BEC.2: A fast and relevant Multi-Scale Graph Clustering algorithm in nPnB framework

Bruno Gaume

Complex Systems Institute of Paris Île-de-France (ISC-PIF, UAR3611), CNRS, France
bruno.gaume@iscpif.fr

ABSTRACT:

In a recent paper, a unified theoretical framework $nPnB$ is defined for the evaluation of graph clusterings with respect to the following two properties:

$\mathbf{P_{DC}}$: **Each community is *Densely Connected*;**

$\mathbf{P_{WC}}$: **Communities are *Weakly Connected* to each other.**

In this framework, the authors propose BEC, a graph clustering method which they show returns outperforming results than most state-of-the-art methods, i.e. better satisfying the two properties P_{DC} and P_{WC}. However, this method is relatively slow and difficult to work on terrain graphs having more than a million nodes, the computation time is then counted in hours.

This makes it difficult to apply BEC to real-world graphs or graphs resulting from artificial intelligence where the number of nodes is very often greater than one million.

To solve this difficulty we propose $BEC.2$ in the $nPnB$ framework, a new graph clustering method. We compare on different classical benchmarks the computation times and the quality of clusterings returned by $BEC.2$, and $Louvain$ (one of the faster and most popular clustering method optimizing $Modularity$) as time base line, and BEC as quality base line.

We show that $BEC.2$ remains slower than $Louvain$ but 100 time faster than BEC with computation time now counted in seconds and returning better or equivalent results.

Introduction

In the recent paper (*10*), a unified theoretical framework $nPnB$ is defined for the evaluation of graph clusterings with respect to the following two properties:

$\mathbf{P_{DC}}$: Each community is *Densely Connected*;

$\mathbf{P_{WC}}$: Communities are *Weakly Connected* to each other.

In this theoretical framework a clustering of a graph is interpreted as a constrained binary classifier of node pairs intended to find the edges of the graph. The authors propose a graph clustering method BEC which they show returns outperforming results than most state-of-the-art methods (including $spectral\ graph\ clustering$, one of the best efficient state-of-the-art methods). However, this method is relatively slow and difficult to work on terrain graphs having more than a million nodes, the computation time is then counted in hours.

This makes it difficult to apply BEC to real-world graphs or graphs resulting from artificial intelligence where the number of nodes is very often greater than one million. To solve this difficulty we propose $BEC.2$ in the $nPnB$ framework, a new graph clustering method. We compare on different classical benchmarks the computation times and the quality of clusterings returned by $BEC.2$, and $Louvain$ (one of the faster and most popular clustering method optimizing $Modularity$) as time base line, and BEC as quality base line. We show that $BEC.2$ remains slower than $Louvain$ but 100 time faster than BEC with computation time now counted in seconds and returning better or equivalent results.

In section 1 we present the graph clustering method BEC in the $nPnB$ framework. In section 2 we define $BEC.2$, a method to find partitional clustering which we evaluate in section 3. In section 4 we extend $BEC.2$ to clustering with overlaps and conclude with perspectives in section 5.

This article follows the recent paper (*10*) which defines the unified $nPnB$ theoretical framework for the evaluation of graph clusterings. Therefore, in order to place the novelties of our present article in context, we use in sections 1, 2, 3, and 5, several passages that overlap with the article (*10*).

1 BEC in the $nPnB$ framework

Notation: For a set of vertices V, let's note $\mathcal{P}(V)$ the subsets of V and $\mathcal{P}_2(V) \subset \mathcal{P}(V)$ the pairs of elements from V. For a set $E \subset \mathcal{P}_2(V)$, $G = (V, E)$ defines a graph on V. A set $\mathcal{C} \in \mathcal{P}(\mathcal{P}(V))$ such that $\mathcal{C} = \{C_i |\ C_i \subset V, C_i \neq \emptyset, i \in I\}$ defines a *clustering* on V with *communities* C_i if and only if $\bigcup_{i\in I} C_i = V$. It is a *partitional clustering* if communities do not overlap ($\forall i \neq j \in I, C_i \cap C_j = \emptyset$), else it is an *overlapping clustering*. A *clustering of a graph* $G = (V, E)$ is a clustering on V. Let's note $\mathcal{G}(V) = \{G|\ G \text{ is a graph on } V\}$, and $\mathcal{C}(V) = \{\mathcal{C}|\ \mathcal{C} \text{ is a clustering on } V\}$.

Definition 1. nPnB: Let a graph $G = (V, E)$. A $nPnB$ is a *binary classifier of* **n***odes* **P***airs by* **n***odes* **B***locks*. Instead of providing two complementary sets of nodes pairs as $Positive\ Pairs$ and $Negative\ Pairs$, a $nPnB$ classifier has to provide its predictions in the form of nodes blocks $\{C_i|\ C_i \subset V, C_i \neq \emptyset, i \in I\} = \mathcal{C}$, a clustering such that $\{x, y\}$ is a $Positive\ Pair$ if and only if $\exists C_i \in \mathcal{C}$ such $\{x, y\} \in C_i$, otherwise $\{x, y\}$ is a $Negative\ Pair$. We will note $nPnB^P$ clusterings for which blocks are not overlapping and $nPnB^O$ those allowing overlap. ■

This can be formalized by defining $\widehat{\mathcal{C}}$ the *derived graph* from a clustering $\mathcal{C}$ using the three functions of Definition 2.

Definition 2. U$(\cdot)$, $\Xi(\cdot)$ and $\widehat{\cdot}$: Let a clustering $\mathcal{C} \in \mathcal{C}(V)$ of a graph $G = (V, E) \in \mathcal{G}(V)$.

$$U(\cdot) : \mathcal{P}(\mathcal{P}(V)) \longrightarrow \mathcal{P}(V),\ U(\mathcal{C}) = \bigcup_{C_i \in \mathcal{C}} C_i \tag{1}$$

$$\Xi(\cdot) : \mathcal{P}(\mathcal{P}(V)) \longrightarrow \mathcal{P}_2(V),\ \Xi(\mathcal{C}) = \bigcup_{C_i \in \mathcal{C}} \mathcal{P}_2(C_i) \tag{2}$$

$$\widehat{\cdot} : \mathcal{C}(V) \longrightarrow \mathcal{G}(V),\ \widehat{\mathcal{C}} = \big(U(\mathcal{C}),\ \Xi(\mathcal{C})\big) \tag{3}$$

■

For any clustering $\mathcal{C}$ of a graph $G = (V, E)$ we can then compute the two classical metrics in diagnostic binary classification $precision$ and $recall$, which assess the capacities of the graph $\widehat{\mathcal{C}} = \big(U(\mathcal{C}),\ \Xi(\mathcal{C})\big)$ to detect the edges of the graph $G = (V, E)$ (see Fig. 1).

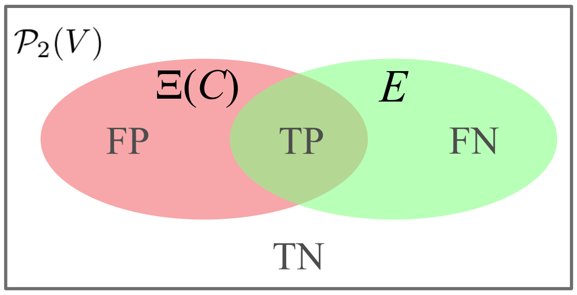
Definition 3. precision & recall: Let a clustering $\mathcal{C} \in \mathcal{C}(V)$ of a graph $G = (V, E) \in \mathcal{G}(V)$.

$$P(\cdot,\cdot) : \mathcal{G}(V) \times \mathcal{G}(V) \longrightarrow [0,1],\ P(\widehat{\mathcal{C}}, G) = \frac{|TP|}{|TP| + |FP|} = \frac{|\Xi(\mathcal{C}) \cap E|}{|\Xi(\mathcal{C})|} \tag{4}$$

The **precision** $P(\widehat{\mathcal{C}}, G)$ is the probability that an edge drawn at random in $\Xi(\mathcal{C})$ (edges of the graph $\widehat{\mathcal{C}}$), actually belongs to E (edges of the graph G);

$$R(\cdot,\cdot) : \mathcal{G}(V) \times \mathcal{G}(V) \longrightarrow [0,1],\ R(\widehat{\mathcal{C}}, G) = P(G, \widehat{\mathcal{C}}) = \frac{|TP|}{|TP| + |FN|} = \frac{|\Xi(\mathcal{C}) \cap E|}{|E|} \tag{5}$$

The **recall** $R(\widehat{\mathcal{C}}, G)$ is the probability that an edge drawn at random in E (edges of G), belongs to $\Xi(\mathcal{C})$ (edges of the graph $\widehat{\mathcal{C}}$). ■



TP $= \Xi(\mathcal{C}) \bigcap E$ is the set of True Positives;

TN $= \overline{\Xi(\mathcal{C})} \bigcap \overline{E}$ is the set of True Negatives;

FP $= \Xi(\mathcal{C}) \bigcap \overline{E}$ is the set of False Postives;

FN $= \overline{\Xi(\mathcal{C})} \bigcap E$ is the set of False Negatives;

Figure 1: **Venn diagram of the capacities of the graph $\widehat{\mathcal{C}} = \big(U(\mathcal{C}),\ \Xi(\mathcal{C})\big)$ to detect the edges of the graph $G = (V, E)$.** Interpreting graph clustering as binary classifier of node Pairs by node Blocks with to formula 2.

The two properties P_{DC} and P_{WC} are then faithfully formalized by these two metrics $precision$ and $recall$ in the sense that:

- $\mathbf{P_{DC}}$: The more each module is densely connected, the higher the $precision$ and vice versa;
- $\mathbf{P_{WC}}$: The less the modules are connected to each other, the higher the $recall$ and vice versa.

Unless $G = (V, E)$ is reduced to a set of unconnected cliques, the following proposition (see a proof in (*10*)) implies that there is no partitional clustering $\mathcal{C}$ such $P(\widehat{\mathcal{C}}, G) = R(\widehat{\mathcal{C}}, G) = 1$. The two metrics $precision$ and $recall$ are thus generally $antagonistic$.

Proposition 1. *Let a partitional clustering $\mathcal{C} \in \mathfrak{C}(V)$ of a graph $G = (V, E) \in \mathcal{G}(V)$. Then:*

$$P(\widehat{\mathcal{C}}, G) = R(\widehat{\mathcal{C}}, G) = 1$$
$$\Downarrow$$
The graph $G = (V, E)$ is reduced to a set of unconnected cliques.

Since $precision$ and $recall$ are generally $antagonistic$, finding sets of non overlapping clusters on networks can be envisioned as a non trivial bi-objective task in the $nPnB^P$ framework. Then to compare two clustering $\mathcal{C}_1$ and $\mathcal{C}_2$ such that $P(\widehat{\mathcal{C}_1}, G) < P(\widehat{\mathcal{C}_2}, G)$ and $R(\widehat{\mathcal{C}_2}, G) < R(\widehat{\mathcal{C}_1}, G)$ or vice versa, we need to specify our priorities in terms of $precision = \frac{|TP|}{|TP|+|FP|}$ faithfully formalizing P_{DC} and $recall = \frac{|TP|}{|TP|+|FN|}$ faithfully formalizing P_{WC}. This decision will depend on the needs of the modeler, from which she will define what constitutes a "good clustering". It's only once a trade-off between $precision$ and $recall$ has been made that the modeler can assess the performances of different clustering methods.

The best known metric, the most used to evaluate binary classifier, with a hands s on the trade-off between $precision$ and $recall$ is the F-score function: $\frac{(1+f(s)^2).|TP|}{(1+f(s)^2).|TP|+f(s)^2.|FN|+|FP|} =$

$$F_s(P,R) = \frac{(1+f(s)^2).(P.R)}{R+f(s)^2.P} \tag{6}$$

With $P = P(\widehat{\mathcal{C}}, G)$, $R = R(\widehat{\mathcal{C}}, G)$, $s \in [0,1]$, $f(s) = tan(\frac{\pi.s}{2})$ and $F_1(P,R) = R$.

- For $s = 0$, only the $precision$ counts (i.e. the P_{DC} property);
- For $s = 1$, only the $recall$ counts (i.e. the P_{WC} property);
- For $s = 0.5$, $precision$ and $recall$ are of the same importance, and can be interpreted as a 'middle point of view'. It has both homogeneity and completeness, two fundamental properties for metrics intending compare clusterings (*1*). On contrary, $precision$ has only homogeneity property –it is the archetypal metric of homogeneity– and $recall$ has only completness property –it is the archetypal metric of completness.
- For $s \in\ ]0,\ 0.5[$: In order to improve $F_s(P,R)$, $precision$ need to be higher with a greater number of smaller and denser modules;
- For $s \in\ ]0.5,\ 1[$: In order to improve $F_s(P,R)$, $recall$ need to be higher with a fewer number of bigger but less dense modules.

Thereby, the trade-off s between $precision$ and $recall$ can be used to adjust the desired *'description scale'* (i.e. the size of communities).

