Applied Network Science (Preprint Version)

MinerLSD: Efficient Mining of Local Patterns on Attributed Networks

Martin Atzmueller · Henry Soldano · Guillaume Santini · Dominique Bouthinon

Received: / Accepted:

Abstract Local pattern mining on attributed networks is an important and interesting research area combining ideas from network analysis and data mining. In particular, local patterns on attributed networks allow both the characterization in terms of their structural (topological) as well as compositional features. In this paper, we present *MinerLSD*, a method for efficient local pattern mining on attributed networks. In order to prevent the typical pattern explosion in pattern mining, we employ closed patterns for focusing pattern exploration. In addition, we exploit efficient techniques for pruning the pattern space: We adapt a local variant of the standard Modularity metric used in community detection that is extended using optimistic estimates, and furthermore include graph abstractions. Our experiments on several standard datasets demonstrate the efficacy of our proposed novel method MinerLSD as an efficient method for local pattern mining on attributed networks.

Keywords complex networks · attributed networks · closed pattern mining · network analysis and mining · graph mining · community detection

Martin Atzmueller
Tilburg University, Department of Cognitive Science and Artificial Intelligence
Warandelaan 2, 5037 AB Tilburg, The Netherlands
Université Sorbonne Paris Cité, Paris, France
E-mail: m.atzmuller@uvt.nl

Henry Soldano · Guillaume Santini · Dominique Bouthinon
LIPN, Université Paris-13, SPC UMR-CNRS 7030, 93430, Villetaneuse, France
E-mail: {guillaume.santini,dominique.bouthinon}@lipn.univ-paris13.fr

Henry Soldano
ISYEB UMR 7205, Museum National d'Histoire Naturelle Paris, France
E-mail: henry.soldano@mnhn.fr

1 Introduction

The analysis of complex networks, e. g., by investigating structural properties and identifying interesting patterns, is an important task to make sense of such networks, in order to ultimately enable an understanding of their phenomena and structures, e. g., [2, 3, 5, 7, 8, 21, 30, 38, 39, 43, 52–54, 59, 69, 72, 75]. In this context, data mining on such networks represented as attributed graphs has recently emerged as a prominent research topic, e. g., [3, 8, 21, 30, 38, 57, 72, 75]. Methods for mining attributed graphs focus on the identification and extraction of patterns using topological information as well as compositional information on nodes and/or edges given by a set of attributes, e. g., [6, 80]. In particular, local pattern mining focuses on the identification of dense substructures in a graph that are captured by specific patterns composed of the given attributes, e. g., for detecting communities [8, 30, 57, 69, 72, 75].

In this paper, an adapted and substantially extended revision of [15], we present *MinerLSD* a method for the efficient mining of local patterns on attributed networks. Compared to our work described in [15], we have added onto the discussion of the *MinerLSD* algorithm, also considering further related approaches for putting the proposed method into context. Furthermore, we have considerably extended the evaluation and discussion of the proposed novel algorithm with new experiments, also using new (larger) datasets, and by illustrating the pattern mining approach using exemplary patterns.

MinerLSD focuses both on local pattern mining (e. g., for local community detection) using the local modularity metric [8, 58, 60], as well as graph abstraction that reduces graphs to k-core subgraphs [75]. In order to prevent the typical pattern explosion in pattern mining, we employ closed patterns. In addition, we exploit optimistic estimates for the local modularity for focussing pattern exploration inspired by community detection methods and for pruning the pattern space. Essentially, the optimistic estimate technique provides two advantages: First, it neglects the importance of a minimal support threshold which is typically applied in pattern mining. Second, it enables a very efficient pattern exploration approach, given a suitable threshold for the local modularity, as we will show below. Then, this threshold can of course alternatively be entirely eliminated in a top-k approach. We demonstrate the efficacy of our presented novel method MinerLSD by performing experiments on several standard datasets, in relation to two baselines for local pattern mining.

Our contributions are summarized as follows:

1. For local pattern mining on attributed graphs, we analyze the impact of generating closed patterns compared to standard pattern mining in terms of the search effort.
2. Using two baseline algorithms, we further investigate the impact of pruning the pattern exploration space using an optimistic estimate of the local modularity measure with different thresholds.
3. Finally, we propose the MinerLSD method for efficient local pattern mining on attributed graphs. MinerLSD relies on closed pattern mining, optimistic estimate pruning, and graph abstraction.

The rest of this paper is organized as follows: Section 2 discusses related work, before Section 3 introduces basic notions and concepts. After that, Section 4 presents the novel MinerLSD method. Next, Section 5 introduces the applied datasets. Sections 6 discusses our experimental results. Finally, Section 7 concludes with a summary and interesting directions for future work.

2 Related Work

The detection of local patterns is a prominent approach in knowledge discovery and data mining, e. g., [41, 55, 56]. Below, we discuss related work in the areas of local pattern mining, closed patterns, graph abstractions, and community detection on attributed graphs.

In particular, the proposed novel *MinerLSD* algorithm builds on methods for those fields. Thus, similar to the approaches discussed below, the proposed MinerLSD approach also utilizes closed patterns, and graph abstractions, i. e., core subgraphs. However, it extends this using optimistic estimate pruning using an interestingness measure adapted from (local) community detection. In Section 6, we perform an extensive evaluation of the impact of closed patterns, optimistic estimates, and core structures on the pattern mining effort.

2.1 Pattern Mining

In general, local pattern mining, e. g., [1, 4, 35, 41, 47, 48, 55, 56] has many flavors, including association rule mining, subgroup discovery, and graph mining. At its core, it considers the support set of any pattern, i. e., the set of objects, often called transactions, in which the pattern occurs. The goal then is to enumerate the set of all patterns that satisfy some constraint. In the case of association rules [1, 35] typically the frequency of a pattern, or the frequency of a contained implication in the pattern, respectively, are considered. Whenever the constraint is anti-monotonic, as the frequency, a top-down search may be efficiently pruned. Still this results in investigating a lot of patterns. In the field of subgroup discovery, more complex constraints formalized in quality (or interestingness) functions have been proposed; here, these do not necessarily fulfill anti-monotonicity. To handle that, optimistic estimates for those quality functions have been proposed [9, 33, 47, 82] in order to efficiently prune the pattern search space. Closed pattern mining (see for instance [66]) reduces the search by considering patterns as equivalent when having the same support set, and generating only *closed patterns*, i. e., a most specific pattern among all equivalent patterns. Efficient enumeration algorithms have been provided, e. g., [22, 78]). Various algorithms and methodologies using closure operators have also been proposed in the domain of formal concept analysis [81], which goes further than the enumeration alone, being interested in the lattice structure of the set of closed patterns [31].

2.2 Local Pattern Mining on Attributed Networks

For investigating complex networks, a popular approach consists of extracting a *core subgraph* from the network, i. e., some essential part of the graph whose nodes satisfy a local property. The k-core definition was first proposed in [70]. It requires all nodes in the core subgraph to have a degree of at least k. The idea was further extended to a wide class of so-called generalized cores [18]. The resulting subgraphs may be made of several connected components that are then considered as structural communities. However, as this may be too weak to obtain cohesive communities, some post-processing may then be necessary. A successful method, for example, identifies k-communities [64] that are extracted from the connected components of a graph derived from the original graph.

