Introducing a mesoscopic scale with Conf luence in Modularity, to improve graph clustering resolution

Bruno Gaume, Alexandre Delanoe, Alp Mestanogullari

▶ **To cite this version:**

Bruno Gaume, Alexandre Delanoe, Alp Mestanogullari. Introducing a mesoscopic scale with Conf luence in Modularity, to improve graph clustering resolution. 2021. hal-03468437

HAL Id: hal-03468437
https://hal.science/hal-03468437

Preprint submitted on 7 Dec 2021

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.

Introducing a mesoscopic scale with *Confluence* in *Modularity*, to improve graph clustering resolution

Bruno Gaume, CNRS, CLLE, ISCPIF, gaume@univ-tlse2.fr
Alexandre Delanoë, CNRS, ISCPIF, alexandre.delanoe@iscpif.fr
Alp Mestanogullari, Well-Typed LLP, alp@well-typed.com

2021
May

Abstract

Given a graph $G = (V, E)$ and two vertices $i, j \in V$, we introduce $Confluence(G, i, j)$, a vertex mesoscopic closeness measure which brings together vertices from the same link-dense region of the graph G, and separates vertices coming from two distinct dense regions.

Confluence becomes a useful tool to avoid the resolution problems of the standard $Modularity(G, \Gamma)$ measure for a given clustering Γ, as evidenced by our comparative study between these two measures on toy graphs. We additionally present a heuristic to find a partitional clustering of a graph that tentatively optimizes a clustering quality function derived from *Confluence*, comparing the new heuristic's behaviour to the state of the art *Louvain* and *Infomap* methods on real terrain networks, while introducing a way to control the size of the resulting clusters along the way.

Contents

TODO dans l'ordre:

(1) **DONE** Confirmer que Version-Haskel=Version-Cython: Alp

(2) **DONE** Choisir les graphes de l'état de l'art:

- `https://snap.stanford.edu/data/com-DBLP.html`
- `https://snap.stanford.edu/data/com-Amazon.html`
- `https://networkrepository.com/actor-movie.php`

(3) **DONE** Rédiger la section 4.1 Implémentation: Alp et Alexandre

(4) **DONE** Rédiger la section 6 Conclusion et Perspectives: Bruno

(5) **DONE** Traduction en bon anglais: Alp et Alexandre

(6) Relecture: David et Quentin

(7) Formater le .tex selon les consignes d'édition de Physical-Review(E) : Bruno

(8) Soumettre l'article à Physical-Review(E) et Déposer l'article sur arxiv : Bruno

Cet article est en cours de rédaction et il est déposé sur `https://arxiv.org/user/` pour permetre au logiciel libre Gartgantexte de concourir à un prix du logiciel libre.

Le code Haskell provisoire est déposé ici : `https://gitlab.iscpif.fr/gargantext/gargantext-graph` sous la Licence accessible ici : `https://gitlab.iscpif.fr/gargantext/haskell-gargantext/blob/dev/LICENSE`

1 Introduction

Terrain networks are real world networks that model data gathered by field work, in diverse fields such as sociology, linguistics, biology, or graphs from the internet. Most terrain networks contrast with artificial graphs (deterministic or random) and share four similar properties [Watts and Strogatz, 1998, Albert and Barabasi, 2002, Newman, 2003]. They exhibit:

p1 : A low density (not many edges);

p2 : Short paths (the average number of edges L on the shortest path between two vertices is low);

p3 : A high clustering rate $C = \frac{3 \times number\ of\ triangles}{number\ of\ connected\ triplets}$ (locally densely connected subgraphs can be found whereas the whole graph is globally sparse);

p4 : A heavy-tailed degree distribution (the distribution of the degrees of the vertices of the graph can be approximated by a power law).

Clustering a terrain network consists in grouping together in modules vertices that belong to the same densely connected region of the graph (property p_3), while keeping separate vertices that do not (property p_1). The difference with a classification task is that the number of groups is not known in advance.

Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be a graph:

−**Module:** A *module* γ of G is a non-empty subset of the graph's vertices: $\gamma \neq \varnothing$ and $\gamma \subset V$;

−**Clustering:** A *clustering* Γ of G is a set of modules of G such that $\bigcup_{\gamma \in \Gamma} \gamma = V$;

−**Partitional clustering:** If $\forall \gamma_i, \gamma_j \in \Gamma,\ (i \neq j) \Rightarrow (\gamma_i \cap \gamma_j = \varnothing)$, then Γ is a *partitional clustering* of G, where modules of G are not allowed to overlap. Given such a Γ we can define an equivalence relation $\overset{\Gamma}{\sim}$ on the set of vertices: $\forall u, v \in V,\ (u \overset{\Gamma}{\sim} v) \Leftrightarrow$ ($\exists \gamma \in \Gamma$ such that $u \in \gamma$ and $v \in \gamma$);

−**Clustering quality function:** A clustering quality function $Q(G, \Gamma)$ is an $\mathbb{R}$-valued function whose goal is to measure the adequacy of the modules with the densely connected regions of terrain networks (property p_3).

In order to establish a good partitional clustering for a graph $G = (V, E)$, given a clustering quality function Q, it would in theory be sufficient to build all the possible partitionings of the set of vertices V, and to pick a partitioning Γ such that $Q(G, \Gamma)$ is optimal. This method is however obviously impractical, since the number of partionings of a set of size $n = |V|$ is equal to the n^{th} Bell number, a sequence known to grow exponentially [Knuth, 1968].

Many graph clustering methods therefore consist in defining a heuristic that can find in a reasonable amount of time a clustering Γ that tentatively optimises $Q(G, \Gamma)$ for a given clustering quality function Q. Several partitional clustering methods, such as *Louvain* [Blondel et al., 2008], use the $Modularity$ quality function suggested in 2004 by Newman and Girvan [Newman and Girvan, 2004].

In section 2 we present the $Modularity$ quality function, describing its limits in section 2.1. To avoid these limits, section 3 revisits the definition of $Modularity$ to introduce a mesoscopic scale, with a vertices mesoscopic closeness measure $Confluence$, that we define in section 3.1. We then compare the clusterings which maximize this new quality function with the ones that maximize $Modularity$, on a few small toy graphs, in section 3.3.

We then describe, in section 4, a new heuristic to optimize this new quality function on bigger graphs, comparing in section 4.2 the results with those obtained using two of the most used state of the art heuristics: *Louvain* [Blondel et al., 2008] which tries to maximize $Modularity$, and $Infomap$, among the most elegant and efficient heuritics, which tries to maximize the quality function described in 2008 by Rosvall and Bergstrom [Rosvall and Bergstrom, 2008].

Finally, in section 5, we describe a way to control the size of the modules and conclude in section 6.

2 Modularity

The modularity of a partitional clustering for a graph $G = (V, E)$ with $m = |E|$ edges is equal to the difference between the proportion of links internal to modules of the clustering, and the same quantity in a null model, where no community structure is expected. The null model is a random graph G_{Null} with the same number of vertices and edges, as well as the same distribution of degrees as G, where the probability of having an edge between two vertices x and y is equal to $\frac{d_G(x).d_G(y)}{2m}$, with $d_G(i) = |\{v \in V / \{i, v\} \in E\}|$ is the degree of vertex i in G.

Let $G = (V, E)$ be a graph with m edges and Γ a partitioning of V. The modularity of Γ can be defined as follows.

$$Modularity(G,\Gamma) = \frac{1}{2m}\sum_{\gamma\in\Gamma}\sum_{i,j\in\gamma} Pedge(G,i,j) - Pedge(G_{Null},i,j) \tag{1}$$

Where $Pedge(G,x,y)$ is a symmetrical vertex closeness measure equal to the probability of $\{x,y\}$ being an edge of G, that is:

$$Pedge(G,i,j) = \begin{cases} 1 \text{ if } \{i,j\} \in E, \\ 0 \text{ otberwise.} \end{cases} \tag{2}$$

$$Pedge(G_{Null},i,j) = \frac{d_G(i).d_G(j)}{2m} \tag{3}$$

In equation 1, the first term $\frac{1}{2m}$ is purely conventional, so that modularity values all live in the $[-1,1]$ interval, but plays no role when maximizing modularity, since it is constant for a given graph G.

We then define Q_{Pedge} as Newman and Girvan's quality function, to be maximized:

$$\begin{aligned} Q_{Pedge}(G,\Gamma) &= \sum_{\gamma\in\Gamma}\sum_{i,j\in\gamma} Pedge(G,i,j) - Pedge(G_{Null},i,j) \\ &= \sum_{\gamma\in\Gamma}\sum_{i,j\in\gamma} \begin{cases} 1 - \frac{d_G(i).d_G(j)}{2m} \text{ if } \{i,j\} \in E, \\ -\frac{d_G(i).d_G(j)}{2m} \text{ otherwise.} \end{cases} \end{aligned} \tag{4}$$

A good partitional clustering Γ as per 4 is one that groups in the same module vertices that are linked (especially ones with low degrees, but also to a lesser extent ones with high degrees), while avoiding as much as possible the grouping of non-linked vertices (especially ones with high degrees, but to a lesser extent ones with low degrees).

2.1 Limits of Modularity

Several authors [Fortunato and Barthelemy, 2006, Kumpula et al., 2007] showed that optimizing *Modularity* leads to merging small modules into larger ones, even when those small modules are well defined and weakly connected to one another. To address this problem, some authors [Reichardt and Bornholdt, 2006, Arenas et al., 2008] defined multiresolution variants of *Modularity*, adding a resolution parameter to control module sizes.

