Efficient Algorithms for a Robust Modularity-Driven Clustering of Attributed Graphs

Patricia Iglesias Sánchez [•] Emmanuel Müller [•°] Uwe Leo Korn [•]
Klemens Böhm [•] Andrea Kappes [•] Tanja Hartmann [•] Dorothea Wagner [•]

[•] Karlsruhe Institute of Technology, Germany
{firstname.lastname}@kit.edu, patricia.iglesias@kit.edu, uwe.korn@alumni.kit.edu

[°] University of Antwerp, Belgium
{emmanuel.mueller}@ua.ac.be

Abstract

Clustering methods based on modularity are well-established and widely used for graph data. However, today's applications store additional attribute information for each node in the graph. This attribute information may even be contradicting with the graph structure, which raises a major challenge for the simultaneous mining of both information sources. For attributed graphs it is essential to be aware of such contradicting effects caused by irrelevant attributes and highly deviating attribute values of outlier nodes. In this work, we focus on the robustness of graph clustering w.r.t. irrelevant attributes and outliers. We propose a modularity-driven approach for parameter-free clustering of attributed graphs and several efficient algorithms for its computation. The efficiency is achieved by our incremental calculation of attribute information within these modularity-driven algorithms. In our experiments, we evaluate our modularity-driven algorithms w.r.t. the new challenges in attributed graphs and show that they outperform existing approaches on large attributed graphs.

1 Introduction

A wide range of applications in industry and sciences use graph clustering for knowledge discovery. Analysis of social networks or gene interactions in biological networks are two examples. Many domains require the extraction of clusters as sets of similar nodes. Different definitions of a graph cluster have been proposed [5]. The most commonly used notion is to group nodes that are densely connected within a cluster and have sparse connections to other clusters. A quality measure is used to score each cluster based on this model. The overall clustering result is then obtained by optimizing a function (e.g., the sum of scores) over all clusters. In particular, *modularity* [14] is a well-established quality measure for graph clustering, for which scalable processing schemes exist [2, 17, 21]. It has several benefits such as parameter-freeness.

However, each object is not only characterized by its relationships to other objects (graph structure), but also by individual properties (node attributes). For example, a social network consists of both: (1) the friendship relationship between persons represented by the graph and (2) the personal information of each person such as *age*, *income* or *gender*. A core challenge with attributed graphs is that they contain a large number of attributes, and not all of them are relevant for each cluster [8, 1]. Some attributes may have scattered values, or they may contradict the graph structure (e.g., *dissasortative* networks [15]). A social group *athletes* may be characterized by both its graph connectivity and their *sports activity level*, but the node attribute *income* is not relevant due to the different salaries within this group. This group only forms a cluster w.r.t. the attribute *sports activity level* and the graph structure. If *income* is considered, this reduces the similarity of these nodes. In addition, some nodes may have highly deviating attribute values (i.e., low *sports activity level* of a coach) although they are embedded in a well-connected cluster considering the graph structure. Overall, enhancing modularity with attribute information requires to be robust w.r.t. irrelevant attributes and outliers.

In contrast to existing approaches [8, 10, 9, 6, 23, 30, 29, 20, 1, 25, 26, 24], our modularity-driven approach enables a parameter-free clustering of graphs with numerical node attributes that is robust w.r.t. both irrelevant attributes and outliers. To achieve this, we introduce *attribute compactness* that quantifies the relevance of the attributes within a cluster. With this, we consider only the relevant attributes for assessing the attribute similarity. It allows us to detect outliers with highly deviating attribute values within a cluster. However, taking both attribute compactness and con-

Algorithms	Numerical Attributes	Robustness		Parameter-free
		Irrelevant Attributes	Outlier	
random walks [29, 30]	✗	✗	✗	✗
MDL principle [1]	✗	✓	✓	✓
efficient graph transformations [18]	✓	✗	✗	✗
statistical models [25, 26, 24]	✗	✓\✓\✗	✗	✗
subspace selection paradigm [8, 7]	✓	✓	✗	✗
modularity-based [20, 23]	✓	✗	✗	✗
spectral clustering [9]	✓	✓	✗	✗
outlier-aware clustering CODA [6]	✓	✗	✓	✗
this work	✓	✓	✓	✓

Table 1: *Related clustering methods for attributed graphs*

ventional modularity into account for the graph clustering calls for a careful algorithmic design to ensure efficiency. We first prove the NP-hardness of our approach for attributed graphs. We then provide algebraic expressions of *attribute compactness* required for its numerically stable and incremental calculation. With this, we ensure efficiency and accuracy when generalizing well-established concepts of modularity maximization to attributed graphs. We focus on such generalizations in order to leverage well-established and efficient ideas from modularity maximization. Finally, we compare our approach to conventional modularity maximization and state-of-the-art algorithms for clustering attributed graphs. We do not only show an improvement of the results, but also better runtimes due to the incremental calculation.

2 Related Work

In Table 1, we summarize existing clustering techniques grouped by their paradigms. The robustness w.r.t. both outliers and irrelevant attributes and its parameter-freeness are the unique value proposition of our work. Basic approaches in attributed graph clustering consider all attributes [29, 30, 24, 20, 23, 6]. Although some techniques have considered the local selection of the relevant attributes [25, 26, 8, 7, 9], none of these provide a parameter-free approach that is robust w.r.t. outliers. Next, there are proposals for pre-processing steps to select relevant attributes. Feature selection methods [22] select a single attribute projection, while subspace search [10] considers multiple projections that are correlated with the entire graph structure. However, these global selection schemes do not select attributes relevant for each cluster locally.

