\cap Y|}{|C||Y|}\right]$$

Entropy is instead a measure of a cluster's diversity based on a given labeling:

$$H(C,\boldsymbol{Y}) = \sum_{Y\in\boldsymbol{Y}}\left[\frac{|C\cap Y|}{|Y|}\log_2\frac{|C\cap Y|}{|Y|}\right] \quad\Rightarrow\quad H_{tot}(\boldsymbol{C},\boldsymbol{Y}) = \sum_{C\in\boldsymbol{C}}\frac{|C|}{n}H(C,\boldsymbol{Y})$$

Exploiting the above concepts, the Normalized Mutual Information metric is computable as:

$$NMI(\boldsymbol{C},\boldsymbol{Y}) = \frac{MI(\boldsymbol{C},\boldsymbol{Y})}{\max\left(H_{tot}(\boldsymbol{C},\boldsymbol{Y}),H_{tot}(\boldsymbol{Y},\boldsymbol{C})\right)}$$

(F1) – **F1 score**

(Amigó, Gonzalo and Artiles)

Set matching metrics are based on a bijective relation between clusters and categories, relying on the concepts of Precision and Recall as defined below:

$$Precision(C,Y) = \frac{|C\cap Y|}{|C|} \qquad Recall(C,Y) = Precision(Y,C) = \frac{|Y\cap C|}{|Y|}$$

Taking into consideration the most common category of each cluster, Purity is computed as:

$$Purity(\boldsymbol{C},\boldsymbol{Y}) = \sum_{C\in\boldsymbol{C},Y\in\boldsymbol{Y}|map(C)=Y}\frac{|C|}{n}Precision(C,Y)$$

This metric clearly penalizes clusters' noise without rewarding the grouping of same-category elements.

On the opposite, focusing on each category's maximum-recall cluster and thus not penalizing the mix of different-categories items, Inverse Purity is defined as:

$$Inverse\ Purity(\boldsymbol{C},\boldsymbol{Y}) = \sum_{C\in\boldsymbol{C},Y\in\boldsymbol{Y}|map(C)=Y}\frac{|Y|}{n}Recall(C,Y)$$

The F1 score, obtainable combining the above metrics by harmonic mean, is a more robust indicator:

$$F1(\boldsymbol{C},\boldsymbol{Y}) = \sum_{C\in\boldsymbol{C},Y\in\boldsymbol{Y}|map(C)=Y}\frac{|Y|}{n}\frac{2\cdot Recall(C,Y)\cdot Precision(C,Y)}{Recall(C,Y)+Precision(C,Y)} \quad\Rightarrow\quad F1(A,B)\not\equiv F1(B,A)$$

1.2 – **Network clustering algorithms**

The landscape of graph clustering techniques is large and varied, but certain strategy types are more commonly used in practice; for each of them, the basic concepts and a significant example is hereafter reported.

1.2.1 – **Vector quantization**

Used primarily in signal processing and data compression, vector quantization involves approximating a large set of high-dimensional vectors via a smaller representative one (i.e., the codebook); thus, data complexity is reduced by mapping each original vector to the nearest one in the just-produced set.

(K) – **K-Means**

(Laura, Naldi and Italiano)

Being the most widely used vector-quantization method due to its simplicity and adaptable efficiency, K-means aims to divide data into a specific user-defined number of clusters k. The process begins randomly selecting k initial centroids from the dataset, data points are then assigned to the nearest centroid by a specified similarity measure, generating a first clustering. Afterwards, centroids are iteratively recalculated as averages of their cluster's data points, replacing the previous partition. The sequential updating process continues until the centroids stabilize; thus, they are returned as relevant codebook together with the last data points' assignment. While optimized versions of the algorithm ensure K-means' scalability to large datasets, its sensitivity to the initial centroids' selection is intrinsic and might cause convergence to local optima.

Algorithm: K-means (K)

Input: n featured vectors, the desired clusters' number k.

Output: A partition of the n vectors into k clusters, the relevant codebook of k centroids.

1. Initialization: randomly select k initial centroids from the n vectors.
2. Iterate the following steps until convergence (i.e., centroids stabilization):
    2.1. Assignment: compute each vector's distances from centroids, label it under the nearest centroid.
    2.2. Update: compute the location of each cluster's centroid as the mean of its member vectors.
3. Output the final centroids and the partition of the n vectors into k clusters based on these centroids.

1.2.2 – **Label Propagation**

Initializing each node in a different community, Label Propagation methods lean on edges-dependent probability values to iteratively spread the labels between neighboring nodes; the effect is a stable convergence of the vertices into interconnected clusters, based on the specific iteratively applied local-update rule. Besides being fairly scalable, this type of algorithm presents a distinctive output format highlighting 'exemplar' nodes which may represent the clusters they initiated and propagated through direct conflict with other labels.

(ALP) **– Asynchronous Label Propagation**

(Raghavan, Albert and Kumara)

Constituting the most naïve method of its kind, ALP involves updating nodes one at a time in a random order and examining the frequency of their neighbors' labels with a random tie-breaker strategy. Despite possibly leading to a lower rigor compared to synchronous algorithms' executions, asynchrony – understood as independent update of each node with respect to the others – engenders a linear time complexity, making this LP version particularly advantageous for large networks. Moreover, this non-deterministic method clearly demonstrates consistent adaptability to the network's structure without prior clusters' number specification.

Algorithm: Asynchronous Label Propagation (ALP)

Input: A graph $G = (V,E)$.

Output: A partition of the graph's nodes into communities.

1. Initialization: Assign to each node a unique label (e.g., its own identifier).
2. Iterate the following steps until convergence (i.e., nodes having the most frequent label among neighbors):
    2.1. Randomly shuffle the order of nodes.
    2.2. For each node i in the shuffled order, eventually update its label:
        2.2.1. Retrieve the labels of all the neighbors of i and identify the most frequent.
        2.2.2. Assign this most frequent label to node i, breaking ties randomly if necessary.
3. Output the final labels of the nodes, where each label represents a community.

1.2.3 – Modularity maximization

Presenting a scalability strictly dependent on the specific Q-optimizing method, Modularity-maximization strategies base on the connectivity framework to sort nodes into clusters which display a high within-clusters and a low extra-clusters relationships' concentration compared with a random distribution.

(L) **– Louvain**

(Dugué and Perez)

Particularly known for its ability to uncover hierarchical community structures, the Louvain heuristic algorithm initially assigns each node to its own cluster and proceeds to iterate two main phases. The first one involves the repetition of two steps: the Modularity variation by moving each node to the community of each of its neighbors is evaluated, subsequently each vertex is eventually reassigned to the cluster providing the highest Q gain. This process is stopped when no further improvement in Modularity is achievable, thus the second phase is enacted: the network is aggregated by treating each previously identified community as a single node and constructing a renovated graph. In the latter, edges between the new entities are weighted by the total traffic originally exchanged between vertices of the different communities. The two phases are iterated – applying the first one to the aggregated network – until an ultimate Modularity peak is reached.

Algorithm: Louvain (L)

Input: A graph $G = (V,E)$.

Output: A partition of the graph's nodes into communities.

1. Initialization: Assign each node to its own community.
2. Iterate the following steps until convergence (i.e., no further Modularity gain possible):
   2.1. For each node i apply Modularity optimization:
      2.1.1. Evaluate the Modularity gain by moving i to the community of each neighbor.
      2.1.2. Place i in the community that maximizes the Modularity gain.
   2.2. Aggregation: build a new network where each community is a single node, construct edges between these entities weighted by the total inter-community traffic values in the original network.
3. Output the final partition of nodes into communities.

