HAL
open science

Clustering graphs using random trees

Kevin Dalleau, Miguel Couceiro, Malika Smaïl-Tabbone

To cite this version:

Kevin Dalleau, Miguel Couceiro, Malika Smaïl-Tabbone. Clustering graphs using random trees. 2019. hal-02282207

HAL Id: hal-02282207

https://inria.hal.science/hal-02282207v1

Preprint submitted on 9 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Clustering graphs using random trees

Kevin Dalleau[1], Miguel Couceiro[1], and Malika Smail-Tabbone[1]

Universite de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France
{kevin.dalleau,miguel.couceiro, malika.smail}@loria.fr

Abstract. In this work-in-progress paper, we present GraphTrees, a novel method that relies on random decision trees to compute pairwise distances between vertices in a graph. We show that our approach is competitive with the state of the art methods in the case of non-attributed graphs in terms of quality of clustering. By extending the use of an already ubiquitous approach – the random trees – to graphs, our proposed approach opens new research directions, by leveraging decades of research on this topic.

1 Introduction

Identifying community structure in graphs is a challenging task in many applications: computer networks, social networks, etc. In this paper, we propose a decision tree based method to compute pairwise distances between vertices that we call GraphTrees (GT). These dissimilarities between vertices in a graph can be used for clustering, but also for other purposes such as filtering or information retrieval.

While many graph clustering algorithms have been proposed, all of them fall into the three following categories:

- Top-down approaches, where the set of vertices is split into partitions in a iterative fashion,
- Bottom-up approaches, where each vertex is first assigned to its own cluster and then merged with other vertices according to a specific measure,
- Local optimization approaches, where a random clustering is first performed and vertices are then migrated.

A thorough review of the methods already available has been performed by Schaeffer *et al.* in [10]. Other interesting approaches have been proposed since, such as the Louvain clustering approach [1], based on modularity optimisation.

In this paper, we present a novel tree-based method enabling the computation of vertex-vertex similarities in graphs[1] and that we call *GraphTrees* (GT).

[1] To avoid terminology-related issues, we will exclusively use the terms vertex for graphs and node for random trees throughout the paper.

Using random trees on graphs leverages the use of decades of research on the subject, opening a wide array of possible improvements. Moreover, using different types of trees, the approach could be extended to vertex-attributed graph clustering.

The paper is organized as follows. In Section 2, we present our method. We then discuss its performance in Section 3 through an empirical study on real and synthetic datasets. In the last section of the paper, we present a brief discussion of our results and state some perspectives for future research.

2 Method

In this section, we begin by describing our proposed method. We then compare two different approaches to compute a (dis)similarity matrix. The first one is inspired by the approach of unsupervised extra-trees (UET) [3], while the second one uses the mass-based distance proposed by [11].

The intuition behind GT is to leverage a recursive partition of the vertices to compute (dis)similarities. This partition is performed using random trees. This approach has already been studied in the case of data represented as feature vectors [3],[11] but not to graphs.

UET is a tree-based method extending random trees to compute the similarities between instances. The idea is that all instances ending up in the same leaves are more similar to each other than to other instances. The pairwise similarities $s(i, j)$ are hence obtained by increasing $s(i, j)$ for each leaf where both i and j appear. A normalisation is finally performed when all trees have been constructed, to have values in the interval $[0, 1]$.

Leaves, and, more generally, nodes of the trees can be viewed as partitions of the original space. Enumerating the number of co-occurrences in the leaves is then the same as enumerating the number of co-occurrence of instances in the smallest regions of a specific partitioning.

While the method has been applied to feature vectors, we show in this paper that the same approach can be applied to graphs, with a simple change to the splitting method. In UET, the split is performed by selecting a random cut-point on one attribute, sampled from the attribute set without replacement. When applied to graphs, the split can be performed in a similar fashion by choosing randomly one vertex v_s from the graph. Let $N(v)$ be the set of neighbours of a vertex v in a graph. When a split is done, the resulting left node contains all vertices $v \in N(v_s)$ and v_s, while the right node contains the other vertices.

