Starling: Introducing a mesoscopic scale with Confluence for Graph Clustering

Bruno Gaume

PLOS ONE

RESEARCH ARTICLE

Starling: Introducing a mesoscopic scale with *Confluence* for Graph Clustering

Bruno Gaume *

Centre National de la Recherche Scientifique, CLLE, ISCPIF, Toulouse, France

* gaume.bruno@gmail.com

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 based on short Random walks, which brings together vertices from a same overconnected region of the Graph G, and separates vertices coming from two distinct overconnected regions. *Confluence* becomes a useful tool for defining a new Clustering quality function $Q_{Conf}(G, \Gamma)$ for a given Clustering Γ and for defining a new heuristic *Starling* to find a partitional Clustering of a Graph G intended to optimize the Clustering quality function Q_{Conf}. We compare the accuracies of *Starling*, to the accuracies of three state of the art Graphs Clustering methods: *Spectral-Clustering*, *Louvain*, and *Infomap*. These comparisons are done, on the one hand with artificial Graphs (a) Random Graphs and (b) a classical Graphs Clustering *Benchmark*, and on the other hand with (c) *Terrain-Graphs* gathered from real data. We show that with (a), (b) and (c), *Starling* is always able to obtain equivalent or better accuracies than the three others methods. We show also that with the *Benchmark* (b), *Starling* is able to obtain equivalent accuracies and even sometimes better than an *Oracle* that would only know the expected overconnected regions from the *Benchmark*, ignoring the concretely constructed edges.

Citation: Gaume B (2023) *Starling*: Introducing a mesoscopic scale with *Confluence* for Graph Clustering. PLoS ONE 18(8): e0290090. https://doi.org/10.1371/journal.pone.0290090

Editor: Ilya Safro, University of Delaware, UNITED STATES

Received: April 26, 2022

Accepted: August 1, 2023

Published: August 24, 2023

Data Availability Statement: All relevant data are within the manuscript.

Funding: Financed by Centre National de la Recherche Scientifique.

Competing interests: The authors have declared that no competing interests exist.

1 Introduction

Terrain-Graphs are real world Graphs that model data gathered by field work, in diverse fields such as sociology, linguistics, biology, or Graphs from the internet. Most *Terrain-Graphs* contrast with artificial Graphs (deterministic or Random) and share four similar properties [1–3]. They exhibit:

- $\mathbf{p_1}$:. Not many edges : m being $O(n.\log(n))$ (where m is the number of edges and n the number of vertices);
- $\mathbf{p_2}$:. Short paths (L, the average number of edges on the shortest path between two vertices is low);
- $\mathbf{p_3}$:. A high Clustering rate $C = \frac{3 \times number\ of\ triangles}{number\ of\ connected\ triplets}$ (many overconnected local subGraphs in a globally sparse Graph);
- $\mathbf{p_4}$:. 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-Graph* consists of grouping together in Modules vertices that belong to the same overconnected region of the Graph (property p_3), while keeping separate vertices that do not (property p_1). These groups of overconnected vertices form an essential feature of the structures of most *Terrain-Graphs*. Their detection is central in a wide variety of fields, such as in biology [4], in sociology [5], in linguistics [6] or in computer sciences [7], for many tasks as the grouping of most diverse entities [8–13], the pattern detection in data [14], the prediction of links [15], the model training [16], the label assignment [17], the recommender Algorithms [18], the data noise removal [19], or the feature matching [20].

In section 2 we put in context in the state of the art, the methods with which we compare our results: in section 2.1 we present the *Spectral-Clustering*, one of the most popular and efficient Graph Clustering methods, in section 2.2.1 *Louvain*, one of the most used Graph Clustering method optimizing *Modularity* the most popular Graph Clustering quality function, and in section 2.2.2 *Infomap*, one of the most efficient Graph Clustering method optimizing the most elegant Graph Clustering quality function.

In section 3 we present the *Confluence*, a vertex mesoscopic closeness measure and a new Clustering quality function Q_{Conf} based on the *Confluence*. In section 4 we compare optimality for *Modulatity* and optimality for Q_{Conf}. In section 5 we propose to consider a clustering method, as Binary Edge-Classifier By nodes Blocks (*BECBB*) trying to classify each pairs of vertices into two classes: the edges and the non-edges. In section 6 we propose a heuristic *Starling* for optimizing the objective function Q_{Conf}.

In section 7, we compare the accuracies as *BECBB*, of *Starling*, *Louvain*, *Infomap* and *Spectral-Clustering*. These comparisons are done, on the one hand with artificial Graphs (a) Random Graphs and (b) a classical Graphs Clustering *Benchmark*, and on the other hand with (c) *Terrain-Graphs* gathered from real data. We show that with (a), (b) and (c), *Starling* is always able to obtain equivalent or better accuracies than the three others methods. We show also that with the *Benchmark* (b), *Starling* is able to obtain equivalent accuracies and even sometimes better than an *Oracle* that would only know the expected overconnected regions from the *Benchmark*, ignoring the concretely constructed edges that are to be predicted by the *Oracle* as *BECBB*.

In section 8 we discuss the choice of parameters, and conclude in section 9.

2 Previous work

The literature on Graph Clustering is too extensive for a comprehensive review here. We concentrate on placing in the state of art, the methods to which we compare our results.

Let G = (V, E) be a Graph with n = |V| vertices and m = |E| edges.

$\mathcal{P}_2^V : \mathcal{P}_2^V = \{\mathcal{X} \subseteq V \text{ such } |\mathcal{X}| = 2\}$;

Degree: The degree of a vertex i in G is $d_G(i) = |\{j \in V / \{i, j\} \in E\}|$;

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

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:
$\forall u, v \in V,\ (u \overset{\Gamma}{\sim} v) \Leftrightarrow (\exists \gamma \in \Gamma \text{ such that } u \in \gamma \text{ and } v \in \gamma)$.

2.1 Spectral Graph Clustering

Spectral Graph Clustering is one of the most popular and efficient Graph Clustering Algorithms. It generally use the classical *kmeans* Algorithm whose original idea was proposed by Hugo Steinhaus [21]. Spectral Graph Clustering Algorithms work as follows (see [22]):

Algorithm 1 *SGC*: Spectral Graph Clustering

Input:
$G = (V, E)$ an undirected Graph with $|V| = n$
$\kappa \in \mathbb{N}$ such $0 < \kappa \leqslant n$ (κ is the desired number of Modules).

Output: A Partitional Clustering of G with κ Modules

(1) Form the Adjacency Matrix $A = (a_{i,j})_{i,j \in V}$ with $a_{i,j} = \begin{cases} 1 \text{ if } \{i,j\} \in E, \\ 0 \text{ otherwise.} \end{cases}$

(2) Form the Degree Matrix $D = (d_{i,j})_{i,j \in V}$ with $d_{i,j} = \begin{cases} d_G(i) \text{ if } \{i = j\}, \\ 0 \text{ otherwise.} \end{cases}$

(3) Let $L \in \mathbb{R}^{n \times n}$ the *Normalized Graph Laplacian*: $L = I - D^{-1}A$ (where I is the identity matrix $\in \mathbb{R}^{n \times n}$).

(4) Compute the first κ eigenvectors $u_1, \ldots, u_\kappa$ of L (see [23]).

(5) Let $U \in \mathbb{R}^{n \times \kappa}$ be the matrix containing the vectors $u_1, \ldots, u_\kappa$ as columns.

(6) For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^\kappa$ be the vector corresponding to the i-th row of U.

(7) Cluster the points $(y_i)_{i=1,\ldots,n} \in \mathbb{R}^\kappa$ with the *k-means* Algorithm into κ clusters $C_1, \ldots, C_k$.

(8) For $i = 1, \ldots, \kappa$, let $c_i = \{j | y_j \in C_i\}$.

Return $\{c_1, \ldots, c_\kappa\}$

We can notice that for Spectral Graph Clustering in Algorithm 1, we need to know κ the number of groups of vertices in advance in the *Input*. It is an advantage because it makes it possible to have a handle on the desired number of Modules, but how to choose κ when one does not know the structure of the Graph? The choice of the number κ of groups is fundamental, it is not a simple problem (see [23–31]), and the quality of the results varies greatly depending on κ, what we confirm in section 7.2.1 with Figs 7 and 8.

2.2 When we don't know the number of groups in advance

Let G = (V, E) be a Graph and Γ a Partitional Clustering of its vertices.

Clustering quality function: A Clustering quality function $Q(G, \Gamma)$ is an $\mathbb{R}$-valued function designed to measure the adequacy of the Modules with the overconnected regions of *Terrain-Graphs* (property p_3).

When we don't know κ the number of groups of vertices in advance, given a Clustering quality function Q, in order to establish a good Partitional Clustering for a Graph $G = (V, E)$, it would 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 concretely 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 [32]. 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.

With methods optimizing a quality function Q, we do not need to know κ the number of vertices groups in advance in the input, because κ is then a direct consequence of the quality function Q: κ will be automatically built by the optimisation of Q.

2.2.1 Louvain. The **Louvain** method proposed in 2008 by Blondel, Guillaume, Lambiotte, and Lefebvre in [33] is a heuristic for tentatively maximizing the quality function *Modularity*

proposed in 2004 by Newman and Girvan [34]. 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 expected 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 $\frac{d_G(x).d_G(y)}{2m}$.

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. The definition of modularity given by Newman and Girvan in [34], is equivalent to that we propose here in Formula 1:

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

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

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

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

In Eq 1, the first term $\frac{1}{2m}$ is purely conventional, so that the 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 $Q_{P_{edge}}$ as Newman and Girvan's quality function, to be maximized:

$$Q_{P_{edge}}(G, \Gamma) = \sum_{\gamma \in \Gamma} \sum_{i,j \in \gamma} P_{edge}(G, i, j) - P_{edge}(G_{Null}, i, j) \tag{4}$$

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

For Louvain, a good Partitional Clustering Γ as per 5 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).

However, several authors [35, 36] 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 [37, 38] defined multiresolution variants of *Modularity*, adding a resolution parameter to control the size of the Modules.

For instance [37] introduces a parameter $\lambda \in \mathbb{R}$ in Eq 5:

$$Q_{\lambda} = \sum_{\gamma \in \Gamma} \sum_{i,j \in \gamma} \begin{cases} 1 - \lambda.\frac{d_G(i).d_G(j)}{2m} \text{ if } \{i, j\} \in E, \\ -\lambda.\frac{d_G(i).d_G(j)}{2m} \text{ otherwise.} \end{cases} \tag{6}$$

where λ is a resolution parameter: the higher the resolution λ, the smaller the Modules get.

Nevertheless, in [39], 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 [40, 41]. *Therefore there is no characteristic cluster size and tuning a resolution parameter may not help.*"

The *Louvain* method https://github.com/10XGenomics/louvain is non-deterministic, i.e. each time *Louvain* is run on the same Graph, the results may vary slightly. In the rest of this paper all the results concerning the *Louvain* method on a given Graph are the result of a single run on this Graph.

2.2.2 Infomap. The **Infomap** method is a heuristic for tentatively maximizing the quality function described in 2008 by Rosvall and Bergstrom [42]. This quality function is based on the minimum description length principle [43]. 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 trapped when it enters a good Module, which is supposed to be a overconnected region that is only weakly connected to other Modules.

For Infomap, a good Partitional Clustering Γ is then one that groups in same Module vertices allowing a good compression ratio for describing the trajectory of a Random walker on *G*.

However, as we will see in section 7, *Infomap* only identifies a single Module when the overconnected regions are only sligthly pronounced.

The *Infomap* method https://github.com/mapequation/ is non-deterministic, in the rest of this paper all the results concerning the *Infomap* method on a given Graph are the result of a single run on this Graph.

3 *Confluence*, a vertices mesoscopic closeness measure

The definition of *Confluence* proposed in this section is an adaptation of these proposed in [44] to compare the structures of two *Terrain-Graphs*.

In Eq 5, with regards to a Graph *G*:

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

To avoid the resolution limits of *Modularity* described in [35–39], we introduce here *Confluence*(*G*, *i*, *j*), an intermediate **mesoscopic** vertices closeness measure relative to a Graph *G*, that we define below.

If $G = (V, E)$ is a reflexive 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 a uniform probability. This process is called a simple Random walk [45]. It can be defined by a Markov chain on V with an $n \times n$ transition Matrix $[G]$:

$$[G] = (g_{i,j})_{i,j \in V} \quad \text{with} \quad g_{i,j} = \begin{cases} \dfrac{1}{d_G(i)} & \text{if } \{i,j\} \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

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: $\forall i \in V, \Sigma_{j \in V} g_{i,j} = 1$. The probability $P^t_G(i \rightsquigarrow j)$ of a walker starting on vertex i and reaching vertex j after t steps is:

$$P^t_G(i \rightsquigarrow j) = ([G]^t)_{i,j} \tag{8}$$

Proposition 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 $e_{i,j} = \frac{d_G(i).d_G(j)}{2m}$.*

$$\forall t \in \mathbb{N}^*, \ \forall i, j \in V, \ P^t_{G_{null}}(i \rightsquigarrow j) = \frac{d_G(j)}{2m} \tag{9}$$

Proof by induction on t:

(a) True for t = 1:

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

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

$$\forall i, j \in V, P^{t+1}_{G_{null}}(i \rightsquigarrow j) = \sum_{k \in V} (P^t_{G_{null}}(i \rightsquigarrow k) . P^1_{G_{null}}(k \rightsquigarrow j))$$

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

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

(a) & (b) ⇒ 9

On a Graph G = (V, 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 $P^t_G(i \rightsquigarrow j)$ of reaching j from i.

On the Graph G_{null} the trajectory of a Random walker is only governed by the degrees of the vertices i and j, and no longer by the topology of the Graph in the vicinity of these to nodes.

