Parameter-free Clustering of Weighted Graphs using PaCCo

Tim Berger, Student of Business Informatics (B.Sc.)

Abstract—Since the first decade of the 21st century network science is undergoing a huge increase of popularity. [1] Data in form of graphs is increasingly used to observe behavior of agents in networks, to find inter-object interactions and to map out objects like diseases [2] to use the network's topology to discover subtle relationships that might not be discovered otherwise. Weighted graphs enrich these findings by adding a dimension of interaction strength and allowing for deeper understanding of object interaction. With the rising amount of data, elaborate algorithms are needed to find structures and patterns within the graphs to facilitate analysis and interpretation. One very important tool used is the clustering of graphs to detect interesting and meaningful substructures to make sense of the data. But most state of the art clustering algorithms require input parameters like the expected number of clusters. This creates the need to compute the approximate number of clusters in beforehand thus complicating the utilization of such algorithms and their application on real world data as well as research. This paper is presenting and analyzing the PaCCo *(Parameter-free Clustering by Coding costs)* algorithm as introduced by Mueller et al. [3] that overcomes those issues by using the Minimum Description Length (MDL) principle together with a bisecting k-Means approach to automatically and parameter independently cluster weighted graphs on the basis of coding costs and data compression.

Keywords—Weighted Graph, Coding Cost, Clustering, Data Compression

1 Introduction

The analysis of graph data benefits majorly from detecting substructures to infer meaning from a graph's topology and interpret data correctly. Even though a lot of graph clustering algorithms were developed by research, few of them are applicable to weighted graphs, which additionally to their adjacency information contain a dimension of interaction magnitude represented by edge weights. To leverage the supplemental information contained in weighted graphs, the clustering strategy of the PaCCo algorithm as introduced by Mueller et al. [3] maximizes the similarity of nodes, represented as edge weights, as well as the interconnectivity inside a cluster by accounting for few inter-cluster links by employing a strategy based on data compression and the MDL principle. It assigs nodes to clusters that induce the minimal coding effort for data compression. The potential application area for weighted graph clustering is immense, ranging from biological and medical research like the Human Disease Network [2], social network and meta information analysis, the improvement of database systems to stock market analysis and many more. This paper will, after describing related state of the art (SOA) graph clustering algorithms, frame out the structure of the PaCCo technique and analyze its performance in relation to selected SOA algorithms.

2 Related Algorithms

Research developed many useful graph algorithms, but only few can be applied to graphs containing weighted edges. The following section gives a brief overview of existing well-known graph algorithms to later compare their performance with that of PaCCo.

Spectral Clustering is a class of algorithms, that utilize the Eigenstructure of a similarity matrix to find graph partitions. It divides a graph into subgraphs by deleting the weakest links between clusters which results into a disjoint clustering of the graph. A challenge is to find the suitable cluster number of a graph in order to execute the algorithm. Additionally, they are prone to outliers. Zelnik-Manor et al. [4] developed a spectral clustering algorithm

that infers the number of clusters k, making it independent of the user's best choice of k.

Markov Clustering (MCL) can be applied on weighted graphs. By checking for high-flowing regions it determines clusters in the graph [5]. An inflation parameter defines the granularity on how weak and high-flowing regions are separated. It can therefore be compared to the cluster parameter k.

Metis describes a class of techniques for graph partitioning originally proposed by Karypis and Kumar [6]. It recursively bisects a graph and subsequently improves the subgraphs with the help of refinement algorithms. As with other graph clustering algorithms an appropriate number of clusters has to be chosen in beforehand of the execution of the algorithm.

MDL-based Clustering of Unweighted Graphs uses the Cross-Association approach of Chakrabati et al. [7] that employs MDL to create a parameter-free algorithm for partitioning unweighted graphs. The inspiration for the design of PaCCo can be seen here.

With the exception of [7] [4], all presented algorithms are parameter-dependent and suffer from excessive runtime in comparison to PaCCo (see section 4). The algorithmic design of PaCCo, based on MDL, resolves the issues of parameter dependence and also functions as convergence criterion that creates a fully automatic clustering process.

3 The PaCCo Algorithm

This section describes the mathematical and algorithmic formulation of the PaCCo algorithm and is split into two parts: the basic recursion steps and its details of computing the various costs measures used in determining the best fit clustering.

