Hierarchical Graph Clustering by Node Pair Sampling

Thomas Bonald
Telecom ParisTech
Paris, France
thomas.bonald@telecom-paristech.fr

Bertrand Charpentier
Telecom ParisTech
Paris, France
bertrand.charpentier@telecom-paristech.fr

Alexis Galland
Inria
Paris, France
alexis.galland@inria.fr

Alexandre Hollocou
Inria
Paris, France
alexandre.hollocou@inria.fr

ABSTRACT
We present a novel hierarchical graph clustering algorithm inspired
by modularity-based clustering techniques. The algorithm is ag-
gglomerative and based on a simple distance between clusters in-
duced by the probability of sampling node pairs. We prove that this
distance is reducible, which enables the use of the nearest-neighbor
chain to speed up the agglomeration. The output of the algorithm
is a regular dendrogram, which reveals the multi-scale structure
of the graph. The results are illustrated on both synthetic and real
datasets.

CCS CONCEPTS
• Computing methodologies → Cluster analysis;

KEYWORDS
Hierarchical clustering, dendrogram, agglomerative algorithm

ACM Reference format:
https://doi.org/

1 INTRODUCTION
Many datasets can be represented as graphs, being the graph ex-
plicitly embedded in data (e.g., the friendship relation of a social
network) or built through some suitable similarity measure be-
tween data items (e.g., the number of papers co-authored by two
researchers). Such graphs often exhibit a complex, multi-scale com-
nunity structure where each node is involved in many groups of
nodes, so-called communities, of different sizes.

One of the most popular graph clustering algorithm is known as
Louvain in name of the university of its inventors [2]. It is based
on the greedy maximization of the modularity, a classical objective
function introduced in [19]. The Louvain algorithm is fast, memory-
efficient, and provides meaningful clusters in practice. It does not
enable an analysis of the graph at different scales, however [8, 11].
While the current version of the algorithm1 includes a resolution
parameter, this parameter is not directly related to the target cluster
size and thus hard to adjust in practice.

In this paper, we present a novel algorithm for hierarchical clus-
tering that captures the multi-scale nature of real graphs. The algo-
rithm is fast, memory-efficient and parameter-free. It relies on a
novel notion of distance between clusters induced by the probabil-
ity of sampling node pairs. We prove that this distance is reducible,
which guarantees that the resulting hierarchical clustering can be
represented by regular dendrograms and enables a fast implemen-
tation of our algorithm through the nearest-neighbor chain scheme,
a classical technique for agglomerative algorithms [16].

The rest of the paper is organized as follows. We present the
related work in Section 2. The notation used in the paper, the dis-
tance between clusters used to aggregate nodes and the clustering
algorithm are presented in Sections 3, 4 and 5. The link with modu-
ularity and the Louvain algorithm is explained in Section 6. Section 7
shows the experimental results and Section 8 concludes the paper.

2 RELATED WORK
Most graph clustering algorithms are not hierarchical and rely on
some resolution parameter that allows one to adapt the clustering to
the dataset and to the intended purpose [1, 10, 17, 21]. This param-
erter is hard to adjust in practice, which motivates the present work.
The classical hierarchical clustering techniques apply to vector data
[16, 25]. They do not directly apply to graphs, unless the graph is
embedded in some metric space, through spectral techniques for
instance [7, 14].

A number of hierarchical clustering algorithms have been de-
veloped specifically for graphs. The most popular algorithms are
agglomerative and characterized by some distance between clusters,
see [3, 9, 18, 20]. None of these distances has been proved to be re-
ducible, a key property of our algorithm. Among non-agglomerative
algorithms, the divisive approach of [19] is based on the notion
of edge betweenness while the iterative approach of [23] and [12]
look for local maxima of modularity or some fitness function; other
approaches rely on statistical interference [4], replica correlations [22]
and graph wavelets [24]. To our knowledge, non of these algorithms
has been proved to lead to regular dendrograms (that is, without
inversion).

Finally, the Louvain algorithm also provides a hierarchy, induced
by the successive aggregation steps of the algorithm [2]. This is not
a full hierarchy, however, as there are typically a few aggregation
steps. Moreover, the same resolution is used in the optimization of

1See the python-louvain Python package.
modularity across all levels of the hierarchy, while the numbers of clusters decrease rapidly after a few aggregation steps. We shall see that our algorithm may be seen as a modified version of Louvain using a sliding resolution, that is adapted to the current agglomeration step of the algorithm.

