Benchmark graphs for the evaluation of Clustering Algorithms

Technical solution

***Abstract*— Artificial graphs are commonly used for the evaluation of community mining and clustering algorithms. Each artificial graph is assigned a pre-specified clustering, which is compared to clustering solutions obtained by the algorithms under evaluation. Hence, the pre-specified clustering should comply with specifications that are assumed to delimit a good clustering. However, existing construction processes for artificial graphs do not set explicit specifications for the pre-specified clustering. We call these graphs, randomly clustered graphs. Here, we introduce a new class of benchmark graphs which are clustered according to explicit specifications. We call them optimally clustered graphs. We present the basic properties of optimally clustered graphs and propose algorithms for their construction. Experimentally, we compare two community mining algorithms using both randomly and optimally clustered graphs. Results of this evaluation reveal interesting insights both for the algorithms and the artificial graphs.**

Graph clustering; Community structure; Artificial graph; Modularity; Intra linkage ratio

I. INTRODUCTION

A common problem related to the evaluation of clustering and community mining algorithms is the lack of datasets with precisely known structure. Typically, the evaluation of these algorithms is based on a clustering proposed by a human reviewer and compared with clustering solutions produced by the algorithms under evaluation [1]. However, the clustering problem is NP-complete [2]; thus, human reviewers suggestions are rather based on intuition than on an exhaustive examination of the relations in the input data set.

Lately, artificial graphs have been introduced [2,3,4,5,6] to facilitate the evaluation of clustering algorithms. Each artificial graph is assigned a pre-specified clustering, which we call the reference clustering. Solutions produced by clustering algorithms are evaluated in comparison to the reference clustering. Thus, the reference clustering should comply with specifications that are assumed to delimit a good clustering. However, the abovementioned approaches do not set explicit specifications for the reference clustering. In contrast, they assume that a random graph clustering is a good one, if edges have been distributed with higher probability for intra-cluster than inter-cluster ones. Thus, they construct artificial graphs by first creating a set of randomly clustered vertices and then adding edges with a probability, z_{in}, for intra-cluster edges and a probability, $z_{out}<z_{in}$, for inter-cluster ones. We call graphs constructed by such a process, randomly clustered graphs.

However, this implicit specification of a good clustering does not always hold true. Consider the set of vertices in figure 1.a and the reference clustering $\{\{0,1,2,3\},\{4,5,6\}\}$. Edges of graph in figure 1.b have been distributed with probability $7/10$ for intra-cluster edges and $3/10$ for inter-cluster ones. Vertex 3, in this graph, is more strongly connected to cluster $\{4,5,6\}$ than to its cluster. Therefore, we do not consider that the reference clustering is a good one. Clustering solutions, which correctly assign vertex 3 in the same cluster with vertices $4,5,6$ are underestimated when compared to the reference clustering.

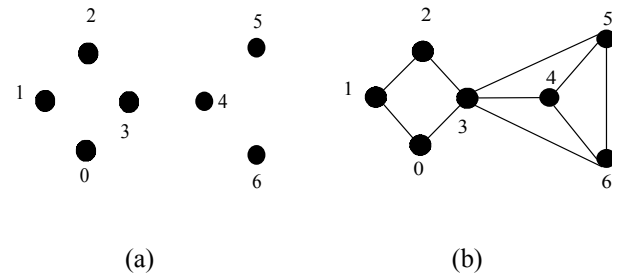

Figure 1. Construction of a graph in two phases:. (a) A set of vertices together with the reference clustering {{0,1,2,3},{4,5,6}} are created. (b) After the distribution of edges, the reference clustering is not considered a good one.

We urge that the reference clustering of artificial graphs should satisfy specifications that are assumed to delimit a good clustering. Besides, the notion of a good clustering is essentially related to community mining and clustering algorithms. Implicitly or explicitly, most of these algorithms adopt assumptions regarding the characteristics of a good clustering. Thus, it is a good practice to evaluate an algorithm using a reference clustering that complies with the algorithm assumptions about the good clustering. Therefore, we introduce optimally clustered graphs that imply a reference clustering of explicit specifications. We propose two sets of specifications, each corresponding to a type of clustering, which many clustering and community mining algorithms seek for. In the benefits of our approach, we include the following.

- Randomly clustered graphs are only constructed for $z_{out}/z_{in}<1$. This limitation is not always necessary for optimally clustered graphs. Besides, there are real clustering tasks where clusters or communities are structured such that $z_{out}/z_{in}>1$ [7, 8].
- The reference clustering of a randomly clustered graph may contain clusters that correspond to non-connected subgraphs. We remind that a graph is called connected if there is a path connecting any two of its vertices. As we demonstrate in section II, a reference clustering containing non-connected clusters is not considered a good one. Optimally clustered graphs do not contain non-connected clusters.
- Existing construction processes do not take into consideration loops, i.e. edges having both their endpoints in the same vertex. We show that loops may introduce ambiguity regarding the quality of the

reference clustering. Optimally clustered graphs do not contain loops.

Concerning the experimentation, existing approaches implicitly assume that the difficulty of a clustering task primarily depends on the ratio z_{out}/z_{in}. The higher this ratio, the more difficult the clustering task. Thus, they typically evaluate clustering or community mining algorithms using a set of randomly clustered graphs constructed for various values of the ratio z_{out}/z_{in}. Here, we introduce the notion of internal density, which is a measure for the intra-cluster links density. We proceed to an experimentation using both randomly and optimally clustered graphs of various values of the z_{out}/z_{in} ratio and various values of the internal density. We compare a well known community mining algorithm [3] with a new community mining algorithm [8]. Results of this experimentation show that

- Optimally clustered graphs are essential for the reliable evaluation of clustering and community mining algorithms
- The difficulty of a clustering task is primarily depends on the internal density, therefore the evaluation of algorithms should include graphs of scaled internal densities.

