Multiplex Aggregation and Community Detection Using Quasi-Arithmetic Means

Etienne Cuvelier[1] and Marie-Aude Aufaure[2]

[1] ICHEC - Brussels Management School, Quaresmi Laboratory
etienne.cuvelier@ichec.be
https://quaresmi.hypotheses.org/
[2] Datarvest, Versailles, marie-aude.aufaure@datarvest.com

Abstract. Muliplex networks permits to model many interesting situations happening in real life. Among the data analysis tools for multiplex, similarly to classical graphs, the community detection process is a central one. Several approaches have been proposed recently, one og them if the flattening or aggregation of the multiplex into a classical monoplex graph. Even if this approach can be seen as naive, it have as main advantage of offering the opportunity to use the whole set of detection algorithms developed for classical networks. In this paper we propose to perform the flattening process using quasi-arithmetic means on adjacency values across the levels of the multiplex. Using a spectral clustering on the resulting graph we applied this method on a new real dataset on the Human Resource sector. The main idea is to link individuals and job offers with competences and thus eventually make recommendations. The objective is to cluster groups of individuals and offers that share the same skills, as well as the relations linking them.

Keywords: Multiplex · Community Detection · Aggregation.

1 Introduction

Since the rising of online social networks, the power of graphs in data analysis is not to demonstrate any more. Since the beginnings of sociology they were used to study relationships in society [3, 32], but they were also used in biology [4, 27] transport networks studies [9, 21] and, of course, social networks [13].

But if graphs are able to model one type of link at the time, the need to model several relationships at once has appeared and lead to the development of multilayer networks and they were also used in biology [6, 12] transport networks studies [8, 20] and, of course, social networks [30]. In both case community detection is an intensively used technique [11, 16, 17, 22, 28].

Many community detection algorithms were developed for multilayer networks and some of them perform a layers aggregation before clustering, and that is what we propose in this paper, a new aggregation technique in order to use classical detection community algorithms.

The rest of this document is organized as follows: section 1.1 is a state of the art in the field, then section 1.2 recalls some graph definitions. Section 2.1

describes the objectives of the algorithm that will be part of Batch 3. Finally, section 4 concludes and opens some perspectives.

1.1 Related work

Simple graphs do not contain any attributes, neither in the nodes nor in the arcs, except possibly a weight associated with the arcs to indicate the strength of the connection. Social networks are a good example of a simple graph. Many algorithms rely on the structure of the graph. Types of clustering algorithms include partitioning algorithms, hierarchical algorithms and spectral algorithms. Spectral algorithms are specially designed for graphs and are based on the notion of connected components [26]. These algorithms work with a Laplacian matrix based on the adjacency matrix. The algorithm proposes to project the original data in a space defined by the k largest eigenvectors of a normalized adjacency matrix, and to apply a standard algorithm as k-means on these new coordinates. One disadvantage of this method is its complexity, since the k eigenvectors of the Laplacian matrix must be computed. If the graph is large, the exact calculation is in $O(n^3)$, which penalizes this type of graph. In the following, we will describe the related work on complex graphs [1]. These complex graphs can be of several types. Edge-attributed graphs are used to represent rich information about the interactions between nodes. They are also called multi-layer, multiplex, multidimensional graphs. Terms related to multiplex graphs are numerous in the scientific literature [22]. A general definition of multilevel graphs is given in [18]. Node-attributed graphs contain vectors associated with the nodes, representing the information contained in the nodes such as name, age, address, etc. As in the case of edge-attributed graphs, the scientific literature abounds in terminologies and associated models when considering the attributes of nodes, without any real consensus.

Arc-based clustering approaches - the multiplex case *Multiplex or multilayer graphs* can have multiple types on both arcs and nodes and allow to increase the understanding of complex systems and to discover phenomena that could not be explained on a simple graph (see [18] for a state of the art). In this type of graph, a node can belong to any layer and there are connections between nodes of different layers. In multiplex graphs, the sets of nodes are the same in the different layers. The analysis of these complex graphs requires to generalize the measures and algorithms designed for simple graphs and to be able to answer more specific questions such as whether one layer influences the evolution of another, which layers are the most relevant to measure the similarity between two nodes, etc. The approaches proposed in the literature are of two types: (1) the transformation into a single-layer graph problem such as level aggregation, hypergraph transformation approaches or ensemble clustering methods, or (2) the generalization of existing monoplex algorithms to multiplex graphs.

Aggregation approaches involve transforming a multiplex network into a weighted graph. Several weighting calculation methods can be applied, such as binary weights (two nodes are linked in the resulting graph if they are linked in at least one layer of the multiplex graph), weights based on frequency or similarity, or a linear combination taking into account the contributions of each layer for the weighting.

Ensemble type clustering approaches consist of first applying a community detection algorithm on each of the layers and then applying a clustering algorithm to combine all the partitions. There are several set-type clustering algorithms. Among these approaches, a widely used method is to construct a *consensus graph*. Two nodes are linked in this graph if there is at least one partition in which these two nodes are in the same cluster. The quality of the obtained partition is evaluated using the modularity criterion.

The last category consists in extending the monoplex approaches to the multiplex case. Among the proposed algorithms, one of the ideas is based on a ranking of the multiplex layers [15]. A community detection algorithm is applied on the first layer. Then, for each following level, a community detection algorithm is applied by optimizing two criteria: the modularity of the obtained partition and the maximization of the similarity with the partition found at the lower level. The main problem of this approach is to find a partition ranking function. An alternative approach [14] is based on the seed-centric concept. The idea is to find nodes (the seeds) that identify communities. The interest of this type of approach is that they are based on calculations located around each node identified as a seed, which allows their use on very large graphs. The detection of communities is then carried out from each local community.