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:

$$Conf^t(G,i,j) = \begin{cases} 0 \text{ if } i = j, \\ \dfrac{P^t_G(i \rightsquigarrow j) - P^t_{G_{null}}(i \rightsquigarrow j)}{P^t_G(i \rightsquigarrow j) + P^t_{G_{null}}(i \rightsquigarrow j)} = \dfrac{P^t_G(i \rightsquigarrow j) - \frac{d_G(j)}{2m}}{P^t_G(i \rightsquigarrow j) + \frac{d_G(j)}{2m}} \text{ otherwise.} \end{cases} \quad (10)$$

Proposition 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 $e_{i,j} = \frac{d_G(i).d_G(j)}{2m}$.*

$$\forall t \in \mathbb{N}^*, \ \forall i, j \in V, \ Conf^t(G_{Null}, i, j) = 0 \quad (11)$$

Proof:

$$\text{If } i = j, \text{ the result follows directly from definition } 10.$$

$$\text{If } i \neq j, Conf^t(G_{Null}, i, j) = \frac{P^t_{G_{Null}}(i \rightsquigarrow j) - \frac{d_{G_{Null}}(j)}{2m}}{P^t_{G_{Null}}(i \rightsquigarrow j) + \frac{d_{G_{Null}}(j)}{2m}} \text{ (by definition } 10)$$

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

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

$$= 0$$

To prove that $Conf^t(G, \cdot, \cdot)$ is symmetric, we first need to prove proposition 3.
Proposition 3 *Let $G = (V, E)$ be a reflexive Graph.*

$$\forall t \in \mathbb{N}^*, \ \forall i, j \in V, P^t_G(i \rightsquigarrow j) = \frac{d_G(j)}{d_G(i)}.P^t_G(j \rightsquigarrow i) \quad (12)$$

Proof by induction on t:
(a) True for t = 1:

$$\forall i, j \in V, \text{if } \{i, j\} \notin E, \text{ then } P^1_G(i \rightsquigarrow j) = 0 \text{ and } P^1_G(j \rightsquigarrow i) = 0,$$

$$\text{therefore } P^1_G(i \rightsquigarrow j) = \frac{d_G(j)}{d_G(i)}.P^1_G(j \rightsquigarrow i) = 0$$

$$\text{otherwise } P^1_G(i \rightsquigarrow j) = \frac{1}{d_G(i)} = \frac{d_G(j)}{d_G(i)}.\frac{1}{d_G(j)} = \frac{d_G(j)}{d_G(i)}.P^1_G(j \rightsquigarrow i)$$

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

$$\forall i, j \in V, P^{t+1}_G(i \rightsquigarrow j) = \sum_{k \in V}(P^t_G(i \rightsquigarrow k).P^1_G(k \rightsquigarrow j))$$

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

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

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

(a) & (b) ⇒ 12

Proposition 4 *Let $G = (V, E)$ be a reflexive Graph.*

$$\forall t \in \mathbb{N}^*,\ \forall i, j \in V, Conf^t(G, i, j) = Conf^t(G, j, i) \tag{13}$$

Proof:

If $i = j$: it follows directly from definition 10.

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

$$= \frac{\left(\frac{d_G(j)}{d_G(i)}.P^t_G(j \rightsquigarrow i) - \frac{d_G(j)}{2m}\right).\frac{d_G(i)}{d_G(j)}}{\left(\frac{d_G(j)}{d_G(i)}.P^t_G(j \rightsquigarrow i) + \frac{d_G(j)}{2m}\right).\frac{d_G(i)}{d_G(j)}} = \frac{P^t_G(j \rightsquigarrow i) - \frac{d_G(i)}{2m}}{P^t_G(j \rightsquigarrow i) + \frac{d_G(i)}{2m}} = \frac{P^t_G(j \rightsquigarrow i) - P^t_{G_{null}}(j \rightsquigarrow i)}{P^t_G(j \rightsquigarrow i) + P^t_{G_{null}}(j \rightsquigarrow i)}$$

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

Most *Terrain-Graphs* 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-Graph* (because of property p_2). On the contrary, *Confluence* allows us to identify vertices living in a same overconnected region of G (property p_3):

If i, j are in a same overconnected region:

$$P^t_G(i \rightsquigarrow j) > P^t_{G_{null}}(i \rightsquigarrow j),\ \text{thus}\ \ Conf(G, i, j) > 0 \tag{14}$$

If i, j are in two distinct overconnected regions:

$$P^t_G(i \rightsquigarrow j) < P^t_{G_{null}}(i \rightsquigarrow j),\ \text{thus}\ \ Conf(G, i, j) < 0 \tag{15}$$

Where the notion of *region* varies according to t:

- **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 a **microscopic** vertices closeness measure relative to G. The notion of **region in this case has a radius = 1**, it is the notion of *neighborhood*. *Confluence* is then **independent of the intermediate structures** between the two vertices i and j in G;
- **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 a **mesoscopic** vertices closeness measure relative to G. The notion of **region in this case has a 1 < radius = t < ∞**, it is no longer a local notion as the notion of *neighborhood*. *Confluence* is then **sensitive to the t-intermediate structures (t-mesoscopicity)** between the two vertices i and j in G (see 14 and 15);
- **When t → ∞**: $\lim_{t\to\infty} Conf^t(G, i, j) = 0$, and *Confluence* is no longer sensitive to any structure in G. ($\lim_{t\to\infty} Conf^t(G, i, j) = 0$ because we can prove with the Perron-Frobenius theorem [46] that if G is reflexive and strongly connected, then the Matrix $[G]$ is ergodic [47], then $\lim_{t\to\infty} P^t_G(i \rightsquigarrow j) = \frac{d_G(j)}{2m}$. So by definition 10 and proposition 1: $\lim_{t\to\infty} Conf^t(G, i, j) = 0$).

Confluence actually defines an infinity of mesoscopic vertex closeness measures, one for each Random walk of length $1 < t < \infty$. For clarity, in the rest of this paper, we set $t = 3$ and define $Conf(G, i, j) = Conf^3(G, i, j)$.

3.1 Using a mesoscopic scale with *Confluence* for a new Clustering quality function

We propose here Q^{τ}_{Conf}, a new Clustering quality function, which introduces a mesoscopic scale through *Confluence* with a resolution parameter $\tau \in [0, 1]$ to promote density of the Modules:

$$\mathbf{Q_1}(\mathbf{G}, \mathbf{\Gamma}) = \sum_{\gamma \in \mathbf{\Gamma}} \sum_{\mathbf{i} \neq \mathbf{j} \in \gamma} \begin{cases} +\mathbf{1} - \dfrac{\mathbf{d_G(i).d_G(j)}}{\mathbf{2m}} \text{ if } \{\mathbf{i}, \mathbf{j}\} \in \mathbf{E}, \\ -\mathbf{1} - \dfrac{\mathbf{d_G(i).d_G(j)}}{\mathbf{2m}} \text{ otherwise.} \end{cases} \tag{16}$$

$$\mathbf{Q_0}(\mathbf{G}, \mathbf{\Gamma}) = \sum_{\gamma \in \mathbf{\Gamma}} \sum_{\mathbf{i} \neq \mathbf{j} \in \gamma} \mathbf{Conf}(\mathbf{G}, \mathbf{i}, \mathbf{j}) \tag{17}$$

$$\mathbf{Q^{\tau}_{Conf}}(\mathbf{G}, \mathbf{\Gamma}) = \tau.\mathbf{Q_1}(\mathbf{G}, \mathbf{\Gamma}) + (\mathbf{1} - \tau).\mathbf{Q_0}(\mathbf{G}, \mathbf{\Gamma}) \tag{18}$$

In Eq 16, with regard to a Graph G, the term $\begin{cases} +1 \text{ if } \{i, j\} \in E, \\ -1 \text{ otherwise.} \end{cases}$ is a **local (microscopic)** vertices closeness measure, and the term $\frac{d_G(i).d_G(j)}{2m}$ is a **global (macroscopic)** vertices closeness measure, when in Eq 17, the term $Conf(G, i, j)$ is an **intermediate local/global (mesoscopic)** vertices closeness measure.

Therefore in Eq 18, $Q^{\tau}_{Conf}(G, \Gamma)$ gives a weight of τ to the microscopic and macroscopic structure of Γ with regards to the Graph G and a weight of $(1 - \tau)$ to the mesoscopic structure. The closer the $\tau \in [0, 1]$ parameter is to 1, the less *Confluence* is taken into account.

4 Optimality

A Partitional Clustering **Δ is optimal** for a quality function Q **iff** for all partitioning Γ of V, $Q(G, \Delta)) \geqq Q(G, \Gamma))$. Computing a Δ that maximizes $Q_{P_{edge}}(G, \Delta)$ is $\mathcal{NP} - complete$ [48], and the same holds for computing a Clustering that maximizes Q^{τ}_{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 [48]): 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: $\forall i, j \in V : X_{ij} = X_{ji}$;

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:

$$\textbf{For } \mathbf{Q}_{\mathbf{P_{edge}}} : \sum_{i,j \in V} X_{ij} \begin{cases} 1 - \dfrac{d_G(i).d_G(j)}{2m} \text{ if } \{i,j\} \in E, \\ -\dfrac{d_G(i).d_G(j)}{2m} \text{ otherwise.} \end{cases} \tag{19}$$

$$\textbf{For } \mathbf{Q}^{\tau}_{\mathbf{Conf}} : \sum_{i \neq j \in V} X_{ij} \cdot \begin{cases} (1-\tau).Conf(G,i,j) + \tau - \tau.\dfrac{d_G(i).d_G(j)}{2m} \text{ if } \{i,j\} \in E, \\ (1-\tau).Conf(G,i,j) - \tau - \tau.\dfrac{d_G(i).d_G(j)}{2m} \text{ otherwise.} \end{cases} \tag{20}$$

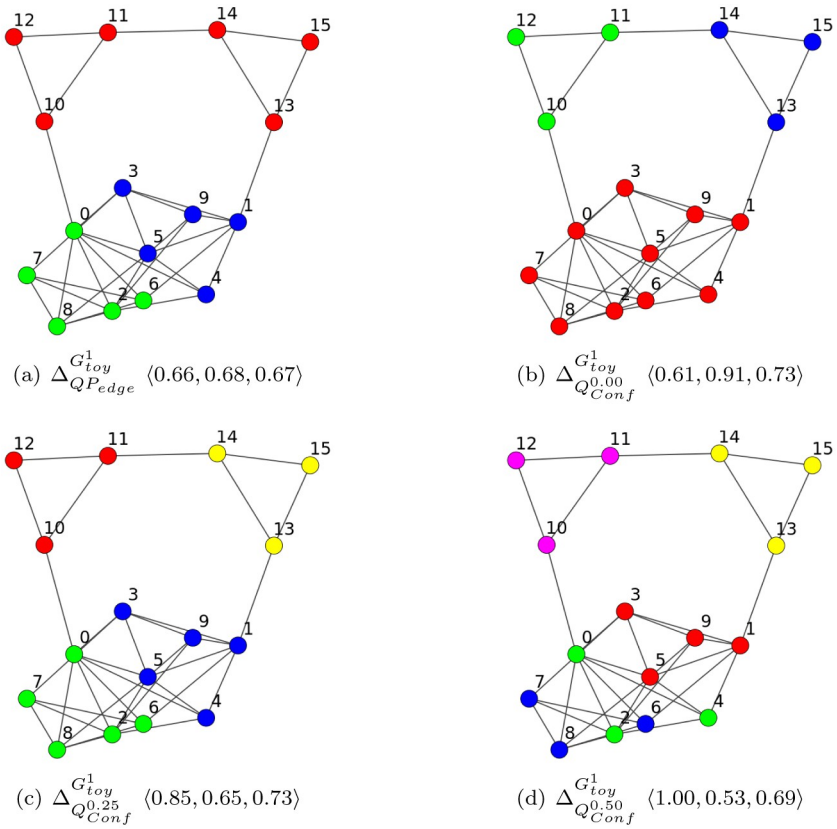
The method *SGC* described in Algorithm 1 do not optimize a quality function, and the quality function used by *Infomap* can not be expressed as $\sum_{\gamma \in \Gamma} \sum_{i,j \in \gamma} sim(G,i,j)$, with $sim(G,.,.)$ an $\mathbb{R}$-valued symmetric similarity measure between vertices of G. We therefore left out this functions in our study of optimality, not having the ability to define their corresponding objective function to maximize in a similar fashion to what was done for $Q_{P_{edge}}$ and Q^{τ}_{Conf} with the formulas 19 and 20. In Fig 1, on a small artificial Graph G^1_{toy}, we compare the optimal Clusterings $\Delta^{G^1_{toy}}_{Q_{P_{edge}}}$, $\Delta^{G^1_{toy}}_{Q^{0.00}_{Conf}}$, $\Delta^{G^1_{toy}}_{Q^{0.25}_{Conf}}$ and $\Delta^{G^1_{toy}}_{Q^{0.50}_{Conf}}$ (with the Graph G^1_{toy}, if $0.50 < x < 1$ then $\Delta^{G^1_{toy}}_{Q^{\tau=x}_{Conf}} = \Delta^{G^1_{toy}}_{Q^{0.50}_{Conf}}$) where:

- $\mathbf{\Delta^{G^1_{toy}}_{Q_{P_{edge}}}} = \left\{ \delta^1_{Q_{P_{edge}}} = \{0,2,6,7,8\},\ \delta^2_{Q_{P_{edge}}} = \{1,3,4,5,9\}, \right.$
  $\left. \delta^3_{Q_{P_{edge}}} = \{10,11,12,13,14,15\} \right\}$;
- $\mathbf{\Delta^{G^1_{toy}}_{Q^{0.0}_{Conf}}} = \left\{ \delta^1_{Q^{0.0}_{Conf}} = \{0,1,2,3,4,5,6,7,8,9\}, \delta^2_{Q^{0.0}_{Conf}} = \{10,11,12\}, \delta^3_{Q^{0.0}_{Conf}} = \{13,14,15\} \right\}$;
- $\mathbf{\Delta^{G^1_{toy}}_{Q^{0.25}_{Conf}}} = \left\{ \delta^1_{Q^{1.0}_{Conf}} = \{0,2,6,7,8\},\ \delta^2_{Q^{1.0}_{Conf}} = \{1,3,4,5,9\}, \delta^3_{Q^{1.0}_{Conf}} = \{10,11,12\}, \right.$
  $\left. \delta^4_{Q^{1.0}_{Conf}} = \{13,14,15\} \right\}$;
- $\mathbf{\Delta^{G^1_{toy}}_{Q^{0.50}_{Conf}}} = \left\{ \delta^1_{Q^{0.50}_{Conf}} = \{0,2,4\}, \delta^2_{Q^{0.50}_{Conf}} = \{1,3,5,9\}, \delta^3_{Q^{0.50}_{Conf}} = \{6,7,8\},\ \delta^4_{Q^{0.50}_{Conf}} = \{10,11,12\}, \right.$
  $\left. \delta^5_{Q^{0.50}_{Conf}} = \{13,14,15\} \right\}$.