3 Problem Overview

An attributed graph is a connected undirected graph $G = (V, E, \alpha, \omega)$ where the function $\alpha : V \longrightarrow \mathbb{R}^d$ maps each node to a vector. The graph G has a positive edge weighting $\omega : E \longrightarrow \mathbb{R}_0^+$. If two nodes u and v are not connected by an edge, we set $\omega(\{i, j\}) = 0$. Self-loops are allowed. The set of attributes is $D = \{d_1, \ldots, d_d\}$, and its cardinality is $d = |D|$. $\mathcal{C} = \{C_1, \cdots, C_k\}$ is a partitioning of the nodes V into disjoint clusters, and $\mathcal{C}(v)$ is the cluster node v belongs to. The set of all possible clusterings of G is $\mathcal{A}(G)$. $W(E) = \sum_{e \in E} \omega(e)$ is the sum of all edge weights in G. $deg(v) = \sum_{\{u,v\} \in E} \omega(\{u, v\})$ is the total weighted degree of a node v. It can be generalized to a set of vertices $S \subseteq V$: $deg(S) = \sum_{u \in S} deg(u)$. Let $E_k = \{\{v, w\} \in E : v, w \in C_k\}$ be the set of intra-cluster edges and $W(E_k) = \sum_{e \in E(k)} \omega(e)$ the sum of all edge weights of E_k. $\alpha_i(o)$ is the projection of vector $\alpha(o)$ on dimension d_i.

In this work, we propose a parameter-free modularity-driven approach to cluster attributed graphs that fulfills the following requirements:

Robustness: Basic algorithms exploit the dependencies between the graph G and all attributes D for graph clustering [6, 23, 30, 29, 20]. However, some attribute information may be contradicting with the graph structure. First, the assumption of homophily [12] (i.e. connected nodes tend to have similar characteristics) may not be fulfilled for the full attribute space [8, 1, 9, 10]. Thus, irrelevant attributes not showing dependencies with the graph structure do not have to contribute to the attribute similarity assessment within a cluster. Additionally, contradicting effects between graph structure and attribute information are observed by outlier nodes. Outliers are strong embedded within a well-formed graph cluster with similar attribute values, but they have highly deviating attribute values [6]. Specifically, they only show highly deviating attribute values within the relevant attributes. As a consequence, outliers hinder the detection of homogeneous graph clusters with similar attribute values. These objects with highly deviating values are recognizable if the cluster is known previously. Thus, outliers have to be recognized during the clustering process. Overall, the de-

sign of a robust measure for the attribute information is challenging since it has to exclude all irrelevant attributes while being robust w.r.t. outliers for the assessment of the nodes similarity. Furthermore, this measure has to enable an efficient calculation. In particular, a modularity-driven approach for clustering attributed graphs requires an incremental computation in order to generalize well-established efficient algorithms for modularity maximization to attributed graphs.

Efficiency: We prove that our modularity-driven approach for clustering attributed graphs is at least as hard as finding an optimal clustering under conventional modularity. The theorem and its proof can be found as supplementary material in our website. This justifies the use of heuristics for clustering attributed graphs. We exploit well-established ideas from multilevel algorithms [17, 2] and hierarchical agglomerative clustering [16]. In general, these greedy strategies proposed for modularity maximization are based on the increase and decrease of the score when moving a node between clusters or joining clusters. For every possible move, the algorithms have to compute a new score for each newly generated clustering. Thus, efficiency relies on a careful design that avoids to recompute the new scores for each new possible clustering result. However, they are not directly applicable to attributed graphs. Therefore, both modularity and attribute information have to allow an incremental calculation.

4 Attribute Information

To exclude irrelevant attributes for the assessment of a cluster, we introduce attribute compactness. Due to its sensitivity to outliers, our graph-cluster score based on attribute compactness decreases if an outlier is part of a cluster. For its efficient computation we provide the algebraic transformations required for an incremental calculation. This allows us to score both *modularity* and *attribute compactness* of a cluster for large attributed graphs.

4.1 Robustness The variance indicates how scattered data points are. However,it does not measure the relevance of an attribute within a group of objects. To this end, a measure known as *relevance* has been introduced [27]. Given the local variance $\sigma_i^2(C_k)$ of projected values on dimension d_i within the cluster C_k and the global variance $\bar{\sigma}_i^2$ on d_i, the relevance of attribute d_i for cluster C_k is defined as follows:

$$R_i(C_k) = \begin{cases} 1 - \frac{\sigma_i^2(C_k)}{\bar{\sigma}_i^2} & \text{if } \bar{\sigma}_i^2 \neq 0 \\ 1 & \text{if } \bar{\sigma}_i^2 = 0 \end{cases} \tag{4.1}$$

Relevance compares the variance within a cluster (local variance) to the one of the entire data set (global variance) for an attribute. Low variance within the cluster means that the attribute value shows high compactness compared to the whole database. When the attribute shows scattered values in a cluster ($\sigma_i^2(C_k) \geq \bar{\sigma}_i^2$), the relevance of the attribute is negative ($R_i(C_k) \leq 0$).

