2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining

Efficiently Clustering Very Large Attributed Graphs

Alessandro Baroni
University of Pisa, Italy
baroni@di.unipi.it

Alessio Conte
University of Pisa, Italy
conte@di.unipi.it

Maurizio Patrignani
Roma Tre University, Italy
patrigna@dia.uniroma3.it

Salvatore Ruggieri
University of Pisa and
ISTI-CNR, Italy
ruggieri@di.unipi.it

***Abstract*—Attributed graphs model real networks by enriching their nodes with attributes accounting for properties. Several techniques have been proposed for partitioning these graphs into clusters that are homogeneous with respect to both semantic attributes and to the structure of the graph. However, time and space complexities of state of the art algorithms limit their scalability to medium-sized graphs. We propose SToC (for *S*emantic-*T*opological *C*lustering), a fast and scalable algorithm for partitioning large attributed graphs. The approach is robust, being compatible both with categorical and with quantitative attributes, and it is tailorable, allowing the user to weight the semantic and topological components. Further, the approach does not require the user to guess in advance the number of clusters. SToC relies on well known approximation techniques such as bottom-k sketches, traditional graph-theoretic concepts, and a new perspective on the composition of heterogeneous distance measures. Experimental results demonstrate its ability to efficiently compute high-quality partitions of large scale attributed graphs.**

I. Introduction

Several approaches in the literature aim at partitioning a graph into communities that share some sets of properties (see [14] for a survey). Most criteria for defining communities in networks are based on topology and focus on specific features of the network structure, such as the presence of dense subgraphs or other edge-driven characteristics. However, real world graphs such as the World Wide Web and Social Networks are more than just their topology. A formal representation that is gaining popularity for describing such networks is the *attributed graph* [6], [16], [19], [38].

An attributed graph is a graph where each node is assigned values on a specified set of attributes. Attribute domains may be either categorical (e.g., sex) or quantitative (e.g., age). Clustering attributed graphs consists in partitioning them into disjoint communities of nodes that are both well connected and similar with respect to their attributes. State of the art algorithms for clustering attributed graphs have several limitations [38]:

ASONAM '17, July 31 - August 03, 2017, Sydney, Australia

ACM ISBN 978-1-4503-4993-2/17/07?/$15.00
http://dx.doi.org/10.1145/3110025.3110030

they are too slow to be compatible with big-data scenarios, both in terms of asymptotic time complexity and in terms of running times; they use data-structures that need to be completely rebuilt if the input graph changes; and they ask the user to specify the number of clusters to be produced. Moreover, they only work with categorical attributes, forcing the user to discretize the domains of quantitative attributes, leading to a loss of information in distance measures.

We offer a new perspective on the composition of heterogeneous distance measures. Based on this, we present a distance-based clustering algorithm for attributed graphs that allows the user to tune the relative importance of the semantic and structural information of the input data and only requires to specify as input qualitative parameters of the desired partition rather than quantitative ones (such as the number of clusters). This approach is so effective that can be used to directly produce a set of similar nodes that form a community with a specified input node without having to cluster the whole graph first. We rely on state-of-the-art approximation techniques, such as bottom-k sketches for approximating similarity between sets [10] and the Hoeffding bound (see [15]) to maintain high performance while keeping the precision under control. Regarding efficiency, our approach has an expected time complexity of $O(m \log n)$, where n and m are the number of nodes and edges in the graph, respectively. This performance is achieved via the adoption of a distance function that can be efficiently computed in sublinear time. Experimental results demonstrate the ability of our algorithm to produce high-quality partitions of large attributed graphs.

II. Related work

Graph clustering, also known as community discovery, is an active research area (see e.g., the survey [14]). While the overwhelming majority of the approaches assign nodes to clusters based on topological information only, recent works address the problem of clustering graphs with semantic information attached to nodes or edges (in this paper, we restrict to node-attributed graphs). In fact, topological-only or semantic-only clustering approaches are not as effective as approaches that exploit both sources of information [17]. A survey of the area [6] categorizes the following classes for node-attributed graphs (here we recall only key references and uncovered recent papers). *Reduction-based* approaches translate the semantic information into the structure of the graph (for example adding weights to its edges) or vice versa (for example encoding topological information into new attributes), then perform a traditional clustering on the obtained data. They are

efficient, but the quality of the clustering is poor. *Walk-based* algorithms augment the graph with dummy nodes representing categorical only attribute values and estimate node distance through a neighborhood random walk [38]. The more attributes two nodes shares the more paths connect them the more the nodes are considered close. A traditional clustering algorithm based on these distances produces the output. *Model-based* approaches statistically infer a model of the clustered attributed graphs, assuming they are generated accordingly to some parametric distribution. The **BAGC** algorithm [35] adopts a Bayesian approach to infer the parameters that best fit the input graph, but requires the number of clusters to be known in advance and does not handle quantitative attributes. The approach has been recently generalized to weighted attributed graphs in [36]. The resulting **GBAGC** algorithm is the best performing of the state-of-the-art (in Section VIII we will mainly compare to this work). A further model-based algorithm is **CESNA** [37], which addresses the different problem of discovering overlapping communities. *Projection-based* approaches focus on the reduction of the attribute set, omitting attributes are irrelevant for some clusters. A recent work along this line is [30]. Finally, there are some approaches devoted to variants of attributed graph clustering, such as: **I-Louvain** [12], which extends topological clustering to maximize both modularity and a newly introduced measure called 'inertia'; **PICS** [1], which addresses a form of co-clustering for attributed graphs by using a matrix compression-approach; **FocusCO** [28] and **CGMA** [8], which start from user-preferred clusters; **M-CRAG** [20], which generates multiple non-redundant clustering for exploratory analysis; **CAMIR** [27], which considers multi-graphs.