3 NOTATION

Consider a weighted, undirected graph $G = (V, E)$ of $n$ nodes, with $V = \{1, \ldots, n\}$. Let $A$ be the corresponding weighted adjacency matrix. This is a symmetric, non-negative matrix such that for each $i, j \in V$, $A_{ij} > 0$ if and only if there is an edge between $i$ and $j$, in which case $A_{ij}$ is the weight of edge $\{i, j\} \in E$. We refer to the weight of node $i$ as the sum of the weights of its incident edges,

$$w_i = \sum_{j \in V} A_{ij}.$$ 

Observe that for unit weights, $w_i$ is the degree of node $i$. The total weight of the nodes is:

$$w = \sum_{i \in V} w_i = \sum_{i,j \in V} A_{ij}.$$ 

We refer to a clustering $C$ as any partition of $V$. In particular, each element of $C$ is a subset of $V$, we refer to as a cluster.

4 NODE PAIR SAMPLING

The edge weights induce a probability distribution on node pairs,

$$\forall i, j \in V, \quad p(i, j) = \frac{A_{ij}}{w},$$

and a probability distribution on nodes,

$$\forall i \in V, \quad p(i) = \sum_{j \in V} p(i, j) = \frac{w_i}{w}.$$ 

Observe that the joint distribution $p(i, j)$ depends on the graph (in particular, only neighbors $i, j$ are sampled with positive probability), while the marginal distribution $p(i)$ depends on the graph through the node weights only. Since the graph is undirected, we have $p(i, j) = p(j, i)$; the probability of sampling edge $i, j$ (no matter the order of the nodes) is $p(i, j) + p(j, i) = 2p(i, j)$.

We now define the distance between two distinct nodes $i, j$ as the node pair sampling ratio:

$$d(i, j) = \frac{p(i)p(j)}{p(i, j)},$$

with $d(i, j) = +\infty$ if $p(i, j) = 0$ (i.e., $i$ and $j$ are not neighbors). Nodes $i, j$ are close for this distance if the pair $i, j$ is sampled much more frequently through the joint distribution $p(i, j)$ than through the product distribution $p(i)p(j)$. For unit weights, the joint distribution is uniform over the edges, so that the closest node pair is the pair of neighbors having the lowest degree product.

Another interpretation of the node distance $d$ follows from the conditional probability,

$$\forall i, j \in V, \quad p(i|j) = \frac{p(i, j)}{p(j)} = \frac{A_{ij}}{w_j}.$$ 

This is the conditional probability of sampling $i$ given that $j$ is sampled (from the joint distribution). The distance between $i$ and $j$ can then be written

$$d(i, j) = \frac{p(i)p(j)}{p(i, j)} = \frac{p(j)}{p(j|j)}.$$ 

The closer the nodes $i, j$ for this distance, the more likely $i$ is sampled given that $j$ is sampled (equivalently, for sampling $j$ given $i$).

Similarly, consider some clustering $C$ of the graph (that is, a partition of $V$). The probability distribution on node pairs defined above induce a probability distribution on cluster pairs,

$$\forall a, b \in C, \quad p(a, b) = \sum_{i \in a, j \in b} p(i, j),$$

and a probability distribution on clusters,

$$\forall a \in C, \quad p(a) = \sum_{i \in a} p(i) = \sum_{b \in C} p(a, b).$$

Note that $p(a, b)$ is the probability of sampling a node in $a$ and a node in $b$ (in this order), while $p(a)$ is the probability of sampling a node in cluster $a$. By symmetry, we have $p(a, b) = p(b, a)$ and $2p(a, b)$ is the probability of sampling distinct clusters $a, b$ (no matter the order).

We define the distance between two distinct clusters $a, b$ as the cluster pair sampling ratio:

$$d(a, b) = \frac{p(a)p(b)}{p(a, b)},$$

with $d(a, b) = +\infty$ if $p(a, b) = 0$ (i.e., there is no edge between clusters $a$ and $b$). Defining the conditional probability

$$\forall a, b \in C, \quad p(a|b) = \frac{p(a, b)}{p(b)},$$

which is the conditional probability of sampling $a$ given that $b$ is sampled, we get

$$d(a, b) = \frac{p(a)}{p(a|b)} = \frac{p(b)}{p(b|a)}.$$ 

Again, the closer the clusters $a, b$ for this distance, the more likely $a$ is sampled given that $b$ is sampled (and the same for $b$ given $a$).