We can already notice that growing τ does not imply a simple splitting of the Modules (an approach that would be hierarchical), which we can see by going from $\tau = 0.25$ to $\tau = 0.50$ where there is no $\delta_{Q^{0.25}_{Conf}}$ such that $\delta^1_{Q^{0.50}_{Conf}} = \{0,2,4\} \subseteq \delta_{Q^{0.25}_{Conf}}$.

5 Binary edge-classifier by nodes blocks

What metric to use to estimate the accuracy of the four Clusterings in Fig 1? Much literature addresses this fundamental question [49–51]. Here we propose the definition of **B***inary* **E***dge-***C***lassifier* **B***y nodes* **B***locks* (*BECBB*). To measure the quality of a Clustering Γ on a Graph $G = (V, E)$, an intuitive, simple and efficient approach is to consider a Clustering Γ (with or witout



(a) $\Delta^{G^1_{toy}}_{QP_{edge}}$ $\langle 0.66, 0.68, 0.67\rangle$ (b) $\Delta^{G^1_{toy}}_{Q^{0.00}_{Conf}}$ $\langle 0.61, 0.91, 0.73\rangle$

(c) $\Delta^{G^1_{toy}}_{Q^{0.25}_{Conf}}$ $\langle 0.85, 0.65, 0.73\rangle$ (d) $\Delta^{G^1_{toy}}_{Q^{0.50}_{Conf}}$ $\langle 1.00, 0.53, 0.69\rangle$

Fig 1. Optimal Clusterings for $Q_{P_{edge}}$ and Q_{Conf} on G^1_{toy}. If two vertices have same color, then they are in a same Module, with $\langle$**P, R, F**$\rangle$ where $\mathbf{P} = Precision(\mathcal{P}airs(\Gamma), E)$, $\mathbf{R} = Recall(\mathcal{P}airs(\Gamma), E)$, $\mathbf{F} = Fscore(\mathcal{P}airs(\Gamma), E)$.

https://doi.org/10.1371/journal.pone.0290090.g001

overlaps), as a *BECBB* trying to predict the edges of a Graph: classifying each pairs of vertices into two classes, the *PositiveEdge* and the *NegativeEdge*.

Definition: A BECBB is a pairs of nodes binary classifier trying to predict the edges of a Graph. It is not allowed to give two complementary sets of pairs of nodes, one for its predictions as *PositiveEdge* and its complementary set for its predictions as *NegativeEdge*, but is forced to provide its predictions in the form of nodes blocks $B_i \subseteq V$: classifying as *PositiveEdge* a pair $\{x, y\}$ if $\exists i$ such $x, y \in B_i$ else classifying it as *NegativeEdge*. If blocks are allowed to overlap then it is a $BECBB^{OV}$ else it is a $BECBB^{NO}$.

Let Γ a Clustering (with or witout overlaps) of a Graph $G = (V, E)$

$$\mathcal{P}airs(\Gamma) = \bigcup_{\gamma \in \Gamma} \mathcal{P}_2^{\gamma}; \tag{21}$$

$\mathcal{P}airs(\Gamma) \bigcap E = \mathbf{TP}(\mathbf{\Gamma}, \mathbf{E})$ are the True Postives of Γ according to E;
$\overline{\mathcal{P}airs(\Gamma)} \bigcup E = \mathbf{TN}(\mathbf{\Gamma}, \mathbf{E})$ are the True Negatives;
$\mathcal{P}airs(\Gamma) \bigcap \overline{E} = \mathbf{FP}(\mathbf{\Gamma}, \mathbf{E})$ are the False Postives;
$\overline{\mathcal{P}airs(\Gamma)} \bigcap E = \mathbf{FN}(\mathbf{\Gamma}, \mathbf{E})$ are the False Negatives.

We can then measure the Γ's accuracy with the classical measures in diagnostic binary Classification [52, 53]:

$$Precision(\mathcal{P}airs(\Gamma), E) = \frac{|TP(\Gamma, E)|}{|\mathcal{P}airs(\Gamma)|} \in [0, 1]; \tag{22}$$

$$Recall(\mathcal{P}airs(\Gamma), E) = \frac{|TP(\Gamma, E)|}{|E|} \in [0, 1]; \tag{23}$$

$$Fscore(\mathcal{P}airs(\Gamma), E) = 2.\frac{Precision(\mathcal{P}airs(\Gamma), E).Recall(\mathcal{P}airs(\Gamma), E)}{Precision(\mathcal{P}airs(\Gamma), E) + Recall(\mathcal{P}airs(\Gamma), E)} \in [0, 1]. \tag{24}$$

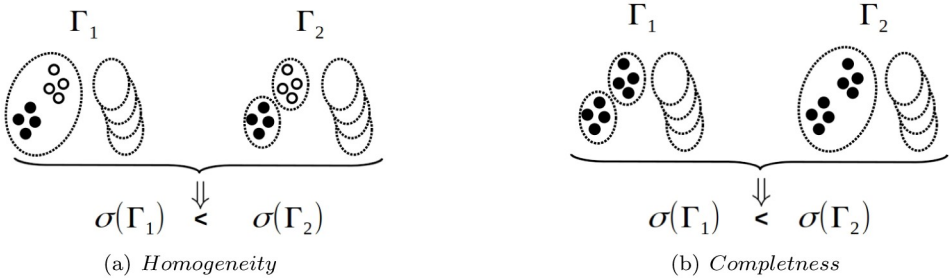
We can use these three measures indifferently on Clusterings with or without overlaps, because the Eq 21 makes sense with Clusterings with or without overlaps.

BECBB$^{\mathbf{OV}}$: For any Graph $G = (V, E)$, the set of all edges $\Gamma = E$ can be considered as a $BECBB^{OV}$. Then $\Gamma = E$ is optimal because: $Prec(\Gamma = E, E) = 1$ (Γ does not include any non-edge in its Modules) and $Rec(\Gamma = E, E) = 1$ (Γ include all the edges in its Modules). It is also true for $\Gamma =$ *The set of all the maximal cliques*.

BECBB$^{\mathbf{NO}}$: The metric $Precision(\mathcal{P}airs(\Gamma = Method(G)), E)$ measures the ability of a *Method* not to include non-edges in the Modules it returns, whereas the metric $Recall(\mathcal{P}airs(\Gamma = Method(G)), E)$ measures its ability to include the edges in the Modules it returns. For a $BECBB^{NO}$, a good *Precision* and a good *Recall* are two ability that oppose each other (because a $BECBB^{NO}$ is forced to provide its classifications in the form of blocks B_i without overlaps) but are simultaneously bothtogether desirable for a good Clustering method. The whole point of a good Clustering method, as $BECBB^{NO}$, is therefore to favor *Precision* without disfavoring *Recall* too much or even favoring *Recall* without disfavoring the *Precision* too much, that is what the metric $Fscore(\mathcal{P}airs(\Gamma = Method(G)), E)$ measures (it is the harmonic mean of *Precision* and *Recall*).

5.1 Properties

As showed in [51], it is better that a metric $\sigma(\Gamma)$, to estimate the accuracy of a Clustering Γ, has the *Homogeneity* and *Completeness* [50] properties (see Fig 2 inspired by Figs 1 and 3 in [51]).



(a) *Homogeneity* (b) *Completness*

Fig 2. Homogeneity and Completeness: $\{x, y\} \in E$ iff x and y have same color.

https://doi.org/10.1371/journal.pone.0290090.g002

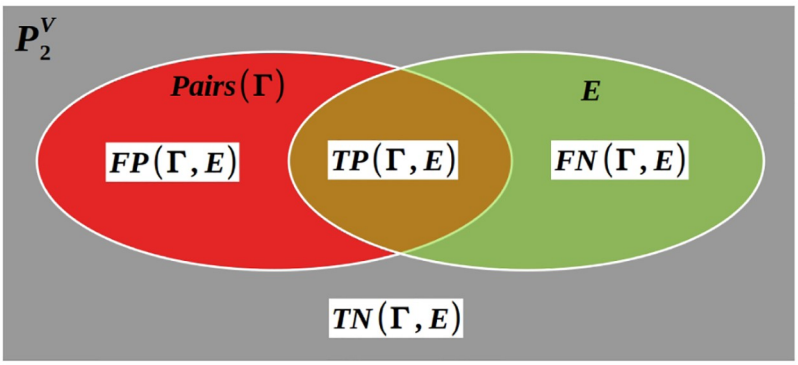


Fig 3. Binary classifiers of nodes pairs by nodes blocks.

https://doi.org/10.1371/journal.pone.0290090.g003

- A metric σ to estimate the accuracy of Clusterings, has the *Homogeneity* property iff:

$$\left.\begin{array}{l} TP(\mathcal{P}airs(\Gamma_1), E) = TP(\mathcal{P}airs(\Gamma_2), E) \\ \quad \text{and} \\ FP(\mathcal{P}airs(\Gamma_1), E) > FP(\mathcal{P}airs(\Gamma_2), E) \end{array}\right\} \Rightarrow \ \sigma(\Gamma_1) < \sigma(\Gamma_2) \tag{25}$$

- A metric σ to estimate the accuracy of Clusterings, has the *completeness* property iff:

$$\left.\begin{array}{l} TP(\mathcal{P}airs(\Gamma_1), E) < TP(\mathcal{P}airs(\Gamma_2), E) \\ \quad \text{and} \\ FP(\mathcal{P}airs(\Gamma_1), E) = FP(\mathcal{P}airs(\Gamma_2), E) \end{array}\right\} \Rightarrow \ \sigma(\Gamma_1) < \sigma(\Gamma_2) \tag{26}$$

It is clear that the metric $Fscore(\mathcal{P}airs(\Gamma),\ E)$ has these two properties, for any Clustering Γ with or without overlaps. Moreover the metric $Fscore(\mathcal{P}airs(\Gamma), E)$ is independent of any extrinsic expectation to the Graph, we only need to trust the Graph itself. It is a good objective way to evaluate and compare Clusterings. So, to estimate the accuracy of Clustering methods $Method_i$ and compare them on a Graph $G = (V, E)$, we will use the three metrics:

Precision(Method$_i$(G = (V, E)), E): Measuring the ability of the $Method_i$ not to include non-edges in the Modules it returns;

Recall(Method$_i$(G = (V, E)), E): Measuring its ability to include the edges in the Modules it returns;

Fscore(Method$_i$(G = (V, E)), E): Measuring the harmonic mean of its *Precision* and *Recall*.

Fig 1 shows the accuracy of $\Delta^{G^1_{toy}}_{Q_{P_{edge}}}$, $\Delta^{G^1_{toy}}_{Q^{0.00}_{Conf}}$, $\Delta^{G^1_{toy}}_{Q^{0.25}_{Conf}}$ and $\Delta^{G^1_{toy}}_{Q^{0.50}_{Conf}}$ considering these Clusterings as *BECBB*.

6 *Starling*, a heuristic for maximizing Q^{τ}_{Conf}

In this section we describe *Starling*, a heuristic for tentatively maximizing Q^{τ}_{Conf}. *Confluence* gives us an ordering on the edges of the Graph $G = (V, E)$, in particular, sorting the edges $\{i, j\}$

$\in E$ by descending *Confluence*, forms the basis of a new Module merging strategy, described in Algorithm 2, intended to optimize Q^{τ}_{Conf}.

Algorithm 2 *Starling*: Graph Partitional Clustering

Input:
$G = (V, E)$ an undirected Graph
$\tau \in [0, 1]$
Output: C_{out} a Partitional Clustering of G
$X \leftarrow \{\{i, j\} \in E \text{ such } i \neq j\}$
for $i \in V$ **do** ▸ Initialization
$mod_i \leftarrow \{i\}$ ▸ One vertex per Module
$M_i \leftarrow i$ ▸ Vertex i is in Module i
$\Upsilon \leftarrow \emptyset$
While $\Upsilon \neq X$ **do**
$\{i, j\} \leftarrow \underset{\{x,y\} \in X - \Upsilon}{\arg\max}\, Conf(G, x, y)$ ▸ Line 1: Strategy based on *Confluence*
$\Upsilon \leftarrow \Upsilon \cup \{\{i, j\}\}$
if $M_i \neq M_j$ **then** ▸ mod_i and mod_j have not yet been merged together

$$profit \leftarrow \sum_{u \in mod_i} \sum_{v \in mod_j} \begin{cases} (1-\tau).Conf(G,u,v) + \tau.\left(+1 - \frac{d_G(u).d_G(v)}{2m}\right) & \text{if } \{u,v\} \in E, \\ (1-\tau).Conf(G,u,v) + \tau.\left(-1 - \frac{d_G(u).d_G(v)}{2m}\right) & \text{otherwise.} \end{cases}$$

if $0 \leqslant profit$ **then**
$mod_i \leftarrow mod_i \cup mod_j$ ▸ mod_j merge with mod_i in mod_i
$mod_j \leftarrow \emptyset$ ▸ mod_j is dead
for $k \in V$ **do** ▸ Updating the membership list
if $M_k = j$ ▸ Vertex k was in mod_j
$M_k \leftarrow i$ ▸ Vertex k is now in mod_i
$C_{out} \leftarrow \emptyset$
for $i \in V$ **do**
if $mod_i \neq \emptyset$ ▸ mod_i is alive
$C_{out} \leftarrow C_{out} \cup \{mod_i\}$
return C_{out}

Different edges $\{i_1, j_1\} \in E$ and $\{i_2, j_2\} \in E$ might happen to have the exact same *Confluence* value ($Conf(G, i_1, j_1) = Conf(G, i_2, j_2)$), making the process (in `Line 1`) non-deterministic in general, because of its sensitivity on the order in which the edges with identical *Confluence* values are processed. A simple solution to this problem is to sort edges by first comparing their *Confluence* values and then using the lexicographic order on the words i_1j_1 and i_2j_2 when *Confluence* values are strictly identical.