For instance [Reichardt and Bornholdt, 2006] introduces a parameter $\lambda \in \mathbb{R}$ in equation 4:

$$Q_\lambda = \sum_{\gamma\in\Gamma}\sum_{i,j\in\gamma} \begin{cases} 1 - \lambda.\frac{d_G(i).d_G(j)}{2m} \text{ if } \{i,j\} \in E, \\ -\lambda.\frac{d_G(i).d_G(j)}{2m} \text{ otherwise.} \end{cases} \tag{5}$$

where λ is a resolution parameter: the higher λ is, the smaller the modules get (high resolution).

However, in [Lancichinetti and Fortunato, 2011], the authors show that *"... multiresolution Modularity suffers from two opposite coexisting problems: the tendency to merge small subgraphs, which dominates when the resolution is low; the tendency to split large subgraphs, which dominates when the resolution is high. In benchmark networks with heterogeneous distributions of cluster sizes, the simultaneous elimination of both biases is not possible and multiresolution Modularity is not capable to recover the planted community structure, not even when it is pronounced and easily detectable by other methods, for any value of the resolution parameter. This holds for other multiresolution techniques and it is likely to be a general problem of methods based on global optimization.*

... real networks are characterized by the coexistence of clusters of very different sizes, whose distributions are quite well described by power laws [Clauset et al., 2004, Radicchi et al., 2004]. Therefore there is no characteristic cluster size and tuning a resolution parameter may not help."

3 Avoiding *Modularity* limits by introducing a mesoscopic scale

In equation 4, with regards to a graph G:

- $Pedge(G,i,j) = \begin{cases} 1 \text{ if } \{i,j\} \in E, \\ 0 \text{ otherwise.} \end{cases}$ is a **local** (microscopic) vertices closeness measure relative to G;
- $Pedge(G_{Null},i,j) = \frac{d_G(i).d_G(j)}{2m}$ is a **global** (macroscopic) vertices closeness measure relative to G.

To avoid limits described in section 2.1, we introduce in equation 4 an intermediate **mesoscopic**[1] vertices closeness measure relative to G: $Confluence(G,i,j)$ that we define below.

3.1 Confluence, a vertices mesoscopic closeness measure

If $G = (V,E)$ is a reflexive[2] and undirected graph, let us imagine a walker wandering on the graph G: at time $t \in \mathbb{N}$, the walker is on one vertex $i \in V$; at time $t+1$, the walker can reach any neighbouring vertex of i, with uniform probability. This process is called a simple random walk [Bollobas, 2002]. It can be defined by a Markov chain on V with an

[1] A **meso**scopic scale is an intermediate scale between a local **micro**scopic scale and a global **macro**scopic scale.

[2] i.e. each vertex is connected to itself. If such self-loops do not exist in the data, they may be added without loss of information.

$n \times n$ transition matrix $[G]$:

$$[G] = (g_{i,j})_{i,j \in V} \quad \text{with} \quad g_{i,j} = \begin{cases} \dfrac{1}{d_G(i)} \text{ if } \{i,j\} \in E, \\ 0 \text{ otherwise.} \end{cases} \tag{6}$$

Since G is reflexive, each vertex has at least one neighbour (itself) and $[G]$ is therefore well defined. Furthermore, by construction, $[G]$ is a stochastic matrix: $\forall i \in V, \sum_{j \in V} g_{i,j} = 1$. The probability $P_G^t(i \rightsquigarrow j)$ of a walker starting on vertex i and reaching vertex j after t steps is:

$$P_G^t(i \rightsquigarrow j) = ([G]^t)_{i,j} \tag{7}$$

Propsition 1 *Let $G = (V, E)$ be a reflexive graph with m edges, and $G_{null} = (V, E_{null})$ its null model such that the probability of the existence of a link between two vertices i and j is $e_{i,j} = \frac{d_G(i).d_G(j)}{2m}$.*

$$\forall t \in \mathbb{N}^*, \ \forall i, j \in V, \ P_{G_{null}}^t(i \rightsquigarrow j) = \frac{d_G(j)}{2m} \tag{8}$$

Proof by induction on t :

(a) True for $\mathbf{t = 1}$:

$$\forall i, j \in V, P_{G_{null}}^1(i \rightsquigarrow j) = e_{i,j}.\frac{1}{d_G(i)} = \frac{d_G(i).d_G(j)}{2m}.\frac{1}{d_G(i)} = \frac{d_G(j)}{2m}$$

(b) If true for $\mathbf{t}$ then true for $\mathbf{t+1}$:

$$\forall i, j \in V, P_{G_{null}}^{t+1}(i \rightsquigarrow j) = \sum_{k \in V} \left(P_{G_{null}}^t(i \rightsquigarrow k).P_{G_{null}}^1(k \rightsquigarrow j) \right)$$

$$= \sum_{k \in V} \left(P_{G_{null}}^t(i \rightsquigarrow k).\frac{d_G(j)}{2m} \right) = \frac{d_G(j)}{2m}.\sum_{k \in V} P_{G_{null}}^t(i \rightsquigarrow k)$$

$$= \frac{d_G(j)}{2m}.\sum_{k \in V} \frac{d_G(k)}{2m} = \frac{d_G(j)}{2m}$$

(a) & (b) $\Rightarrow$ 8

■

On a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ the trajectory of a random walker is completely governed by the topology of the graph in the vicinity of the starting node: after t steps, any vertex j located at a distance of t links or less can be reached. The probability of this event depends on the number of paths between i and j, and on the structure of the graph around the intermediary vertices along those paths. The more short paths exist between vertices i and j, the higher the probability $P_G^t(i \rightsquigarrow j)$ of reaching j from i.

On the graph $\mathbf{G_{null}}$ the trajectory of a random walker is only governed by the degrees of the vertices, and no longer at all by the topology of the graph in the vicinity of the starting node.

We want to consider as "close" each pair of vertices $\{i, j\}$ having a probability of reaching j from i after a short random walk in G, greater than the probability of reaching j from i in G_{null}. We therefore define the t-confluence $Conf^t(G, i, j)$ between two vertices i, j on a graph G as follows:

$$Conf^t(G,i,j) = \begin{cases} 0 \text{ if } i = j, \\ \frac{P^t_G(i \rightsquigarrow j) - P^t_{G_{null}}(i \rightsquigarrow j)}{P^t_G(i \rightsquigarrow j) + P^t_{G_{null}}(i \rightsquigarrow j)} = \frac{P^t_G(i \rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^t_G(i \rightsquigarrow j) + \frac{d_G(j)}{2m}} \text{ otherwise.} \end{cases} \tag{9}$$

Propsition 2 *Let $G = (V, E)$ be a reflexive graph with m edges, and G_{null} its null model such that the probability of the existence of a link between two vertices i and j is $e_{i,j} = \frac{d_G(i).d_G(j)}{2m}$.*

$$\forall t \in \mathbb{N}^*,\ \forall i, j \in V,\ Conf^t(G_{Null}, i, j) = 0 \tag{10}$$

Proof :
If $i = j$, the result follows directly from definition 9.
If $i \neq j$, $Conf^t(G_{Null}, i, j) = \frac{P^t_{G_{Null}}(i \rightsquigarrow j) - \frac{d_{G_{Null}}(j)}{2m}}{P^t_{G_{Null}}(i \rightsquigarrow j) + \frac{d_{G_{Null}}(j)}{2m}}$ (by definition 9)

$= \frac{P^t_{G_{Null}}(i \rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^t_{G_{Null}}(i \rightsquigarrow j) + \frac{d_G(j)}{2m}}$ (by definition of G_{Null})

$= \frac{\frac{d_G(j)}{2m} - \frac{d_G(j)}{2m}}{\frac{d_G(j)}{2m} + \frac{d_G(j)}{2m}}$ (by proposition 1)

$= 0$

■

To prove that $Conf^t(G, \cdot, \cdot)$ is symmetric, we first need to prove proposition 3.

Propsition 3 *Let $G = (V, E)$ be a reflexive graph.*

$$\forall t \in \mathbb{N}^*,\ \forall i, j \in V, P^t_G(i \rightsquigarrow j) = \frac{d_G(j)}{d_G(i)}.P^t_G(j \rightsquigarrow i) \tag{11}$$

Proof by induction on t :

(a) True for t = 1:

$\forall i, j \in V$,
if $\{i, j\} \notin E$, then $P_G^1(i \rightsquigarrow j) = 0$ and $P_G^1(j \rightsquigarrow i) = 0$, therefore $P_G^1(i \rightsquigarrow j) = \frac{d_G(j)}{d_G(i)}.P_G^1(j \rightsquigarrow i) = 0$
otherwise $P_G^1(i \rightsquigarrow j) = \frac{1}{d_G(i)} = \frac{d_G(j)}{d_G(i)}.\frac{1}{d_G(j)} = \frac{d_G(j)}{d_G(i)}.P_G^1(j \rightsquigarrow i)$

(b) If true for t then true for t + 1:

$\forall i, j \in V, P_G^{t+1}(i \rightsquigarrow j) = \sum_{k \in V} \left(P_G^t(i \rightsquigarrow k).P_G^1(k \rightsquigarrow j) \right)$

$= \sum_{k \in V} \left(P_G^t(k \rightsquigarrow i).\frac{d_G(k)}{d_G(i)}.P_G^1(k \rightsquigarrow j) \right) = \sum_{k \in V} \left(P_G^t(k \rightsquigarrow i).\frac{d_G(k)}{d_G(i)}.P_G^1(j \rightsquigarrow k).\frac{d_G(j)}{d_G(k)} \right)$

$= \sum_{k \in V} \left(P_G^t(k \rightsquigarrow i).P_G^1(j \rightsquigarrow k).\frac{d_G(j)}{d_G(i)} \right) = \frac{d_G(j)}{d_G(i)} \sum_{k \in V} \left(P_G^1(j \rightsquigarrow k).P_G^t(k \rightsquigarrow i) \right)$

$= \frac{d_G(j)}{d_G(i)}.P_G^{t+1}(j \rightsquigarrow i)$

(a) & (b) ⇒ 11

■

Propsition 4 *Let $G = (V, E)$ be a reflexive graph.*

$$\forall t \in \mathbb{N}^*, \ \forall i, j \in V, Conf^t(G, i, j) = Conf^t(G, j, i) \tag{12}$$

Proof :

If i=j : it follows directly from definition 9.

