Tree Sampling Divergence: An Information-Theoretic Metric for Hierarchical Graph Clustering

Bertrand Charpentier, Thomas Bonald

Tree Sampling Divergence: An Information-Theoretic Metric for Hierarchical Graph Clustering

Bertrand Charpentier[1]* and **Thomas Bonald**[2]

[1]Technical University of Munich
[2]Telecom ParisTech
charpent@in.tum.de, thomas.bonald@telecom-paristech.fr

Abstract

We introduce the tree sampling divergence (TSD), an information-theoretic metric for assessing the quality of the hierarchical clustering of a graph. Any hierarchical clustering of a graph can be represented as a tree whose nodes correspond to clusters of the graph. The TSD is the Kullback-Leibler divergence between two probability distributions over the nodes of this tree: those induced respectively by sampling at random *edges* and *node pairs* of the graph. A fundamental property of the proposed metric is that it is interpretable in terms of graph reconstruction. Specifically, it quantifies the ability to reconstruct the graph from the tree in terms of information loss. In particular, the TSD is maximum when perfect reconstruction is feasible, i.e., when the graph has a complete hierarchical structure. Another key property of TSD is that it applies to *any* tree, not necessarily binary. In particular, the TSD can be used to compress a binary tree while minimizing the information loss in terms of graph reconstruction, so as to get a compact representation of the hierarchical structure of a graph. We illustrate the behavior of TSD compared to existing metrics on experiments based on both synthetic and real datasets.

1 Introduction

Many datasets have a graph structure. Examples include infrastructure networks, communication networks, social networks, databases and co-occurence networks, to quote a few. These graphs often exhibit a complex, multi-scale structure where each node belong to many groups of nodes, so-called clusters, of different sizes [Caldarelli, 2007].

Hierarchical graph clustering is a common technique for the analysis of the multi-scale structure of large graphs. Rather than looking for a simple partition of the set of nodes, as in usual clustering techniques, the objective is to represent the graph by some rooted binary tree, whose leaves are the nodes of the graph. This tree can then be used to find relevant clusterings at different resolutions by suitable cuts at different levels of the tree.

While many hierarchical graph clustering algorithms have recently been proposed, see for instance [Newman, 2004; Pons and Latapy, 2005; Sales-Pardo *et al.*, 2007; Clauset *et al.*, 2008; Lancichinetti *et al.*, 2009; Huang *et al.*, 2010; Chang *et al.*, 2011; Tremblay and Borgnat, 2014; Bateni *et al.*, 2017; Bonald *et al.*, 2018], it proves very difficult to evaluate their performance due to the absence of public datasets of graphs with ground-truth hierarchy. A natural approach is then to define some quality metric based on the graph itself, just like modularity is a popular metric for assessing the quality of a simple clustering [Newman and Girvan, 2004].

A cost function has recently been proposed by Dasgupta [Dasgupta, 2016] to assess the quality of a hierachical clustering and has been further analysed and extended in [Roy and Pokutta, 2016; Cohen-Addad *et al.*, 2017; Cohen-Addad *et al.*, 2018]; it can be viewed as the expected size of the smallest cluster encompassing two nodes sampled at random from the edges of the graph.

In this paper, we assess the quality of a hierarchical clustering through the Kullback-Leibler divergence between two probability distributions over the nodes of the tree: those induced respectively by sampling clusters encompassing random *edges* and *node pairs* of the graph. We refer to this metric as *Tree Sampling Divergence* (TSD). If the tree captures the hierarchical structure of the graph, the TSD tends to be large since most edges are sampled from nodes belonging to small clusters of the graph (i.e., deep nodes of the tree), whereas most node pairs, sampled independently, belong to large clusters of the graph (i.e., shallow nodes of the tree, close to the root).