Finally, an interesting hypothesis reported in the literature is that clusters could emerge with a specific combination of graphs and disappear when more graphs are added or removed [1].

Node-based approaches The objective is to detect groups of nodes with the same characteristics by considering both their attributes and their position in the graph. Most approaches are based on partitioning and homophilia: nodes belong to a single group and a group has homogeneous values for attributes. Some approaches generate overlapping classes, considering different combinations of attributes. Finally, there are also aggregation approaches based on the neighborhood of nodes with the same characteristics.

The first category of approaches is based on a transformation of the graph into a weighted graph in which the weights represent the similarity of the attributes. It is then possible to apply any clustering algorithm adapted to the weighted graphs. The resulting graph no longer contains attributes in the nodes because they are transformed into weights on the edges: the higher the weight, the more nodes are connected. These methods favour the creation of clusters in which the nodes are similar and connected. Discrete and continuous attributes can be taken into account as in the work of [29]. Once weights are calculated and normalized, arc-connected nodes with a weight greater than a threshold are grouped together in the same cluster.

In the previous case, the information contained in the nodes is transferred to the arcs. The opposite approach is to integrate the similarity between nodes in the node description. Classical distance-based clustering methods can then be applied. The main feature of this type of approach is that nodes that are structurally distant but similar in attributes can be clustered together, and therefore clusters can contain unconnected portions of the graph.

Some approaches focus on clustering networks of documents, each document being seen as a node characterized by a complex attribute defined by the words contained in the documents. Latent Dirichlet Allocation (LDA) methods can then be used extensively to identify groups and not just topics.

It should be noted that not all attributes are necessarily relevant and may generate clusters that are not very informative due to their high dimensionality. *Subspace clustering* approaches address this problem with the objective of selecting the most discriminating attributes to produce well separated clusters. They are designed to select the best subset of dimensions and consist of projecting data in different dimensions and identifying locally relevant clusters in certain subspaces. The disadvantage of these methods is that the search for relevant projections is costly and the result produces overlapping clusters (each relevant for a sub-area) and therefore redundancy must be controlled for.

Finally, the K-SNAP algorithm [31] is an aggregation approach based on the description of nodes and arcs, which allows the user to intervene in the aggregation scenario. It has been designed to summarize large graphs and allow their visualization. This algorithm works in two steps: (1) the user selects the variables (description of nodes), the relationships (description of arcs) and sets the size of the aggregated graph (number of clusters), and (2) the aggregation procedure is composed of two completely independent steps: (a) aggregation based on the set of variables: A-grouping - all nodes belonging to a cluster must have the same values on all variables and (b) aggregation based on the set of relationships: (A,R) clustering - all nodes belonging to the same cluster must have the same list of neighboring clusters.

If this algorithm is based on node and arc descriptions, the starting point is a grouping on attributes, then the groups are divided according to neighborhood to find the summary of size k with the best quality. This algorithm has several limitations: (1) it only applies to homogeneous graphs, i.e. where all nodes have the same description, (2) aggregation is very rigid and is based on a Cartesian product of all modalities of categorical variables, and (3) it is inefficient when the number of categorical variables is high and there are heterogeneous relationships, as it leads to an increase in the number of small clusters.

An extension of this algorithm has been proposed in [23], also allowing to process heterogeneous graphs. The algorithm is based on a first phase to reduce the number of clusters during the *A-clustering phase* and two new criteria for evaluating the quality of aggregation during the *(A-R)-clustering phase* have been defined. Two nodes are assigned to a cluster according to their common neighbors. The algorithm starts with the attribute-based clustering phase and then iteratively divides the existing clusters until k clusters are reached. The division phase is based on the notion of a *central node* to separate a group into two: the first group contains the central node and its neighbors, and the second group contains the other nodes.

1.2 Graphs and Multiplex

In the paper we will use the following definitions.

Definition 1 (Graph).
A Graph *or* single-layer graph *G is defined as a pair of sets: $G = (V, E)$, where V is the set of* vertices *or* nodes*, and $E \subseteq V \times V$ is the set of* edges *or* links *which connect vertices.*

Two nodes $u, v \in$ are connected *if the couple $(u, v) \in E$.*

A graph can be directed *or* undirected*. In the later case, if $(u, v) \in E$ then we have also $(v, u) \in E$.*

If V is finite of size $n \in \mathbb{N}$, then G is called finite graph *and nodes of V can be labelled $v_i, i \in \{1, \cdots, n\}$ and $V = \{v_1, \cdots, v_n\}$.*

We denote $\mathbb{G}_{[n]}$ the set of all graphs with n nodes.

Definition 2 ((Un)Weighted Graph).
The topology of a graph G a can be captured using a mapping $\omega : V \times V \to \mathbb{I} : (u, v) \mapsto \omega(u, v)$.

If $\mathbb{I} = \{0, 1\}$, then G is called an unweighted graph *and then :*

$$\omega(u, v) = \begin{cases} 1 \text{ if } (u, v) \in E, \\ 0 \text{ otherwise.} \end{cases} \tag{1}$$

If $\mathbb{I} = [0, 1]$ then G is called a weighted graph.