We coded this Algorithm in C^{++} and in the following we used this program to analyze *Starling*'s results. With G^1_{toy}, $Starling(\tau, G^1_{toy})$ find the optimal Clusterings for Q^{τ}_{Conf}:

$Starling(0.00, G^1_{toy}) = \Delta^{G^1_{toy}}_{Q^{0.00}_{Conf}}$, $Starling(0.25, G^1_{toy}) = \Delta^{G^1_{toy}}_{Q^{0.25}_{Conf}}$, $Starling(0.50, G^1_{toy}) = \Delta^{G^1_{toy}}_{Q^{0.50}_{Conf}}$.

7 Performance

In this section we estimate the accuracy of *Starling* and compare it with the methods *Louvain*, *Infomap* and *SGC*. We can Estimate the accuracy of Clustering Algorithms on:

Real Graphs: A set of *Terrain-Graphs* built from real data;

A Benchmark$_{\mathcal{B}}$: A set of computer-generated Graphs and its gold standard $\Gamma_{\mathcal{B}}$ its expected Modules as expected overconnected regions.

Because we do not need to know κ the number of vertex groups in advance in the input of *Louvain* and *Infomap*, whereas we need it with *SGC*, for greater clarity, we compare on

the one hand *Starling* versus *Louvain*, and *Infomap*, and on the other hand *Starling* versus *SGC*.

7.1 Starling versus Louvain and Infomap

7.1.1 Performance on Real Terrain-Graphs. In this section we estimate the accuracy of Algorithms with three *Terrain-Graphs*:

- $\mathbf{G_{Email}}$: The Graph was generated using email data from a large European research institution [54, 55]. The Graph contains an undirected edge $\{i, j\}$ if person i sent person j at least one email https://snap.stanford.edu/data/email-Eu-core.html.
- $\mathbf{G_{DBLP}}$: The DBLP computer science bibliography provides a comprehensive list of research papers in computer science [56]. Two authors are connected if they have published at least one paper together https://snap.stanford.edu/data/com-DBLP.html.
- $\mathbf{G_{Amazon}}$: A Graph was collected by crawling the Amazon website. It is based on the *Customers Who Bought This Item Also Bought* feature of the Amazon website [56]. If a product i is frequently co-purchased with product j, the Graph contains an undirected edge $\{i, j\}$ https://snap.stanford.edu/data/com-Amazon.html.

Table 1 illustrates the pedigrees of these *Terrain-Graphs* and Table 2 shows the accuracies of *Louvain*, *Infomap* and *Starling* Considering each Clustering as a *BECBB*. We show also the number of Modules, the Length of the biggest Module and the computation time in seconds (All times are based on computations with a Quad Core Intel i5 and 32 Go RAM).

- **Louvain**: This is the fastest method, however its *Precision* is small, producing very few Modules, one of which is very large;
- **Infomap**: It gets a good *Fscore*, higher than this of *Louvain*.
- **Starling**$^{\tau}$: $\exists \tau \in [0, 1]$ such that *Starling*(G, τ) gets the highest *Fscore*. By default $\tau = 0.25$ is a good compromise to obtain at the same time a good *Precision* and a good *Recall*. If we want to promote *Recall* (more edges in Modules) then we can decrease τ, and if we want to promote *Precision* (less non-edges in Modules) then we can increase τ.

7.1.2 Performance on Benchmark$_{ER}$. $Benchmark_{ER}$ is the class of Random Graphs studied by Erdös and Rényi [57, 58] with parameters N the number of vertices and p the connection probability between two vertices. Random Graphs do not have a meaningful group structure, and they can be used to test if the Algorithms are able to recognize the absence of Modules. Therefore, we set $N = 128$, and we will study the accuracy of the methods with $Benchmark_{ER}$ according to p.

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	⟨k⟩	C	$L_{lcc}(n_{lcc})$	$\lambda(r^2)$
$\mathbf{G_{Email}}$	1005	16064	31.97	0.27	2.59(986)	−1.02(0.81)
$\mathbf{G_{DBLP}}$	317080	1049866	6.62	0.31	6.79(317080)	−2.71(0.95)
$\mathbf{G_{Amazon}}$	334863	925872	5.53	0.21	11.95(334863)	−2.81(0.93)

https://doi.org/10.1371/journal.pone.0290090.t001

Table 2. Graph Clustering as BECBBs: With ⟨P,R,F⟩ where : P = *Precision*($\mathcal{P}$*airs*(Γ), *E*), R = *Recall*($\mathcal{P}$*airs*(Γ), *E*), F = *Fscore*($\mathcal{P}$*airs*(Γ), *E*), with [N, M] where N is the Number of Modules of Γ, M the Length of the biggest Module of Γ, and with (T) the computation time in seconds of Γ.

Graph	$G = G_{Email}$	$G = G_{DBLP}$	$G = G_{Amazon}$
Louvain	⟨0.11, 0.62, 0.18⟩	⟨0.00, 0.84, 0.00⟩	⟨0.00, 0.94, 0.00⟩
	[26, 334] (0*s*)	[212, 22422] (12*s*)	[237, 12810] (6*s*)
Infomap	⟨0.13, 0.60, 0.21⟩	⟨0.13, 0.72, 0.22⟩	⟨0.11, 0.82, 0.20⟩
	[43, 319] (0*s*)	[16997, 811] (2165*s*)	[17265, 380] (1567*s*)
Starling$^{0.000}$	⟨0.16, 0.57, 0.24⟩	⟨0.08, 0.70, 0.15⟩	⟨0.10, 0.80, 0.18⟩
	[63, 213] (9*s*)	[20044, 433] (752*s*)	[20479, 486] (160*s*)
Starling$^{0.125}$	⟨0.18, 0.51, 0.27⟩	⟨0.10, 0.70, 0.17⟩	⟨0.11, 0.80, 0.20⟩
	[72, 140] (5*s*)	[21809, 396] (714*s*)	[22400, 435] (147*s*)
Starling$^{0.250}$	⟨0.26, 0.45, 0.33⟩	⟨0.12, 0.69, 0.20⟩	⟨0.13, 0.78, 0.23⟩
	[102, 98] (2*s*)	[24852, 296] (584*s*)	[25906, 374] (134*s*)
Starling$^{0.375}$	⟨0.36, 0.40, 0.37⟩	⟨0.16, 0.67, 0.26⟩	⟨0.17, 0.76, 0.28⟩
	[154, 84] (1*s*)	[29545, 252] (465*s*)	[31374, 308] (121*s*)
Starling$^{0.500}$	⟨0.49, 0.35, 0.41⟩	⟨0.25, 0.63, 0.36⟩	⟨0.26, 0.72, 0.38⟩
	[235, 72] (1*s*)	[40905, 171] (347*s*)	[44597, 199] (108*s*)
Starling$^{0.625}$	⟨0.63, 0.29, 0.40⟩	⟨0.61, 0.52, 0.56⟩	⟨0.52, 0.59, 0.55⟩
	[284, 52] (1*s*)	[87286, 116] (298*s*)	[80104, 32] (101*s*)
Starling$^{0.750}$	⟨0.69, 0.27, 0.38⟩	⟨0.83, 0.45, 0.58⟩	⟨0.70, 0.49, 0.57⟩
	[319, 47] (1*s*)	[121392, 116] (252*s*)	[115637, 19] (78*s*)
Starling$^{0.875}$	⟨0.75, 0.23, 0.35⟩	⟨0.87, 0.43, 0.58⟩	⟨0.75, 0.45, 0.56⟩
	[327, 30] (0*s*)	[124338, 113] (246*s*)	[121999, 16] (74*s*)
Starling$^{1.000}$	⟨0.77, 0.22, 0.35⟩	⟨0.94, 0.40, 0.56⟩	⟨0.86, 0.38, 0.52⟩
	[378, 30] (0*s*)	[142371, 113] (239*s*)	[153712, 13] (68*s*)

https://doi.org/10.1371/journal.pone.0290090.t002

Let $G_{ER} = (V_{G_{ER}}, E_{G_{ER}})$ a Random Graph built by $Benchmark_{ER}$, $\Gamma_{ER} = \{V\}$ with only one Module, and $Oracle_{ER}(G_{ER}) = \Gamma_{ER} = \{V\}$ the Oracle's method who knows Γ_{ER}. Fig 4 shows the accuracy of the methods according to p considering each Clustering as a *BECBB*. We can see that:

- $Oracle_{ER}$ knows Γ_{ER}, but does not know the concretely constructed edges $E_{G_{ER}}$. Its number of Modules is always = 1. Its *Precision* increases when p increases, because *density* increases. Its *Recall* is always = 1. Its *Fscore* increase;

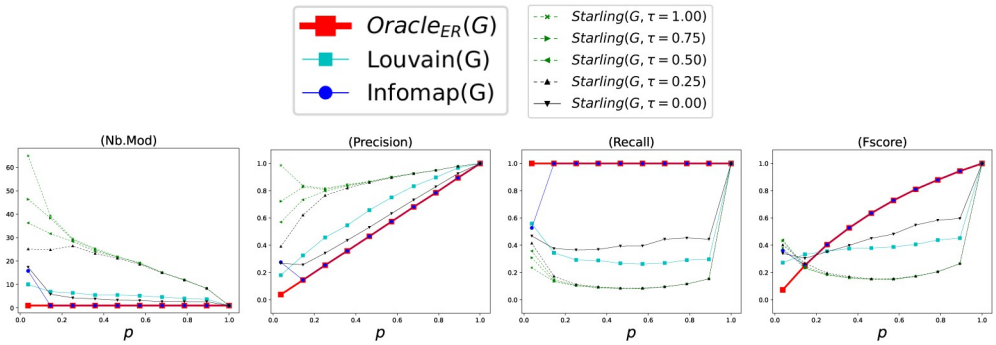


Fig 4. Performance with $Benchmark_{ER}$. Each point (x, y) is the average over 100 Graphs with $p = x$.

https://doi.org/10.1371/journal.pone.0290090.g004

- The best *Precision*s are done with $Starling^{\tau=0.25}$ (but with a lot of Modules);
- The best *Recall*s are done with *Infomap*;
- The best *Fscore*s are done with *Infomap*, except while $p <\approx 0.2$, then it is with $Starling^{\tau=0.25}$;

7.1.2.1 Starling detects the slightly-overconnected regions. To observe more closely the behavior of *Starling*, we draw at Random one of the 100 Graphs with $p = 0.25$ which made it possible to construct the Fig 4. This Graph has a number of vertices $n = 128$, a number of edges $m = 2077$, a mean degree of vertices $\langle k \rangle = 32.45$, a density $d = 0.26$, a Clustering coefficient $C = 0.27$, and a average shortest path length between any two nodes $L = 1.74$.

In this Random Graph with ($n = 128$, $d = 0.26$), $Starling^{\tau=0.00}$ finds four Modules δ_1 with ($n_1 = 51$, $d_1 = 0.33$), δ_2 with ($n_2 = 39$, $d_2 = 0.35$), δ_3 with ($n_3 = 31$, $d_3 = 0.34$), δ_4 with ($n_4 = 7$, $d_4 = 0.76$), where n_i are their number of vertices and d_i are their edge density. So, the four Modules δ_i found by $Starling^{\tau=0.00}$ have a density greater than the one of the entire Graph, specially for δ_4: $d_4 = 0.76 > d = 0.26$.

The phenomenon of overconnected regions is particulary clear in *Terrain-Graphs*, but also occur in Erdős-Rényi Random Graphs. Indeed such Graphs are not completely uniform, they present an *embryo* of structure with slightly-overconnected regions resulting from Random fluctuations (for exemple the Module δ_4 which is clearly overconnected in this Graph).

It is these slightly-overconnected regions present in Random Graphs that are exploited and amplified in [59] to transform a Random Graph into a shaped-like *Terrain-Graph* and that *Starling* detects in a Random Graph, and so accepts as Modules (especially if τ increases). This is why in the Fig 4 the *Precision* of *Starling* is greater than that of $Oracle_{ER}$. It is because the densities of the Modules found by *Starling* are greater than the density of the single Module V of $Oracle_{ER}$ (which increases with p). However the number of edges between the Modules found by *Starling* remains large, this is why the *Recall* of *Starling* stays small (especially if τ increases).

7.1.2.2 Behavior.

(i) *Infomap* usually returns $\Gamma = \{V\}$. Which means: **Infomap identify the absence of strong structures**;

(ii) $Starling^{\tau}$ returns Modules which have a density greater than the one of the entire Graph, the slightly-overconnected regions (especially if τ increases). Which means: **$Starling^{\tau}$ identifies the presence of weak structures**.