A fundamental property of the proposed metric is that it is interpretable in terms of graph reconstruction. Specifically, it quantifies the ability to reconstruct the graph from the tree. In particular, the TSD is maximum when perfect reconstruction is feasible, i.e., when the graph has a full hierarchical structure and can be reconstructed exactly from the corresponding tree. While a similar property has been proved in [Cohen-Addad *et al.*, 2018] for Dasgupta's cost, this applies only to a graph that has a full hierarchical structure, when perfect reconstruction is possible; there is no interpretation in terms of graph reconstruction for an arbitrary graph. The TSD quantifies the gap to perfect reconstruction for *any* graph: this gap is

*Contact Author.

exactly the information lost in the reconstruction of the graph.

Another key property of TSD is that it applies to *any* tree, not necessarily binary. In particular, the TSD applies to trees of height 2, corresponding to the case of usual clustering (not hierarchical) where the output is a partition of the set of nodes. The TSD can thus be viewed as a *universal* metric, applicable to any type of clustering. Moreover, the TSD can be used in practice to compress a binary tree while minimizing the information loss in terms of graph reconstruction, so as to get a compact representation of the hierarchical structure of a graph.

The paper is structured as follows. We first introduce in sections 2 and 3 the notions of graph sampling and graph distance, which play a key role in our approach. The Tree Sampling Divergence (TSD) is presented in section 4. In section 5, we show how to compute the TSD in practice through graph aggregation. The properties of the TSD in terms of graph reconstruction are presented in section 6. The case of general trees is considered in section 7, the practically interesting case of flat clustering (trees of height 2) being presented in section 8. Some extensions of the TSD are described in section 9. The behavior of the TSD is illustrated by experiments on both synthetic and real data in section 10. Section 11 concludes the paper.

2 Graph Sampling

Consider a weighted, undirected, connected graph $G = (V, E)$ of n nodes and m edges. Let $w(u, v)$ be equal to the weight of edge u, v, if any, and to 0 otherwise. We refer to the weight of node u as:

$$w(u) = \sum_{v \in V} w(u, v).$$

We denote by w the total weight of nodes:

$$w = \sum_{u \in V} w(u) = \sum_{u,v \in V} w(u, v).$$

These weights induce two ways to sample the graph.

Edge sampling. Sampling *edges* at random in proportion to their weights yields the following joint probability distribution on V:

$$\forall u, v \in V, \quad P(u, v) = \frac{w(u, v)}{w}.$$

This distribution is *symmetric* in the sense that:

$$\forall u, v \in V, \quad P(u, v) = P(v, u).$$

Observe that the graph G is fully characterized by its edge sampling distribution P and its total weight w, since the weight of edge u, v is given by $w(u, v) = wP(u, v)$.

Node sampling. Sampling *nodes* at random in proportion to their weights yields the following probability distribution on V:

$$\forall u \in V, \quad P(u) = \frac{w(u)}{w}.$$

This is the marginal distribution of the edge sampling distribution:

$$\forall u \in V, \quad P(u) = \sum_{v \in V} P(u, v)$$

Observe that the graph structure cannot be recovered from the node sampling distribution. The best reconstruction given this distribution and the total weight w consists in assigning the weight $wP(u)P(v)$ to edge u, v. Following Newman [Newman and Girvan, 2004], we refer to this graph as the *null model* associated to graph G.

3 Graph Distance

We have seen that any graph is characterized by its edge sampling distribution and its total weight. Thus the edge sampling distribution can be used to define a distance between two graphs, up to some multiplicative constant on the edge weights. Specifically, let G_1, G_2 be two graphs over the same set of nodes V and denote by P_1, P_2 the corresponding edge sampling distributions. We define the distance between G_1 and G_2 by the Kullback-Leibler divergence between P_1 and P_2, that is:

$$D(G_1, G_2) = D(P_1||P_2) = \sum_{u,v \in V} P_1(u, v) \log \frac{P_1(u, v)}{P_2(u, v)}.$$

Observe that, like the Kullback-Leibler divergence, this distance is *not* symmetric. The graphs G_1 and G_2 are equivalent for this distance, in the sense that $D(G_1, G_2) = 0$, if and only if $P_1 = P_2$, i.e., edge weights are equal up to a multiplicative constant.

