Entropy-based Graph Clustering - A Simulated Annealing Approach

Frédérique Oggier
Division of Mathematical Sciences
Nanyang Technological University, Singapore
Email: frederique@ntu.edu.sg

Silivanxay Phetsouvanh and Anwitaman Datta
School of Computer Science and Engineering
Nanyang Technological University, Singapore
Email: silivanx002@e.ntu.edu.sg, anwitaman@ntu.edu.sg

***Abstract*—We revisit a Renyi entropy based measure introduced originally for image clustering [1], and study its application to graph clustering. To effectuate Renyi entropy based graph clustering, we propose a simulated annealing algorithm. We explore our algorithm's efficacy and limitations with the Karate club graph [2], as well as some other real world network.**

I. Introduction

Given N d-dimensional points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in $\mathbb{R}^d$, the problem of data clustering consists of assigning a label $l \in \{1, \ldots, C\}$ to each $\mathbf{x}_i$. The points with the same label l then form a cluster, and the goal is to put in the same cluster points which are "similar", where similar means close with respect to a given measure which describes the considered problem. It is known [3] that one possible information theoretic formulation for the clustering problem is to suppose that each cluster corresponds to samples from a given probability distribution, and separating the clusters becomes maximizing the distance among distributions. There are standard information theoretic measures that could be considered for this. For example, the Kullback-Leibler divergence $D_{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx$ indeed characterizes how one probability distribution diverges from another expected probability distribution, and the Bhattacharya distance $D_B(p,q) = -\ln \int \sqrt{p(x)q(x)}dx$ measures the similarity of two probability distributions. One issue in using them for clustering though, is that they need to be estimated without making too restrictive assumptions about the data distribution. The mutual information $I(X;Y)$ between two random variables X, Y quantifies the amount of information obtained about one random variable given the other, this is another information theoretic measure which has been well studied for clustering, see e.g. [4], [5], [6], [7], [8].

In [1], an information theoretic measure was proposed for clustering, based on Renyi quadratic entropy, which can be easily computed once given data points. The aim of this paper is to study this measure in the context of graph clustering (see [9] for a survey of classical graph clustering techniques). The contributions of this paper are then:

(1) to propose one possible adaptation of the entropy measure from [1] for graph clustering (in Subsection II-C),

(2) to test this measure on small graphs (in Section II-C),

(3) and to combine this measure with a simulated annealing search, to scale the clustering to larger graphs. This is elaborated in Section III, where we describe our clustering algorithm and study its performance using the Karate club [2] graph and some subgraph of the Bitcoin network. Early results suggest that the entropy measure based clustering algorithm provides a meaningful clustering, even for graphs where the other tested classical clustering algorithms are not convincing.

II. An Entropy-based Measure for Clustering

A. A Sample-based Estimator for Renyi Entropy

The quadratic Renyi entropy of a vector $\mathbf{x} \in \mathbb{R}^d$ is

$$H_2(p) = -\ln \int p^2(\mathbf{x}) d\mathbf{x}$$

where $p(\mathbf{x})$ is the pdf of $\mathbf{x}$. When $p(\mathbf{x})$ is unknown, but samples are available, the pdf can be replaced by a sample-based estimator. The probability P that a vector with pdf $p(\mathbf{x})$ falls in a region $\mathcal{R}$ of $\mathbb{R}^d$ is given by

$$P = \int_{\mathcal{R}} p(\mathbf{x}) d\mathbf{x}.$$

If $\mathcal{R}$ is small enough to ensure that $p(\mathbf{x})$ does not vary too much within it, we can approximate P by

$$P \approx p(\mathbf{x}) \int_{\mathcal{R}} d\mathbf{x} \tag{1}$$

where $\int_{\mathcal{R}} d\mathbf{x}$ is the volume of $\mathcal{R}$. Now if we have N samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ which are independently drawn according to $p(\mathbf{x})$, and there are say k out of the N samples falling with $\mathcal{R}$, then

$$P = \tfrac{k}{N}. \tag{2}$$

Combining (1) and (2) yields

$$\hat{p}(\mathbf{x}) = \frac{k/N}{\int_{\mathcal{R}} d\mathbf{x}}$$

as an obvious estimate for $p(\mathbf{x})$. If now $\mathcal{R}$ is a hypercube in $\mathbb{R}^d$ with edge length h centered at $\mathbf{x}$, then $\int_{\mathcal{R}} d\mathbf{x} = h^d$, and we can introduce the following window function W which indicates whether $\mathbf{x}_i$ is inside $\mathcal{R}$:

$$W_h(\mathbf{x}_i - \mathbf{x}) = \begin{cases} 1 & \text{if } \frac{|x_j - x_{ij}|}{h} \leq \frac{1}{2},\ j = 1, \ldots, d \\ 0 & \text{else.} \end{cases}$$