The remaining of the paper is structured as follows: In section II, we define optimally clustered graphs. Section III presents their construction process. Section IV demonstrates their utilization for the evaluation of two clustering algorithms. Finally, section V reports conclusions and highlights further research topics.

II. OPTIMALLY CLUSTERED GRAPHS

A. Basic Definitions and Notations

Throughout this paper we assume that $G=(V,L)$ is an undirected graph, where V is the set of vertices and L is the set of links such that $L \subseteq V \times V$. Vertices u,v are called the endpoints of link $\{u,v\}$. The degree of vertex v, denoted by $d(v)$, equals to the number of links having v as one of their endpoints. Moreover, we denote by $C=\{c_1,\ldots,c_k\}$, a clustering of G vertices into k clusters with $\cup c \in C = V$ and $c_i \cap c_{j,i \neq j} = \varnothing$. The size of cluster c is denoted by $|c|$. We call a pair (G,C), a clustered graph. Clustering C in (G,C) is called reference clustering. The following definitions and notations apply on a clustered graph. Graph $g(c)$ denotes a subgraph of G induced on cluster c, i.e. $g(c)=\{c,\{v,u\} \in L : v,u \in c\}$. The cluster at which vertex v belongs to is denoted by $c(v)$. The set $L_C \subseteq L$ denotes the set of inter-cluster links of clustered graph (G,C). The number of links having both their endpoints in cluster c is denoted by I_c and the number of links having only one of their endpoints in cluster c is denoted by X_c.

Definition 1: The linkage between a cluster, c, and a vertex v, is denoted by $l(c,v)$ and is equal to the number of links having one of their endpoint in c and the other in v.

Definition 2: The transitional linkage ratio of clustered graph (G,C), $tlr(G,C)$ is the number of inter-cluster links divided by the number of intra-cluster links.

$$tlr(G,C)=|L_C|/|L-L_C|$$

Note that transitional linkage ratio is equivalent to z_{out}/z_{in} ratio. Henceforth, instead of referring to z_{out}/z_{in} ratio, we refer to the transitional linkage ratio.

As we have already mentioned, a graph is called connected if there is a path between any two of its vertices. In addition, a graph is called completely connected if there is a link between any two of its vertices.

Definition 3: The internal linkage density of vertex v, $ild(v)$, is equal to the linkage between v and $c(v)$ divided by the size of $c(v)$ minus 1:

$$ild(v)=l(c(v),v)/(|c(v)|-1).$$

Note, that the internal linkage density lies in the range $0..1$. If a vertex is not connected to its cluster, then its internal linkage density equals to 0. If a vertex is linked to all other vertices in its cluster then its internal linkage density equals to 1.

Definition 4: The internal density of a clustered graph, (G, C), denoted by $id(G,C)$ is the average internal linkage density over all of its vertices.

Note that the internal density $id(g(c))$ of a completely connected cluster c equals to 1. In addition, the internal density of a graph clustered such that all of its clusters are completely connected is also equal to 1.

B. Two classes of optimally clustered graphs

What are the specifications that a reference clustering should comply with? The answer to this question is related to the type of clustering that clustering algorithms are assumed to seek for. Many graph clustering [4, 9-13, 39] and community mining algorithms [13,2,3,15,16] seek for a type of clustering, which we call traditional and define next.

Definition 5: A clustering C is called traditional if the following condition is satisfied

$$l(c(v),v) > l(V-c(v),v) \ \forall v \in V \qquad \text{(S1)}$$

Condition S1 requires that the linkage between a vertex and its cluster is higher than the linkage between the vertex and the remaining graph.

In [7,8], we propose two graph clustering algorithms, which seek for a type of clustering that we call detailed and define next.

Definition 6: A clustering C is called detailed if the following condition is satisfied.

$$l(c(v),v) > l(c,v) \ \forall v \in V, c \in C-c(v) \qquad \text{(S2)}$$

Condition S2 requires that the linkage between a vertex and its cluster is higher than the linkage between the vertex and any other cluster. Condition S2 corresponds to a more detailed

clustering than condition S1. We demonstrate the difference between these two types of clustering with the following example. Consider the graph in figure 2. Clustering $\{\{0,1,2\},\{3,4,5\},\{6,7,8\}\}$ is a detailed but not a traditional one. Condition S2 holds true for all vertices, whereas condition S1 is violated by vertex *2*, as $l(\{0,1,2\},2)=2=l(\{3,4,5,6,7,8\},2)$.

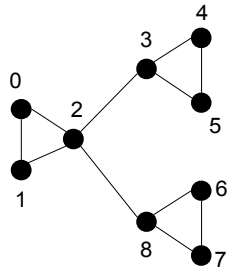

Figure 2. Clustering {{0,1,2},{3,4,5},{6,7,8}} is a detailed but not a traditional one.

It is trivial to prove that if a clustering satisfies condition S1, it also satisfies condition S2.

Definition 7: A clustered graph (G,C) is called optimally clustered if C is either traditional or detailed and the following conditions are satisfied.

$$\{v,v\} \notin L \ \forall v \in V \qquad \text{(S3)}$$

$$g(c) \text{ is connected } \forall c \in C \qquad \text{(S4)}$$

Condition S3 requires that graph G does not contain loops. The existence of loops in G actually violates the purpose of conditions S1 and S2. Consider the graph in figure 3. Clustering $\{\{0,1,2,3\},\{4,5,6\}\}$ satisfies both conditions S1 and S2 and we consider it a good clustering. Due to the existence of two $\{3,3\}$ loops, clustering $\{\{0,1,2\},\{3,4,5,6\}\}$ also satisfies both conditions S1 and S2. However, vertex *3* is more strongly connected to cluster $\{0,1,2\}$ than to cluster {4,5,6}. Therefore, we do not consider that clustering $\{\{0,1,2\},\{3,4,5,6\}\}$ is a good one.