This distance will be used in the agglomerative algorithm to merge the closest clusters. We have the following key results.

**PROPOSITION 1 (UPDATE FORMULA).** For any distinct clusters $a, b, c \in C$,

$$d(a \cup b, c) = \left( \frac{p(a)}{p(a \cup b)} \frac{1}{d(a, c)} + \frac{p(b)}{p(a \cup b)} \frac{1}{d(b, c)} \right)^{-1}.$$ 

**Proof.** We have:

$$p(a \cup b)p(c) = p(a \cup b, c) = p(a, c) + p(b, c),$$

$$= p(a)p(c|a) + p(b)p(c|b) \geq d(a, c) + d(b, c).$$

from which the formula follows. $\square$

**PROPOSITION 2 (REDUCIBILITY).** For any distinct clusters $a, b, c \in C$,

$$d(a \cup b, c) \geq \min(d(a, c), d(b, c)).$$
5 CLUSTERING ALGORITHM

The agglomerative approach consists in starting from individual clusters (i.e., each node is in its own cluster) and merging clusters recursively. At each step of the algorithm, the two closest clusters are merged. We obtain the following algorithm:

(1) Initialization
\[ C \leftarrow \{\{1\}, \ldots, \{n\}\} \]
\[ L \leftarrow \emptyset \]

(2) Agglomeration
For \( t = 1, \ldots, n - 1 \),
- \( a, b \leftarrow \arg \min_{a', b' \in C, a' \neq b'} d(a', b') \)
- \( C \leftarrow C \setminus \{a, b\}; C \leftarrow C \cup \{a \cup b\} \)
- \( L \leftarrow L \cup \{(a, b)\} \)

(3) Return \( L \)

The successive clusterings \( C_0, C_1, \ldots, C_{n-1} \) produced by the algorithm, with \( C_0 = \{\{1\}, \ldots, \{n\}\} \), can be recovered from the list \( L \) of successive merges. Observe that clustering \( C_t \) consists of \( n - t \) clusters, for \( t = 0, 1, \ldots, n - 1 \). By the reducibility property, the corresponding sequence of distances \( d_0, d_1, \ldots, d_{n-1} \) between merged clusters, with \( d_0 = 0 \), is non-decreasing, resulting in a regular dendrogram (that is, without inversions) [16].

It is worth noting that the graph \( G \) does not need to be connected. If the graph consists of \( k \) connected components, then the clustering \( C_{n-k} \) gives these \( k \) connected components, whose respective distances are infinite; the \( k - 1 \) last merges can then be done in an arbitrary order. Moreover, the hierarchies associated with these connected components are independent of one another (i.e., the algorithm successively applied to the corresponding subgraphs would produce exactly the same clustering). Similarly, we expect the clustering of weakly connected subgraphs to be approximately independent of one another. This is not the case of the Louvain algorithm, whose clustering depends on the whole graph through the total weight \( w \), a shortcoming related to the resolution limit of modularity (see Section 6).

Implementation using the aggregate graph. In view of (2), for any clustering \( C \) of \( V \), the distance \( d(a, b) \) between two clusters \( a, b \in C \) is the distance between two nodes \( a, b \) of the following aggregate graph: nodes are the elements of \( C \) and the weight between \( a, b \in C \) (including the case \( a = b \), corresponding to a self-loop) is \( \sum_{i \in a, j \in b} A_{ij} \). Thus the agglomerative algorithm can be implemented by merging nodes and updating the weights (and thus the distances between nodes) at each step of the algorithm. Since the initial nodes of the graph are indexed from 0 to \( n - 1 \), we index the cluster created at step \( t \) of the algorithm by \( n + t \). We obtain the following equivalent version of the above algorithm, where the clusters are coded by their respective indices in the aggregate graph:

(1) Initialization
\[ V \leftarrow \{1, \ldots, n\} \]
\[ L \leftarrow \emptyset \]