Overall, state-of-the-art approaches to partition attributed graphs are affected by several limitations, the first of which is efficiency. Although, the algorithm in [38] does not provide exact bounds, our analysis assessed an $\Omega(n^2)$ time and space complexity, which restricts its usability to networks with thousands of nodes. The algorithm in [36] aims at overcoming these performance issues, and does actually run faster in practice. However, as we show in Section VIII, its time and space performances heavily rely on assuming a small number of clusters. Second, similarity between elements is usually defined with exact matches on categorical attributes, so that similarity among quantitative attributes is not preserved. Further, data-structures are not maintainable, so that after a change in the input graph they will have to be fully recomputed. Finally, most of the approaches require as input the number of clusters that have to be generated [22], [33], [38], [36]. In many applications it is unclear how to choose this value or how to evaluate the correctness of the choice, so that the user is often forced to repeatedly launching the algorithm with tentative values.

III. Contributions

We propose an approach to partition attributed graphs that aims at overcoming the limitations of the state of the art discussed in Section II. Namely: (i) We propose a flexible concept of distance that can be efficiently computed and that is both tailorable (allowing the user to tune the relative importance of the semantic and structural information) and robust (accounting for both categorical and quantitative attributes). Further, our structures can be maintained when entities are added or removed without reindexing the whole dataset. (ii) We present the **SToC** algorithm to compute non-overlapping communities. **SToC** allows for a declarative specification of the desired clustering, i.e., the user has to provide the sensitivity with which two nodes are considered close rather than forecast the number of clusters in the output. (iii) We describe an experimental comparison with state-of-the-art approaches showing that **SToC** uses less time/space resources and produces better quality partitions. In particular, in addition to good quality metrics for the obtained clusters, we observe that **SToC** tends to generate clusters of homogeneous size, while most partitioning algorithms tend to produce a giant cluster and some smaller ones.

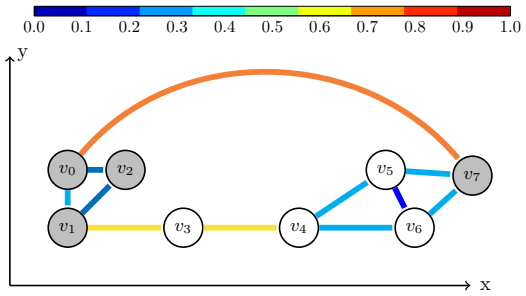
IV. Problem statement

Intuitively, attributed graphs are an extension of the structural notion of graphs to include attribute values for every node. Formally, an *attributed graph* $G(V, E, F)$ consists of a set $V = \{v_1, \ldots, v_n\}$ of *nodes*, a set $E = \{e_1, \ldots, e_m\} \subseteq V \times V$ of *edges*, and a set of mappings $F = \{f_1, \ldots, f_A\}$ such that, for $i \in [1..A]$, $f_i : V \to dom(a_i)$ assigns to a node v the value $f_i(v)$ of attribute a_i, where $dom(a_i)$ is the domain of attribute a_i. Notice that the definition is stated for directed graphs, but it readily applies to undirected ones as well. A *distance function* $d : V \times V \to \mathbb{R}_{\geq 0}$ quantifies the dissimilarity between two nodes through a non-negative real value, where $d(v_1, v_2) = d(v_2, v_1)$, and $d(v_1, v_2) = 0$ iff $v_1 = v_2$. Given a threshold τ, the *ball centered at node* v, denoted $B_d(v, \tau)$, consists of all nodes at distance at most τ: $B_d(v, \tau) = \{v' \in V \mid d(v, v') \leq \tau\}$. We distinguish between *topological distances*, that are only based on structural properties of the graph $G(V, E)$, *semantic distances*, that are only based on node attribute values F, and *multi-objective distances*, that are based on both [25]. Using on distance functions, a *cluster* can be defined by considering nodes that are within a maximum distance τ from a given node. Namely, for a distance function $d()$ and a threshold τ, a *τ-close cluster* C is a subset of the nodes in V such that there exists a node $v \in C$ such that for all $v' \in C$, $d(v, v') \leq \tau$. The node v is called a *centroid*. Observe that $C \subseteq B_d(v, \tau)$ is required but the inclusion may be strict. In fact, a node $v' \in B_d(v, \tau)$ could belong to another τ-close cluster C' due to a lower distance from the centroid of C'.

A *τ-close clustering* of an attributed graph is a partition of its nodes into τ-close clusters $C_1, \ldots, C_k$. Notice that this definition requires that clusters are disjoint, i.e., non-overlapping.

V. A multi-objective distance

In this section we introduce a multi-objective distance function that will be computable in sublinear time (Section VI-A) and that allows for tuning the semantic and topological components (Section VII). Regarding semantic distance, we assume that attributes can be either categorical or quantitative. For clarity, we assume that attributes



ID	sex	x	y	$d_S(v_0, v_i)$	$d_T(v_0, v_i)$
0	1	0	0.1	0	0
1	1	0	0	0.00471	0.4
2	1	0.1	0.1	0.0066	0.25
3	0	0.2	0	0.3427	0.6
4	0	0.4	0	0.3521	0.86
5	0	0.55	0.1	0.3596	1
6	0	0.6	0	0.3616	1
7	1	0.7	0.09	0.3327	0.86

Fig. 1. A sample attributed graph. Attributes include: sex (categorical) and spatial coordinate x, y (quantitative). Edges are colored by topological distance between the connected nodes. Topological distance $d_T()$ considers 1-neighborhoods.