If $i \neq j : Conf^t(G, i, j) = \frac{P_G^t(i \rightsquigarrow j) - P_{G_{null}}^t(i \rightsquigarrow j)}{P_G^t(i \rightsquigarrow j) + P_{G_{null}}^t(i \rightsquigarrow j)} = \frac{P_G^t(i \rightsquigarrow j) - \frac{d_G(j)}{2m}}{P_G^t(i \rightsquigarrow j) + \frac{d_G(j)}{2m}} = \frac{\frac{d_G(j)}{d_G(i)}.P_G^t(j \rightsquigarrow i) - \frac{d_G(j)}{2m}}{\frac{d_G(j)}{d_G(i)}.P_G^t(j \rightsquigarrow i) + \frac{d_G(j)}{2m}}$

$= \frac{(\frac{d_G(j)}{d_G(i)}.P_G^t(j \rightsquigarrow i) - \frac{d_G(j)}{2m}).\frac{d_G(i)}{d_G(j)}}{(\frac{d_G(j)}{d_G(i)}.P_G^t(j \rightsquigarrow i) + \frac{d_G(j)}{2m}).\frac{d_G(i)}{d_G(j)}} = \frac{P_G^t(j \rightsquigarrow i) - \frac{d_G(i)}{2m}}{P_G^t(j \rightsquigarrow i) + \frac{d_G(i)}{2m}} = \frac{P_G^t(j \rightsquigarrow i) - P_{G_{null}}^t(j \rightsquigarrow i)}{P_G^t(j \rightsquigarrow i) + P_{G_{null}}^t(j \rightsquigarrow i)}$

$= Conf^t(G, j, i)$

■

Confluence actually defines an infinity of symmetrical vertex closeness measures, one for each random walk length t. For clarity, in the rest of this paper, we set $t = 3$ and define $Conf(G, i, j) = Conf^3(G, i, j)$.

Most terrain networks exhibit the properties p_2 (short paths) and p_3 (high clustering rate). With a classic distance such as *the shortest path between two vertices*, all vertices would be close to each other in a terrain network (because of property p_2). On the contrary, $Confluence$ allows us to identify vertices living in the same higher density zones of G (property p_3):

If i, j are in the same high local density region:

$$P^3_G(i \rightsquigarrow j) > P^3_{G_{null}}(i \rightsquigarrow j), \text{ thus } Conf(G, i, j) > 0 \tag{13}$$

If i, j are in two distinct high local density regions:

$$P^3_G(i \rightsquigarrow j) < P^3_{G_{null}}(i \rightsquigarrow j), \text{ thus } Conf(G, i, j) < 0 \tag{14}$$

3.2 Introducing a mesoscopic scale in Modularity

To avoid the limits of $Modularity$ described in section 2.1, we propose Q_{Conf}, a new clustering quality function, which introduces a mesoscopic scale into $Modularity$ through $Confluence$:

$Q_{Conf}(G, \Gamma) = \sum_{\gamma \in \Gamma} \sum_{i \neq j \in \gamma} Conf(G, i, j) + \Big(Pedge(G, i, j) - Pedge(G_{null}, i, j) \Big)$

By the definitions of $Conf$ (equation 9) and $Pedge$ (equations 2 and 3) :

$$Q_{Conf}(G, \Gamma) = \sum_{\gamma \in \Gamma} \sum_{i \neq j \in \gamma} \frac{P^3_G(i \rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^3_G(i \rightsquigarrow j) + \frac{d_G(j)}{2m}} + \begin{cases} 1 - \frac{d_G(i).d_G(j)}{2m} \text{ if } \{i, j\} \in E, \\ -\frac{d_G(i).d_G(j)}{2m} \text{ otherwise.} \end{cases}$$

Leading us to the following definition of Q_{Conf} :

$$Q_{Conf}(G, \Gamma) = \sum_{\gamma \in \Gamma} \sum_{i \neq j \in \gamma} \begin{cases} \left(\frac{P^3_G(i \rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^3_G(i \rightsquigarrow j) + \frac{d_G(j)}{2m}} \right) + \left(1 - \frac{d_G(i).d_G(j)}{2m} \right) \text{ if } \{i, j\} \in E, \\ \left(\frac{P^3_G(i \rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^3_G(i \rightsquigarrow j) + \frac{d_G(j)}{2m}} \right) - \left(\frac{d_G(i).d_G(j)}{2m} \right) \text{ otherwise.} \end{cases} \tag{15}$$

To make it easier to interpret Q_{Conf}, we define the $GooDness$ of a clustering Γ for a graph $G = (V, E)$:

$$GooDness(G, \Gamma) = \frac{1}{2|E|}.Q_{Conf}(G, \Gamma) \tag{16}$$

and the $ConFness$ of a set of vertices $\gamma \in \Gamma$ for a graph $G = (V, E)$:

$$ConFness(G, \gamma) = \frac{1}{|\gamma|.(|\gamma| - 1)} \sum_{i \neq j \in \gamma} Confluence(G, i, j) \tag{17}$$

3.3 Optimality for $\mathbf{Q_{Pedge}}$ versus optimality for $\mathbf{Q_{Conf}}$

A partitional clustering $\mathbf{\Delta}$ **is optimal** for a quality function Q **iff**: For all partitioning Γ of V, $Q\big(G,\Delta)\big) \geqq Q\big(G,\Gamma)\big)$. Computing a Δ that maximizes $Q_{Pedge}(G,\Delta)$ is $\mathcal{NP}-complete$ [Brandes et al., 2008], and the same holds for computing a clustering that maximizes Q_{Conf}.

However, when the number of vertices of a graph $G=(V,E)$ is small, the problem of maximizing the modularity can be turned into a reasonably tractable Integer Linear Program (see [Brandes et al., 2008]): We define n^2 decision variables $X_{ij} \in \{0,1\}$, one for each pair of vertices $\{i,j\} \in V$. The key idea is that we can build an equivalence relation on V ($i \sim j$ iff $X_{ij}=1$) and therefore a partitioning of V. To guarantee that the decision variables give rise to an equivalence relation, they must satisfy the following constraints:

Reflexivity: $\forall i \in V, X_{ii}=1$;

Symmetry: $\forall i,j \in V : X_{ij}=X_{ji}$;

$$\textbf{Transitivity: } \forall i,j,k \in V : \begin{cases} \forall i,j,k \in V : X_{ij}+X_{jk}-2.X_{ik} \leq 1; \\ \forall i,j,k \in V : X_{ik}+X_{ij}-2.X_{jk} \leq 1; \\ \forall i,j,k \in V : X_{jk}+X_{ik}-2.X_{ij} \leq 1. \end{cases}$$

With the following objective functions to maximize:

$$\begin{aligned} &\textbf{For } \mathbf{Q_{Pedge}} : \sum_{i,j \in V} X_{ij}.\Big(Pedge(G,i,j) - Pedge(G_{Null},i,j)\Big) \\ &\textbf{For } \mathbf{Q_{Conf}} \ : \sum_{i \neq j \in V} X_{ij}.\Big(Conf(G,i,j) + Pedge(G,i,j) - Pedge(G_{null},i,j)\Big) \end{aligned} \tag{18}$$

On four small atificial graphs, G^1_{toy}, ... G^4_{toy}, we compare optimal clusterings Δ^G_{Pedge} and Δ^G_{Conf} respectively computed for Q_{Pedge} and Q_{Conf}, with results illustrated in Figure 1.

The optimal clusterings for Q_{Conf} do not necessarily have a higher resolution than with Q_{Pedge}, they can even have a lower resolution[3]:

• **Fig. 6(b):** $\Delta^{G^1_{toy}}_{Conf}\ \langle 0.20,\ 0.40\rangle$ versus $\Delta^{G^1_{toy}}_{Pedge}\ \langle 0.21,\ 0.38\rangle$:

$\mathbf{\Delta^{G^1_{toy}}_{Conf}}$ **has a higer resolution than** $\mathbf{\Delta^{G^1_{toy}}_{Pedge}}$, with $\delta^1_{Pedge} = \delta^2_{Conf} \cup \delta^3_{Conf}$:

- $\Delta^{G^1_{toy}}_{Conf} = \Big\{\delta^1_{Conf}{=}\{0,4,5,6\}\ \langle 0.10\rangle,\ \delta^2_{Conf}{=}\{1,2,3\}\ \langle 0.18\rangle,\ \delta^3_{Conf}{=}\{7,8\}\ \langle 0.18\rangle\Big\}$;
- $\Delta^{G^1_{toy}}_{Pedge} = \Big\{\delta^1_{Pedge}{=}\{1,2,3,7,8\}\ \langle 0.04\rangle,\ \delta^2_{Pedge}{=}\{0,4,5,6\}\ \langle 0.10\rangle\Big\}$.

