Comparison of Community Detection Algorithms on *arXiv* physics collaboration networks

M. Bencheikh lehocine[1], **S. Bouchama**[1], **and J. Lamine**[1]

[1] *University of paris cité, distributed artificial intelligence master*
[*] *Corresponding author: mohammed-amine.ben-cheikh-le-hocine@etu.u-paris.fr*

Compiled April 25, 2022

in this report, we conduct a comparative study of three different approaches to the community detection problem on collaboration networks in the physics field. the data was taken from SNAP website, the Stanford Network Analysis Project. all experiments were performed on google cloud platform. we used intra-community metrics to assess and evaluate resulting partitions and communities from the Louvain algorithm, label propagation and node2vec+kmeans pipeline.

1. INTRODUCTION

Community detection is an arisen field in graph mining and a prominent topic in network science [2] due to its large real-world applications in social networks analysis, biological related problems like protein-protein interactions PPI, and collaboration networks discovery in scientific papers publications or social interactions between users on social media networks (Twitter, Reddit, Facebook ..). techniques of community detection generally refers to the task of finding and discovering the topological structure and organization of complex networks [9]. by constructing a partition of all vertices in a such a way that connections and links between nodes of the same group or partition are dense or compact and links to nodes of other partitions are sparse.

In weighted graphs, the strength of connections is also taken into consideration. discovered communities are also considered as clusters of nodes with similar properties and interactions inside the network. Although the huge efforts and work conducted in this field, This problem is very hard and not yet satisfactorily solved [2], finding communities that satisfy the conditions we mentioned above is considered an NP-hard problem[6]. the optimal community detection is then expensive and nearly impossible for large scale networks. the most popular algorithms are heuristics based on optimizing an objective function to find a satisfactory solution with reasonable time and low computation cost. Figure **??** shows an example figure.

In this work, we present a quantitative comparison of different objective function based community detection algorithms on four distinct graphs representing networks of collaboration between authors of physics-related subjects on *arXiv*. More specifically, we aim to better explicate how research communities are shaped and formed by the topological structure of each network, we explore only the existing edges of an unweighted and undirected graph of collaboration, if there is a citation or

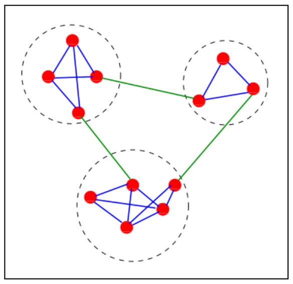

Fig. 1. A simple graph with three communities, enclosed by the dashed circles. Reprinted figure from Ref. (Fortunato and Castellano, 2009)

co-authoring of one or several papers, only an edge between authors vertices is added. To do this, we used four preprocessed datasets from SNAP. First, we apply two heuristics algorithms to these graph-structured datasets to detect and identify communities. we used in this step the Louvain algorithm and label propagation. Next, we use a node embedding based model to construct a continuous representation of the graph and then perform the Kmeans algorithm to cluster to find partitions of different datasets. Lastly, we conducted then a comparative study of our generated partitions by each algorithm using several intra-community measures of communities. to apply inter-communities measures that compute similarities and differences between whole partitions, we need to have the ground truth communities of networks, although, the networks we are studying don't have known partitions as ground truth (although we can do this task by constructing communities of authors of the same laboratory or teams that collaborate together, we don't have this possibility, unfortunately). The remainder of this work is as follows: Section II contains an overview of related works. Section

III presents a detailed explanation of the community detection problem and an overview of collaboration networks. Section IV details the data characteristics upon which we performed our analysis, then presents our methodology and includes relevant details on applied algorithms. Section V contains our experimental results and findings. Section VI consists of the discussion of our results. Section VII details our challenges and future work.

2. LITERATURE REVIEW

Considerable work has been conducted on the community detection problem, we find a wide range of different graph clustering and community discovery algorithms which follow different strategies of constructing the communities as well as methods to assess and evaluate the quality of the identified partitions.