(2) Agglomeration
For \( t = 1, \ldots, n-1 \),
- \( i, j \leftarrow \arg \min_{i', j' \in V, i' \neq j'} d(i', j') \)
- \( L \leftarrow L \cup \{(i', j')\} \)
- \( V \leftarrow V \setminus \{i, j\} ; V \leftarrow V \cup \{n + t\} \)
- \( p(n + t) \leftarrow p(i) + p(j) \)
- \( p(n + t, u) \leftarrow p(i, u) + p(j, u) \) for \( u \in V \setminus \{n + t\} \)

(3) Return \( L \)

Observe that the aggregate graph has \( n - t \) nodes after step \( t \). The associate sampling distribution \( p \) is updated, which in turn modifies the distance \( d \) between nodes in the aggregate graph, through the formula (1). In practice, a copy of the graph is done at the beginning of the algorithm to avoid the destruction of the data structure containing the initial graph \( G \).

Implementation using the nearest-neighbor chain. By the reducibility property of the distance, the algorithm can be implemented through the nearest-neighbor chain scheme [16]. Starting from an arbitrary node of the aggregate graph, a chain a nearest neighbors is formed. Whenever two nodes of the chain are mutual nearest neighbors, these two nodes are merged and the chain is updated recursively, until the initial node is eventually merged. This scheme reduces the search of a global minimum (the pair of nodes \( i, j \) that minimizes \( d(i, j) \)) to that of a local minimum (any pair of nodes \( i, j \) such that \( d(i, j) = \min_{i \neq j} d(i, j') = \min_{i \neq j} d(i', j) \)), which speeds up the algorithm while returning exactly the same hierarchy. It only requires a consistent tie-breaking rule for equal distances (e.g., any node at equal distance of \( i \) and \( j \) is considered as closer to \( i \) if and only if \( i < j \)). Observe that the space complexity of the algorithm is in \( O(m) \), where \( m \) is the number of edges of \( G \) (i.e., the graph size).

6 LINK WITH MODULARITY

The modularity is a standard metric to assess the quality of a clustering \( C \) (any partition of \( V \)). Let \( \delta_C(i, j) = 1 \) if \( i, j \) are in the same cluster under clustering \( C \), and \( \delta_C(i, j) = 0 \) otherwise. The modularity of clustering \( C \) is defined by [19]:

\[
Q(C) = \frac{1}{w} \sum_{i, j \in V} (A_{ij} - \frac{w_i w_j}{w}) \delta_C(i, j),
\]

which can be written in terms of probability distributions,

\[
Q(C) = \sum_{i, j \in V} (p(i, j) - p(i)p(j)) \delta_C(i, j).
\]

Thus the modularity is the difference between the probabilities of sampling two nodes of the same cluster under the joint distribution \( p(i, j) \) and under the product distribution \( p(i)p(j) \). It can also be expressed from the probability distributions at the cluster level,

\[
Q(C) = \sum_{a \in C} (p(a, a) - p(a)^2).
\]

It is clear from (3) that any clustering \( C \) maximizing modularity has some resolution limit, as pointed out in [8], because the second term is normalized by the total weight \( w \) and thus becomes negligible for too small clusters. To go beyond this resolution limit,
it is necessary to introduce a multiplicative factor $\gamma$, called the resolution. The modularity becomes:

$$Q_\gamma(C) = \sum_{i,j \in V} (p(i,j) - \gamma p(i)p(j))\delta_C(i,j),$$

or equivalently,

$$Q_\gamma(C) = \sum_{a \in C} (p(a,a) - \gamma p(a)^2).$$

This resolution parameter can be interpreted through the Potts model of statistical physics \cite{roberts2003modularity}, random walks \cite{lima2012inferring}, or statistical inference of a stochastic block model \cite{peixoto2014community}. For $\gamma = 0$, the resolution is minimum and there is a single cluster, that is $C = \{1, \ldots, n\}$; for $\gamma \to +\infty$, the resolution is maximum and each node has its own cluster, that is $C = \{\{1\}, \ldots, \{n\}\}$.

The Louvain algorithm consists, for any fixed resolution parameter $\gamma$, of the following steps:

1. **Initialization**
   
   $C \leftarrow \{\{1\}, \ldots, \{n\}\}$

2. **Iteration**
   
   While modularity $Q_\gamma(C)$ increases, update $C$ by moving one node from one cluster to another.