1.1 $nPnB$ as Unified Graph Clustering Framework

To simplify the notations, let's note $\mathcal{P}(\mathcal{C}, G) = P(\widehat{\mathcal{C}}, G)$; $\mathcal{R}(\mathcal{C}, G) = R(\widehat{\mathcal{C}}, G)$ and $\mathcal{F}_s(\mathcal{C}, G) = F_s(P(\widehat{\mathcal{C}}, G), R(\widehat{\mathcal{C}}, G))$.

Interpreting graph clustering as $nPnB$, and using the Ξ function to define the four metrics TP, TN, FP, FN makes it possible to use these four metrics for clustering with and without overlaps, intrinsically against the original graph $G = (V, E)$ (as in Fig. 1) or extrinsicaly against a ground-truth $\mathcal{C}_G^{ref}$ of G, by replacing E, the edges of the graph G, by $\Xi(\mathcal{C}_G^{ref})$, the edges of the graph $\widehat{\mathcal{C}_G^{ref}}$: **TP**$= \Xi(\mathcal{C}) \bigcap \Xi(\mathcal{C}_G^{ref})$; **TN**$= \overline{\Xi(\mathcal{C})} \bigcap \overline{\Xi(\mathcal{C}_G^{ref})}$; **FP**$= \Xi(\mathcal{C}) \bigcap \overline{\Xi(\mathcal{C}_G^{ref})}$; **FN**$= \overline{\Xi(\mathcal{C})} \bigcap \Xi(\mathcal{C}_G^{ref})$.

Let $\mathcal{C}$ and $\mathcal{C}^{ref}$ be two clusterings of a graph $G = (V, E)$. The $nPnB$ framework makes it possible to measure the capacity of the graph $\widehat{\mathcal{C}}$ to detect the edges of the graph $\widehat{\mathcal{C}^{ref}}$. If $\mathcal{C}^{ref} = E$ then $\widehat{\mathcal{C}^{ref}} = G$, it is an intrinsic measurement else it is extrinsic. We can measure this capacity:

In the 2-dimensional space precision $\times$ recall: with $\mathcal{P}(\mathcal{C}, \widehat{\mathcal{C}^{ref}})$ and $\mathcal{R}(\mathcal{C}, \widehat{\mathcal{C}^{ref}})$, where $\mathcal{P}(\cdot,\cdot)$ the metric $precision$ faithfully formalizes the P_{DC} property and $\mathcal{R}(\cdot,\cdot)$ the metric $recall$ faithfully formalizes the P_{WC} property.

Then we can objectively conclude that $\mathcal{C}_2$ performs better or equal than $\mathcal{C}_1$ in regard of the gold standard $\mathcal{C}^{ref}$ **iff**

$$\left(\mathcal{P}(\mathcal{C}_2, \widehat{\mathcal{C}^{ref}}), \mathcal{R}(\mathcal{C}_2, \widehat{\mathcal{C}^{ref}})\right) \in \left[\mathcal{P}(\mathcal{C}_1, \widehat{\mathcal{C}^{ref}}), 1\right] \times \left[\mathcal{R}(\mathcal{C}_1, \widehat{\mathcal{C}^{ref}}), 1\right]$$

In 1-dimensional space: Since $precision$ and $recall$ are antagonistic, optimizing both of them is a multi-objective optimization that most of the time does not have a unique optimal solution as shown in (*10*). If we cannot conclude one way or the other in the 2-dimensional space $precision \times recall$, the choice of a clustering can be reduced to 1-dimensional space, provided that we combine:

- **A subjective decision:** defining a trade-off between $precision$ and $recall$ through the choice of a *scale of description* $\sigma \in [0, 1]$;
- **An objective methodology:** using then $\mathcal{F}_\sigma$ as objective criteria for the evaluation of graph clustering methods in regard of the subjective *scales of description* σ.

Then we can objectively conclude that $\mathcal{C}_2$ performs better or equal than $\mathcal{C}_1$ in regard of the gold standard $\mathcal{C}^{ref}$ under the subjective *scale of description* σ **iff**

$$\mathcal{F}_\sigma(\mathcal{C}_1, \widehat{\mathcal{C}^{ref}}) \leqslant \mathcal{F}_\sigma(\mathcal{C}_2, \widehat{\mathcal{C}^{ref}})$$

In a 1-dimensional decision space, it is necessary to choose a subjective σ description scale *before* being able to objectively compare two clusterings with $\mathcal{F}_\sigma$.

1.2 BEC a clustering family algorithms based on $\mathcal{F}_s$ optimization

We present in this section BEC^s (Binary Edges Classifier) the clustering family detreminist algorithms proposed in (*10*) to optimize $\mathcal{F}_s$ in regard of a graph $G = (V, E)$.

The trivial clustering $\mathcal{C} = \{\{i\}|\ i \in V\}$ where each vertice is assigned to its own cluster is a partitional clustering. Then $\forall s \in [0, 1]$ its $\mathcal{F}_s$ score $\mathcal{F}_s(\mathcal{C}, G) = F_s(P(\widehat{\mathcal{C}}, G), R(\widehat{\mathcal{C}}, G)) = 0$ since its $recall$ $R(\widehat{\mathcal{C}}, G) = 0$.

BEC^s improve this trivial clustering by an agglomeration process that reviews each edge of G only once and merges the clusters of their vertices if this operation does not decrease $\mathcal{F}_s(\mathcal{C}, G)$ (see (*10*) for pseudo-code).

The order in which edges are traversed is essential. The proposed algorithm involves choosing a ranking function on E derived from a similarity measure $Sim(G, x, y)$ on the vertices of G. Edges $\{x, y\} \in E$ are then reviewed by descending order of $Sim(G, x, y)$.

The quality of the process depends on the choice of the similarity measure, and there is no guarantee of obtaining an optimal clustering $\mathcal{C}^*$ such $\mathcal{F}_s(\mathcal{C}^*, G)$ is maximal. BEC^s use the similarity cos as ranking function:

$$cos(G = (V, E), x, y) = Cosinus\Big(\overrightarrow{(P^2_G(x{\leadsto}x), P^2_G(x{\leadsto}y))}, \overrightarrow{(P^2_G(y{\leadsto}x), P^2_G(y{\leadsto}y))}\Big)$$

Where: $P^\lambda_G(x{\leadsto}y)$ is the probability that a random walker wandering on the graph G through its edges, reaches the node y after λ steps starting from the node x.

In the $nPnB$ framework, using the similarity cos to rank the edges (cos ranking), the partitional clusterings returned by BEC^s outperform most state-of-the-art partitional clusterings methods (including $spectral\ graph\ clustering$, one of the best performing state-of-the-art methods) i.e. better satisfying the two properties P_{DC} and P_{WC} (see (*10*)).

For clarity and readability we will denote in the following BEC^s as $MER.cos^s$ (cluster **MER**ging strategy with edge ranking by **cos**).

2 $BEC.2$ to optimize $\mathcal{F}_s$ better and faster

In this section we propose a new clustering family algorithms based on $\mathcal{F}_s$ optimization, faster an more relevant than the family $MER.cos^s$.

2.1 Replace $cos\ ranking$ by $random\ ranking$

Most of the time consumed by $MER.cos^s$ is spent calculating the similarities $cos(G = (V, E), x, y)$ for all $\{x, y\} \in E$. The worst-case time complexity to compute edges $cos\ ranking$ is $O(|V|^2)$, which can consume many time with large graphs. So to save computation time let's start by examining how the $MER.rand^s$ method reacts, where $MER.rand^s$ has the same merging clusters strategy that $MER.cos^s$ but with a edges $random\ ranking$ replacing the edges $cos\ ranking$.

To illustrate how the $MER.rand^s$ method reacts, we compare $MER.rand^s$, with $MER.cos^s$ (outperforming state-of-the-art partitional clusterings methods in the $nPnB$ framework (*10*)) and $Louvain$ (*5*) (one of the faster and most popular clustering method optimizing $Modularity$ (*15*)).

We compare these methods on a standard real-world graph $G_{em} = (V_{em}, E_{em})$ frequently used for benchmarking (*19*). G_{em} describes e-mail data from a large research institution composed of a set V_{em} of employees. This is a little graph as standard benchmark with $|V_{em}| = 1,005$, $|E_{em}| = 16,064$. The graph contains an undirected edge $\{i, j\}$ if employee i and employee j have exchanged at least one e-mail either way. The dataset[1] on which G_{em} is build, also contains the list of the 42 departments of the research institute that are often considered as a 'ground-truth' partition $\mathcal{C}_{Dep}$ on G_{em}.

We add to this clusterings comparison the Oracle method met_{Dep} returning the 'ground-truth' partition itself ($\mathcal{C} = \mathcal{C}_{Dep} \in \mathcal{P}(\mathcal{P}(V))$) and the omniscient overlapping clustering method met_E returning the edges of graph itself ($\mathcal{C} = E \in \mathcal{P}(\mathcal{P}(V))$).

Fig. 2 displays methods applied to G_{em} on the $precision \times recall$ plane, and Fig. 3 displays the performances of each clustering method under the scale of description $\sigma \in [0, 1]$.

■ **I: We can see in Fig. 2** that generally $\mathcal{P}(MER.rand^s(G_{em}), G_{em}) \not\approx \mathcal{P}(MER.cos^s(G_{em}), G_{em})$ and $\mathcal{R}(MER.rand^s(G_{em}), G_{em}) \not\approx \mathcal{R}(MER.cos^s(G_{em}), G_{em})$.

■ **II: We can see in Fig. 3** that $\forall \sigma \in [0, 1], \mathcal{F}_\sigma(MER.rand^{s=\sigma}(G_{em}), G_{em}) \leqslant \mathcal{F}_\sigma(MER.cos^{s=\sigma}(G_{em}), G_{em})$.

Points **I** and **II** show that with the merging cluster strategy $MER.rank^s$, $precision$, $recall$ and $\mathcal{F}_\sigma$ of the retrurned clusterings are sensitive to the ranking edges $rank$ which needs a well adapted ranking edge (for example as cos ranking) to return relevant results (see (*10*) for more details).

[1] Available at `https://snap.stanford.edu/data/email-Eu-core.html`

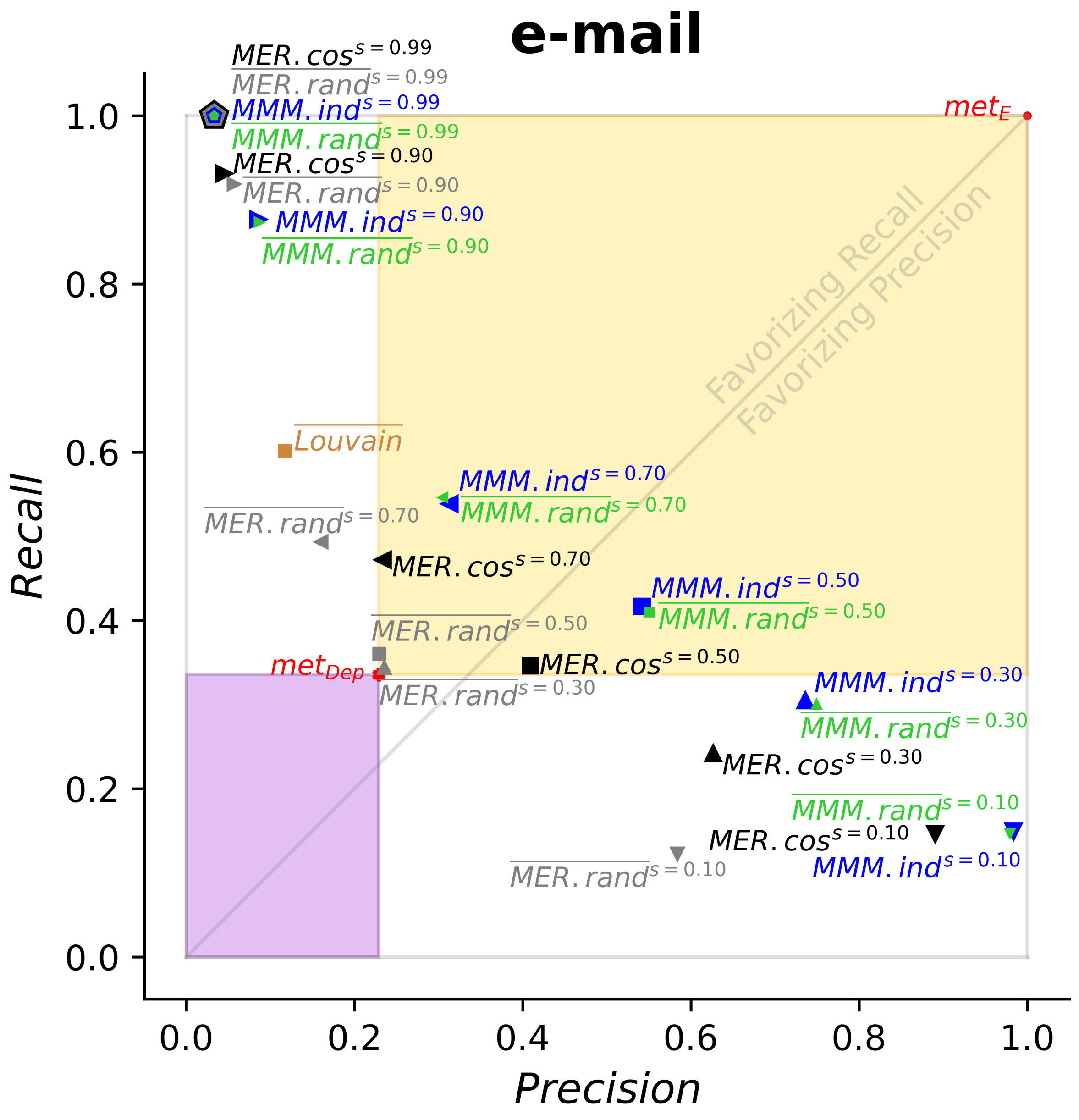


Figure 2: **Performances in the 2-dimensional space** $precision \times recall$ **of clustering methods when applied to the e-mail graph $\mathbf{G_{em}}$.** The Oracle method met_{Dep} and the Omniscient method met_E are highligthed in red. For each non deterministic methode ($Louvain$, $MER.rand^s$ and $MMM.rand^s$), the $precision$ and the $recall$ are the averages computed over 100 runing (the standard deviations are always lower than 0.02).