While the relevance quantifies the similarity degree of an attribute in a cluster, it does not exclude irrelevant attributes as it has been designed for a single attribute. Thus, we introduce attribute compactness to summarize the similarity of all attribute values within a cluster being robust with the irrelevant ones:

DEFINITION 1. *Attribute Compactness*
Given a cluster $C_k \in \mathcal{C}$, we define the attribute compactness $AC(C_k)$ as follows:

$$AC(C_k) = \frac{1}{d} \sum_{d_i \in D} max(R_i(C_k), 0)$$

$AC(C_k)$ is the sum of the relevances of each attribute for the cluster C_k. Irrelevant attributes show highly scattered values within the cluster C_k compared to the entire database. As a consequence, the relevance of such attributes is negative or zero ($R_i(C_k) \leq 0$). The attribute compactness leaves aside such scattered values since it considers only the positive values (relevant attributes). In consequence, irrelevant attributes do not have any impact on the quality assessment. The higher the value of $AC(C_k)$, the more attributes have similar values within the graph cluster. Since the number of relevant attributes may be different between clusters, we normalize the score with the dimensionality of the database, to ensure the same upper bound w.r.t. modularity. Hence, the range of this measure is: $0 \leq AC(C_k) \leq 1$. The maximum value is achieved when all the attributes have the same values within the cluster. If all attributes show locally scattered attributes, the attribute compactness has its minimum value. A global variance of zero $\bar{\sigma}_i^2 = 0$ indicates that the attribute has one value for the whole graph structure. In this case, attribute d_i is relevant regardless of the selected cluster w.r.t. the graph structure.

4.2 Incremental Calculation Attribute compactness requires the local variance of each cluster. The sum of squares S_i required for calculating the local variance of attribute i can be computed efficiently by scanning the data points once if one deploys the traditional *one-pass equation* for the sum of squares. However, this traditional expression leads to numerically unstable algorithms. In particular, precision decreases when $|C_k|$ is large and the variance is small. This loss of preci-

sion can be so severe that there is a negative value for $\sigma_i^2(C_k)$ even though the variance must be always positive [4]. In the following, we introduce the expressions required for the local variance and *attribute compactness* when adding a node to a cluster.

For the stable calculation of the unweighted sum of squares, [28] has proposed the following scheme. v is the node to be added, C_k is the cluster v is added to, and u is a node forming a singleton cluster:

$$\begin{aligned} T_{i,\{u\}} &= \alpha_i(u), S_{i,\{u\}} = 0 \\ T_{i,C_k\cup\{v\}} &= T_{i,C_k} + \alpha_i(v) \\ S_{i,C_k\cup\{v\}} &= S_{i,C_k} + \frac{\left(|C_k|\cdot\alpha_i(v) - T_{i,C_k\cup\{v\}}\right)^2}{|C_k|\cdot(|C_k|-1)} \end{aligned}$$

$T_{i,C_k\cup\{v\}}$ is the sum of the attribute values of C_k together with the one of node v. Then the sum of squares $S_{i,C_k\cup\{v\}}$ is updated as well. This avoids the subtraction of squared terms, and it is equivalent to the traditional expression. At the same time, it allows a numerically stable calculation of the sum of squares in constant time. Given this, we can compute the change of the variance when adding v to C_k as follows:

$$\begin{aligned} \Delta\sigma^2_{i,C_k\cup\{v\}} &= \frac{S_{i,C_k\cup\{v\}}}{|C_k|+1} - \frac{S_{i,C_k}}{|C_k|} \\ &= \frac{\left(|C_k|\cdot\alpha_i(v) - T_{i,C_k}\right)^2 - (|C_k|+1)\cdot S_{i,C_k}}{(|C_k|+1)^2\cdot|C_k|} \end{aligned}$$

We determine the increase in *attribute compactness* when adding a node v to a cluster as follows:

(4.4)

$$\begin{aligned} \Delta AC_{C_k\cup\{v\}} =& \frac{1}{d}\sum_{d_i\in D}\Big[max(1 - \frac{\sigma_i^2(C_k) + \Delta\sigma^2_{i,C_k\cup\{v\}}}{\bar{\sigma_i^2}}, 0) \\ & - max(1 - \frac{\sigma_i^2(C_k)}{\bar{\sigma_i^2}}, 0)\Big] \end{aligned}$$

In consequence, we cannot only update the attribute information for each node or cluster movement in constant time, but we can ensure high quality results due to the numerically stable calculation of the attribute compactness. These expressions are essential for the generalization of all strategies for modularity maximization that compute the quality of a clustering incrementally.

4.3 Attributes and Graphs The next step is to combine attribute compactness with modularity of the graph structure to one score. A clustering taking graph structure and vector data into account has to trade off: (1) high intra-edge and low inter-edge density and (2) similar attribute values of nodes within a cluster. In this work, we use the following simple function, but other alternatives are possible. This particular function has shown good results in our experimental evaluation (cf. Section 6) and it fulfills some interesting properties described in the following.