3. **Aggregation**
   
   Merge all nodes belonging to the same cluster, update the weights and apply step 2 to the resulting aggregate graph while modularity is increased.

4. **Return**
   
   The result of step 2 depends on the order in which nodes and clusters are considered; typically, nodes are considered in a cyclic way and the target cluster of each node is that maximizing the modularity increase.

Our algorithm can be viewed as a modularity-maximizing scheme with a *sliding* resolution. Starting from the maximum resolution where each node has its own cluster, we look for the first value of the resolution parameter $\gamma$, say $\gamma_1$, that triggers a single merge between two nodes, resulting in clustering $C_1$. In view of (4), we have:

$$\gamma_1 = \max_{i,j \in V} \frac{p(i,j)}{p(i)p(j)}.$$

These two nodes are merged (corresponding to the aggregation phase of the Louvain algorithm) and we look for the next value of the resolution parameter, say $\gamma_2$, that triggers a single merge between two nodes, resulting in clustering $C_2$, and so on. By construction, the resolution at time $t$ (that triggers the $t$-th merge) is $\gamma_t = 1/d_t$ and the corresponding clustering $C_t$ is that of our algorithm. In particular, the sequence of resolutions $\gamma_1, \ldots, \gamma_n$ is non-increasing.

To summarize, our algorithm consists of a simple but deep modification of the Louvain algorithm, where the iterative step (step 2) is replaced by a single merge, at the best current resolution (that resulting in a single merge). In particular, unlike the Louvain algorithm, our algorithm provides a full hierarchy. Moreover, the sequence of resolutions $\gamma_1, \ldots, \gamma_n$ can be used as an input to the Louvain algorithm. Specifically, the resolution $\gamma_1$ provides exactly $n - t$ clusters in our case, and the Louvain algorithm is expected to provide approximately the same number of clusters at this resolution.

7 **EXPERIMENTS**

We have coded our hierarchical clustering algorithm, we refer to as Paris\(^3\), in Python. All material necessary to reproduce the experiments presented below is available online\(^4\).

**Qualitative results.** We start with a simple hierarchical stochastic block model, as described in \cite{peixoto2014community}. There are $n = 160$ nodes structured in 2 levels, with 4 blocks of 40 nodes at level 1, each block of 40 nodes being divided into 4 blocks of 10 nodes at level 2 (see Figure 1).

![Level 1 and Level 2](image)

**Figure 1:** A hierarchical stochastic block model with 2 levels of hierarchy.

The output of Paris is shown in Figure 2 as a dendrogram where the distances (on the $y$-axis) are in log-scale. The two levels of hierarchy clearly appear.

![Dendrogram](image)

**Figure 2:** Dendrogram associated with the clustering of Paris on a hierarchical stochastic block model of 16 blocks.

We also show in Figure 3 the number of clusters with respect to the resolution parameter $\gamma$ for Paris (top) and Louvain (bottom). The results are very close, and clearly show the hierarchical structure of the model (vertical lines correspond to changes in the number of clusters). The key difference between both algorithms is that, while Louvain needs to be run for each resolution parameter $\gamma$ (here 100 values ranging from 0.01 to 20), Paris is run only once, the relevant resolutions being direct outputs of the algorithm, embedded in the dendrogram (see Section 6).

\(^3\)Paris = Pairwise Agglomeration Induced by Sampling.

\(^4\)See https://github.com/tbonald/paris
We now consider four real datasets, whose characteristics are summarized in Table 1.

<table>
<thead>
<tr>
<th>Graph</th>
<th># nodes</th>
<th># edges</th>
<th>Avg. degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStreet</td>
<td>5,993</td>
<td>6,957</td>
<td>2.3</td>
</tr>
<tr>
<td>OpenFlights</td>
<td>3,097</td>
<td>18,193</td>
<td>12</td>
</tr>
<tr>
<td>Wikipedia Schools</td>
<td>4,589</td>
<td>106,644</td>
<td>46</td>
</tr>
<tr>
<td>Wikipedia Humans</td>
<td>702,782</td>
<td>3,247,884</td>
<td>9.2</td>
</tr>
</tbody>
</table>

Table 1: Summary of the datasets.