Recently an extension of the *closed pattern mining* methodology to attributed graphs has been proposed. It relies on the reduction of the support set of a pattern to the core of the pattern subgraph [74]. This results in less and larger classes of equivalent patterns, and hence less closed patterns. The MinerLC algorithm proposed by Soldano et al. [76] is a generic method to enumerate the set of such core closed patterns. The algorithm MinerLSD that we propose in Section 4, closely follows the MinerLC algorithm and adds requirements regarding the local modularity of the pattern core subgraphs. This is performed efficiently using the optimistic estimate pruning strategy of the COMODO algorithm for community detection, mentioned in Section 2.3.

2.3 Community Detection on Attributed Graphs

Communities and cohesive subgroups have been extensively studied in network science, e. g., using social network analysis methods [80]. Fortunato [27] presents a thorough survey on the state of the art community detection algorithms in graphs, focussing on detecting disjoint communities, e. g., [28, 58]. In contrast to such partitioning approaches, overlapping communities allow an extended modeling of actor–actor relations in social networks: Nodes of a corresponding graph can then participate in multiple communities, e. g., [44,65,84]. A comprehensive survey on algorithms for overlapping community detection is provided in [83]. In contrast to the algorithms and approaches discussed above, the proposed approach utilizes further descriptive information of attributed graphs, e. g., [23].

Attributed (or labeled) graphs as richer graph representations enable approaches that specifically exploit the descriptive information of the labels assigned to nodes and/or edges of the graph. Exemplary approaches include density-based methods, e. g., [25, 88], distance-based methods, e. g., [32, 77], entropy-based methods, e. g., [73,89], model-based methods, e. g., [16,85], seed-centric methods, e. g., [20, 36, 37, 86] and finally pattern mining approaches, which we will describe in the following in more detail.

Pattern mining approaches for community detection on attributed graphs typically connect (local) pattern mining and community detection according to several interestingness measures or optimization criteria. Moser et al. [57], for example, combine the concepts of dense subgraphs and subspace clusters for mining cohesive patterns. Starting with quasi-cliques, those are expanded until constraints regarding the description or the graph structure are violated. Similarly, Günnemann et al. [34] combine subspace clustering and dense subgraph mining, also interleaving quasi-clique and subspace construction. Galbrun et al. [30] propose an approach for the problem of finding overlapping communities in graphs and social networks, that aims to detect the top-k communities so that the total edge density over all k communities is maximized. This is also related to a maximum coverage problem for the whole graph. For labeled graphs, each community is required to be described by a set of labels. The algorithmic variants proposed by Galbrun et al. apply a greedy strategy for detecting dense subgroups, and restrict the resulting set of communities, such that each edge can belong to at most one community. This partitioning involves a global approach on the community quality, in contrast to our local approach. Silva et al. [72] study the correlation between attribute sets and the occurrence of dense subgraphs in large attributed graphs. The proposed method considers frequent attribute sets using an adapted frequent item mining technique, and identifies the top-k dense subgraphs induced by a particular attribute set, called structural correlation patterns. The DCM method presented by Pool et al. [69] includes a two-step process of community detection and community description. A heuristic approach is applied for discovering the top-k communities, utilizing a special interestingness function which is based on counting outgoing edges of a community similar; for that, they also demonstrate the trend of a correlation with the Modularity function.

The COMODO algorithm proposed by Atzmueller et al. [8] applies an adapted subgroup discovery [4, 14] approach for community detection on attributed graphs. That is, COMODO applies subgroup discovery for detecting interesting patterns (constructed from the set of compositional attributes) for which their interestingness is evaluated on the graph topological structure. The algorithm works on an edge dataset that is attributed with common attributes of the respective nodes. Then, communities are detected in a top-k approach maximizing a given community interestingness measure. This includes, among others, the local modularity, which is derived from the (global) measure, i. e., the (Newman) Modularity [58, 60]. For an efficient community detection approach, COMODO utilizes optimistic estimate pruning.

In this paper, we adapt the COMODO approach integrating optimistic estimate pruning for the local modularity as proposed by COMODO with closed abstract pattern mining of the MinerLC algorithm. This results in the efficient and effective MinerLSD algorithm, making use of efficient techniques based on abstract closed pattern mining and branch-and-bound pruning according to the local modularity. At the same time, these techniques allow effective selection strategies utilizing graph abstractions together with local modularity, as we will show below.

3 Background

In the following, we outline the background on closed local pattern mining, introduce pruning based on optimistic estimates, and discuss pattern exploration, abstraction, and selection combining principles from pattern mining and graph mining, i. e., utilizing closure on the attribute space and topological criteria based on local modularity (estimates) and k-cores.

3.1 Mining Closed Patterns to Enumerate Core Subgraphs

We consider the following general problem: Let G be an attributed graph, i. e., a graph where each vertex v is described by an itemset $D(v)$ taken from a set of items I. We want to enumerate all (maximal) vertex subsets W in G such that there exists an itemset q which is a subset of all itemsets $D(v), v \in W$. W is furthermore required to satisfy some graph related constraints. In standard terminology, q is a *pattern* that *occurs* in all element of W which is also called the *support set* or *extension* $\text{ext}(q)$ of q. Efficient top-down enumeration algorithms exist as far as the constraints are anti-monotonic: whenever the constraint fails to be satisfied by some pattern, it also fails for all more specific patterns. This is obviously the case for the *minimum support* constraint that requires the size of $\text{ext}(q)$ to be above some minimal support threshold s.

A first way to reduce the overall search space and the size of the solution set is to avoid duplicates, i. e., patterns q, q' that occur in the same subgroup, for which $\text{ext}(q) = \text{ext}(q')$. This is obtained by only enumerating *closed patterns*. Given any pattern q the associated closed pattern is the most specific pattern $f(q)$ which occurs in the same subgroup as q, i. e., $\text{ext}(f(q)) = \text{ext}(q)$. Furthermore, since we consider the vertices of a graph, it is natural to consider graph related constraints, as for instance requiring that all vertices have a degree of at least k in the subgroup graph G_W. For that purpose, each candidate subgroup X is reduced to its core $p(X) = W$ using the *core operator* p.

We start with the definition of closure: The operator f that returns for any pattern q the closed pattern $f(q)$ is a *closure operator* (see below) defined by $f(q) = \text{int} \circ p \circ \text{ext}(q)$; the respective operators are defined as follows (note that $\circ$ denotes function composition):

– The intersection operator $\text{int}(X)$ returns the most specific pattern occurring in the vertex subset X.
– The core operator $p(X)$ returns the core, according to some core definition, of the subgraph G_X of G induced by the vertex subset X. p is an *interior operator* (see below).

Definition 1 *Let S be an ordered set and $f : S \to S$ a self map such that for any $x, y \in S$, f is monotone, i.e. $x \leq y$ implies $f(x) \leq f(y)$ and idempotent, i.e. $f(f(x)) = f(x)$:*
- If $f(x) \geq x$, f is called a closure operator.
- If $f(x) \leq x$, f is called an interior operator.

Essentially, core closed pattern mining relies on three main results:

1. It has been shown that whenever p is an interior operator, $f = \mathrm{int} \circ p \circ \mathrm{ext}$ is a closure operator [68].
2. Furthermore, core definitions rely on a monotone property of a vertex within an induced subgraph [17]. For instance, the k-core of a subgraph G_X is defined as the largest vertex subset $W \subseteq X$ such that in the induced subgraph G_W all vertices v have a degree of at least k. The property is monotone in the sense that when increasing G_X to $G_{X'}$ the degree of v cannot decrease.
3. Finally, it has been shown that the core operator which returns the core of some subgraph G_X, according to a monotone property, is an interior operator [74].