The above distance can be used to quantify the difference between the graph G and its null model. This distance is:

$$I = \sum_{u,v \in V} P(u, v) \log \frac{P(u, v)}{P(u)P(v)}, \tag{1}$$

the mutual information between nodes when sampled from the edges. This is a simple yet meaningful metric to quantify the clustering structure of the graph: if nodes are organized in clusters, the mutual information between two nodes u, v sampled from an edge is expected to be large (given node u, you get information on the other node v); on the other hand, if the graph has no clustering structure (e.g., a complete graph), then the mutual information between two nodes u, v sampled from an edge is small (given node u, you have little information on the other node v; this information is null for a complete graph).

We shall see in section 6 that the mutual information I is directly related to our metric: this is the maximum value of the TSD, which is attained by a tree if and only if perfect graph reconstruction from this tree is feasible.

4 Tree Sampling Divergence

Consider some hierarchical clustering of a graph, represented as a rooted binary tree whose leaves are the nodes of the graph. For any $u, v \in V$, we denote by $u \wedge v$ the closest common ancestor of leaves u and v in the tree.

Let T be the set of nodes of the tree. Note that $V \subset T$ and $|T| = 2n - 1$. The graph sampling distributions introduced in section 2 induce the following probability distributions on T:

$$\forall x \in T, \quad p(x) = \sum_{u,v : u \wedge v = x} P(u, v),$$

and

$$\forall x \in T, \quad q(x) = \sum_{u,v:u\wedge v=x} P(u)P(v).$$

The former is that induced by edge sampling while the latter is that induced by independent node sampling. Equivalently, these are the distributions induced by edge sampling for the original graph and the null model, respectively. We expect these distributions to differ significantly if the tree indeed represents the hierarchical structure of the graph. Specifically, we expect p to be mostly concentrated on deep nodes of the tree (far from the root), as two nodes u, v connected with high weight $w(u, v)$ in the graph typically belong to a small cluster, representative of the clustering structure of the graph; on the contrary, we expect q to be concentrated over shallow nodes (close to the root) as two nodes u, v sampled independently at random typically belong to large clusters, less representative of the clustering structure of the graph.

This motivates the following metric for the quality of the tree T as hierarchical structure of the graph:

$$Q = \sum_{x\in T} p(x) \log \frac{p(x)}{q(x)}. \tag{2}$$

This is the Kullback-Leibler divergence[1] between the probability distributions p and q. The larger the divergence, the better the tree for representing the graph. In the following, we refer to this metric as the *Tree Sampling Divergence* (TSD). We shall see in section 6 that the quantity $I - Q$, with I given by (1), can be interpreted in terms of information loss in the problem of graph reconstruction from the tree.

5 Graph Aggregation

The sampling distributions p, q can be simply computed by graph aggregation (see Algorithm 1). The algorithm consists in iteratively merging two nodes u, v of the graph that are leaves of the tree into a single node $x = u \wedge v$, their common parent in the tree. This node has weight:

$$w(x) = w(u) + w(v),$$

and a self-loop with weight:

$$w(x, x) = w(u, u) + w(v, v) + 2w(u, v)$$

in the aggregate graph. We get the sampling probabilities of node $x \in T$ as:

$$p(x) = \frac{w(x, x)}{w}, \quad q(x) = \left(\frac{w(x)}{w}\right)^2,$$

and the TSD follows from (2).

6 Graph Reconstruction

We consider the problem of graph reconstruction from the tree. Specifically, given the tree T and the weights of nodes, we would like to build some graph $\hat{G}$ that is as close as possible to G. The hierarchical clustering can then be viwed as the latent representation of the graph G in the auto-encoding process:

$$G \to T \to \hat{G}.$$

We first need to define the decoding scheme, that is the process to build some graph $\hat{G}$ from the tree T. To do this, we assign some weight $\sigma(x)$ to each node $x \in T \setminus V$. This corresponds to the weight of cluster x and will be determined so as to minimize the distance between the original graph G and its reconstruction $\hat{G}$. We let the weight of edge u, v in graph $\hat{G}$ equal to:

$$\hat{w}(u, v) = w(u)w(v)\sigma(u \wedge v).$$

Thus the weight of edge u, v is proportional to the respective weights of nodes u, v and to the weight of their smallest common cluster $u \wedge v$, as given by the tree.