DEFINITION 2. *Attribute-aware modularity $AQ(\mathcal{C})$*
For a clustering $\mathcal{C}$, the attribute-aware modularity $AQ(\mathcal{C})$ is:

$$AQ(\mathcal{C}) = \sum_{C_k\in\mathcal{C}} AC(C_k)\cdot Q(C_k)$$

Since $0 \geq AC(C_k) \leq 1$, the range of *modularity* is preserved as $-\frac{1}{2} \leq AQ(\mathcal{C}) \leq 1$ with this score function [3]. A value of $AQ(\mathcal{C}) = 0$ means that no attributes has any dependency with the structure (e.g., disassortative network [15]), or the edges within the cluster are not better than random. In contrast to traditional modularity, a clustering with maximum $AQ(\mathcal{C})$ may have a cluster which consists of a single node regardless of its connections. This case appears for nodes with highly deviating values (outliers), as they reduce the compactness of a cluster when being added to it. Thus, the quality is decreased when adding them to a cluster result ($AC(C_k) \approx 0$) instead of considering these nodes as singletons ($AC(C_k) = 1$). Overall, the higher the value of $AQ(\mathcal{C})$, the better is the result w.r.t. both the graph structure and the attribute values. Thus, we have to find an optimal clustering for the maximization of $AQ(\mathcal{C})$. In the following we generalize of several relevant components of multilevel algorithms [17] for this new score.

5 Algorithms

In this work, we leverage well-established ideas from multilevel algorithms [17, 2]. However, these algorithms have been proposed for conventional graphs (without attributes), and are not applicable to attributed graphs. In addition to the formal property of incremental calculation of the quality measure (cf. Section 4.2), further considerations specific to more complex data structures (attributed graphs) are necessary. The so-called *coarsening phase* of multilevel algorithms requires the contraction of the graph structure and movements of nodes in this new structure (cf. Section 5.2). Attributes have to be considered in each of its steps. In particular, it has two major phases: (1) The input graph is contracted to a new graph whose nodes are clusters from an initial clustering $\mathcal{C}_0$ (*contraction* step), and (2) local movements are applied to this new graph [2]. In the following, we describe the generalizations of them

to attributed graphs since they are the basis for further development of multilevel algorithms.

5.1 Local Move (LM) This heuristic considers only local moves of single nodes. For each possible node move, when v is added to C_k, the change of attribute-aware modularity is calculated. v is moved to the cluster with the highest increase in attribute-aware modularity.

Algorithm 1 Local Move (LM)

Input: $G = (V, E, \alpha, \omega)$, initial clustering $\mathcal{C}_0$
Output: new clustering result $\mathcal{C}$

1: $\mathcal{C} := \mathcal{C}_0$ ▷ Initialize with an initial clustering
2: **do**
3: **for** $v \in V$ **do**
4: $C_k := \mathcal{C}(v)$ ▷ Select graph cluster of node v
5: decrease := $\Delta AQ_{C_k \setminus \{v\}}$
6: remove v from C_k
7: cand := $\{\mathcal{C}(u) \mid \{u, v\} \in E\}$
8: $C_{dest} := \arg\max_{C_n \in cand} \Delta AQ_{C_n \cup \{v\}}$
9: **if** increase + decrease ≥ 0 **then**
10: add v to C_{dest} and update $\mathcal{C}$
11: **end if**
12: **end for**
13: **while** node movement

Algorithm: Given an initial clustering $\mathcal{C}_0$, we first compute the change of attribute-aware modularity $\Delta AQ_{C_k \setminus \{v\}}$ when v is removed from its current cluster C_k. As candidates for possible movements, we select all the graph clusters in $\mathcal{C}_0$ that contain a node adjacent to v. Then we compute the increase of attribute-aware modularity $\Delta AQ_{C_n \cup \{u\}}$ if v is added to $C_n \in candidates$. We only consider node moves that increase the attribute-aware modularity. Finally, we add v to the cluster C_{dest} where the increase of *attribute-aware modularity* is maximal. If no move of v to other clusters does not increase the quality of the clustering, then node v is not moved. Additionally, some nodes increase the score when they form a singleton. For example, an outlier with highly deviating attribute values heavily brings down attribute compactness when seen as part of a cluster. So, to remove this node from a cluster improves the attribute compactness; this in turn leads to an increase of attribute-aware modularity. Therefore, we also consider the empty cluster in the set of candidates (Line 7). The algorithm terminates if there are no more score-increasing vertex movements.

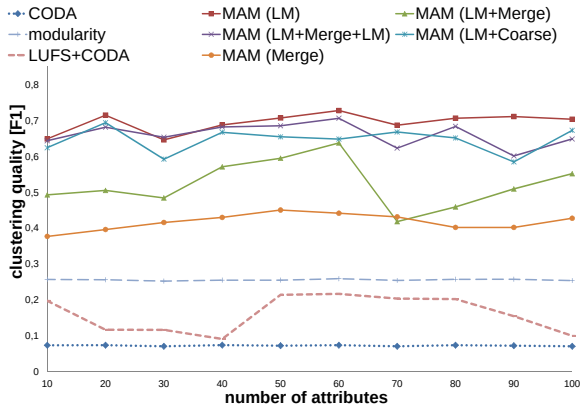
Complexity: As each node is considered once for a local movement, the algorithm requires at least $|V|$ iterations. The current cluster of a vertex u can be retrieved in $\mathcal{O}(1)$ time (Line 4). For the decrease of attribute-aware modularity, we have to compute the decrease of modularity and attribute compactness. Updating the value of modularity when removing a node v from its cluster C_k can be done in constant time. Due to the incremental calculation of attribute compactness, we determine it in constant time for each dimension ($\mathcal{O}(d)$) as well. Overall, the decrease of attribute-aware modularity is calculated in $\mathcal{O}(d)$ time. For the selection of candidates (Line 7), we have to scan all edges of node u to determine the neighboring clusters. For all candidates, the increase of attribute-aware modularity is calculated in $\mathcal{O}(deg(u) \cdot d)$. Thus, all nodes are analyzed once in $\mathcal{O}\Big(|V| + \sum_{u \in V} d \cdot deg(u)\Big) = \mathcal{O}(|V| + |E| \cdot d)$. Regarding the graph size, we have achieved the same time complexity as the same heuristic that only considers the graph structure [17], due to the incremental calculation of attribute compactness. On the other hand, the time complexity w.r.t. the number of attributes is linear. This enables an efficient calculation of attribute-aware modularity for both large and high-dimensional graphs, as shown in Section 6.