The total number k of samples falling in $\mathcal{R}$ is thus given by

$$k = \sum_{i=1}^{N} W_h(\mathbf{x} - \mathbf{x}_i)$$

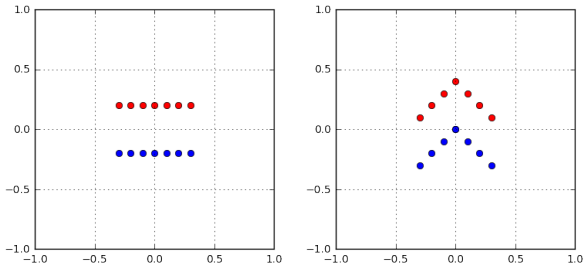

Fig. 1. Two shapes in the plane being distinguished by the considered between-cluster evaluation function, with $\sigma^2 = 0.01$.

and the Parzen window estimator [10] is given by

$$\hat{p}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{h^d}W_h(\mathbf{x}-\mathbf{x}_i).$$

Another typical choice for the window function W_h is the Gaussian function given by $W_\sigma(\mathbf{x}-\mathbf{x}_i) = \frac{1}{\sqrt{2\pi^d}}\exp\left(-\frac{1}{2\sigma^2}||\mathbf{x}-\mathbf{x}_i||^2\right)$, where the covariance matrix Σ is simplified to be $\Sigma = \sigma^2\mathbf{I}_d$ and $h = \sigma$ is a scale parameter, for which

$$\begin{aligned}\hat{p}(\mathbf{x}) &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sigma^d}W_h(\mathbf{x}-\mathbf{x}_i)\\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sigma^d\sqrt{2\pi^d}}\exp\left(-\frac{1}{2\sigma^2}||\mathbf{x}-\mathbf{x}_i||^2\right).\end{aligned}$$

The product of two such Gaussian distributions is [11, 8.1.8]

$$\begin{aligned}&\tfrac{1}{\sigma^d\sqrt{2\pi^d}}\exp(-\tfrac{||\mathbf{x}-\mathbf{x}_i||^2}{2\sigma^2})\tfrac{1}{\sigma^d\sqrt{2\pi^d}}\exp(-\tfrac{||\mathbf{x}-\mathbf{x}_j||^2}{2\sigma^2})\\ =\ &\tfrac{1}{\sqrt{(2\sigma^2)^d(2\pi)^d}}\exp(-\tfrac{||\mathbf{x}_i-\mathbf{x}_j||^2}{4\sigma^2})\mathcal{N}_{\mathbf{x}}(\tfrac{1}{2}(\mathbf{x}_i+\mathbf{x}_j),\tfrac{\sigma^2}{2}\mathbf{I}_d)\end{aligned}$$

where $\mathcal{N}_{\mathbf{x}}(\frac{1}{2}(\mathbf{x}_i+\mathbf{x}_j),\frac{\sigma^2}{2}\mathbf{I}_d)$ denotes a multivariate Gaussian distribution with mean $\frac{1}{2}(\mathbf{x}_i+\mathbf{x}_j)$ and covariance matrix $\frac{\sigma^2}{2}\mathbf{I}_d$. Thus taking the integral over the product simplifies to

$$\tfrac{1}{\sqrt{(2\sigma^2)^d(2\pi)^d}}\exp(-\tfrac{||\mathbf{x}_i-\mathbf{x}_j||^2}{4\sigma^2})$$

and a Renyi entropy sample-based estimator $\hat{H}_2(p)$ is then

$$\begin{aligned}&-\ln\int\tfrac{1}{N}\sum_{i=1}^{N}\tfrac{1}{\sigma^d}W_\sigma(\mathbf{x}-\mathbf{x}_i)\tfrac{1}{N}\sum_{j=1}^{N}\tfrac{1}{\sigma^d}W_\sigma(\mathbf{x}-\mathbf{x}_j)d\mathbf{x}\\ =\ &-\ln\frac{1}{N^2}\sum_{i,j=1}^{N}\tfrac{1}{(2\sigma)^d\sqrt{(2\pi)^d}}\exp(-\tfrac{||\mathbf{x}_i-\mathbf{x}_j||^2}{4\sigma^2}) = \hat{H}_2(p). \quad (3)\end{aligned}$$

B. Entropy Measures for Clustering

The quantity $\hat{H}_2(p)$ in (3) is interpreted as a *within-cluster entropy* [12], since if we consider a single cluster, associated to a single pdf $p(\mathbf{x})$, $\hat{H}_2(p)$ computes an estimate of its entropy.