In [1]., Blondel et al. presented a simple heuristic method to extract the community structure by trying to optimize the modularity objective function. they were able at that time to study the structure of large networks with more than 100 million nodes whereas other methods in that time have a limitation of no more than 5 million nodes. they also surpassed the algorithm of Clauset, Newman and Moore CNM in accuracy and speed. the Louvain algorithm inspired a lot of further works, in the University of Virginia, researchers in [6] used a similar technique to find communities inside GitHub open-source software developers. their work offers insights into developers' communities and how that coding language shapes collaboration tendencies to be important contributions [6]. Further details and explanations will be discussed about this method in the 4th section of this report.

the problem with the Louvain method and all maximizing modularity methods resides with the limitations of using only the modularity score which is suitable for cases where we are interested in the process that constructed the graph, not in its interactions and dynamics. Another approach to the community detection problem was proposed by M.Rosvall and D.Axelsson in the map equation paper [10]. they explored the duality that exists between the problem of compressing a data set, and the problem of detecting and extracting structures in this data set. the proposed map equation was based on the Huffman coding process and puts on an abstract form of this duality[10]. Infomap is the implementation of this equation to find communities inside flow-based graphs, it uses random walks in the network and then uses the map equation to optimize the representation of the walking path using Huffman coding, the prefixes of resulting codes for each node refer to the communities codes.

Although the advantages and promising results of this method, we decided not to use it since our data sets present collaboration between researchers. in [4] Satya Keerthi Gorripati1 and Valli Kumari Vatsavayi proposed a distributed implementation of the label propagation algorithm on the spark framework. their approach was based on using a speaker/listener architecture which allows vertices in the graph to communicate community labels to their neighbours and receive labels from their adjacent nodes in the graph. each vertex will be considered as a speaker and listener at the same time and a policy of choosing which labels to propagate was also proposed and tested by the same authors. parallel community detection was aborded to speed up slow methods like LPA in large scale networks. they proved that the proposed algorithm scales well compared to traditional, sequential methods. in the survey [3], authors compared different extensions of the LPA algorithm and works that tried to solve the problem for bipartite networks, directed and weighted graphs. these methods represent a framework based on Label propagation.

representation learning is another promising approach to the community detection problem. embedding based models try to find continuous representations of the network that preserve the spatial and topological properties of each node. DeepWalk algorithm in [8] was the first approach that proposes to use random walks on the graph to generate a corpus of sequences of nodes identifiers that the walker visits while exploring the network. the constructed corpus is then used to train a language model SkipGram that is used for generating word embedding in Naturel language processing related tasks. making use of the latent structure represented by the generated embeddings and possible parallelization that could be applied to this model, especially in the walks generation step makes this approach attracts attention for further ameliorations. Node2vec proposed by [5] uses an ameliorated way of sampling sequences of nodes by making use of biased random walkers and combing between Depth-first search and breadth-first search techniques to explore the neighbourhood of each vertice in the graph[5]. the return parameter P and the in-out parameter Q are used to adapt the algorithm according to the type of the graph and the neighbouring we are looking for. the resulting embeddings for each node can be used further to find clusters using Kmeans, DBscan or Gaussian mixture models to find communities that form a partition of the graph.

The more important related task to community detection is the metrics that are used to compare different approaches and to measure the qualities of a given partition. In [7] the authors explain that analyzing complex networks requires looking at microscopic and macroscopic levels, their main interest was to find out if using automatically generated networks (that are usually used as benchmarks like LFR) can give similar properties with regard to real networks. the normalized mutual information measure NMI is also used to compare the similarity between two partitions generally between the ground truth and the resulting communities. datasets like amazon co-purchasing products graph, and DBLP collaboration network have their ground truth which is not the case for our datasets. in our work, we used intra-community measures like embeddedness, modularity and average distance to compare the implemented methods.