The algorithm of GT is presented Algorithm 1. Algorithm 2 presents how to obtain a similarity matrix from the forest of random trees. This method of computing a similarity matrix from the trees is the same as the one proposed in [3], the trees being GraphTrees.

Algorithm 1: Graph tree algorithm

```
Data: A graph G(V, E), an uninitialized stack S
Parameter: Minimum node size to split n_min
Result: A list of leaves L
root_node = V;
v_s = a vertex sampled without replacement from V;
V_left = N(v_s) ∪ {v_s} ;   //N(v) returns the set of neighbours of v
V_right = V \ V_left ;
Push V_left and V_right to S ;
leaves = []; //leaves is an empty list
while S is not empty do
    V_node = pop the last element of S;
    if |V_node| < n_min then
        Append V_node to leaves;   //node size in lower than n_min, it is a leaf
          node
    end
    else
        v_s = a vertex sampled without replacement from V_node;
        V_left = (V_node ∩ N(v_s)) ∪ {v_s};
        V_right = V_node \ V_left ;
        Push V_left to S;
        Push V_right to S;
    end
end
return leaves;
```

Algorithm 2: From a graph to a similarity matrix using graph trees

```
Data: Adjacency matrix A
Result: Similarity matrix S
T = graph_tree_ensemble(D, K)   //; S = 0_{n_obs,n_obs}   // Initialization of a
  zero matrix of size n_obs ;
for I_i ∈ I do
    for I_j ∈ I do
        S_{i,j} = number of times the instances I_i and I_j fall in the same leaf node
          in each tree of T = {t_1, t_2, ..., t_M};
    end
end
return S_{i,j} = S_{i,j} / M;
```

This approach has some limitations. First, when a dissimilarity is needed, it is necessary to apply a transformation to the similarity matrix. The choice of transformation can have a great impact on the subsequent clustering. Moreover, this adds up to the time complexity of the method, as a transformation of an $n \times n$ similarity matrix is at least in $O(n^2)$, and can become an issue in large graphs.

In [11], Ting *et al.* presents a probability mass-based dissimilarity measure, obtained using random trees. The key property of their proposed measure is that the dissimilarity between two instances in a dense region is higher than the same interpoint dissimilarity between two instances in a sparse region of the same space. This mass-based dissimilarity between two instances depends on the mass probability of the smallest region that covers these two instances.

Let $H \in \mathcal{H}(D)$ be a hierarchical partitioning of the original space of a dataset D into non-overlapping and non-empty regions, and let $R(x, y|H)$ be the smallest local region covering x and y with respect to H. The mass-based dissimilarity estimated by a finite number t of models is given by the following equation:

$$m_e(x, y|D) = \frac{1}{t} \sum_{i=1}^{t} P(R(x, y|H_i)) \tag{1}$$

where $P(R) = \frac{1}{|D|} \sum_{z \in D} 1(z \in R)$.

Fig. 1 presents an example of hierarchical partition H of a dataset D containing 8 instances. For the sake of the example, let us compute $m_e(1, 4)$ and $m_e(1, 8)$. We have $m_e(1, 4) = \frac{1}{8}(2) = 0.25$, as the smallest region where instances 1 and 4 co-appear contains 2 instances. However, $m_e(1, 8) = \frac{1}{8}(8) = 1$, since instances 1 and 8 only appear in one region of size 8, the original space. These instances are vertices in our case.

Similarly to the first approach to obtain a dissimilarity matrix, mass-based dissimilarity can be applied to graphs. The method is based on two steps: (i) partition the vertices using random trees, (ii) compute pairwise dissimilarities between vertices using 1.

This approach has multiple advantages over the first one. First, as many clustering algorithms require a distance matrix as an input, a similarity matrix requires a transformation to be used. Using the mass-based approach, this transformation is no longer needed, as a dissimilarity matrix is obtained. Secondly, a mass-based pairwise distance between vertices might be more effective to find communities of varying densities.