Overall, this means that $f(q)$ returns the largest pattern which occurs in the core of the vertex subset $\mathrm{ext}(q)$ in which q occurs. This is exploited in core closed pattern mining [76], performing a top-down search of the pattern space jumping from closed pattern to closed pattern: each closed pattern q is augmented with some item x, then the next closed pattern $f(q \cup \{x\})$ is computed.

3.2 Pruning Local Patterns in Graphs Using Optimistic Estimates

Another way to reduce the solution set is to consider some interestingness measure M and require a subgroup W to induce a subgraph G_W with an interestingness $M(W)$ above some threshold. However such measures, for example, the local modularity (see below), are usually not anti-monotonic. This difficulty may be overcome by using some optimistic estimate of M which is both anti-monotonic and allows an efficient pruning of the search space. Optimistic estimates are one prominent option in local pattern mining to prune search spaces by complementing non-(anti)-monotonic interestingness measures by their respective optimistic estimators, e. g., [33, 82]. Intuitively, if for a given pattern (and all of its potential specializations) it can be proven that their quality is either below the quality of the current top patterns, or below a specified threshold, then pattern exploration does not need to continue for that pattern, and the search space can often be pruned significantly.

In the scope of local pattern mining on graphs, several standard community quality functions have been investigated, also specifying optimistic estimates for a number of such community evaluation functions. As shown in [8] these lead to a quite efficient approach for descriptive community detection using local pattern mining. In summary, using optimistic estimates we can enumerate pairs (c, W), of pattern c and subgroup W inducing the subgraph G_W. Then, we can select subgraphs according to an interestingness measure M of the subgraph using an anti-monotonic optimistic estimate of M to prune the search. Additionally, a minimal support constraint can also be applied in order to improve the effectiveness of pruning.

Below, we summarize main results on using optimistic estimate pruning for community detection, specifically addressing the (local) modularity quality measure. Here, the concept of a *community* intuitively describes a group W of individuals out of a population such that members of W are strongly "connected" to each other but sparsely "connected" to those individuals that are not contained in W. This notion translates to communities as vertex sets $W \subseteq V$ of an undirected graph $G = (V, E)$; in the following, we adopt the notation of [8] for introducing the main concepts: $n := |V|$, $m := |E|$, and $m_W := |\,\{\{u,v\} \in E \,:\, u,v \in W\}\,|$ denotes the number of *intra-edges* of W.

There are different interestingness measures for estimating the quality of a community $2^V \to \mathbb{R}$, also according to different criteria and intuitions about what "makes up" a good community. One particular community quality function is the Modularity [58,60]. In the context of local pattern mining, we aim to *maximize* local quality functions for single communities. For that, we apply an adaptation of the Modularity interestingness measure, which essentially is a global measure estimating the quality of a community partitioning. Then, we focus on the *modularity contribution* of each individual community in order to obtain a local measure for each community, cf., [8], which we further call *local modularity* (MODL).

Overall, the *Modularity* MOD [58,60,61] of a graph clustering with k communities $C_1, \ldots, C_k \subseteq V$ focuses on the number of edges *within* a community and compares that with the *expected* such number given a null-model (i.e., a corresponding random graph where the node degrees of G are preserved). It is given by

$$\mathrm{MOD} = \frac{1}{2m} \sum_{u,v \in V} \left(A_{u,v} - \frac{\mathrm{d}(u)\,\mathrm{d}(v)}{2m} \right) \delta(C(u), C(v)) \,, \tag{1}$$

where $C(i)$ denotes for $i \in V$ the community to which node i belongs. $A_{u,v}$ denotes the respective entry of the adjacency matrix A. $\delta(C(u), C(v))$ is the *Kronecker delta* symbol that equals 1 if $C(u) = C(v)$, and 0 otherwise.

The *modularity contribution* of a single community given by a vertex set $W, W \subseteq V$ in a *local context* (e.g., in a subgraph induced by the pattern), i.e., the *local modularity* (MODL), can then be computed (cf., [8,61,63]) as follows:

$$\mathrm{MODL}(W) = \frac{m_W}{m} - \sum_{u,v \in W} \frac{\mathrm{d}(u)\,\mathrm{d}(v)}{4m^2} \,. \tag{2}$$

For the above (MODL), an optimistic estimate has been introduced in [8]. It can be derived based only on the number of edges m_W within the community:

$$\mathrm{oe}(\mathrm{MODL})(W) = \begin{cases} 0.25, & \text{if } m_W \geq \frac{m}{2}\,, \\ \frac{m_W}{m} - \frac{m_W^2}{m^2}\,, & \text{otherwise.} \end{cases} \tag{3}$$

For a detailed discussion, the derivation of the local measure, and the respective proofs, we refer to [8].

3.3 Local Pattern Exploration, Abstraction, and Selection

Pattern mining commonly aims at discovering a set of *novel*, *potentially useful*, and ultimately *interesting* patterns from a given (large) data set [26]. For pattern exploration, we apply local pattern mining, in particular, (abstract) closed pattern mining [22, 66, 74, 76, 78] due to its efficient traversal of the search space for pattern enumeration and abstraction as discussed above.

Regarding pattern selection, we discuss the choices of core abstraction and modularity-based selection in the following: In contrast to many methods used in network analysis and graph mining, pattern mining on attributed graphs specifically aims at a description-oriented view, by including patterns on attributes, but also considering the topological structure. Many community mining algorithms, for example, only collect sets of nodes denoting the individual communities thus merely focusing on structural/topological aspects of the graph; typically, then there is no simple and easily interpretable description, such that a community would be represented mainly as a set of IDs, cf., [8].

For local pattern mining, the goal is typically to detect a set of the most interesting patterns according to a given quality function, e. g., with a quality above a certain threshold, or the top-k patterns according to the ranking of the quality function denoting their interestingness. For subgroup discovery, as an exemplary instance, the goal is then to obtain the set of patterns covering subgroups that are "as large as possible and have the most unusual statistical characteristic with respect to the property of interest" [82]. Thus, the interestingness of a pattern can then be flexibly defined, e. g., by a significant deviation from a model that is derived from the total population [41, 55, 56]. Therefore, typically the size of a pattern or the size of its extension, respectively, and the deviation compared to some null-model specifies the interestingness which is formalized in the quality function for ranking the patterns.

For pattern mining on networks and graphs, there exist several quality measures, usually taking into account the *support* of the pattern, i. e., its size, similar to the criteria discussed above. Furthermore, the topological structure of the subgraph induced by the pattern is also taken into account. Here, standard quality functions include the *segregation index* [29], the *average out degree fraction* [87], the *conductance* [50] and the *Modularity* [58], as we have discussed in the previous section. In general, the core idea of the evaluation function is to apply an objective evaluation criterion, for example, for the Modularity the number of connections within the community compared to the statistically "expected" number based on all available connections in the network, and to prefer those communities that optimize the evaluation function.