Definition 3 (Adjacency Matrix).
If G is a finite graph then the complete topology of G can be captured in the adjacency matrix *$A = (a_{i,j}) \in [0, 1]^{n \times n}$ seen as the result of the mapping ω :*

$$\omega : V \times V \to [0, 1] : (v_i, v_j) \mapsto \omega(v_i, v_j) = a_{i,j} \tag{2}$$

where $[0, 1]^{n \times n}$ is the set of square matrix of order n with values in $[0, 1]$.

If G is undirected then A is a symmetrical matrix.

Definition 4 (Adjacency Matrix Mapping).
The adjacency matrix A can be seen as the result of a mapping μ such

$$\mu : \mathbb{G}_{[n]} \to [0, 1]^{n \times n} : G \mapsto \mu(G) = A. \tag{3}$$

Conversely, as any given matrix $A \in [0, 1]^{n \times n}$ can be interpreted as a weighted adjacency matrix, we denote $\mu^{[-1]}$ the mapping

$$\mu^{[-1]} : [0, 1]^{n \times n} \to \mathbb{G}_{[n]} : A \mapsto \mu^{[-1]}(A) = G_A \tag{4}$$

where G_A is a generic graph with adjacency matrix A.

Of course $\mu(G_A) = \mu(\mu^{[-1]}(A)) = A$, but $\mu^{[-1]}(A) = \mu^{[-1]}(\mu(G)) = G_A$ where G and G_A are equivalent (i.e. they have the same set of edges).

Definition 5 (Multiplex).
A multiplex network *is a finite set of graphs $\mathcal{G} = \{G^{[1]}, \cdots, G^{[p]}\}$, where each graph $G^{[\alpha]} = (V, E^{[\alpha]})$ has the same set of nodes V, but a distinct set of edges $E^{[\alpha]} \subseteq V \times V$.*

Edges of $\mathcal{G}$ are thus determined by three values : the nodes u, v implied in the relationship and α, the level of the layer wher the relation take place. Then $\mathcal{G}$ can also be denoted $\mathcal{G} = (V, \mathcal{E})$ where

$$\mathcal{E} = \{(u, v, \alpha) | u, v \in V, \alpha \in \{1, \cdots, n\}\} \tag{5}$$

We denote $\mathbb{M}_{[n]}$ the set of all multiplex with n nodes.

Symmetrically to the term *multiplex*, single layer graph can be called *monoplex* graphs.

Following [7, 18] the tensor notation can be used to describe the set of adjacency matrices.

Definition 6 (Adjacency Tensor).
For a multiplex *$\mathcal{G} = \{G^{[1]}, \cdots, G^{[p]}\}$, the adjacency matrix of $G^{[\alpha]}$ is denoted $A^{[\alpha]}$. And the* adjacency tensor *is formed by the set of adjacency matrices of its single layers components :*

$$\mathbf{A} = \left(A^{[\alpha]}_{i,j}\right) \tag{6}$$

where $\mathbf{A}$ is a tensor with one contravariant *index (denoted with an upper index, superscript), and two* covariant *index (denoted with a lower index, subscript).*

Definition 7 (Adjacency Tensor Mapping). *The adjacency tensor $\mathbf{A}$ can be seen as the result of a mapping τ such*

$$\tau : \mathbb{M}_{[n]} \to [0, 1]^{1}_{\mathbf{2}} : \mathcal{G} \mapsto \tau(\mathcal{G}) = \mathbf{A} \tag{7}$$

where $[0, 1]^{1}_{\mathbf{2}}$ is the set of tensors with one contravariant index in $\{1, \cdots, p\}$ and two covariant index in $\{1, \cdots, n\}$ with values in $[0, 1]$, and $p, n \in \mathbb{N}$.

Conversely, as any given tensor $\mathbf{A} \in [0, 1]^{1}_{\mathbf{2}}$ can be interpreted as an weighted adjacency tensor, we denote $\tau^{[-1]}$ the mapping

$$\tau^{[-1]} : [0, 1]^{1}_{\mathbf{2}} \to: \mathbb{M}_{[n]} : \mathbf{A} \mapsto \tau^{[-1]}(\mathbf{A}) = \mathcal{G}_{\mathbf{A}} \tag{8}$$

where $\mathcal{G}_A$ is a generic multiplex with adjacency tensor $\mathbf{A}$.

Of course $\tau(\mathcal{G}_{\mathbf{A}}) = \tau(\tau^{[-1]}(\mathbf{A})) = \mathbf{A}$, but $\tau^{[-1]}(\mathbf{A}) = \tau^{[-1]}(\mu(\mathcal{G})) = \mathcal{G}_{\mathbf{A}}$ where $\mathcal{G}$ and $\mathcal{G}_{\mathbf{A}}$ are equivalent (i.e. they have the same set of edges).

In the single layer graphs framework, the goal of community detection is to divide the nodes into communities, such nodes of a community must be more densely connected with nodes of his cluster, than with nodes outside of the cluster [11, 28].

Definition 8 (Community).
For a given graph G, a community *is a set of node more densely connected with nodes of inside of the community, than with nodes outside of the community.*
A community detection *algorithm is any algorithm κ which, given G produce a* labelled *graph G_κ, where each produced label gives the community of the corresponding node.*

Definition 9 (Multidimensional Community).
For a given multiplex $\mathcal{G}$, a multidimensional community *is a set of node more densely connected with nodes of inside of the community, than with nodes outside of the community in the multiplex graph.*
A multidimensional community detection *algorithm is any algorithm K which, given $\mathcal{G}$ produce a* labelled *multiplex graph $\mathcal{G}_K$, where each label gives the community of the corresponding node.*

In the multiplex framework several definitions of what is a "good multiplex community" were proposed, and then several types of algorithms exist. Among these we find algorithms which perform a flattening of the multiplex into a single layer graph, followed by classical (monoplex) detection community detection applied on the resulting graph. Even if this approach can be seen as "naive", it have the main advantage that monoplex algorithms can be used to perform the community detection.