The computation of pairwise vertex-vertex dissimilarities using Graph Trees and the mass-based dissimilarity we just described has a time complexity of $O(t\Psi log(\Psi) + n^2 t log(\Psi))$ [11], where t is the number of trees, Ψ the maximum height of the trees, and n is the number of vertices. When $\Psi << n$, this time complexity becomes $O(n^2)$.

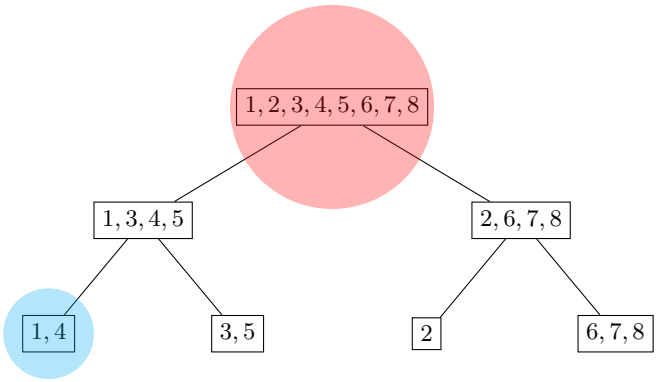

Fig. 1. Example of partitioning of 8 instances in non-overlapping non-empty regions using a random tree structure. The blue and red circles denote the smallest nodes (*i.e*, regions) containing vertices 1 and 4 and vertices 1 and 8, respectively.

In the next section, we evaluate GT on both real-world and synthetic datasets.

3 Evaluation

This section is divided into 3 subsections. In Subsection 3.1 we present some preliminary work, where we assess the different methods to compute dissimilarities between vertices. We then perform an assessment of the discrUsing random trees on graphs leverages the use of decades of research on the subject, opening a wide array of possible improvements. Moreover, using different types of trees, the approach could be extended to vertex-attributed graph clustering.

iminative power of our proposed method 3.2. Finally in 3.3, we present its clustering performance.

3.1 Comparison of GT using UET and graph trees using mass-based dissimilarities

Even though the mass-based approach is more appealing than the UET approach complexity-wise, we empirically assess here how the two differ on various graphs. The graphs used in this subsection are described in Table 1.

Table 1. Graphs used in Subsection 3.1

Graph	# vertices	# edges
Stochastic Block Model 1	115	1226
Stochastic Block Model 2	1005	25571
Email-Eu-Core	2708	5429
Random graph	200	12928

The stochastic bloc models (SBM) graphs are composed of k blocks of a user-defined size, connected by edges depending on a specific probability which is a parameter. In this paper, we carried out an empirical study on two SBM, denoted $SBM1$ and $SBM2$ that differ by the probabilities of intra- and inter-cluster edge creation.
The Email-Eu-Core graph [12] [7] represents relations between members of a research institution, where edges represents communication between those members. The dataset also contains ground-truth community memberships of the vertices. Each individual belongs to exactly one of 42 departments at the research institute.
Finally, the random graph is an Erdos-Renyi random graph [5], where edges are created with a probability $p = 0.7$ in our case. This graph does not exhibit any cluster structure, and is used as a *negative control*.

We applied our proposed mass-based approach and naive approach to compute vertex dissimilarities on these graphs. We then used t-SNE [8] to obtain a 2D projection of these vertices. We finally plotted the distance distribution. All results are presented in Fig. 2, 3, 4 and 5 .

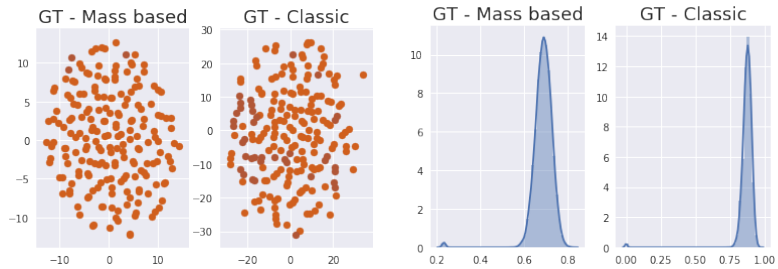

Fig. 2. Random graph: the computed dissimilarities effectively shows an absence of separated clusters. The lowermost part presents the distribution of pairwise distance between vertices.