[3] Where $\langle \_\ ,\ \_\ \rangle$ for the $\langle Modularity,\ GooDness\rangle$ of the clustering, and $\langle \_\rangle$ for the $\langle ConFness\rangle$ of the modules

- **Fig 1(b):** $\Delta^{G^2_{toy}}_{Conf}\ \langle 0.37,\ 0.75\rangle$ versus $\Delta^{G^2_{toy}}_{Pedge}\ \langle 0.38,\ 0.70\rangle$:

  $\mathbf{\Delta^{G^2_{toy}}_{Conf}}$ **has a lower resolution than** $\mathbf{\Delta^{G^2_{toy}}_{Pedge}}$, with $\delta^1_{Conf} = \delta^1_{Pedge} \cup \delta^2_{Pedge}$:

  - $\Delta^{G^2_{toy}}_{Conf} = \Big\{\delta^1_{Conf}{=}\{0,1,2,3,4,5,6,7,8,9\}\ \langle 0.13\rangle,$ $\delta^2_{Conf}{=}\{13,14,15,16\}\ \langle 0.59\rangle,$ $\delta^3_{Conf}{=}\{10,11,12\}\ \langle 0.67\rangle\Big\}$;
  - $\Delta^{G^2_{toy}}_{Pedge} = \Big\{\delta^1_{Pedge}{=}\{0,1,3,4,8\}\ \langle 0.17\rangle,\ \delta^2_{Pedge}{=}\{2,5,6,7,9\}\ \langle 0.22\rangle,$ $\delta^3_{Pedge}{=}\{13,14,15,16\}\ \langle 0.59\rangle,\ \delta^4_{Pedge}{=}\{10,11,12\}\ \langle 0.67\rangle\Big\}$.

- **Fig 1(c):** $\Delta^{G^3_{toy}}_{Conf}\ \langle 0.30,\ 0.62\rangle$ versus $\Delta^{G^3_{toy}}_{Pedge}\ \langle 0.33,\ 0.61\rangle$:

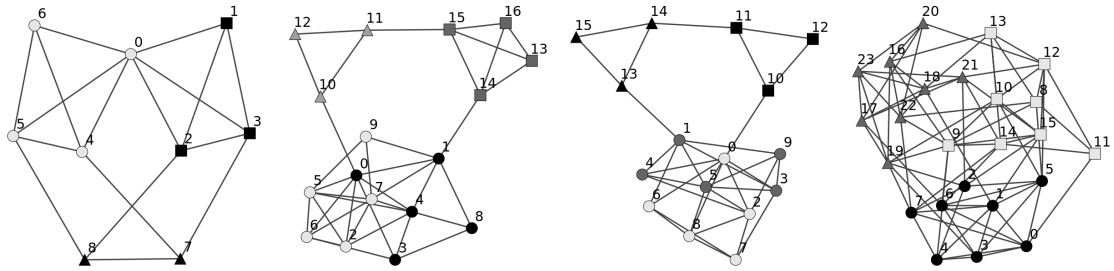
  $\mathbf{\Delta^{G^3_{toy}}_{Conf}}$ **is different from** $\mathbf{\Delta^{G^3_{toy}}_{Pedge}}$, with $\delta^1_{Conf} = \delta^2_{Pedge} \cup \delta^3_{Pedge}$ and $\delta^1_{Pedge} = \delta^2_{Conf} \cup \delta^3_{Conf}$:

  - $\Delta^{G^3_{toy}}_{Conf} = \Big\{\delta^1_{Conf}{=}\{0,1,2,3,4,5,6,7,8,9\}\ \langle 0.10\rangle,\ \delta^2_{Conf}{=}\{10,11,12\}\ \langle 0.65\rangle,$ $\delta^3_{Conf}{=}\{13,14,15\}\ \langle 0.66\rangle\Big\}$;
  - $\Delta^{G^3_{toy}}_{Pedge} = \Big\{\delta^1_{Pedge}{=}\{10,11,12,13,14,15\}\ \langle 0.26\rangle,\ \delta^2_{Pedge}{=}\{0,2,6,7,8\}\ \langle 0.17\rangle,$ $\delta^3_{Pedge}{=}\{1,3,4,5,9\}\ \langle 0.15\rangle\Big\}$.

- **Fig 1(d):** $\Delta^{G^4_{toy}}_{Conf}\ \langle 0.32,\ 0.53\rangle$ versus $\Delta^{G^4_{toy}}_{Pedge}\ \langle 0.32,\ 0.53\rangle$:

  $\mathbf{\Delta^{G^4_{toy}}_{Conf}}$ **is equal to** $\mathbf{\Delta^{G^4_{toy}}_{Pedge}}$:

  - $\Delta^{G^4_{toy}}_{Conf} = \Delta^{G^4_{toy}}_{Pedge} = \Big\{\delta^1{=}\{0,1,2,3,4,5,6,7\}\ \langle 0.19\rangle,\ \delta^2{=}\{8,9,10,11,12,13,14,15\}\ \langle 0.12\rangle,$ $\delta^3{=}\{16,17,18,19,20,21,22,23\}\ \langle 0.19\rangle\Big\}$.



(a) G^1_{toy} (b) G^2_{toy} (c) G^3_{toy} (d) G^4_{toy}

Figure 1: **Optimality with respect to $\mathbf{Q_{Pedge}}$ versus optimality with respect to $\mathbf{Q_{Conf}}$**. Shapes describe an optimal clustering for Q_{Conf} (if two vertices have same shape, then they are in a same module for Q_{Conf}). Colors describe an optimal clustering for Q_{Pedge} (if two vertices have same color, then they are in a same module for Q_{Pedge}).

4 A heuristic by edges confluence

We describe in this section a heuristic for tentatively maximizing Q_{Conf}. To this end, we start with a variant on $Conf$, named $Confr$, where the confluence between two adjacent vertices is computed by removing the edge between them[4]:

$$Confr(G,i,j) = \begin{cases} Conf(G,i,j) \text{ if } \{i,j\} \notin E, \\ Conf(G = (V, E - \{i,j\}), i, j) \text{ otherwise.} \end{cases} \tag{19}$$

$Confr$ gives us an ordering on the edges $\{i,j\} \in E$ of the graph. In particular, sorting the edges by decreasing $Confr(G,i,j)$ forms the basis of a new strategy for constructing the cluster's modules, $Hbec$ (*Heuristic by edges confluence*), described[5] in algorithm 1. On G^5_{toy}, the graph depicted in Figure 2, $Hbec$ yields the following clusters: $Hbec(G^5_{toys}) = \big\{\{0,1,3,4,5\},\{6,7,8,9,10\},\{2\}\big\}$. It is interesting to note how vertex 2 is left isolated in its own cluster by $Hbec$, whereas it joins the $\{6,7,8,9,10\}$ module in the optimal clustering for Q_{Conf}: $\Delta^{G^5_{toy}}_{Conf} = \big\{\{0,1,3,4,5\},\{2,6,7,8,9,10\}\big\}$.

[4] A comparative study of $Confr$ with over 80 similarity measures between vertices of a graph has been done by Emmanuel Navarro in his thesis, looking into the sensitivity of various methods to the density of graphs, paths of length 1 and the degrees of vertices, among other aspects. This thesis is available at `https://oatao.univ-toulouse.fr/12024/1/navarro.pdf`. The study shows that, among the 80 similarity measures considered, $Confr(G,i,j)$ is the only one that, even though it does not systematically regroup pairs of vertices $\{i,j\} \in E$, is correlated to the local edge density around vertices i and j, while being independent of the global density of the graphs.

[5] Different edges might happen to have the exact same $Confr$ value, making the process non-deterministic in general, because of its sensitivity on the order in which the edges with identical $Confr$ values are processed. A simple solution to this problem is to sort edges by first comparing their $Confr$ values and then using the lexicographic order on (i,j) when $Confr$ values are strictly identical.

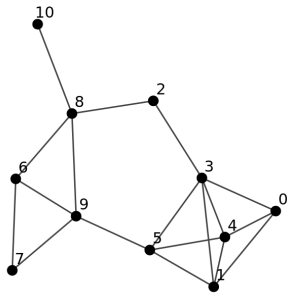


Figure 2: $\mathbf{G^5_{toys}}$

Indeed, when a vertex is linked to two or more distinct dense regions of a graph, *Hbec* tends to leave that vertex isolated due to its $Confr$-centered strategy, which does not push the said vertex towards one of those regions over the other. It is therefore desirable to add one last step to the process, to get even closer to the optimal clustering by trying to merge more modules with a greedy algorithm:

> **FastGreedy**(**Hbec**(**G**)) : we iteratively merge pairs of modules resulting from $Hbec(G)$ when the union of modules is locally optimal with respect to Q_{Conf}, stopping when merging any pair of the remaining modules would not result in an increase of Q_{Conf}.

We call **Kodex**(**G**) the complete algorithm, including the final **FastGreedy** step. We now get: $Kodex(G^5_{toys}) = \Delta^{G^5_{toy}}_{Conf}$.

4.1 Implementation

Our implementation of *Kodex* reflects the high-level description from the previous sections and is therefore split into three main parts: pre-computing the confluence of all the edges of the graph (and sorting the said list of edges in decreasing confluence order), doing a first clustering pass (*Hbec*), and the final greedy step. Let $G = (V, E)$, and $n = |V|$.

Edges confluence:

- First, construct the reflexive sparse adjacency matrix of the input graph, $Adj(G) \in \mathcal{M}_n(\mathbb{R})$, represented using the Compressed Sparse Column format, supported by the vast majority of sparse linear algebra libraries, by supplying the appropriate routine with all the $(i, j, 1)$ triplets, where $\{i, j\} \in E$ or $i = j$. That is, $Adj(G)_{i,j} = 1$ if $i, j \in E$ or $i = j$, 0 otherwise .
- Then compute the corresponding transition matrix, $T(G)$, assuming a uniform probability of reaching any neighbor from a given vertex. This amounts to setting all non-zero values of a column j to $\frac{1}{deg_G j}$. That is, $T(G)_{i,j} = \frac{1}{deg_G(j)}$ if $i, j \in E$, 0 otherwise .