Definition 10 (Flattening).
Let $\mathcal{G}$ a multiplex. A flattening *process is any mapping ϕ*

$$\phi : \mathbb{M}_{[n]} \to \mathbb{G}_{[n]} : \mathcal{G} \to G_\phi$$

which produce a monoplex starting from a multiplex.

Using the previous definitions the multidimensional community detection process K_ϕ using the flattening mapping ϕ can be summarized by the following workflow :

$$K_\phi : \mathcal{G} \xrightarrow{\phi} G_\phi \xrightarrow{\kappa} G_{\phi,\kappa} \xrightarrow{\chi} \mathcal{G}_K \tag{9}$$

where, κ is any community detection algorithm, χ is a function that, for each monoplex community c, links the multidimensional community for the nodes of c in $\mathcal{G}$.

Let us define the type of functions used for ϕ in this paper.

2 Aggregation using quasi-arithmetic means

2.1 Statement of the problem

If $\mathcal{G} = \{G^{[1]}, \cdots, G^{[m]}\}$ then τ can be seen as a multiple "use" of μ :

$$\tau(\mathcal{G}) \approx \Big(\mu(G^{[1]}), \cdots, \mu(G^{[p]})\Big) \tag{10}$$

if we interpret the tensor $\mathbf{A}$ as collection of matrix. In this case the flattening can be performed using an aggregation function $\mathcal{M}$ on these matrices :

$$\mathcal{M}\left(\mu(G^{[1]}),\cdots,\mu(G^{[p]}\right) = A_{\mathcal{M}} \tag{11}$$

Therefore in the process described in (9), ϕ can be decomposed as the following composition :

$$\phi : \mathcal{G} \xrightarrow{\tau} \mathbf{A} \xrightarrow{\mathcal{M}} A_{\mathcal{M}} \xrightarrow{\mu^{[-1]}} G_{A_{\mathcal{M}}} \tag{12}$$

and then the full community discovery process is given by :

$$K_{\mathcal{M}} : \mathcal{G} \xrightarrow{\tau} \mathbf{A} \xrightarrow{\mathcal{M}} A_{\mathcal{M}} \xrightarrow{\mu^{[-1]}} G_{A_{\mathcal{M}}} \xrightarrow{\kappa} G_{A_{\mathcal{M}},\kappa} \xrightarrow{\chi} \mathcal{G}_{\mathcal{M},K} \tag{13}$$

The difficulty is to know which function to use for $\mathcal{M}$.

2.2 Quasi-arithmetic means

The idea of performing the flattening operation using the arithmetic mean on the adjacency matrices collection $\left(\mu(G^{[1]}),\cdots,\mu(G^{[p]}\right)$ is considered as a naive approach [22], and we agree because it is well known that this measure is sensible to extreme values. But, arithmetic means are very simple tools to aggregate values. A most general type of means are the class of *quasi-arithmetic means*. They were already used to cluster complex data [5], and one of these means, the power mean, has already used on Laplacian of graphs [25].

Definition 11 (quasi-arithmetic means).
Let $[a,b]$ be a closed real interval and $p \in \mathbb{N}_0$. A quasi-arithmetic mean is a function $\mathcal{M}_{\varphi}^{(p)} : [a,b]^p \to [a,b]$ defined as follows:

$$\mathcal{M}_{\varphi}^{(p)}\left(\overrightarrow{u}\right) = \mathcal{M}_{\varphi}^{(p)}\left(u_1,\ldots,u_p\right) = \varphi^{-1}\left(\sum_{i=1}^{p}\alpha_i\varphi(u_i)\right) \tag{14}$$

where φ is a strictly monotonous continuous function defined on $[a,b]$, $\forall i \in \{1,\cdots,p\} : \alpha_i{\in}[0,1]$ and $\sum_{i=1}^{p}\alpha_i = 1$

Quasi-arithmetic averages are an extension of conventional averages [10]. If $\varphi(x)$ is equal to $x, x^2, \log x$ or x^{-1} then the above expression generates respectively the following classical averages: arithmetic, quadratic, geometric and harmonic. In the following we will denote $\mathcal{M}_{id}$ the classical arithmetic mean, because its generator is the identity function $f(x) = x$. A basic property of quasi-arithmetic averages is that [2]:

$$\min\left(u_1,\ldots,u_p\right) \leq \mathcal{M}_{\varphi}^{(p)}\left(u_1,\ldots,u_p\right) \leq \max(u_1,\ldots,u_p) \tag{15}$$

Then using such aggregation tools on the values of adjacency matrices in $[0,1]^{n\times n}$ leads to values in the same interval that can be interpreted as adjacency values.