We observe that in both SBM models, the density-based approach leads to 3 clearly separated clusters, whereas the approach using UET leads to a sparser cluster. The difference between both methods is even greater in SBM2 here the density of edges between different clusters is higher. In that case, we observe that using GT with UET we effectively lose one cluster in the projection. In the case of the Email-Eu-Core graph, the projection shows that the mass-based approach better separates clusters than the naive one, where all clusters tend to gather. Finally, using both approaches on the random graph gives an expected result. Indeed, no clear cluster structure can be observed in the projection using both approaches.

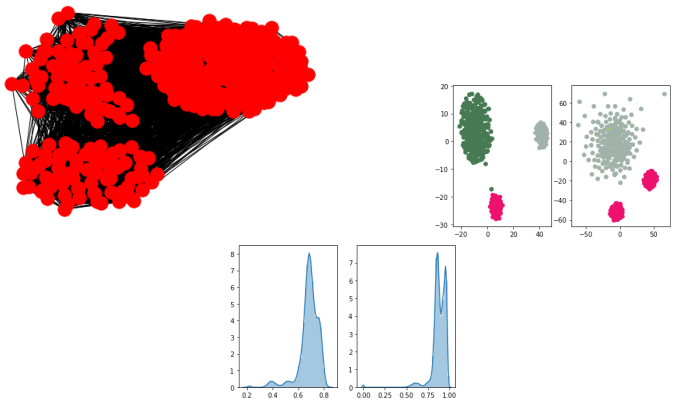

Fig. 3. SBM1 graph: the computed dissimilarities shows that the density-based approach leads to clearly separated clusters, with a dense core cluster, while the naive one tends to a sparser core cluster. Left graphs corresponds to distances using GT-Mass-based, and right graphs to distances using GT-Classic.

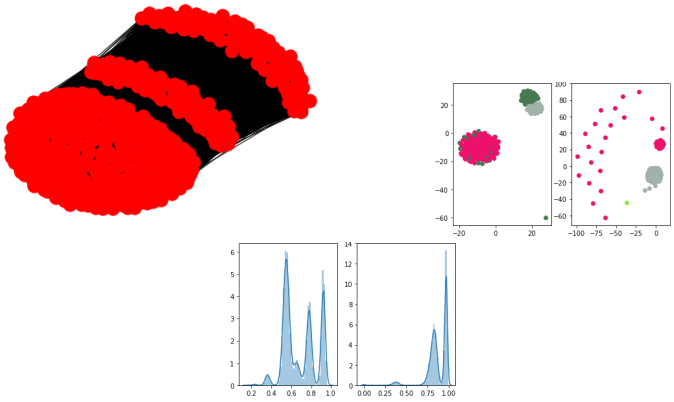

Fig. 4. SBM2 graph: the computed dissimilarities shows that the density-based approach leads to clearly separated clusters, unlike the naive approach. Moreover, the distribution of pairwise distance between vertices is more spread out when the mass-based approach is used. Left graphs corresponds to distances using GT-Mass-based, and right graphs to distances using GT-Classic.

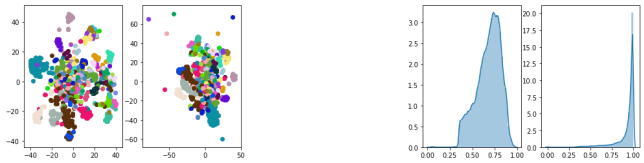

Fig. 5. Email-Eu-Core: the mass-based approach better separates clusters than the naive approach, where all clusters are close. Left graphs corresponds to distances using GT-Mass-based, and right graphs to distances using GT-Classic. The mass-based approach better separates clusters than the naive one, where all clusters tend to gather.