A thorough empirical analysis of the impact of different community mining algorithms and their corresponding objective function on the resulting community structures is presented in [51], based on the analysis of community structure in graphs (as presented in [50]). Furthermore, Atzmueller et al. [8, 11, 12] have empirically investigated different community quality functions in the scope of local pattern mining. As shown there for the provided experiments, the local modularity quality function indicated the best results for pattern filtering and pruning in local pattern mining applications, since it provides large high quality communities, i. e., subgroups referring to the induced subgraphs, smaller patterns in terms of their description, as well as statistically significant patterns compared to the other mentioned quality functions which focus on smaller subgroups; those were typically also not statistically significant as specifically presented in [8].

Furthermore, the local modularity quality function (see Equation 2) intuitively provides the prominent property of assigning a higher ranking to larger (core) subgraphs under consideration, if these are considerably more densely connected than expected by chance. Therefore, these criteria conveniently capture the notion of larger subgraphs and having the most unusual statistical characteristics with respect to the null-model. In the following, we show how these criteria are directly implemented in the local modularity measure.

Consider the local modularity $\mathrm{MODL}(W)$ of a subgraph W:

$$\mathrm{MODL}(W) = \frac{m_W}{m} - \sum_{u,v \in W} \frac{\mathrm{d}(u)\,\mathrm{d}(v)}{4m^2} = \frac{1}{m}\left(m_W - \sum_{u,v \in W} \frac{\mathrm{d}(u)\,\mathrm{d}(v)}{4m}\right) .$$

Since the first factor $\frac{1}{m}$ is a constant, we can consider the second factor of the former expression: It is easy to see that this factor itself is order equivalent to the local modularity function MODL, since it only depends on a fixed constant $\frac{1}{m}$; by not including that it is thus not normalized relatively to the number of edges of the graph. Instead, it focuses on the number of edges of the (core) subgraph (the minuend of the term) and its deviation assessed by the null-model which is captured by the subtrahend of that term.

Thus, it is easy to see that the MODL function tends to focus on larger patterns (larger subgraphs) having the most unusual statistical characteristics with respect to the null-model. By utilizing appropriate constraints on the graph structure, e. g., using k-core abstractions we can further focus on the unusual distributional characteristics. By applying $k-$core abstractions, for example, with increasing k we tend to focus on increasingly denser pattern structures (subgraphs). We will also show this by our experiments in Section 6 when we discuss our results.

To sum up, we apply the local modularity measure MODL as introduced above for focusing pattern exploration on the statistically most unusual subgraphs. Applying k-core constraints helps due to its focus on denser subgraphs, as also theoretically analyzed in [67] for k-cores. Overall, we specifically focus on "nuggets in the data" [40], i. e., on exceptional patterns according to the principles of local pattern mining. In addition, the local modularity neglects the importance of a minimal support threshold which is typically applied in pattern mining, since it directly includes the size of the pattern as a criterion. This enables a very efficient pattern mining approach, given either a suitable threshold for the local modularity, or by targeting the top-k patterns.

4 The MinerLSD Algorithm

In the following, we describe our proposed novel method *MinerLSD* in detail. MinerLSD integrates core subgraph closed pattern mining with pattern selection according to the local modularity MODL function, and optimistic estimate pruning according to a specific optimistic estimator, i. e., oe(MODL).

As input parameters, MinerLSD requires a graph $G = (V, E)$, a set of items I, a dataset D describing vertices as itemsets and a core operator p. p depends on G and to any image $p(X) = W$ we associate the core subgraph C whose vertex set is $\mathrm{vs}(C) = W$. In our experiments, $p(X)$ returns the k-core of X. As further parameters, MinerLSD considers the corresponding value k as well as a frequency threshold s (defaulting to 0) and a local modularity threshold lm. The algorithm outputs the frequent pairs (c, W) where c is a core closed pattern and $W = p \circ \mathrm{ext}(c)$ its associated k-core. For evaluation purposes, we also count the number of patterns above the local modularity threshold (#lm), and the number of patterns for which their estimate is above the local modularity threshold (#lme). It is important to note, that in the enumeration step MinerLSD ensures that each pair (c, W) is enumerated (at most) once.

```
MinerLSD (G, I, D, p, s, lm)
    #lme← #lm← 0
    $W \leftarrow p(V)$
    // also defines the associated core subgraph $C = G_W$
    if $\mid W \mid < s$ or oe(MODL)($W$) $< lm$ then exit
    enum(int($W$), $C$, $\emptyset$) // int($W$) is the closure of $\emptyset$
-
Function enum($c, C, EL$)
ensure: outputs the frequent $(c', W')$ pairs
where $c' \supseteq c$ and contains no items of $EL$
    Increase #lme
    if MODL($C$) $\geq lm$ then
        Increase #lm and Output $(c, \mathrm{vs}(C))$
    end if
    for all $x \in I \setminus c$ do
        /* Generate all augmentations of c*/
        $W = p \circ ext(c \cup \{x\})$ // with core subgraph $C^x$
        $c \leftarrow \mathrm{int}(W)$
        if $\mid W \mid \geq s$ and oe(MODL)($W$) $\geq lm$ and $c \cap \mathrm{EL} = \emptyset$ then
            enum($c, C^x, EL$)
            // enumerate the subtree rooted on c
            EL ← EL $\cup \{x\}$
        end if
    end for
-
Function int($W$)
    return $\cap_{v \in W} D(v)$
```

Table 1 Datasets/Characteristics: Number of edges ($|E|$), vertices ($|V|$), labels ($|L|$), the average vertex degree ($\overline{deg(v)}$), and average number of labels per vertex ($\overline{|l(v)|}$)

Nom	$\boldsymbol{\lvert V\rvert}$	$\boldsymbol{\lvert E\rvert}$	$\boldsymbol{\lvert L\rvert}$	$\overline{\boldsymbol{deg(v)}}$	$\overline{\boldsymbol{\lvert l(v)\rvert}}$
S50	50	74	14	2.96	7
Lawyers	71	556	42	15.66	20
CoExp	151	1849	36	24.49	18
LastFM	1892	12717	17625	13.44	40.07
Delicious	1867	7664	52800	8.21	123.47
DBLP.C	3140	10689	4588	6.81	15.02
DBLP.P	45131	228173	32	10.11	2.15
DBLP.S	108032	276658	23254	5.12	13.93
DBLP.XL	929937	3461697	92164	7.44	10.16

5 Datasets

We performed our experiments utilizing a variety of attributed graph datasets ranging from small to medium graphs with small to large sets of items. Table 1 depicts the main characteristics of these datasets (see also [30]), which have been previously used in pattern mining tasks on attributed graphs. For each dataset, we indicate the number of edges ($|E|$), vertices ($|V|$) and labels ($|L|$), the average vertex degree ($\overline{deg(v)}$) and average number of labels per vertex ($\overline{|l(v)|}$) in the table.

- S50 is a standard attributed graph dataset[1] used in a previous work about graph abstractions [74]. It represents 148 friendship relations between 50 pupils of a school in the West of Scotland; the labels concern the students' substance use (tobacco, cannabis and alcohol) and sporting activity. The values of the corresponding variables are ordered (see [74] for details).
- The Lawyers dataset concerns a network study of corporate law partnership that was carried out in a Northeastern US corporate law firm from 1988 to 1991 in New England [46]. It concerns 71 attorneys (partners and associates) of this firm who are the vertices of four networks. In the resulting data[2], each attorney is described using various attributes. We consider the advice network which is originally a directed graph in a undirected version, so that two lawyers are connected if at least one asks for advice to the other one.
- The CoExp dataset models a representative regulatory network for yeast obtained from Microarray expression data processed by the CoRegNet [62] program. In the CoExp dataset the vertices are co-regulators and they are linked if they share a common set of target genes. The vertices are labeled

[1] Available at:
`http://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm`

[2] Available at:
`https://www.stats.ox.ac.uk/~snijders/siena/Lazega_lawyers_data.htm`

with their influence profile along a metabolic transition of the organism. Each influence value represents the regulation activity of the considered co-regulator at some instant of the metabolic transition.