$1, \ldots, Q$ are quantitative, and attributes $Q+1, \ldots, A$ are categorical. Thus, the attribute values $(f_1(v), \ldots, f_A(v))$ of a node v, boil down to a relational tuple, which we denote by $\mathbf{t}_v$. Semantic distance $d_S()$ can then be defined as:

$$d_S(v_1, v_2) = f(\mathbf{t}_{v_1}, \mathbf{t}_{v_2}) \tag{1}$$

where $f()$ is any distance function over relational tuples (see e.g., [21]). In our implementation and in experiments, we first normalize quantitative attributes in the $[0, 1]$ interval using min-max normalization [26]. Then, we adopt the following distance:

$$d_s(v_1, v_2) = \frac{\sqrt{\sum_{i=1}^{Q}((f_i(v_1) - f_i(v_2))^2} \cdot \sqrt{Q} + \sum_{i=Q+1}^{A} J(f_i(v_1), f_i(v_2))}{A} \tag{2}$$

Quantitative attributes are compared using Euclidean distance, and categorical attributes are compared using Jaccard distance ($J(X,Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|}$). The scaling factor $\sqrt{Q}$ balances the contribution of every quantitative attribute in the range $[0, 1]$. The overall distance function ranges in $[0, 1]$.

Regarding topological distance, we assume it is defined as a distance of the node neighborhoods. Let us recall the notion of l-neighborhood from [19]: the l-neighborhood $N_l(v)$ of v is the set of nodes reachable from v with a path of length at most l. $N(v)$ is a shorthand for $N_1(v)$, namely the nodes linked by v.

Topological distance $d_T()$ can then be defined as:

$$d_T(v_1, v_2) = g(N_l(v_1), N_l(v_2)) \tag{3}$$

where $g()$ is any distance function over sets of nodes, e.g., Dice, Jaccard, Tanimoto [25]. In particular, we adopt the Jaccard distance. Fig. 1 shows an example on distances. Fix node v_0, and consider distances (semantic and topological) between v_0 and the other nodes in the graph. For topological distance (with $l = 1$), e.g., we have $d_T(v_0, v_2) = J(\{v_0, v_1, v_2, v_7\}, \{v_0, v_1, v_2\}) = 1 - \frac{|\{v_0, v_1, v_2\}|}{|\{v_0, v_1, v_2, v_7\}|} = 0.25$. Thus, v_0 is closer to v_2 than to v_1, since $d_T(v_0, v_1) = 0.4$. For semantic distance (Equ. 2), instead, it turns out that v_0 is closer to v_1 than to v_2. In fact, all three of them have the same sex attribute, but v_1 is spatially closer to v_0 than to v_2.

Finally, semantic and topological distance can be combined into a multi-objective distance $d_{ST}()$ as follows [6]:

$$d_{ST}(v_1, v_2) = h(d_S(v_1, v_2), d_T(v_1, v_2)) \tag{4}$$

where $h : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is such that $h(x, y) = 0$ iff $x = y = 0$. This and the assumptions that $d_S()$ and $d_T()$ are distances imply that $d_{ST}()$ is a distance. [11], [32], [38] set $h(x, y) = x + y$ as the sum of semantic and topological distances. If $x \gg y$ then $h(x, y) \approx x$, so the largest distance weights more. However, if $x \approx y$ then $h(x, y) \approx 2x$, which doubles the contribution of the equal semantic and topological distances. In this paper, we consider instead $h(x, y) = max(x, y)$. If $x \gg y$ then $h(x, y) \approx x \approx y$, as before. However, when $x \approx y$ then $h(x, y) \approx x$, which agrees with the distance of each component.

VI. The SToC Algorithm

For a given distance threshold τ, **SToC** iteratively extracts τ-close clusters from the attributed graph starting from random seeds. This is a common approach in many clustering algorithms [6], [14]. Nodes assigned to a cluster are not considered in the subsequent iterations, thus the clusters in output are not overlapping. The algorithm proceeds until all nodes have been assigned to a cluster. This section details the **SToC** algorithm. We will proceed bottom-up, first describing the **STo-Query** procedure which computes a τ-close cluster with respect to a given seed s, and the data structures that make its computation efficient. Then, we will present the main **SToC** procedure.

A. The STo-Query Procedure

The **STo-Query** procedure computes a connected τ-close cluster C for a given seed s and threshold τ. With our definition of $d_{ST}()$, it turns out that $C \subseteq B_{d_{ST}}(s, \tau) = B_{d_S}(s, \tau) \cap B_{d_T}(s, \tau)$. We define C as the set of nodes in $B_{d_{ST}}(s, \tau)$ that are connected to s. Computing C is then possible through a partial traversal of the graph starting from s. This is the approach of the **STo-Query** procedure detailed in Algorithm 1.

The efficiency of **STo-Query** relies on two data structures, S and T, that we adopt for computing semantic and topological distances respectively and, a fortiori, for the test $d_{ST}(s, x) \leq \tau$ at line 7. Recall that $d_{ST}(s, x) = max(d_S(s, x), d_T(s, x))$. Semantic distance is computed by directly applying (Equ. 2). We store in a dictionary S the mapping of nodes to attribute values. Thus, computing

Algorithm 1: STo-Query algorithm.

Input : G, τ, s
Output: C, a τ-close connected cluster
Global : S and T data structures (described in Section VI-A, used to compute d_{ST})

```
1  Q ← empty queue
2  C ← {s}
3  ENQUEUE(Q, s)
4  while Q ≠ ∅ do
5  |  v ← DEQUEUE(Q)
6  |  foreach x ∈ N(v) do
7  |  |  if x ∉ C and d_ST(s, x) ≤ τ then
8  |  |  |  C ← C ∪ {x}
9  |  |  |  ENQUEUE(Q, x)
10 |  |  end
11 |  end
12 end
13 return C;
```

Algorithm 2: SToC algorithm.