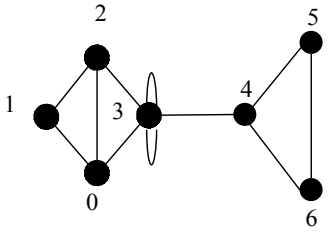

Figure 3. Although clustering {{0,1,2},{3,4,5,6}} satisfies both conditions S1 and S2 is not considered a good clustering.

Condition S4 requires that all clusters in C correspond to connected subgraphs of G. Clusters, which are not connected violate the purpose of conditions S1 and S2. Consider the graph in figure 4. Clustering $\{\{0,1,2\},\{3,4,5,6,7,8\},\{9,10,11\}\}$ satisfies both conditions S1 and S2. However, it is not a good clustering as cluster $\{3,4,5,6,7,8\}$ consists of two disconnected components. For this graph, most clustering algorithms will produce (correctly) the clustering $\{\{0,1,2,3,4,5\},\{6,7,8,9,10,11\}\}$, which is underestimated in comparison with clustering $\{\{0,1,2\},\{3,4,5,6,7,8\},\{9,10,11\}\}$.

For convenience, we call an optimally clustered graph (G,C) traditionally clustered if C satisfies condition S1 and detailed clustered if C satisfies condition S2 but not S1.

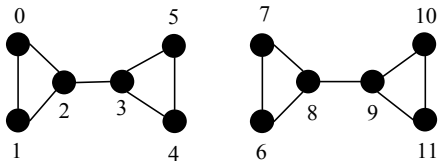

Figure 4. Clustering {{0,1,2},{3,4,5,6,7,8}, {9,10,11}} satisfies both conditions S1 and S2 but it is not considered a good clustering because cluster {3,4,5,6,7,8} consists of two disconnected components

III. CONSTRUCTION ALGORITHM

Experimentation with artificial graphs should include graphs of various levels of transitional linkage ratio and internal density. Therefore, we need a construction process that allows us to adjust the internal density and the transitional linkage ratio of the derived graphs. To achieve accurate adjustment of these factors between multiple graph instances, we introduce the notion of a base graph.

Definition 8: A clustered graph (G,C) is called base if it is optimally clustered and the set of its inter-cluster links is empty, $L_C=\varnothing$.

Note that a graph with empty the set of its inter-cluster links is optimally clustered if it does not contain loops and non-connected clusters. Optimally clustered graphs can be constructed for various levels of transitional linkage ratio by adding inter-cluster links in a base graph, preserving either condition S1 or S2. Note that the internal density of a graph is not altered by the addition of inter-cluster links. Thus, all graphs constructed on the same base graph have common internal density. Therefore, we propose a construction process consisting of two phases: In the first phase, we construct a base graph for a required internal density. In the second phase, we construct an optimally clustered graph by adding inter-cluster links in a base graph. Thus, using the same base graph, we can construct multiple optimally clustered graphs for various values of transitional linkage ratio, all having the same internal density. However, it should be noted that there are limitations on the internal density of a base graph.

Proposition 1: The internal density of a base graph (G,C) is higher or at least equal to $1/(|c_{min}|-1)$, where $c_{min} \in C$ denotes a cluster of minimum size.

Proof : Since all clusters are required to be connected, each vertex should be linked to its cluster with at least one link. Therefore, the internal linkage density of a vertex, v, should be higher or at least equal to $1/(|c(v)|-1)$. This value depends only on the cluster size and is common for all vertices in a

cluster. Since, $1/(|c_{min}|-1) \geq 1/(|c|-1)\ \forall c \in C$, it follows that the internal density of each vertex should be higher or equal to $1/(|c_{min}|-1)$. Thus, $ild(v) \geq 1/(|c_{min}|-1)\ \forall v \in V$. Summing up for all vertices of G and dividing by the size of G, we have $id(G) \geq 1/(|c_{min}|-1)$.

A. Construction of a base graph

A straightforward approach for the construction of a base graph is to create clusters, i.e. set of vertices and then insert randomly selected intra-cluster links checking that the internal linkage density of both the endpoints of a candidate link are within a required limit. However, such a construction process often results in graphs which contain non-connected clusters. We present this construction process in figure 5.

Algorithm 1

Input: The number of clusters m, the minimum and the maximum number of vertices per cluster, v_{min} and v_{max} respectively, and a real number $r \in [0,1]$ representing a required internal density.

Output: A base graph (G, C)

1. $\forall i \in 1..m$
2. Generate randomly integer $v_a \in [v_{min}, v_{max}]$.
   Create v_a vertices. Assign them to cluster c_i
   $V = V \cup ci; C = C \cup ci$
3. $l = \{\{u,v\} : u,v \in c_i$ and $u \neq v\}$.Permutate l.
4. $\forall \{u,v\} \in l$
5. If $ild(u) < r$ and $ild(v) < r$
   $L = L \cup \{u,v\}$
6. If $g(ci)$ is not connected return null
7. Return graph $((V,L),C)$

Figure 5. An algorithm for the construction of base graphs

Due to the random nature of algorithm 1, a created cluster may be non-connected. For example, consider the construction of a cluster consisting of 6 vertices with required internal density equal to 0.4. A possible outcome is displayed in figure 6.