– LastFM, DBLP.C and DBLP.XL were used in Galbrun et al. [30]. LastFM models the social network of last.fm where individuals are described by the artists or groups they have listened to. DBLP.C contains a co-authorship graph built from a set of publication references extract from DBLP of researchers that have published in the ICDM conference. The authors are labeled by keywords extracted from the papers' titles. DBLP.XL is the complete labeled DBLP co-authorship network used in [30].
– DBLP.P was used in Bechara-Prado et al. [19]. It represents a co-authorship graph built from a set of publication references extract from DBLP, published between January 1990 and February 2011 in the major conferences or journals of the Data Mining and Database communities. Three labels have been added to the original dataset based on the scope of the conferences and journals, respectively: DB (databases), DM (data mining) and AI (artificial intelligence).
– Delicious consists of the social (friendship) network of the resource sharing system delicious where individuals are described by their bookmarks' tags. The dataset is publicly available and was obtained from the HetRec workshop [24] at Recsys 2011.
– DBLP.S was used in Silva et al. [71]. It also represents a co-authorship network from a set of publication references extracted from DBLP.

6 Experiments and Results

In the following, we first summarize the applied baseline methods that were used in the comparison with the presented MinerLSD method. After that, we present our experimental results on the datasets described in Section 5.

6.1 Baseline Methods

The applied set of baseline methods consists of MinerLC – an efficient algorithm for mining core closed patterns, and COMODO– an efficient algorithm for descriptive community detection using optimistic estimates.

6.1.1 MinerLC

MinerLC[3] (cf., [76]) enumerates pairs (c, W) where G_W is the core subgraph of pattern c, i. e., subgroup $W = p \circ \text{ext}(c)$ where $\circ$ is the composition operator, p is a *core operator* and c is the largest pattern that occurs in W and is called a *core closed pattern.* A threshold on the core sizes allows to select frequent

[2] `https://grouplens.org/datasets/hetrec-2011/`
[3] `https://lipn.univ-paris13.fr/MinerLC/`

core closed patterns and to accordingly prune the search. The selection process relies then partly on the anti-monotonic support constraint and partly on the fact that there are less pattern core subgraphs than pattern subgraphs as various pattern subgraphs $G_{\mathrm{ext(q)}}$ may be reduced to the same core subgraph.

6.1.2 COMODO

The COMODO algorithm[4] presented in [8] performs *description-oriented community detection* in order to discover the top-k communities. In summary, COMODO enumerates pairs (c, W) where G_W is the subgraph of pattern c for vertex subset W. It selects top k subgraphs according to an interestingness measure M of the subgraph and uses an efficient anti-monotonic optimistic estimate of M to prune the search. Additionally, a minimal support constraint can also be applied in order to improve the effectiveness of pruning.

6.1.3 Similarities and Differences in Pattern Selection

Both the considered baseline methods, i. e., MinerLC and COMODO output a set of pairs (pattern, vertex subset). However, in order to compare their outputs we have to consider the following differences:

- In COMODO the vertex subset W is obtained as the extremities of the set of edges in which a pattern occurs and a pattern occurs in an edge whenever it occurs, in the original dataset, in both connected vertices. That is, for each edge we assign the set of common items of both nodes, such that a pattern always covers two nodes connected by an edge. As a consequence, W ignores isolated nodes in which p occurs. To obtain the same vertex subset in MinerLC (and MinerLSD) it is necessary to remove isolated nodes, which is enabled by applying a 1-core graph abstraction.
- Since COMODO does not enumerate closed patterns, the same subgroup may be associated to several patterns. For that case, a post-processing is needed to eliminate the duplicates from the list of subgroups which may then be compared to the subgroups in the MinerLC pairs. This post-processing is one of the standard post-processing options of COMODO.
- MinerLC is run with a core definition while COMODO uses various parameters to limit the enumeration, as for instance the *top-k* parameter.

To compare the results, MinerLC (as well as MinerLSD) should be run with the same minimum support threshold as COMODO and should only use a 1-core abstraction. The other parameters of COMODO should then have a value that does not limit the enumeration, e. g., by providing a sufficiently large top-k parameter to enable an exhaustive enumeration.

Furthermore, MinerLC and COMODO select patterns according to different criteria. This is exemplified in Figure 1, in which we have three graphs and three subgraphs induced by three vertices (in red). The subgraph G_{123} of the

[4] `http://www.vikamine.org` [10]

top graph G is a 2-core with a local modularity of 0.178. Within the central graph, the subgraph G_{123} is also a 2-core but with a low local modularity of -0.15. Finally, within the bottom graph, G_{123} is not a 2-core (since it has an empty 2-core subgraph) with a high local modularity of 0.16.

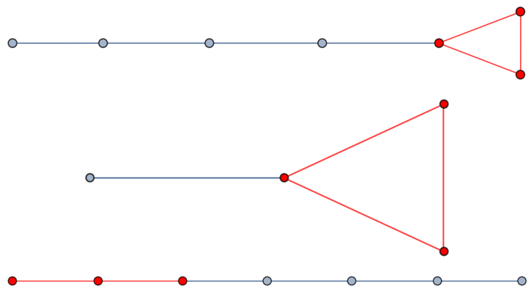

Fig. 1 Three graphs (top, center, bottom) each with a subgraph displayed in red. The two topmost subgraphs are 2-cores while the bottom subgraph has an empty 2-core. The top and bottom graphs have a local modularity above 0.15 while the central one has a negative local modularity score of -0.15.

6.2 Results and Discussion

In our experiments below, we first investigate the impact of closure, before we focus on the k-core abstraction. We perform a detailed analysis of the efficiency of using the local modularity estimate for pruning the search space. Finally, we provide a structural pattern set analysis considering different metrics, and discuss exemplary patterns for illustrating the efficacy of the proposed approach.

6.2.1 Parameters and Datasets

For MinerLSD, it is important to note that in our experiments described below we did not have to use the minimal support s, since the local modularity threshold is efficient enough to strongly reduce the number of patterns.

Below, we consider the following pattern quantities, where the (closed pattern, support set) pairs (c, e) are output by MinerLC unless specified; also, we consider a given local modularity threshold lm.

- #c the number of pairs (c, e).
- #lme: the number of pairs (c, e) such that $\text{oe}(\text{MODL})(e) \geq lm$.
- #nec: the number of (*necessary*) pairs (c, e) a top-down search has to consider to ensure that no pair with $\text{oe}(\text{MODL})(e) \geq lm$ is lost. See Section 6.2.5 for details and results on #nec.
- #lm the number of pairs (c, e) such that $\text{MODL}(e) \geq lm$
- #lmeSD: the number of pairs (c, e) such that $\text{oe}(\text{MODL})(e) \geq lm$ as generated by COMODO.

We ran the original COMODO and MinerLC programs as available. MinerLSD is derived from the sources of MinerLC and is to be found on the MinerLC web site[5]. A new MinerLC version integrates the MinerLSD developments. The experimental results presented here may then be obtained using appropriate parameters and options of the new software.