3. PROBLEM STATEMENT

A community detection problem is defined which a groups of nodes that are more similar to each other than to other nodes, more clearly a community B is defined as a set of different nodes in a network $G = (V, E)$:

$$C = \{c_1, c_2, c_3, ..., c_p\}$$

where every $c_i = n_k, n_{k+1}, ...$ and $n_k \in V$ One of the issues that must be addressed is the consistent definition of what a community in a complex network should represent. Nonetheless, an ambiguous definition of the problem is one of the reasons why researchers continue to suggest new approaches in the hopes of resolving a well-defined implementation of this task - often to resolve context-specific applications [2]. The community detection problems change depending on the algorithms used. For the algorithm of Louvain, it is a modularity problem, since the method allows us to partition a network by optimizing the modularity, Modularity is determined by dividing a graph's or network's nodes into communities, the modularity of partitioning c can be defined by

$$M = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{k_i * k_j}{2m}) * \delta(c_i, c_j)$$

where Aij is the value of the adjacency matrix between vertices I and j, ki is the sum of the weights of the edges next to I m is the number of edges in the graph, and is the Kronecker delta. Modularity is mostly utilized in the study of social networks. Modularity optimization is an NP-complete problem modularity maximization can be reformulated as a clique partitioning problem, this magnifies the problem because the resolution of modularity has limits. Optimizing does not identify the graph's inherent community structure, which is represented by individual cliques. For the LPA, it is a method that assigns labels to unlabelled nodes randomly, the algorithm starts with a subset and then propagates it to the whole graph. Possible problems that usually occur are forming excessively large communities, this problem is due to randomness, the behaviour of LPA is not stable. During different runs on the same network, various communities can be observed [4] Also LPA does not look at the neighbours of the node it chooses, so it does not optimize its selections. For the Kmeans, the algorithm assigns data points to one of the K clusters attractively based on their proximity to the cluster centre. When clustering data with various sizes and densities, k-means have problems and there is also the fact that we have to choose k manually. There are problems with the size of the data Community detection methods are very efficient for small graphs but become inefficient when the size of the graph increases, this problem is interesting because it shows that the more data there is, the less the algorithm manages to find the communities, this is a paradox because having a lot of data is seen as an advantage in data analysis.

4. SOLUTION AND METHODOLOGY

A. overview

Given a graph or a network of nodes $G = (V, E)$ where V is the set of all vertices and E the set of all edges, the graph partitioning task consists of grouping vertices into clusters or communities in such a manner that vertices within the same cluster have more connections (edges, and weighted links) than nodes in different clusters. this leads to more similar nodes in the same cluster with properties and interactions that are nearly the same compared to the rest of the network or the graph. although this is just an intuition that is not precise about community detection and we can find other definitions in the literature that point to other aspects of the problem, in our work, this idea is sufficient for our needs. in the next sections of this report, we consider a graph $G = (V, E)$ undirected and unweighted graph. Our graph also can contain self-loops (although initial graphs doesn't contain edges between a node and its self, but the algorithms used can generate such situations). for the sake of clarity, we denote by $n = |V|$the number of nodes and $m = |E|$ the number of edges. $degree(i)$ is the number of edges connected to node i.

We have indicated that the datasets we are using are not categorized into natural clusters, i.e we don't have ground-truth communities to serve as a baseline. our nodes are just identifiers, there are no more features about the authors or scientists in our dataset, Therefore, we can not use a classification model based on node features to do the multiclass classification and use it as a baseline. random partitions may serve as an alternative for that, but we had chosen to only compare the methods to each other because we are already certain that for example, Louvain gives far better results than just assigning randomly labels to graph nodes[**?**].. Thus, We propose two routes for forming communities:

- We will use direct heuristic methods to cluster communities based on network structure and edge relationships. Such methods include Louvain and Label propagation method.

- We cluster each graph G indirectly: Given the single fixed graph G, we generate node embeddings with the trained node2vec model, We feed these node embeddings downstream to vector-based clustering algorithms such as K-Means.

Partitions generated by each route are then compared based on fitness metrics or intera-community measures.

B. heuristic methods:

Louvain algorithm The Louvain algorithm is a hierarchical clustering heuristic that works directly on the graph without any transformation. it is a fast greedy approach that was proven to work effectively on networks with more than 100 million nodes [**?**].. the intuition behind this algorithm is that modularity measure the density of connections within a module or a community. the algorithm then tries to find partitions that maximise this measure. the greedy property of the method means that if we put two nodes in the same, in the next iterations, we can not revise this decision and both nodes will be always in the same community. this can be viewed as a disadvantage of the Louvain method but it is essential to consider its speed and flexibility since it is a hierarchical clustering method, it generates multiple hierarchical partitions that have different community numbers.