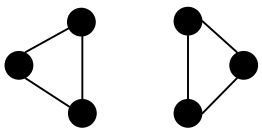

Figure 6. A non-connected cluster derived by algorithm 1

To overcome this limitation, we follow a reverse process. We first create all clusters as completely connected components and then we delete links down to a required internal density preserving that the deletion of a link does not produce a non-connected cluster. Figure 7 displays the proposed construction process

Algorithm 2

Input: The number of clusters m, the minimum and the maximum number of vertices per cluster, v_{min} and v_{max} respectively, and a real number $r \in [0,1]$ representing a required internal density.

Output: A base graph (G, C)

1. $\forall i \in 1..m$
2. Generate randomly integer $v_a \in [v_{min}, v_{max}]$.
3. Create v_a vertices. Assign them to cluster c_i
4. $V = V \cup c_i; C = C \cup c_i$
5. $L = L \cup \{\{u,v\} : u,v \in c_i$ and $u \neq v\}$;
6. Permutate L
7. $\{u,v\} = L$.getFirstElement;
8. while $\{u,v\} \neq null$ and $id((V,L)) > r$
9. If $c(v)$ will not become non-connected $L = L - \{u,v\}$
10. $\{u,v\} = L$.getNextElement
11. return $((V,L),C)$

Figure 7. The proposed algorithm for the construction of base graphs

For each requested cluster (line 1), we generate randomly the cluster size (line 2), then we create the cluster vertices (line 3), we update the sets V and C (line 4) and we insert links in L for each pair of vertices of the created cluster (line 5). Thus, each cluster is created as a completely connected component. Therefore, when the loop (line 1) exits a base graph $((V,L),C)$ has been constructed with $g(c)$ being a completely connected subgraph of G for each $c \in C$. Note that since all clusters are completely connected, the internal density of the constructed graph is maximum, $id((V,L)) = 1$.

In line 6, the set of links is randomly permutated. Next, a link is deleted while the actual internal density is higher than the required one (line 8), if its deletion does not result to a non-connected cluster (line 9). This process constructs base graphs with internal density nearly equal to the required one. However, we remind that a base graph can not be constructed for all values of the input parameters. According to proposition 1, a base graph can not be constructed for $r < 1/(|c_{min}|-1)$. For example, a base graph can not be constructed for $c_{min} = 4$, $r < 1/3$. If the value of parameter r is lower than $1/(|c_{min}|-1)$ then algorithm 2 produces a base graph of the minimum possible internal density, whereas algorithm 1 fails to return a base graph.

In the following, we describe a base graph by the number of its clusters, the average cluster size, and the standard distribution of cluster size, the internal density and the corresponding standard deviation. Figure 8 displays a base graph with 5 clusters, average cluster size equal to 4, standard deviation equal to 0.7, internal density equal to 1 and standard deviation equal to 0.

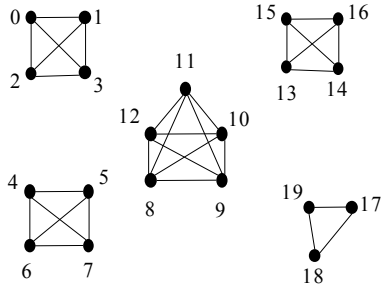

Figure 8. A base graph with 5 clusters, average cluster size equal to 4, standard deviation equal to 0.7, internal density equal to 1 and standard deviation equal to 0.

B. Construction of optimally clustered graphs

Phase II produces an optimally clustered graph by adding inter-cluster links to an input base graph preserving condition S1 or S2. Figure 9 presents an algorithm for the construction of an optimally clustered graph.

Algorithm 3

Input: A base graph $((V,L'),C)$, a condition $S \in \{S1,S2\}$ and the real number $r_{out} > 0$.

Output: An optimally clustered graph $((V,L),C)$ with $L \supseteq L'$

1. $l = \{\{u,v\} : c(u) \neq c(v)\}$. Permutate l.
2. $\forall \{u,v\} \in l$
3. If $tlr((V,L),C) \geq r_{out}$ go to step 5
4. If S is not violated $L = L \cup \{u,v\}$
5. return ((V,L),C)

Figure 9. Algorithm for the construction of an optimally clustered graph

Note that r_{out} sets an upper bound for the transitional linkage ratio. Since inter-cluster links are inserted in L only if a required condition is preserved, the produced graph is optimally clustered. Moreover, clustering C is not altered by algorithm 3. Therefore, all graphs produced on the same base graph have as common reference clustering the reference clustering of the base graph.

The description of an optimally clustered graph includes the description of its base graph and the transitional linkage ratio. Figure 10 presents an optimally clustered graph constructed on the base graph of figure 7 with transitional linkage ratio equal to 0.38.

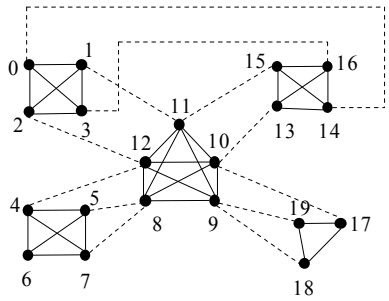

Figure 10. An optimally clustered graph with transitional linkage equal to 0.38

Note that condition S1 implies that the transitional linkage ratio of a traditionally clustered graph is lower than 1. Detailed clustered graphs do not have this limitation. Figure 11 demonstrates a detailed clustered graph with transitional linkage equal to 1.5.