7.1.3 Performance on $Benchmark_{LFR}$. In most *Terrain-Graphs*, the distribution of degrees is well approximated by a power law. Similarly, in most *Terrain-Graphs*, the distribution of community sizes is well approximated by a power law [40, 60]. Therefore, in order to produce artificial Graphs with a meaningful group structure similar to most *Terrain-Graphs*, Lancichinetti, Fortunato and Radicchi proposed $Benchmark_{LFR}$ [61] (Code to generate $Benchmark_{LFR}$ Graphs can be downloaded from Andrea Lancichinetti's homepage https://sites.google.com/site/andrealancichinetti/home). The Graphs in $Benchmark_{LFR}$ are parameterized with:

- N their number of vertices;
- k their average degree;
- γ the power law exponent of their degree distribution;

- β the power law exponent of their community sizes distribution;
- $\mu \in [0, 1]$ their mixing parameter: Each vertex shares a fraction $1 - \mu$ of its links with the other vertices of its community and a fraction μ with the other vertices of the Graph.

With $Benchmark_{LFR}$, when the mixing parameter μ is weak, the overconnected regions are well separated from each other, and when μ increases, the overconnected regions are less clear. Therefore, we set $N = 1000$, and $k = 15$ or $k = 25$, and ($\gamma = 2, \beta = 1$) or ($\gamma = 2, \beta = 2$) or ($\gamma = 3, \beta = 1$) and for each of these six configurations, we will study the accuracy of the methods according to μ.

Let $G_{LFR} = (V_{G_{LFR}}, E_{G_{LFR}})$ a Graph built by $Benchmark_{LFR}$, $\Gamma_{G_{LFR}}$ its expected Modules as expected overconnected regions, and $Oracle_{LFR}(G_{LFR}) = \Gamma_{G_{LFR}}$ the Oracle's method which knows the $\Gamma_{G_{LFR}}$ of each G_{LFR}.

We show in Figs 5 and 6 the accuracy of the methods according to μ, considering each Clustering as a *BECBB*. We can see that:

- $Oracle_{LFR}$ knows the $\Gamma_{G_{LFR}}$ of each G_{LFR}, but does not know their concretely constructed edges $E_{G_{LFR}}$. Its number of Modules is always $|\Gamma_{G_{LFR}}|$. Its *Precision* decreases when μ increase, because there are more and more non-edges in the expected Modules, but $Oracle_{LFR}$ does not know it. Its *Recall* decreases when μ increase, because there are more and more edges outside the expected Modules, but $Oracle_{LFR}$ does not know it. Its *Fscore* decreases when μ increase, because its *Precision* and its *Recall* decreases;
- The best *Precision*s are done with $Starling^{\tau=0.25}$, but with a lot of Modules when the overconnected regions are less clear (because here again (see section 7.1.2.2) *Starling* identifies the presence of the large number of small slightly-overconnected regions as Modules present in these Graphs);
- The best *Recall*s are done with *Infomap*, but with very few Modules, and often only one, when the overconnected regions are less clear (because there is no way to compress the description of the path of a Random walker in these Graphs);
- The best *Fscore*s are done with *Infomap* and $Starling^{\tau=0.25}$ except when the overconnected regions are less clear, then it is with $Starling^{\tau=0.25}$.

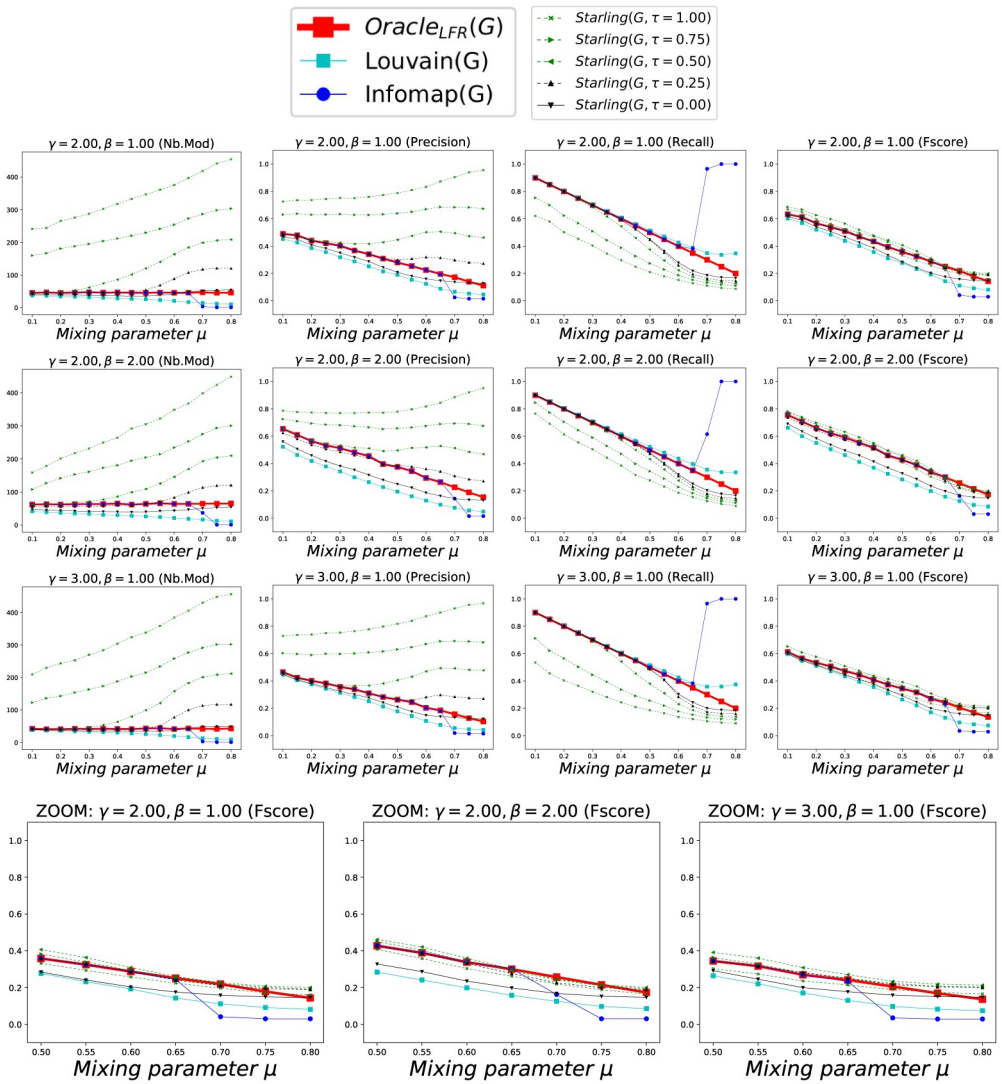
7.2 Starling versus SGC

7.2.1 Performance on Real Terrain-Graphs. In this section, we compare $Starling(G, \tau)$ with respect to $SGC(G, \kappa)$, κ varying, on three little *Terrain-Graphs*:

- G_{Email}: The Graph seen in section 7.1.1;
- $G_{dblp_{811}}$: The subGraph of G_{dblp} on the vertices of the larger Module of $Infomap(G_{dblp})$ which has 811 vertices;
- $G_{amazon_{380}}$: The subGraph of G_{amazon} on the vertices of the larger Module of $Infomap(G_{amazon})$ which has 380 vertices;

Table 3 illustrates the pedigrees of these *Terrain-Graphs*.

The dataset describing G_{Email} contains "ground-truth" community memberships of the nodes $\mathcal{C} : V_{G_{Email}} \rightarrow D$. Each individual belongs to exactly one of 42 departments $D = \{d_1, \ldots d_{42}\}$ at the research institute from which the emails are extracted. Let Γ_{Dep} the Gold-



(a) Zoom: $\mu \in [0.5, 0.8]$ (b) Zoom: $\mu \in [0.5, 0.8]$ (c) Zoom: $\mu \in [0.5, 0.8]$

Fig 5. Performance with Benchmark$_{\mathbf{LFR}}$ (k = 15). Each point (x, y) is the average over 100 Graphs with $\mu = x$. Fig 5 (a)–5(c) are zooms on the *Fscores* when the overconnected regions are less clear (i.e. when we can no longer trust $Oracle_{LFR}(G_{LFR}) = \Gamma_{G_{LFR}}$).

https://doi.org/10.1371/journal.pone.0290090.g005

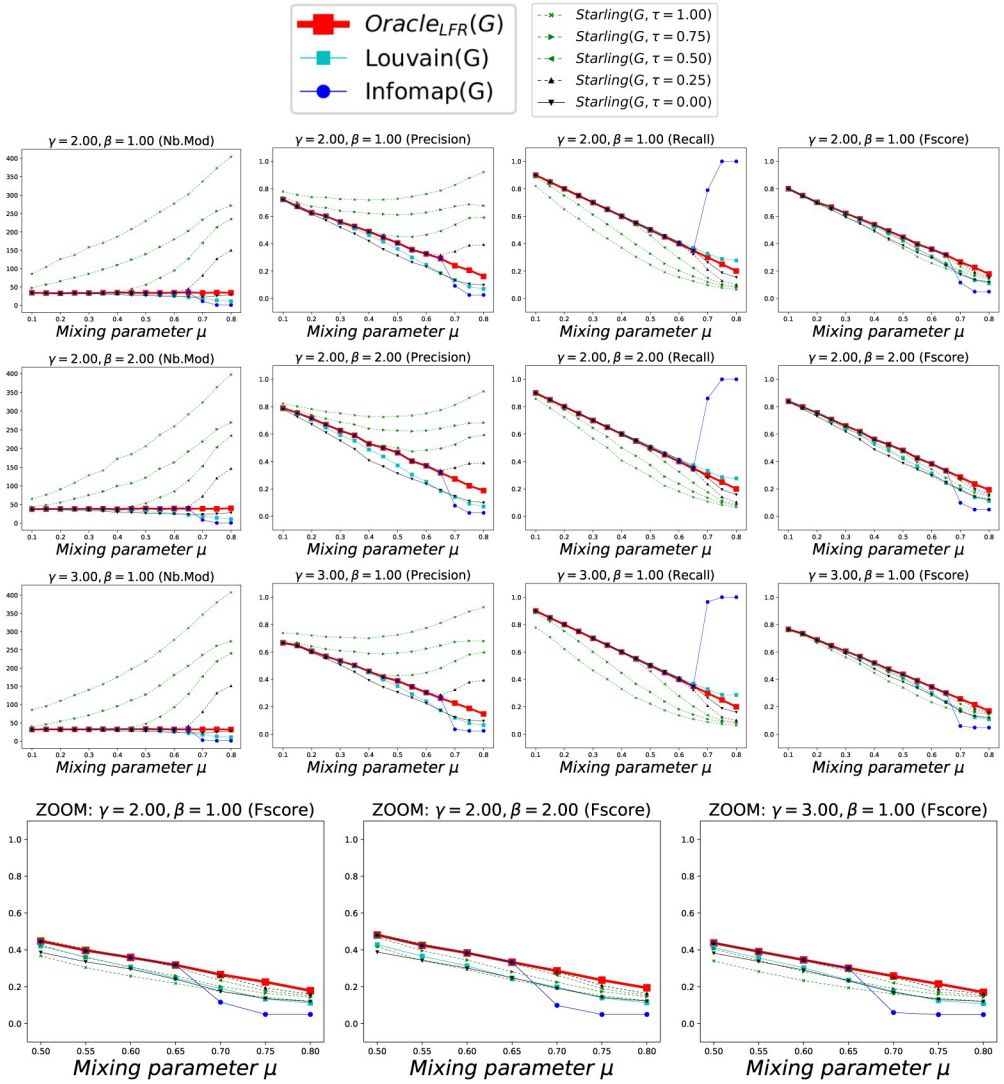
Standart partition of $V_{G_{Email}}$ such:

$$\Gamma_{Dep} = \bigcup_{d_i \in D} \{\{j, \text{ such } \mathcal{C}(j) = d_i\}\}$$

We can therefore evaluate the quality of a Clustering by partition on G_{Email} according to two kinds of truths:

Intrinsic-Truth: The edges of G_{Email} as we did in the previous sections with *Precision*, *Recall*, *Fscore* respectively defined by the formulas 22, 23 and 24;

Extrinsic-Truth: $\mathcal{P}airs(\Gamma_{Dep})$ by replacing E by $\mathcal{P}airs(\Gamma_{Dep})$ in the three formulas 22, 23 and 24.



(a) Zoom: $\mu \in [0.5, 0.8]$ (b) Zoom: $\mu \in [0.5, 0.8]$ (c) Zoom: $\mu \in [0.5, 0.8]$

Fig 6. Performance with Benchmark$_{\mathbf{LFR}}$ (k = 25). Each point (x, y) is the average over 100 Graphs with $\mu = x$. Fig 6 (a)–6(c) are zooms on the *Fscores* when the overconnected regions are less clear (i.e. when we can no longer trust $Oracle_{LFR}(G_{LFR}) = \Gamma_{G_{LFR}}$).

https://doi.org/10.1371/journal.pone.0290090.g006

Table 3. Pedigrees: The notations are identical to those of Table 1.