Louvain executes two stages in a repetitive way until modularity is maximized. The algorithm is defined as follows:

- Initialize all nodes to be in their own community, for a total of n communities. Then, repeat the following 2 steps:
    * Stage One: modularity optimization.
    * Stage Two: community aggregation.

we implemented a python module for this algorithm, using a scipy sparse matrix to optimize the use of RAM. although it works on small graphs like the karate club network, and the eu-email graph from SNAP, we faced problems when launching it on our datasets, the problem was with the large adjacent matrix we used to calculate modularity. we used then an optimized implementation of Louvain in the python_louvain package.

Label propagation algorithm The label propagation algorithm (LPA) is a local partitional algorithm inspired by epidemic spreading [3]. LPA requires some prepossessing of the input graph such that removing nodes with no edges connected to it (degree(i) = 0) or those with one connected edge (these nodes will take the label of the nodes of their connected nodes, so we remove them to speed up the algorithm). initially, all nodes are given a unique label representing their community (the same thing as in Louvain initialization).

Algorithm 1. Modularity optimization

```
1: procedure STAGE ONE($G, C$)    ▷ inputs : G(V,E), partition C
2:     $L \leftarrow rand(V)$ list of randomly ordred nodes
3:     $stop \leftarrow False$
4:     $Q_p = Modularity(G, C)$
5:     while $stop == Flse$ do▷ continue if Modularity changes
6:         for n in L do
7:             $Ln \leftarrow neighbours(n)$
8:             for j in Ln - C(n) do
9:                 compute $\triangle Q_{n->c(j)}$
10:                $i \leftarrow argMax_j(\triangle Q_{n->c(j)})$
11:        $Q_c = Modularity(G, C)$
12:        if $Q_c == Q_p$ then
13:            $stop \leftarrow True$
14:        else
15:            $Q_p \leftarrow Q_c$
16:    return $C$
```

Algorithm 2. Community aggregation

```
1: procedure STAGE TWO($G, C$)    ▷ inputs : G(V,E), partition C
2:     for $cinC$ do
3:         compute sum of edges to each $c_i inC$
4:         remove all nodes of c from V.
5:         add a unique identifier for c in V.
6:         add edges computed in E.
7:     return $G, C$
```

we perform than a given number of iterations (generally a number of iterations equal to 5 are sufficient to reach a good partition in more than 95% of studied datasets in [3]). in each iteration, all nodes are treated sequentially or in parallel [4] in a random ordering. at each node, we collect all its neighbours' labels and then select the most frequent label among them. if there is more than one frequent label, choose randomly one of them. different expansions and modifications were proposed by researchers to the way of ordering nodes and visiting them, the definition of the neighbourhood of a node and the strategy used to calculate the frequency of labels. an exhaustive study of a lot of implemented LPA can be found in the survey [3]. in the following part, we present a pseudo algorithm that generalises the intuitive idea behind the LPA algorithm.

Algorithm 3. LPA algorithm

```
1: procedure LPA($G, C$)    ▷ inputs : G(V,E), max_iter = 5
2:     generate unique label for each node in V
3:     $L \leftarrow randomize(V)$
4:     $i \leftarrow 0$
5:     while $i < max\_iter$ do
6:         for n in L do
7:             $Lb \leftarrow labels\_from\_nieghbors(n)$
8:             if $|Lb| == 1$ then
9:                 take the unique label in $Lb$ for n
10:            else
11:                choose randomly a label from $Lb$
12:            $i \leftarrow i + 1$
13:    return $Glabels(G)$
```