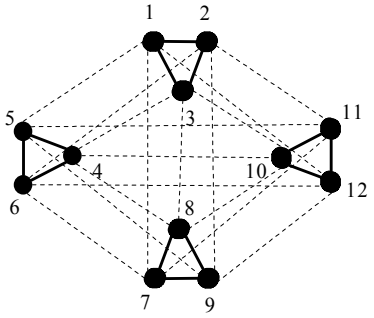

Figure 11. A detailed clustered graph with average external linkage equal to 1.5

Finally, note that randomly clustered graphs may also be described by means of a base graph and the transitional linkage ratio.

IV. EXPERIMENTATION

In this experimentation, we compare the agglomerative optimization of modularity [3], or Mod for short, with the agglomerative optimization of Intra Linkage Ratio [8], or ILR for short.

Both algorithms are hierarchical agglomerative optimizations. In each step of agglomeration, Mod merges together these two clusters that result in the greatest increase or smallest decrease of modularity criterion function Q given by $Q = \sum_i (e_{ii} - a_i^2)$, where $e_{ij} = \frac{l(c_i,c_j)}{|L|}$ *if* $i \neq j$, $e_{ii} = \frac{I_{c_i}}{|L|}$ and $a_i = \sum_j e_{ij}$. Mod assumes that communities are vertex subsets within which vertex to vertex connections are dense but between which vertex to vertex connections are sparser. This delimitation of a community apparently regards all vertices in a community and corresponds to a traditional or a detailed clustering.

In each step of agglomeration, ILR merges together these two clusters that result in the greatest increase or smallest decrease of ILR criterion function given by $\sum_{c \in C} \frac{I_c}{I_c + X_c}$. ILR also seeks for a traditional or a detailed clustering.

A. Data Sets

For this experimentation, we use a total of 360 both optimally and randomly clustered graphs of various internal densities and transitional linkage ratios. Half of these graphs are optimally clustered and half are randomly clustered. We identify the subset of optimally clustered graphs with letter O, and the subset of randomly clustered graphs with letter R. The set of optimally clustered graphs is further divided in 3 subsets, the LO, MO and HO dataset. Graphs in each of these datasets have been constructed on the same base graph. The base graph of the LO dataset has low internal density; the base graph of the MO dataset has medium internal density and the base graph of the HO dataset has high internal density. The description of each base graph is given in table 1.

TABLE I. DESCRIPTION OF BASE GRAPHS

Dataset	Number Of Clusters	Cluster Size		Internal Density	
		Average	*Standard Deviation*	*Average*	*Standard Deviation*
LO	10	8.5	2.64	0.30	0.16
MO	10	9.5	4.30	0.60	0.16
HO	10	9.3	3.06	0.90	0.13

Each of datasets LO, MO and HO is further divided into 6 subsets, recognized by identifications d1, d2, d3, d4, d5 and d6. Thus, we have a total of 18 subsets, the LO.d1-LO.d6, MO.d1-MO.d6 and HO.d1-HO.d6 datasets. We call each of these subsets, a uniform dataset. Each of the uniform datasets contains 10 graphs. Thus, we have a total of 180 graphs. Graphs in a uniform dataset have common transitional linkage ratio but the transitional linkage ratio is scaled in successive uniform datasets, i.e. d1 uniform dataset contains graphs with low transitional linkage ratio, d2 dataset contains graphs with higher transitional linkage ratio, etc. Table II, displays the transitional linkage ratio for datasets containing optimally clustered graphs. We remark that graphs in datasets d1-d3 have been constructed such that they satisfy condition S1, whereas graphs in datasets d4-d6 have been constructed such that they satisfy condition S2.

TABLE II. TRANSITIONAL LINKAGE RATIO FOR O DATASET

Datasets	d1	d2	d3	d4	d5	d6
LO	0.51	0.57	0.57	1.10	1.30	1.51
MO	0.50	0.70	0.83	1.10	1.30	1.50
HO	0.50	0.70	0.86	1.10	1.30	1.50

Numbers in this table represent averages for 10 graphs in a uniform dataset

The set of randomly clustered graphs is structured identically to the set of optimally clustered graphs. Therefore, it contains subsets LR.d1-LR.d6, MR.d1-MR.d6 and HR.d1-HR.d6. Each randomly clustered graph has been constructed such that its description is equal to the description of a corresponding optimally clustered graph, i.e. graphs in LR.d1 have the same description with graphs in LO.d1, graphs in LR.d2 have the same transitional linkage ratio with graphs in LO.d2, etc. Thus, the description of randomly clustered graphs is also given by tables I and II.

B. Results

For the evaluation of clustering solutions, we employ the similarity measure [27], μ_σ, between two clustering C and S given by

$$\mu_\sigma(C,S) = \frac{\dfrac{\sum_{c\in C}\max_{s\in S}\dfrac{|c\cap s|}{|c\cup s|}}{|C|} + \dfrac{\sum_{s\in S}\max_{c\in C}\dfrac{|s\cap c|}{|s\cup c|}}{|S|}}{2}$$

Similarity measure, μ_σ, compares a clustering solution to a reference clustering. Its maximum value is 1 and implies that C is equal to S. The larger the value of similarity measure, the better the clustering solution is. From the hierarchy derived by each algorithm, we test the solution for 10 clusters, which is the size of the reference clustering for all graphs (table I). In the following, all values of similarity measure are averages for 10 graphs in a uniform dataset.

Next, we first present results and then discuss them.

Fig. 12 presents a comparison of ILR and Mod performance on low internal density datasets containing optimally clustered graphs.