Quasi-arithmetic means has several interesting properties [19]: *idempotence* ($\mathcal{M}_\varphi^{(p)}(u,\ldots,u) = u$, $\forall u \in [a,b]$) , *continuity* in each argument ($\forall p \in \mathbb{N}_0$, $\mathcal{M}_\varphi^{(p)}$ is a continuous function on $[a,b]^p$), *strictly increasing* function in each argument ($u_i < u_i' \Rightarrow \mathcal{M}_\varphi^{(p)}(u_1,\ldots,u_i,\ldots,u_p) < \mathcal{M}_\varphi^{(p)}(u_1,\ldots,u_i',\ldots,u_p)$), *symetry* (if π is a permutation on $\{1,\ldots,p\}$, then $\mathcal{M}_\varphi^{(p)}(u_1,\ldots,u_p) = \mathcal{M}_\varphi^{(p)}(u_{\pi(1)},\ldots,u_{\pi(p)})$) and *decomposable* (if $\mathcal{M}_k = \mathcal{M}_\varphi^{(k)}(u_1,\ldots,u_k)$), then $\mathcal{M}_\varphi^{(p)}(u_1,\ldots,u_k,u_{k+1},\ldots,u_p) = \mathcal{M}_\varphi^{(p)}(\mathcal{M}_k,\ldots,\mathcal{M}_k,u_{k+1},\ldots,u_p)$).

An interesting property of quasi-arithmetic means is given using the Jensen's inequality which states that if φ is convex then

$$\varphi\left(\frac{\sum \alpha_i u_i}{\sum \alpha_i}\right) \leq \frac{\sum \alpha_i \varphi(u_i)}{\sum \alpha_i} \tag{16}$$

and as the growth of φ implies the growth of its inverse, we therefore conclude that in the case of a convex and increasing generator the arithmetic mean will be lower than the quasi-arithmetic mean. Of course if the function is decreasing, the order between the two means will be reversed. Finally if we take into account that the inequality (16) is reversed if φ is concave, then we get the table 1. As

φ	Increasing	Decreasing
Convex	$\mathcal{M}_{id}^{(p)}(\overline{u}) \leq \mathcal{M}_\varphi^{(p)}(\overline{u})$	$\mathcal{M}_{id}^{(p)}(\overline{u}) \geq \mathcal{M}_\varphi^{(p)}(\overline{u})$
Concave	$\mathcal{M}_{id}^{(p)}(\overline{u}) \geq \mathcal{M}_\varphi^{(p)}(\overline{u})$	$\mathcal{M}_{id}^{(p)}(\overline{u}) \leq \mathcal{M}_\varphi^{(p)}(\overline{u})$

Table 1. Comparison between $\mathcal{M}_{id}^{(p)}(\overline{u})$ and $\mathcal{M}_\varphi^{(p)}(\overline{u})$ according to the properties of φ.

a direct consequence of the Jensen inequality, if φ' is a decreasing and convex generator, while φ'' is decreasing and concave generator, then for any vector of value $\overrightarrow{u} = (u_1,\ldots,u_p)$ we have

$$\min(\overrightarrow{u}) \leq \mathcal{M}_{\varphi'}^{(p)}(\overrightarrow{u}) \leq \mathcal{M}_{id}^{(p)}(\overrightarrow{u}) \leq \mathcal{M}_{\varphi''}^{(p)}(\overrightarrow{u}) \leq \max(\overrightarrow{u}) \tag{17}$$

In our case we will use $\mathcal{M}^{(p)}$ with $u_k = A_{i,j}^k$. Each value $A_{i,j}^k$ give the strength link between node i and node j in level k of $\mathcal{G}$. We know that these adjacency values can be interpreted as a *similarity* between the two nodes, and conversely in spectral clustering, similarities between two objects are interpreted as adjacency values [24]. With this interpretation we can understand (17) as the fact that $\mathcal{M}_{\varphi'}^{(p)}$ promotes dissimilarity between node i and node j while $\mathcal{M}_{\varphi''}^{(p)}$ promotes similarity. Then quasi-arithmetic means offer some fine tuning compared to arithmetic mean depending of the chosen generator and its convexity (or concavity), and furthermore when the generator is parameterized, whith an influence of the parameter θ on the shape of φ.

Then we have to choose the generator φ. The shape can be ascending or descending, concave or convex. Taking into account that the values $A^k_{i,j}$ are adjacency values, but could also coming from the similarity or dissimilarity measures between the attributes of nodes i and node j we prefer to use a strictly decreasing generator, because such generators are able to join similarities and dissimilarities in the same calculation, knowing that measures of similarity and dissimilarity are linked by a strictly decreasing function. But this choice is not restrictive.

In this paper, the chosen generator is a simple strictly decreasing function which will be presented in section 3.

Once the adjacency values/similarities are merged, it is possible to apply a classical clustering algorithms κ based on similarity, such as spectral clustering or other algorithms such as community detection based on betweeness measurement or other.

3 Experiments

This section first presents the dataset and the algorithm parameters. Then, the results obtained are detailed.

3.1 Dataset and parameters

The idea in our case of graph clustering is to use quasi-arithmetic averages to merge similarities between different nodes by considering skills as the link or similarity.

The experiments were performed on the Human Resource sector. The main idea is to link individuals and job offers with competences and thus eventually make recommendations. The objective is to cluster groups of individuals and offers that share the same skills, as well as the relations linking them. The added value of this approach is to take into account several types of relationships between individuals, offers, individuals and offers (via skills) and to consider several types of attributes in nodes. The data used for this experiment are individuals, job offers and skills, as well as the relationships between individuals and skills and between skills and job offers. The initial dataset includes 410,256 individuals, 19,988 jobs and 13,382 skills. This dataset does not contain any personal data.