In the following section, we will assess the performance of GT. This evaluation will be performed using the mass-based method, as we observed its versatility in this subsection.

3.2 Assessment of the dissimilarities

All the experiments presented in this subsection are based on the comparison of the mean intracluster dissimilarities and the mean intercluster dissimilarities, as well as their differences, taking values in the interval $[0, 1]$. These metrics enable a comparison that is agnostic to a subsequent clustering method.

The mean difference is computed as follows. First, the arithmetic mean of the pairwise dissimilarities between all samples having the same label is computed, corresponding to the mean intracluster dissimilarity μ_{intra}. Then the same process is done for samples with a different label, giving the mean intercluster dissimilarity μ_{inter}. We finally compute the difference $\Delta = |\mu_{intra} - \mu_{inter}|$. In our experiments, this difference Δ is computed 10 times. In the following section, $\bar{\Delta}$ denotes the mean of differences between runs, and σ its standard deviation.

The datasets used in this subsection are described in Table 2. We compare the values obtained using GT and two other methods, SimRank [6] and ASCOS [2]. Since these approaches compute similarities between vertices, we used the relation $d(v_i, v_j) = 1 - s(v_i, v_j)$ to obtain pairwise dissimilarities.

Table 2. Datasets used for the evaluation of GT without any subsequent clustering

Dataset	# vertices	# edges	# clusters
Karate	34	78	2
Polbooks	105	441	3
Football	115	1226	10
WebKB	877	1480	4

The results are presented Table 3. We also computed the running time for each computation, that are presented in Table 4.

Table 3. Comparison of the mean difference between mean intracluster dissimilarities and mean intercluster dissimilarities on benchmark graph datasets. Best results are in boldface.

Dataset	GT - $\bar{\Delta}$ (σ)	SimRank - $\bar{\Delta}$ (σ)	ASCOS - $\bar{\Delta}$ (σ)
Karate	0.131 (0.008)	**0.112 (0.000)**	0.113 (0.000)
PolBooks	0.062 (0.002)	0.080 (0.000)	**0.053 (0.000)**
Football	0.215 (0.002)	0.139 (0.000)	**0.054 (0.000)**
WebKB	0.050 (0.002)	0.106 (0.000)	**0.023 (0.000)**

Table 4. Comparison of the mean durations on benchmark graph datasets. Best results are in boldface.

Dataset	GT - Time (s) (σ)	SimRank - Time (s) (σ)	ASCOS - Time (s) (σ)
Karate	0.130 (0.0163)	0.296 (0.01)	**0.094 (0.002)**
PolBooks	**0.454 (0.001)**	6.876 (0.105)	1.238 (0.030)
Football	**0.477 (0.012)**	11.538 (0.151)	1.663 (0.003)
WebKB	**45.798 (1.417)**	139.590 (1.922)	53.439 (2.036)

ASCOS outperform the other distances in terms of $\bar{\Delta}$, as it is lower in most cases. However, values of $\bar{\Delta}$ are similar and remain small for all methods and for all datasets but Football. In terms of running time, GT outperforms both methods, except in the case of the Karate graph. This is explained by the fact that this graph is the smallest, with 34 vertices. We notice than for more complex graph however, GT is competitive.

It is also interesting to compare the distribution of distance for each method according to the choice of algorithm (GT, SimRank or ASCOS). Fig. 6 and 7 display these distribution for the Football and WebKB graphs (respectively). When GT is used, the variance of the distribution is higher than its counterparts, and multiple peaks can be observed. This is also observed for all other datasets in this empirical evaluation, but not presented here.

3.3 Evaluation using a clustering algorithms

Using the NMI We then evaluated our approach on simple graphs against two other methods, in order to assess if our proposed method is competitive in such graphs. This evaluation is performed on both synthetic and real-world graphs. We compare the NMI obtained using GT with the NMI obtained using two state-of-the-art clustering method on simple graphs, MCL [4] and Louvain [1]. NMI is a clustering quality metric. Its values lie in the range $[0, 1]$, with a value of 1 being a perfect assignment.