Input : G(V, E, F) attributed graph, τ distance threshold
Output: R, a τ-close clustering of G

```
1  S ← global dictionary of the semantic vectors
2  T ← global topological similarity table of G
3  R ← ∅
4  V' ← V
5  while V' ≠ ∅ do
6  |  s ← select_node(G)
7  |  C ← STo-Query(G, τ, s)
8  |  V' ← V' \ C
9  |  G ← G[V']
10 |  R ← R ∪ {C}
11 end
12 return R
```

$d_S(s,x)$ requires $O(A)$ time, where A is the number of attributes. For topological distance, instead, a naïve usage of (Equ. 3) would require to compute online the l-neighborhood of nodes. This takes $O(n)$ time for medium-to-large values of l, e.g., for small-world networks. We overcome this issue by approximating the topological distance with a bounded error, by using *bottom-k* sketch vectors [10]. A sketch vector is a compressed representation of a set, in our case an l-neighborhood, that allows the estimation of functions, in our case topological distance, with bounded error. The bottom-k sketch $S(X)$ consists in the first k elements of a set X with respect to a given permutation of the domain of elements in X. The Jaccard distance between $N_l(v_1)$ and $N_l(v_2)$ can be approximated using $S(N_l(v_1))$ and $S(N_l(v_2))$ in their place, with a precision ϵ by choosing $k = \frac{\log n}{\epsilon^2}$ (see [15]). We store in a dictionary T the mappings of nodes v to the sketch vector of $N_l(v)$. T allows for computing $d_T(s,x)$ in $O(\log n)$ time.

B. The SToC Procedure

The algorithm consists of repeated calls to the **STo-Query** function on selected seeds. τ-close clusters returned by **STo-Query** are added to output and removed from the set of active nodes V' in Algorithm 2 (lines 7–9). We denote by $G[V']$ the subgraph of G with only the nodes in V'. Seeds are chosen randomly among the active nodes through the `select_node` function (line 6). This philosophy can be effective in real-world networks (see, e.g., [13]), and is inherently different from selecting a set of random seeds in the beginning (as in k-means), since it guarantees that each new seed will be at a significant distance from previously chosen ones. Calls to **STo-Query** return non-overlapping clusters, and the algorithm terminates when all nodes have been assigned to some cluster, i.e., $V' = \emptyset$.

C. Time and space complexity

Let us consider time complexity first. For a seed s, **STo-Query** iterates over nodes in $C \subseteq N_l(s)$ through queue Q. For each node $v \in C$, the distance $d_{ST}(v,x)$ is calculated for all neighborhoods x of v. Using the data structures S and T, this takes $O(\log n + A)$. **SToC** iterates over seeds s by removing from the graph nodes that appear in the cluster C returned by **STo-Query**. This implies that a node is enqued in Q exactly once. In summary, worst-case complexity of **SToC** is $O(\sum_{x \in V} |N(x)|(\log n + A)) = O(m(\log n + A))$. According to related work, we consider A to be constant in real world datasets. This leads to an overall time complexity of $O(m \log n)$. The initialization of the data structures S and T has the same cost. In fact, S can be filled in linear time $O(n)$ through a scan of the input attributed graph. Moreover, bottom-k sketches can be computed in $O(mk)$ time [5], hence, for $k \in O(\log n)$, building T requires $O(m \log n)$.

Regarding space usage, the dictionary S requires $O(nA) = O(n)$ space, assuming A constant. Moreover, since each sketch vector in T has size $O(\log n)$, T requires $O(n \log n)$ space. Thus, **SToC** requires $O(n \log n)$ space, in addition to the one for storing the input graph.

VII. Auto-tuning of parameters

The **SToC** algorithm assumes two user parameters[1] in input: the value l to be used in topological distance (Equ. 3), and the distance threshold τ tested at line 7 of **STo-Query**. The correct choice of such parameters can be critical and non-trivial. For example, consider the cumulative distributions of $d_S()$ and $d_T()$ shown in Fig. 2 for the datasets that will be considered in the experiments. Small values of l make most of the pairs very distant, and, conversely, high values of l make most of the pairs close w.r.t. topological distance. Analogously, high values of threshold τ may lead to cluster together all nodes, which have semantic and topological distance both lower than τ. E.g., almost every pair of nodes has a semantic distance lower than 0.4 for the DIRECTORS dataset in Fig. 2.

Another issue with parameters l and τ is that they are operational notions, with no clear intuition on how they impact on the results of the clustering problem. In

[1]The error threshold ϵ is more related to implementation performance issues rather than to user settings.

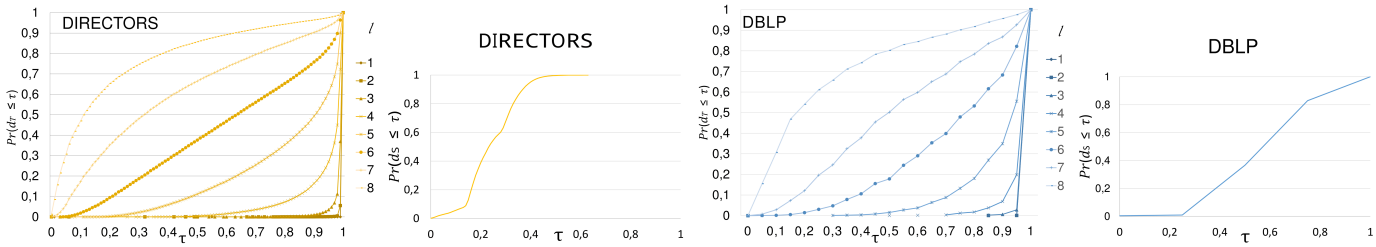


Fig. 2. Cumulative distributions of $d_T()$, for varying l-neighborhoods, and of $d_S()$ for datasets DIRECTORS (left) and DBLP (right).

this section, we introduce a declarative notion, with a clear intuitive meaning for the user, and that allows to derive optimal values for l and τ. We define the *attraction ratio* α as a value between 0 and 1, as a specification of the expected fraction of nodes similar to a given one. Extreme values $\alpha = 1$ or $\alpha = 0$ mean that all nodes are similar to each other and all nodes are different from each other respectively. In order to let the user weight separately the semantic and the topological components, we actually assume that a semantic attraction ratio α_S and a topological attraction ratio α_T are provided by the user. We describe next how the operational parameters l and τ are computed from the declarative ones α_S and α_T.