The first dataset, extracted from OpenStreetMap, is the graph formed by the streets of the center of Paris. To illustrate the quality of the hierarchical clustering returned by our algorithm, we have extracted the two “best” clusterings, in terms of ratio between successive distance merges in the corresponding dendrogram; the results are shown in Figure 4. The best clustering gives two clusters, Rive Droite (with Ile de la Cité) and Rive Gauche, the two banks separated by the river Seine; the second best clustering divides these two clusters into sub-clusters.

The second dataset, extracted from OpenFlights, is the graph of airports with the weight between two airports equal to the number of daily flights between these airports. We run Paris and extract the best clusterings from the largest component of the graph, as for the OpenStreet graph. The first two best clusterings isolate the Island/Greenland area and the Alaska from the rest of the world, the corresponding airports forming dense clusters, lightly connected with the other airports. The following two best clusterings are shown in Figure 5, with respectively 5 and 10 clusters corresponding to meaningful continental regions of the world.

---

5https://openstreetmap.org
6https://openflights.org
The third dataset is the graph formed by links between pages of Wikipedia for Schools\textsuperscript{7}, see [26]. The graph is considered as undirected. Table 2 (top table) shows the 10 largest clusters of \( C_{n-100} \), the 100 last clusters found by Paris. Only pages of highest degrees are shown for each cluster. Observe that the ability of selecting the clustering associated with some target number of clusters is one of the key advantage of Paris over Louvain. Moreover, Paris gives a full hierarchy of the pages, meaning that each of these clusters is divided into sub-clusters in the output of the algorithm. Table 2 (bottom table) gives for instance, among the 500 clusters found by Paris (that is, in \( C_{n-500} \)), the 10 largest clusters that are subclusters of cluster #1, related to taxonomy and animals. The subclusters tend to give meaningful groups of animals, revealing the multi-scale structure of the dataset.

### Top-10 clusters (among 100 clusters)

<table>
<thead>
<tr>
<th>#</th>
<th>Size</th>
<th>Top articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>288</td>
<td>Scientific classification, Animal, Chordate</td>
</tr>
<tr>
<td>2</td>
<td>231</td>
<td>Iron, Oxygen, Electron, Hydrogen, Phase</td>
</tr>
<tr>
<td>3</td>
<td>196</td>
<td>England, Wales, Elizabeth II of the United Kingdom</td>
</tr>
<tr>
<td>4</td>
<td>164</td>
<td>Physics, Mathematics, Science, Albert Einstein</td>
</tr>
<tr>
<td>5</td>
<td>148</td>
<td>Portugal, Ethiopia, Mozambique, Madagascar</td>
</tr>
<tr>
<td>6</td>
<td>139</td>
<td>Washington, D.C., President of the United States</td>
</tr>
<tr>
<td>7</td>
<td>129</td>
<td>Earth, Sun, Astronomy, Star, Gravitation</td>
</tr>
<tr>
<td>8</td>
<td>127</td>
<td>Plant, Fruit, Sugar, Tea, Flower</td>
</tr>
<tr>
<td>9</td>
<td>104</td>
<td>Internet, Computer, Mass media, Latin alphabet</td>
</tr>
<tr>
<td>10</td>
<td>99</td>
<td>Jamaica, The Beatles, Hip hop music, Jazz, Piano</td>
</tr>
</tbody>
</table>

### Top-10 subclusters of cluster #1

<table>
<thead>
<tr>
<th>#</th>
<th>Size</th>
<th>Top pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>71</td>
<td>Dinosaur, Fossil, Reptile, Cretaceous, Jurassic</td>
</tr>
<tr>
<td>2</td>
<td>51</td>
<td>Binomial nomenclature, Bird, Carolus Linnaeus</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>Mammal, Lion, Cheetah, Giraffe</td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>Animal, Ant, Arthropod, Spider, Bee</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>Dog, Bat, Vampire, George Byron</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>Eagle, Glacier National Park, Golden Eagle</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>Chordate, Parrot, Gull, Surtsey, Herring Gull</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>Feather, Extinct birds, Mount Rushmore</td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>Miocene, Eocene, Bryce Canyon National Park</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>Crow, Dove, Pigeon, Rock Pigeon</td>
</tr>
</tbody>
</table>

Table 2: Clusters of the Wikipedia Schools graph extracted from the hierarchical clustering returned by Paris.