To perform this evaluation, we used commonly used benchmark datasets, presented Table 5. We first computed the distance matrices using GT, with a total number of trees $n_{trees} = 200$. We then applied k-means on the points obtained

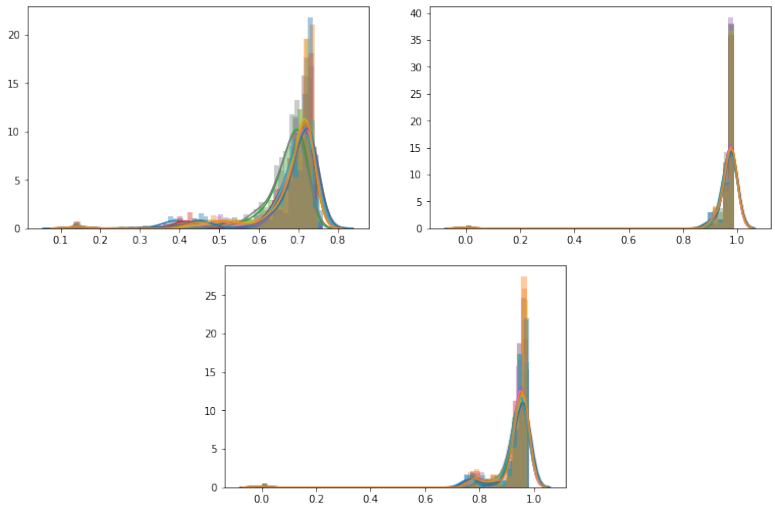

Fig. 6. Distribution of distances in Footbal obtained using GT (upper left), SimRank (upper right) and ASCOS (bottom). The different colours correspond to the data from different clusters, according to the provided ground truth.

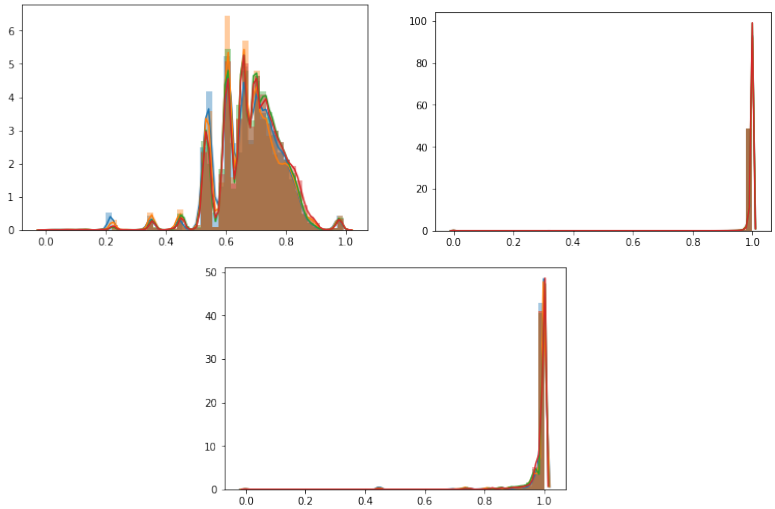

Fig. 7. Distribution of distances in WebKB obtained using GT (upper left), SimRank (upper right) and ASCOS (bottom). The different colours correspond to the data from different clusters, according to the provided ground truth.

using t-SNE on the distance matrix. We repeated this procedure 20 times and computed means and standard deviations of the NMI.

During our experiments, we found out that preprocessing the distances prior to the clustering phase may lead to better results, in particular with *Scikit learn*'s [9] *QuantileTransformer*. This transformation tends to spread out the most frequent values and to reduce the impact of outliers.

Fig. 8 shows the distribution of vertex-vertex distance for PolBooks with and without prior preprocessing using this approach. We observe that the distribution of the pairwise distance is spread out, with visible peaks.