3.1 PaCCo - Basic Steps

In PaCCo, node information is compressed using Huffman coding and obeying to the MDL principle, which allow for similar nodes, in terms of edge weights and connectivity, to be clustered thus resulting in stronger graph compression and lower coding costs. MDL, originally introduced by Rissanen [8] is a measure in information theory that takes advantage of regularities in data and can be defined as "Find the model with which data and the model can be encoded with shortest code length" [9] and thus finds an automatic balance of Goodness-of-fit and model complexity.

The single input of the algorithm is the adjacency matrix A of an undirected weighted graph $G = (V, E)$ with its set of nodes $n = |V|$ and edges $|E|$, and the matrix consisting of $n \times n$ entries of $a_{i,j}$ containing the edge weights $w_{i,j}$ between all nodes v_i and v_j of G:

$$a_{i,j} = \begin{cases} w_{i,j}, & \text{if } e_{i,j} \in E \\ 0 & otherwise. \end{cases} \tag{3.1}$$

As the algorithm adjusts automatically for the number of clusters on basis of coding costs, PaCCo works parameter independent.

PaCCo utilizes a recursive algorithmic pattern to discern the optimal clustering of G into $|C|$ clusters with $C = \{C_1, ..., C_n\}$ by following two basic steps:

I **Graph bisection**
II **Graph clustering** using k-PaCCo {

(a) if split "pays off":
keep clustering
(b) otherwise stop
}

where $k = 2$ in every iteration, since the graph is bisected.

In every recursion step the graph bisection is driven by a better-then-random heuristic that takes into account the weight distribution of the current graph and the potential subgraphs. To initialize two new subgraphs, both centers of the probability density functions (PDFs) of the edge weights of the prospective subgraphs get torn apart and shifted up respectively down by one standard deviation. The figure 1 depicts this process using a Gaussian Distribution (GD) as PDF.

It can be seen that the two new resulting subgraphs share less similarities, but in both distributions the variation of edge weights decreased thus forcing the to-be-assigned nodes of each cluster to be more similar than before.

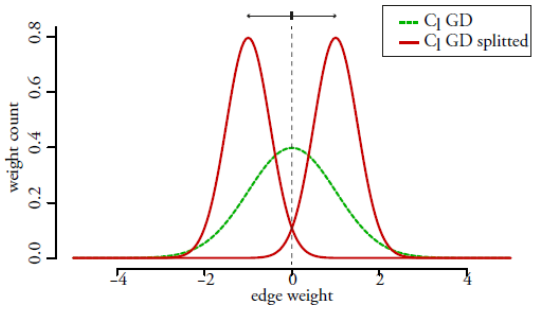

Fig. 1: Initialization of subgraphs;
Source: [3]

After computing various cost measures explained in section 3.2, nodes get reassigned to one of the new clusters $C_l \in C$ by using a k-Means approach and considering the minimal coding costs for v_i in the respective cluster C_l by

$$C_{new}(v_i) = \min_{C_l \in C} c(v_i|C_l). \tag{3.2}$$

Since the sum of the node compression costs for all nodes in a cluster results in the cluster costs

$$c(C_l) = \sum_{v_i \in C_l} c(v_i|C_l) \tag{3.3}$$

the objective function for PaCCo can be modeled as

$$min\ Modelcost(G|C) = \sum_{l=1}^{k} c(C_l) + c(p). \tag{3.4}$$

To account for the costs of storing the clustering model, meaning the μ_{C_l} and σ_{C_l} of every cluster's weight distribution, the term $c(p)$ is added. Since every valid graph splitting will increase the number of distributions by one, this term grows linearly. Ultimately, following the objective function, the split version of the subgraph will be kept if

$$Modelcost(G_{l_{split}}|C_{l_{split}}) < Modelcost(G_l|C_l). \tag{3.5}$$

To get the initial modelcost of the original input graph, k-PaCCo is initialized with k = 1. Note, that after one iteration the membership of all nodes stays the same since it will converge after one step. Afterwards, the graph will be split top-down until the algorithm finds subgraphs that are more expensive then their predecessors in every recursion branch-off.