Computing τ. We approximate the cumulative distribution of $d_S()$ among all the n^2 pairs of nodes by looking at a sample of $\frac{2\log n}{\epsilon^2}$ pairs. By the Hoeffding bound [15], this guarantees error ϵ of the approximation. Then, we set $\tau = \hat{\tau}$ as the α_S-quantile of the approximated distribution. By definition, $d_S()$ will be lower or equal than $\hat{\tau}$ for the fraction α_S of pairs of nodes. Fig. 2 (second and fourth plots) show the approximated distributions of $d_S()$ for the DIRECTORS and DBLP datasets. E.g., an attraction ratio $\alpha_S = 0.4$ can be reached by choosing $\tau = 0.2$ for DIRECTORS and $\tau = 0.45$ for DBLP.

Computing l. In the previous step, we fixed $\tau = \hat{\tau}$ using the semantic distance distribution. We now fix l using the topological distance distribution. The approach consists in approximating the cumulative distribution of $d_T()$, as done before, for increasing values of l starting from $l = 1$. For each value of l, we look at the quantile of $\hat{\tau}$, namely the fraction $\alpha_l = Pr(d_T \leq \hat{\tau})$ of pairs of nodes having topological distance at most $\hat{\tau}$. We choose the value l for which $|\alpha_l - \alpha_T|$ is minimal, namely for which α_l is the closest one to the attraction ratio α_T. Since α_l is monotonically decreasing with l, we stop when $|\alpha_{l+1} - \alpha_T| > |\alpha_l - \alpha_T|$. Fig. 3 shows an example for $\alpha_T = 0.3$ and $\hat{\tau} = 0.6$. The value $l = 6$ yields the quantile α_l closest to the expected attraction ratio $\alpha_T = 0.3$.

We now show that the cost of the auto-tuning phase is bounded by $O(m \log n)$, under the assumptions that both ϵ^{-1} and l are $O(1)$. Such assumption are realistic. In fact, values for ϵ cannot be too small, otherwise the performance improvement of using bottom-k sketches would be lost [10]. Regarding l, it is bounded by the graph diameter, which for real-world networks, can be considered bounded by a constant [2], [31]. Let us consider then the computational cost of auto-tuning. Computing τ requires calculating semantic distance among $\frac{2\log n}{\epsilon^2}$ pairs of nodes,

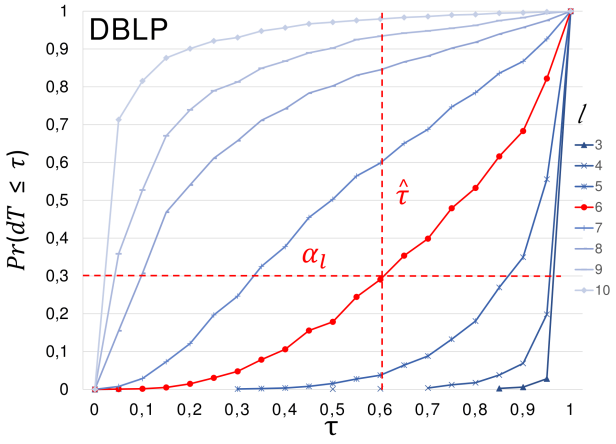


Fig. 3. Computing the l value for $\alpha_T = 0.3$ and $\tau = 0.6$ on the DBLP dataset.

which requires $O(\log^2 n)$, and in sorting the pairs accordingly, which requires $O((\log n)(\log \log n))$. Overall, the cost is $O(\log^2 n)$. Computing l requires a constant-bounded loop. In each iteration, we need to build an approximate cumulative distribution of the topological distance $d_T()$, which, as shown before, is $O(\log^2 n)$. In order to compute $d_T()$ we also have to construct the data structure T for the value l at each iteration, which requires $O(m \log n)$. In summary, computing l has a computational cost that is in the same order of the **SToC** algorithm.

VIII. Experiments

We first present the evaluation metrics and the experimental datasets, and then show the results of our experiments, which aim at showing the effectiveness of the approach and at comparing it with the state of the art.

A. Evaluation metrics

According to existing approaches [18], [36], [38]we adopt both a semantic and a topological metric. The semantic metric is the *Within-Cluster Sum of Squares* (WCSS) also called *Sum of Squared Error* (SSE), widely used by widespread approaches such as the k-means [23]:

$$WCSS = \sum_{i=1}^{k} \sum_{v \in C_i} \|v - \mu_i\|^2 \quad (5)$$

where $C_1, \ldots, C_k$ is the clustering of the graph, and, for $i \in [1, k]$, μ_i is the centroid of nodes in C_i w.r.t. the semantics distance $d_S()$ [29]. WCSS ranges over non-negative

Topology		*Attributes*	
DBLP			
Nodes	*Edges*	*Categorical*	*Quantitative*
60,977	774,162	topic	prolific
DIRECTORS			
Nodes	*Edges*	*Categorical*	*Quantitative*
3,609,806	12,651,511	sectors, sex	age, birthplace, residence
DIRECTORS-gcc			
Nodes	*Edges*	*Categorical*	*Quantitative*
1,390,625	10,524,042	sectors, sex	age, birthplace, residence

TABLE I. SUMMARY OF EXPERIMENTAL DATASETS.

numbers, with lower values denoting better clusterings. Alternative semantic metrics, such as entropy [36], [38], are suitable for categorical/discretized attributes only.