Graph	n	m	$\langle k \rangle$	C	$L_{lcc}(n_{lcc})$	$\lambda(r^2)$
G_{Email}	1005	16064	31.97	0.27	2.59(986)	−1.02(0.81)
$G_{dblp_{811}}$:	811	3774	9.31	0.19	3.33(811)	−1.35(0.91)
$G_{amazon_{380}}$	380	959	5.06	0.06	2.92(380)	−1.11(0.66)

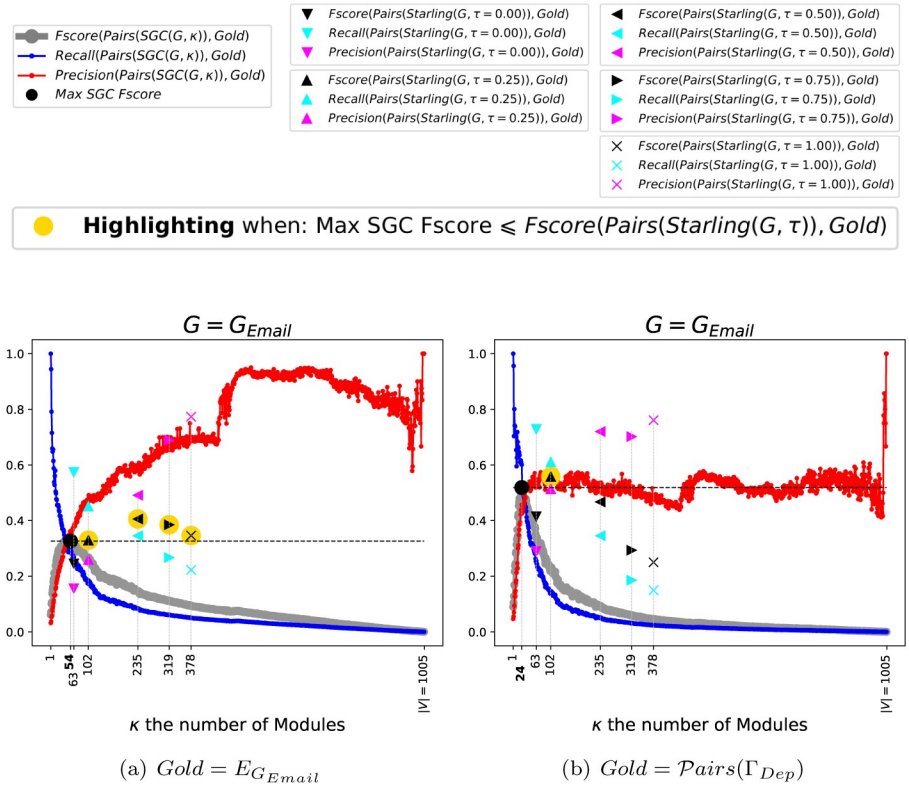
https://doi.org/10.1371/journal.pone.0290090.t003

Fig 7 shows the performances of $SGC(G_{Email}, \kappa)$, one hand according to the *Intrinsic-Truth* $E_{G_{Email}}$ in Fig 7(a), and on the other hand according to the *Extrinsic-Truth* $\mathcal{P}airs(\Gamma_{Dep}) = \bigcup_{\gamma \in \Gamma_{Dep}} \mathcal{P}_2^{\gamma}$ in Fig 7(b). We can see that:

- According to the *Intrinsic-Truth* $E_{G_{Email}}$ in Fig 7(a): $Starling(G_{Email}, \tau = 0.25)$, with 102 Modules, gets *Precision* = 0.26, *Recall* = 0.45, *Fscore* = 0.33. The maximum *Fscore* of *SGC* is geted for $\kappa = 54$ with *Precision* = 0.36, *Recall* = 0.30, *Fscore* = 0.33. On the other hand for $\tau \in \{0.50, 0.75, 1.00\}$, *Starling* gets a beter *Fscore* than the best *Fscore* of *SGC*.
  As BECBB: $\exists \tau \in [0, 1]$ such *Starling* gets a beter *Fscore* than the best *Fscore* of *SGC*.
- According to the *Extrinsic-Truth* $\mathcal{P}airs(\Gamma_{Dep})$ in Fig 7(b): $Starling(G_{Email}, \tau = 0.25)$ gets *Precision* = 0.51, *Recall* = 0.61, *Fscore* = 0.56. The maximum *Fscore* of *SGC* is geted for $\kappa = 24$ with *Precision* = 0.46, *Recall* = 0.60, *Fscore* = 0.52.
  According to $\mathcal{P}$**airs**$(\Gamma_{\mathbf{Dep}})$: $\exists \tau \in [0, 1]$ such *Starling* gets a beter *Fscore* than the best *Fscore* of *SGC*.

Fig 8 shows the performances as *BECBB*s of $SGC(G_{dblp_{811}}, \kappa)$ and $SGC(G_{amazon_{380}}, \kappa)$, according to their *Intrinsic-Truth* respectively $E_{G_{dblp_{811}}}$ and $E_{G_{amazon_{380}}}$. We can see that: $\exists \tau \in [0, 1]$ such *Starling* gets a beter *Fscore* than the best *Fscore* of *SGC*.

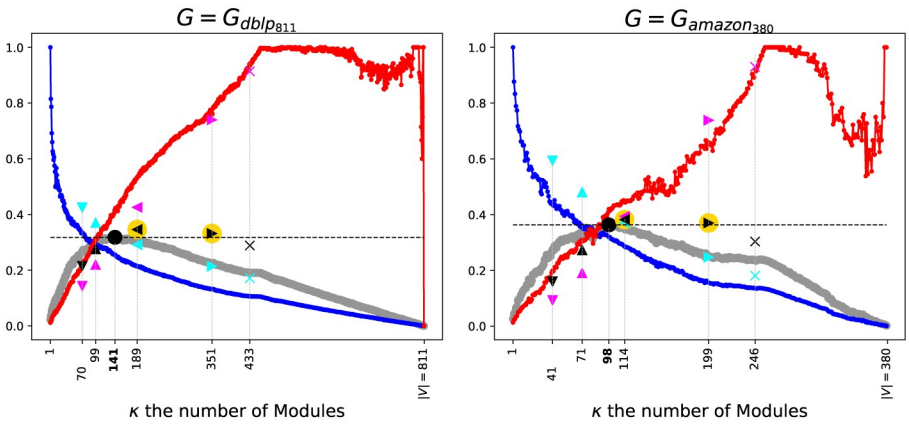
7.2.1.1 Extrinsic-Truth according to Intrinsic-Truth of G_{Email}. Because $Precision(\mathcal{P}airs(\Gamma_{Dep}), E_{G_{Email}}) = 0.23$, $Recall(\mathcal{P}airs(\Gamma_{Dep}), E_{G_{Email}}) = 0.34$,



(a) $Gold = E_{G_{Email}}$ (b) $Gold = \mathcal{P}airs(\Gamma_{Dep})$

Fig 7. Performance of SGC($\mathbf{G_{Email}} = (\mathbf{V_{G_{Email}}}, \mathbf{E_{G_{Email}}})$, κ), κ varying. According to the intrinsic truth $E_{G_{Email}}$ in Fig 7(a), and in Fig 7(b) according to the extrinsic truth $\mathcal{P}airs(\Gamma_{Dep}) = \cup_{\gamma \in \Gamma_{Dep}} \mathcal{P}_2^{\gamma}$.

https://doi.org/10.1371/journal.pone.0290090.g007



(a) $Gold = E_{G_{dblp_{811}}}$ (b) $Gold = E_{G_{amazon_{380}}}$

Fig 8. Performance of SGC(G = (V,E), κ) according to the intrinsic truth E, κ varying.

https://doi.org/10.1371/journal.pone.0290090.g008

$Fscore(\mathcal{P}airs(\Gamma_{Dep}), E_{G_{Email}}) = 0.27$, as *BECBB*, Γ_{Dep} is less efficient than $Starling(G_{Email}, \tau = 0.25)$ or $SGC(G_{Email}, \kappa = 54)$ the best *BECBB* of *SGC*.

That is to say that Gold-Standards are not always the best *BECBB*s, we can not always trust Gold-Standards provided by Benchmarks or built using human assessors, which as showed in [62], generaly do not always agree with each other, even when their judgements are based on the same protocol.

In our present example with G_{Email}, we can think that two individuals from the same department can communicate in real life more often than two individuals from different departments: *Two individuals from the same department do not necessarily need to communicate more by email than two individuals from different departments.*

7.2.2 Performance on Benchmark$_{ER}$. Because we need to know κ the number of groups of vertices in advance in the Input of *SGC*, to be able to compare *Starling* with *SGC* we define: $SGC^{\tau}(G) = SGC(G, \kappa = |Starling(G, \tau)|)$.

Let $G_{ER} = (V_{G_{ER}}, E_{G_{ER}})$ a Random Graph built by $Benchmark_{ER}$, $\Gamma_{ER} = \{V\}$ with only one Module, and $Oracle_{ER}(G_{ER}) = \Gamma_{ER} = \{V\}$ the Oracle's method who knows Γ_{ER}.

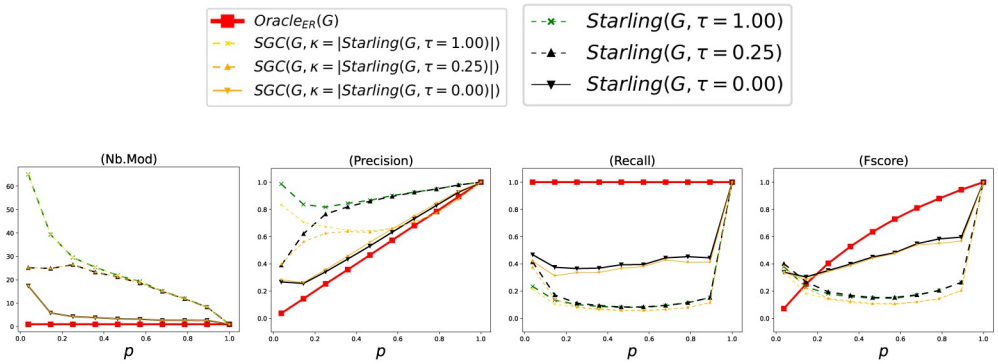


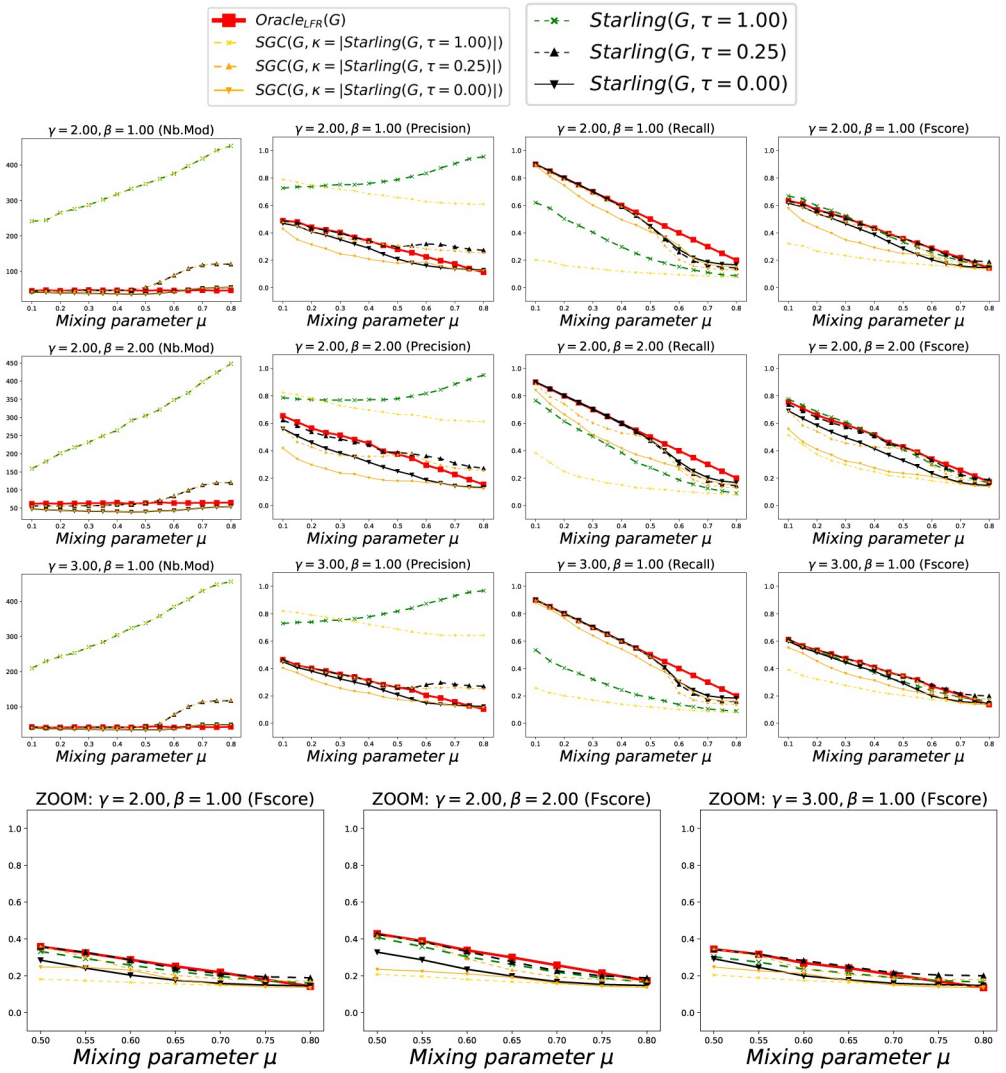
Fig 9. Performance with Benchmark$_{ER}$. Each point (x, y) is the average over 100 Graphs with $p = x$.

https://doi.org/10.1371/journal.pone.0290090.g009

Fig 9 shows the accuracy of the methods according to p considering each Clustering as a *BECBB*. We can see that: $\forall\tau \in \{0.00, 0.25, 1.00\}$ on these Figures, the *Fscore*s geted by *Starling*(G, τ) are always equal or greater than the *Fscore*s geted by $SGC^{\tau}(G)$.

7.2.3 Performance on Benchmark$_{\mathbf{LFR}}$. Let $G_{LFR} = (V_{G_{LFR}}, E_{G_{LFR}})$ a Graph built by *Benchmark*$_{LFR}$, $\Gamma_{G_{LFR}}$ its expected Modules as expected overconnected regions, and $Oracle_{LFR}(G_{LFR}) = \Gamma_{G_{LFR}}$ the Oracle's method which knows the $\Gamma_{G_{LFR}}$ of each G_{LFR}.