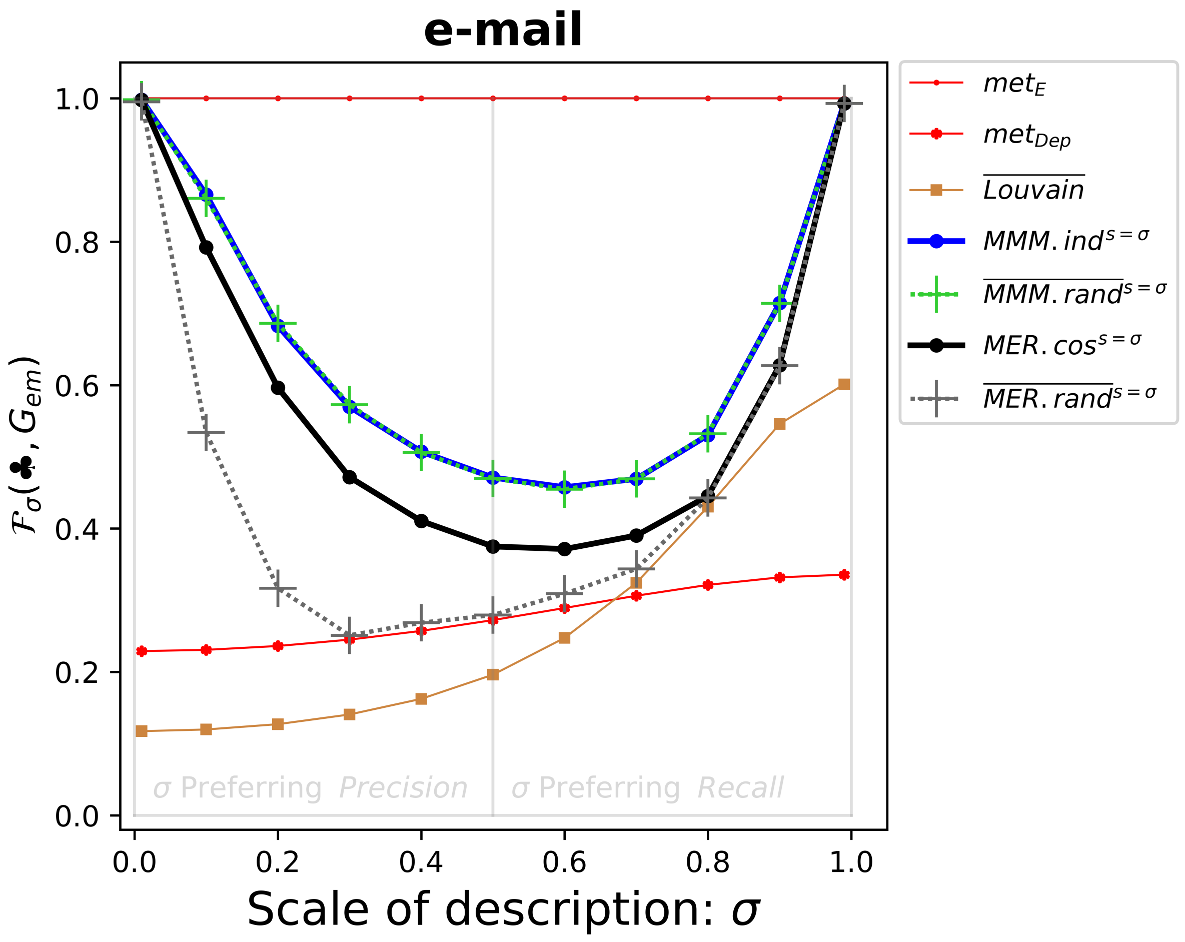


Figure 3: **Performances** $\mathcal{F}_\sigma(♣, G_{em})$ **of clustering methods** $method(G_{em}) = ♣$ **under the description scale** σ**.** The Oracle method met_{Dep} and the Omniscient method met_E are highligthed in red. For each non deterministic $method \in \{Louvain,\ MER.rand^{s=\sigma},\ MMM.rand^{s=\sigma}\}$, the Fscores $\mathcal{F}_\sigma$ are the averages computed over 100 runing (the standard deviations are always lower than 0.01).

2.2 $Moving\ nodes$ and $Merging\ clusters$

In this section we propose $MMM.ind^s$ (**M**oving nodes then **M**erging clusters then **M**oving nodes strategy with index nodes ranking) a new cluster construction strategy to optimize $\mathcal{F}_s(\mathcal{C}, G = (V, E))$. General principle of $MMM.ind^s$ is as follows:

Algorithm: $\mathcal{C} = MMM.ind^s(G, \varepsilon)$**:** To find Partitional Clustering in $nPnB$ framework

```
Input: $G = (V, E) \in \mathcal{G}(V)$
        $s \in [0, 1]$ (To find a partitional clustering $\mathcal{C}$ optimizing $\mathcal{F}_s(\mathcal{C}, G)$)
        $\varepsilon \in [0, 1]$ (Default $\varepsilon = 0.01$)
Output: $\mathcal{C} \in \mathcal{C}(V)$ such $\forall C_i \neq C_j \in \mathcal{C}(V), C_i \cap C_j = \emptyset$

    Initialization: Build the trivial clustering where each node is assigned to its own cluster:
    $\mathcal{C} \leftarrow \{\{i\} | \, i \in V\}$;

    Do$_1$
        (1) Move Loop on i $\in$ V: For each node $i$, improve the curent clustering $\mathcal{C}$ by
        moving $i$ into the cluster of one of its neighbors or by isolation in its own cluster. If
        no edit can improve $\mathcal{F}_s(\mathcal{C}, G)$ then we examine the next node, else we adopt the edit
        that most improves $\mathcal{F}_s(\mathcal{C}, G)$;
    While (Total improvement during Move Loop (1) is greater than $\varepsilon$);

    Do$_2$
        (2) Merge Loop on i $\in$ V: For each node $i$, improve the curent clustering $\mathcal{C}$ by
        merging cluster of $i$ with the cluster of one of its neighbors. If no edit can improve
        $\mathcal{F}_s(\mathcal{C}, G)$ then we examine the next node, else we adopt the edit that most improves
        $\mathcal{F}_s(\mathcal{C}, G)$;
    While (At least one edit is done during Merge Loop (2));

    Do$_3$
        (3) Move Loop on i $\in$ V: For each node $i$, improve the curent clustering $\mathcal{C}$ by
        moving $i$ into the cluster of one of its neighbors or by isolation in its own cluster. If
        no edit can improve $\mathcal{F}_s(\mathcal{C}, G)$ then we examine the next node, else we adopt the edit
        that most improves $\mathcal{F}_s(\mathcal{C}, G)$;
    While (Total improvement during Move Loop (3) is greater than $\varepsilon$);

    Return: Return the curent clustering $\mathcal{C}$.
```

The input variable ε is only used to speed up the calculation time in **Do**$_1$ and **Do**$_3$. With $\varepsilon = 0$ the computation time is longer for results of quality only slightly better than with $\varepsilon = 0.01$.

$MMM.ind$ can have remorses, meaning that in **Do**$_1$ and **Do**$_3$, a move from a vertex to a module can be undone later (e.g. in Tab. 1: in **Do**$_1$, edit [b] is subsequently undone by edits [c] and [d]). Conversely, $Merge.rank$ is remorseless, meaning that when two modules are merged, this merge cannot be undone later (e.g., edit [a] in Tab. 1). This is why $Merge$ is very sensitive to rank edges.

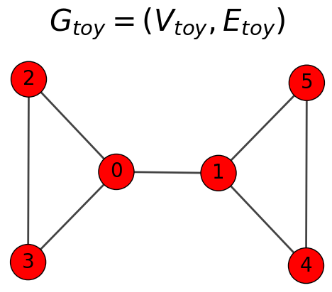
 $G_{toy} = (V_{toy}, E_{toy})$ 2 5 0 1 3 4	
Merge.rank$^{\mathbf{s=0.5}}$ with ranked edge RE : $[(0,1),(2,3),(4,5),(0,2),(1,5),(0,3),(1,4)]$	**MMM.ind$^{\mathbf{s=0.5}}$** with ranked vertices RV : $[0,1,2,3,4,5]$
• Init: $\mathcal{C} \longleftarrow \{\{0\},\{1\},\{2\},\{3\},\{4\},\{5\}\}$ • Merge Loop with RE: $\mathcal{C} \longleftarrow \{\{0,1\},\{2\},\{3\},\{4\},\{5\}\}$ [a] $\mathcal{C} \longleftarrow \{\{0,1\},\{2,3\},\{4\},\{5\}\}$ $\mathcal{C} \longleftarrow \{\{0,1\},\{2,3\},\{4,5\}\}$ $\mathcal{C} \longleftarrow \{\{0,1,2,3\},\{4,5\}\}$ • Return $\mathcal{C}$	• Init: $\mathcal{C} \longleftarrow \{\{0\},\{1\},\{2\},\{3\},\{4\},\{5\}\}$ • Move Loop (1) with RV: $\mathcal{C} \longleftarrow \{\{0,1\},\{2\},\{3\},\{4\},\{5\}\}$ [b] $\mathcal{C} \longleftarrow \{\{0,1\},\{2,3\},\{4\},\{5\}\}$ $\mathcal{C} \longleftarrow \{\{0,1\},\{2,3\},\{4,5\}\}$ • Move Loop (1) with RV: $\mathcal{C} \longleftarrow \{\{1\},\{0,2,3\},\{4,5\}\}$ [c] $\mathcal{C} \longleftarrow \{\{0,2,3\},\{1,4,5\}\}$ [d] • Move Loop (1) with RV: (no edit) • Merge Loop (2) with RV: (no edit) • Move Loop (3) with RV: (no edit) • Return $\mathcal{C}$
$\mathcal{C} = \{\{0,1,2,3\},\{4,5\}\}$ $\mathcal{P}(\mathcal{C}, G_{toy}) = 0.71, \quad \mathcal{R}(\mathcal{C}, G_{toy}) = 0.71$ $\mathcal{F}_{\mathbf{s=0.5}}(\mathcal{C}, \mathbf{G_{toy}}) = \mathbf{0.71}$	$\mathcal{C} = \{\{0,2,3\},\{1,4,5\}\}$ $\mathcal{P}(\mathcal{C}, G_{toy}) = 1.00, \quad \mathcal{R}(\mathcal{C}, G_{toy}) = 0.86$ $\mathcal{F}_{\mathbf{s=0.5}}(\mathcal{C}, \mathbf{G_{toy}}) = \mathbf{0.92}$

Table 1: **Edits of** $Merge.rank^{s=0.5}$ **and** $MMM.ind^{s=0.5}$ **on the toy graph** G_{toy}**.**

With $MMM.ind^s$ in the loops (1), (2), (3), the nodes are traversed according to ranking ind, the order induced by their indices. But the nodes could just as easily be traversed according to a random ranking $rand$, we will note then such method $MMM.rand^s$ (**M**oving nodes then **M**erging clusters then **M**oving nodes strategy with random nodes ranking). Note that, unlike the $MMM.rand^s$ method, the $MMM.ind^s$ method has the avantage to be deterministic.

■ **III: We can see in Fig. 2** that for all $s \in [0,1]$: $\mathcal{P}(MMM.rand^s(G_{em}), G_{em}) \approx \mathcal{P}(MMM.ind^s(G_{em}), G_{em})$ and $\mathcal{R}(MMM.rand^s(G_{em}), G_{em}) \approx \mathcal{R}(MMM.ind^s(G_{em}), G_{em})$.

■ **IV: We can see in Fig. 3** that for all σ: $\mathcal{F}_\sigma(Louvain(G_{em}), G_{em}) \leqslant \mathcal{F}_\sigma(MER.rand^{s=\sigma}(G_{em}), G_{em})$ $\leqslant \mathcal{F}_\sigma(MER.cos^{s=\sigma}(G_{em}), G_{em}) \leqslant \mathcal{F}_\sigma(MMM.rand^{s=\sigma}(G_{em}), G_{em}) \approx \mathcal{F}_\sigma(MMM.ind^{s=\sigma}(G_{em}), G_{em})$.