The fourth dataset is the subgraph of Wikipedia restricted to pages related to humans. We have done the same experiment as for the Wikipedia Schools graph, and the results are shown in Table 3. Again, we observe that clusters form relevant groups of people, with cluster #1 corresponding to political figures for instance, this cluster consisting of meaningful subgroups as shown in the bottom of Table 3. All this information is embedded in the dendogram returned by Paris.

### Top-10 clusters (among 100 clusters)

<table>
<thead>
<tr>
<th>#</th>
<th>Size</th>
<th>Main pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41363</td>
<td>George W. Bush, Barack Obama, Bill Clinton</td>
</tr>
<tr>
<td>2</td>
<td>34291</td>
<td>Alex Ferguson, David Beckham, Pelé</td>
</tr>
<tr>
<td>3</td>
<td>25225</td>
<td>Abraham Lincoln, George Washington</td>
</tr>
<tr>
<td>4</td>
<td>23488</td>
<td>Madonna, Woody Allen, Martin Scorsese</td>
</tr>
<tr>
<td>5</td>
<td>23044</td>
<td>Wolfgang Amadeus Mozart, J. Sebastian Bach</td>
</tr>
<tr>
<td>6</td>
<td>22236</td>
<td>Elvis Presley, Bob Dylan, Elton John, David Bowie</td>
</tr>
<tr>
<td>7</td>
<td>20429</td>
<td>Queen Victoria, George III of the UK, Edward VII</td>
</tr>
<tr>
<td>8</td>
<td>19105</td>
<td>Sting, Jawaharlal Nehru, Rabindranath Tagore</td>
</tr>
<tr>
<td>9</td>
<td>18348</td>
<td>Edward I of England, Edward III of England</td>
</tr>
<tr>
<td>10</td>
<td>14668</td>
<td>Jack Kemp, Brett Favre, Peyton Manning</td>
</tr>
</tbody>
</table>

### Top-10 subclusters of cluster #1

<table>
<thead>
<tr>
<th>#</th>
<th>Size</th>
<th>Top pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2722</td>
<td>Barack Obama, John McCain, Dick Cheney</td>
</tr>
<tr>
<td>2</td>
<td>2443</td>
<td>Arnold Schwarzenegger, Jerry Brown, Ralph Nader</td>
</tr>
<tr>
<td>3</td>
<td>2058</td>
<td>Osama bin Laden, Hamid Karzai, Alberto Gonzales</td>
</tr>
<tr>
<td>4</td>
<td>1917</td>
<td>Dwight D. Eisenhower, Harry S. Truman</td>
</tr>
<tr>
<td>5</td>
<td>1742</td>
<td>George W. Bush, Condoleezza Rice, Colin Powell</td>
</tr>
<tr>
<td>6</td>
<td>1700</td>
<td>Bill Clinton, Thurgood Marshall, Mike Huckabee</td>
</tr>
<tr>
<td>7</td>
<td>1559</td>
<td>Ed Rendell, Arlen Specter, Rick Santorum</td>
</tr>
<tr>
<td>8</td>
<td>1545</td>
<td>Theodore Roosevelt, Herbert Hoover</td>
</tr>
<tr>
<td>9</td>
<td>1523</td>
<td>Ronald Reagan, Richard Nixon, Jimmy Carter</td>
</tr>
<tr>
<td>10</td>
<td>1508</td>
<td>Rudy Giuliani, Michael Bloomberg</td>
</tr>
</tbody>
</table>

Table 3: Clusters of the Wikipedia Humans graph extracted from the hierarchical clustering returned by Paris.

**Quantitative results.** To assess the quality of the hierarchical clustering, we use the cost function proposed in [6] and given by:

\[
\sum_{a,b} p(a,b)(|a| + |b|),
\]

where the sum is over all left and right clusters \( a, b \) attached to any internal node of the tree representing the hierarchy. This is the expected size of the smallest subtree containing two random nodes \( i,j \), sampled from the joint distribution \( p(i,j) \) introduced in Section 4. If the tree indeed reflects the underlying hierarchical structure of the graph, we expect most edges to link nodes that are close in the tree, i.e., whose common ancestor is relatively far from the root. The size of the corresponding subtree (whose root is this common ancestor) is expected to be small, meaning that (5) is a relevant cost function. Moreover, it was proved in [5] that if the graph is perfectly hierarchical, the underlying tree is optimal with respect to this cost function.