6.2.2 Impact of Closed Patterns in Reducing the Search Space

MinerLSD searches a space of closed patterns while COMODO searches the whole pattern space. Therefore, we will investigate the impact of the closure reduction, for each local modularity threshold *lm*. For that, we first consider the quantity #lme of core closed patterns with a local modularity estimate above *lm*, as provided by MinerLSD, when using 1-cores. We consider then the quantity #lmeSD of patterns developed by COMODO using the same threshold. Table 2 reports #lme and #lmeSD for our datasets under investigation.

We observe two very different situations. In the Lawyers and CoExp datasets there is a large difference between #lmeSD and #lme, while there are considerable but not so strongly expressed differences in the other datasets compared to the former. Large differences typically occur when items have strong dependencies hence leading to a large reduction of the search space when applying a closure operator. For instance, in the Lawyers dataset vertices are described by various numeric attributes. In our representation, a single numeric attribute x leads to a set of $x \leq s_i$ and of $x > s_i$ items with various thresholds s_i. This allows to include interval constraint as $x \in]s_j, s_k]$ within patterns. However there are then several equivalent patterns in which the same interval is represented in various ways. For instance, consider 4 thresholds $s_1, \ldots, s_4$, the interval $x \in]s_2 s_3]$ is represented by $x > s_2, x \leq s_3$, $x > s_1, x > s_2, x \leq s_3$ and $x > s_1, x > s_2, x \leq s_3, x \leq s_4$. The latter is the only one found in a closed pattern. COMODO has then to generate many equivalent patterns while MinerLC, which applies a closure operator at each specialization step never generates two equivalent patterns, thus reducing the exploration of the pattern space effectively.

In the DBLP.P datasets at the contrary the items are tags, with no taxonomic order relating them. Therefore, the values of #lme and #lmeSD are much closer, and even identical regarding the DBLP.C dataset.

6.2.3 k-core sizes of the various networks

Before considering how reducing support sets to k-cores affects the number of closed patterns in each dataset, we consider the various networks and compute their k-core sizes for a range of values of k. This pre-analysis aims to evaluate which level of k we should use in our experiments. For small datasets for which computing closed patterns does need much resources this is not that important. However, for large datasets with many attributes, i. e., potentially

[5] `https://lipn.univ-paris13.fr/MinerLC/`

Table 2 Number of patterns to develop in MinerLSD and COMODO (according to the respective local modularity threshold 0.005 ... 0.15) using a 1-core abstraction for MinerLSD.

Data / #c	0.005	0.01	0.02	0.03	0.05	0.15
S50	83					
#lmeSD	493	493	357	326	259	83
#lme	83	83	77	72	67	36
CoExp	196					
#lmeSD	1232895	991231	806911	468991	285183	77823
#lme	178	166	150	133	114	64
DBLP.P	2396					
#lmeSD	148	32	18	9	5	3
#lme	34	22	15	9	5	3
Lawyers	3221					
#lmeSD	3021675	1535949	677089	420699	168689	10339
#lme	2929	2512	1970	1640	1146	295
DBLP.C	14820					
#lmeSD	179	66	24	16	7	1
#lme	179	66	24	16	7	1

large numbers of closed patterns, it is much better to have a rough guideline for selecting appropriate parameters for optimizing the computational effort.

In Figure 2 we display the k-core sizes for a range of values of k, for each dataset. As we will see below, the small but densest networks for which local-modularity-based pruning has a weak efficiency, namely coExp and Lawyers, also exhibit a (relatively) slow decay with respect to increasing k values, whereas for the other (larger) datasets we observe a quite considerable decrease in terms of the k-core sizes.

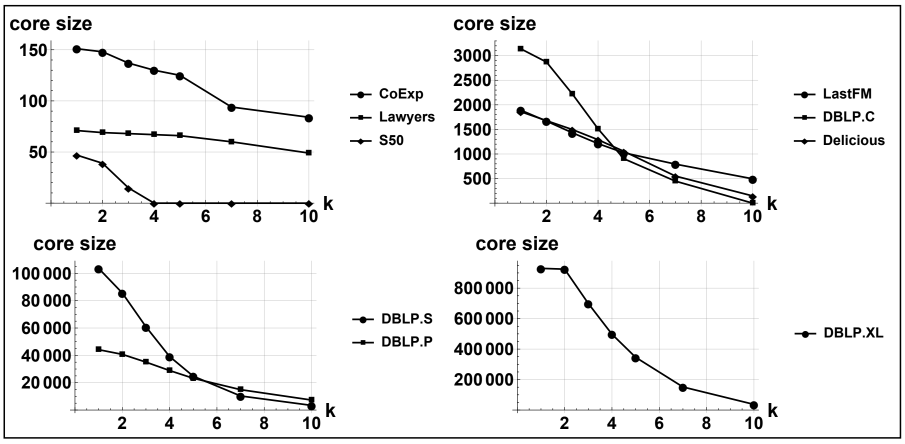

Fig. 2 k-core sizes of the networks associated to our datasets *versus* k.

6.2.4 Modularity Distributions

As a prerequisite for the further analysis of the local modularity optimistic estimate, we aimed to get a more detailed insight into the distribution, similar to our pre-analysis for the k-cores discussed above. Figures 3-4 show the detailed results. The plots indicate the "meaningful" values for estimating the local modularity thresholds, which support our selections of parameters in the subsequent evaluations. Furthermore, Figure 3 also indicates the pruning potential of the local modularity threshold, even using our rather approximating sampling strategy.

6.2.5 Pruning: Efficiency of the Local Modularity Estimate

For investigating the efficiency of pruning using the modularity estimate, we compare our proposed algorithm MinerLSD to the MinerLC algorithm, which applies no optimistic estimate pruning. For the other baseline, i. e., COMODO we already investigated the efficiency of MinerLSD which showed a considerable reduction in the number of considered patterns, cf., Section 6.2.2. Regarding the number of output patterns, both actually yield the same numbers, if a postprocessing step of COMODO is applied for keeping only the subset of closed patterns (as discussed in Section 6.1.3), i. e., by considering all pairs (c, e) with the same (vertex) subgroup e and only keeping the most specific ones. With this postprocessing COMODO returns exactly the same patterns as those output by MinerLSD in our experiments. However, this approach is quite inefficient, cf., Section 6.2.2, since the number of considered patterns is typically considerably larger for COMODO compared to MinerLSD.

Regarding the modularity estimate, we first investigate how the local modularity constraint affects the number of output pairs. In general, as oe(MODL) is an optimistic estimator, we may consider the best possible optimistic estimator which would only develop the $\#nec$ nodes that have at least a descendant (c, e) with local modularity MODL$(e) \geq lm$. We have then $\#lm \leq \#nec \leq \#lme$. Whenever $\#lm$ is far from $\#nec$ this means that there does not exist any good optimistic estimator. Whenever $\#lm$ is close to $\#nec$ which in turn is far from $\#lme$ this means that there could be some optimistic estimator that is much better than oe(MODL). By computing these numbers, we can then state separately for each dataset whether the oe(MODL) estimate is efficient in pruning the search with respect to the best possible estimator nec and whether nec would be efficient in pruning the search, if such an estimator would be found.