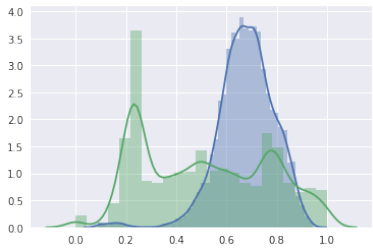

Fig. 8. Distribution of pairwise vertex distances without preprocessing (blue) and after quantile transformation (green).

In our evaluations, we performed this quantile transformation prior to every clustering, with $n_{quantile} = 10$.

The results are presented Table 6. We compared the mean NMI using the t-test, and checked that the differences between the obtained values are statistically significant.

We observe that our approach is competitive with state of the art methods in the case of non-attributed graphs on the benchmark datasets. In one specific case, we even observe that Graph trees significantly outperforms state of the art results, on the SBM graphs. This seems to confirm that our method performs particularly well in the case of clusters of different size.

Table 5. Datasets used for the evaluation of clustering on simple graphs using Graph-trees

Dataset	# vertices	# edges	Average degree	# clusters
Football	115	1226	10.66	10
Email-Eu-Core	1005	25571	33.24	42
Polbooks	105	441	8.40	3
SBM3	450	65994	293.307	3

Table 6. Comparison of NMI on benchmark graph datasets. Best results are in boldface.

Dataset	Graph-trees	Louvain	MCL
Football	**0.923 (0.007)**	**0.924 (0.000)**	0.879 (0.015)
Email-Eu-Core	**0.649 (0.008)**	0.428 (0.000)	0.589 (0.012)
Polbooks	0.524 (0.012)	0.521 (0.000)	**0.544 (0.02)**
SBM3	**0.998 (0.005)**	0.684 (0.000)	0.846 (0.000)

Using the modularity of the partitions Another way to assess the performance of GT is to use the modularity.

Modularity measures the strength of a specific partition, and is obtained by (2).

$$modularity(V_1, \ldots, V_k) = \sum_{k=1}^{K} \frac{|E_{kk}|}{2|E|} - (\sum_{l=1}^{K} \frac{|E_{kl}|}{2|E|})^2 \tag{2}$$

The intuition is that the higher the modularity, the lower the proportion of edges between vertices in different clusters.

Our approach here is to compare the modularities obtained using partitions obtained with GT, and the ones obtained using the ground truth values as partitions.

We performed this evaluation on the datasets presented Table 7. WebKB represents relations between web pages of four universities, where each vertex label corresponds to the university and the attributes represent the words that appear in the page. The Parliament dataset is a graph where the vertices represent french parliament members, linked by an edge if they cosigned a bill together. The vertex attributes indicate their constituency, and each vertex has a label that corresponds to their political party. The lawyers dataset represents relations between attorneys, that are described with categorical attributes for each of them. Finally, the HVR graph represent relations between malaria genes.

Table 7. Datasets used for the evaluation of clustering using the modularity

Dataset	# vertices	# edges	# clusters
WebKB	877	1480	4
Parliament	451	11646	7
HVR	307	6526	2
Lawyers	71	575	2

The procedure is the following. We first compute the vertex-vertex distances using GT. We then apply t-SNE and use the k-means algorithm on the points in the embedded space. We set k to the number of clusters, since we have the

ground truths. We computed the modularity of the clusterings obtained in 2 settings: (i) using a distance only on the structures using GT and (ii) using the ground truth values. We repeat these steps 5 times and report the means and standard deviations. The results are presented Table 8.

Table 8. Modularities using GT or the ground truth labels. Best results are indicated in boldface.

Dataset	GT	Ground truth
WebKB	**0.74**	**0.74**
HVR	0.15	0.15
Parliament	**0.46**	0.20
Lawyers	**0.27**	**0.26**

We notice that the structural distance using GT leads to good results, even higher than the modularities obtained using the ground truths. This can be explained by the fact that ground truth labels do not always constitute an absolute truth, especially in terms of network structure.

Overall, GT seems to be a good candidate to compute dissimilarities between vertices of a graph.