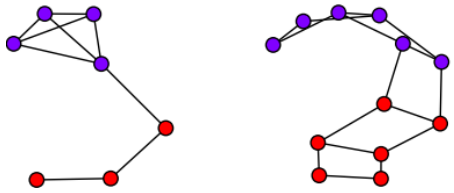

Fig. 2. Two clusters found on a lollipop graph (on the left) and on a ladder graph (on the right), $\sigma^2 = 0.01$.

However, if we use (3) but by summing over two different clusters (there is a probability distribution $p_1(\mathbf{x}), p_2(\mathbf{x})$ associated to each cluster), then as proposed in [1], [12], we get a *between-cluster entropy*

$$D(\hat{p}_1,\hat{p}_2) = -\ln\frac{1}{N_1N_2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\tfrac{1}{(2\sigma)^d\sqrt{(2\pi)^d}}\exp(-\tfrac{||\mathbf{x}_i-\mathbf{x}_j||^2}{4\sigma^2})$$

as a cluster evaluation function that estimates the distance between two clusters, since the between-cluster entropy estimates the distance between sample-based estimation of their distributions. This also gives an estimator of the "cross Renyi entropy" $-\ln\int p_1(\mathbf{x})p_2(\mathbf{x})d\mathbf{x}$ (which is close to, but differs from the Bhattacharya distance).

This between-cluster entropy function was proposed in [1] for image processing applications. Figure 1 illustrates how it helps separating shapes in the plane, where an exhaustive search is done to test all possible labellings for two clusters.

In the case of several clusters with respective pdfs $p_1,\ldots,p_C$, we have the generalized *between-cluster entropy*

$$D(\hat{p}_1,\ldots,\hat{p}_C) = -\ln\frac{1}{N^2}\sum_{i,j=1}^{N}\tfrac{\delta_{ij}}{(2\sigma)^d\sqrt{(2\pi)^d}}\exp(-\tfrac{||\mathbf{x}_i-\mathbf{x}_j||^2}{4\sigma^2})$$

where $\delta_{ij} = 0$ if both $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same cluster and 1 otherwise.

C. Entropy Measures for Graph Clustering

The between-cluster evaluation function depends on the distance $||\mathbf{x}_i-\mathbf{x}_j||^2$ between two data points. In the case of graph clustering, given an undirected graph $G = (V,E)$ with set V of nodes and set E of edges, data points become the nodes, and the distance between two nodes depends on E. Typically, the distance between two nodes in a graph is the number of edges in a shortest path, and we assume that the graph is connected (otherwise, we consider the connected components of the graph separately). If the graph is weighted, then the weight of the edges can be taken into account, but we will consider the case where the edges have no weight, thus the distance $d(u,v)$ between u and v is the length of a shortest path between u and v. Figure 2 shows two small graphs on which the between-cluster evaluation function has been tried for two clusters. Numerical computations are done using Networkx [13], and an exhaustive search is done to test all possible node labellings for two clusters. Figure 3 illustrates the behavior of the between-cluster evaluation function during an exhaustive search for the lollipop graph with 7 nodes shown on the left of Figure 2.

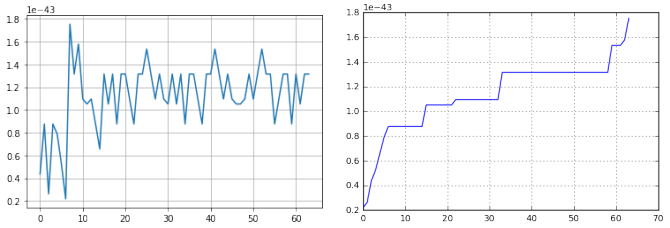

Fig. 3. Fluctuation of the between-cluster evaluation function, for the lollipop graph of Figure 2: on the left, the values taken by the between-cluster evaluation function as a function of the labellings in an exhaustive search; on the right, the same values (but) sorted in increasing order.