Algorithm 1 *Hbec*(G) **H**euristic **b**y **e**dges **c**onfluence for optimizing Q_{Conf}

Input: $G = (V, E)$ an undirected graph
Output: C a partitional clustering of G

for $i \in V$ **do**
  $mod_i \leftarrowtail \{i\}$
  $v_i \leftarrowtail i$
$q \leftarrowtail 0$
$\Upsilon \leftarrowtail \varnothing$
While $\Upsilon \neq E$ **do**
  $\{i, j\} \leftarrowtail \underset{\{x,y\} \in E - \Upsilon}{\arg\max}\ Confr(G, x, y)$ ► **(a)**
  $\Upsilon \leftarrowtail \Upsilon \cup \{\{i, j\}\}$
  if $v_i \neq v_j$ **then** ► i and j are not in the same module
    $q_{candidate} \leftarrowtail q + 2.\sum_{x \in m_i} \sum_{y \in m_j} Conf(G, x, y) + Pedge(G, x, y) - Pedge(G_{null}, x, y)$ ► **(b)**
    if $q \leqslant q_{candidate}$ **then**
      $q \leftarrowtail q_{candidate}$
      $mod_i \leftarrowtail mod_i \cup mod_j$
      for $y \in mod_j$ **do**
        $v_y \leftarrowtail v_i$
      $m_j \leftarrowtail \varnothing$
$C \leftarrowtail \varnothing$
for $i \in V$ **do**
  if $mod_i \neq \varnothing$
    $C \leftarrowtail C \cup \{mod_i\}$
return C

- Then compute the proxemy matrix, $P(G) = T(G)^3$.
- Next, we iterate over the list of edges $\{a, b\} \in E$, constructing another list:
  - Define $T(G - \{a, b\})$ to be the same as $T(G)$ except for the entries at (a, b) and (b, a) that should be set to 0, to emulate the removal of the link between vertices a and b, and an update to the other entries of the two relevant columns to reflect a decrease by 1 in the degree of vertices a and b, to account for the freshly removed link.
    That is,
    $$T(G - \{a, b\})_{i,j} = \begin{cases} 0 \text{ if } \{i, j\} = \{a, b\} \\ \frac{1}{deg_G(j)} \text{ if } j = a \text{ or } j = b \text{ but } i \notin \{a, b\} \\ T(G)_{i,j} \text{ otherwise} \end{cases}$$
  - Define $e_b \in \mathbb{R}^n$ as $(e_b)_i = 1$ if $i = b$, 0 otherwise , ideally using a sparse representation.
  - Multiply $T(G - \{a, b\})$ and e_b, and multiply the result by $T(G - \{a, b\})$, and again, as many times as the length of the random walk is, which has been 3 in most of this paper. That is, for $t = 3$, we compute: $T(G - \{a, b\})T(G - \{a, b\})T(G - \{a, b\})e_b$. We call the resulting vector $w \in \mathbb{R}^n$
  - Finally, compute $conf_{a,b} = \frac{w_a - \frac{deg_G(a)-1}{nnz(Adj(G))-2}}{w_a + \frac{deg_G(a)-1}{nnz(Adj(G))-2}}$ where nnz is a function returning the number of non-zero entries in a sparse matrix, typically provided by sparse linear algebra libraries. Return $(a, b, conf_{a,b})$.
- Sort the list of $(a, b, conf_{a,b})$ according to $conf_{a,b}$, in decreasing order, breaking equalities by discriminating on the lexicographic ordering on a and then b.

Hbec

- We create a clustering with all vertices isolated in their own 1-point cluster. Clusters are represented with associative maps from cluster identifiers to sets of vertex identifiers, i.e $clust[i] = \{a, b, c\}$ means that vertices a, b, c are in cluster number i, while maintaining another associative map from vertex identifiers to cluster identifier, i.e $index[i] = c$ means that vertex i belongs to cluster c. We additionally maintain a mapping from cluster identifiers to scores, $score[i]$, to avoid save some computations that we will keep reusing, instead of performing them from scratch every time.
- We set the score of our clustering, $clust_score$ to 0 initially.
- We iterate over the sorted list of edges from the previous phase. When processing entry $(a, b, conf_{a,b})$:

– We compute the clustering quality improvement we would get by merging clusters $index[a]$ and $index[b]$:

$$score(index[a] \text{ and } index[b]) = \sum_{i \in clust[index[a]], j \in clust[index[b]]} f(i,j) \tag{20}$$

where f is defined as:

$$f(i,j) = \frac{P(G)_{i,j} - \frac{deg_G(i)}{nnz(Adj(G)}}{P(G)_{i,j} + \frac{deg_G(i)}{nnz(Adj(G)}} + Adj(G)_{i,j} - \frac{(deg_G(i)-1)(deg_G(j)-1)}{nnz(Adj(G)) - n}$$

– If $score(index[a] \text{ and } index[b]) - score[index[a]] - score[index[b]] > 0$, we merge $index[b]$ into $index[a]$:

* $clust_score := clust_score + score(index[a] \text{ and } index[b]) - score[index[a]] - score[index[b]]$
* $clust[index[a]] := clust[index[a]] \cup clust[index[b]]$
* $score[index[a]] := score(index[a] \text{ and } index[b])$
* For all $i \in clust[index[b]]$, we set $index[i] := index[a]$.

Otherwise, we move on to the next edge.

- Once all the edges have been processed, *clust* describes a partitional clustering of G.

Final greedy step

- For all pairs of non-empty clusters resulting from *Hbec*, compute the sum described in equation 20, using an associative container of some sort to record the mapping between the pair of cluster identifiers and the resulting score.
- If there is a pair of clusters for which the sum yields a strictly positive number (i.e a pair that would improve the quality of the clustering, if merged), merge them, updating the scores computed in the first step to account for the merge, wherever one of those clusters is involved.
- Repeat until no such pair of clusters exists anymore.

4.2 Tests

We compare three heuristics for computing a partitional clustering:

- **Louvain** : The *Louvain* method [Blondel et al., 2008] is the state of the art heuristic that tentatively maximizes $Modularity$;

- **Infomap** : The $Infomap$ method is a heuristic for tentatively maximizing the quality function described in 2008 by Rosvall and Bergstrom [Rosvall and Bergstrom, 2008]. This quality function is based on the minimum description length principle [Grünwald, 2007]. It consists in measuring the compression ratio that a given partitioning Γ provides for describing the trajectory of a random walk on a graph. The trajectory description happens on two levels. When the walker enters a module, we write down its name. We then write the vertices that the walker visits, with a notation local to the module, so that an identical short name may be used for different vertices from different modules. A concise description of the trajectory, with a good compression ratio, is therefore possible when the modules of Γ are such that the walker tends to stay in them, which corresponds to the idea that the walker is "captured" when it enters a good module, which is supposed to be a densely linked region that is only weakly connected to other modules. The quality[6] of a partitional clustering is the compression ratio that Γ provides for describing the trajectory of a random walker on G;
- **Kodex** : The heuristic described in section 4 that tentatively maximizes Q_{Conf}.

4.2.1 Louvain, Infomap and Kodex on four G_{toys} graphs

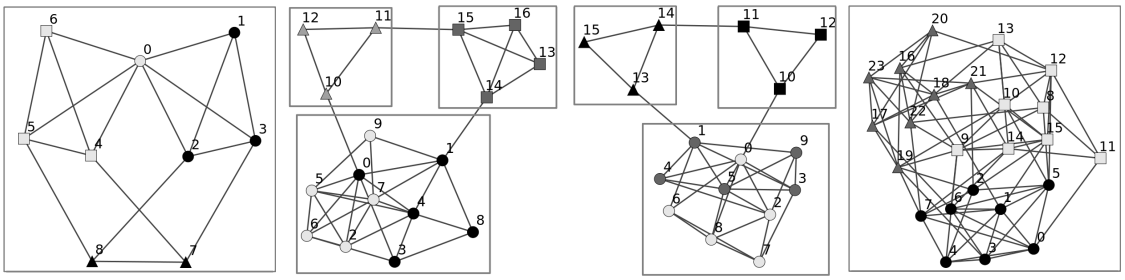
We start by testing $Louvain$, $Infomap$ and $Kodex$ on the four G_{toys} graphs of section 3.3. The results are summarized in Figure 3.

On these four G_{toys} graphs, $Louvain$ finds the optimal modules with respect to Q_{Pedge} while $Kodex$ finds the optimal modules with respect to Q_{Conf}. On G^2_{toy} and G^3_{toy}, $Infomap$ and $Kodex$ agree on the clustering, both finding the optimal modules with respect to Q_{Conf}. However, on G^1_{toy} and G^4_{toy}, $Infomap$ collects all the vertices into a single module, because there is no way to compress the description of the path of a random walker on either of those graphs.

Indeed, G^4_{toy} has been artificially constructed as follow: $G^4_{toy} = (V, E)$ where V is the union of 3 sets of 8 vertices, $\gamma_1 = \{0, ..., 7\}$, $\gamma_2 = \{8, ..., 15\}$, $\gamma_3 = \{16, ..., 23\}$, and edges are randomly drawn using two different probabilities: we use a probability of 0.5 for drawing an edge between two vertices from the same γ_i, and a probability of 0.25 when the two vertices come from different sets.

Vertices from a given γ_i are expected to have as many internal links (within γ_i) as they have links to vertices from the two other sets. At every single step, the random walker is equally likely to leave a γ_i or stay within it, which effectively prevents any path compression to take place in G^4_{toy}.