5.2 Coarsening Phase Considering only local movements of a single node can lead to a local maximum. To avoid this, multilevel algorithms create a new graph to allow movements on a higher level [2, 17]. In this graph, nodes represent clusters. Thus, a set of objects are moved for each of these local movements instead of single nodes. Therefore, the contraction of the attribute values has to be done carefully, to ensure the efficiency in the contraction phase. As the nodes of G' represent a set of nodes, we use expressions to compute incrementally the attribute compactness for joining and removing a cluster. Since our modularity-driven approach follows the general definition of modularity (weighted graphs and self-loops are allowed), we are able to contract the graph using attribute-aware modularity.

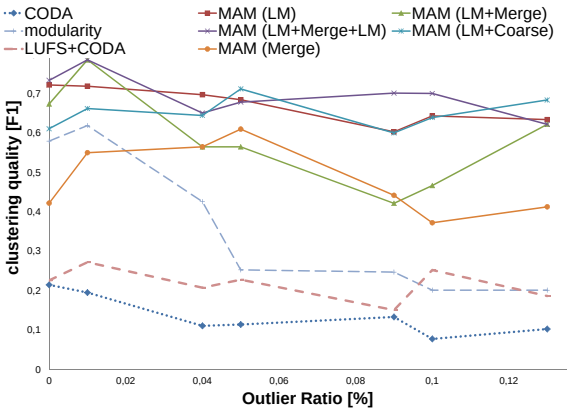
Algorithm: First, we build the nodes of the new contracted graph G' for the coarsening phase. For each cluster C_k, we create a new node v_k and a new edge e_{new} which is a self loop. It stores the sum of the weights of the edges within C_k. Regarding the attributes, both the sum of the attribute values and their sum of squares is also stored in the new graph as attributes. Then, we create the edges that connect the nodes in the new graph G_{new}. If two clusters are connected in the original graph, we add an edge connecting the nodes representing these two clusters. The weight of this new edge is the sum of all the edge weights connecting both clusters in the original graph. To complete the coarsening phase, we apply the algorithm *Local Move* (cf. Algorithm 1) to the contracted graph.

Complexity: The coarsening phase requires the creation of a new graph and the update of the attribute information. In the worst case, each node is a singleton in the initial clustering $\mathcal{C}_0$. The creation of the new nodes in the contracted graph is $\mathcal{O}(|V| \cdot d)$. The creation of connecting edges between the new nodes can be done in $\mathcal{O}(|E|)$ if we only scan the edges twice. In total, the contraction phase can be done in $\mathcal{O}(|V| \cdot d + |E|)$. Finally, the *Local Move* (cf. Algorithm 1) on a contracted graph can be done in $\mathcal{O}(|V| + |E| \cdot d)$, due to the incremental calculation when clusters are removed or moved to other clusters. Overall, coarsening requires $\mathcal{O}((|V| + |E|) \cdot d)$. A careful design considering the attribute values and the incremental calculation of attribute-aware modularity has allowed to preserve the efficiency of the original strategy regarding the graph size. At the same time, the attribute information has been considered by providing a strategy which is linear with the number of attributes.

In the framework proposed in [17], other algorithms can be combined for the generation of the clustering result for the contraction phase (e.g., merging clusters instead of local movements). Thus, we also have generalized the approach proposed in [14] for merging clusters globally instead of local movements of the nodes. However, we show that local approaches (i.e., *LM*) show better quality results and more robustness w.r.t. outliers. Furthermore, the combination of several techniques by a scaffold algorithm may lead to better or faster results than using a single one.



(a) *Quality w.r.t.* $|D|$



(b) *Quality w.r.t. the Outlier Ratio*

Figure 1: *Quality on synthetic data*

6 Experiments

We compare our approach *MAM (maximization of attribute-aware modularity)* with *CODA*, an outlier-aware graph clustering algorithm for attributed graphs [6], *LUFS+CODA*, a feature selection algorithm as preprocessing [22], and *modularity* in a conventional multi-level algorithm [17]. For all competitors we tried to find optimal parameters, while our method does not require fixing any parameters. Internally, we evaluate *MAM* with its different strategies *Merge*, *LM*, *Coarse*, and its combinations (*LM+Merge*, *LM+Merge+LM*). We use both synthetic and real world data and the *F1-value* (harmonic mean of precision and recall) for quality assessment. In order to facilitate comparability and repeatability of our experiments, we provide all datasets, algorithms, and parameter settings[1].