3.2 PaCCo - Inside the clusters

As the cluster costs and thus the modelcosts are depending on the costs of coding each node in its respective cluster, the questions arises how the computation takes place. In reference to [3] the costs of compressing a node v_i inside C_l is defined as

$$c(v_i|C_l) = c_{weights}(v_i|C_l) + c_{linkage}(v_i|C_l) \tag{3.6}$$

and is depicted in figure 2.

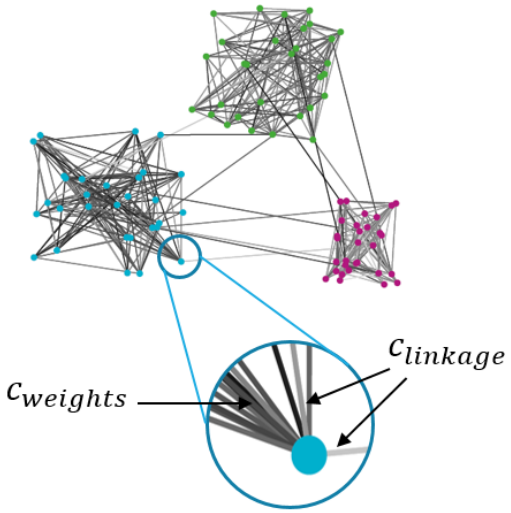

Fig. 2: Visualization of node costs;
Source: own representation based on [3]

As PaCCo tries to maximize the number of nodes that share similar weight information in one cluster, cluster weight representatives (medoids as used in k-Means [10]) are introduced to cluster the nodes accordingly (see equation (3.2)). To evaluate the edge weights of a node v_i according to the edge weights of nodes in its prospective cluster C_l, its set of edges is put into relation to the underlying distribution of weights in C_l. To achieve this, a probability density function with μ_{C_l} and σ_{C_l}

of the cluster's edge weights is computed and every edge weight of v_i evaluated in respect to the PDF. To compress the PDF, Huffman coding is used, which is defined as the inverse logarithm of the probability of an object and is an entropy encoding algorithm for lossless data compression. The costs of coding the weights of a node v_i inside a Cluster C_l can therefore be described as

$$c_{weights}(v_i|C_l) = -log_2(f_{PDF}(v_i)) \tag{3.7}$$

Since a lot of biological processes are originated from a Gaussian distribution (GD), it will be used to further illustrate the mechanics of the algorithm. Note that PaCCo is also able to handle other PDFs and to automatically detect them. To fully evaluate all edges of a node with respect to the weight distribution of the cluster

$$f_{GD}(v_i; \mu_{C_l}, \sigma_{C_l}) = \frac{1}{|C_l|} \sum_{\forall v_j \in C_l} \frac{1}{\sqrt{2\pi\sigma^2_{C_l}}} e^{-\frac{(w_{i,j}-\mu_{C_l})^2}{2\sigma^2_{C_l}}} \tag{3.8}$$

is used.

When assigning a node to a cluster C_l, linkage costs have to be considered. They are computed as follows

$$c_{linkage}(v_i|C_l) = -.5log_2(|e_{i,j'}|) + .5log_2(|e_{i,j''}|) \tag{3.9}$$

where $v_{j'} \in C_l$, $\forall\, e_{i,j'} \in E$ and $v_{j''} \in C$, $\forall\, e_{i,j''} \in E$. The first summand stands for the number of edges from v_i to nodes inside its prospective cluster, the second summand represents the number of edges from v_i to every its adjacent node respectively. Therefore PaCCo punishes edges between clusters and tries to maximize the inner-cluster connectivity.

After all nodes were reassigned to their respective clusters, the μ_{C_l} and σ_{C_l} of every cluster's weight distribution are updated.

With its design, PaCCo maximizes the similarity of nodes in one cluster, the cluster interconnectivity and accounts for few intra-cluster links.

4 PERFORMANCE OF PACCO

To test the performance of PaCCo various tests on synthetic and real world were performed. To compare the results of the synthetic data clustering with the performance of the algorithms described in section 2 the adjusted mutual information (AMI) was used. It is a measure from information theory with a scale of $[0, 1]$ that makes clustering results comparable while correcting for the chance, that a clustering may have been valid through randomness.