■ **V: We can see in Tab. 2** that $\forall s \in [0,1]$, the averages of $\mathcal{F}_{0.5}(MMM.rand^s(G_{em})), MM\widehat{M.ind^s}(G_{em})),$

over these 100 runings of $MMM.rand^s(G_{em})$ are always equal or greater than 0.66 with standarts deviation always equal or smaller than 0.08. That is, the hundred different executions of $MMM.rand^s(G_{em})$ return clusterings that all *'look like'* $MMM.ind^s(G_{em})$.

methode	$\overline{\mathcal{F}_{0.5}}$ **(std)**
$MER.rand^{s=0.10}$	0.27(0.03)
$MER.rand^{s=0.30}$	0.15(0.02)
$MER.rand^{s=0.50}$	0.23(0.02)
$MER.rand^{s=0.70}$	0.26(0.03)
$MER.rand^{s=0.90}$	0.69(0.03)
$MER.rand^{s=0.99}$	1.00(0.00)

methode	$\overline{\mathcal{F}_{0.5}}$ **(std)**
$MMM.rand^{s=0.10}$	0.66(0.04)
$MMM.rand^{s=0.30}$	0.82(0.03)
$MMM.rand^{s=0.50}$	0.90(0.03)
$MMM.rand^{s=0.70}$	0.71(0.08)
$MMM.rand^{s=0.90}$	0.95(0.03)
$MMM.rand^{s=0.99}$	1.00(0.00)

Table 2: $\overline{\mathcal{F}_{0.5}}$ **and standart deviation over 100 pairs of clusterings:** In Fig. 2 and 3, for each non deterministic methode $met \in \{MER.rand,\ MMM.rand\}$, the y-values are the averages computed over 100 runing. In the first column we then calculate the average of $\mathcal{F}_{0.5}(MER.rand(G_{em})), \widehat{MER.cos}(G_{em}))$ over these 100 runings of $MER.rand(G_{em})$ and the standard deviations. We do the same thing with $MMM.rand$ and $MMM.ind$ in the second column.

■ **VI: We can see in Tab. 3** that $\forall s \in [0, 1]$, over the 4950 pairs $\{\mathcal{C}_1, \mathcal{C}_2\}$ such that $\mathcal{C}_1$ is returned by one of 100 runings of $MMM.rand^s(G_{em})$ and $\mathcal{C}_2$ is returned by an other of the 100 runings of $MMM.rand^s(G_{em})$, then the averages of $\mathcal{F}_{0.5}(\mathcal{C}_1, \widehat{\mathcal{C}_2})$ are always equal or gerater than 0.63 with standarts deviation always equal or smaller than 0.05. The hundred different runings of $MMM.rand^s s(G_{em})$ return clusterings that *'look alike'*. That is, if two graphs G_1 and G_2 are isomorphic, then $\forall s \in [0, 1]$ the two clustrings $\mathcal{C}_1 = MMM.rand^s(G_1)$ and $\mathcal{C}_2 = MMM.rand^s(G_2)$ are not necessarily equal but they *'look alike'* in the sense that $\mathcal{F}_{0.5}(\mathcal{C}_1, \widehat{\mathcal{C}_2})$ is strong.

Points **III-VI** show that with the MMM strategy, $precision$, $recall$ and $\mathcal{F}_\sigma$ of the retrurned clusterings are only very slightly sensitive to the order in which the nodes are traversed in loops (1), (2) and (3). Furthermore we will see in the section 3 that $MMM.ind$ is faster than $MER.cos$.

2.2.1 $MMM.ind$ is able to outperform met_{Dep}

In Fig. 2, all methods in the yellow square intrinsically objectively outperforms met_{Dep} (returning $\mathcal{C}_{Dep}$ often considered as a *'ground-truth'* of the graph G_{em}). Indeed any method $method$ in the yellow square is such $\mathcal{P}(met_{Dep}, G_{em}) < \mathcal{P}(method(G_{em}), G_{em})$ and in same time $\mathcal{R}(met_{Dep}, G_{em}) < \mathcal{R}(method(G_{em}), G_{em})$.

Since both methods $MMM.ind^{0.50}$ and $MMM.ind^{0.70}$ are in the yellow square, we can

methode	$\overline{\mathcal{F}_{0.5}}$ **(std)**
$MER.rand^{s=0.10}$	0.17 (0.02)
$MER.rand^{s=0.30}$	0.25 (0.02)
$MER.rand^{s=0.50}$	0.26 (0.02)
$MER.rand^{s=0.70}$	0.34 (0.04)
$MER.rand^{s=0.90}$	0.78 (0.04)
$MER.rand^{s=0.99}$	1.00 (0.00)
$Louvain$	0.78(0.08)

methode	$\overline{\mathcal{F}_{0.5}}$ **(std)**
$MMM.rand^{s=0.10}$	0.63 (0.04)
$MMM.rand^{s=0.30}$	0.85 (0.03)
$MMM.rand^{s=0.50}$	0.90 (0.04)
$MMM.rand^{s=0.70}$	0.69 (0.05)
$MMM.rand^{s=0.90}$	0.94 (0.04)
$MMM.rand^{s=0.99}$	1.00 (0.00)

Table 3: $\overline{\mathcal{F}_{0.5}}$ **and standart deviation over 4950 pairs of clusterings:** In Fig. 2 and 3, for each non deterministic methode $met \in \{Louvain,\ MER.rand,\ MMM.rand\}$, the y-values are the averages computed over 100 runing. For each of these methods met, there are therefore $\frac{100\times 99}{2} = 4950$ pairs $\{\mathcal{C}_1, \mathcal{C}_2\}$ such that $\mathcal{C}_1$ is returned by a runing of $met(G_{em})$ and $\mathcal{C}_2$ is returned by an other runing of $met(G_{em})$. We then calculate the average of $\mathcal{F}_{0.5}(\mathcal{C}_1, \widehat{\mathcal{C}_2})$ over these 4950 pairs $\{\mathcal{C}_1, \mathcal{C}_2\}$ and the standard deviations.

therefore objectively state that these two methods outperform met_{Dep}. This means that the two derived graphs $\widehat{MMM.ind^{0.50}(G_{em})}$ and $\widehat{MMM.ind^{0.70}(G_{em})}$ objectively detect the edges of G_{em} better than the derived graph $\widehat{\mathcal{C}_{Dep}}$ from the *'ground-truth'* $\mathcal{C}_{Dep}$.

To be able to compare objectively the methods outside the yellow and purple rectangles we need to subjectively choose before a trade-off σ between $precision$ and $recall$ relatively to what we will expect betwwen P_{DC} and P_{WC} propeties from clustering of G_{em}.

For example with the subjective trade-off $\sigma = 0.3$ then $\mathcal{F}_{0.3}(Louvain(G_{em}), G_{em}) = 0.15 < 0.25 = \mathcal{F}_{0.3}(met_{Dep}, G_{em})$ while with the subjective trade-off $\sigma = 0.8$ then $\mathcal{F}_{0.8}(Louvain(G_{em}), G_{em}) = 0.43 > 0.32 = \mathcal{F}_{0.8}(met_{Dep}, G_{em})$.

3 Evaluation

In the following we will note BEC^s for $MER.cos^s$ and $BEC.2^s$ for $MMM.ind^s$

$Louvain$ is one of the fastest state-of-the-art methods, and it is shown in (*10*) that BEC outperform state-of-the-art partitional clusterings methods in the $nPnB$ framework. Also, in this section we compare on different graphs the computation times and the quality of clusterings returned by $BEC.2$, and $Louvain$ as time base line and BEC as quality base line.

	$\mathbf{G_{dblp}}$ $\lvert V\rvert = 317\,080$ $<\!\multimap 43\,181>$ $\lvert E\rvert = 1\,049\,866$	$\mathbf{G_{amazon}}$ $\lvert V\rvert = 334\,863$ $<\!\multimap 25\,709>$ $\lvert E\rvert = 925\,872$	$\mathbf{G_{retweet}}$ $\lvert V\rvert = 1\,112\,702$ $<\!\multimap 759\,734>$ $\lvert E\rvert = 2\,278\,852$	$\mathbf{G_{youtube}}$ $\lvert V\rvert = 1\,134\,890$ $<\!\multimap 602\,539>$ $\lvert E\rvert = 2\,987\,624$	$\mathbf{G_{WikiTalk}}$ $\lvert V\rvert = 2\,516\,783$ $<\!\multimap 1\,843\,811>$ $\lvert E\rvert = 5\,021\,410$
met	**P R** $\mathcal{F}_{0.5}$	**P R** $\mathcal{F}_{0.5}$	**P R** $\mathcal{F}_{0.5}$	**P R** $\mathcal{F}_{0.5}$	**P R** $\mathcal{F}_{0.5}$
$B^{0.30}$	**96 40 57** $\langle\mathcal{F}_{0.30} : 74\rangle$ [143 005; 74 287] (659s)	**81 40 53** $\langle\mathcal{F}_{0.30} : 67\rangle$ [143 381; 81 201] (731s)	**35 15 21** $\langle\mathcal{F}_{0.30} : 27\rangle$ [769 620; 79 409] (!)	**62 18 28** $\langle\mathcal{F}_{0.30} : 41\rangle$ [647 177; 244 839] (!)	**26 5 9** $\langle\mathcal{F}_{0.30} : 15\rangle$ [2 243 002; 66 328] (!)
$B^{0.50}$	**78 47 59** $\langle\mathcal{F}_{0.50} : 59\rangle$ [110 572; 63 979] (664s)	**63 52 57** $\langle\mathcal{F}_{0.50} : 57\rangle$ [99 830; 68 640] (739s)	**23 24 23** $\langle\mathcal{F}_{0.50} : 23\rangle$ [574 168; 78 148] (!)	**41 23 29** $\langle\mathcal{F}_{0.50} : 29\rangle$ [499 570; 206 340] (!)	**13 8 10** $\langle\mathcal{F}_{0.50} : 10\rangle$ [2 102 765; 56 596] (!)
$B^{0.70}$	**47 58 52** $\langle\mathcal{F}_{0.70} : 55\rangle$ [61 342; 43 956] (675s)	**42 64 51** $\langle\mathcal{F}_{0.70} : 58\rangle$ [56 561; 47 322] (746s)	**13 37 19** $\langle\mathcal{F}_{0.70} : 27\rangle$ [274 001; 74 138] (!)	**23 29 26** $\langle\mathcal{F}_{0.70} : 27\rangle$ [354 773; 147 994] (!)	**6 13 8** $\langle\mathcal{F}_{0.70} : 10\rangle$ [1 882 873; 46 418] (!)
$B2^{0.30}$	**99 42 59** $\langle F_{0.30} : 77\rangle$ [149939; 71974] (5s)	**90 37 52** $\langle F_{0.30} : 69\rangle$ [160431; 81969] (4s)	**42 13 20** $\langle F_{0.30} : 29\rangle$ [839308; 74970] (18s)	**63 18 28** $\langle F_{0.30} : 42\rangle$ [672965; 222911] (98s)	**27 6 9** $\langle F_{0.30} : 15\rangle$ [2268294; 64422] (644s)
$B2^{0.50}$	**82 48 61** $\langle F_{0.50} : 61\rangle$ [115340; 68121] (6s)	**67 51 58** $\langle F_{0.50} : 58\rangle$ [107351; 67874] (4s)	**25 22 23** $\langle F_{0.50} : 23\rangle$ [644731; 72683] (27s)	**41 24 30** $\langle F_{0.50} : 30\rangle$ [557701; 192483] (141s)	**14 8 10** $\langle F_{0.50} : 10\rangle$ [2122955; 55169] (686s)
$B2^{0.70}$	**52 58 55** $\langle F_{0.70} : 56\rangle$ [69970; 49250] (9s)	**44 65 52** $\langle F_{0.70} : 59\rangle$ [63373; 47332] (5s)	**13 35 19** $\langle F_{0.70} : 26\rangle$ [358767; 67960] (49s)	**22 31 26** $\langle F_{0.70} : 29\rangle$ [427653; 147255] (417s)	**6 13 8** $\langle F_{0.70} : 10\rangle$ [1898536; 45225] (901s)
Louv	**0 84 0** [207; 207] (8s)	**0 94 0** [237; 237] (3s)	**0 74 0** [430; 430] (14s)	**0 84 0** [6 878; 6 878] (13s)	**0 68 0** [6 074; 6 074] (31s)

Table 4: **Compare the performances of $BEC.2$ against $Louvain$ as a time baseline and BEC as a quality baseline.** $precision$, $recall$ and $\mathcal{F}_\sigma$ are multiplied by 100; $<\!\multimap x>$ for $\Big|\Big\{i \in V$ such $\big|\{\{i,j\} \in E\}\big| = 1\Big\}\Big| = x$; B for BEC; $B2$ for $BEC.2$; $Louv$ for $Louvain$; $\langle\mathcal{F}_x : y\rangle$ for $\mathcal{F}_x(met(G), G) = y$; $[x; y]$ for $|met(G)| = x$ and $|\{\mathcal{C}_i \in met(G)$ such $|\mathcal{C}_i| > 1\}| = y$; (xs) for x is the computation time in seconds of $met(G)$ and (!) if it took over one hour to run.