4 Discussion and future work

In this paper, we presented a method based on the construction of random trees to compute similarities between graph vertices, called GT. The method works by recursively splitting the set of vertices. Each split is performed by randomly sampling a new vertex from the vertex set of the previous node. Direct neighbours of this vertex then create a new node in the tree, while the other vertices create another node in the tree. A distance between the vertices can then be defined, as the leaves correspond to a partition of the vertex space, and where each region contains similar vertices. In this work, we propose to use a mass-based approach to compute this distance.

We show that the method we propose is competitive with state of the art methods in terms of quality of clustering on graphs. In our benchmark datasets, the mean differences $\bar{\Delta}$ between the mean intracluster similarities and the mean intercluster similarities tends to be higher when GT is used. However, it should be noted that although this result is not promising, our proposed method has a lower running time. Moreover, our assessment performing an actual clustering on our data showed that the method we propose is competitive in terms of NMI against to state-of-the-art methods, *i.e* Louvain and MCL. This is particularly true in the case on the stochastic block model graph, where GT outperforms Louvain and MCL by 45% and 22%, respectively. This result might be a strong hint that the mass-based approach we use is particularly efficient in the case of graphs with clusters of variable size and density.

Comparing the modularities of partitions obtained using GT *versus* the ground truth partitions also shows that GT exhibits a good performance.

One of the appealing aspects of GT is that by computing forests of graph trees and other trees that specialize in other types of input data, *e.g*, feature vectors, it is then possible to compute pairwise dissimilarities between vertices in vertex-attributed graphs. While many existing methods that tackle this problem are limited to categorical attributes and/or require a costly transformation of the graph, a tree-based approach would not suffer from these shortcomings. Indeed, the trees being built on the vertex attributes can take as an input both numeric and symbolic data. We already performed some experiments in the attributed graph settings that we didn't present in this paper. While the approach seems promising, our first results showed that the quality of the clusterings varies greatly between different datasets. This observation may be due to multiple factors, such as the aggregation method chosen to aggregate the graph trees and the attribute trees.

It should be noted that most of our empirical results depend on the choice of a specific clustering algorithm. Indeed, GT is not a clustering method *per se*, but a method to compute pairwise distances between nodes. As other distance-based methods, this is a strength of the method we propose in this paper. Indeed, the clustering task can be performed using many algorithms, leveraging their respective strengths and weaknesses.

One of the shortcomings of our proposed approach is that in graphs where the average degree is low, the splitting method leads to large unbalanced trees. Other approaches can be used, such as a splitting based on a specific criterion. As this approach merges the fields of graph clustering and decision trees, interesting research directions can be pursued.

References

1. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008)
2. Chen, H.H., Giles, C.L.: ASCOS: an asymmetric network structure context similarity measure. In: 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013). pp. 442–449. IEEE (2013)
3. Dalleau, K., Couceiro, M., Smail-Tabbone, M.: Unsupervised extremely randomized trees. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 478–489. Springer (2018)
4. Dongen, S.: A cluster algorithm for graphs (2000)
5. ERDdS, P., R&wi, A.: On random graphs I. Publ. Math. Debrecen **6**, 290–297 (1959)
6. Jeh, G., Widom, J.: SimRank: a measure of structural-context similarity. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 538–543. ACM (2002)

7. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. ACM Transactions on Knowledge Discovery from Data (TKDD) **1**(1), 2 (2007)
8. Maaten, L.v.d., Hinton, G.: Visualizing data using t-SNE. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V.: Scikit-learn: Machine learning in Python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
10. Schaeffer, S.E.: Graph clustering. Computer science review **1**(1), 27–64 (2007)
11. Ting, K.M., Zhu, Y., Carman, M., Zhu, Y., Zhou, Z.H.: Overcoming key weaknesses of distance-based neighbourhood methods using a data dependent dissimilarity measure. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1205–1214. ACM (2016)
12. Yin, H., Benson, A.R., Leskovec, J., Gleich, D.F.: Local higher-order graph clustering. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 555–564. ACM (2017)