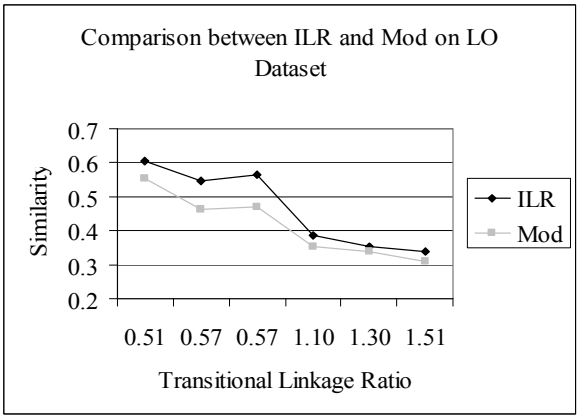

Figure 12. Comparison of ILR and Mod performance on low internal density datasets containing optimally clustered graphs (LO dataset)

Fig. 13 presents a comparison of ILR and Mod performance on medium internal density datasets containing optimally clustered graphs.

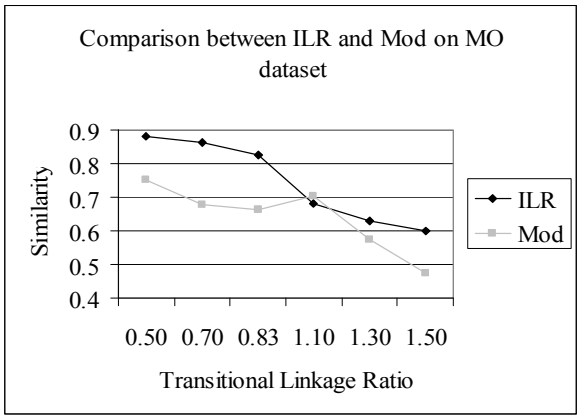

Figure 13. Comparison of ILR and Mod performance on medium internal density datasets containing optimally clustered graphs (MO dataset)

Fig. 14 presents a comparison of ILR and Mod performance on high internal density datasets containing optimally clustered graphs (HO dataset).

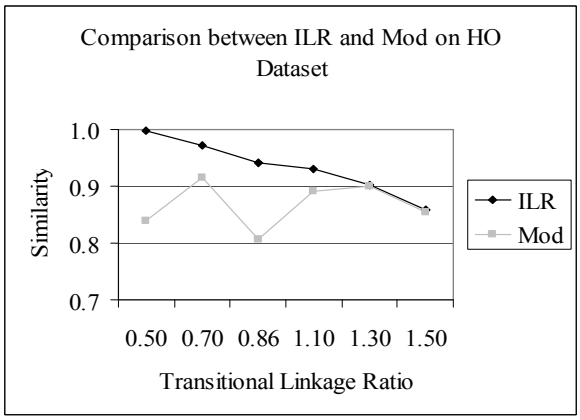

Figure 14. Comparison of ILR and Mod performance on high internal density datasets containing optimally clustered graphs (HO dataset)

Fig. 15 presents a comparison of ILR and Mod performance on low internal density datasets containing randomly clustered graphs.

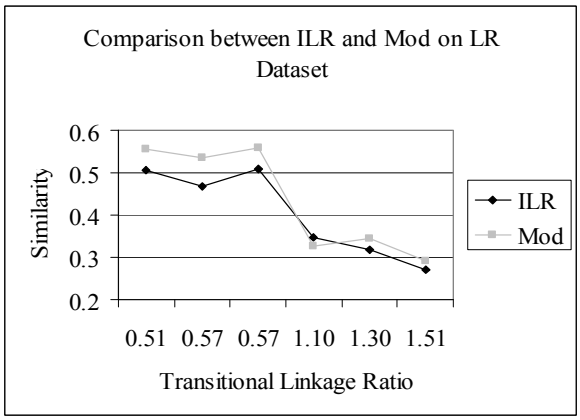

Figure 15. Comparison of ILR and Mod performance on low internal density datasets containing randomly clustered graphs (LR dataset)

Fig. 16 presents a comparison of ILR and Mod performance on medium internal density datasets containing randomly clustered graphs.

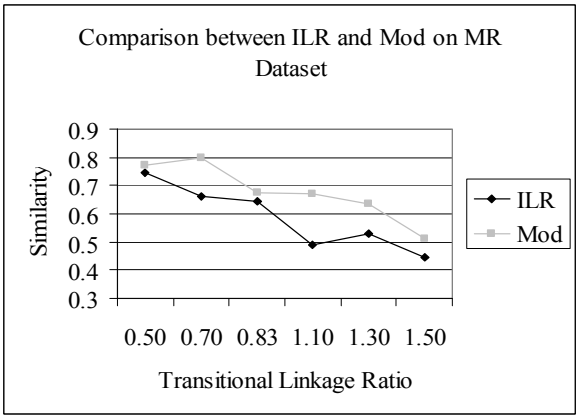

Figure 16. Comparison of ILR and Mod performance on medium internal density datasets containing randomly clustered graphs (MR dataset)

Fig. 17 presents a comparison of ILR and Mod performance on high internal density datasets containing randomly clustered graphs.

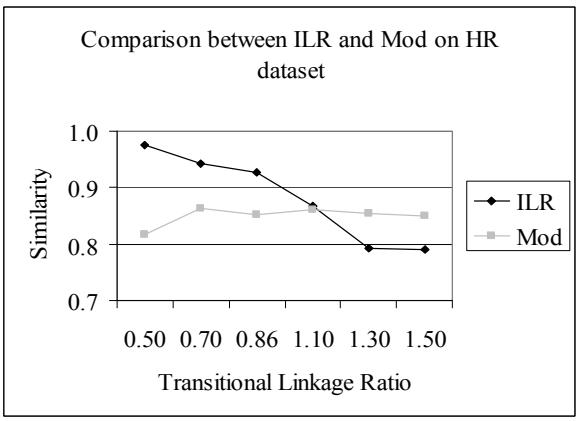

Figure 17. Comparison of ILR and Mod performance on high internal density datasets containing randomly clustered graphs (HR dataset)

Fig. 18 shows the variance of ILR performance between optimally and randomly clustered graphs of medium internal density.