III. Numerical Results

An exhaustive search across all possible combinations of clustering for a graph with n nodes, even for 2 clusters, would require the enumeration of $2^{n-1}-1$ combinations, discounting for the equivalent clusters obtained by interchanging all the labels and the two single cluster cases, a solution obviously impractical. We consider a simulated annealing heuristic [14] instead. Simulated annealing is a widely used randomized algorithm technique, which has also been utilized in various graph algorithms, e.g., determine graph bisection [15], thickness [16], coloring [17] and link prediction by deanonymization [18] to name a few. The details of our approach are contained in Algorithm 1.

Algorithm 1 Simulated annealing based graph clustering

for every node in the graph **do** ▷ Initialization
  Choose a label uniformly at random ▷ Create l^0
  ▷ Number of label choices is predetermined by the number of clusters being sought
for chosen # of times **do** ▷ Annealing
  ▷ other termination criteria may be used instead
  ▷ At iteration $t \geq 0$, make a transition from the overall labelling l^t at t to the labelling l^{t+1} at $t+1$ as follows:
  Select a node $v \in V$ uniformly at random
  Assign v a different label chosen uniformly at random, to obtain new overall labelling l^{new}
  Assign $\Delta = D(l^{new}) - D(l^t)$
  ▷ $D(.)$ is the between-cluster entropy
  if $\Delta \leq 0$ **then**
    Set $l^{t+1} = l^{new}$ ▷ Accept the new labelling
  else ▷ The case when $\Delta > 0$
    With probability $e^{-\beta\Delta}$: Set $l^{t+1} = l^{new}$
    With probability $1 - e^{-\beta\Delta}$: Set $l^{t+1} = l^t$
    ▷ parameter β determines the rate of decay
for chosen # of times **do** ▷ Optional: post-processing
  $\forall$ node X s.t. X is in a cluster boundary:
    Move X to best neighbor cluster if it reduces D(.)
  $\forall$ node X s.t. X has multiple neighbors:
    Assign all neighbors the same cluster as X if doing so reduces D(.)

Algorithm 1 assumes a predefined number of clusters, that is, of distinct labels, and mutually disjoint clusters (not fuzzy ones), meaning that at any time point, a unique label is assigned to any node to indicate which cluster it belongs to. The guessed clustering at any step t of the algorithm is denoted by l^t. We initialize the labels uniformly at random (to obtain the labelling l^0). Then, at every step, we pick a node uniformly at random and alter its label to any of the other possible labels uniformly at random, to obtain a new candidate clustering corresponding to l^{new}. If the new clustering decreases the between-cluster evaluation function, then we found a better clustering, chosen for the next round. However, even if the new clustering increases the evaluation function, thus yielding a worse clustering, it may still be considered as the new candidate - but with a probability that decays exponentially with the difference $\Delta = D(l^{new}) - D(l^t)$ of the entropy measures. The rate of decay is parameterized (by β). The rationale behind allowing such "bad choices" occasionally is at the heart of the simulated annealing heuristic to extricate the search from being trapped in local extrema. We will discuss the optional post-processing step afterwards, together with the results shown in Figure 6. The rest of the results discussed are derived from solely the simulated annealing, without the (optional) post-processing step.

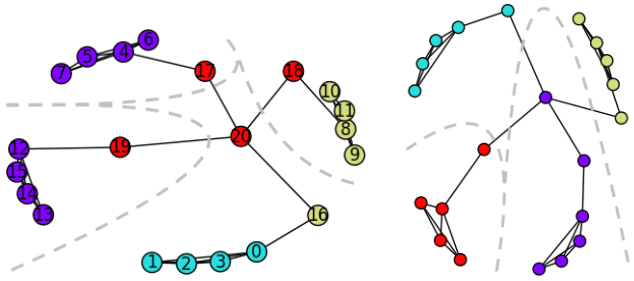

Fig. 4. A clustering with 4 labels: on the left, an interesting (though not optimal) clustering found by the simulated annealing algorithm; on the right, the best clustering found (so far). Specific nodes (may) have different colorings in the two representations. Given the symmetry of the network, the clustering shown on the right is the expected one, given that the central node could have belonged to any of the four clusters. We indicate this expected clustering with dotted lines in the left figure.