[6] Note that this quality function cannot be expressed as $\sum_{\gamma\in\Gamma}\sum_{i,j\in\gamma} sim(G,i,j)$, with $sim(G,.,.)$ an $\mathbb{R}$-valued symmetric similarity measure between vertices of G. We therefore left out this quality function in our study of optimality in section 3.3, not having the ability to define the corresponding objective function to maximize in a similar fashion to what was done for Q_{Pedge} and Q_{Conf} with the formulas 18.



(a) G^1_{toy} (b) G^2_{toy} (c) G^3_{toy} (d) G^4_{toy}

Figure 3: **Clusterings with Louvain, Infomap and Kodex:** shapes describe $Kodex(G)$, colors describe $Louvain(G)$, and blocks describe $Infomap(G)$.

- **On $\mathbf{G^1_{toy}}$**: $Louvain(G^1_{toy}) = \Delta^{G^1_{toy}}_{Pedge}$, $Infomap(G^1_{toy}) = \{V\}$, $Kodex(G^1_{toy}) = \Delta^{G^1_{toy}}_{Conf}$
- **On $\mathbf{G^2_{toy}}$**: $Louvain(G^2_{toy}) = \Delta^{G^2_{toy}}_{Pedge}$, $Infomap(G^2_{toy}) = Kodex(G^2_{toy}) = \Delta^{G^2_{toy}}_{Conf}$
- **On $\mathbf{G^3_{toy}}$**: $Louvain(G^3_{toy}) = \Delta^{G^3_{toy}}_{Pedge}$, $Infomap(G^3_{toy}) = Kodex(G^3_{toy}) = \Delta^{G^3_{toy}}_{Conf}$
- **On $\mathbf{G^4_{toy}}$**: $Louvain(G^4_{toy}) = Kodex(G^4_{toy}) = \Delta^{G^4_{toy}}_{Conf} = \Delta^{G^4_{toy}}_{Pedge}$, $Infomap(G^4_{toy}) = \{V\}$

4.2.2 Louvain, Infomap and Kodex on terrain graphs

We now compare *Louvain*, *Infomap* and *Kodex* on four state-of-the-art terrain graphs.

- **$\mathbf{G_{FrSynV}}$**: The *Dicosyn* resource comes from a collaboration between IBM and the CNRS laboratory ATILF `http://www.atilf.fr/`. Starting with seven classical French dictionaries (Bailly, Benac, Du Chazaud, Guizot, Lafaye, Larousse and Robert), they extracted synonymic relationships, and the resulting graph G_{FrSyn} was then symmetrized and split by *PoS* into three graphs: G_{FrSynV} for verbs, $G_{FrSyn.N}$ for nouns and $G_{FrSyn.A}$ for adjectives.
- **$\mathbf{G_{Actors}}$:** The well known network of actors [Watts and Strogatz, 1998] has been built from the Internet Movie Database (April 1997) `http://us.imdb.com`; nodes are actors, and two actors are linked if they have played in a film together [Rossi and Ahmed, 2015] `https://networkrepository.com/actor-collaboration.php`.
- **$\mathbf{G_{DBLP}}$:** The DBLP computer science bibliography provides a comprehensive list of research papers in computer science [Leskovec and Krevl, 2014]. Two authors are connected if they have published at least one paper together `https://snap.stanford.edu/data/com-DBLP.html`.

- **G_{Amazon}:** A network was collected by crawling the Amazon website. It is based on the *Customers Who Bought This Item Also Bought* feature of the Amazon website [Leskovec and Krevl, 2014]. If a product i is frequently co-purchased with product j, the graph contains an undirected edge from i to j `https://snap.stanford.edu/data/com-Amazon.html`.

Table 1 illustrates the pedigrees[7] of these terrain graphs, and Table 2 illustrates the execution times, number of modules, and the length of the biggest module found by the different methods..

Table 1: **Pedigrees:** n and m are the number of vertices and edges, $\langle k \rangle$ is the mean degree of vertices, C is the clustering coefficient of the graph, L_{lcc} is the average shortest path length between any two nodes of the largest connected component (largest subgraph in which there exist at least one path between any two nodes) and n_{lcc} the number of vertices of this component, λ is the coefficient of the best fitting power law of the degree distribution and r^2 is the correlation coefficient of the fit, measuring how well the data is modelled by the power law.

Graph	**n**	**m**	$\langle \mathbf{k} \rangle$	**C**	$\mathbf{L_{lcc}(n_{lcc})}$	$\lambda(\mathbf{r^2})$
$\mathbf{G_{FrSynV}}$	9,147	51,423	11.24	0.14	4.20(8,993)	-1.88(0.91)
$\mathbf{G_{Actors}}$	68,684	3,007,298	87.57	0.20	3.51(68,019)	-1.49(0.90)
$\mathbf{G_{DBLP}}$	317,080	1,049,866	6.62	0.31	6.79(317,080)	-2.71(0.95)
$\mathbf{G_{Amazon}}$	334,863	925,872	5.53	0.21	11.95(334,863)	-2.81(0.93)

(TODO: Commenter les resultats de la table 2: Bruno)

[7] The power law estimation we give, $\lambda(r^2)$ is not very accurate (see for instance [Newman, 2005]). However, giving a correct estimation of the odds that a given discrete distribution is heavy-tailed is a difficult issue ([Goldstein et al., 2004, Clauset et al., 2009]), and refining the power-law estimates is beyond the scope of this paper.

Table 2: **Clusterings of four Terrain Graphs with Louvain, Kodex, Infomap:** (T) Time computation in seconds (Louvain and Infomap in C++, Kodex in Haskell (see section 4.1), (N) Number of modules, (M) Length of the biggest module and $\langle - \ ; \ - \ ; \ - \rangle$ for $\langle Louvain \ ; \ Kodex \ ; \ Infomap \rangle$

	$\mathbf{G_{FrSynV}}$	$\mathbf{G_{Actors}}$	$\mathbf{G_{DBLP}}$	$\mathbf{G_{Amazon}}$
(T)	$\langle 0 \ ; \ ?? \ ; \ 13 \rangle$	$\langle ?? \ ; \ ?? \ ; \ ?? \rangle$	$\langle 11 \ ; \ 9,639 \ ; \ 1,933 \rangle$	$\langle 6 \ ; \ 7,830 \ ; \ 1,795 \rangle$
(N)	$\langle 90 \ ; \ 671 \ ; \ 575 \rangle$	$\langle ?? \ ; \ ?? \ ; \ ?? \rangle$	$\langle 238 \ ; \ 17,437 \ ; \ 16,960 \rangle$	$\langle 234 \ ; \ 20,703 \ ; \ 17,292 \rangle$
(M)	$\langle 1,189 \ ; \ 155 \ ; \ 173 \rangle$	$\langle ???? \ ; \ ??? \ ; \ ??? \rangle$	$\langle 25,893 \ ; \ 249 \ ; \ 599 \rangle$	$\langle 12,783 \ ; \ 535 \ ; \ 347 \rangle$

5 Modulating the size of modules

In many domains such as linguistics, sociology or biology, it can be useful to look into sets of objects (documents, sentences, words, individuals, cells, ...) at different levels of granularity. In order to gain some control over the size of modules, we introduce a β parameter into Q_{Conf}, resulting in the following $Q^{\beta}_{Conf}(G,\Gamma)$ quality function:

$$Q^{\beta}_{Conf}(G,\Gamma) = \sum_{\gamma\in\Gamma}\sum_{i\neq j\in\gamma} \begin{cases} Conf(G,\ i,\ j) + \big(1 - \frac{d_G(i).d_G(j)}{2m}\big) \text{ if } \{i,j\}\in E, \\ Conf(G,\ i,\ j) - \big(\frac{d_G(i).d_G(j)}{2m}\big) - \beta * \big(1 - Conf(G,\ i,\ j)\big) \text{ otherwise.} \end{cases}$$

$$= \sum_{\gamma\in\Gamma}\sum_{i\neq j\in\gamma} \begin{cases} \frac{P^3_G(i\rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^3_G(i\rightsquigarrow j) + \frac{d_G(j)}{2m}} + \big(1 - \frac{d_G(i).d_G(j)}{2m}\big) \text{ if } \{i,j\}\in E, \\ \frac{P^3_G(i\rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^3_G(i\rightsquigarrow j) + \frac{d_G(j)}{2m}} - \big(\frac{d_G(i).d_G(j)}{2m}\big) - \beta * \big(1 - \frac{P^3_G(i\rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^3_G(i\rightsquigarrow j) + \frac{d_G(j)}{2m}}\big) \text{ otherwise.} \end{cases} \quad (21)$$

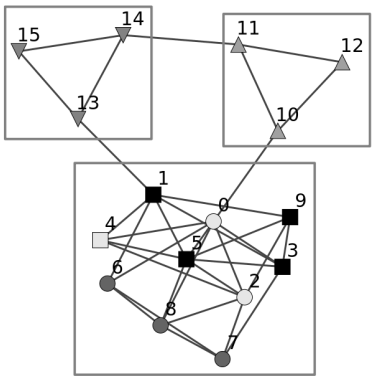
Since $P^3_G(i\rightsquigarrow j)\in[0,1]$, it follows that $\frac{P^3_G(i\rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^3_G(i\rightsquigarrow j) + \frac{d_G(j)}{2m}} \in [-1,1]$, and therefore:
if $\beta > 0$, then $-\beta * \big(1 - \frac{P^3_G(i\rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^3_G(i\rightsquigarrow j) + \frac{d_G(j)}{2m}}\big) \leqslant 0$. That is, with $\beta > 0$:

- If $\{i,j\}\in E$, then noting changes ;
- If $\{i,j\}\notin E$ and i,j are in the same high density region, then we slightly penalize the $\{i,j\}$ pair (by 13);
- If $\{i,j\}\notin E$ and i,j are in two distinct high density regions, then we strongly penalize the $\{i,j\}$ pair (by 14).