It remains to find the cluster weights σ that minimize the distance between the original graph G and its reconstruction $\hat{G}$. For this, we use the graph distance defined in section 3, corresponding to the Kullback-Leibler divergence between the corresponding edge sampling distributions, P and $\hat{P}$. Observe that:

$$\hat{P}(u, v) = CP(u)P(v)\sigma(u \wedge v),$$

with:

$$C = \left(\sum_{u,v\in V} P(u)P(v)\sigma(u \wedge v)\right)^{-1}.$$

Algorithm 1: Computation of TSD (binary trees)

Input: Graph $G = (V, E)$ with sampling distribution P; Tree T
Output: Tree sampling distributions p, q

for $x \in V$ **do**
  $p(x) \leftarrow P(x, x)$; $q(x) \leftarrow P(x)^2$
while $|V| > 1$ **do**
  $u, v \leftarrow$ leaves of the tree T, with common parent x
  `// aggregate nodes` u, v `into` x
  $V \leftarrow V \setminus \{u, v\}$; $V \leftarrow V \cup \{x\}$
  `// update` P
  for y neighbor of u or v **do**
    if $y \neq x$ **then**
      $P(x, y) \leftarrow P(u, y) + P(v, y)$;
      $P(y, x) \leftarrow P(x, y)$
    else
      $P(x, x) \leftarrow P(u, u) + P(v, v) + 2P(u, v)$
    $P(x) \leftarrow P(u) + P(v)$
  `// update` p, q
  $p(x) \leftarrow P(x, x)$; $q(x) \leftarrow P(x)^2$
  `// update` T
  $T \leftarrow T \setminus \{u, v\}$

[1] Note that Q is finite since the support of q is T (because $P(u) > 0$ for each $u \in V$).

We obtain:

$$\begin{aligned} D(G,\hat{G}) &= D(P||\hat{P}), \\ &= \sum_{u,v\in V} P(u,v)\log\frac{P(u,v)}{\hat{P}(u,v)}, \\ &= \sum_{u,v\in V} P(u,v)\log\frac{P(u,v)}{P(u)P(v)} \\ &\quad - \sum_{u,v\in V} P(u,v)\log\sigma(u\wedge v) \\ &\quad + \log\left(\sum_{u,v\in V} P(u)P(v)\sigma(u\wedge v)\right), \\ &= I - J, \end{aligned}$$

where I , given by (1), is the mutual information between nodes when sampled from the edges and

$$\begin{aligned} J &= \sum_{u,v\in V} P(u,v)\log\sigma(u\wedge v) \\ &\quad - \log\left(\sum_{u,v\in V} P(u)P(v)\sigma(u\wedge v)\right), \\ &= \sum_{x\in T} p(x)\log\sigma(x) - \log\left(\sum_{x\in T} q(x)\sigma(x)\right). \end{aligned}$$

Observe that I is independent of $\hat{P}$ so that J is the quantity to be maximized. Finding where the derivative of J with respect to $\sigma(x)$ is equal to 0 yields:

$$\frac{p(x)}{\sigma(x)} = \frac{q(x)}{\alpha} \quad \text{with} \quad \alpha = \sum_{x\in T} q(x)\sigma(x).$$

We get:

$$\forall x \in T, \quad \sigma(x) = \alpha\frac{p(x)}{q(x)}.$$

For this optimal value of σ, we obtain:

$$J = \sum_{x\in T} p(x)\log\frac{p(x)}{q(x)},$$

which is equal to Q, the tree sampling divergence.

Finally, the optimal reconstruction of the graph corresponds to the sampling distribution:

$$P^\star(u,v) \propto P(u)P(v)\frac{p(u\wedge v)}{q(u\wedge v)}$$

and the distance between the original graph G and its optimal reconstruction $G^\star$ is:

$$D(G,G^\star) = D(P||P^\star) = I - Q.$$

We deduce that the information loss in the optimal auto-encoding process $G \to T \to G^\star$ is $I - Q$. In particular, $Q \leq I$, with equality if and only if perfect reconstruction from the tree T is possible.