We build matrices on the fly by selecting a set of skills, initially about 30 that appear more than 400 times in the competency offer correspondence, with the understanding that one matrix would have to be built per skill. We build matrices offer-offer, individual-individual, offer-individual whose intersection is not empty if the nodes are linked by a skill. The post code is available in both the offer and the skill node, so we will exploit it by building an additional matrix, with the idea of making groupings on both skills and geographical distance. The total number of matrices will correspond to the set of matrices for the selected competencies plus a distance matrix based on postal codes.

Weights can be set on geographical distance, skills, or other attributes to be taken into account in calculating the arithmetic mean. We defined a threshold for the distance (30 km) to avoid having a complete graph as a result of quasi-arithmetic means. The end user could set sliders for these weights at the application level, e.g. indicate a maximum geographical distance, select a number of skills, weight the geographical distance more heavily in relation to the skills or vice versa, etc.

Competency matrices contain binary values: set to 1 when there is a competency linking the nodes (Individual-Individual, Offer-Individual and Offer-Offer) and to 0 otherwise. The distance matrix contains the geographical distance between postal codes with a maximum distance threshold that can be set (always on the same sub-matrices). The objective is then to merge all the matrices to produce a global view and to make blocks of skills emerge. The Gaussian density function where used to transform distances/dissimilarities into similarities/adjacencies.

We have chosen to implement the function shown in equation (18) as a generator for computing the quasi-arithmetic means.

$$\varphi_\theta(t) = 1 - t^\theta \tag{18}$$

because this decreasing function has as main advantage the fact that according to the value of the parameter θ, the function is convex or concave. If the value of the parameter is less than 0, the function is convex and the quasi-arithmetic mean will be lower than the arithmetic mean, otherwise it is concave and the quasi-arithmetic mean will be higher than the arithmetic mean.

Then, clusters can be more or less well separated, one will go rather towards similarity when there is a clear separation of clusters, otherwise towards dissimilarity by varying the value of θ around 1. In the first case, the similarity values will be lower than in the case of the arithmetic mean, and the reverse is true in the second case.

We then applied a spectral clustering algorithm on the resulting similarity matrix by testing several configurations of number of clusters using the silhouette method to determine the number of clusters k from the resulting matrix. Many clustering algorithms seek to form clusters according to a more or less established structure (spherical, ellipsoid, etc.), which is not the case with spectral clustering, justifying the choice of this algorithm.

The next step consists in reconstructing a graph from the resulting matrix to visualize it. The choice of what will be visualized must be made in order to obtain a readable rendering to be able to interpret the results. The choice can be made on sub-matrices (individual-individual, individual-offer, offer-offer), on each cluster separately from the resulting matrix or by combining a sub-matrix and a cluster visualization by sub-matrix.

3.2 Results

The first experiment was carried out on the offer-offer sub-matrix, taking into account the most frequent skills, i.e. those with more than 400 occurrences in the offer-skills relationship file. This gives us 30 frequent skills and there are 17 offers containing at least 1 frequent skill. A matrix per skill has been built, and the quasi-arithmetic mean using the generator defined previously has been calculated, with parameter values of θ set to 0.5 and 1.5. These value are chosen to have, respectively a concave generator and a convex one. The same values were applied for all the experiments, i.e. on the combination of individuals and offers.

The test dataset includes the following elements: in the skills node, we have the 32 skills with more than 400 occurrences on the offer-skills link, the individuals node includes the first 50,000 (we have taken a ratio that seems reasonable compared to the 1760 offers), the offers node contains all the skills with those not linked to any skills removed. In the individual-skills relationships, the first million lines (out of 10 million) have been retained.

We now present the results obtained for the offer-offer sub-matrix.

The silhouette was applied, and if we look at the Figure 1 (value of $\theta = 0.5$), the choice of the number of clusters could be dependent on what is wanted from a business point of view. If we choose the value of $k = 4$ which is the best (silhouette at 0.71), we will find clusters of nodes strongly connected by skills. We have

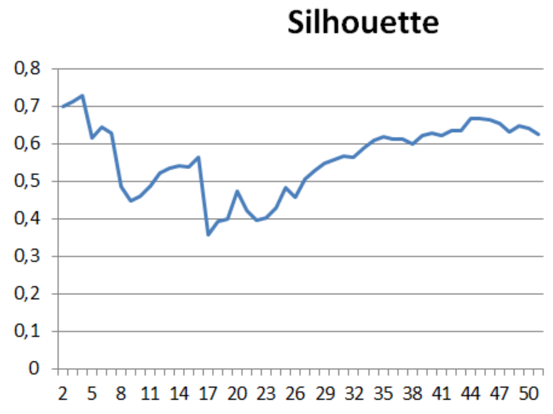

Fig. 1. Curve of silhouette values from 1 to 50 clusters for $\theta = 0.5$.

chosen by default the best silhouette value indicating that the optimal number of clusters is 4. We have three very well identified clusters , and a cluster containing the other nodes (a priori isolated nodes not sharing skills). With a value of $\theta = 1.5$, the optimal number of clusters is 17 (with a silhouette of 0.51). We can note that more clusters are extracted with a concave function than a convex one, but the clusters are of lower quality: the silhouette value is lower so the objects are less well classified. Figure 3 concern the experiment with the value of $\theta = 0.5$ and producing 4 clusters, and shows the histograms of competences on the 4 clusters: we can see that cluster 0 is not well delimited and that it con-