6.1 Synthetic Data We generated synthetic datasets of different graph sizes, dimensionalities, and outlier ratios. For each experiment, we generate five graphs to average over random effects in the generation process. The generated graphs are based on benchmark graphs [11], with power law distributions in both degree and community size. We have extended this generator adding numeric node attributes as well as community outliers. For each graph cluster C_k, we randomly select a number of relevant attributes $x \in [1, d]$ and choose their attribute values based on a Gaussian distribution with a mean μ_{C_k} and a very small standard deviation ($\sigma = 0.001$). The values for irrelevant attributes are chosen following a Gaussian distribution with $\sigma = 1$. Finally, we generate outliers by selecting clustered nodes and manipulating a random number of their relevant attribute values.

Quality and Robustness: We evaluate quality on graphs with an increasing number of attributes and of outliers, see Figures 1(a) and 1(b). The robustness of our attribute-aware modularity is clearly visible for all of our strategies. Results are best for *LM*, *LM+Coarse*, and *LM+Merge+LM*. Since traditional modularity does not consider attribute information, the obtained clusters show dissimilarity w.r.t. the attribute values. For instance, they include outliers with highly deviating

[1] http://ipd.kit.edu/~muellere/mam/

values. On the other hand, *CODA* does not perform a local attribute selection and is prone to irrelevant attributes. Although *CODA* is slightly improved by the feature selection (*LUFS + CODA*), it does not work well as a pre-processing step. This is because it does not select the relevant attributes for each cluster locally.

Runtime: In contrast to the traditional modularity, the runtime of the algorithms for attributed graphs depends on both the graph size and the number of attributes, see Figure 2. The runtime of our attribute-aware modularity is slightly higher than traditional modularity due to the combination of both information sources. However, due to the incremental calculation, all our strategies scale linearly with the number of attributes (cf. Figure 2(a)). In contrast, *CODA* does not scale w.r.t. these parameters. *LUFS+CODA* perform better w.r.t. the number of attributes since *CODA* only considers few attributes. However, the pre-processing step *LUFS* does not scale with the graph size. The strategy with the worst performance is *Merge*, due to its time complexity cf. [16]. However, the combination with other strategies (*LM+Merge*) causes significant speedups. Overall, our incremental and numerically stable calculation of attribute-aware modularity leads to accurate results (Figure 1) as well as to efficient computation (Figure 2).

6.2 Real World Data We have evaluated our approach on attributed graphs from different application domains. Table 2 shows a summary of statistics for these real-world networks: All of them have already been public available in [13, 10, 9, 19]. Since there is no ground truth given, it is difficult to impossible to do a quality assessment. However, we compare algorithms regarding the attribute-aware modularity achieved and runtime. This is commonly done in the literature to compare strategies for modularity maximization [11, 17]. Further, we describe interesting clusters and outliers found as case studies.

Graph	#nodes	#edges	#attributes
Disney [13]	124	333	28
DFB [9]	100	1106	5
ARXIV [9]	856	2660	30
IMDB [9]	862	2660	21
DBLP [19]	28112	95410	46
PATENTS [9]	100000	188631	5
Amazon [10]	314824	882930	28

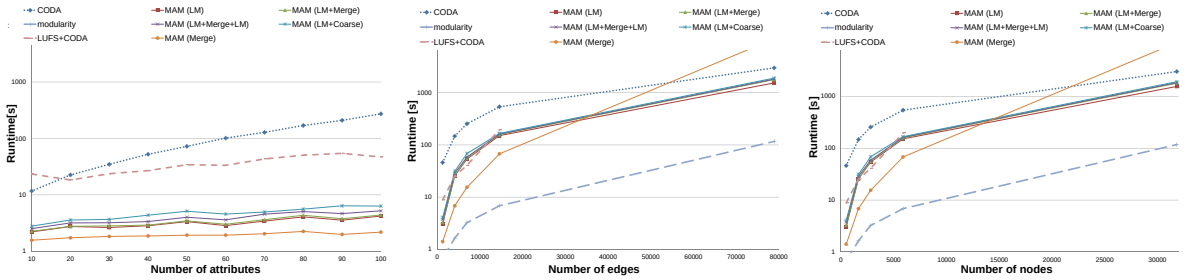
Table 2: *Real World Networks*

Attribute-Aware Modularity and Runtime: We analyze performance (i.e., $AQ(\mathcal{C})$ and runtime) w.r.t. different graph sizes and number of attributes in real world networks. See Table 3. Similarly to the synthetic data, *Merge* performs worst. Considering both runtime and quality, local movement *LM* achieves very good results compared to more complex strategies such as *LM+Merge+LM*. Further, it is the only algorithm on attributed graphs that scales up to the largest network in our evaluation. Other schemes were not able to provide a result within 5 hours. This scalability is due to the incremental calculation of attribute-aware modularity. Traditional modularity achieves good clustering results; however, it neglects attribute information and does not find homogeneous attribute values in each cluster. Although CODA takes attribute information into account, it degenerates with an increasing number of attributes. $AC(\mathcal{C})$ is best on low dimensional data (*DFB* and *PATENTS* with 5 attributes only). On the other hand, pre-processing for the selection of attributes (*LUFS+CODA*) increases the quality. However, it does not scale with the graph size, and it does not select the relevant attributes for each cluster locally.

Case Studies: We now discuss some results from the real world networks.