The topological metric is the *modularity* [9], a de-facto standard for graph clustering evaluation [36], [38]:

$$Q = \frac{1}{2m} \sum_{v,w \in V} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w) \qquad (6)$$

where A is the adjacent matrix of the graph, k_v is the degree of node v, c_v is the cluster ID of node v, and δ is the identity function ($\delta(i,j) = 1$ if $i = j$ and 0 otherwise). Q is defined in $[-\frac{1}{2}, 1]$, with a random clustering expected to have $Q = 0$, and a good clustering $Q > 0$ [9].

B. Experimental datasets

We will run experiments on two datasets, whose summaries are shown in Table I.

DIRECTORS: A social network of directors. A *director* is a person appointed to serve on the board of a company. We had a unique access to a snapshot of all Italian boards of directors stored in the official Italian Business Register. The attributed graph is build as follows: nodes are distinct directors, and there is an (undirected) edge between two directors if they both seat in a same board. In other words, the graph is a bipartite projection of a bipartite graph directors-companies. In the following we distinguish the whole graph (DIRECTORS) from its giant connected component (DIRECTORS-gcc). Node attributes include quantitative characteristics of directors (age, geographic coordinates of birth place and residence place) and categorical characteristics of them (sex, and the set of industry sectors of companies they seats in the boards of – e.g., such as IT, Bank, etc.). Clustering this network means finding communities of people tied by business relationships. A combined semantic-topological approach may reveal patterns of structural aggregation as well as social segregation [4]. For example, clusters may reveal communities of youngster/elderly directors in a specific sub-graph.

DBLP: Scientific coauthor network. This dataset consists of the DBLP bibliography database restricted to four research areas: databases, data mining, information retrieval, and artificial intelligence. The dataset was kindly provided by the authors of [38], where it is used for evaluation of their algorithm. Nodes are authors of scientific publications. An edge connect two authors that have co-authored at least one paper. Each node has two attributes: *prolific* (quantitative), counting the number of papers of the author, and *primary topic* (categorical), reporting the most frequent keyword in the author's papers.

Fig. 2 shows the cumulative distributions of semantic and topological distances for the datasets. In particular, the $1st$ and $3rd$ plots show the impact of the l parameter on the topological distance. The smaller (resp. larger) l, the more distant (resp. close) are nodes. This is in line with the small-world phenomenon in networks [34, Fig.2].

C. Experimental results

We will compare the following algorithms:

- **Inc-C**: the *Inc-Clustering* algorithm by Zhou et al. [38]. It requires in input the number k of clusters to produce. Implementation provided by the authors.
- **GBAGC**: the *General Bayesian framework for Attributed Graph Clustering* algorithm by Xu et al. [36], which is the best performing approach in the literature. It also takes k as an input. Implementation provided by the authors.
- **SToC**: our proposed algorithm, which takes in input the attraction ratios α_S and α_T, and the error threshold ϵ.
- **ToC**: a variant of **SToC** which only considers topological information (nodes and edges). It takes in input α_T and the error threshold ϵ (τ and l are computed as for **SToC**, with a dummy $\alpha_S = \alpha_T$).
- **SC**: a variant of **SToC** which only considers semantic information (attributes). It takes in input α_S and the error threshold ϵ.

All tests were performed on a machine with two Intel Xeon Processors E5-2695 v3 (35M Cache, 2.30 GHz) and 64 GB of main memory running Ubuntu 14.04.4 LTS. **SToC**, **ToC** and **SC** are implemented in Java 1.8, **Inc-C** and **GBAGC** are developed in MatLab.

Table II shows the results of **SToC** for $\alpha_S = \alpha_T = \alpha$ varying from 0.1 to 0.9, and a fixed $\epsilon = 0.9$. For every dataset and α, we report the number of clusters found (k), the evaluation metrics Q and $WCSS$, the running time, and the main memory usage. As general comments, we can observe that k and $WCSS$ are inversely proportional to α, Q is always non-negative and in good ranges, memory usage is limited, and running times are negligible for DBLP and moderate for DIRECTORS and DIRECTORS-gcc. For every α, we executed 10 runs of the algorithm, which uses random seeds, and reported mean value and standard deviation. The low values of the standard deviations of Q and $WCSS$ show that the random choice of the seeds does not impact the stability of the results. For lack of space, the results of **ToC** and **SC** can be found in in [3]: in summary, the exploitation of both semantic and topological information leads to a superior performance of **SToC** w.r.t. both Q and $WCSS$.

α	k	Q	$WCSS$	Time (s)	Space (GB)
DBLP					
0.1	25,598 (1,279)	0.269 (0.014)	5,428 (520)	0.627 (0.21)	0.65
0.2	26,724 (1,160)	0.270 (0.015)	5,367 (534)	0.616 (0.214)	0.64
0.4	6,634 (5,564)	0.189 (0.12)	23,477 (3,998)	0.272 (0.151)	0.67
0.6	9,041 (7,144)	0.153 (0.08)	21,337 (5,839)	0.281 (0.1)	0.7
0.8	11,050 (4,331)	0.238 (0.1)	20,669 (3,102)	0.267 (0.11)	0.69
0.9	15,041 (5,415)	0.188 (0.045)	16,688 (4,986)	0.28 (0.05)	0.64
DIRECTORS					
0.1	2,591,184 (311,927)	0.1347 (0.0237)	7,198 (5,382)	3,698 (485)	9
0.2	2,345,680 (162,444)	0.1530 (0.0084)	9,793 (1,977)	3,213 (167.3)	8.2
0.4	1,891,075 (34,276)	0.22 (0.023)	26,093 (1,829)	2,629 (178.6)	8.6
0.6	1,212,440 (442,011)	0.28 (0.068)	72,769 (4,129)	17,443 (2,093)	8.7
0.8	682,800 (485,472)	0.1769 (0.066)	49,879 (8,581)	8,209 (12,822)	9.5
DIRECTORS-gcc					
0.1	886,427 (123,420)	0.102 (0.016)	6158 (2886)	278.4 (44.7)	10
0.2	901,306 (47,486)	0.103 (0.02)	4773 (2450)	274 (14.1)	10.4
0.4	811,152 (28,276)	0.1257 (0.0164)	8,050 (1450)	239.1 (11.6)	4.3
0.6	664,882 (63,334)	0.248 (0.0578)	13,555 (3,786)	181.6 (24.4)	4.2
0.8	584,739 (408,725)	0.189 (0.0711)	49,603 (9,743)	5,979 (10,518)	8.8