1.2.4 – **Hierarchical**

Depicting tree-like structures (a.k.a. dendrograms), hierarchical clustering algorithms either follow an agglomerative strategy – of pairwise bottom-up cluster iterative merge starting from an own-labeled node situation - or a divisive strategy – initializing all the vertices into a single cluster which is subsequently split top-down. To accomplish this type of procedure, a measure of nodes' similarity or edges' importance is to be specified, which is used to determine where to enact a division or unification at each iteration. An intrinsic upside of hierarchical methods is their output's format: dendrograms are interpretable visual representations of the clustering procedure, precious to select the most significant k in retrospect. On the other hand, a definitely high complexity makes these strategies non-scalable to even moderately large datasets.

(GN) **– Girvan-Newman**

(Newman and Girvan, Finding and evaluating community structure in networks)
(Newman, Analysis of weighted networks)

The Girvan-Newman algorithm proceeds by division steps: subsequently removing the edge presenting the highest betweenness score – computable with different methods depending on the context of interest, the algorithm recognizes as clustering each set of components derived from an edge removal which further disconnects the graph. The procedure can be iterated until all the edges are removed in order to obtain a hierarchical clustering, representing the suggested nodes' division into 1,2,…,n communities. Otherwise, to obtain an optimal partition without calculating the whole decomposition, it is sufficient to select the one corresponding to a peak – thus preceding a decrement – in Modularity, since this metric reportedly increments to a maximum until the best number of clusters is considered. Strengths of this algorithm are the unnecessity to specify the number of communities as an input and its applicability on many graph types by simply selecting the appropriate edge's importance metric.

Algorithm: Girvan-Newman (GN)

Input: A graph $G = (V,E)$.

Output: A partition of the graph's nodes into communities.

1. Until no more edges remain or the Modularity starts decreasing:
   1.1. Calculate betweenness-based edge-removal priority scores for all edges in the network.
   1.2. Find the edge with the highest score and remove it from the network.
   1.3. If the graph has been disconnected in 1.2, compute the obtained clustering's Modularity.
2. Return as best clustering the one corresponding to the Modularity peak.

1.2.5 – **Traffic embedding**

Aiming to represent a network structure's characteristics through vectors in a low-dimensional space, the embedding of a graph's interconnection matrix (i.e., X or A) gathers information on its framework thanks to a plethora of strictly mathematical methods. Avoiding complex representation learning techniques, the structural clustering can then be entrusted to a basic algorithm (e.g., K-means).

(SF) **– Spectral Filtering**
(Gkantsidis, Mihail and Zegura)
(Naldi and Miani)

Spectral clustering methods typically exploit eigenvalues to perform a dimensionality reduction on a network's adjacency or traffic matrix, based on the correlation between graphs' framework and the relevant connectivity matrix's spectrum. In particular, the SF was originally applied to unweighted networks, operating on a matrix $M \coloneqq A \cdot A^T$; a later variation allowed using this method on weighted graphs by an adaptation of the symmetry transformation to $M \coloneqq X + X^T$. Finally, the ancillary use of a basic clustering algorithms permits the automated grouping of weights, engendering the necessity of a desired number of communities as input.

Algorithm: Spectral Filtering (SF)

Input: A weighted graph's traffic matrix X (or an unweighted graph's A), the desired clusters' number k.

Output: A partition of the graph's nodes into k communities.

1. Apply the symmetry transformation $M \coloneqq X + X^T$ (or $M \coloneqq A \cdot A^T$ for unweighted graphs).
2. Apply stochastic normalization, dividing each entry of M by the sum of all its entries.
3. Extract the largest eigenvalues (excepting the largest one) and their corresponding eigenvectors.
4. Sort the weights of each eigenvector, keeping the association with the nodes of the graph.
5. Identify the eigenvector whose sequence of weights shows the sharper jumps.
6. Return the communities obtained by clustering the above weights through a basic algorithm.

1.3 – **Number of clusters choice**

(Zhou, Zhu and Liu)

As noticeable from the above descriptions, some clustering algorithms – both when a graph's nodes are to be grouped and in general – require the desired number of output communities k to be specified as an input. Therefore, some specific strategies to wisely select this fundamental parameter must be enacted; the most renowned ones are grounded on the following k-dependent measures based on elements' features:

(S) – **Silhouette value** Comparing an entity's similarity to its own cluster with respect to other communities:

x_i: Data point of element i.

C_i^X: Most x-similar $C \not\equiv C_i'$. $\Rightarrow$ $a_i = \frac{1}{|C_i|}\sum_{j \in C_i} d(x_i, x_j)$ $\quad b_i = \frac{1}{|C_i^X|}\sum_{j \in C_i^X} d(x_i, x_j)$ $\Rightarrow$ $S_i = \begin{cases} 1 - \frac{a_i}{b_i}, if\ a_i < b_i \\ 0, if\ a_i = b_i \\ \frac{b_i}{a_i} - 1, if a_i > b_i \end{cases}$

$d(x_i, x_j)$: x_i to x_j distance.

(VRC) – **Calinski–Harabasz criterion** (a.k.a. Variance Ratio Criterion) Based on cluster-related variances:

$SS_B(C)$: between-cluster variance.

$SS_W(C)$: within-cluster variance. $\Rightarrow$ $VRC(C) = \frac{SS_B(C)}{SS_W(C)} \cdot \frac{n-k}{k-1}$

(GC) – **Gap criterion** Calculates the difference between the within-cluster dispersion and its expected value:

$D(C)$: C's nodes pairwise distances sum. $\Rightarrow$ $W(\boldsymbol{C}) = \sum_{C \in \boldsymbol{C}} \frac{D(C)}{|C|}$ $\Rightarrow$ $GC(\boldsymbol{C}) = \underset{\uparrow}{E}\left[\log\left(W(\boldsymbol{C})\right)\right] - \log\left(W(\boldsymbol{C})\right)$

(Expectation value determined by Monte Carlo sampling from a reference distribution)

(EM) – **Elbow Method**

Plotting the Distortion (i.e., Within-Cluster Sum of Squares), average Silhouette, or average VRC value against k, a decreasing trend is visualizable until – beyond a specific 'elbow' point – adding more clusters does not produce a significant drop in the relevant metric. The Elbow Method suggests that specific k as the optimal one, preceding a sharp slowdown in clustering return's variation at considering an additional cluster.

(GS) – **Gap Statistic**

Plotting GC against a set of established clusters' numbers, the Gap Statistic strategy grounds on the comparison at different k of the studied method's results on relevant data and on null reference ones. The number of communities producing the most significant 'gap' with respect to chance-based expectations is thus indicated as most sensible, probably corresponding to a meaningful and non-random data partition.

(SA) – **Silhouette Analysis**

Generating a test clustering for each communities' number and depicting the S value for each data point into k-specific charts, the Silhouette Analysis allows the evaluation of how similar the entities are to the other inter- and intra-cluster ones. As a result, the best number of clusters can be visually selected by also probing for all the communities their Silhouette scores' distribution and sizes' consistency.

1.4 – Representation learning algorithms

Aiming to automatically discover meaningful patterns and reducing the noise of raw network data, graph representation-learning strategies range from classical methods (e.g., PCA-alike) to complex non-linear approaches such as autoencoders, allowing unsupervised learning applications on complex networks.

1.4.1 – Random walks

Random-walk strategies leverage the process of traversing a graph starting from a source node and successively moving to a random neighbor, with walk's length and number of per-node walks being determined hyperparameters. Transforming the graph into a sequence of node visits to be fed into a NN-based model, these methods allow to learn networks' low-dimensional embeddings which capture local and global structures, reaching outstanding performances in social networks' classification tasks in particular. The random-walk strategies' strengths are indeed adaptability – especially to incomplete or sparse labeling – and scalability, being suitable for parallelization and incremental updates to deal with large and evolving networks.