7 Tree Compression

While existing quality metrics are typically defined for binary trees (see for instance Dasgupta's cost and extensions in [Cohen-Addad *et al.*, 2017]), the TSD equally applies to general trees. The definition is exactly the same. As for the computation, Algorithm 1 applies except that the aggregation must be applied simultaneously to *all* leaves of the tree having the same parent (see Algorithm 2).

Algorithm 2: Computation of TSD (general trees)

Input: Graph $G = (V, E)$ with sampling distribution P; Tree T
Output: Tree sampling distributions p, q

for $x \in V$ **do**
  $p(x) \leftarrow P(x,x)$; $q(x) \leftarrow P(x)^2$
while $|V| > 1$ **do**
  $S \leftarrow$ leaves of the tree T, with common parent x
  `// aggregate nodes S into x`
  $V \leftarrow V \setminus S$; $V \leftarrow V \cup \{x\}$
  `// update P`
  for y neighbor of a node in S **do**
    if $y \neq x$ **then**
      $P(x,y) \leftarrow \sum_{u\in S} P(u,y)$;
      $P(y,x) \leftarrow P(x,y)$
    else
      $P(x,x) \leftarrow \sum_{u,v\in S} P(u,v)$
    $P(x) \leftarrow \sum_{u\in S} P(u)$
  `// update p, q`
  $p(x) \leftarrow P(x,x)$; $q(x) \leftarrow P(x)^2$
  `// update T`
  $T \leftarrow T \setminus S$

This is a very interesting property as dense trees are sought in practice, the information provided by a binary tree being to rich to be directly exploited by data scientists or engineers. Moreover, it provides a way to compress a binary tree in an optimal way, by minimizing the information loss caused by tree compression. In view of (2), the information loss caused by the merge of cluster $x \in T$ (internal node of the tree) with its parent y is:

$$\begin{aligned} \Delta(x,y) &= (p(x)+p(y))\log\frac{p(x)+p(y)}{q(x)+q(y)} \\ &\quad - p(x)\log\frac{p(x)}{q(x)} - p(y)\log\frac{p(y)}{q(y)}. \end{aligned}$$

Observe that the merge of x and y does not modify the edge sampling distributions on the other nodes of the tree. In particular, the information loss $\Delta(x,y)$ can be computed for all branches of the tree, with a complexity in $O(n)$ given p and q, and successive merges be done in increasing order of these information losses. The stopping criterion may be related to some target number of internal nodes in the tree, or some maximum total information loss.

8 Graph Clustering

Since the TSD applies to any tree, it applies in particular to trees of height 2, corresponding to usual graph clustering (i.e., a partition of the nodes).

Let $C_1, \dots, C_K$ be K clusters, forming a partition of V. This can be represented as a tree of height 2, with k internal nodes $x_1, \dots, x_K$ at level 1 (the clusters) and n leaves at level 2 (the nodes of the graph). For each cluster $k = 1, \dots, K$, we have:

$$p(x_k) = \sum_{u,v \in C_k} P(u,v) = \frac{w_k}{w},$$

where

$$w_k = \sum_{u,v \in C_k} w(u,v)$$

denotes the internal weight of cluster k (total weight of edges within the cluster). Similarly,

$$q(x_k) = \left(\sum_{u \in C_k} P(u) \right)^2 = \left(\frac{W_k}{w} \right)^2,$$

where

$$W_k = \sum_{u \in C_k} w(u)$$

denotes the weight of cluster k (total weight of nodes in the cluster). Assuming that there are no self loops, we get for the root, which we denote by 0,

$$p(0) = 1 - \sum_{k=1}^{K} p(x_k), \quad q(0) = 1 - \sum_{k=1}^{K} q(x_k).$$

Thus the TSD takes the form:

$$Q = \sum_{k=1}^{K} \frac{w_k}{w} \log \frac{w w_k}{W_k^2} + \left(1 - \sum_{k=1}^{K} \frac{w_k}{w} \right) \log \left(\frac{1 - \sum_{k=1}^{K} \frac{w_k}{w}}{1 - \sum_{k=1}^{K} \left(\frac{W_k}{w} \right)^2} \right).$$