We first tested our algorithm with some synthetic graphs. In Figure 4 we demonstrate multiple (four) clusters obtained in a network which comprises four lollipop graphs interconnected with a star at the center. Since the annealing approach is a randomized algorithm which does not carry out an exhaustive search, the results obtained from different runs of the algorithm can be different, and they may diverge from the optimal. We notice this in the left graph of Figure 4, where the obtained result is visibly sub-optimal. One usual remedy is to run the algorithm multiple times, with distinct initializations, and pick the best result from multiple runs of the algorithm, the one we obtained is shown on the right of Figure 4.

Having validated the proposed approach with some simple synthetic graphs, we next study it using a small real social network — the Karate club graph [2], because it is a canonically studied graph in the clustering literature [19], with a

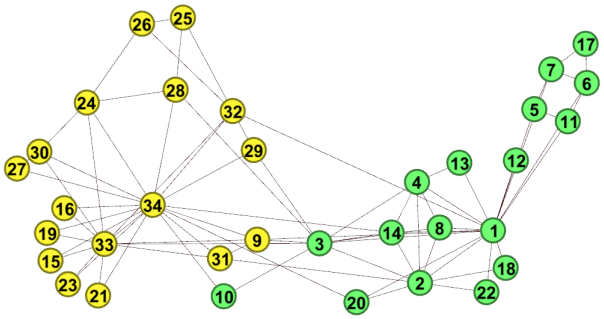

Fig. 5. A clustering of the Karate club graph [2] with two labels.

well understood ground truth. The clusters obtained with our algorithm are shown in Figure 5 [1]. When compared with several existing clustering techniques (using implementations available in Python through *sklearn.cluster* [20]), including k-means, affinity propagation, spectral and agglomerative clustering – the clusters we obtain by our method agree with the results from these other techniques, but for the node 10. In fact, in different runs of our algorithm, the node 10 fluctuates across the two clusters. We show this particular instance because of three reasons: (i) to emphasize the non-determinism of the randomized approach, (ii) it so happens that as per our between-cluster entropy measure, the other possible clustering is in fact marginally (0.347%) worse, and (iii) most interestingly, node 10 had indeed no strong affiliation with either of the factions [2], though the person had eventually joined the faction represented by the left cluster (in yellow). But this sort of "error" is not surprising. For instance, node 9 had in fact a slight leaning towards the left cluster, and our clustering is "correct" (and even agrees with the results from all the above mentioned clustering algorithms), but node "9" still had finally joined the faction represented by the cluster on the right (green nodes). Quoting [2]: '*This can be explained by noting that he was only three weeks away from a test for black belt (master status) when the split in the club occurred. Had he joined the officers' club he would have had to give up his rank and begin again in a new style of karate with a white (beginner's) belt, ...*'.

The next issue we wanted to investigate was how the algorithm performs in the 'wild', for an arbitrary graph which may or not even have very well defined clusters to begin with. We have experimented with subgraphs from the Bitcoin transactions network (undirected version), and report our results and insights using a subgraph comprising of 178 nodes. The results for 2 and 5 clustering are shown in Figure 6. We also show k-means, Ward's agglomerative (with Euclidean affinity) and spectral clustering.

We ran our simulated annealing algorithm for 2000 generations, and repeated the experiments ten times with distinct random initializations. We report the best result obtained from the ten runs. That is 20000 samples as opposed to 2^{177} samples an exhaustive search would have involved. This is to elaborate