TABLE II. **SToC** RESULTS (MEAN VALUES OVER 10 RUNS, STDEV IN BRACKETS).

k	Q	$WCSS$	Time (s)	Space (GB)
15	-0.499	38,009	870	31.0
100	-0.496	37,939	1,112	31.0
1,000	-0.413	37,051	1305	31.1
5,000	-0.136	32,905	1,273	32.1
15,000	0.083	13,521	1,450	32.2

TABLE III. **Inc-C** RESULTS FOR THE DBLP DATASET.

k (actual)	Q	$WCSS$	Time (s)	Space (GB)
DBLP				
10 (10)	0.0382	27,041	17	0.5
50 (14)	0.0183	27,231	25	0.6
100 (2)	$1 \cdot 10^{-7}$	27,516	13	0.7
1,000 (3)	$6 \cdot 10^{-4}$	27,465	37	3
5,000 (2)	$2 \cdot 10^{-5}$	27,498	222	14.2
15,000 (1)	0	27,509	663	50.182
DIRECTORS				
10 (8)	0.0305	198797	18	4.3
50 (10)	0.0599	198792	63	12.4
100 (8)	0.1020	198791	120	22.4
500 (5)	0.0921	198790	8,129	64.3
1000 (–)	-	-	-	out of mem
DIRECTORS-gcc				
10 (8)	0.1095	75103	94	3.02
50 (14)	0.0563	75101	161	5.47
100 (15)	0.0534	75101	234	9.34
500 (5)	0.0502	75101	1,238	40.3
1000 (7)	0.0569	75101	3,309	59
1500 (–)	–	–	–	out of mem

TABLE IV. **GBAGC** RESULTS.

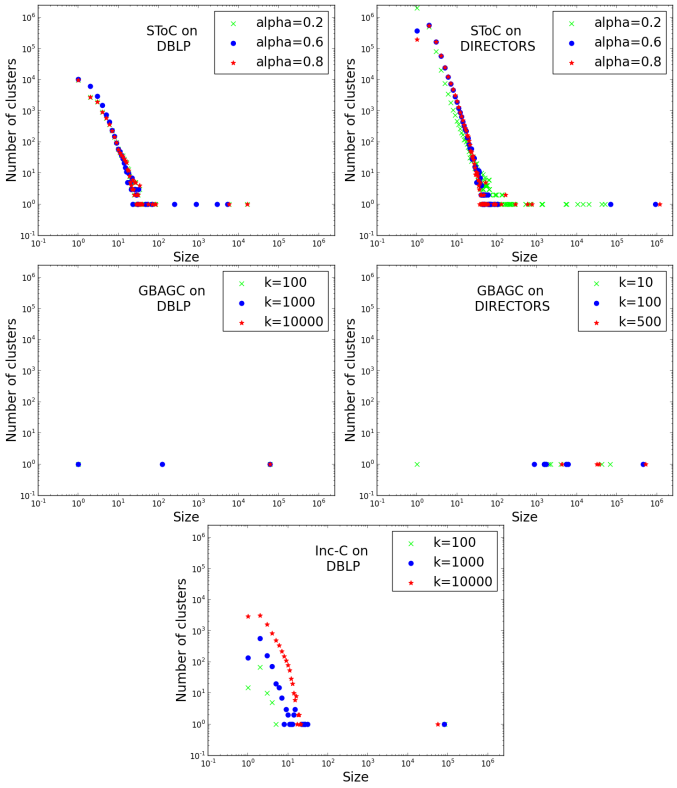


Fig. 4. Size distribution of clusters found by **SToC**, **GBAGC** and **Inc-C** for some of the tests in Tables II, III, IV.

Tables III and IV report the results for **Inc-C** and **GBAGC** respectively. Due to the different input parameters of such algorithms, we can compare the results of **SToC** only by looking at rows with similar k. Let us consider **Inc-C** first. Running times are extremely high, even for the moderate size dataset DBLP. It was unfeasible to obtain results for DIRECTORS. Space usage is also high, since the algorithm is in $O(n^2)$. Values of Q are considerably worse than **SToC**. $WCSS$ tends to generally high. Consider now **GBAGC**. Quality of the results is considerably lower than **SToC** both w.r.t. Q and $WCSS$. The space usage and elapsed time increase dramatically with k, which is non-ideal for large graphs, where a high number of cluster is typically expected. On our experimental machine, **GBAGC** reaches a limit with $k = 500$ for the DIRECTORS dataset by requiring 65GB of main memory. Even more critical is the fact that the number of clusters actually returned by **GBAGC** is only a fraction of the input k, e.g., it is 1 for $k = 15,000$ for the DBLP dataset. The user is not actually controlling the algorithm results through the required input.

Figure 4 clarifies the main limitation of **Inc-C** and **GBAGC** over **SToC**. It reports for some of the executions the size distributions of clusters found. **Inc-C** (bottom plot) tends to produce a single giant cluster including most of the nodes. **GBAGC** (middle plots) produces a small number of clusters regardless of the input parameters. Instead, **SToC** (top plots) produces more balanced results, typically expected in sociology [7], [24], with a size distribution in line with common power-laws found in real-world network and with the input graphs in particular (see [4] for the DIRECTORS graph).