The results are presented in Table 5 for the graphs considered so far and the graphs of Table 4, selected from the SNAP datasets [13]. The cost function is normalized by the number of nodes \( n \) so as to get a value between 0 and 1. We compare the performance of Paris to that of a spectral algorithm where the nodes are embedded in a space of dimension 20 by using the 20 leading eigenvectors of the Laplacian matrix \( \mathbf{L} = \mathbf{D} - \mathbf{A} \) (\( \mathbf{D} \) is the diagonal matrix of node weights) and applying the Ward method in the embedding space. The spectral decomposition of the Laplacian is based on standard functions on sparse matrices available in the Python package scikit-learn.

\[\text{https://schools-wikipedia.org}\]
Hierarchical Graph Clustering

Observe that we do not include Louvain in these experiments as this algorithm does not provide a full hierarchy of the graph, so that the cost function (5) is not applicable.

<table>
<thead>
<tr>
<th>Graph</th>
<th># nodes</th>
<th># edges</th>
<th>Avg. degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>4,039</td>
<td>88,234</td>
<td>44</td>
</tr>
<tr>
<td>Amazon</td>
<td>334,863</td>
<td>925,872</td>
<td>5.5</td>
</tr>
<tr>
<td>DBLP</td>
<td>317,080</td>
<td>1,049,866</td>
<td>6.6</td>
</tr>
<tr>
<td>Twitter</td>
<td>81,306</td>
<td>1,342,310</td>
<td>33</td>
</tr>
<tr>
<td>Youtube</td>
<td>1,134,890</td>
<td>2,987,624</td>
<td>5.2</td>
</tr>
<tr>
<td>Google</td>
<td>855,802</td>
<td>4,291,352</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4: Summary of the considered graphs from SNAP.

The results are shown when the algorithm runs within some time limit (less than 1 hour on a computer equipped with a 2.8GHz Intel Core i7 CPU with 16GB of RAM). The best performance is displayed in bold characters. Observe that both algorithms have similar performance on those graphs where the results are available. However, Paris is much faster than the spectral algorithm, as shown by Table 6 (for each algorithm, the initial load or copy of the graph is not included in the running time; running times exceeding 1 hour are not shown). Paris is even faster than Louvain in most cases, while providing a much richer information on the graph.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Spectral</th>
<th>Paris</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStreet</td>
<td>0.0102</td>
<td>0.0102</td>
</tr>
<tr>
<td>OpenFlights</td>
<td>0.125</td>
<td>0.130</td>
</tr>
<tr>
<td>Facebook</td>
<td>0.0479</td>
<td>0.0469</td>
</tr>
<tr>
<td>Wikipedia Schools</td>
<td>0.452</td>
<td>0.402</td>
</tr>
<tr>
<td>Amazon</td>
<td>–</td>
<td>0.0297</td>
</tr>
<tr>
<td>DBLP</td>
<td>–</td>
<td>0.110</td>
</tr>
<tr>
<td>Twitter</td>
<td>–</td>
<td>0.0908</td>
</tr>
<tr>
<td>Youtube</td>
<td>–</td>
<td>0.185</td>
</tr>
<tr>
<td>Wikipedia Humans</td>
<td>–</td>
<td>131</td>
</tr>
<tr>
<td>Google</td>
<td>–</td>
<td>0.0121</td>
</tr>
</tbody>
</table>

Table 5: Performance comparison of a spectral algorithm and Paris in terms of normalized Dasgupta’s cost.

8 CONCLUSION

We have proposed a hierarchical graph clustering algorithm based on a reducible distance between clusters. The algorithm is parameter-free, fast and memory-efficient. Future work will be dedicated to the automatic extraction of clusterings from the dendrogram, at the most relevant resolutions.

ACKNOWLEDGEMENT

Part of this work has been done while Bertrand Charpentier was a Master student at KTH, Sweden.

REFERENCES

[9] Jianbin Huang, Heli Sun, Jiawei Han, Hongbo Deng, Yiyu Sun, and Yizhang Liu. 2010. SHRINK: A Structural Clustering Algorithm for Detecting Hierarchical Communities in Networks. In Proceedings of ACM International Conference on Information and Knowledge Management.