The *synthetic data* experiments comprise of three categories (number of noise edges added, variation of weight spacing intervals and variation of the cluster number k) each with three different underlying distributions (Gaussian, Uniform, Laplacian). Only at the experiment of varying the cluster number k, MCL and Metis were able to perform equivalently to PaCCo. In all other experiments PaCCo outperformed every compared algorithm in terms of AMI. Also, it performed well independent of the underlying distribution.

Also the running times of the different algorithms were compared. However it has to be noted that only the approach of Zelnik-Manor et al. (SpectralZM) [4] can be directly compared as it is the only parameter-independent clustering algorithm tested besides PaCCo. To put the performance of PaCCo into full perspective regarding parameter-dependent techniques, the computation time for evaluating the optimal cluster numbers should have been added. The original paper did not include this while comparing running times of the different algorithms, but mentioned the deficiency aside. It was found that PaCCo can outperform SpectralZM by order of 70 and runs in super-linear complexity.

For comparing the results of clustering *real world data*, the information-theoretic measure Modularity was used and adjusted to be applicable on weighted graphs. It takes a higher value if the cluster interconnectivity increases and intra-cluster linking decreases. As data, gene interactions of organisms were used. In terms of finding meaningful clusters, PaCCo outperformed the other algorithms. When the test graphs where enriched for a special activity

inside, only PaCCo was able to identify the new resulting cluster.

Except for the real world data, the experiments can be easily reproduced and checked for bias, as a detailed description of the experiment categories was given. Being tested on synthetic and real world data a full overview of the perfomance of the PaCCo algorithm was given.

5 Conclusion

Conluding from the analysis of the experiments and the description of the algorithmic design, PaCCo offers various benefits in comparison to other state of the art graph clustering techniques: parameter-independent, automatic clustering of weighted graphs in reduced runtime, nearly independent of the underlying distributions and making it easily applicable on real world data and research to discover interesting graph structures.

Although crisp, disjoint clustering facilitates the interpretation of graph data, fuzzy graph clustering [11] could be considered as an optional add-on for PaCCo as it can be in some cases of importance to discover alternative structures of graphs.

In its current state, PaCCo is only able to process data that can be fully loaded into the virtual memory thus imposing a limit on clustering networks that overgrow that size. Even though virtual memory can be manually changed, operation system configuration can complicate that process. To further evolve the potential of PaCCo and expand its applicability in large scale networks, redesigning PaCCo to a distributed computation paradigm is recommended to solve bigger problems and secure its future competitiveness.

References

[1] A.-L. Barabsi, M. Martino, M. Psfai, and P. Hvel, *Network Science*. self published online, 2012.

[2] K.-I. Goh, M. E. Cusick, D. Valle, B. Childs, M. Vidal, and A.-L. Barabsi, "The human disease network," *Proceedings of the National Academy of Sciences*, vol. 104, no. 21, pp. 8685–8690, 2007. [Online]. Available: http://www.pnas.org/content/104/21/8685.abstract

[3] N. Mueller, K. Haegler, J. Shao, C. Plant, and C. Böhm, "Weighted graph compression for parameter-free clustering with pacco." in *SDM*. SIAM, 2011, pp. 932–943.

[4] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering." in *NIPS*, vol. 17, no. 1601-1608, 2004, p. 16.

[5] S. v. Dongen, "A cluster algorithm for graphs," *Information Systems [INS]*, no. R 0010, pp. 1–40, 2000.

[6] G. Karypis and V. Kumar, "A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," 1998.

[7] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos, "Fully automatic cross-associations," in *In KDD*, 2004.

[8] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[9] ——, "Minimum description length principle," in *Encyclopedia of Machine Learning*, C. Sammut and G. Webb, Eds. Springer US, 2010, pp. 666–668.

[10] T. A. Runkler, "Clustering," in *Data Analytics*. Vieweg+Teubner Verlag, 2012, pp. 103–122.

[11] S. Miyamoto and K. Umayahara, "Methods in hard and fuzzy clustering," in *Soft Computing and Human-Centered Machines*, ser. Computer Science Workbench, Z.-Q. Liu and S. Miyamoto, Eds. Springer Japan, 2000, pp. 119–125.