Because we must avoid as much as possible that the modules contain pairs $\{i,j\}\notin E$ (with increasing efficiency as β grows), this forces modules to be smaller, as illustrated by Figure 4 on graph G^3_{toy}. Increasing β does not simply split modules (resulting in a hierarchical approach). This can be observed when going from $\beta = 0.5$ to $\beta = 1.0$, noticing that there is no $\delta\in\Delta^{G^3_{toy}}_{Q^{0.5}_{Conf}}$ such that $\delta^2_{Q^{1.0}_{Conf}} = \{0,2,4\}\subset\delta$.

Revisiting the previous algorithm to account for the β parameter, yielding $Kodex(G,\beta)$, simply amounts to replacing line ▶(♭) in algorithm 1 by:

$$q_{candidate} \leftarrow q + 2.\sum_{x\in m_i}\sum_{y\in m_j} \begin{cases} Conf(G,x,y) + Pedge(G,x,y) - Pedge(G_{null},x,y) \text{ if } \{x,y\}\in E, \\ (1+\beta).Conf(G,x,y) + Pedge(G,x,y) - Pedge(G_{null},x,y) - \beta \text{ otherwise.} \end{cases}$$



(a) G^3_{toy}

Figure 4: **Optimal clusterings of G^3_{toy} for $\mathbf{Q}^{\beta=0.0}_{\mathbf{Conf}}$, $\mathbf{Q}^{\beta=0.5}_{\mathbf{Conf}}$, $\mathbf{Q}^{\beta=1.0}_{\mathbf{Conf}}$:** In Fig. 4(a), blocks describe $\Delta^{G^3_{toy}}_{Q^{0.0}_{Conf}}$ an optimal clustering of G^3_{toy} for $Q^{\beta=0.0}_{Conf}$, shapes describe $\Delta^{G^3_{toy}}_{Q^{0.5}_{Conf}}$ for $Q^{\beta=0.5}_{Conf}$, and colors describe $\Delta^{G^3_{toy}}_{Q^{1.0}_{Conf}}$ for $Q^{\beta=1.0}_{Conf}$.

- $\Delta^{G^3_{toy}}_{Q^{0.0}_{Conf}} = \Big\{\delta^1_{Q^{0.0}_{Conf}}=\{0,1,2,3,4,5,6,7,8,9\}\ \langle 0.10\rangle,\ \delta^2_{Q^{0.0}_{Conf}}=\{10,11,12\}\ \langle 0.65\rangle,$
  $\delta^3_{Q^{0.0}_{Conf}}=\{13,14,15\}\ \langle 0.66\rangle\Big\};$
- $\Delta^{G^3_{toy}}_{Q^{0.5}_{Conf}} = \Big\{\delta^1_{Q^{0.5}_{Conf}}=\{0,2,6,7,8\}\ \langle 0.17\rangle,\ \delta^2_{Q^{0.5}_{Conf}}=\{1,3,4,5,9\}\ \langle 0.15\rangle,\ \delta^3_{Q^{0.5}_{Conf}}=\{10,11,12\}\ \langle 0.65\rangle,$
  $\delta^4_{Q^{0.5}_{Conf}}=\{13,14,15\}\ \langle 0.66\rangle\Big\};$
- $\Delta^{G^3_{toy}}_{Q^{1.0}_{Conf}} = \Big\{\delta^1_{Q^{1.0}_{Conf}}=\{1,3,5,9\}\ \langle 0.17\rangle,\ \delta^{\mathbf{2}}_{\mathbf{Q^{1.0}_{Conf}}}=\{\mathbf{0,2,4}\}\ \langle \mathbf{0.14}\rangle,\ \delta^3_{Q^{1.0}_{Conf}}=\{6,7,8\}\ \langle 0.27\rangle,$
  $\delta^4_{Q^{1.0}_{Conf}}=\{10,11,12\}\ \langle 0.65\rangle,\ \delta^5_{Q^{1.0}_{Conf}}=\{13,14,15\}\ \langle 0.66\rangle\Big\}.$

Table 3 shows the results for $Kodex(G,\beta)$ with $\beta=\mathbf{0.0},\ \beta=\mathbf{0.5},\ \beta=\mathbf{1.0}$ on four real terrain graphs. To compare two clusterings Γ_1 and Γ_2, of a graph $G=(V,E)$ we define:

$$Precision(\Gamma_1,\Gamma_2)=\frac{\left|\left(\bigcup_{\gamma\in\Gamma_1}\{\{x,y\}\in\gamma\times\gamma\}\right)\bigcap\left(\bigcup_{\gamma\in\Gamma_2}\{\{x,y\}\in\gamma\times\gamma\}\right)\right|}{\left|\bigcup_{\gamma\in\Gamma_1}\{\{x,y\}\in\gamma\times\gamma\}\right|}$$

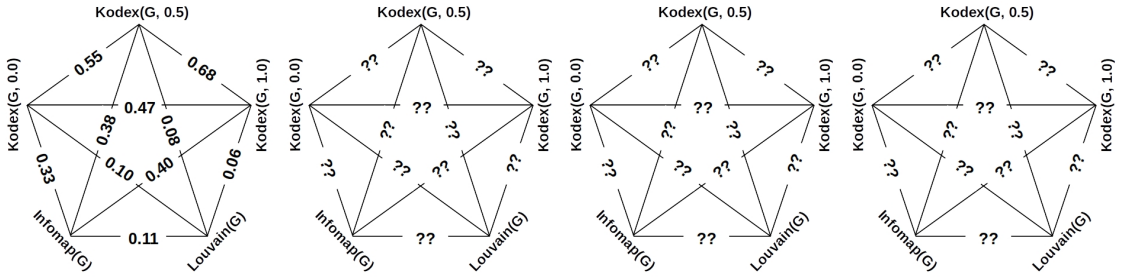
$$Recall(\Gamma_1,\Gamma_2)=\frac{\left|\left(\bigcup_{\gamma\in\Gamma_1}\{\{x,y\}\in\gamma\times\gamma\}\right)\bigcap\left(\bigcup_{\gamma\in\Gamma_2}\{\{x,y\}\in\gamma\times\gamma\}\right)\right|}{\left|\bigcup_{\gamma\in\Gamma_2}\{\{x,y\}\in\gamma\times\gamma\}\right|}$$

$$Fscore(\Gamma_1,\Gamma_2)=2.\frac{Precision(\Gamma_1,\Gamma_2).Recall(\Gamma_1,\Gamma_2)}{Precision(\Gamma_1,\Gamma_2)+Recall(\Gamma_1,\Gamma_2)}$$

Figure 5 shows $Fscore(\Gamma_1, \Gamma_2)$ between clusterings $Kodex(G, 0.0)$, $Kodex(G, 0.5)$, $Kodex(G, 1.0)$, $Louvain(G)$ et $Infomap(G)$ on the graphs G_{FrSynV}, G_{Actors}, G_{DBLP} and G_{Amazon}.

Table 3: **Kodex clusterings of four terrain graphs with $\beta = 0.0$, $\beta = 0.5$, $\beta = 1.0$** $\langle -\ ;\ - \rangle$ for $\langle$ Number of modules ; Length of the biggest modules $\rangle$

	$\mathbf{G_{FrSynV}}$	$\mathbf{G_{Actors}}$	$\mathbf{G_{DBLP}}$	$\mathbf{G_{Amazon}}$
$\beta = \mathbf{0.0}$	$\langle 671\ ;\ 155 \rangle$	$\langle ??\ ;\ ?? \rangle$	$\langle 17,437\ ;\ 249 \rangle$	$\langle 20,703\ ;\ 535 \rangle$
$\beta = \mathbf{0.5}$	$\langle 785\ ;\ 126 \rangle$	$\langle ??\ ;\ ?? \rangle$	$\langle ??\ ;\ ?? \rangle$	$\langle ??\ ;\ ?? \rangle$
$\beta = \mathbf{1.0}$	$\langle 861\ ;\ 89 \rangle$	$\langle ??\ ;\ ?? \rangle$	$\langle ??\ ;\ ?? \rangle$	$\langle ??\ ;\ ?? \rangle$



(a) G_{FrSynV} (b) G_{Actors} (c) G_{DBLP} (d) G_{Amazon}

Figure 5: **Fscore(Γ_1, Γ_2)**: Comparison of clusterings $Kodex(G, 0.0)$, $Kodex(G, 0.5)$, $Kodex(G, 1.0)$, $Louvain(G)$ and $Infomap(G)$ on G_{FrSynV}, G_{Actors}, G_{DBLP} et G_{Amazon}.

(TODO: Commenter les resultats de la table 3 et de la figure 5: Bruno)

6 Conclusions and perspectives

In this paper, we defined $Confluence$, a mesoscopic vertex closeness measure based on short random walks, that we introduced into $Modularity$ to define Q_{Conf}, a new clustering quality function. With four small toy graphs, we showed that optimal clusterings for Q_{Conf} improve the resolution of optimal clusterings for $Modularity$.

We then introduced $Kodex(G)$, a heuristic based on the $Confluence$ of edges to optimize Q_{Conf} on a graph G. On the same four little toy graphs, we showed that $Kodex$ finds an optimal clustering for Q_{Conf} and that for two of those toy graphs, $Kodex(G) = Infomap(G)$, while on the other two, $Kodex(G)$ identifies the modules we expected while $Infomap(G)$ regroups all vertices into a single module.

We showed that $Kodex$ avoids the resolution problem of $Louvain$ on four real terrain graphs, and that the modules computed by $Infomap$ are closer to those computed by

Kodex than the ones produced by *Louvain*.