C. embedding based methods :

node2vec model node2vec is a representational learning model that provides a continuous vector-based representation of the given graph[5]. since we have great advancements in supervised and unsupervised machine learning problems that are based on continuous representation, one can think that using these algorithms in the field of graph mining, especially the community detection problem will do better than traditional heuristics that rely only on the graphical structure of networks. this method is based on a semi-supervised learning approach that tries to learn the latent structure of nodes in the graph by optimizing an objective function using gradient descent. the learning algorithm aims to find node vectors that keep neighbourhood information from the network[5]. the main stages of node2vec are two: first, the model generates sequences of nodes based on random walks on the network. these walks are passed to the skip-gram model that learns a vectorial representation of nodes. the main difference compared to the DeepWalk model resides in the fact that node2vec uses two parameters that orient the random walks to search certain characteristics of the network by exploring more in-depth the neighbourhood of a given node or by exploring more in breadth the graph (doing a tradeoff between depth-first search and breadth-first search). In our case, the choice of these two hyperparameters was difficult due to the weakness of our hardware configuration; if we perform a grid search on two intervals for both p and q of 10 values, we will train approximately 100 models and because the training of models is specific to each graph (this is one of the downsides of node2vec and embedding based models also in general that the trained model is specific to the graph studied). the train of one model on the smallest dataset among the four takes nearly 15 minutes (without counting the time needed to train a clustering method on the embeddings) we need therefore nearly one hundred days using a machine with 8 RAM and intel i5 gen6 processeur. We make a choice to use p = 10 and q = 0.1 to favourite exploring the neighbourhood of the vertices. for the length of the sequence, we had chosen the smallest suggested length by the authors of [5] of 32 nodes and only 10 walks for each node.

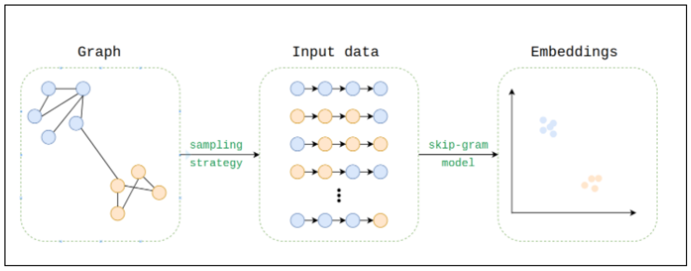

Fig. 2. node2vec intuitions, figure taken from Meduim - towardsdatascience

Kmeans clustering algorithm due to the computation limitation and hardware constraints that we have, K-means was the right choice to cluster 10 * 19k sequence in the largest dataset that we have (in total we have nearly 53000 nodes with 10 samples for each node of 30 dimensions). K-Means is a standard iterative clustering algorithm that partitions a given set of data into k clusters, given a choice of k. Arguably, the best advantage of K-Means is its fast and scalable performance. In the algorithm, cluster centroids determine the centre of clusters, and each data point is assigned to its closest cluster centroid based on the euclidean distance. the problem with k-means is the need to choose the number k of clusters. the choice of this hyper-parameter passes an exhaustive search and evaluates the loss function and chooses the value of K at which the speed of decreasing the loss function starts to decrease. this technique is called the elbow method. this method is not precise, we try a different approach using the silhouette score. for all values of K, we compute a mean silhouette score, then we choose the value that maximizes this objective function. to reduce the number of tested values for K, we used the number of communities found by the Louvain method. for each dataset, we constructed an interval centred on the number of communities given by Louvain and then conducted an exhaustive search on that interval.