This is a novel metric for evaluating the quality of a clustering, different from modularity [Newman and Girvan, 2004], the usual metric given by:

$$M = \sum_{k=1}^{K} p(x_k) - \sum_{k=1}^{K} q(x_k) = \sum_{k=1}^{K} \frac{w_k}{w} - \sum_{k=1}^{K} \left(\frac{W_k}{w} \right)^2.$$

The key property of the TSD is that it quantifies the ability to reconstruct the graph in terms of information loss; this applies to any tree and thus to usual graph clustering as well. There is no such interpretation with modularity.

9 Extensions

The TSD can be extended in several ways. First, the node pair sampling distribution q can be replaced by the uniform distribution (i.e., the null model is the complete graph). This boils down to the initialization $P(x) \leftarrow \frac{1}{n}$ for all $x \in V$ before applying Algorithm 1 or 2. Second, the Kullback-Leibler divergence can be replaced by some other divergence, like the square Euclidean distance between the probability measures, $\sum_{x \in T} (p(x) - q(x))^2$. The experiments presented below suggest however that it is preferable to use the Kullback-Leibler divergence.

10 Experiments

The Python code and the datasets used for the experiments are available online[2].

Graph Representation

We first show how TSD is able to detect the quality of a tree in terms of graph representation. For this, we generate two noisy versions of the same graph, say G_1 and G_2, and return the corresponding trees, say T_1 and T_2, by the hierarchical clustering algorithm described in [Bonald *et al.*, 2018]. We then assess the ability of TSD to identify the best tree associated to each graph (e.g., T_1 should be better than T_2 for graph G_1). The classification score (in proportion of correct answers) is given in Table 1 for two metrics, TSD and Dasgupta's cost. Each classification score is based on 1000 samples of the graphs G_1 and G_2; the original graph G, generated at random from a random hierarchy, has 100 nodes and average degree 10; the two graphs G_1 and G_2 are derived from G by replacing some random subset of the edges at random (the higher the noise, the easier the classification task).

Noise	2%	5%	10%
TSD	**64%**	**82%**	**94%**
Dasgupta's cost	59%	72%	86%

Table 1: Classification scores.

Graph Reconstruction

Next, we show how TSD captures the quality of a tree in terms of graph reconstruction. To this end, we consider a hierarchical stochastic block model (HSBM) of 80 nodes and 3 levels of hierarchy (binary tree with clusters of 10 nodes on leaves) with different parameters, see [Lyzinski *et al.*, 2017]. For each graph, we apply various graph clustering algorithms, so as to get hierarchies of different qualities. The reconstruction procedure is that described in section 6.

Figure 1 shows the quality of the reconstruction for usual scores (detailed below) with respect to the quality / cost of the hierarchy, for three metrics: TSD, Dasgupta's cost, and the version of the TSD based on the Euclidean distance, we refer to as the Euclidean divergence (see section 9). The results suggest that TSD is the best metric in terms of graph reconstruction, with a clear correlation between the metric and the reconstruction scores. The correlation is also present but more noisy for Dasgupta's cost; there is no correlation with the Euclidean divergence.

The quality of the reconstruction is based on the following classical scores [Bordes *et al.*, 2013; Nickel and Kiela, 2017; Nickel *et al.*, 2016; Bojchevski and Günnemann, 2018]. First, keeping all edges of the reconstructed graph whose weight is larger than some threshold provides a set of graphs with different false positive and false negative rates, from which we get the Area Under ROC Curve (AUC) and the Average Precision Score (APS). Second, we compute the rank of each edge of the original graph among all possible edges ending in one of the two nodes in the reconstructed graph, in decreasing order of weights. This rank score is averaged over all nodes

[2] See https://github.com/sharpenb/Tree-Sampling-Divergence.