Disney: This dataset is a subgraph of the Amazon co-purchase network. Each product (node) is described by attributes such as prices or review ratings [13]. In contrast to traditional modularity, the clusters found by our method are smaller and more specialized. For instance, modularity extracts one cluster with family films such as *Spy Kids*, *Inspector Gadget* or *Flubber*. In contrast, our method splits this into two clusters. This is because the sequels of *Honey, We Shrunk Ourselves* have very good ratings and many reviews, compared to the other movies. Similarly to modularity, our approach also finds a cluster consisting of *Read Along films* since all these films show similar Amazon prices. However, the overpriced film *The Jungle Book* is detected as an outlier. Figure 3 shows more examples of the differences and similarities of the extracted clusters.

DBLP: The DBLP graph contains authors (nodes) with co-authorship information (edges) and 40 numeric attributes (publication ratios at certain conferences) [19]. Graph clustering based on attribute-aware modularity retrieves research communities with similar publication ratios on some conferences. Traditional modularity neglects the attribute values and brings down this similarity. For instance, it includes highly deviating outlier nodes in one of the clusters. For example, we have detected *Ted E. Senator* as an outlier. He has published several papers as single author on *ICDM* and *KDD*. He also has collaborated with individuals from two different clusters: *Henry G. Goldberg* belonging to a data mining and machine learning cluster (publishing in *ICDM*, *KDD*, *ICML*, etc.) and *Ping Shyr*, *J. Dale Kirkland*, and *Tomasz Dybala*, belonging to an artificial intelligence cluster ($AAAI \backslash IAAI$) ($AAAI \backslash IAAI$)



(a) *Runtime w.r.t.* $|D|$ (b) *Runtime w.r.t.* $|E|$ (c) *Runtime w.r.t.* $|V|$

Figure 2: *Runtimes on synthetic data*

	LM		Merge		LM+Merge		LM+Merge+LM		Coarse		modularity		CODA		LUFS+CODA	
Dataset	$AQ(\mathcal{C})$	Runt.[s]	$AQ(\mathcal{C})$	Runt. [s]	$AQ(\mathcal{C})$	Runt. [s]	$AQ(\mathcal{C})$	Runt. [s]	$AQ(\mathcal{C})$	Runt. [s]	$AQ(\mathcal{C})$	Runt. [s]	$AQ(\mathcal{C})$	Runt. [s]	$AQ(\mathcal{C})$	Runt. [s]
Disney	**0.368**	0.91	0.038	0.63	0.182	0.99	0.210	1.27	0.329	1.32	0.306	0.40	0.0	6.35	0.164	2.2
DFB	**0.123**	0.88	0.001	0.71	0.001	0.94	0.0031	1.11	**0.122**	1.25	0.0686	0.47	0.001	1.68	0.015	1.76
ARXIV	**0.246**	4.09	0.0126	3.52	0.135	4.47	0.209	7.33	0.235	8.51	0.159	0.97	0.0	46.45	0.059	13.43
IMDB	0.205	5.17	0.012	4.99	0.077	6.57	0.0806	9.66	**0.218**	10.90	0.128	1.89	0.0	13.03	0.0	10.87
DBLP	0.429	135	-	-	0.155	270.46	0.224	403.84	**0.436**	317.15	0.365	32.94	0.0	1294.47	-	-
PATENTS	0.353	477.11	-	-	0.389	729.52	**0.406**	933.41	-	-	0.162	95.27	0.064	711.601	-	-
Amazon	0.513	1354.98	-	-	-	-	-	-	-	-	0.162	493.5	-	-	-	-

Table 3: *Clustering quality achieved* $AQ(\mathcal{C})$ *and runtime on real world networks*

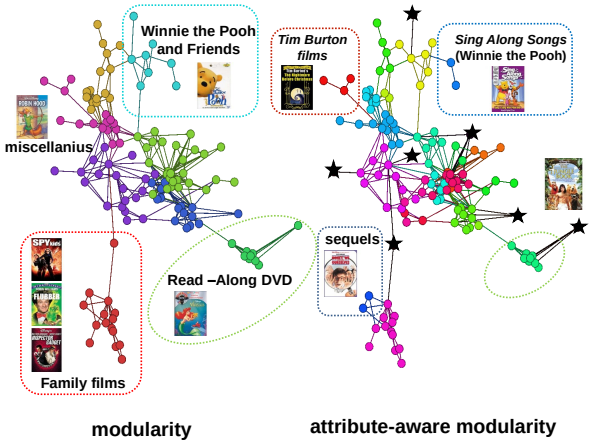


Figure 3: *Comparison between extracted clusters: traditional modularity vs. attributed-aware modularity. Outliers nodes are marked with stars and clusters are marked with different colors.*

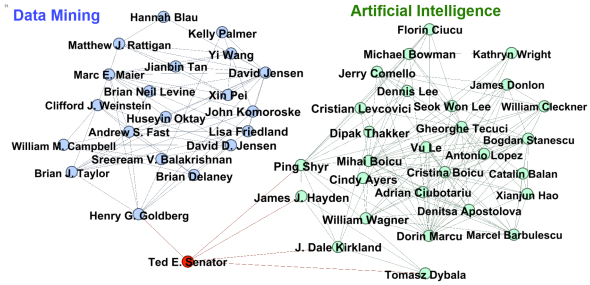


Figure 4: *Two clusters (data mining and artifical intelligence) and the outlier Ted. E. Senator with highly publishing ratios on KDD and ICDM compared to the clusters he is connected to*

(cf. Figure 4). *Ted. E. Senator* has highly deviating attribute values compared to both of these communities. Including him in one of the clusters would reduce the clustering quality.