(DW) **– DeepWalk**

(Perozzi, Al-Rfou and Skiena)

Considering the network's interconnectivity framework only, this purely traversing-based algorithm conceives walks analogous to sentences in Natural Language Processing. DeepWalk indeed extends advancements in unsupervised feature learning from words to graphs, constructing social representations of vertices – sequences of which are akin to a vocabulary – reflecting their neighborhood similarity and community membership. Skip-gram – a model grounded on Neural Network techniques – and Hierarchical Softmax – representing the prediction problem through a binary tree structure – are thus employed to map nodes to their latent representation vectors with reduced computational complexity.

(TADW) **– Text-Associated DeepWalk**

(Yang, Liu and Zhao)

Regarding the above presented algorithm as matrix factorization method, TADW extends it by integrating nodes' attributes into the representation learning process, aiming to enhance the outcome's quality by combining structural and textual information. In fact, DeepWalk factorizes a matrix where each entry represents the logarithm of the average probability that a vertex will reach another one within a fixed number of steps. On the other hand, TADW produces a matrix $M \coloneqq (A + A^2)/2$, balancing between accuracy and computational efficiency in random walk sampling. M is thus factorized into three matrices: W, H, and the text feature matrix T; then, the network representation W is concatenated to the attributes-based representation H^T, creating a unified vertices' embedding. This strategy is particularly powerful in dealing with noisy graphs and limited training-data scenarios, since the inclusion of text features helps to mitigate data sparsity issues.

1.4.2 – **Deep Learning**

Novel graph embedding strategies often extensively exploit Neural Networks' learning capacities and especially Graph Convolutional Networks (GCN) strengths in autonomous data interpretation.

(DCRN) **– Dual Correlation Reduction Network**
(Liu, Tu and Zhou) (Liu, Xia and Zhou) (Kipf and Welling)

Designed to overcome the problem of representation collapse in previous GCN-based clustering methods, this self-supervised algorithm employs a Siamese network to encode graph samples and introduces a dual information correlation reduction mechanism to avoid representations' redundancy. The latter is implemented at both sample and feature levels, ensuring different features' decorrelation and enhancing the discriminative power of nodes' representations. Furthermore, in order to mitigate GCNs' over-smoothing problems, a propagation regularization term is incorporated to capture long-distance information without deepening the network. All this first training step is enacted by AutoEncoder (AE): a NN-based strategy consisting in compressing the input data into a lower-dimensional latent representation and reconstructing them back.

Successively, Graph AutoEncoder (GAE) – a method extending AE's concepts to graph structure by using Graph Convolutional Networks – uses two graph distortion types (i.e., Gaussian-noise feature corruption and PPR-diffusion edge perturbation) to create different views of the graph which are encoded by the Siamese network. The overall objective function of DCRN combines the dual information correlation reduction loss with reconstruction and clustering loss, ensuring respectively that features-enriched structural information is preserved and that the learned representations align with target distributions.

2 – Implementation

In order to assess AI's capacities in examining networks in a business context; all the methods presented in the above section are hereafter applied in analyzing a delivery-logistic sub-graph granted by Poste Italiane.

2.0 – Preliminary analysis

The database's section in question contains information about the movements of all the shipped packages of a particular Business-service type sent from Lombardia and delivered to Puglia in a specific 2-months period.

Tabular format data (31914 rows)					
Events' identification			**Attributes of the relevant…**		
ID_TRACED_ENTITY	ID_CHANNEL	WHEN_HAPPENED	…entity	…channel	…event
E1	C1	DT1	tipo vettore_1 peso peso_eff	officename officename_clean competenza regione cod_provincia zipcode	WHAT_HAPPENED DESCRIPTION ID_STATUS MACROESITO FLAG_FINALE
E1	C2	DT2			
E1	C3	DT3			
E2	C4	DT4			
E2	C5	DT5			

Each entity is actually a parcel and information is given about its shipping-type, appointed delivery company, declared and effective weight. Each channel is instead an office through which parcels are worked while proceeding to their delivery point; thus, information is given about its name, the name of the hub of which it is eventually part, its type (e.g., PI center, post office, authorized pick-up point) and location (i.e., Italian Region, District and zip code). Moreover, information about the event – the transition of a package through an office at a determined date and time – is specified in each row; in particular, the events are described both in a coded and explicit way, and the shipping phase is reported. Below a count of unique values for the IDs and for some features which in the subsequent analysis are found interesting:

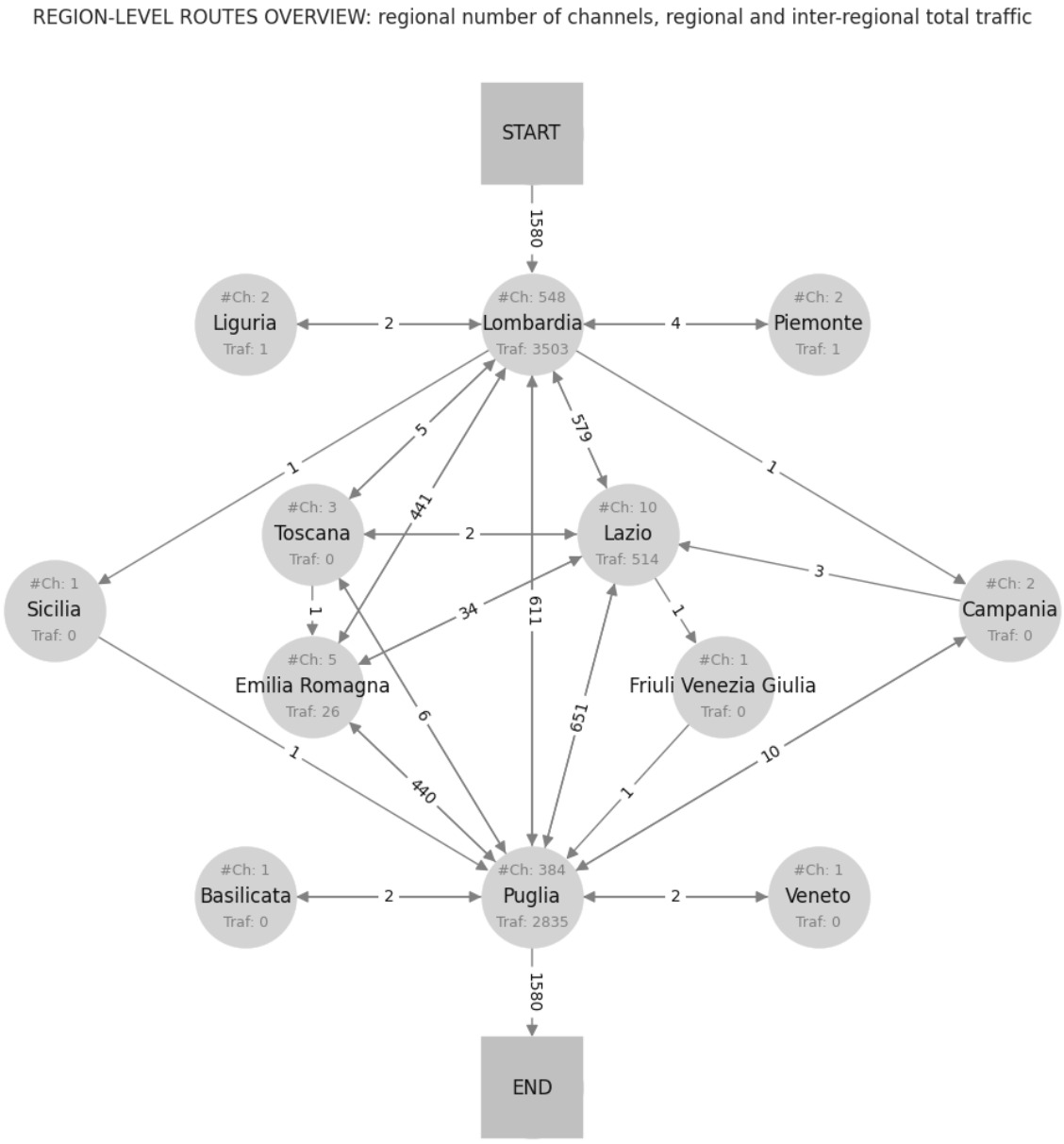
UNIQUE VALUES	ID_TRACED_ENTITY	ID_CHANNEL	tipo	officename	competenza	regione
	1580	960	1	935	12	12

The information is imported in DataFrame format, and a few verifications executed: the data were granted by Poste Italiane already checked and cleaned for internal usage.