IX. Conclusions

We proposed **SToC**, a clustering algorithm for large attributed graphs. It extracts non-overlapping clusters using a combined distance that accounts for network topology and semantic features, based on declarative parameters (attraction ratios) rather than on operational ones (number of clusters) typically required by other approaches. Experimental results showed that **SToC** outperforms the state of the art algorithms in both time/space usage and in quality of the clustering found.

Acknowledgements

The authors would like to thank Andrea Marino and Andrea Canciani for useful conversations.

References

[1] L. Akoglu, H. Tong, B. Meeder, and C. Faloutsos. Pics: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SDM*, pages 439–450. SIAM, 2012.

[2] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. Four degrees of separation. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 33–42. ACM, 2012.

[3] A. Baroni, A. Conte, M. Patrignani, and S. Ruggieri. Efficiently clustering very large attributed graphs. *CoRR*, abs/1703.08590, 2017.

[4] A. Baroni and S. Ruggieri. Segregation discovery in a social network of companies. In *IDA*, pages 37–48. Springer, 2015.

[5] P. Boldi, M. Rosa, and S. Vigna. Hyperanf: Approximating the neighbourhood function of very large graphs on a budget. In *WWW*, pages 625–634. ACM, 2011.

[6] C. Bothorel, J. D. Cruz, M. Magnani, and B. Micenková. Clustering attributed graphs: models, measures and methods. *Network Science*, 3(03):408–444, 2015.

[7] J. G. Bruhn. *The sociology of community connections*. Springer Science & Business Media, 2011.

[8] J. Cao, S. Wang, F. Qiao, H. Wang, F. Wang, and S. Y. Philip. User-guided large attributed graph clustering with multiple sparse annotations. In *PAKDD*, pages 127–138. Springer, 2016.

[9] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Phys rev E*, 70(6):066111, 2004.

[10] E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In *PODS*, pages 225–234. ACM, 2007.

[11] D. Combe, C. Largeron, E. Egyed-Zsigmond, and M. Géry. Combining relations and text in scientific network clustering. In *ASONAM*, pages 1248–1253. IEEE, 2012.

[12] D. Combe, C. Largeron, M. Géry, and E. Egyed-Zsigmond. I-louvain: An attributed graph clustering method. In *IDA*, pages 181–192. Springer, 2015.

[13] A. Conte, R. Grossi, and A. Marino. Clique covering of large real-world networks. In *ACM Symposium on Applied Computing*, SAC '16, pages 1134–1139. ACM, 2016.

[14] M. Coscia, F. Giannotti, and D. Pedreschi. A classification for community discovery methods in complex networks. *Statistical Analysis and Data Mining*, 4(5):512–546, 2011.

[15] P. Crescenzi, R. Grossi, L. Lanzi, and A. Marino. A comparison of three algorithms for approximating the distance distribution in real-world graphs. In *TAPAS*, pages 92–103. Springer, 2011.

[16] R. Diestel. Graph theory. 2005. *Grad. Texts in Math*, 2005.

[17] Y. Ding. Community detection: Topological vs. topical. *Journal of Informetrics*, 5(4):498–514, 2011.

[18] K.-C. Duong, C. Vrain, et al. A filtering algorithm for constrained clustering with within-cluster sum of dissimilarities criterion. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 1060–1067. IEEE, 2013.

[19] S. Günnemann, B. Boden, and T. Seidl. Db-csc: a density-based approach for subspace clustering in graphs with feature vectors. In *Machine Learning and Knowledge Discovery in Databases*, pages 565–580. Springer, 2011.

[20] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *RecSys*, pages 53–60, New York, NY, USA, 2009. ACM.

[21] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publ. Inc., 3rd edition, 2011.

[22] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE T Pattern Anal*, 24(7):881–892, 2002.

[23] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[24] D. W. McMillan and D. M. Chavis. Sense of community: A definition and theory. *J Community Psychol*, 14(1):6–23, 1986.

[25] M.M. and E. Deza. *Encyclopedia of distances*. Springer, 2009.

[26] I. B. Mohamad and D. Usman. Standardization and its effects on k-means clustering algorithm. *Res J Appl Sci Eng Technol*, 6(17):3299–3303, 2013.

[27] A. Papadopoulos, D. Rafailidis, G. Pallis, and M. D. Dikaiakos. Clustering attributed multi-graphs with information ranking. In *DEXA*, pages 432–446. Springer, 2015.

[28] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller. Focused clustering and outlier detection in large attributed graphs. In *ACM SIGKDD*, pages 1346–1355. ACM, 2014.

[29] M. H. Protter and C. B. Morrey. *College calculus with analytic geometry*. Addison-Wesley, 1977.

[30] P. I. Sánchez, E. Müller, K. Böhm, A. Kappes, T. Hartmann, and D. Wagner. Efficient algorithms for a robust modularity-driven clustering of attributed graphs. In *SDM*. SIAM, 2015.

[31] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[32] N. Villa-Vialaneix, M. Olteanu, and C. Cierco-Ayrolles. Carte auto-organisatrice pour graphes étiquetés. In *Atelier Fouilles de Grands Graphes (FGG)-EGC*, pages Article–numéro, 2013.

[33] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[34] D. J. Watts. Networks, dynamics, and the small-world phenomenon 1. *Am J Sociol*, 105(2):493–527, 1999.

[35] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *SIGMOD*, pages 505–516. ACM, 2012.

[36] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. Gbagc: A general bayesian framework for attributed graph clustering. *TKDD*, 9(1):5, 2014.

[37] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM*, pages 1151–1156. IEEE, 2013.

[38] Y. Zhou, H. Cheng, and J. X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*, pages 689–698. IEEE, 2010.