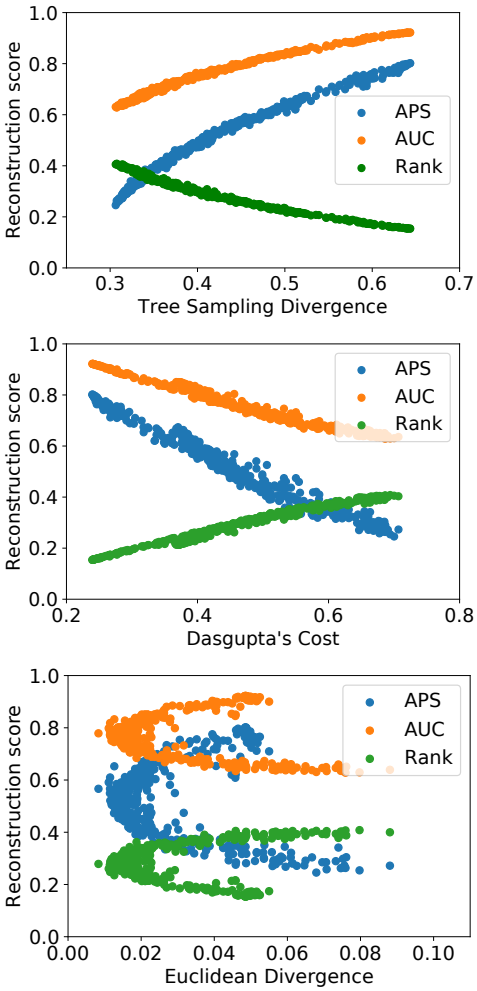


Figure 1: Reconstruction scores against quality / cost metrics.

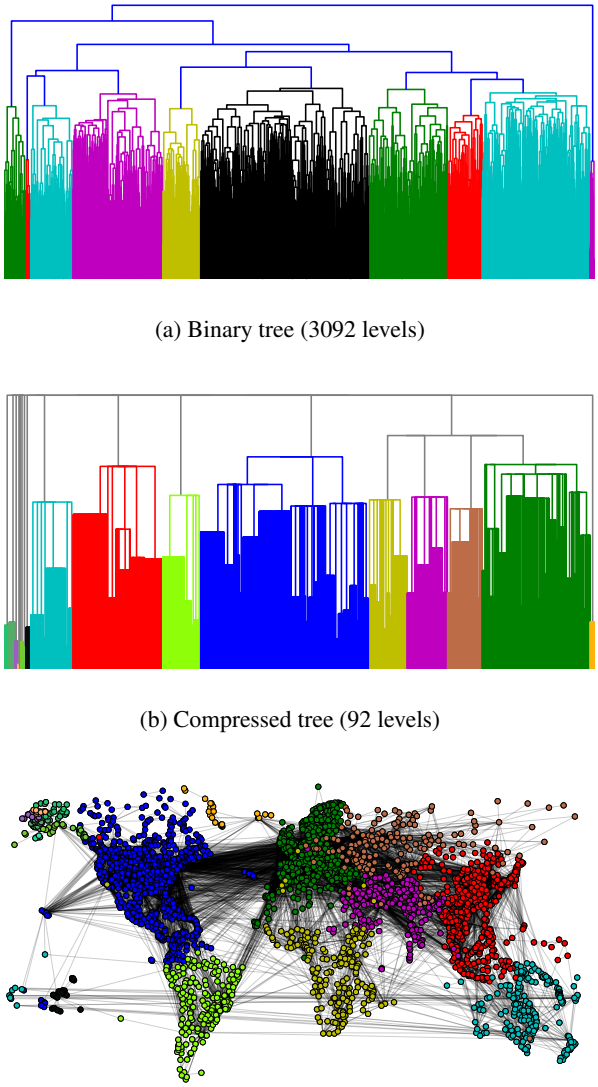
and normalized by the number of nodes to get a score between 0 and 1; the lower the rank, the better the reconstruction.