2.1 – Graphs construction

The information is thus translated in network format: an "events" unweighted MultiDiGraph is firstly built, representing offices as nodes and package transitions – obtained grouping rows by entity and ordering the traversed channels based on the datetime value – as edges. Since many events are characterized by a change in shipping status without an actual movement of the package to a different location, self-loops are present in the so-produced graph and are removed to favor the structural cleanness needed for the purpose of analyzing the network from a delivery itinerary viewpoint. Afterwards, a more significant DiGraph is obtained, with the same nodes but with edges' weight representing the total number of entities which moved from one specific

vertex to another. The latter would be called $routesG$ (960 nodes, 1385 edges) and depicts the parcels' traffic between offices in a proper way for the subsequent analyses. A visualization is produced grouping the offices by region to have a preliminary cognition of the interconnection framework in question:



Noticing a process-like distribution and previewing the related difficulties for some of the intended algorithms to interpret a similarly 'linear' structure, two sub-graphs are built abstracting only nodes from the mainly active Regions (i.e., *Lombardia* and *Puglia*). Since no significant output-quality improvement is then noticed applying the clustering methods on these sub-graphs, $routesG$ remained the study's standard input.

2.2 – **Network clustering**

In this section, all the already discussed methods are applied on $routesG$ with the aim of probing its nodes' characteristics. Specifically, channels are analyzed and grouped in clusters through the different proposed algorithms firstly based on nodes' attributes only, then on the whole graph structure only, and finally

leveraging both the information sets. To do so, the previously introduced representation-learning strategies are exploited too, generating embeddings from the relevant data and applying K-means clustering on them. Being most of the algorithms non-deterministic, a seed value is used throughout the procedure to generate significant but constant outputs, since testing via multiple iterations without applying the latter de-randomization technique led to very slight results variations in terms of nodes partition and subsequent clustering's quality.

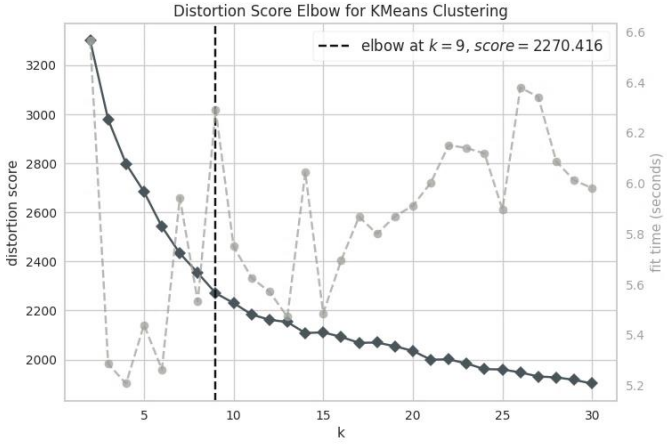
In order to assess the properties of the resulting communities sets, two functions are defined: one studying the input clustering's quality considering the relevant network structure - applying all the graph-based metrics already discussed (i.e., Cov, SCM, TS, Q, AC) – and one testing the exactness of the partitioning with respect to a given classification (e.g., another already generated clustering or a ground-truth set of communities) – computing all the above presented label-based indexes (i.e., Acc, J, FM, ARI, NMI, F1). A function is specially prepared to calculate the traffic-aware performance TS, obviating its absence in standard python libraries.

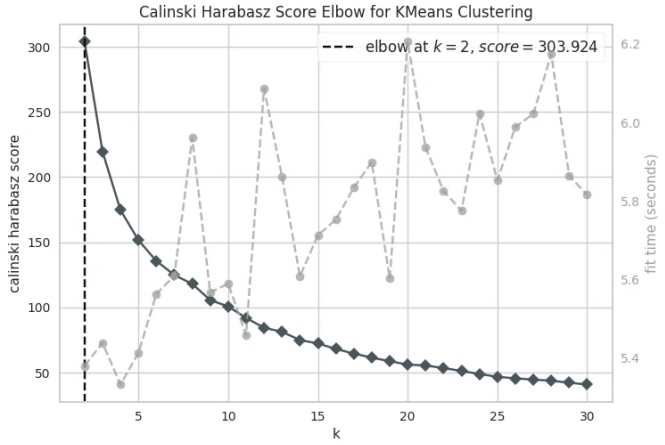
2.2.0 – Number of clusters choice

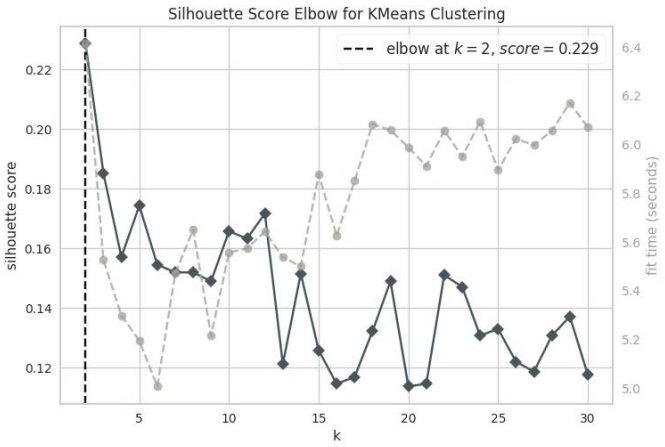
As some of the intended algorithms need the desired number of communities *k* to be chosen, the already discussed ad-hoc strategies are applied, conceiving features' K-means clustering as a baseline method.

(EM) – Elbow Method

The *k*-dependent trend is studied for three clustering quality scores: Distortion, Calinski-Harabasz and Silhouette; representative results for communities' number up to 30 are depicted here.

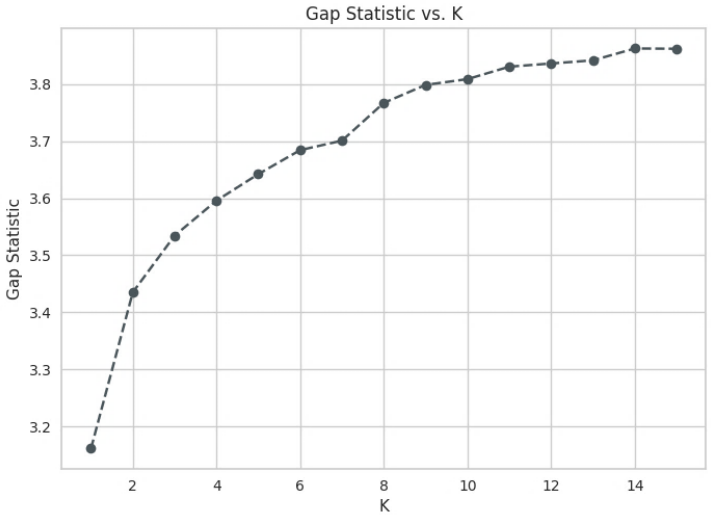






Suggesting that a significant *k* based on nodes' attributes would be to choose around 9 or around 2, the Elbow Method's outcomes are not surprising at all. In fact, channels' features are only six, all categorical, and four of them (i.e., *officename*, officename_clean, *cod_provincia*, *zipcode*) have an excessively high number of possible values to be considered as particularly relevant by a clustering algorithm. Therefore, only two attributes remain possibly leading in determining each office's allocation to a community: *competenza* – with only 9 over the 12 possible categories being consistently

represented – and *regione* – depicting a distribution explained at 97% just by *Lombardia* and *Puglia* comparably-sized groups of channels.