3.1 Evaluation on terrain graphs

In this section we compare the performances on five terrain graphs:

- $\mathbf{G_{DBLP}}$: The DBLP computer science bibliography provides a comprehensive list of research papers in computer science (*14*). 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 (*14*). 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`.

- $\mathbf{G_{retwwet}}$: Nodes are twitter users and edges are retweets. These were collected from various social and political hashtags. `https://networkrepository.com/rt-retweet-crawl.php`.

- $\mathbf{G_{youtube}}$: Youtube is a video-sharing web site that includes a social network. The dataset contains a list of all of the user-to-user links. `https://networkrepository.com/soc_youtube_snap.php`.

- $\mathbf{G_{WikiTalk}}$: Nodes in the network represent Wikipedia users (*13*) and a directed edge from node i to node j represents that user i at least once edited a talk page of user j, then the graph is symmetrized. `https://snap.stanford.edu/data/wiki-Talk.html`.

■ **We can see in Tab. 4** that $BEC.2$, performs the same or better than BEC and is ~ 100 times faster than it, but still remains slower than $Louvain$. It should be noted that:

- $Louvain$ returns few modules strongly favorizing $recall$ at the cost of a very large degradation of $precision$ (always $P \approx 0$ on these five graphs);

- $BEC.2$ returns a lot of modules, it is because when there are many nodes with one alone neighbor in a graph (e.g., as we can see in Tab. 4, $G_{WikiTalk}$ has 2 516 783 nodes, however it has 1 843 811 nodes with only one neighbor.) then these nodes are often isolated in their own modules by $BEC.2$ in order to best optimize $\mathcal{F}_\sigma$.

3.2 Evaluation on artificial graphs

3.2.1 Evaluation on Random graph

$Benchmark_{ER}$ is the class of Random graphs studied by Erdös and Rényi (*8,9*) with parameters N the number of nodes and p the connection probability between two nodes. Random graphs do not have meaningful group structures, and they can be used to test if the algorithms are able to recognize the absence of group structures. An algorithm is able to recognize the absence

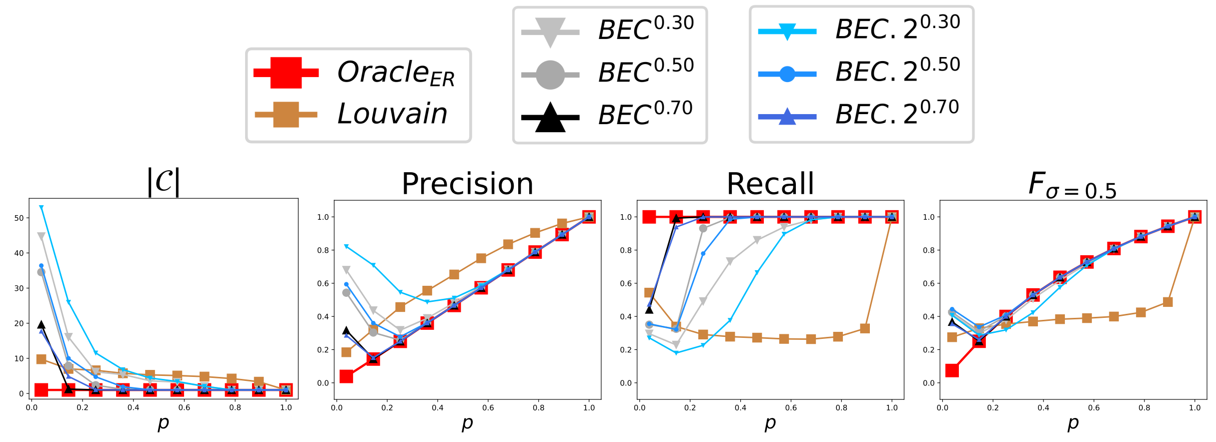


Figure 4: **Performance with** $Benchmark_{ER}$**.** In these four figures, for each point $(x = p, y)$ of each of the 8 curves per figure, y is the average over 100 random graphs $G_{x=p}^{N=128}$. For each point $(x = p, y)$ of the $|\mathcal{C}|$ curves, the standard deviation of y is always lower than 3.

of group structures in a random graph G_p^N if its returned clustering $\mathcal{C}$ is such that $|\mathcal{C}| = 1$ or $|\mathcal{C}| = N$. Therefore, we set $N = 128$ and we will study the accuracy of the methods with $Benchmark_{ER}$ according to p.

Let $G_{ER} = (V_{ER}, E_{ER})$ a random graph built by $Benchmark_{ER}$, $\Gamma_{ER} = \{V\}$ with only one cluster, and $Oracle_{ER}(G_{ER}) = \Gamma_{ER} = \{V\}$ the Oracle's method who knows $\Gamma_{ER} = \{V\}$ but ignores E_{ER} the concretly constructed edges.

We show in Fig. 4 the accuracy of the methods according to p. We can see that:

- $\mathbf{Oracle_{ER}}$: It knows $\Gamma_{ER} = \{V\}$, but does not know the concretely constructed edges E_{ER}. Its numbers of clusters are always $= 1$. Its *precision* scores increases when p increases, because *density* increases. Its *recall* scores are always $= 1$. Its $\mathcal{F}_{\sigma=0.5}$ scores increase;
- **Louvain** : $|Louvain(G_p)| = 1$ only when $p = 1$, it therefore does not recognize the absence of group structures in Random Graphs;
- $\mathbf{BEC^s}$: When p is large enough, they returns one alone cluster $\{V\}$, thus recognizing the absence of group structures in Random Graphs. The larger s, the less p needs to be large to they recognize the absence of group structure;
- $\mathbf{BEC.2^s}$: Same as BEC^s. To they recognize the absence of group structure, p needs to be more large than for BEC^s;

3.3 Evaluation on $Benchmark_{LFR}$

In order to allow to systematically study the behavior of clustering methods relative to the complex nature of the community structure in real networks, Lancichinetti, Fortunato and Radicchi proposed $Benchmark_{LFR}$ (*12*). The graphs $G_{\mu}^{N,k,a,b,on,om}$ in $Benchmark_{LFR}$ are parameterized [2] with:

- N their number of nodes;
- k their average degree;
- a the power law exponent of their degree distribution;
- b the power law exponent of their community sizes distribution;
- $\mu \in [0, 1]$ their mixing parameter: Each node shares a fraction $1 - \mu$ of its links with the other nodes of its community and a fraction μ with the other nodes of the graph.
- on the number of overlapping nodes;
- om the number of memberships of the overlapping nodes.

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.

[2]Code to generate $Benchmark_{LFR}$ graphs can be downloaded from Andrea Lancichinetti's homepage `https://sites.google.com/site/andrealancichinetti/home`.

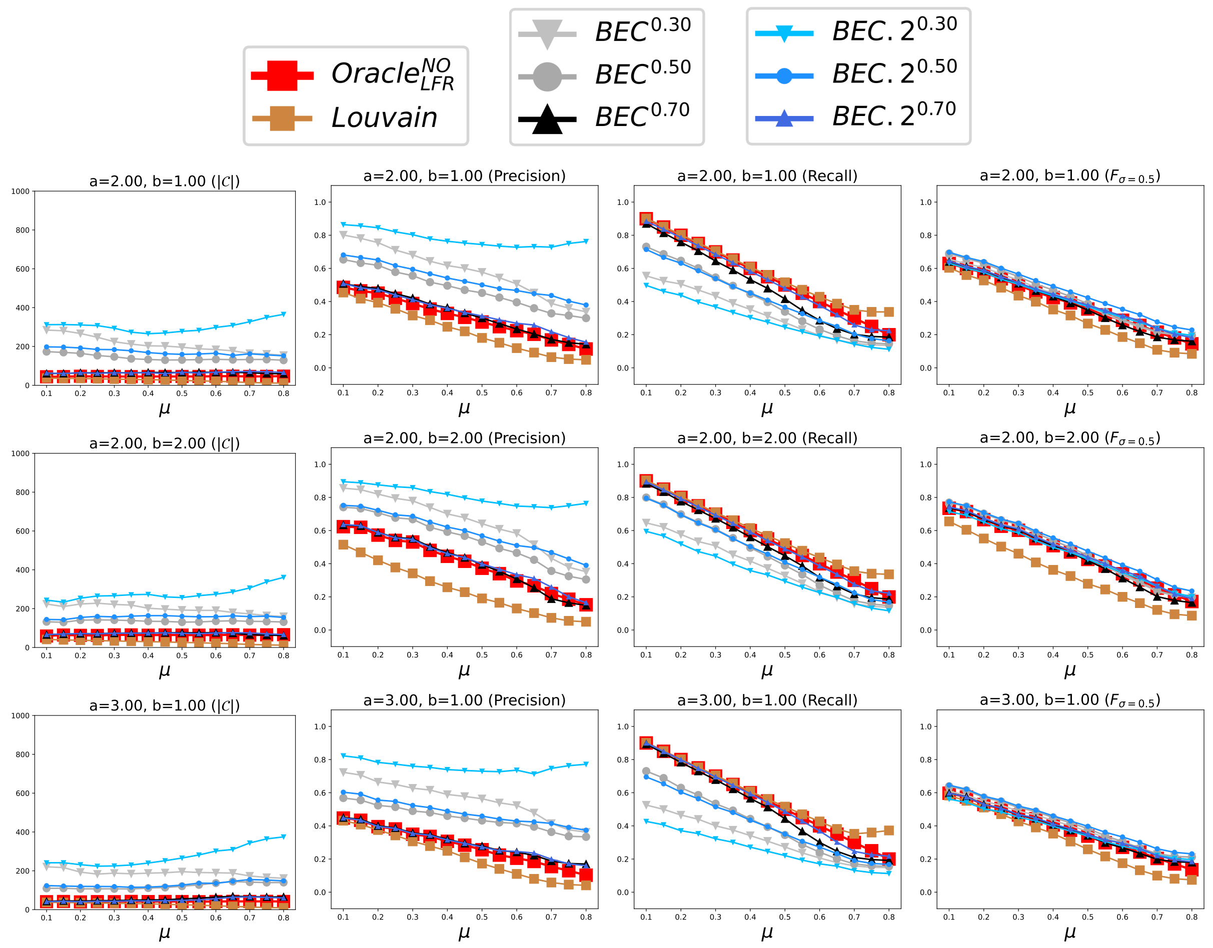


Figure 5: **Performance with** $Benchmark^{NO}_{LFR}$ **(k = 15).** In these 12 figures, for each point $(x = \mu, y)$ of each of the 8 curves per figure, y is the average over 100 graphs $G^{N=1000,k=15,on=0,om=0,a,b}_{x=\mu}$. For each point $(x = \mu, y)$ of the $F_{\sigma=0.5}$ curves the standard deviation of y is always lower than 0.05.

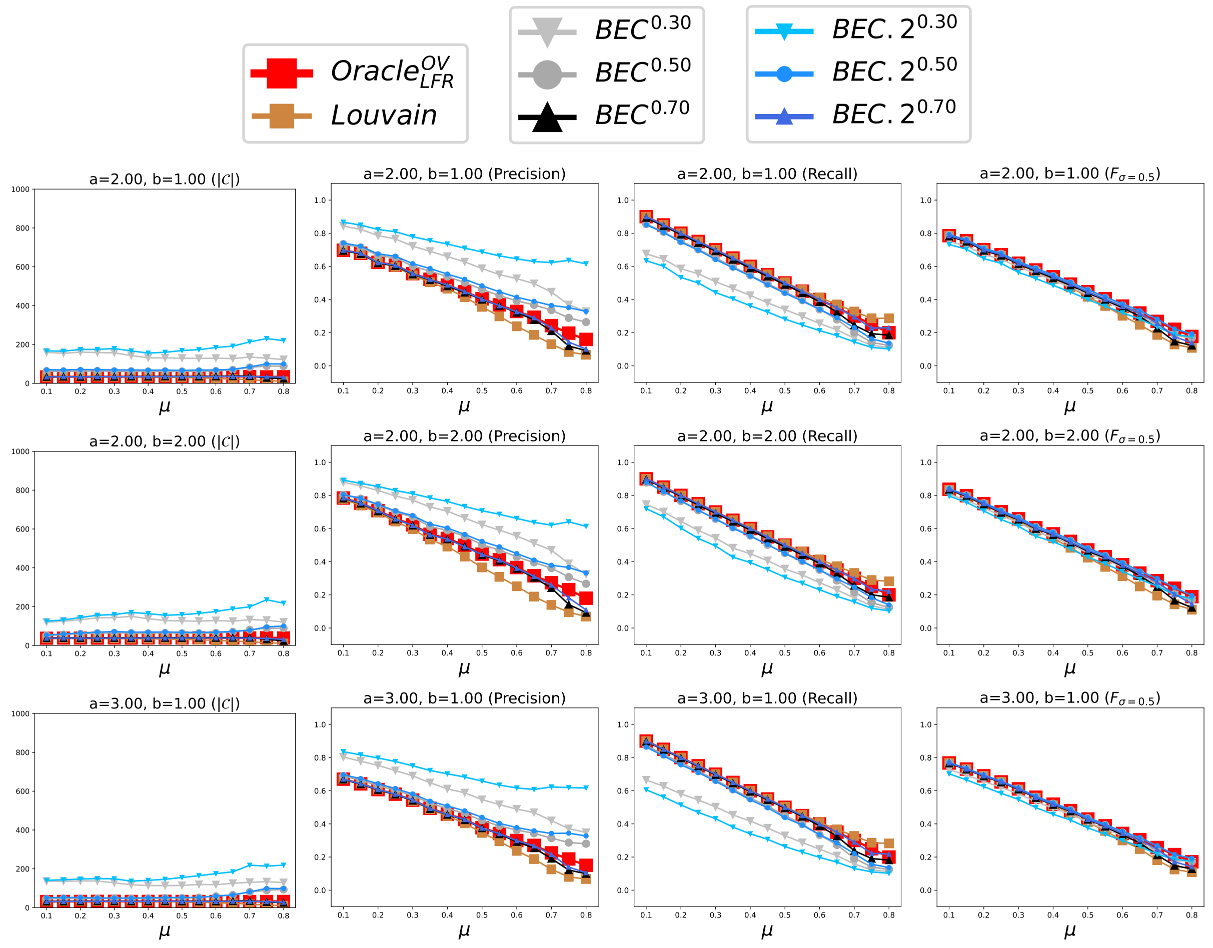


Figure 6: **Performance with** $Benchmark^{NO}_{LFR}$ **(k = 25).** In these 12 figures, for each point $(x = \mu, y)$ of each of the 8 curves per figure, y is the average over 100 graphs $G^{N=1000,k=25,on=0,om=0,a,b}_{x=\mu}$. For each point $(x = \mu, y)$ of the $F_{\sigma=0.5}$ curves the standard deviation of y is always lower than 0.05.