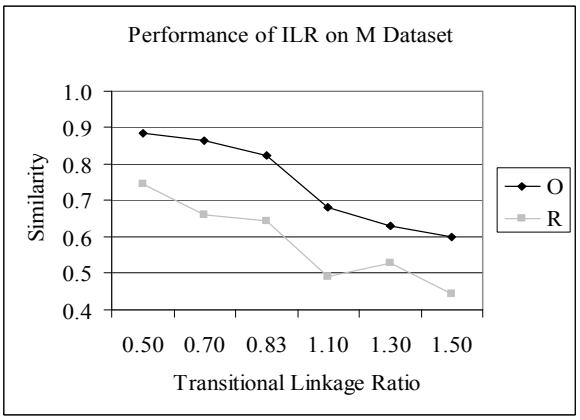

Figure 18. Variance of ILR performance between optimally and randomly clustered graphs of medium internal density (M dataset)

Fig. 19 shows the variance of Mod performance between optimally and randomly clustered graphs of medium internal density.

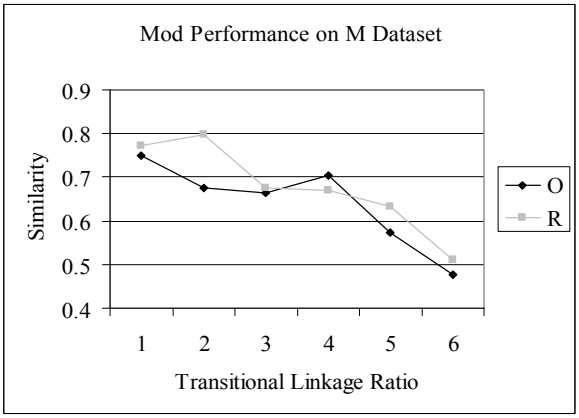

Figure 19. Variance of Mod performance between optimally and randomly clustered graphs of medium internal density (M dataset)

Fig. 20 presents the variation of ILR performance on optimally clustered graphs of various levels of internal density.

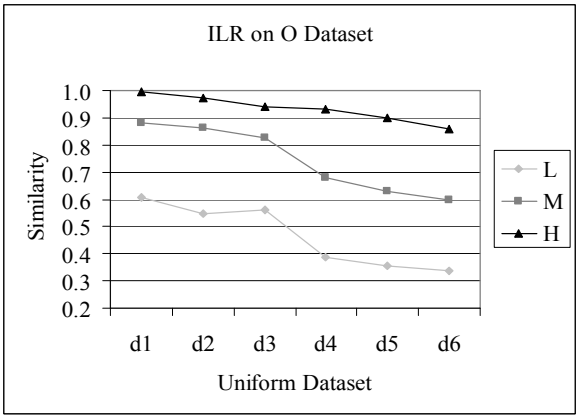

Figure 20. Variation of ILR performance between optimally clustered graphs of various levels of internal density (O Dataset).

Fig. 21 presents the variation of Mod performance on optimally clustered graphs of various levels of internal density.

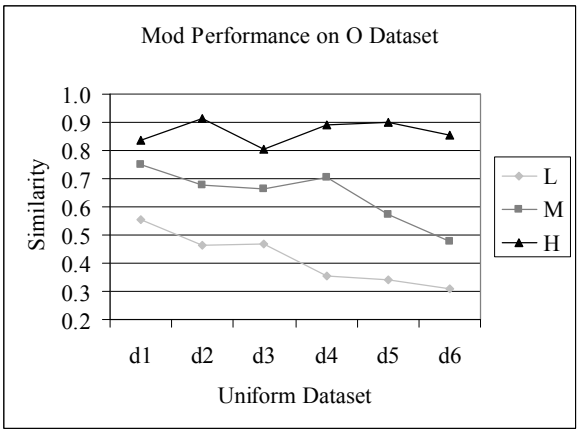

Figure 21. Variation of Mod performance between optimally clustered graphs of various levels of internal density (O Dataset).

C. *Discussion of the results*

We summarize the results as follows:

- Performance between graphs of different internal density. As it is demonstrated in Fig. 20 and 21, both algorithms perform better on optimally clustered graphs of high than low internal density. This result also appears by comparing Fig. 12, 13 and 14. An analogous result is obtained for randomly clustered graphs by comparing Fig. 15, 16 and 17. It is remarkable that both algorithms perform better on graphs of high internal density and high transitional linkage than on graphs of low internal density and low transitional linkage. For example, in Fig. 20 the transitional linkage of dataset HO.d6 is 1.51, whereas the average external linkage for dataset LO.d1 is 0.51 (table II). However, ILR performance is better for HO.6 than for LO.1. Similarly, Mod performs better on HO.d6 than on LO.d1. Thus, if clusters have high internal densities algorithms give good solutions even if clusters are densely interconnected.

- Performance between graphs with scaled transitional linkage ratio. In all figures, we can see that for constant internal density, both algorithm performances are decreasing while the transitional linkage ratio is increasing. An exception to this finding is the performance of Mod for high internal density graphs. As it is shown in Fig. 14 and 17, Mod performance is not decreasing while the transitional linkage is increasing. Modularity clustering is based on the link betweenness, which is a measure that favors inter-cluster links and disfavors intra-cluster links. Thus, modularity performance is favored by an increased number of inter-cluster links. On high internal density graphs, where algorithms perform better in comparison with lower internal density graphs, this favor is clearer.