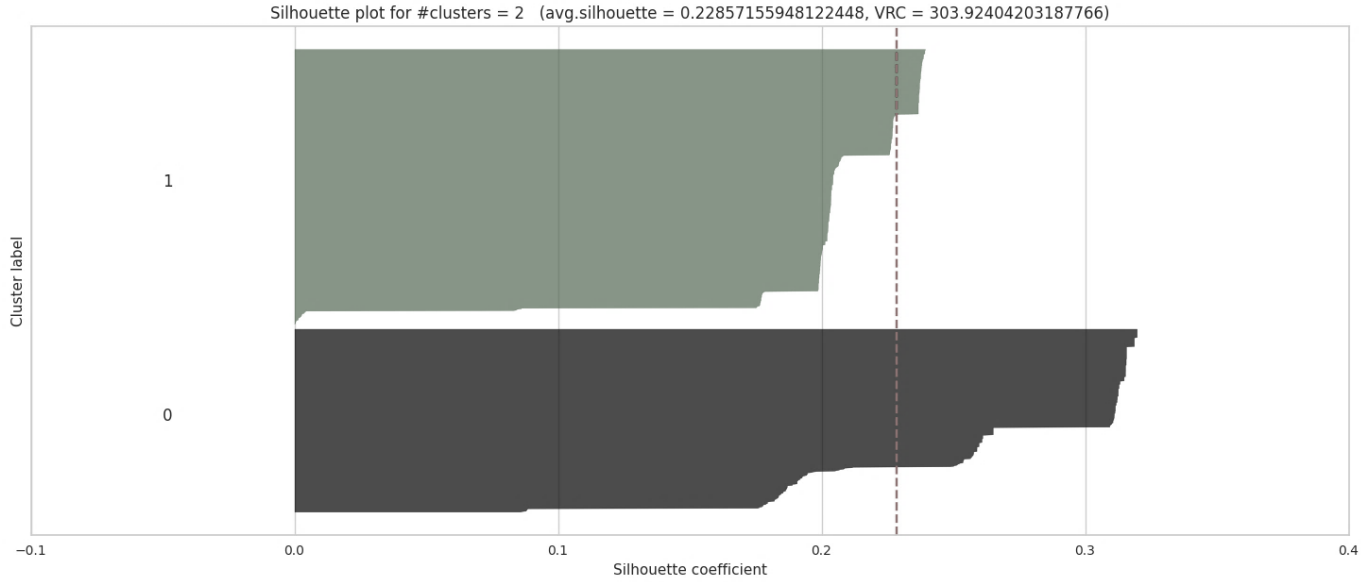


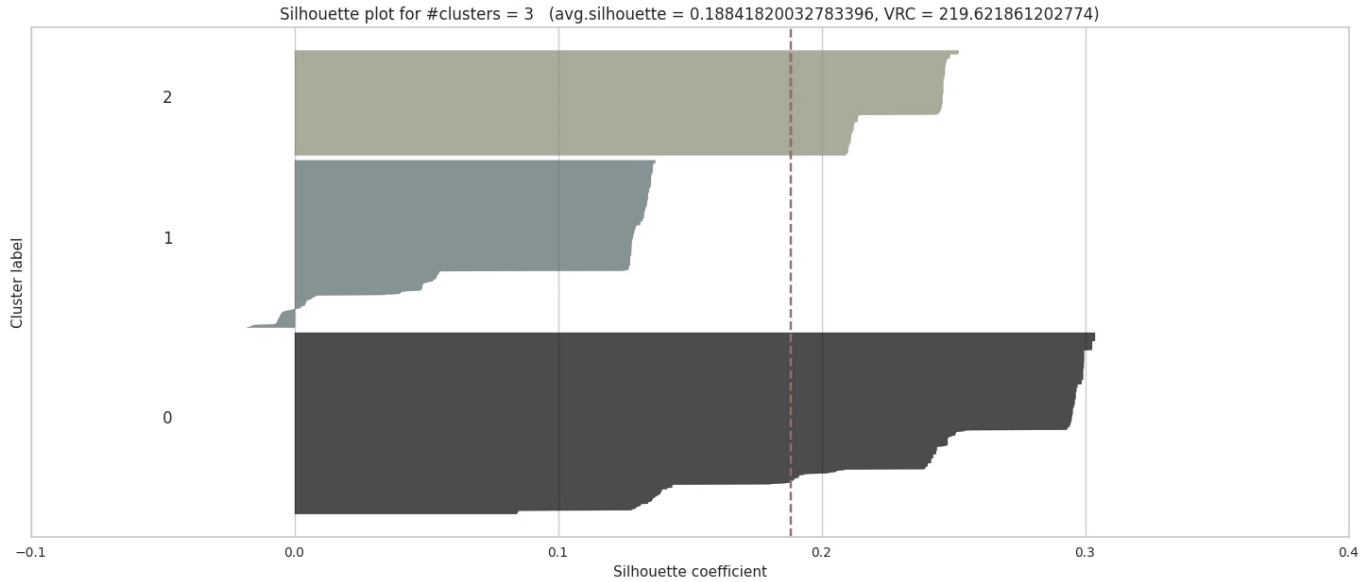
(GS) **– Gap Statistic**

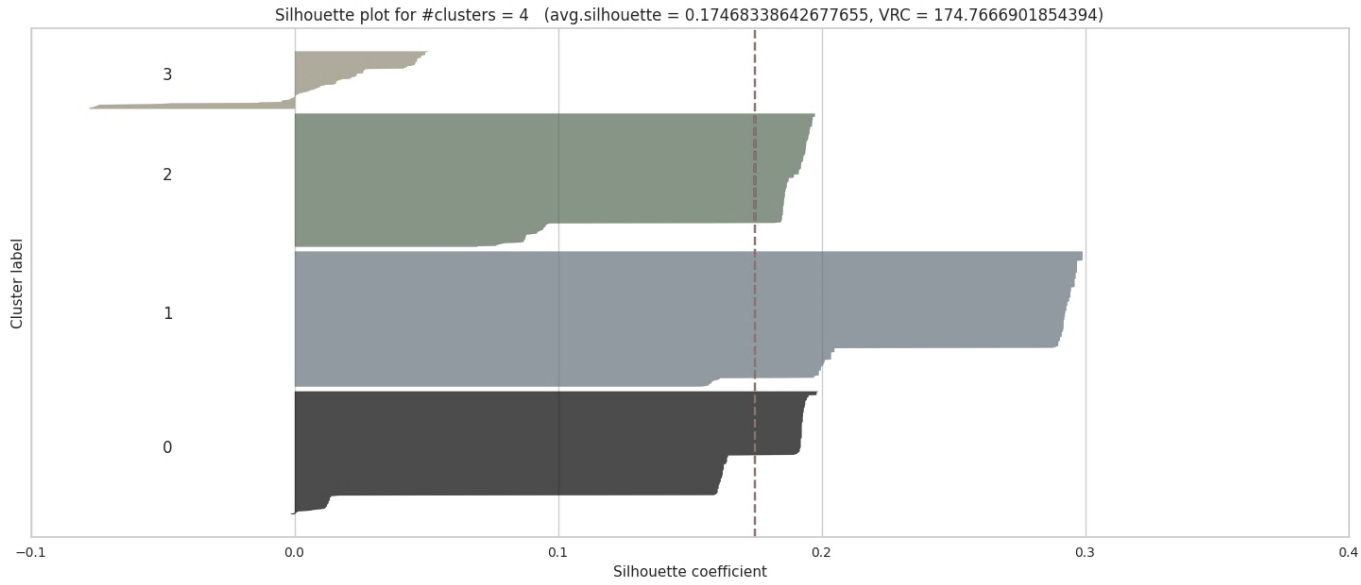
Considering the low score increments at *k* higher than 3, the Gap Statistic's results seem to confirm that priority should be given to values around 2 rather than 9 when choosing the clusters' number.