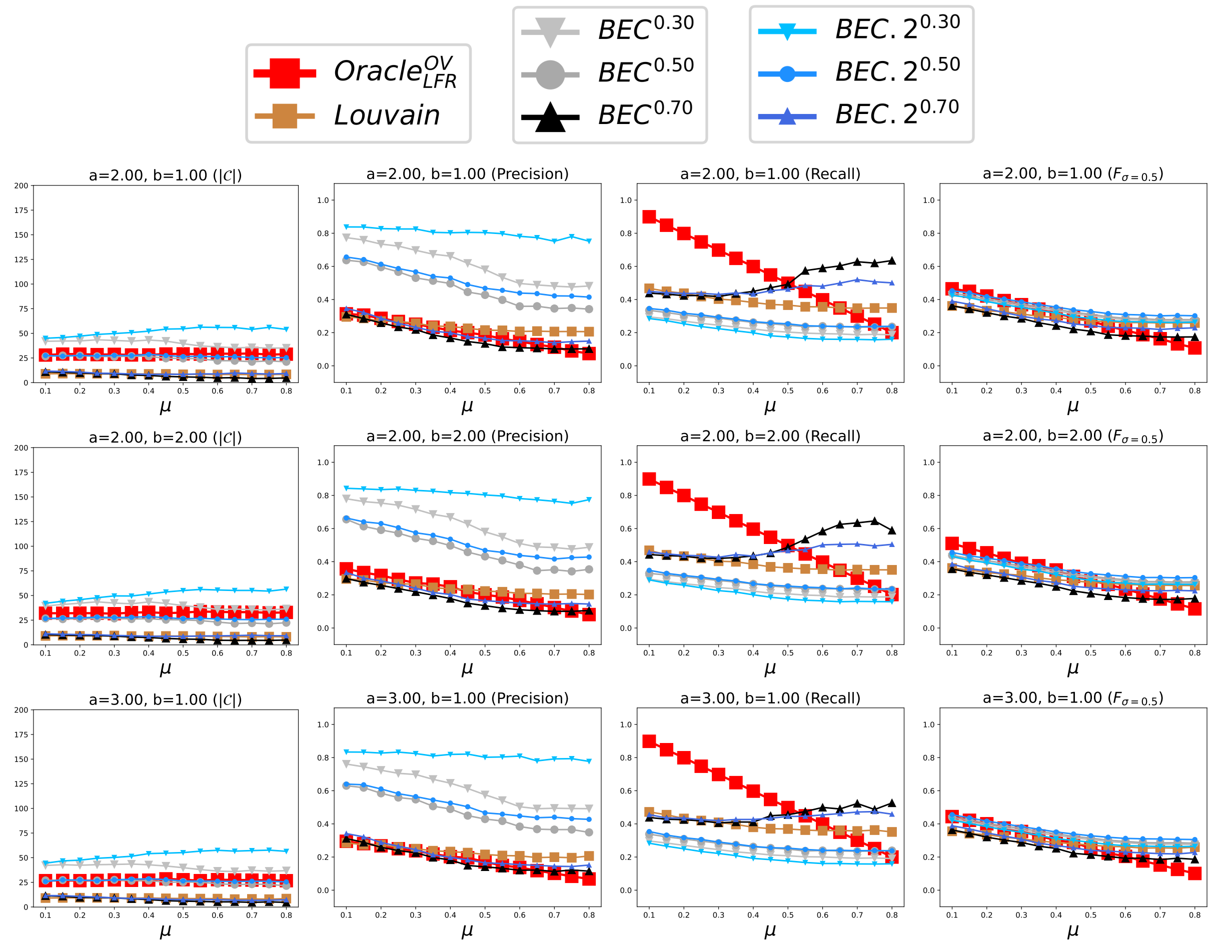


Figure 7: **Performance with** $Benchmark^{OV}_{LFR}$ **(k = 15).** In these 12 figures, for each point $(x = \mu, y)$ of each of the 8 curves per figure, y is the average over 100 graphs $G^{N=200,k=15,on=100,om=4,a,b}_{x=\mu}$. For each point $(x = \mu, y)$ of the $\mathcal{F}_{\sigma=0.5}$ curves the standard deviation of y is always lower than 0.05.

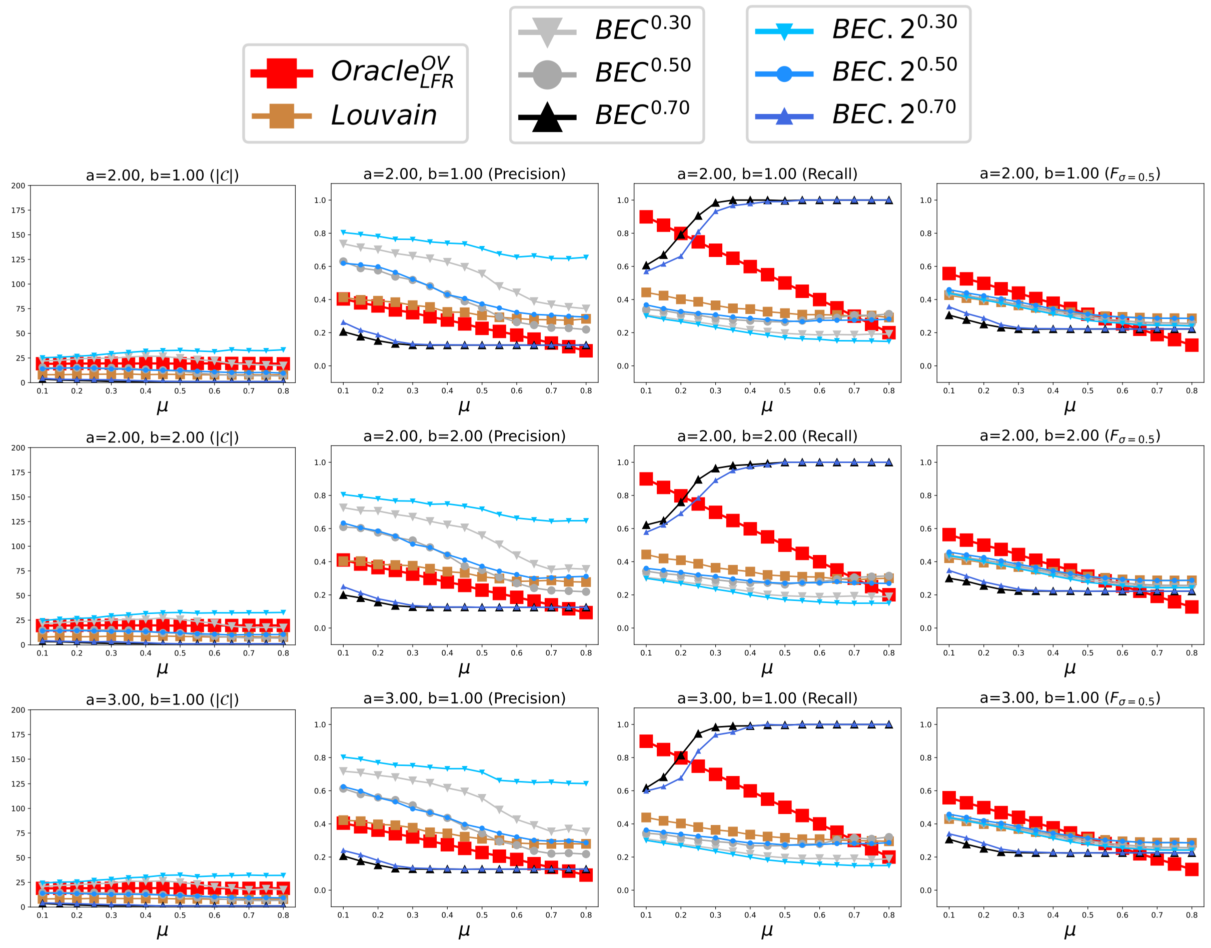


Figure 8: **Performance with** $Benchmark_{LFR}^{OV}$ **(k = 25).** In these 12 figures, for each point $(x = \mu, y)$ of each of the 8 curves per figure, y is the average over 100 graphs $G_{x=\mu}^{N=200,k=25,on=100,om=4,a,b}$. For each point $(x = \mu, y)$ of the $\mathcal{F}_{\sigma=0.5}$ curves the standard deviation of y is always lower than 0.05.

Performance on $Benchmark_{LFR}^{NO}$ **without overlaps:** We set $on = 0$, $om = 0$, $N = 1000$, and $k = 15$ or $k = 25$, $(a = 2, b = 1)$ or $(a = 2, b = 2)$ or $(a = 3, b = 1)$ and for each of these six configurations, we study the accuracy of the methods according to μ.

Let $G_{LFR}^{NO} = (V_{LFR}^{NO}, E_{LFR}^{NO})$ a graph built by $Benchmark_{LFR}^{NO}$, Γ_{LFR}^{NO} its expected modules as expected overconnected regions, and $Oracle_{LFR}^{NO}(G_{LFR}^{NO}) = \Gamma_{LFR}^{NO}$ the Oracle's method which knows Γ_{LFR}^{NO} but ignores E_{LFR}^{NO} the concretly constructed edges. We show in Fig. 5 and 6 the accuracy of the methods according to μ. We can see that:

- **Oracle$_{\mathbf{LFR}}^{\mathbf{NO}}$** : It knows Γ_{LFR}^{NO}, but does not know the concretely constructed edges E_{LFR}^{NO}.

Its number of clusters is always $|\Gamma_{LFR}^{NO}|$. Its $precision$ scores decreases when μ increase, because there are more and more non-edges in the expected modules, but $Oracle_{LFR}^{NO}$ does not know it. Its $recall$ scores decreases when μ increase, because there are more and more edges outside the expected modules, but $Oracle_{LFR}^{NO}$ does not know it. Its $\mathcal{F}_{\sigma=0.5}$ scores decreases when μ increase;

- $\mathbf{Louvain}$: Always $\mathcal{F}_{\sigma=0.5}(Louvain(G_\mu), G_\mu) \leqslant \mathcal{F}_{\sigma=0.5}(Oracle_{LFR}^{NO}, G_\mu)$ and often its $\mathcal{F}_{\sigma=0.5}$ scores are the lowest of the eight methods studied here;
- $\mathbf{BEC^s}$: Always $\mathcal{F}_{\sigma=0.5}(Oracle_{LFR}^{NO}, G_\mu) \leqslant \mathcal{F}_{\sigma=0.5}(BEC^{s=0.5}(G_\mu), G_\mu)$;
- $\mathbf{BEC.2^s}$: Always $\mathcal{F}_{\sigma=0.5}(BEC^{s=0.5}(G_\mu), G_\mu) \leqslant \mathcal{F}_{\sigma=0.5}(BEC.2^{s=0.5}(G_\mu), G_\mu)$;

Performance on $Benchmark_{LFR}^{OV}$ with overlaps: We set $on = 100$ and $om = 4$, $N = 200$, and $k = 15$ or $k = 25$, $(a = 2, b = 1)$ or $(a = 2, b = 2)$ or $(a = 3, b = 1)$ and for each of these six configurations, study the accuracy of the methods according to μ.