Finally, to get a handle on the size of modules, we introduced the $\beta \in \mathbb{R}$ parameter, resulting in the Q^{β}_{Conf} quality function and the $Kodex(G, \beta)$ variant of the algorithm. We then showed that this approach is not hierarchical[8].

(TODO: Alp et Alexandre: Avantages de l'implémentation Haskel: Monotache/Parallelisme, Complexité/Efficacité, Maintenance, Portabilité ...)

Length of Random Walks:

For clarity and simplicity, we restricted the random walks of $P^t_G(i \rightsquigarrow j)$ to a length of $t = 3$. A first study of the impact of the length of those random walks was done in [Gaume et al., 2010], but a deeper one should be carried to understand how the length influences the **meso**scopicity of $Confluence$ and its effect on Q_{Conf} and $Kodex$. Indeed, if G is connected and $i \neq j$:

- **When t = 1:** $Conf^t(G, i, j) = \begin{cases} \frac{2m - d_G(i).d_G(j)}{2m + d_G(i).d_G(j)} \text{ if } \{i, j\} \in E, \\ -1 \text{ otherwise.} \end{cases}$ $Confluence$ is **independent of the intermediate structures** between the two vertices i and j ;
- **When 1 < t < ∞:** $Conf^t(G, i, j) = \frac{P^t_G(i \rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^t_G(i \rightsquigarrow j) + \frac{d_G(j)}{2m}}$, $Confluence$ is **dependent of the t-intermediate structures (t-mesoscopicity)** between the two vertices i and j (see13 and 14);
- **When t → ∞:** $\lim_{t \to \infty} Conf^t(G, i, j) = 0$, $Confluence$ is **constant**[9].

For example we can build the graph $G^{1\bigstar}_{toy}$ from G^1_{toy} by inserting a new vertex in the middle of each edge. Figure 6 illustrates the optimal clusterings on G^1_{toy} and on $G^{1\bigstar_{self.memory}}_{toy}$ for $Q^{0.0}_{Conf}$ with $t = 3$ and with $t = 6$, allowing us to see that:

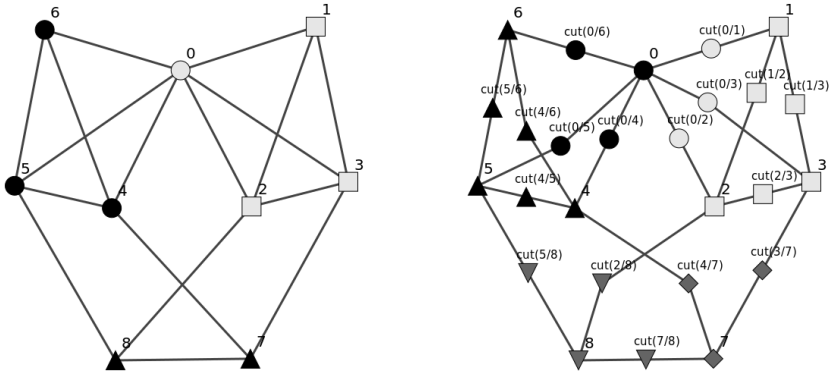
> $\mathbf{G^{1\bigstar}_{toy}}$ with $\mathbf{t = 6}$: we get the modules $\Delta^{G^1_{toy}}_{Conf} = \left\{\delta^1_{Conf} = \{0, 4, 5, 6\},\ \delta^2_{Conf} = \{1, 2, 3, \},\ \delta^3_{Conf} = \{7, 8\}\right\}$ i.e the optimal clusters of G^1_{toy} for $Q^{0.0}_{Conf}$ with $t = 3$:
>
> - $\delta^1_{Conf} = \{0, 4, 5, 6\} \subset \{0, 4, 5, 6, cut(0/4), cut(0/5), cut(0/6), cut(4/5), cut(4/6), cut(5/6)\}$;
> - $\delta^2_{Conf} = \{1, 2, 3, \} \subset \{1, 2, 3, cut(0/1), cut(0/2), cut(0/3), cut(1/2), cut(1/3), cut(2/3)\}$;

[8] If $\beta_1 < \beta_2$ then the optimal modules for $Q^{\beta_2}_{Conf}$ are not necessarily subsets of the optimal modules for $Q^{\beta_1}_{Conf}$.

[9] We can prove with the Perron-Frobenius theorem [Stewart, 1994] that if G is connected, then $\lim_{t \to \infty} P^t_G(i \rightsquigarrow j) = \frac{d_G(j)}{2m}$ and so by proposition 1: $P^t_G(i \rightsquigarrow j) = P^t_{G_{null}}(i \rightsquigarrow j)$, therefore $Conf^t(G, i, j) = 0$.

– $\delta^3_{Conf} = \{7, 8\} \subset \{7, 8, cut(7/8), cut(3/7), cut(4/7), cut(2/8), cut(5/8)\}$.

On $G^{1\bigstar}_{toy}$ with $\mathbf{t = 3}$: we do not get the modules of $\Delta^{G^1_{toy}}_{Conf}$.



(a) G^1_{toy} (b) $G^{1\bigstar}_{toy}$

Figure 6: **Optimal clusterings for $Q^{0.0}_{Conf}$ with $t = 3$ and with $t = 6$:** shapes describe an optimal clustering for $Q^{0.0}_{Conf}$ with $t = 3$, colors describe an optimal clustering for $Q^{0.0}_{Conf}$ with $t = 6$.

Directed graphs:

If G is a positively weighted graph by $W = \{w_{i,j} \text{ such } \{i, j\} \in E\}$, then we can apply Q_{Conf} and $Kodex$ by replacing equations 6 and 9 by 22 and 23 respectively:

$$[G] = (g_{i,j})_{i,j \in V} \quad \text{with} \quad g_{i,j} = \begin{cases} \dfrac{w_{i,j}}{\sum_{k \in V} w_{i,k}} \text{ if } \{i, j\} \in E, \\ 0 \text{ otherwise.} \end{cases} \tag{22}$$

$$Conf^t(G, i, j) = \begin{cases} 0 \text{ if } i = j, \\ \dfrac{P^t_G(i \rightsquigarrow j) - \frac{\sum_{k \in V} w_{k,j}}{\sum_{w \in W} w}}{P^t_G(i \rightsquigarrow j) + \frac{\sum_{k \in V} w_{k,j}}{\sum_{w \in W} w}} \text{ otherwise.} \end{cases} \tag{23}$$

If G is a directed graph, one can also consider using a variant of page rank [Gaume and Mathieu, 2016] in place of equation 7.

Overlaps:

In order to define a heuristic for computing clusters with potential overlaps between modules, one could consider using *Hbec* to isolate vertices between several dense regions of the graph (see section 4) and allowing them to belong to the modules corresponding to those dense regions, identified with the help of *ConFness* (see 17).

References

[Albert and Barabasi, 2002] Albert, R. and Barabasi, A.-L. (2002). Statistical Mechanics of Complex Networks. *Reviews of Modern Physics*, 74:74–47.

[Arenas et al., 2008] Arenas, A., Fernández, A., and Gómez, S. (2008). Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039.

[Blondel et al., 2008] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

[Bollobas, 2002] Bollobas, B. (2002). *Modern Graph Theory*. Springer-Verlag New York Inc.

[Brandes et al., 2008] Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. (2008). On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188.

[Clauset et al., 2004] Clauset, A., Newman, M. E. J., and Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, 70(6).

[Clauset et al., 2009] Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009). Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703.

[Fortunato and Barthelemy, 2006] Fortunato, S. and Barthelemy, M. (2006). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41.

[Gaume and Mathieu, 2016] Gaume, B. and Mathieu, F. (2016). Pagerank induced topology for real-world networks. *HAL:https://hal.archives-ouvertes.fr/hal-01322040*.

[Gaume et al., 2010] Gaume, B., Mathieu, F., and Navarro, E. (2010). Building Real-World Complex Networks by Wandering on Random Graphs. *I3: Information Interaction Intelligence*, 10(1).

[Goldstein et al., 2004] Goldstein, M. L., Morris, S. A., and Yen, G. G. (2004). Problems with fitting to the power-law distribution. *The European Physical Journal B - Condensed Matter and Complex Systems*, 41(2):255–258.

[Grünwald, 2007] Grünwald, P. D. (2007). *The minimum description length principle*. MIT press.

[Knuth, 1968] Knuth, D. E. (1968). *The Art of Computer Programming: Fundamental algorithms*. The Art of Computer Programming. Addison -Wesley.

[Kumpula et al., 2007] Kumpula, J. M., Saramäki, J., Kaski, K., and Kertész, J. (2007). Limited resolution in complex network community detection with potts model approach. *The European Physical Journal B*, 56(1):41–45.

[Lancichinetti and Fortunato, 2011] Lancichinetti, A. and Fortunato, S. (2011). Limits of modularity maximization in community detection. *Physical Review E*, 84(6).

[Leskovec and Krevl, 2014] Leskovec, J. and Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

[Newman, 2003] Newman, M. E. J. (2003). The Structure and Function of Complex Networks. *SIAM Review*, 45:167–256.

[Newman, 2005] Newman, M. E. J. (2005). Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351.

[Newman and Girvan, 2004] Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2).

[Radicchi et al., 2004] Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., and Parisi, D. (2004). Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663.

[Reichardt and Bornholdt, 2006] Reichardt, J. and Bornholdt, S. (2006). Statistical mechanics of community detection. *Physical Review E*, 74(1).

[Rossi and Ahmed, 2015] Rossi, R. A. and Ahmed, N. K. (2015). The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Rosvall and Bergstrom, 2008] Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

[Stewart, 1994] Stewart, G. W. (1994). Perron-frobenius theory: a new proof of the basics. Technical report, College Park, MD, USA.

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective Dynamics of Small-World Networks. *Nature*, 393:440–442.