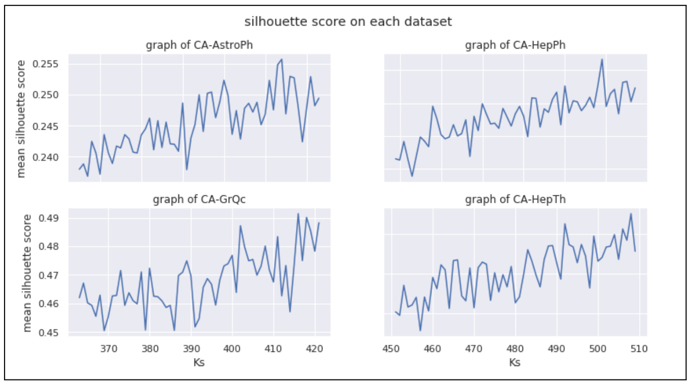

Fig. 3. silhouette mean score for each value of k in the four datasets

5. EXPERIMENTAL RESULTS :

A. Hardware configuration

we used a cluster on google cloud using the student free trial. we configured a machine on Dataproc service with 8 virtual processors on Intel Haswell platform with 30 Go main memory and 50Go as secondary Disque. we used the Bigquery database management system to store our data and a bucket on cloud storage. the used operating system by the cluster was ubuntu 18.04 with Spark of version 2.4. we had chosen to work on google cloud to benefit from its computation power and to make use of BigQuery. our local machines are not enough to do the jobs we performed on the cloud (the main limitation was with the RAM size, Louvain algorithm takes a lot of memory, chrome browser also, which makes the execution on our personal computer nearly impossible). another reason why we chose to use the GCP is the possibility of observing memory processor usage and the number of secondary memory Inputs outputs.

B. Data

we used four collaboration datasets present on the SNAP site, the arXiv Astro Physics collaboration network, arXiv General Relativity and Quantum Cosmology collaboration network, arXiv High Energy Physics - Phenomenology collaboration network and the arXiv High Energy Physics - Theory network. these datasets were constructed from the e-print arXiv and cover scientific collaborations between authors' papers submitted to each category or field. If an author i co-authored a paper with author j, the graph contains an undirected edge from i to j. If the paper

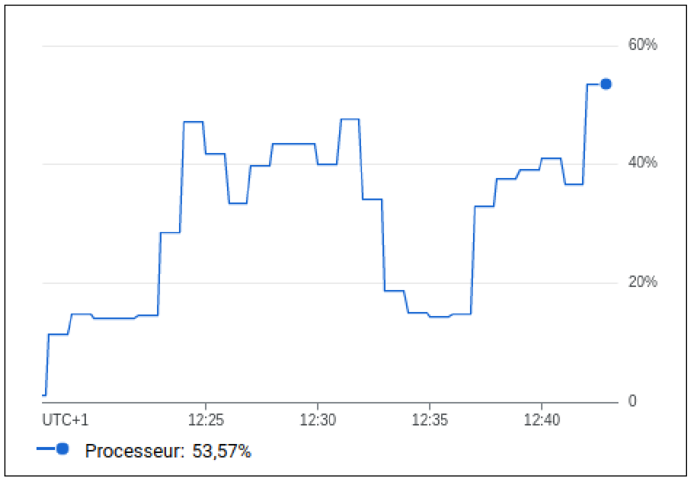

Fig. 4. processor use percentage while training the node2vec model

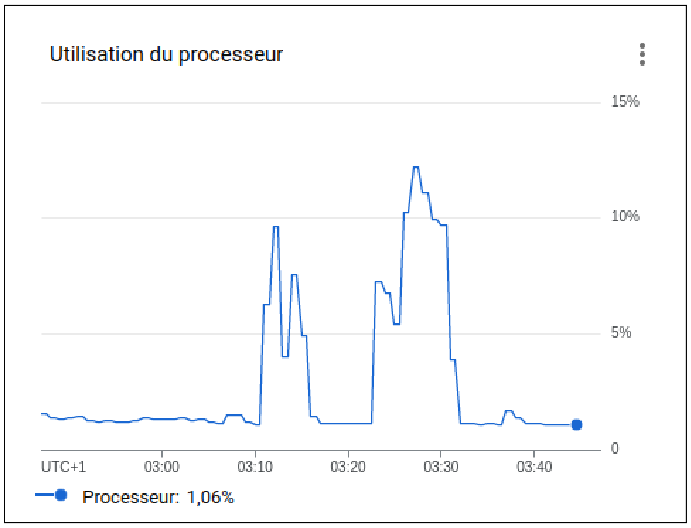

Fig. 5. processor use percentage while runing the louvain algorithm

is co-authored by k authors this generates a completely connected sub-graph on k nodes [site of SNAP]. this table presents a comparison between the four datasets.