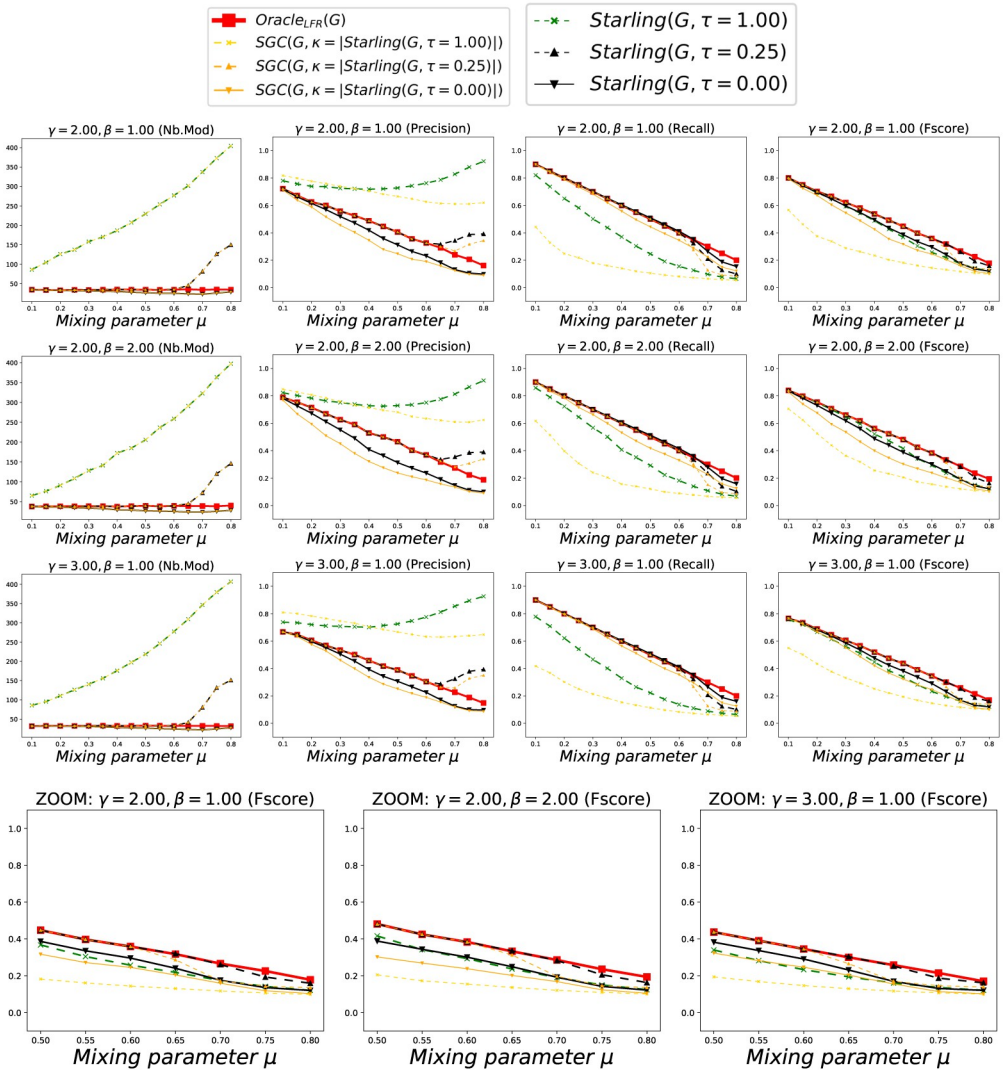
We show in Figs 10 and 11 the accuracy of methods $SGC^{\tau}(G)$ and *Starling*(G, τ) according to μ, considering each Clustering as a *BECBB*. We can see that $\forall\tau \in \{0.00, 0.25, 1.00\}$ on these Figures, the *Fscore*s geted by *Starling*$(G, 0.25)$ are always equal or greater than the *Fscore*s geted by $SGC^{\tau}(G)$.



(a) Zoom: $\mu \in [0.5, 0.8]$ (b) Zoom: $\mu \in [0.5, 0.8]$ (c) Zoom: $\mu \in [0.5, 0.8]$

Fig 10. Performance with Benchmark$_{\mathbf{LFR}}$ (k = 15). Each point (x, y) is the average over 100 Graphs with $\mu = x$. Fig 10 (a)–10(c) are zooms on the *Fscore*s when the overconnected regions are less clear (i.e. when we can no longer trust $Oracle_{LFR}(G_{LFR}) = \Gamma_{G_{LFR}}$).

https://doi.org/10.1371/journal.pone.0290090.g010



(a) Zoom: $\mu \in [0.5, 0.8]$ (b) Zoom: $\mu \in [0.5, 0.8]$ (c) Zoom: $\mu \in [0.5, 0.8]$

Fig 11. Performance with Benchmark$_{\mathbf{LFR}}$ (k = 25). Each point (x, y) is the average over 100 Graphs with $\mu = x$. Fig 11 (a)–11(c) are zooms on the *Fscore*s when the overconnected regions are less clear (i.e. when we can no longer trust $Oracle_{LFR}(G_{LFR}) = \Gamma_{G_{LFR}}$).

https://doi.org/10.1371/journal.pone.0290090.g011

8 Discussion

8.1 Choosing the τ parameter of Starling

When using a Benchmark $\mathbb{B}$ to evaluate the performance of methods on a Graph $G_{\mathbb{B}} = (V_{G_{\mathbb{B}}}, E_{G_{\mathbb{B}}})$, the Oracle's method $Oracle_{\mathbb{B}}$ knows the expected overconnected regions $\Gamma_{G_{\mathbb{B}}}$ but do not knows the concretely constructed edges $E_{G_{\mathbb{B}}}$. Therefore, when the overconnected regions are less clear, as *BECBB* (with Gold $= E_{G_{\mathbb{B}}}$, Intrinsic-Truth), some methods may outperform the $Oracle_{\mathbb{B}}$ method. This happens especially with the $Starling^{\tau}$ method if the τ parameter has been chosen appropriately.

We have seen in Formula 17 that the closer the $\tau \in [0, 1]$ parameter is to 1, the less *Confluence* is taken into account in Q^{τ}_{Conf}. With *Terrain-Graphs*, we propose using $\tau = 0.25$ as a first

approach by default, then decreasing τ if we want to promote *Recall* (because it has the effect of decreasing the number of Modules and of increasing their sizes) or increasing τ if we want to promote *Precision* (because it has the effect of increasing the number of Modules and of decreasing their sizes).

8.2 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 to transform a Random Graph into a shaped-like *Terrain* – *Graph* was done in [59], 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 *Starling*.

For example we can build the Graph $G^{2\star}_{toy}$ from G^2_{toy} by inserting a new vertex in the middle of each edge. Fig 12 illustrates the optimal Clusterings on G^2_{toy} and on $G^{2\star}_{toy}$ for $Q^{0.0}_{Conf}$ with $t = 3$ and also with $t = 6$, allowing us to see that:

On $\mathbf{G^{2\star}_{toy}}$ with $\mathbf{t = 6}$:

$$\delta^{\star 1}_{Conf} = \{0, 4, 5, 6, cut(0/4), cut(0/5), cut(0/6), cut(4/5), cut(4/6), cut(5/6)\};$$

$$\delta^{\star 2}_{Conf} = \{1, 2, 3, cut(0/1), cut(0/2), cut(0/3), cut(1/2), cut(1/3), cut(2/3)\};$$

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

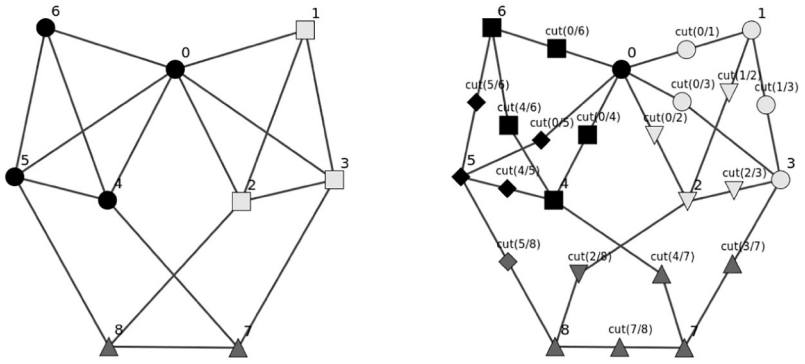
On $\mathbf{G^2_{toy}}$ with $\mathbf{t = 3}$:

$$\delta^1_{Conf} = \{0, 4, 5, 6\} \subset \delta^{\star 1}_{Conf};$$

$$\delta^2_{Conf} = \{1, 2, 3, \} \subset \ \delta^{\star 2}_{Conf};$$

$$\delta^3_{Conf} = \{7, 8\} \subset \delta^{\star 3}_{Conf}.$$

The length of Random walks t could be advantageously chosen taking into account L, the average number of edges on the shortest path between two vertices.



(a) G^2_{toy} (b) $G^{2\star}_{toy}$

Fig 12. 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 $t = 6$.

https://doi.org/10.1371/journal.pone.0290090.g012

8.3 Directed graphs

If G is a positively weighted Graph by $W = \{w_{i,j}$ such $\{i, j\} \in E\}$, then we can apply Q_{Conf} and *Starling* by replacing Eqs 7 and 10 by 27 and 28 respectively:

$$[G] = (g_{i,j})_{i,j\in V} \quad \text{with} \quad g_{i,j} = \begin{cases} \frac{w_{i,j}}{\sum_{k\in V} w_{i,k}} & \text{if } \{i,j\} \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{27}$$

$$Conf^t(G, i, j) = \begin{cases} 0 \text{ if } i = j, \\ \frac{P^t_G(i \rightsquigarrow j) - \frac{\sum_{k\in V} w_{k,j}}{\sum_{w\in W} w}}{P^t_G(i \rightsquigarrow j) + \frac{\sum_{k\in V} w_{k,j}}{\sum_{w\in W} w}} \text{ otherwise.} \end{cases} \tag{28}$$

If G is a directed Graph, one can also consider using a variant of page rank [63–65] in place of Eq 8.

9 Conclusions and perspectives

In this paper, we defined *Confluence*, a mesoscopic vertex closeness measure based on short Random walks, which brings together vertices from the same overconnected region, and separates vertices coming from two distinct overconnected regions. Then we used *Confluence* to define Q^{τ}_{Conf}, a new Clustering quality function, where the $\tau \in [0, 1]$ parameter is a handle on the *Precision* & *Recall*, the size and the number of Modules. With a small toy Graphs, we showed that optimal Clusterings for Q^{τ}_{Conf} improve the *Fscore* of the optimal Clusterings for *Modularity*.

We then introduced *Starling*(G, τ), a new heuristic based on the *Confluence* of edges designed to optimize Q^{τ}_{Conf} on a Graph G. On the same little toy Graph, we showed that *Starling*(G, τ) finds an optimal Clustering for Q^{τ}_{Conf}.

Comparing *Starling*(G, τ) to *SGC*(G, κ), *Infomap*, and *Louvain* we show that:

- Performance with the *Terrain-Graphs* studied in this paper:

  Louvain(G): Returns Clusterings with a low *Fscore*, caused by a to much low *Precision* despite a large *Recall*;

  Infomap(G): Tends to favor *Recall* with good *Fscore*;

  SGC(G, κ): Returns Clusterings with a good *Fscore* if we know the good number of groups of vertices κ in advance;

  Starling(G, τ): Tends to favor *Precision* with good *Fscore*. $\exists\tau \in [0, 1]$ (usually around $\tau \approx 0.25$) such that the *Fscore* of the Clusterings returned by *Starling* is greater than the *Fscore*s of the Clusterings returned by *Infomap* and greater than the best *Fscore*s of the Clusterings returned by *SGC*.

- Performance with $Benchmark_{ER}$:

  SGC(G, $\kappa = |Starling(G, \tau)|$): *Fscore*(*SGC*($G$, $\kappa = |Starling(G, \tau)|$), $\{V\}$) $\approx$ *Fscore*(*Starling*(G, τ), $\{V\}$);

  (i) *Infomap* usually returns $\Gamma = \{V\}$. Which means: *Infomap* identify the absence of strong structures;

(ii) *Starling*$^{\tau}$ returns Modules which have a density greater than the one of the entire Graph, the slightly-overconnected regions (especially if τ increases). Which means: *Starling*$^{\tau}$ identifies the presence of weak structures.

- Performance with $Benchmark_{LFR}$:
  - When the overconnected regions are clear: On the one hand, *Starling*(G, $\tau = 0.25$) gets equivalent *Fscore*s than these of *Infomap* (see Figs 6 and 7). On the other hand *Starling*(G, τ) gets equivalent or greater *Fscore*s than these of *SGC*(G, $\kappa = |Starling(G, \tau)|$) (see Figs 10 and 11);
  - When the overconnected regions become less clear, *Starling* favors *Precision* while *Infomap* favor *Recall*:

(1) On the one hand, *Starling*(G, $\tau = 0.25$) gets then greater *Fscore*s than these of *Infomap* (see Figs 5(a)–5(c) and 6(a)–6(c)). That's because even in Non Erdös and Rényi Graphs, *Starling*$^{\tau}$ identifies the presence of weak structures thanks to its (*ii*) behavior, whereas *Infomap* identify the absence of strong structures because its (*i*) behavior.
On the other hand, *Starling*(G, $\tau = 0.25$) gets then equivalent or greater *Fscore*s than these of *SGC*(G, $\kappa = |Starling(G, \tau = 0.25)|$) (see Figs 10(a)–10(c) and 11(a)–11(c)).

(2) Often (τ dependent) *Starling*(G, τ), thanks to its (*ii*) behavior, is able to get larger *Fscore*s than these of *Oracles* that would only knows their expected overconnected regions (concretely slightly-overconnected), ignoring E their concretely constructed edges. *SGC*(G, $\kappa = |Starling(G, \tau = 0.25)|$) can also succeed (see Fig 10(a) and 10(c)), but still weaker than *Starling*(G, $\tau = 0.25$), whereas *Infomap* can never succeed, because its (*i*) behavior.