[1]The code used can be found at https://github.com/feog/Bitcoin-Analysis

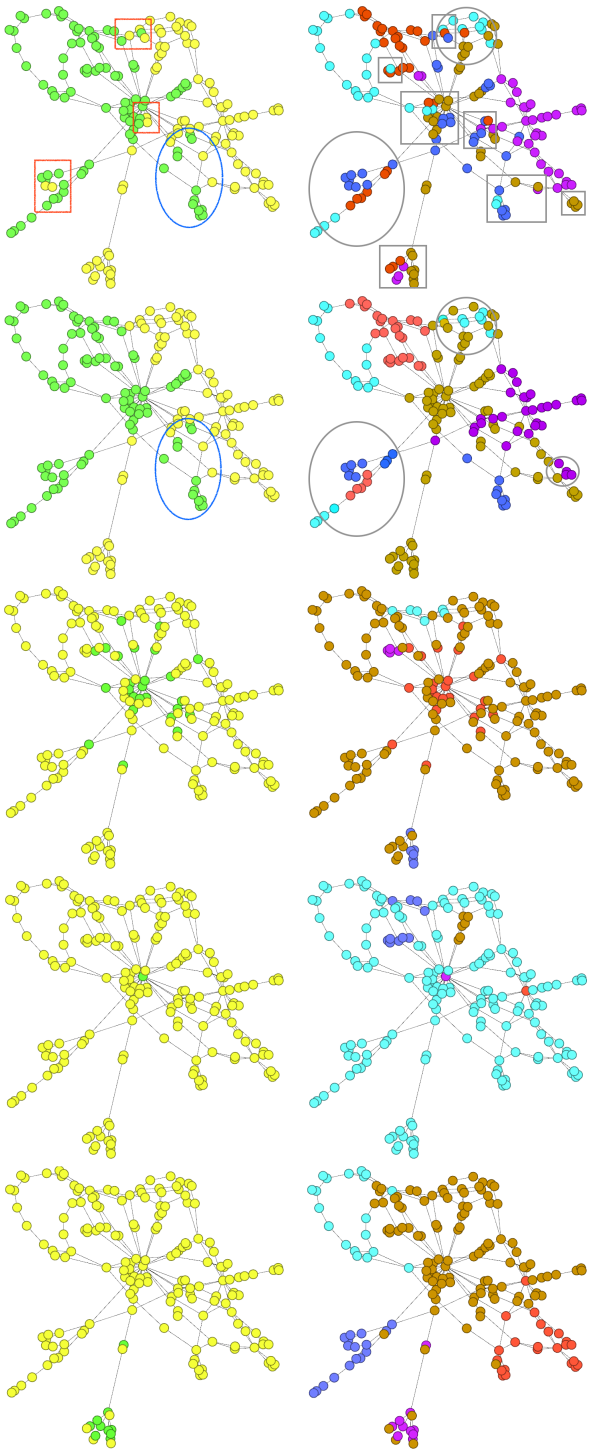

Fig. 6. We show the clusterings of a Bitcoin transactions subgraph of 178 nodes. The plots are organized as follows. On the **left**: 2-clusters; on the **right**: 5-clusters. From **top to bottom**: entropy-based simulated annealing only, simulated annealing with post-processing, k-means, agglomerative (Ward's), spectral clustering algorithms. These plots are drawn using Gephi [21].

on the earlier comment that one can execute the randomized search multiple times, and pick the best result, because an individual run is relatively inexpensive. We used the best result so obtained to further run the post-processing for ten rounds.

In the absence of an established and accepted ground truth (unlike for the Karate club graph case) for the given graph, we discuss the current results qualitatively, and based on the visualization of the clusters. Inspecting the clustering results with simulated annealing, we notice two interesting (and possibly intertwined) issues with our algorithm (we highlight some instances in our plots): There are islands of "sub-clusters" isolated from other nodes supposedly belonging to the same cluster, and embedded inside another cluster. In the extreme case there are isolated nodes. We see such artefacts also from the clustering results from other standard algorithms we tested. Secondly, there are certain regions of the graph, where we notice that if these regions were indeed studied in isolation, there would have been clusters. Yet, when studying the overall graph, we may not want to view these "local" artefacts, particularly if we seek a predefined number of clusters. For the former issue (isolated nodes or islands): given that our simulated annealing carries out a random walk over the search space, by changing the labelling of one node at a time, it may take a very long time to chance upon outliers. In fact, while they are very fast in reaching a reasonably good (approximate) solution, such randomized algorithms are in general known to get very slow as they get close to the optimal. For the second issue (localized clustering artefacts): We are at the moment unsure of the behavior of the between-cluster entropy function, but it certainly does not explicitly discern local versus global differences, and hence may indeed be treating such localized artefacts as separate clusters. Another way to look at this is that, there are just more clusters in the graph than what we are searching for with the parameter choice indicated in the algorithm, which can probably be used to dynamically adjust the parameter itself (something we have not explored yet). Accordingly, we propose some (optional stage in Algorithm 1) heuristic post-processing which make localized decisions based on individual nodes' neighborhood. The essential ideas of the heuristics applied are as follows: Explore whether reallocation of nodes with neighbors from other clusters to one of those neighboring clusters improve the entropy measure; for nodes with multiple neighbors, determine whether assigning all its neighbors to the same cluster as itself improves the measure. We do see the current post-processing heuristics improve the quality of results - but some of the previously mentioned artefacts linger (some instances are highlighted in our plots). Overall, for the two clustering case - only our approach (both variants) resulted in meaningful clusters. For the five cluster case, in additional to ours, the spectral clustering also produced reasonable clusters (and our post-processing heuristic would have further improved its results too). We speculate that for the given graph, our entropy measure favors, and accordingly the algorithm discerns more than five clusters. Further exploration of these issues - as well as quantifying the quality of obtained clusters by other metrics are parts of our ongoing work.