Small Datasets In a first step, we first considered several rather small datasets using no minimal support parameters, and a 1-core abstraction in MinerLSD aiming to provide a comparable setting for COMODO. We also checked the number of patterns retrieved by COMODO with additional postprocessing as discussed above - only keeping the closed patterns. We used parameters that do not limit the enumeration in COMODO, i. e., for an exhaustive search only

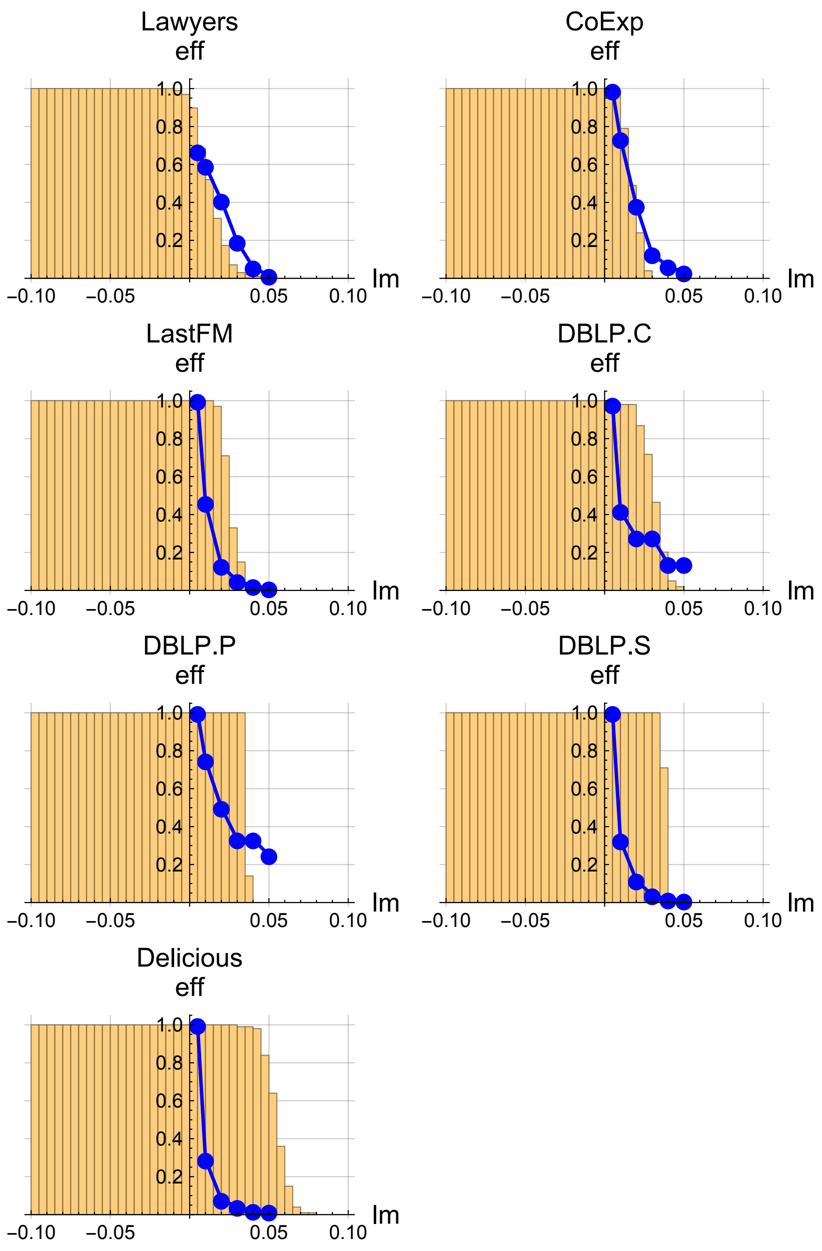

Fig. 3 Detailed Estimated/Observed Modularity Distributions: We consider the unlabeled graph of the dataset. We generate 100 random subgraphs of the unlabelled graph picking randomly half of the vertices. For each random graph, we compute the local modularity of the abstract 5-core subgraph and we report the survival distribution of the local modularity over the 100 experiments (*in orange*), i. e., for each local modularity (***lm***) level, the probability of having at least that level in our sample. In blue, we report the (empirically observed) survival distribution of the local modularity, i. e., the respective MODL values of the core subgraphs of the abstract patterns discovered using the 5-core abstraction.

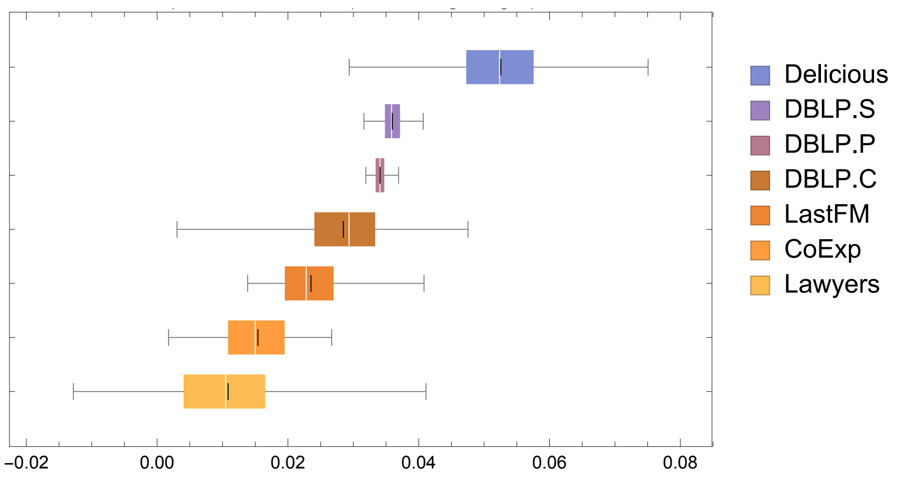

Fig. 4 Distribution of the local modularity of the 5-core abstraction in samples of 100 unlabelled random subgraphs having half of the size (number of vertices) of the original graph.

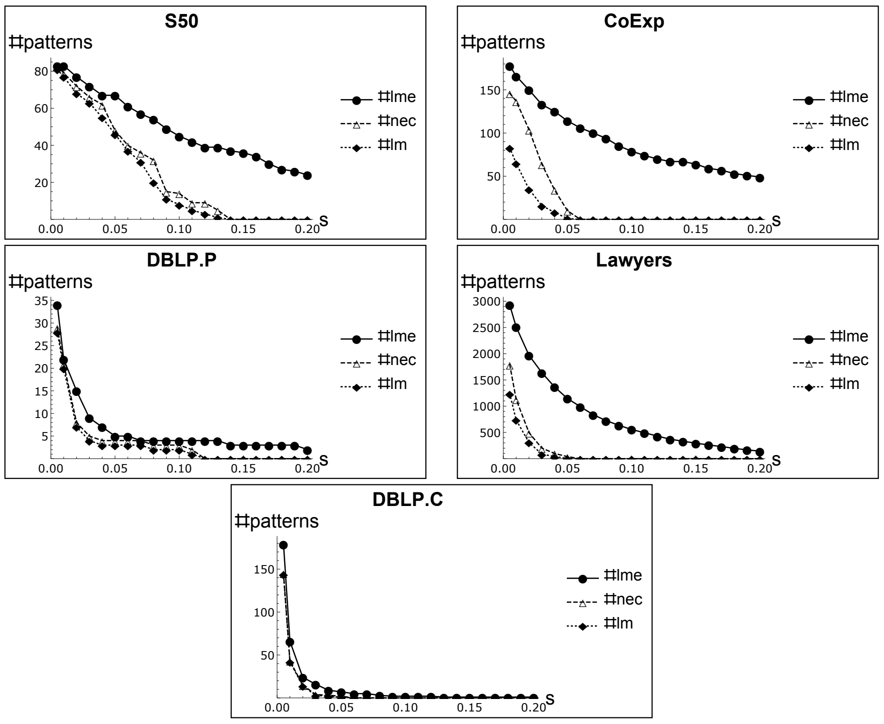

Fig. 5 Numbers of patterns with #lme, #nec and #lm values (on the Y-axis), above the local modularity threshold (on the X-axis) for 5 attributed networks, using a 1-core abstraction.

using the local modularity threshold for pruning. Likewise, for MinerLSD, we select and count vertex subgroups whose induced subgraphs satisfy a local modularity threshold *lm*. In this way, we could confirm (again) that the final number of output patterns is the same for both algorithms, as discussed above.