Tree Compression

Next, we show the pratical interest of TSD in terms of tree compression. We consider OpenFlights[3], a weighted graph of 3,092 nodes and 18,193 edges representing flights between the main airports of the world. We first apply the hierarchical clustering algorithm described in [Bonald *et al.*, 2018] then apply the algorithm described at the end of section 7 to compress the resulting tree with minimum information loss. Specifically, we merge $3,000$ levels among the $3,092$ levels of the original binary tree.

The results are shown in Figure 2, together with a clustering derived from this compressed tree. We observe that the compressed tree still captures the hierarchical structure of the graph, despite the very high compression rate applied here (only 3% of the hierarchical levels are retained). The compressed tree is clearly much easier to exploit in practice than the original tree.

[3]https://openflights.org



(a) Binary tree (3092 levels)

(b) Compressed tree (92 levels)

(c) Clustering extracted from the compressed tree (20 clusters)

Figure 2: Tree compression on the Openflights graph.

11 Conclusion

We have presented a novel quality metric for hierarchical graph clustering. This metric, inspired by information theory, reflects the ability to reconstruct the initial graph from a hierarchy representing this graph. It is applicable to general trees (not only binary trees) and can be used for tree compression.

References

[Bateni *et al.*, 2017] Mohammadhossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab Mirrokni. Affinity clustering: Hierarchical clustering at scale. In *Advances in Neural Information Processing Systems*, 2017.

[Bojchevski and Günnemann, 2018] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

[Bonald *et al.*, 2018] Thomas Bonald, Bertrand Charpentier, Alexis Galland, and Alexandre Hollocou. Hierarchical graph clustering based on node pair sampling. In *Proceedings of the 14th International Workshop on Mining and Learning with Graphs (MLG)*, 2018.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.

[Caldarelli, 2007] Guido Caldarelli. *Large scale structure and dynamics of complex networks: from information technology to finance and natural science*, volume 2. World Scientific, 2007.

[Chang *et al.*, 2011] Cheng-Shang Chang, Chin-Yi Hsu, Jay Cheng, and Duan-Shin Lee. A general probabilistic framework for detecting community structure in networks. In *Proceedings IEEE INFOCOM*, 2011.

[Clauset *et al.*, 2008] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 2008.

[Cohen-Addad *et al.*, 2017] Vincent Cohen-Addad, Varun Kanade, and Frederik Mallmann-Trenn. Hierarchical clustering beyond the worst-case. In *Advances in Neural Information Processing Systems*, 2017.

[Cohen-Addad *et al.*, 2018] Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 2018.

[Dasgupta, 2016] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of ACM symposium on Theory of Computing*, 2016.

[Huang *et al.*, 2010] Jianbin Huang, Heli Sun, Jiawei Han, Hongbo Deng, Yizhou Sun, and Yaguang Liu. Shrink: A structural clustering algorithm for detecting hierarchical communities in networks. In *Proceedings of ACM International Conference on Information and Knowledge Management*, 2010.

[Lancichinetti *et al.*, 2009] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3), 2009.

[Lyzinski *et al.*, 2017] Vince Lyzinski, Minh Tang, Avanti Athreya, Youngser Park, and Carey E Priebe. Community detection and classification in hierarchical stochastic blockmodels. *IEEE Transactions on Network Science and Engineering*, 4(1), 2017.

[Newman and Girvan, 2004] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 2004.

[Newman, 2004] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[Nickel and Kiela, 2017] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6338–6347. Curran Associates, Inc., 2017.

[Nickel *et al.*, 2016] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. Holographic embeddings of knowledge graphs. In *AAAI*, pages 1955–1961, 2016.

[Pons and Latapy, 2005] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*. Springer, 2005.

[Roy and Pokutta, 2016] Aurko Roy and Sebastian Pokutta. Hierarchical clustering via spreading metrics. In *Advances in Neural Information Processing Systems*, 2016.

[Sales-Pardo *et al.*, 2007] Marta Sales-Pardo, Roger Guimera, André A Moreira, and Luís A Nunes Amaral. Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences*, 104(39), 2007.

[Tremblay and Borgnat, 2014] Nicolas Tremblay and Pierre Borgnat. Graph wavelets for multiscale community mining. *IEEE Transactions on Signal Processing*, 62(20), 2014.