IV. Concluding remarks

In this paper we demonstrate one way to adapt Renyi entropy measure for graph clustering, and propose a practical clustering algorithm using simulated annealing. We demonstrate the efficacy but also limitations of the current algorithm; the latter defining our immediate next steps, namely - determine the suitable number of clusters dynamically, enhance computational scalability by combining the algorithm with hierarchical clustering, and improve post-processing heuristics if still needed. Exploration of parameters σ for the Parzen window estimator and β in the simulated annealing, a more rigorous benchmarking with wider range of data sets as well as algorithms, and generalization of the proposed approach for directed graphs are other aspects for future study.

References

[1] E. Gokcay and J.C. Principe, "Information Theoretic Clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2), 2002.

[2] W.W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups", Journal of Anthropological Research 33(4), 1977.

[3] J. Hartigan, "Clustering Algorithms", *Wiley*, 1975.

[4] L. Faivishevsky, J. Goldberger, "A nonparametric information theoretic clustering algorithm", Intl. Conf. on Machine Learning (ICML) 2010.

[5] M. Wang, F. Sha,"Information theoretical clustering via semidefinite programming", Artificial Intelligence and Statistics (AISTATS) 2011.

[6] A. Müller, S. Nowozin, C. Lampert, "Information theoretic clustering using minimum spanning trees", *Pattern Recognition*, 2012.

[7] M. Sugiyama, M. Yamada, M. Kimura, H. Hachiya, "Information-maximization clustering based on squared-loss mutual information", Neural Computation, 26(1), 2014.

[8] G. Ver Steeg, A. Galstyan, F. Sha, S. DeDeo, "Demystifying Information-Theoretic Clustering", *31st International Conference on Machine Learning*, Beijing, China, 2014. JMLR: W&CP volume 32.

[9] S.E. Schaeffer, "Graph clustering", *Computer Science Review I*, 2007.

[10] E. Parzen, "On Estimation of a Probability Density Function and Mode", The Annals of Mathematical Statistics, 33 (3), 1962.

[11] K.B. Petersen, M.S. Pedersen, "The Matrix Cookbook", http://matrixcookbook.com, 2012.

[12] R. Jenssen, K. E. Hild II, D. Erdogmus, J.C. Principe, T. Eltoft, "Clustering using Renyi's Entropy", Neural Networks Joint Conf., 2003

[13] A. A. Hagberg, D. A. Schult, P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX", Python in Science Conference (SciPy2008), https://networkx.github.io.

[14] S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi, "Optimization by Simulated Annealing", Science 220 (4598), pp. 671-680, 1983.

[15] Y. C. Zhao, L. Tao, K. Thulasiraman and M. N. S. Swamy, "An efficient simulated annealing algorithm for graph bisectioning," 1991 Symposium on Applied Computing.

[16] Timo Poranen, "A simulated annealing algorithm for determining the thickness of a graph", Information Sciences 172 (1-2), 2005.

[17] A. Köse, B.A, Sönmez, M. Balaban, "Simulated Annealing Algorithm for Graph Coloring", arXiv:1712.00709, 2017.

[18] A. Narayanan, E. Shi and B. I. P. Rubinstein, "Link prediction by de-anonymization: How We Won the Kaggle Social Network Challenge," The 2011 International Joint Conference on Neural Networks.

[19] M. Girvan, M.E.J. Newman, "Community structure in social and biological networks", Proc. of National Academy of Sciences of the United States of America, 99(12), 2002.

[20] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python", JMLR 12, pp. 2825-2830, 2011.

[21] M. Bastian, S. Heymann, M. Jacomy, "Gephi: an open source software for exploring and manipulating networks", *International AAAI Conference on Weblogs and Social Media*, 2009.