Let $G_{LFR}^{OV} = (V_{LFR}^{OV}, E_{LFR}^{OV})$ a graph built by $Benchmark_{LFR}^{OV}$, Γ_{LFR}^{OV} its expected modules as expected overconnected regions, and $Oracle_{LFR}^{OV}(G_{LFR}^{OV}) = \Gamma_{LFR}^{OV}$ the Oracle's method which knows Γ_{LFR}^{OV} but ignores E_{LFR}^{OV} the concretly constructed edges. We show in Fig. 7 and 8 the accuracy of the methods according to μ. We can see that:

- $\mathbf{Oracle_{LFR}^{OV}}$: It knows Γ_{LFR}^{OV}, but does not know the concretely constructed edges E_{LFR}^{OV}. Its number of clusters is always $|\Gamma_{LFR}^{OV}|$. Its $precision$ decreases when μ increase, because there are more and more non-edges in the expected modules, but $Oracle_{LFR}^{OV}$ does not know it. Its $recall$ decreases when μ increase, because there are more and more edges outside the expected modules, but $Oracle_{LFR}^{OV}$ does not know it. Its $\mathcal{F}_{\sigma=0.5}$ decreases when μ increase;
- $\mathbf{Louvain}$: [When overconnected regions are **clear**, $\mathcal{F}_{\sigma=0.5}$ are low]; [When overconnected regions are **less clear**, $\mathcal{F}_{\sigma=0.5}$ are better than that of $Oracle_{LFR}^{OV}$];
- $\mathbf{BEC^s}$: [When overconnected regions are **clear**, $\mathcal{F}_{\sigma=0.5}$ are low]; [When overconnected regions are **less clear**, $\mathcal{F}_{\sigma=0.5}$ are better than that of $Oracle_{LFR}^{OV}$];
- $\mathbf{BEC.2^s}$: [When overconnected regions are **clear**, $\mathcal{F}_{\sigma=0.5}$ are low]; [When overconnected regions are **less clear**, $\mathcal{F}_{\sigma=0.5}$ are better than that of $Oracle_{LFR}^{OV}$]. Always $\mathcal{F}_{\sigma=0.5}(Louvain(G_\mu), G_\mu) \leqslant \mathcal{F}_{\sigma=0.5}(BEC.2^{s=0.5}(G_\mu), G_\mu)$. Always $\mathcal{F}_{\sigma=0.5}(BEC^{s=0.5}(G_\mu), G_\mu) \leqslant \mathcal{F}_{\sigma=0.5}(BEC.2^{s=0.5}(G_\mu), G_\mu)$.

3.4 Computation times and sapce memory

In Tab. 5 we show the computation times for each method on the 19 005 graphs used for evaluation of the methods in the section 3. We can see that $BEC.2$ is approximately 100 times faster than BEC while it is approximately 10 to 20 times slower than $Louvain$ (depending of s the desired scale of description).

On the other hand, to cluster a graph $G = (V, E)$, in the worst case, the memory space occupied by $Louvain$ and $BEC.2$ is $O(E)$, and it is $O(V^2)$ by BEC (because the computaion of the similarities $cos(G = (V, E), x, y)$ for all $\{x, y\} \in E$). Since in general terrain graphs are sparse where E is $O(|V|.log(|V|))$, the memory space required for $Louvain$ and $BEC.2$ is therefore in general less than for BEC.

	Terrain [5]	$\mathbf{Bench_{ER}}$ [1 000]	$\mathbf{Bench^{NO}_{LFR}}$ [9 000]	$\mathbf{Bench^{OV}_{LFR}}$ [9 000]	Total [19 005]
Louvain	69	2	49	12	**132**
$\mathbf{BEC^{s=0.30}}$	85 041	31	1 026	167	**86 265**
$\mathbf{BEC^{s=0.50}}$	85 366	23	1 368	289	**87 046**
$\mathbf{BEC^{s=0.70}}$	85 887	26	3 177	646	**89 736**
$\mathbf{BEC.2^{s=0.30}}$	769	5	271	37	**1 082**
$\mathbf{BEC.2^{s=0.50}}$	864	5	328	58	**1 255**
$\mathbf{BEC.2^{s=0.70}}$	1 381	1	629	127	**2 138**

Table 5: **Computaion times in seconds.** $[x]$ for x is the number of graphs in the concerned categories.

4 Clustering with overlaps

$\forall G = (V, E) \in \mathcal{G}(V)$, then $E \in \mathcal{C}(V)$, $\mathcal{P}(E, G) = \mathcal{R}(E, G) = 1$ and generaly E is an overlaping clustering such $|V| < |E|$. To enable us to find an overlapping clustering $\mathcal{C}_o$ of a graph $G = (V, E)$ such $|\mathcal{C}_o| \leqslant |V|$, we propose below the Algorithm $OVP.ind_{s_o}(G, \mathcal{C}_p)$ which is based on gluantly extending the clusters of a partitional clustering $\mathcal{C}_p$ through the optimization of $\mathcal{F}_{s_o}(\mathcal{C}_p, G)$ with the stickiness scale s_o defining the desired amount of overlap.

It is clear that $|\mathcal{C}_o| \leqslant |\mathcal{C}_p| \leqslant |V|$ (because `(Line₁)` and $\mathcal{C}_p$ is a partitional clustering). Moreover $\forall s_o \in [0, 1], \Xi(\mathcal{C}_p) \subseteq \Xi(\mathcal{C}_o)$ (because we only add nodes to the clusters of $\mathcal{C}_p$). This has the direct consequence: $\forall\ s_o \in [0, 1]$, $\mathcal{R}(\mathcal{C}_p, G) \leqslant \mathcal{R}(\mathcal{C}_o, G)$.

In the following we will note $BEC.2^{s_p}_{s_o}$ for $OVP.ind_{s_o}(G, \mathcal{C}_p = BEC.2^{s_p}(G))$.

For example with the graph G_{toy} in Tab. 1:

- $\mathcal{C}_p = BEC.2^{0.6}(G_{toy}) = \big\{\{0,2,3\},\{1,4,5\}\big\}$: $\mathcal{P}(\mathcal{C}_p, G) = 1.00$, $\mathcal{R}(\mathcal{C}_p, G) = 0.86$, $\mathcal{F}_{0.5}(\mathcal{C}_p, G) = 0.92$, $\mathcal{F}_{0.6}(\mathcal{C}_p, G) = 0.90$;
- $\mathcal{C}_o = BEC.2^{0.6}_{0.6}(G_{toy}) = \big\{\{0,2,3\},\{0,1,4,5\}\big\}$: $\mathcal{P}(\mathcal{C}_o, G) = 0.78$, $\mathcal{R}(\mathcal{C}_o, G) = 1.00$, $\mathcal{F}_{0.5}(\mathcal{C}_o, G) = 0.88$, $\mathcal{F}_{0.6}(\mathcal{C}_o, G) = 0.91$;

Algorithm: $\mathcal{C}_o = OVP.ind_{s_o}(G, \mathcal{C}_p)$**:** To gluantly extend the clusters of $\mathcal{C}_p$

```
Input: $G = (V, E) \in \mathcal{G}(V)$
       $s_o \in [0,1]$
       $\mathcal{C}_p$ A partitional clustering of $G$
Output: $\mathcal{C}_o \in \mathcal{C}(V)$

    Initialization: $\mathcal{C} \hookleftarrow \mathcal{C}_p$;
    Loop on i $\in$ V: For each node $i$:
        Add Loop on neighbors of i: For each neighbor $j$ of $i$, improve the curent clustering $\mathcal{C}$ by adding $i$ to the cluster of $j$. If it cannot improve $\mathcal{F}_{s_o}(\mathcal{C}, G)$ then we examine the next neighbor of $i$;
    $\mathcal{C}_o \hookleftarrow \{\mathbf{C_i} \in \mathcal{C}$ such $\nexists \mathbf{C_j} \in \mathcal{C} \mid \mathbf{C_i} \subsetneq \mathbf{C_j}\}$ ► (Line$_1$)
    Return: Return $\mathcal{C}_o$.
```

Fig 9 and 10 compare the performances of $BEC.2^s$ and $BEC.2^s_s$ on the the email graph G_{em} and Table 6 on the same five terrain graphs as those in Table 4.

■ **We can see in Fig. 9** that $\mathcal{R}\big(BEC.2^s(G_{em}), G_{em}\big) \leqslant \mathcal{R}\big(BEC.2^s_s(G_{em}), G_{em}\big)$.

■ **We can see in Fig. 10** $\mathcal{F}_\sigma\big(BEC.2^\sigma(G_{em}), G_{em}\big) \leqslant \mathcal{F}_\sigma\big(BEC.2^\sigma_\sigma(G_{em}), G_{em}\big)$.

■ **We can see in Tab. 6** that for these five terrain graphs, $\mathcal{R}\big(BEC.2^s(G), G\big) < \mathcal{R}\big(BEC.2^s_s(G), G\big)$ and $\mathcal{F}_\sigma\big(BEC.2^\sigma(G), G\big) < \mathcal{F}_\sigma\big(BEC.2^\sigma_\sigma(G), G\big)$.

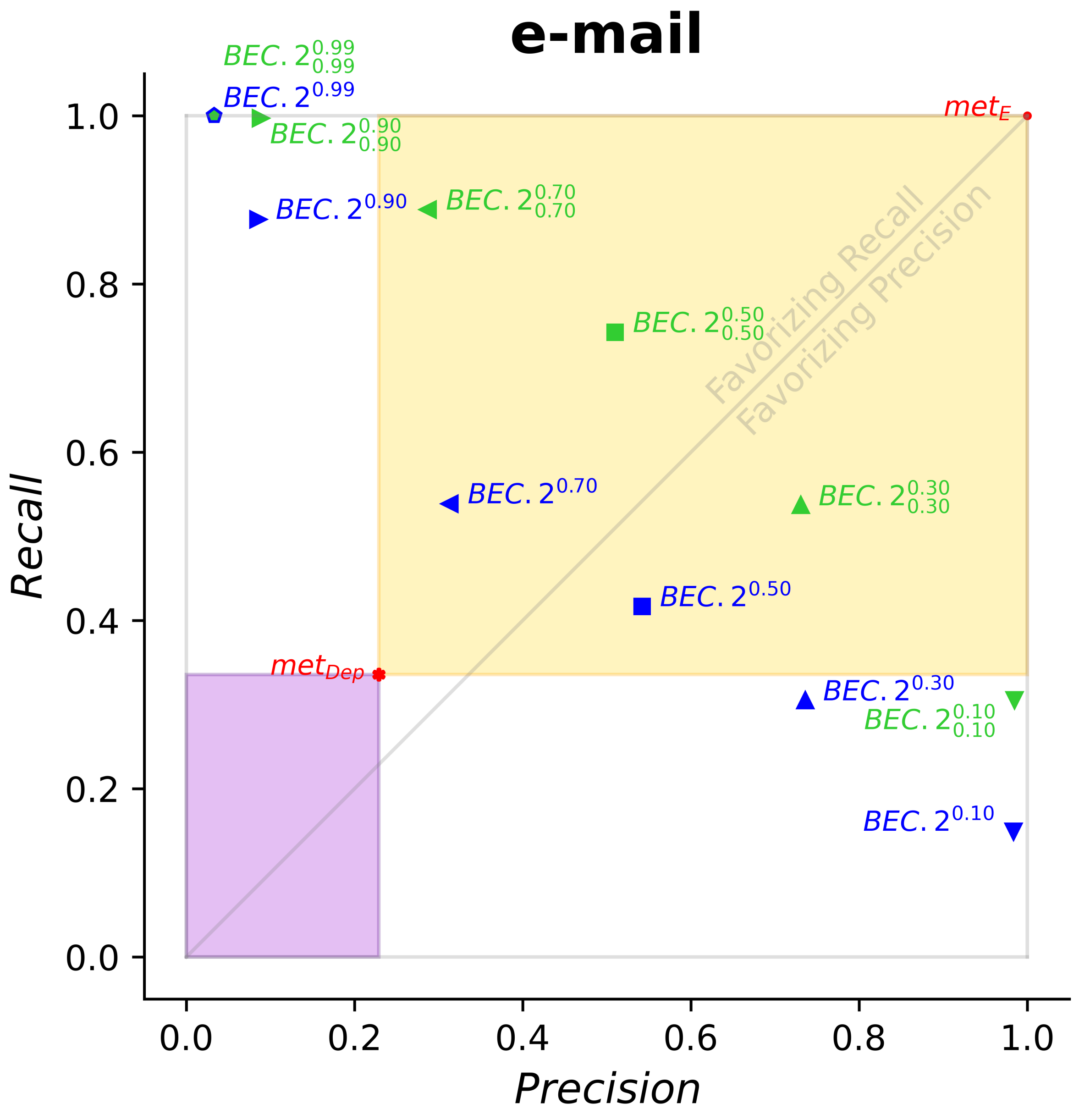


Figure 9: **Performances in the 2-dimensional space** precision × recall **of** BEC.2 $^{\text{sp}}$ **and** BEC.2 $^{\text{sp}}_{\text{sp}}$ **when applied to the e-mail graph** $\mathbf{G_{em}}$**.** The Oracle method met_{Dep} and the Omniscient method met_E are highligthed in red.