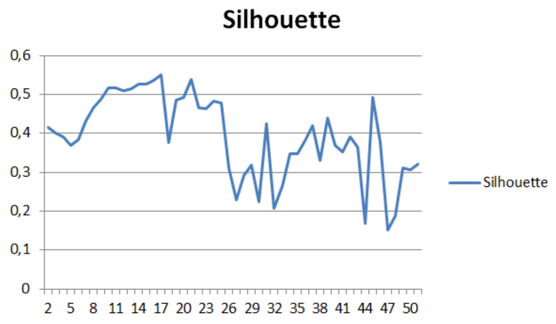

Fig. 2. Curve of silhouette values from 1 to 50 clusters for $\theta = 1.5$ Curve of silhouette values from 1 to 50 clusters for $\theta = 1.5$.

tains a lot of competences, but the other clusters are well dissociated between them. Cluster 0 contains 1379 offers. The skills most represented in Cluster 1 are:

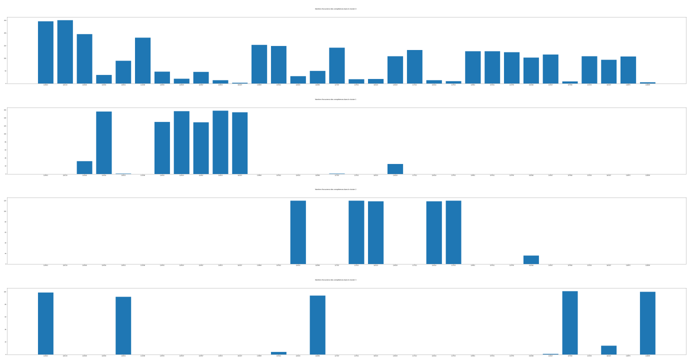

Fig. 3. Competence Histograms for Clusters 0, 1, 2 and 3.

Mixing products and culinary ingredients, preparing dishes for serving, cooking meat, fish or vegetables, peeling vegetables and fruit, dosing culinary ingredients, preparing meat and fish. The cluster is homogeneous in terms of skills and concerns the catering sector mainly at the kitchen level. This cluster contains 160 offers. The skills most represented in Cluster 2 are: clearing a table, setting up the room and the pantry, welcoming the customer when he arrives at the restaurant, cleaning a reception room, setting up tables. The cluster is homogeneous in terms of skills and concerns the catering sector mainly at the room level. This cluster contains 120 offers. The skills most represented in Cluster 3 are: maintenance of premises, monitoring the state of stocks, hygiene and cleanliness rules, dusting floors ..., maintenance of household linen, ... This cluster corresponds to the maintenance of premises and contains 101 offers. Figure 4, shows the graph extracted for clustering on offers with a selection of 4 common skills for visualization. In Figure 4, we see two clearly separated clusters, a third

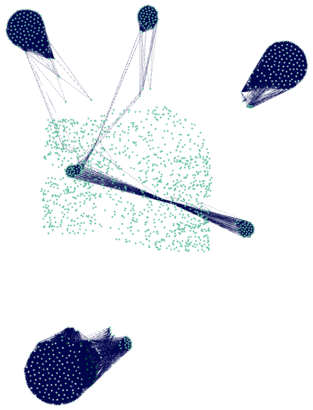

Fig. 4. Complete graph of the clustering supply-skills clustering.

cluster in the upper left a little less well separated and all the other points are in the same cluster. Note that few nodes connect the two denser parts in the centre, and that more clusters would have been formed without these nodes. It may be interesting to see what they correspond to.

4 Conclusion

The experiments carried out made it possible to validate the interest of the proposed approach, with homogeneous clusters produced on the offers according to skills.

The principle of this method is based on the construction of matrices on the elements of interest of the application and leads to many sparse matrices. It is therefore necessary to think about how to store these sparce matrices in an economical way and to optimize the calculations to be carried out.

Parallelisation can be used for sub-matrix calculation or when the user changes the weights. The industrialization of the algorithm depends on the way the application would be used: in batch or real-time, with a run on a server or on the client workstation, etc. It also depends on the volume that will be considered and the expected response time.

Industrialization will require the study of distributed programming frameworks such as Spark or Flink, as well as frameworks for processing large graphs such as GraphX, Pregel, GraphLab or BLADYG proposed in the scientific literature.

A visual mock-up of the application must be created for each type of user. The visualization of the results is also to be studied, depending on the scaling of the different possible tools, and the viewpoint the user is expecting.

Finally, the whole process will need to be automated, including the ability to select nodes and relationships in the graph to perform automatic clustering regardless of the input data. Automation will have to include relevant elements of automatic parameterization, such as the possibility to add weights on the initial matrices, to choose the number of clusters, to select a set of skills, etc. Last, but not least, we have to study and experiment more complex generators to calculate quasi-arithmetic means and study the mathematical properties of convex and concave functions for the generator.