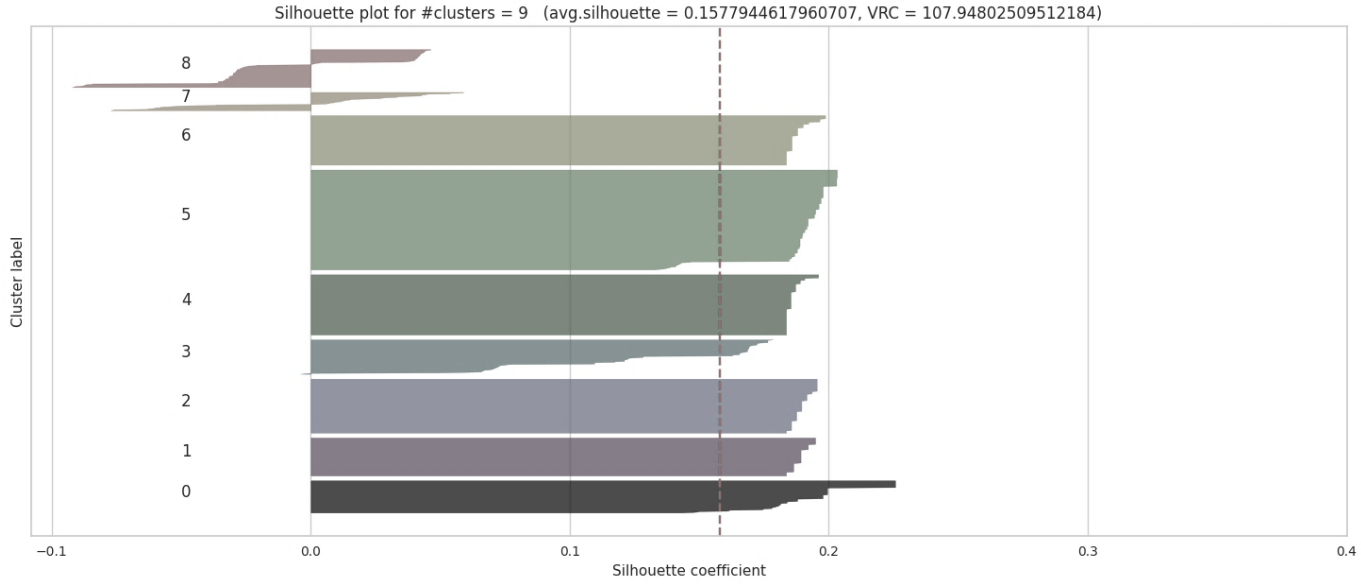
(SA) **– Silhouette Analysis**

Generating visualizations of silhouette multiple distributions for each different *k* value up to 15, no doubt is raisable about the consistency of 2 as number of clusters to pass as relevant algorithms' input. Some significant visualizations are reported to better express the less coherent communities' size distribution and the under-average scores typical of clusterings in more than 2 – and particularly in around 9 – communities.









2.2.1 – **Attributes-based**

Proceeding to actual node clustering, at first channels are indeed grouped based on their features only; the most reasonable method to apply is thus exactly K-means.

(K) **– K-means**

Applying this baseline method on nodes' encoded attributes with a desired communities' number of 2, a Region based clustering is unsurprisingly produced. In particular, the 384 *Puglia* offices are grouped together and apart from the 576 other nodes – all the 548 *Lombardia* ones included, producing a labeling which could practically be used as comparison in evaluating other clusterings' Region-based splitting capacity.

The clustering quality metrics are computed and saved in a DataFrame, which will be henceforth continuously updated to constitute an evaluation and comparison basis for all the applied methods.

2.2.2 – **Structure-based**

Most of the theoretically examined algorithms are specifically intended for clustering of networks grounded on its edges' framework. In particular, some of them rely on this information type only; their application on the graph structure in analysis is discussed in this section.

(ALP) **– Asynchronous Label Propagation**

Due to the sparse input network mainly depicting consolidated routes, Asynchronous Label Propagation's outcome asserts the presence of a very high number of communities: k is auto-assessed to be 297 in the here examined execution instance, but only 2 of the resulting clusters comprise more than 10 channels. These groups indeed represent the core slightly-more interconnected components of *Lombardia* and *Puglia* sub-graphs, approximately including 500 and 60 channels respectively.

(L) **– Louvain**

This Q-maximization method does not require a determined k too, but a parameter is specifiable to favor either larger or smaller clusters: a *resolution* higher than the default one (i.e., 1) would promote narrow communities' creation, while a lower value would foster the partitioning in wider clusters. For this reason, after a default execution at standard parameters – leading to about 20 communities, *resolution* is set to 0.1 in order to achieve a k more comparable to the other strategies' ones – obtaining 10 clusters in the reported LR-L case.

(GN) **– Girvan-Newman**

The Girvan-Newman algorithm is firstly executed with a fixed clusters' number of 2, returning the community division derived from the graph's first disconnection by iterative betweenness-based edge removals. Afterwards, the process is allowed to run and realize the typical hierarchical analysis until the Modularity peak is reached, outputting a partition in 13 clusters (F-GN). In both cases, the Q calculation is based on edges' traffic and the betweenness computations on the inverse of the same value, intended by the latter function as specific 'distance' of the nodes connected by each edge. In fact, routes' sections presenting a higher number of package transfers are assumed to be both the most crucial – attempting to keep them as in-cluster edges, and the logistically-elected most efficient paths – to be conceived as higher betweenness connections.

(SF) **– Spectral Filtering**

The dimensionality-reduction strategy is applied on the original connectivity data; then, K-means is executed on the resulting embeddings with a pre-determined communities' number of 2.

(DW) **– DeepWalk**
[reference repository]

As a final clustering strategy grounded on structure only, this representation learning method is applied on the $routesG$ framework information; then, K-means is run on the resulting embeddings with a fixed k of 2.

For all these methods, the clustering quality metrics are computed and saved in the ad-hoc DataFrame.

2.2.3 – **Deep graph clustering**

The last two hereafter discussed algorithms attempt a deeper understanding of the input network, considering both its interconnectivity structure and nodes' features at once. In the below applied algorithms' cases, the embeddings' quality largely depends on a wise choice of input parameters. Therefore, these variables are selected by tuning over multiple test-executions, with the aim of progressively maximizing the resulting clustering's scores; the best obtained partitions are thus reported only.

(TADW) – **Text-Associated DeepWalk**
[reference repository]

Through this method, random-walk-based representation learning is applied on the analyzed network as a whole; then, K-means is run on the resulting representation with a fixed k of 2.

(DCRN) – **Dual Correlation Reduction Network**
[reference repository]

Finally, this highly NN-based algorithm is executed to generate another deep graph's representation; then, K-means is executed on the results with a pre-determined communities' number of 2.

For both these methods, clustering quality metrics are computed and saved in the ad-hoc DataFrame.

2.3 – **Outcomes analysis**

In this section, the obtained partitions' properties are discussed, assessing each clustering's quality, the related algorithms' performance on $routesG$, and on similarly structured graphs consequently.