C. Quality Metrics :

since we don't have ground-truth communities, we didn't perform external community metrics, although it would be interesting to use them to compare generated partitions of the three algorithms, we used only fitness scores to measure intra-community metrics.

average distance it represents the average of the average sum of the distance between pairs of nodes of the same community. the smallest average means nodes of the same community are near each other.

$$\frac{\sum_{c\in P} \frac{\sum_{i\in c, j\in c} d(i,j)}{|c|}}{|P|}$$

the average embeddedness The embeddedness measure assesses how much the direct neighbours of a node belong to

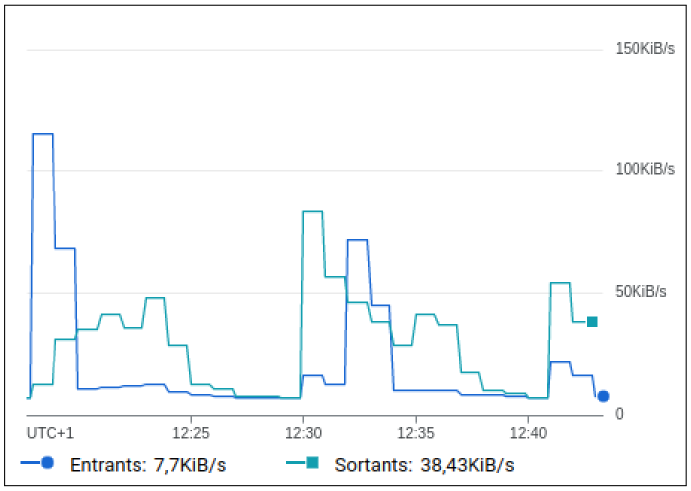

Fig. 6. networks traffic load in the cluster during the training of node2vec

Table 1. statistics of used datasets from SNAP

Dataset statistics	HEP-PH	HEP-TH	ca-GrQc	ca-AstroPh
nodes	12008	9877	5242	18772
edges	118521	25998	14496	198110
Avg_clustering	0.6115	0.4714	0.5296	0.6306
Diameter	13	17	17	14

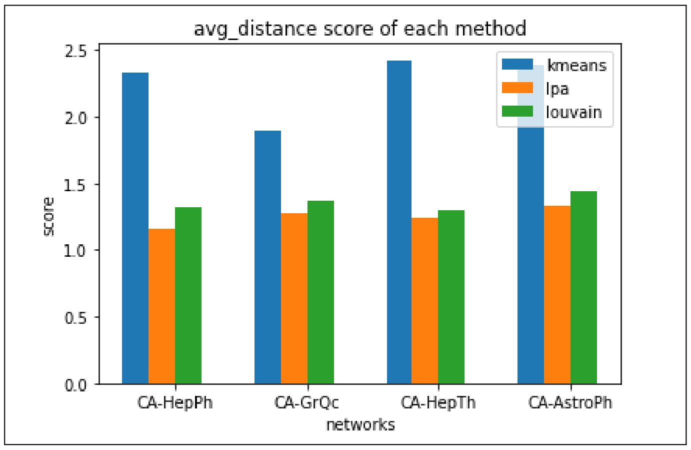

Fig. 7. average distance scores on each dataset

its own community [7].

Table 2. number of generated community by algorithms on each dataset

Algorithm	HEP-PH	HEP-TH	ca-GrQc	ca-AstroPh
Louvain	314	478	392	325
LPA	1201	1764	1026	1070
node2vec+Kmeans	339	508	416	344

$$e = \frac{K_{in}}{K}$$

where K_{in} is the degree of the node inside its community and k its degree in the network.