To sum up: If we know the good number of groups of vertices κ in advance then we can use *SGC*. If we do not know it, then we can use *Infomap* on the one hand with *Starling* on the other hand wich are complementary:

Infomap tend to favor *Recall* with good *Fscore* and is able to identify the absence of strong structures;

Starling$^{\tau=0.25\ by\ default}$ tends to favor *Precision* with good *Fscore* and is able to identify the presence of weak structures. Then if we want to promote *Recall* with a smaller number of larger Modules, we can decrease τ, and if we want to promote *Precision* with a greater number of smaller Modules, we can increase τ.

Our follow-up work: We will focus on the role on the ouputs of *Starling*, played by the length of the Random walks in computing *Confluence*, as well as the development of a Clustering method based on *Confluence* able to detect Clustering in Graphs accounting for edge directions and edge weights, its returns communities possibly overlapping.

Supporting information

S1 Fig.
(TIF)

Acknowledgments

I thank the editors and the anonymous reviewers for their professional and valuable suggestions. I thank also Korantin Auguste for his suggestions and proofreading.

Author Contributions

Conceptualization: Bruno Gaume.

Data curation: Bruno Gaume.

Formal analysis: Bruno Gaume.

Funding acquisition: Bruno Gaume.

Investigation: Bruno Gaume.

Methodology: Bruno Gaume.

Project administration: Bruno Gaume.

Resources: Bruno Gaume.

Software: Bruno Gaume.

Supervision: Bruno Gaume.

Validation: Bruno Gaume.

Visualization: Bruno Gaume.

Writing – original draft: Bruno Gaume.

References

1. Watts DJ, Strogatz SH. Collective Dynamics of Small-World Networks. Nature. 1998; 393:440–442. https://doi.org/10.1038/30918 PMID: 9623998
2. Albert R, Barabasi AL. Statistical Mechanics of Complex Networks. Reviews of Modern Physics. 2002; 74:74–47. https://doi.org/10.1103/RevModPhys.74.47
3. Newman MEJ. The Structure and Function of Complex Networks. SIAM Review. 2003; 45:167–256. https://doi.org/10.1137/S003614450342480
4. Aittokallio T, Schwikowski B. Graph-based methods for analysing networks in cell biology. Briefings in bioinformatics. 2006; 7(3):243–255. https://doi.org/10.1093/bib/bbl022 PMID: 16880171
5. Bonacich P, Lu P. Introduction to mathematical sociology. Princeton University Press; 2012.
6. Steyvers M, Tenenbaum JB. The Large-Scale Structure of Semantic Networks: Statistical Analyses and a Model of Semantic Growth. Cognitive Science. 2005; 29(1):41–78. https://doi.org/10.1207/s15516709cog2901_3 PMID: 21702767
7. Korlakai Vinayak R, Oymak S, Hassibi B. Graph Clustering With Missing Data: Convex Algorithms and Analysis. In: Ghahramani Z, Welling M, Cortes C, Lawrence N, Weinberger KQ, editors. Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc.; 2014.
8. Nugent R, Meila M. An overview of clustering applied to molecular biology. Statistical methods in molecular biology. 2010; p. 369–404. https://doi.org/10.1007/978-1-60761-580-4_12 PMID: 20652512
9. Wolf FA, Hamey FK, Plass M, Solana J, Dahlin JS, Gottgens B, et al. PAGA: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. Genome biology. 2019; 20(1):1–9. https://doi.org/10.1186/s13059-019-1663-x PMID: 30890159
10. Zesch T, Muller C, Gurevych I. Using wiktionary for computing semantic relatedness. In: Proceedings of the 23rd national conference on Artificial intelligence—Volume 2. Chicago, Illinois: AAAI Press; 2008. p. 861–866.
11. de Jesus Holanda A, Pisa IT, Kinouchi O, Martinez AS, Ruiz EES. Thesaurus as a complex network. Physica A: Statistical Mechanics and its Applications. 2004; 344(3-4):530–536. https://doi.org/10.1016/j.physa.2004.06.025
12. Li HJ, Wang L, Zhang Y, Perc M. Optimization of identifiability for efficient community detection. New Journal of Physics. 2020; 22(6). https://doi.org/10.1088/1367-2630/ab8e5e
13. Tran C, Shin WY, Spitz A. Community Detection in Partially Observable Social Networks. ACM Transactions on Knowledge Discovery from Data (TKDD). 2017; 16:1–24. https://doi.org/10.1145/3461339
14. Frey BJ, Dueck D. Clustering by passing messages between data points. science. 2007; 315 (5814):972–976. https://doi.org/10.1126/science.1136800 PMID: 17218491

15. Aditya G, Jure L. node2vec: Scalable Feature Learning for Networks. CoRR. 2016;1607.00653.
16. Yang Y, Wei H, Sun ZQ, Li GY, Zhou Y, Xiong H, et al. S2OSC: A Holistic Semi-Supervised Approach for Open Set Classification. ACM Trans Knowl Discov Data. 2021; 16(2). https://doi.org/10.1145/3468675
17. Cantini R, Marozzo F, Bruno G, Trunfio P. Learning Sentence-to-Hashtags Semantic Mapping for Hashtag Recommendation on Microblogs. Association for Computing Machinery. 2021; 16(2).
18. Zhang Z, Zhang L, Yang D, Yang L. KRAN: Knowledge Refining Attention Network for Recommendation. Association for Computing Machinery. 2021; 16(2).
19. Wu H, Ma T, Wu L, Xu F, Ji S. Exploiting Heterogeneous Graph Neural Networks with Latent Worker/Task Correlation Information for Label Aggregation in Crowdsourcing. Association for Computing Machinery. 2021; 16(2).
20. O'hare K, Jurek-Loughrey A, De Campos C. High-Value Token-Blocking: Efficient Blocking Method for Record Linkage. Association for Computing Machinery. 2021; 16(2).
21. Steinhaus H. Sur la division des corps matériels en parties. Bulletin de l'academie polonaise des sciences. 1957; 4(12):801–804.
22. Luxburg U. A Tutorial on Spectral Clustering. Statistics and Computing. 2007; 17(4):395–416. https://doi.org/10.1007/s11222-007-9033-z
23. Chung FRK. Spectral Graph Theory. American Mathematical Society; 1997.
24. Bolla M. Relations between spectral and classification properties of multigraphs. DIMACS, Center for Discrete Mathematics and Theoretical Computer Science; 1991.
25. Hahn G, Sabidussi G. Graph symmetry: algebraic methods and applications. vol. 497. Springer Science & Business Media; 2013.
26. Still S, Bialek W. How Many Clusters? An Information-Theoretic Perspective. Neural Computation. 2004; 16:2483–2506. https://doi.org/10.1162/0899766042321751 PMID: 15516271
27. C F, R A E. Model-Based Clustering, Discriminant Analysis, and Density Estimation. Journal of the American Statistical Association. 2002; 97:611–631. https://doi.org/10.1198/016214502760047131
28. Tibshirani R, Walther G, Hastie T. Estimating the number of clusters in a dataset via the Gap statistic. 2000; 63:411–423.
29. Ben-Hur A, Elisseeff A, Guyon I. A Stability Based Method for Discovering Structure in Clustered Data. Pacific Symposium on Biocomputing Pacific Symposium on Biocomputing. 2002; p. 6–17. PMID: 11928511
30. Lange T, Roth V, Braun ML, Buhmann JM. Stability-Based Validation of Clustering Solutions. Neural Computation. 2004; 16:1299–1323. https://doi.org/10.1162/089976604773717621 PMID: 15130251
31. Ben-David S, von Luxburg U, Pal D. A Sober Look at Clustering Stability. In: COLT 2006. Max-Planck-Gesellschaft. Berlin, Germany: Springer; 2006. p. 5–19.
32. Knuth DE. The Art of Computer Programming: Fundamental algorithms. The Art of Computer Programming. Addison -Wesley; 1968.
33. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008; 2008(10). https://doi.org/10.1088/1742-5468/2008/10/P10008
34. Newman MEJ, Girvan M. Finding and evaluating community structure in networks. Physical Review E. 2004; 69(2). https://doi.org/10.1103/PhysRevE.69.026113 PMID: 14995526
35. Fortunato S, Barthelemy M. Resolution limit in community detection. Proceedings of the National Academy of Sciences. 2006; 104(1):36–41. https://doi.org/10.1073/pnas.0605965104 PMID: 17190818
36. Kumpula JM, Saramaki J, Kaski K, Kertesz J. Limited resolution in complex network community detection with Potts model approach. The European Physical Journal B. 2007; 56(1):41–45. https://doi.org/10.1140/epjb/e2007-00088-4
37. Reichardt J, Bornholdt S. Statistical mechanics of community detection. Physical Review E. 2006; 74 (1). https://doi.org/10.1103/PhysRevE.74.016110 PMID: 16907154
38. Arenas A, Fernández A, Gómez S. Analysis of the structure of complex networks at different resolution levels. New Journal of Physics. 2008; 10(5):053039. https://doi.org/10.1088/1367-2630/10/5/053039
39. Lancichinetti A, Fortunato S. Limits of modularity maximization in community detection. Physical Review E. 2011; 84(6). https://doi.org/10.1103/PhysRevE.84.066122 PMID: 22304170
40. Clauset A, Newman MEJ, Moore C. Finding community structure in very large networks. Physical Review E. 2004; 70(6). https://doi.org/10.1103/PhysRevE.70.066111 PMID: 15697438
41. Radicchi F, Castellano C, Cecconi F, Loreto V, Parisi D. Defining and identifying communities in networks. Proceedings of the National Academy of Sciences. 2004; 101(9):2658–2663. https://doi.org/10.1073/pnas.0400054101 PMID: 14981240

42. Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences. 2008; 105(4):1118–1123. https://doi.org/10.1073/pnas.0706851105 PMID: 18216267

43. Grunwald PD. The minimum description length principle. MIT press; 2007.

44. Gaume B, Duvignau K, Navarro E, Desalle Y, Cheung H, Hsieh SK, et al. Skillex: a graph-based lexical score for measuring the semantic efficiency of used verbs by human subjects describing actions. vol. 55 of Revue TAL: numéro spécial sur Traitement Automatique des Langues et Sciences Cognitives (55-3). ATALA (Association pour le Traitement Automatique des Langues); 2016. Available from: https://hal.archives-ouvertes.fr/hal-01320416.

45. Bollobas B. Modern Graph Theory. Springer-Verlag New York Inc.; 2002.

46. Stewart GW. Perron-Frobenius theory: a new proof of the basics. College Park, MD, USA; 1994.

47. Gaume B. Random walks in lexical small worlds. Revue I3—Information Interaction Intelligence. 2004; 4(3).

48. Brandes U, Delling D, Gaertler M, Gorke R, Hoefer M, Nikoloski Z, et al. On Modularity Clustering. IEEE Transactions on Knowledge and Data Engineering. 2008; 20(2):172–188. https://doi.org/10.1109/TKDE.2007.190689

49. Meila M. Comparing clusterings. In: Proc. of COLT 03; 2003. Available from: http://www.stat.washington.edu/mmp/www.stat.washington.edu/mmp/Papers/compare-colt.pdf.

50. Rosenberg A, Hirschberg J. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). Prague, Czech Republic: Association for Computational Linguistics; 2007. p. 410–420. Available from: https://aclanthology.org/D07-1043.

51. Amigo E, Gonzalo J, Artiles J, Verdejo F. A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints. Inf Retr. 2009; 12(4):461–486. https://doi.org/10.1007/s10791-008-9066-8

52. Hubert L, Arabie P. Comparing partitions. Journal of Classification. 1985; 2(1):193–218. https://doi.org/10.1007/BF01908075

53. Chakraborty T, Dalmia A, Mukherjee A, Ganguly N. Metrics for Community Analysis: A Survey. ACM Comput Surv. 2017; 50(4). https://doi.org/10.1145/3091106

54. Benson AR, Kumar R, Tomkins A. Sequences of Sets. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD'18. New York, NY, USA: Association for Computing Machinery; 2018. p. 1148–1157.

55. Leskovec J, Kleinberg J, Faloutsos C. Graph Evolution: Densification and Shrinking Diameters. ACM Trans Knowl Discov Data. 2007; 1(1):2–es. https://doi.org/10.1145/1217299.1217301

56. Leskovec J, Krevl A. SNAP Datasets: Stanford Large Network Dataset Collection; 2014. PMID: 25327001

57. Erdös P, Rényi A. On Random Graphs I. Publicationes Mathematicae Debrecen. 1959; 6:290–297.

58. Erdös P, Rényi A. On the evolution of random graphs. Publ Math Inst Hungary Acad Sci. 1960; 5:17–61.

59. Gaum B, Mathieu F, Navarro E. Building Real-World Complex Networks by Wandering on Random Graphs. vol. 10. Revue I3—Information Interaction Intelligence, Cepadues; 2010. p. 73–91.

60. Palla G, Derényi I, Farkas I, Vicsek T. Uncovering the overlapping community structure of complex networks in nature and society. Nature. 2005; 435(7043):814–818. https://doi.org/10.1038/nature03607 PMID: 15944704

61. Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. Physical Review E. 2008; 78(4). https://doi.org/10.1103/PhysRevE.78.046110 PMID: 18999496

62. Murray GC, Green R. Lexical Knowledge and Human Disagreement on a WSD Task. Computer Speech & Language. 2004; 18(3):209–222. https://doi.org/10.1016/j.csl.2004.05.001

63. Gaume B, Mathieu F. PageRank Induced Topology for Real-World Networks; 2016. Available from: https://hal.archives-ouvertes.fr/hal-01322040.

64. Chen F, Zhang Y, Rohe K. Targeted sampling from massive block model graphs with personalized PageRank. Journal of the Royal Statistical Society: Series B (Statistical Methodology). 2019; 82(1).

65. Kloumann IM, Ugander J, Kleinberg J. Block models and personalized PageRank. Proceedings of the National Academy of Sciences. 2016; 114(1):33–38. https://doi.org/10.1073/pnas.1611275114 PMID: 27999183