2.3.1 – **Clusterings' quality**

To examine the various strategies' outputs at first, the structure-based scores are to be considered:

Outputs evaluation		*k*	Cov	SCM	TS	Q	AC
K-means	**(K)**	2	**0.82**	0.48	0.48	**0.31**	**0.01**
Async. Label Propagation	**(ALP)**	297	**0.83**	0.62	0.62	0.07	0.00
Louvain	**(L)**	23	0.65	0.82	0.82	**0.45**	0.08
Low-Resolution Louvain	**(LR-L)**	10	**0.78**	**0.72**	**0.72**	0.46	0.09
Girvan-Newman	**(GN)**	2	0.98	0.14	0.14	0.00	0.50
Free Girvan-Newman	**(F-GN)**	13	**0.74**	**0.77**	**0.77**	0.02	0.84
Spectral Filtering	**(SF)**	2	0.51	0.48	0.48	0.01	**0.07**
Deepwalk	**(DW)**	2	0.52	0.50	0.50	0.01	**0.07**
Text-Associated DeepWalk	**(TADW)**	2	**0.90**	0.50	0.50	0.00	0.47
Dual Correlation Reduction Network	**(DCRN)**	2	0.49	0.12	0.12	-0.12	0.17

The Louvain method seems the best performing one from a network's framework perspective: it obtained significant results in favoring intra-cluster connectivity over inter-cluster one, coherently reaching good scores over all the above metrics and excelling in Coverage and Modularity maximization – at standard and low *resolution* parameter setting respectively. K-means is the only other algorithm boasting a satisfactory

performance, with suitable results in all the metric types – despite a considerable drop at Cov's normalization; on these data, an evaluation of nodes' attributes would indeed lead to a sensible partition also form the network perspective, since the vertices are already strongly polarized towards shipping's start and end Regions.

On the other hand, with its hyper-divisive outcome, Label Propagation obviously achieve a perfectly low Average Conductance and significantly high Coverage-related scores, while failing to reach a satisfactory Modularity – being Q normalized by random-clustering results' expectation; a similarly contradictory scores distribution is noticeable for Spectral Filtering. The Girvan-Newman strategy, instead, displays unfulfilling results in both Q and AC, reaching a very low Modularity peak during iterative execution and an inconsistently high non-normalized Coverage after the first disconnection step; this prefigures the inability of any method based on edges' centrality in operating sensible partitioning on this specific network.

Finally, the clusterings grounded on representation learning algorithms fail to fulfill Modularity's requirements, with considerable GCN's misinterpretation of the graph in analysis. Despite their unweighted perception of edges in the executed instances, DeepWalk-based methods seem slightly better performing; in particular, adding nodes' features to embedding procedures lead to substantially worse network interpretations, as identifiable through the substantial AC drop and non-normalized Coverage burst between DW and TADW.

2.3.2 – **Algorithms comparison**

A second evaluation step would involve the direct comparison among the presented methods, this could be achieved by computing the label-based scores between the clustering-obtained labels; hereafter the results:

Acc	K	ALP	L	LR-L	GN	F-GN	SF	DW	TADW	DCRN
K	1.00	**0.60**	0.03	0.17	**0.59**	0.22	0.51	0.51	0.49	0.58
ALP	0.60	1.00	0.00	0.17	0.57	0.23	0.36	0.31	0.31	0.57
L	0.03	0.00	1.00	0.00	0.03	0.04	0.03	0.04	0.03	0.04
LR-L	0.17	0.17	0.00	1.00	0.22	0.21	0.24	0.26	0.27	0.21
GN	0.59	0.57	0.03	0.22	1.00	0.37	**0.59**	0.52	0.55	**0.87**
F-GN	0.22	0.23	0.04	0.21	0.37	1.00	0.23	0.22	0.30	0.35
SF	0.51	0.36	0.03	0.24	0.59	0.23	1.00	0.51	0.55	**0.59**
DW	0.51	0.31	0.04	0.26	0.52	0.22	0.51	1.00	0.58	0.51
TADW	0.49	0.31	0.03	0.27	0.55	0.30	0.55	0.58	1.00	0.53
DCRN	0.58	0.57	0.04	0.21	0.87	0.35	0.59	0.51	0.53	1.00

J	K	ALP	L	LR-L	GN	F-GN	SF	DW	TADW	DCRN
K	1.00	**0.42**	0.02	0.09	**0.42**	0.12	0.34	0.35	0.32	0.41
ALP	0.42	1.00	0.00	0.09	0.40	0.13	0.22	0.18	0.18	0.40
L	0.02	0.00	1.00	0.00	0.02	0.02	0.02	0.02	0.02	0.02
LR-L	0.09	0.09	0.00	1.00	0.13	0.12	0.13	0.15	0.15	0.11
GN	0.42	0.40	0.02	0.13	1.00	0.23	**0.42**	0.35	0.38	**0.77**
F-GN	0.12	0.13	0.02	0.12	0.23	1.00	0.13	0.12	0.18	0.22
SF	0.34	0.22	0.02	0.13	0.42	0.13	1.00	0.34	0.38	**0.42**
DW	0.35	0.18	0.02	0.15	0.35	0.12	0.34	1.00	0.41	0.34
TADW	0.32	0.18	0.02	0.15	0.38	0.18	0.38	0.41	1.00	0.36
DCRN	0.41	0.40	0.02	0.11	0.77	0.22	0.42	0.34	0.36	1.00

FM	K	ALP	L	LR-L	GN	F-GN	SF	DW	TADW	DCRN
K	1.00	**0.81**	0.56	0.67	0.67	0.34	0.52	0.51	0.51	0.67
ALP	0.81	1.00	0.40	0.40	0.57	0.29	0.44	0.44	0.44	0.57
L	0.56	0.40	1.00	**0.76**	0.39	0.20	0.31	0.30	0.30	0.40
LR-L	0.67	0.40	0.76	1.00	0.49	0.26	0.38	0.38	0.37	0.50
GN	0.67	0.57	0.39	0.49	1.00	0.51	0.67	0.66	0.66	**0.87**
F-GN	0.34	0.29	0.20	0.26	0.51	1.00	0.34	0.34	0.51	0.44
SF	0.52	0.44	0.31	0.38	0.67	0.34	1.00	0.51	0.51	0.68
DW	0.51	0.44	0.30	0.38	0.66	0.34	0.51	1.00	0.51	0.66
TADW	0.51	0.44	0.30	0.37	0.66	0.51	0.51	0.51	1.00	0.66
DCRN	0.67	0.57	0.40	0.50	0.87	0.44	0.68	0.66	0.66	1.00

ARI	K	ALP	L	LR-L	GN	F-GN	SF	DW	TADW	DCRN
K	1.00	**0.64**	**0.30**	**0.43**	0.00	0.00	0.00	0.00	0.00	0.00
ALP	0.64	1.00	0.16	0.10	0.00	-0.01	-0.01	0.00	0.00	-0.01
L	0.30	0.16	1.00	**0.67**	0.00	-0.01	0.00	0.00	0.00	0.00
LR-L	0.43	0.10	0.67	1.00	0.00	0.00	0.00	0.00	0.00	0.00
GN	0.00	0.00	0.00	0.00	1.00	0.09	0.00	0.00	0.01	-0.02
F-GN	0.00	-0.01	-0.01	0.00	0.09	1.00	0.00	0.01	0.23	-0.01
SF	0.00	-0.01	0.00	0.00	0.00	0.00	1.00	0.00	0.01	0.00
DW	0.00	0.00	0.00	0.00	0.00	0.01	0.00	1.00	0.03	0.00
TADW	0.00	0.00	0.00	0.00	0.01	0.23	0.01	0.03	1.00	0.00
DCRN	0.00	-0.01	0.00	0.00	-0.02	-0.01	0.00	0.00	0.00	1.00

NMI	K	ALP	L	LR-L	GN	F-GN	SF	DW	TADW	DCRN
K	1.00	**0.35**	**0.40**	**0.47**	0.00	0.01	0.00	0.00	0.00	0.00
ALP	0.35	1.00	**0.38**	0.27	0.06	0.27	0.13	0.13	0.13	0.05
L	0.40	0.38	1.00	**0.73**	0.01	0.06	0.01	0.01	0.01	0.01
LR-L	0.47	0.27	0.73	1.00	0.00	0.03	0.01	0.00	0.01	0.01
GN	0.00	0.06	0.01	0.00	1.00	0.24	0.00	0.01	0.05	0.00
F-GN	0.01	0.27	0.06	0.03	0.24	1.00	0.02	0.02	0.25	0.02
SF	0.00	0.13	0.01	0.01	0.00	0.02	1.00	0.00	0.01	0.00
DW	0.00	0.13	0.01	0.00	0.01	0.02	0.00	1.00	0.02	0.00
TADW	0.00	0.13	0.01	0.01	0.05	0.25	0.01	0.02	1.00	0.02
DCRN	0.00	0.05	0.01	0.01	0.00	0.02	0.00	0.00	0.02	1.00