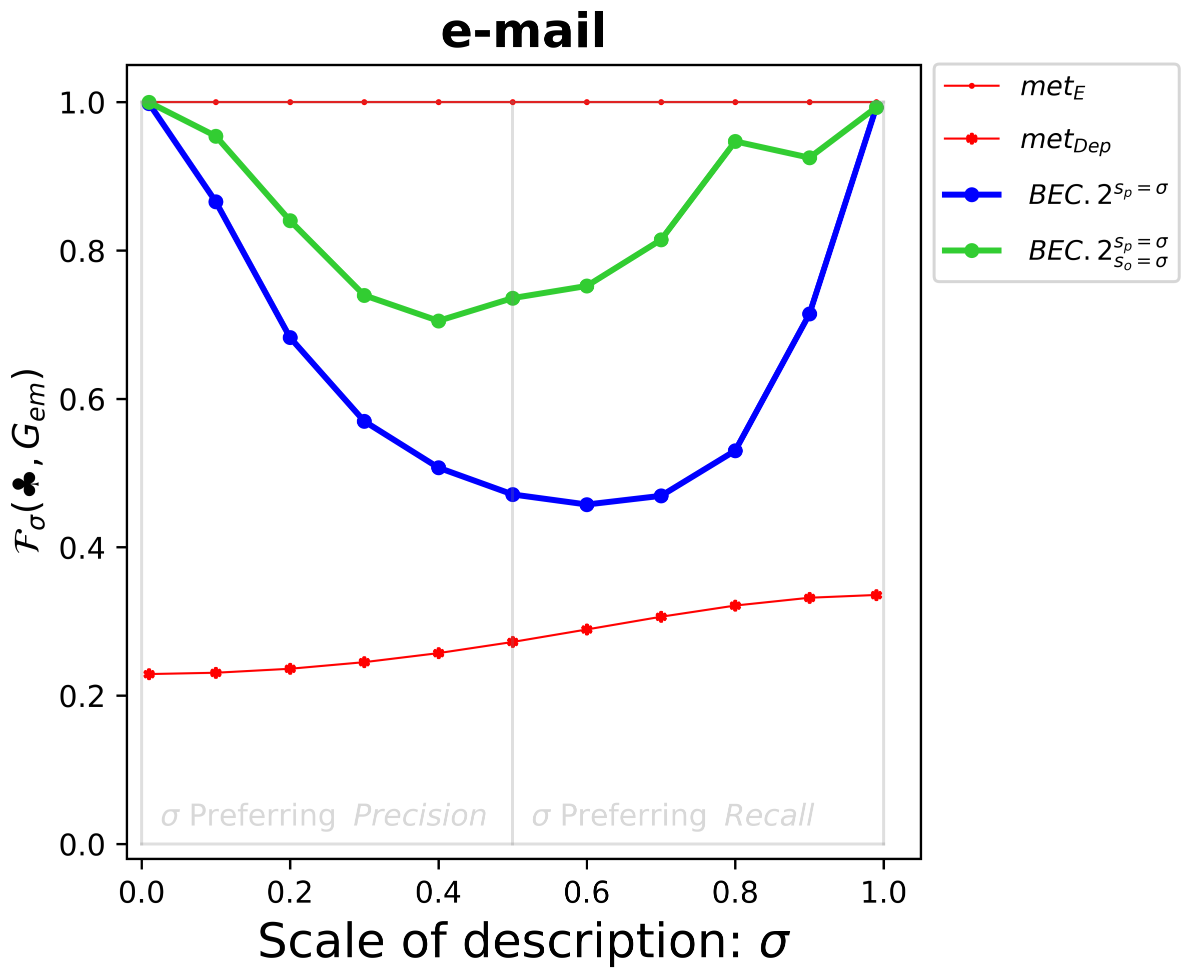


Figure 10: **Performances $\mathcal{F}_\sigma(♣, \mathrm{G_{em}})$ of clustering methods $\mathrm{method}(\mathrm{G_{em}}) = ♣$ under the description scale σ.** The Oracle method met_{Dep} and the Omniscient method met_E are highligthed in red.

met	$\mathbf{G_{dblp}}$ **P R** $\mathcal{F}_{0.5}$	$\mathbf{G_{amazon}}$ **P R** $\mathcal{F}_{0.5}$	$\mathbf{G_{retweet}}$ **P R** $\mathcal{F}_{0.5}$	$\mathbf{G_{youtube}}$ **P R** $\mathcal{F}_{0.5}$	$\mathbf{G_{WikiTalk}}$ **P R** $\mathcal{F}_{0.5}$
$B2^{0.30}$	**99 42 59** $\langle F_{0.30} : 77\rangle$ $(5s)$	**90 37 52** $\langle F_{0.30} : 69\rangle$ $(4s)$	**42 13 20** $\langle F_{0.30} : 29\rangle$ $(18s)$	**63 18 28** $\langle F_{0.30} : 42\rangle$ $(98s)$	**27 6 9** $\langle F_{0.30} : 15\rangle$ $(644s)$
$B2^{0.50}$	**82 48 61** $\langle F_{0.50} : 61\rangle$ $(6s)$	**67 51 58** $\langle F_{0.50} : 58\rangle$ $(4s)$	**25 22 23** $\langle F_{0.50} : 23\rangle$ $(27s)$	**41 24 30** $\langle F_{0.50} : 30\rangle$ $(141s)$	**14 8 10** $\langle F_{0.50} : 10\rangle$ $(686s)$
$B2^{0.70}$	**52 58 55** $\langle F_{0.70} : 56\rangle$ $(9s)$	**44 65 52** $\langle F_{0.70} : 59\rangle$ $(5s)$	**13 35 19** $\langle F_{0.70} : 26\rangle$ $(49s)$	**22 31 26** $\langle F_{0.70} : 29\rangle$ $(417s)$	**6 13 8** $\langle F_{0.70} : 10\rangle$ $(901s)$
$B2^{0.30}_{0.30}$	**99 81 89** $\langle F_{0.30} : 95\rangle$ $(1s)$	**92 64 75** $\langle F_{0.30} : 84\rangle$ $(1s)$	**68 55 61** $\langle F_{0.30} : 65\rangle$ $(3s)$	**71 45 55** $\langle F_{0.30} : 64\rangle$ $(10s)$	**71 60 65** $\langle F_{0.30} : 68\rangle$ $(62s)$
$B2^{0.50}_{0.50}$	**80 86 83** $\langle F_{0.50} : 83\rangle$ $(1s)$	**66 81 72** $\langle F_{0.50} : 72\rangle$ $(1s)$	**39 59 47** $\langle F_{0.50} : 47\rangle$ $(4s)$	**47 58 52** $\langle F_{0.50} : 52\rangle$ $(25s)$	**45 68 54** $\langle F_{0.50} : 54\rangle$ $(135s)$
$B2^{0.70}_{0.70}$	**47 93 62** $\langle F_{0.70} : 77\rangle$ $(1s)$	**39 91 55** $\langle F_{0.70} : 72\rangle$ $(2s)$	**16 64 26** $\langle F_{0.70} : 40\rangle$ $(8s)$	**22 74 34** $\langle F_{0.70} : 50\rangle$ $(72s)$	**20 72 31** $\langle F_{0.70} : 47\rangle$ $(316s)$

Table 6: **Compare the performances of** $BEC.2^{s_p}(G)$ **and** $BEC.2^{s_p}_{s_o}(G)$**.** $precision$, $recall$ and $\mathcal{F}_\sigma$ are multiplied by 100; $B2$ for $BEC.2$; $\langle \mathcal{F}_x : y\rangle$ for $\mathcal{F}_x(met(G), G) = y$; (xs) for x is the computation time in seconds of $met(G)$.

5 Conclusion

(1) It is shown in the recent paper (*10*) that in the $nPnB$ framework, BEC outperforms most state-of-the-art methods (including $spectral\ graph\ clustering$, one of the best efficient state-of-the-art methods);

(2) We showed in section 3 that in the $nPnB$ framework, $BEC.2$ outperforms BEC.

(1) & (2) imply that in the $nPnB$ framework, $BEC.2$ outperforms most state-of-the-art methods, i.e. better satisfying the two properties P_{DC} and P_{WC}:

$\mathbf{P_{DC}}$: Each community is *Densely Connected*;

$\mathbf{P_{WC}}$: Communities are *Weakly Connected* to each other.

From physical sciences to biological or social sciences, complex systems are defined as large sets of entities interacting in a decentralized ways. Graphs are one of the main conceptual structures for modeling them, where nodes represent the basic lowest-level entities and edges represent their interactions. $BEC.2^{s_p}_{s_o}(G)$ is a kind of telescope for observing the graphs modeling complex systems, where s_p defines the desired scale of description of the modules as highest-level entities and s_o the level of overlap of these entities. $BEC.2$ is 10 to 20 times slower than $Louvain$ but is 100 times faster than BEC, which allows working with large graphs in reasonable time.

5.1 Perspectives

Gain speed: $BEC.2$ remains slower than $Louvain$, define a high-performance algorithm in the $nPnB$ framework, as fast as $Louvain$ should be part of further developements.

Directed weighted edges and temporal dimension: Many complex networks have directed weighted edges and have a temporal dimension wich are not been addressed with $BEC.2$ and should be part of further developements.

Attributed graph clustering: When nodes of the graph also have some attributes, it is possible to define clusterings that take these attributes into account (see for exemple (*2–4, 6, 7, 11, 17, 18*)). Attributed graph clustering can be perfectly hybridised with $BEC.2$, but it will still be necessary to assess beforehand the relative weight given to the information contained in the links and that contained in the attributes of the nodes and should be part of further developements.

Intrinsic description scales: The values s_i maximizing $\mathcal{F}_{0.5}(BEC.2^{s_i}_{s_i}(G), BEC.2^{s_i}(G))$ can help define the *intrinsic description scales* of G and its *intrinsic ground truths* without overlaps $\mathcal{C}^{s_i} = BEC.2^{s_i}(G)$ or with overlaps $\mathcal{C}^{s_i}_{s_i} = BEC.2^{s_i}_{s_i}(G)$ (see (*10, 16*)). The search for such s_i with short computation time should be the subject of further developments.

Data availability

The code is available at https://github.com/Brngm/nPnB.BEC2.

Funding

This work was supported by the Complex Systems Institute of Paris Île-de-France (ISC-PIF) and the EU NODES project (LC-01967516).

References

1. E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, A comparison of extrinsic clustering evaluation metrics based on formal constraints, Information Retrieval Journal, 12 (2009), pp. 461–486.

2. K. Berahmand, S. Bahadori, M. N. Abadeh, Y. Li, and Y. Xu, Sdac-da: Semi-supervised deep attributed clustering using dual autoencoder, IEEE Transactions on Knowledge and Data Engineering, (2024).

3. K. Berahmand, Y. Li, and Y. Xu, Dac-hpp: deep attributed clustering with high-order proximity preserve, Neural Computing and Applications, 35 (2023), pp. 24493–24511.

4. K. Berahmand, M. Mohammadi, R. Sheikhpour, Y. Li, and Y. Xu, Wsnmf: Weighted symmetric nonnegative matrix factorization for attributed graph clustering, Neurocomputing, 566 (2024), p. 127041.

5. V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment, 2008 (2008), p. P10008.

6. J. Cao, J. Fang, Z. Meng, and S. Liang, Knowledge Graph Embedding: A Survey from the Perspective of Representation Spaces, ACM Comput. Surv., 56 (2024), pp. 159:1–159:42.

7. P. Chunaev, Community detection in node-attributed social networks: a survey, Computer Science Review, 37 (2020), p. 100286.

8. P. Erdös and A. Rényi, On random graphs i, Publicationes Mathematicae Debrecen, 6 (1959), pp. 290–297.

9. P. Erdos and A. Renyi, On the evolution of random graphs, Publ. Math. Inst. Hungary. Acad. Sci., 5 (1960), pp. 17–61.

10. B. Gaume, I. Achitouv, and D. Chavalarias, Two antagonistic objectives for one multi-scale graph clustering framework, Nature Scientific Reports, 15 (2025), p. 13368.

11. C. He, X. Fei, Q. Cheng, H. Li, Z. Hu, and Y. Tang, A survey of community detection in complex networks using nonnegative matrix factorization, IEEE Transactions on Computational Social Systems, 9 (2021), pp. 440–457.

12. A. Lancichinetti, S. Fortunato, and F. Radicchi, Benchmark graphs for testing community detection algorithms, Physical Review E, 78 (2008), pp. 046110+.

13. J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg, Signed networks in social media, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, (2010).

14. J. Leskovec and A. Krevl, SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

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

16. P. Ronhovde and Z. Nussinov, Multiresolution community detection for megascale networks by information-based replica correlations, Physical Review E, 80 (2009).

17. X. Su, S. Xue, F. Liu, J. Wu, J. Yang, C. Zhou, W. Hu, C. Paris, S. Nepal, D. Jin, et al., A comprehensive survey on community detection with deep learning, IEEE Transactions on Neural Networks and Learning Systems, (2022).

18. C. Wang, S. Pan, P. Y. Celina, R. Hu, G. Long, and C. Zhang, Deep neighbor-aware embedding for node clustering in attributed graphs, Pattern Recognition, 122 (2022), p. 108230.

19. H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich, Local higher-order graph clustering, Proceedings of Conference on Knowledge Discovery and Data Mining, (2017), p. 555–564.