- Performance between optimally and randomly clustered graphs: As it is demonstrated in Fig. 18, ILR performs better on optimally than randomly clustered graphs of medium internal density. This result also

appears by comparing Fig. 13 with Fig. 16. An analogous result appears for low and high internal density graphs by comparing Fig. 12 with Fig. 15 and Fig. 14 with Fig. 17. Therefore, ILR is underestimated when evaluated with randomly clustered graphs. On the other hand, Mod performs rather better on randomly than optimally clustered graphs of medium internal density (Fig. 19). An analogous result appears for low and high internal density graphs by comparing Fig. 12 with Fig. 15 and Fig. 14 with Fig. 17. Modularity function is based on the assumption that links of a graph $G=(V,L)$ are normally distributed in $V \times V$. If intra-cluster links are distributed with equal probability with inter-cluster links, then modularity value is equal to 0 [3]. If intra-cluster links are distributed with a higher probability from inter-cluster links, then the graph has a modularity value higher than 0. Thus, Modularity function rather follows the model of randomly than optimally clustered graphs.

- Comparison between Mod and ILR: In low and medium internal density randomly clustered graphs, Mod outperforms ILR (Fig. 15, 16). In test with high internal density and randomly clustered graphs, ILR is better for graphs of lower transitional ratio and Mod is better for graphs with higher transitional ratio (Fig.17). However, ILR outperforms Mod in all tests with optimally clustered graphs (Fig. 12, 13 and 14). Thus, the result of the comparative evaluation is diversified depending on whether optimally or randomly clustered graphs are used. Since both algorithms seej for a traditional or a detailed clustering, optimally clustered graphs should be included in the experimentation dataset used for a comparative evaluation of these algorithms.

V. CONCLUSIONS

In this paper, we introduce the optimally clustered graphs for the evaluation of clustering and community mining algorithms. We have compared two community mining algorithms by conducting a series of experiments that shows that algorithms performance is diversified between optimally and randomly clustered graphs. Optimally clustered graphs encapsulate either a traditional or a detailed clustering. Therefore, algorithms that are assumed to seek for a traditional or a detailed clustering, as well as algorithms that assume that the community structure of a graph corresponds to such a clustering are more reliable evaluated using optimally clustered graphs. It is remarkable that the difficulty of a clustering task primarily depends on the internal density than on the transitional linkage of a graph. Further research includes the construction of optimally clustered such as the vertex degree and the cluster size distributions are power laws.

REFERENCES

[1] G. Karypis and V. Kumar, “A fast and high quality multilevel scheme for partitioning irregular graphs”, SIAM Journal on Scientific Computing, vol. 20, issue 1, 1998, pp. 359-392.

[2] M. Newman and M. Girvan, “Finding and evaluating community structure in networks”, Physical Review E, vol. 69, 2004, No. 026113.

[3] M. Newman, “Fast algorithm for detecting community structure in networks”, Physical Review E, vol. 69, 2004, No. 066133,

[4] U. Brandes, M. Gaertler and D. Wagner, “Experiments on Graph Clustering Algorithms”, Algorithms - ESA 2003, Lecture Notes in Computer Science, Springer Berlin, Heidelberg, pp. 568-579, 2003.

[5] U. Brandes, M. Gaertler and D. Wagner, “Engineering graph clustering: Models and experimental evaluation”, Journal of Experimental Algorithmics (JEA), ACM New York, NY, USA, vol. 12,2008, No. 1.1.

[6] A. Lancichinetti, S. Fortunato and F. Radicchi, “Benchmark graphs for testing community detection algorithms”, Physical Review E, vol. 78, 2008, no. 4.

[7] L. Moussiades and A. Vakali, “Clustering dense graphs: a web site graph paradighm”, unpublished

[8] L. Moussiades and A. Vakali, “Mining the Community Structure of a Web Site”, Technical Report 2008.

[9] M. Rattigan, M. Maier and D. Jensen, “Graph Clustering with Network Structure Indices”, Proceedings of the 24th international conference on Machine learning, ACM New York, NY, USA, pp. 783-790, 2007.

[10] G. Flake, R. Tarjan and K. Tsioutsiouliklis, “Graph Clustering and Minimum Cut Trees”, Internet Mathematics, vol. 1, issue 4, pp. 385-408, 2004.

[11] V. Kaburlassos, L. Moussiades and Athena Vakali, “Granular graph clustering in the Web”, Proceedings of the 8th International Conference on Natural Computing, Utah: World Scientific Publishing, pp. 1639-1645, 2007.

[12] V. Kaburlassos, L. Moussiades and A. Vakali, “Fuzzy Lattice Reasoning (FLR) Neural Computation for Weighted Graph Partitioning”, Neurocomputing, in press.

[13] B. Stein and O. Niggemann, “On the nature of structure and its identification”, Proceedings of the 25th international Workshop on Graph-theoretic Concepts in Computer Science, Springer-Verlag , London, pp. 122-134, 1999.

[14] M. Girvan and M. Newman, “Community Structure in Social and Biological Networks”, Proceedings of National Academy of Science, In Shepp, L. A., pp. 7821-7826, 2002.

[15] D. Wilkinson and B. Huberman, “A Method for finding communities of related genes”, Proceedings of National Academy of Science, pp. 5241–5248, 2004.

[16] H. Ino, M. Kudo and A. Nakamura, “A Comparative Study of Algorithms for Finding Web Communities”, Proceedings of the 21st International Conference on Data Engineering Workshops, IEEE Computer Society, Washington, USA, No. 1257, 2005.