F1	K	ALP	L	LR-L	GN	F-GN	SF	DW	TADW	DCRN
K	1.00	0.64	**0.71**	**0.80**	0.65	0.42	0.54	0.51	0.51	0.65
ALP	0.01	1.00	0.04	0.02	0.01	0.03	0.01	0.00	0.00	0.00
L	0.12	0.15	1.00	0.45	0.07	0.07	0.07	0.07	0.07	0.08
LR-L	0.24	0.26	**0.91**	1.00	0.15	0.11	0.13	0.10	0.12	0.16
GN	0.43	0.44	0.30	0.35	1.00	**0.78**	0.43	0.42	0.45	0.54
F-GN	0.12	0.12	0.12	0.12	0.20	1.00	0.11	0.13	0.19	0.12
SF	0.54	0.54	0.36	0.43	0.65	0.42	1.00	0.50	0.54	0.65
DW	0.55	0.55	0.37	0.43	0.65	0.43	0.55	1.00	0.59	0.66
TADW	0.55	0.55	0.38	0.43	0.65	0.67	0.54	0.59	1.00	0.66
DCRN	0.43	0.43	0.29	0.33	0.53	0.38	0.43	0.40	0.43	1.00

Differing only by the *resolution* parameter setting, the two Louvain executions' outputs lead to very high values in every comparison metric apart from Acc and J; this might be due to an equal approach but different detected communities' number. Besides directly explaining the significant difference between the two F1

scores – weighted by clusters' size, the more-than-doubled k evidently makes Accuracy's mapping arduous and leads a consistent part of the Jaccard-compared node couples to be split. A totally analogous situation is found when comparing K-means' and Low-Resolution Louvain's results. Similarly to what already noted for ALP vs K – here reaching constantly high values obviously except for the F1 weighted on ALP clusters' sizes; LR-L produces an indeed highly Region-correlated partition, abstracting three *Lombardia* highly-connected components and quite sensibly dividing *Puglia* nodes from the others, as noticeable by the following crosstab.

#Nodes	LR-L									
K	0	1	2	3	4	5	6	7	8	9
0	**161**	**295**	0	**67**	16	2	13	17	2	3
1	38	0	2	1	**343**	0	0	0	0	0

On the other hand, despite sharing the same structural clustering strategy, the two DeepWalk-based algorithms (i.e., DW and TADW) produce well-balanced but totally different halves of the graphs' nodes (distribution shown on the right), inducing medium scores in direct label-comparison measures but being overturned by randomicity-normalized ones. Many other methods comparisons fail to persevere from an NMI and ARI viewpoint even if well-scoring on the other metrics, but these cases are definitely doomed by the totally incoherent community sizes (illustrative crosstabs reported) proposed by either GN, DCRN or both – with the latter case even record-scoring at Acc, J, and FM.

#Nodes	TADW	
DW	0	1
0	**279**	**198**
1	**201**	**282**

#Nodes	GN	
DCRN	0	1
0	**830**	67
1	60	3

#Nodes	DCRN	
SF	0	1
0	**543**	35
1	**354**	28

3 – Conclusions

Considering the above practical analysis and the presented theoretical groundings, Louvain algorithm's strength effectively emerges in efficiently uncovering community structures at multiple levels: LR-L best understand $routesG$ framework, probably by parallelly evaluating both geographical distribution and offices' pyramidal order (i.e., from the most central hubs to the capillary delivery channels). Many clustering algorithms instead struggle when applied to sparse and sequential-steps networks. In fact, specific strategies and software are typically used both in the business and industrial sector when dealing with process-representing structures, focusing on the assessment of edges, routes sections, and only indirectly on nodes.

Considering the poor outcome of DCRN and the satisfactory results given by some simpler clustering algorithms (e.g., K-means and Louvain), it is inferable that deep Artificial Intelligence techniques are sometimes still less competent with respect to more human-guided algorithms in probing networks with peculiar frameworks and implied meanings. Surely bolstered by a restricted train dataset too, the apparent failure of NN-based representation learning methods seems to supply an additional instance in support of prominent theses highlighting these agents' unstable clustering capacities, not in step with their level of rigor in executing other graph-analysis procedures. *(Tsitsulin, Palowitch and Perozzi)*

Bibliography

- Amigó, Enrique, et al. "A comparison of extrinsic clustering evaluation metrics based on formal constraints." *Information retrieval* 12 (2009): 461–486.
- Dugué, Nicolas and Anthony Perez. "Directed Louvain: maximizing Modularity in directed networks." Ph.D. dissertation. 2015.
- Gkantsidis, Christos, Milena Mihail and Ellen Zegura. "Spectral analysis of Internet topologies." *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*. IEEE, 2003. 364–374.
- Khanfor, Abdullah, et al. "Graph neural networks-based clustering for social internet of things." *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2020. 1056–1059.
- Kipf, Thomas N. and Max Welling. "Variational graph auto-encoders." *arXiv preprint arXiv:1611.07308* (2016).
- Laura, Luigi, Maurizio Naldi and Giuseppe F. Italiano. "Traffic-based network clustering." *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*. 2010. 321–325.
- Liu, Xinwang, et al. "Late fusion incomplete multi-view clustering." *IEEE transactions on pattern analysis and machine intelligence* 41 (2018): 2410–2423.
- Liu, Yue, et al. "A Survey of Deep Graph Clustering: Taxonomy, Challenge, Application, and Open Resource." *arXiv preprint arXiv:2211.12875* (2022).
- —. "Deep graph clustering via dual correlation reduction." *Proceedings of the AAAI conference on artificial intelligence*. 2022. 7603–7611.
- Manning, Christopher D. *Introduction to information retrieval*. Syngress Publishing,, 2008.
- Mavromatis, Costas and George Karypis. "Graph infoclust: Leveraging cluster-level node information for unsupervised graph representation learning." *arXiv preprint arXiv:2009.06946* (2020).
- Naldi, M. and A. Miani. "Traffic matrix-based spectral decomposition of networks." *Proc. of FITraMEn* 8 (2008).
- Newman, Mark E. J. "Analysis of weighted networks." *Physical review E* 70 (2004): 056131.
- Newman, Mark E. J. and Michelle Girvan. "Finding and evaluating community structure in networks." *Physical review E* 69 (2004): 026113.
- Perozzi, Bryan, Rami Al-Rfou and Steven Skiena. "Deepwalk: Online learning of social representations." *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014. 701–710.
- Plummer, Michael D. and László Lovász. *Matching theory*. Elsevier, 1986.
- Raghavan, Usha Nandini, Réka Albert and Soundar Kumara. "Near linear time algorithm to detect community structures in large-scale networks." *Physical review E* 76 (2007): 036106.
- Salcedo-Sanz, Sancho, et al. "An evolutionary algorithm for network clustering through traffic matrices." *2011 7th International Wireless Communications and Mobile Computing Conference*. IEEE, 2011. 1580–1584.
- Tsitsulin, Anton, et al. "Graph clustering with graph neural networks." *Journal of Machine Learning Research* 24 (2023): 1–21.
- Yang, Cheng, et al. "Network representation learning with rich text information." *IJCAI*. 2015. 2111–2117.
- Yang, Jaewon and Jure Leskovec. "Defining and evaluating network communities based on ground-truth." *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*. 2012. 1–8.
- Zhou, Sihang, et al. "Subspace segmentation-based robust multiple kernel clustering." *Information Fusion* 53 (2020): 145–154.