References

1. Bothorel, C., Cruz, J.D., Magnani, M., Micenková, B.: Clustering Attributed Graphs: Models, Measures and Methods. Netw. Sci. **03**(03), 408–444 (2015). https://doi.org/10.1017/nws.2015.9
2. Bullen, P., Mitrinovic, D., Vasic, P.: Means and their inequalities. D.Reidel Publishing Company (1988)
3. Cai, D., Shao, Z., He, X., Yan, X., Han, J.: Mining hidden community in heterogeneous social networks. Proc. 3rd Int. Work. Link Discov. - LinkKDD '05 pp. 58–65 (2005). https://doi.org/10.1145/1134271.1134280, http://portal.acm.org/citation.cfm?doid=1134271.1134280
4. Chen, J., Yuan, B.: Detecting functional modules in the yeast protein–protein interaction network. Bioinformatics **22**(18), 2283–2290 (2006)
5. Cuvelier, E., Aufaure, M.: Classification de données complexes par globalisation de mesures de similarité via les moyennes quasi-arithmétiques. In: Lebbah, M., Largeron, C., Azzag, H. (eds.) Extraction et Gestion des Connaissances, EGC 2018, Paris, France, January 23-26, 2018. RNTI, vol. E-34, pp. 47–58. Éditions RNTI (2018), http://editions-rnti.fr/?inprocid=1002368
6. De Domenico, M., Nicosia, V., Arenas, A., Latora, V.: Structural reducibility of multilayer networks. Nat. Commun. **6**, 1–9 (2015). https://doi.org/10.1038/ncomms7864, http://dx.doi.org/10.1038/ncomms7864
7. De Domenico, M., Solé-Ribalta, A., Cozzo, E., Kivelä, M., Moreno, Y., Porter, M.A., Gómez, S., Arenas, A.: Mathematical formulation of multilayer networks. Phys. Rev. X **3**(4), 1–15 (2014). https://doi.org/10.1103/PhysRevX.3.041022
8. De Domenico, M., Solé-Ribalta, A., Gómez, S., Arenas, A.: Navigability of interconnected networks under random failures. Proceedings of the National Academy of Sciences **111**(23), 8351–8356 (2014)
9. Derrible, S., Kennedy, C.: The complexity and robustness of metro networks. Physica A: Statistical Mechanics and its Applications **389**(17), 3678–3691 (2010)
10. Fodor, J., Roubens, M.: Fuzzy Preference Modelling and Multicriteria Decision Support. Kluwer Academic Publishers (1994)
11. Fortunato, S.: Community detection in graphs. Physics Reports **486**(3-5), 75–174 (2010)
12. Halu, A., De Domenico, M., Arenas, A., Sharma, A.: The multiplex network of human diseases. NPJ systems biology and applications **5**(1), 1–12 (2019)

13. Hristova, D., Musolesi, M., Mascolo, C.: Keep your friends close and your facebook friends closer: A multiplex network approach to the analysis of offline and online social ties. In: Eighth International AAAI Conference on Weblogs and Social Media (2014)
14. Kanawati, R.: Community detection in social networks : The power of ensemble methods. In: Data Sci. Adv. Anal. (DSAA), 2014 Int. Conf. pp. 46—-52 (2014). https://doi.org/10.1109/DSAA.2014.7058050
15. Kanawati, R.: Seed-centric approaches for community detection in complex networks. In: International Conference on Social Computing and Social Media. pp. 197–208. Springer (2014)
16. Kanawati, R.: Multiplex Network Mining : A Brief Survey. IEEE Intell. Informatics Bull. **16**(1), 24–27 (2015)
17. Kim, J., Lee, J.G.: Community detection in multi-layer graphs: A survey. ACM SIGMOD Record **44**(3), 37–48 (2015)
18. Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J.P., Moreno, Y., Porter, M.A.: Multilayer networks. J. Complex Networks **2**(3), 203–271 (2014). https://doi.org/10.1093/comnet/cnu016
19. Kolmogorov, A.N.: Sur la notion de moyenne. Rendiconti Accademia dei Lincei **12**(6), 388–391 (1930)
20. Kurant, M., Thiran, P.: Layered complex networks. Physical review letters **96**(13), 138701 (2006)
21. Latora, V., Marchiori, M.: Is the boston subway a small-world network? Physica A: Statistical Mechanics and its Applications **314**(1-4), 109–113 (2002)
22. Loe, C.W., Jensen, H.J.: Comparison of communities detection algorithms for multiplex. Phys. A Stat. Mech. its Appl. **431**, 29–45 (2015). https://doi.org/10.1016/j.physa.2015.02.089
23. Louati, A., Aufaure, M.A., Lechevallier, Y., Chatenay-Malabry, F.: Graph aggregation: Application to social networks. In: HDSDA. pp. 157–177 (2011)
24. von Luxburg, U.: A tutorial on spectral clustering. Tech. Rep. Technical Report 149, Max Planck Institute for Biological Cybernetics (2006)
25. Mercado, P., Gautier, A., Tudisco, F., Hein, M.: The power mean laplacian for multilayer graph clustering. In: Storkey, A., Perez-Cruz, F. (eds.) 21st International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 84, pp. 1828–1838. PMLR, Playa Blanca, Lanzarote, Canary Islands (09–11 Apr 2018), http://proceedings.mlr.press/v84/mercado18a.html
26. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E **69**(2), 026113 (Feb 2004). https://doi.org/10.1103/PhysRevE.69.026113
27. Rives, A.W., Galitski, T.: Modular organization of cellular networks. Proceedings of the national Academy of sciences **100**(3), 1128–1133 (2003)
28. Shaeffer, S.E.: Graph clustering. Computer Science Review **1**, 27–64 (2007)
29. Steinhaeuser, K., Chawla, N.V.: Community detection in a large real-world social network. In: Social computing, behavioral modeling, and prediction, pp. 168—-175. Springer (2008)
30. Tang, L., Liu, H.: Scalable learning of collective behavior based on sparse social dimensions. In: Proceedings of the 18th ACM conference on Information and knowledge management. pp. 1107–1116. ACM (2009)
31. Tian, Y., Hankins, R.A., Patel, J.M.: Efficient aggregation for graph summarization. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. pp. 567–580. ACM (2008)
32. Wasserman, S., Faust, K.: Social Network Analysis. Cambridge Univ. Press (1994)