German Soccer League (DFB): This network contains soccer players characterized by attributes such as the number of goals or of games. Two players are connected if they have played in the same club [9]. One of the clusters found is a subset of players with similar numbers of goals and penalty kicks. One exceptional player is *Ulf Kirsten*, who was one of the best German goalgetters between 1980 and 1990. He has high values in several of the attributes (e.g., *number of games*, *number of penalty kicks* and *number of goals per game*). Although he is not the player with most goals or number of games in the database, the attribute values are highly deviating compared to the ones in his graph neighborhood; therefore, it is appropriate that our clustering has identified him as an outlier.

Amazon: The *Amazon* co-purchase network [10] is most challenging due to both the graph size and the number of attributes. We found a cluster with highly similar ratings, number of reviews, and prices that consists of books for *day trading* (e.g. *Day Trade Part-time*, *Day Trade Online*, or *The Compleat Day Trader*). The prices by private sellers or for used books are similar. Additionally, all books have similar rating ratios for both *rating_5_star* and *rating_4_star*. In contrast to traditional modularity, our algorithm has detected this homogeneous cluster within a large subgraph in-

cluding more general financial books. We conclude that attribute-aware modularity is a reasonable quality measure for cluster structures in real-world networks.

7 Conclusion

In this work, we have proposed a modularity-driven approach to cluster attributed graphs. To achieve this, we have proposed attribute compactness which enables a robust clustering w.r.t. irrelevant attributes and outliers. We have proven that MAM (maximization of attribute-aware modularity) is an NP-hard problem and we have aimed for heuristic solutions. We have provided the formal property of incremental and numerically stable calculation of our measure. Further, we have introduced several strategies for efficient computation of MAM on large attributed graphs. Our evaluation on synthetic and real world networks shows the high quality and scalability of our approach.

Acknowledgments

This work is supported by the Young Investigator Group program of KIT as part of the German Excellence Initiative, by a post-doctoral fellowship of the research foundation Flanders (FWO), and by the Karlsruhe School of Services (KSOS) at KIT.

References

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

[2] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[3] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE TKDE*, 2008.

[4] T. Chan, G. Golub, and R. LeVeque. Algorithms for computing the sample variance: analysis and recommendations. *The American Statistician*, 1983.

[5] S. Fortunato. Community detection in graphs. *Physics Reports*, 2010.

[6] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On community outliers and their efficient detection in information networks. In *KDD*, 2010.

[7] S. Günnemann, B. Boden, and T. Seidl. Db-csc: a density-based approach for subspace clustering in graphs with feature vectors. *ECML PKDD*, pages 565–580, 2011.

[8] S. Günnemann, I. Färber, B. Boden, and T. Seidl. Subspace clustering meets dense subgraph mining: A synthesis of two paradigms. In *ICDM*, pages 845–850, 2010.

[9] S. Günnemann, I. Färber, S. Raubach, and T. Seidl. Spectral subspace clustering for graphs with feature vectors. In *ICDM*, 2013.

[10] P. Iglesias, E. Müller, F. Laforet, F. Keller, and K. Böhm. Statistical selection of congruent subspaces for mining attributed graphs. In *ICDM*, 2013.

[11] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 2008.

[12] M. McPherson, L. S. Lovin, and J. M. Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 2001.

[13] E. Müller, P. Iglesias, Y. Mülle, and K. Böhm. Ranking outlier nodes in subspaces of attributed graphs. In *Workshop on Graph Data Management at ICDE*, 2013.

[14] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 2004.

[15] M. E. Newman. Mixing patterns in networks. *Physical Review E*, 2003.

[16] M. E. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 2004.

[17] R. Rotta and A. Noack. Multilevel local search algorithms for modularity clustering. *ACM JEA*, 2011.

[18] Y. Ruan, D. Fuhry, and S. Parthasarathy. Efficient community detection in large networks using content and links. In *WWW*, 2013.

[19] P. I. Sánchez, E. Müller, O. Irmler, and K. Böhm. Local context selection for outlier ranking in graphs with multiple numeric node attributes. In *SSDBM*, 2014.

[20] M. Shiga, I. Takigawa, and H. Mamitsuka. A spectral clustering approach to optimally combining numericalvectors with a modular network. In *KDD*, 2007.

[21] C. Staudt and H. Meyerhenke. Engineering high-performance community detection heuristics for massive graphs. In *ICPP*, 2013.

[22] J. Tang and H. Liu. Unsupervised feature selection for linked social media data. In *KDD*, 2012.

[23] E. Viennet et al. Community detection based on structural and attribute similarities. In *ICDS*, 2012.

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

[25] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM*, 2013.

[26] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *KDD*, 2009.

[27] K. Yip, D. Cheung, and M. Ng. Harp: A practical projected clustering algorithm. *IEEE TKDE*, 2004.

[28] E. Youngs and E. Cramer. Some results relevant to choice of sum and sum-of-product algorithms. *Technometrics*, 1971.

[29] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2009.

[30] Y. Zhou, H. Cheng, and J. X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM*, 2010.