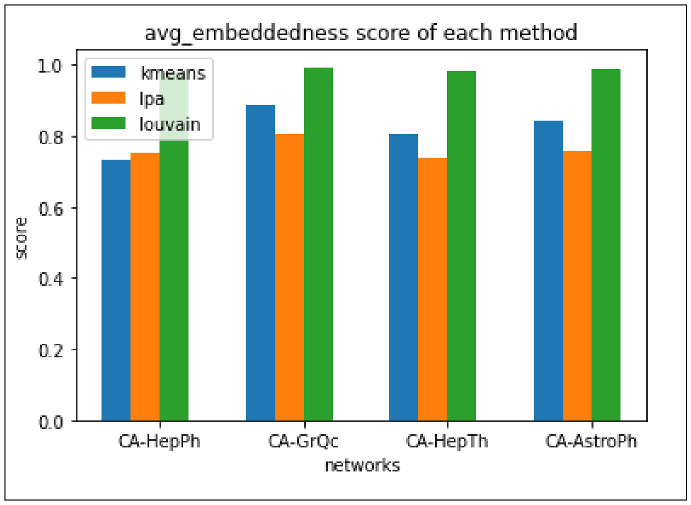

Fig. 8. average embeddedness score on each dataset

the hub dominance the hub dominance of a partition is computed by taking the ratio of the degree of its most connected node (the node with the highest degree) with regard to the theoretically maximal degree within the community (the maximal degree within a community is the cardinal of its nodes set minus one since each node can be connected to all other nodes). this metric represents how much there is dominance by the highest degree node in the community.

$$hub\_dominance = \sum_{c \in P} \frac{maxDegree(c)}{|c| - 1}$$

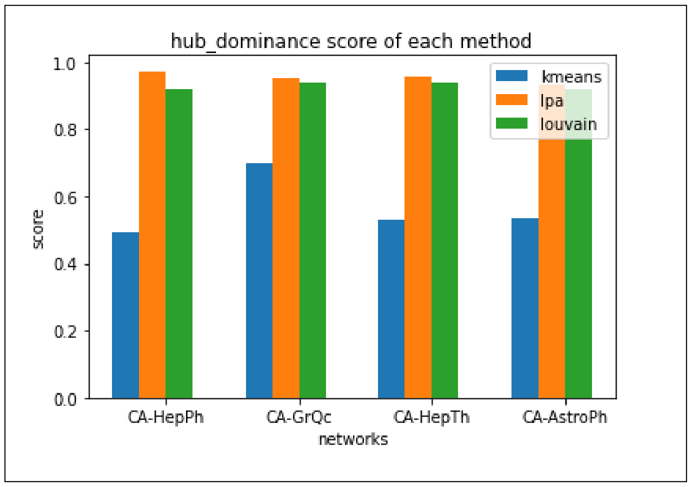

Fig. 9. hub dominance score on each dataset

6. DISCUSSIONS AND CONCLUSION

from given partitions shown in table 2, Louvain algorithm and kmeans generated nearly the same number of clusters on each dataset. this is not suprising since we used the number of clusters obtiend by Louvain to minimize the search cost of parameter k in kmeans. LPA generated approximately a more then the double of communities compared to Louvain and Kmeans. when we observe the processor utilisation by both Louvain and node2vec, we find that training the embedding model takes more the 50% of processor power whereas for Louvain, the use never surpasses 15%. For the network load between the cluster and BigQuery database, we can see the two phases of loading the data and writing the generated embeddings at the end of the training process.

When we observe the fitenss scores, espcially the average distance for partitions generated by each algorithm, Kmeans seems to have the worst scores. its average distance is nearly the double compared to Louvain which proves that the nodes inside a community are not really near to each other in Kmeans partition. the embeddedness scores are nearly the same except for Louvain method which has the highest scores on all the four datasets. LPA wins when it come to the hub dominance score, which is understood-able since nodes with strong connections to its neighbours will certainly force them to have its community label, which conduct to have a dominance score near 1.

REFERENCES

1. Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.
2. Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.
3. Sara E Garza and Satu Elisa Schaeffer. Community detection with the label propagation algorithm: A survey. *Physica A: Statistical Mechanics and its Applications*, 534:122058, 2019.
4. Satya Keerthi Gorripati and Valli Kumari Vatsavayi. Distributed community detection based on apache spark using multi label propagation for digital social networks. *International Journal of Engineering & Technology*, 7(4.5):79–86, 2018.
5. Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.
6. Behnaz Moradi-Jamei, Brandon L Kramer, J Bayoán Santiago Calderón, and Gizem Korkmaz. Community formation and detection on github collaboration networks. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 244–251, 2021.
7. Günce Keziban Orman, Vincent Labatut, and Hocine Cherifi. Qualitative comparison of community detection algorithms. In *International conference on digital information and communication technology and its applications*, pages 265–279. Springer, 2011.
8. Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.
9. Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.
10. Martin Rosvall, Daniel Axelsson, and Carl T Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, 2009.