Table 3 Number of patterns total, developed, necessary and with required local modularity (according to the respective threshold 0.005 ... 0.15) using a 1-core abstraction.

Data / #c	/ 0.005	0.01	0.02	0.03	0.04	0.05	0.15
S50	83						
#lme	83	83	77	72	67	67	36
#nec	83	79	72	66	62	48	0
#lm	81	77	68	63	55	46	0
CoExp	196						
#lme	178	166	150	133	125	114	64
#nec	146	137	104	64	34	10	0
#lm	83	65	35	16	8	1	0
DBLP.P	2396						
#lme	34	22	15	9	7	5	3
#nec	29	21	8	5	4	4	0
#lm	28	20	7	4	3	3	0
Lawyers	3221						
#lme	2929	2512	1970	1640	1365	1146	295
#nec	1792	1131	495	201	99	38	0
#lm	1238	738	308	87	39	5	0
DBLP.C	14820						
#lme	179	66	24	16	9	7	1
#nec	145	43	15	4	3	2	0
#lm	144	42	14	3	2	1	0

Figure 5 depicts the results of the applied five datasets, with the detailed results in Table 3. Overall, the local modularity estimate is efficient in pruning the pattern exploration, on different levels. For instance, in the Lawyers dataset, MinerLSD finds #c=3221 patterns at level *lm*=0.005 and most of them, i. e., 2929, have an oe(MODL) value above 0.005, not too far from the #nec = 1792 patterns any top-down search would have to develop anyway to select the 1238 patterns with local modularity MODL above 0.005. There is then a slow decrease of #lme while the decrease of #nec and #lm is much faster. Yet, pruning does still work, reducing the search effort considerably.

In contrast, for the larger datasets, e. g., for DBLP.P among the #c = 2396 patterns only 34 have a local modularity estimate above 0.005, 29 of them have to be developed and 28 do have a local modularity above 0.005. Furthermore, in the DBLP.C dataset among the #c = 14820 patterns only 179 have a local modularity estimate above 0.005, 145 of them have to be developed and 144 do have a local modularity above 0.005. When the local modularity threshold increases, #lme keeps being close to #lm.

Overall, the Lawyers dataset displays moderate pruning efficiency, still allowing to avoid to develop many nodes, and this is also the case for the S50 and

CoExp datasets. In contrast, DBLP.C and DBLP.P indicate a very efficient optimistic pruning in terms of the numbers of patterns.

Tables 4-5 show the runtime results of MinerLSD for the larger of the small datasets (Lawyers, CoExp, DBLP.C, DBLP.P, runtime in seconds). Here, we observe that MinerLSD is either in the same range or slightly faster than MinerLC for the small datasets, i. e., for Lawyers and CoExp. For DBLP.C, we observe a strongly reduced number of patterns, while the runtimes are always in the same range, especially for stronger (graph-)constraints. Here, we considered k-cores, $k = 1, 2, 3, 5, 7$. Therefore, while strongly reducing the number of patterns the additional computation using the estimate still keeps the runtime of the algorithm in the same range as MinerLC most of the times.

In contrast to the other smaller datasets, for the larger DBLP.P dataset we observe an increase in the runtime of MinerLSD compared to MinerLC. However, this can be explained by some special characteristic of DBLP.P. The DBLP.P dataset contains an extremely limited number of labels (32) which are used in the dataset. Here, the extra effort of the estimation does not help too much in decreasing the runtime, because the enumeration in the label space is extremely fast, and hence the check of the patterns is mainly determined by the core abstraction.

Medium Size Datasets Overall, MinerLSD detects closed patterns with the benefit of pruning using the oe(MODL) $\geq lm$ condition, i. e., only developing the #lme nodes according to Table 2. Furthermore, applying both the k-cores and local modularity constraints makes it possible to find some balance between the k-core and the local modularity constraint to apply when facing large datasets that are difficult to mine. This is investigated on the two datasets LastFM and Delicious, i. e., those with the largest number of closed core patterns when considering the 1-core and no local modularity thresholds – these were not investigated in Tables 2-3, respectively. For these medium sized datasets, we performed experiments using 1-cores, 2-cores, 3-cores, 5-cores and 7-cores with local modularity thresholds 0.01,0.02, 0.03, 0.04, 0.05, and 0.15; the results regarding the number of closed patterns and the total CPU time (including pruning/optimistic estimation) are shown in Figure 6 (runtimes in seconds).

The benefit of applying local modularity constraints in the resulting number of closed patterns is, as expected, quite impressive. When no constraint (outside the 1-core) is applied, MinerLC in comparison finds 1,555,292 and 11,833,577 closed patterns, respectively. For MinerLSD, in the LastFM case there are no strong differences when using 1-cores, 2-cores and 3-cores while we know from Figure 2 that using 4-cores does have an important effect. Corresponding results are also observed for larger sizes of the respective k-cores. Regarding the Delicious dataset, we observe a smaller number of patterns at local modularity levels 0.04 and 0.05 with 1-cores than with 2 and 3-cores. When no local modularity constraint is applied the closed patterns with 2 and 3-cores are a subset of the closed patterns with 1-cores, therefore the results seem counterintuitive at first. However, for the same pattern the 3-

Table 4 MinerLSD #lm, #lme and execution time - small datasets, compared to #c of MinerLC for same core constraints.

Lawyers	1-core	#c = 3221		time = 1			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	2929	2512	1970	1640	1365	1146	295
#lm	1238	738	308	87	39	5	0
time (s)	0	1	0	0	0	0	0
	2-core	#c = 2080		time < 1			
#lme	2080	1938	1670	1454	1265	1089	291
#lm	1262	775	322	104	41	7	0
time (s)	0	0	1	1	0	0	1
	3-core	#c = 1302		time < 1			
#lme	1302	1302	1215	1118	1024	920	282
#lm	1030	746	348	108	43	7	0
time (s)	0	0	1	0	0	0	0
	5-core	#c = 463		time < 1			
#lme	463	463	463	459	449	432	202
#lm	413	366	253	119	36	9	0
time (s)	0	0	1	1	0	0	0
	7-core	#c = 155		time < 1			
#lme	155	155	155	155	155	155	115
#lm	147	133	97	62	36	13	0
time (s)	1	1	0	0	0	1	0
CoExp	1-core	#c = 196		time < 1			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	178	166	150	133	125	114	64
#lm	83	65	35	16	8	1	0
time (s)	0	0	0	0	0	0	1
	2-core	#c = 172		time < 1			
#lme	162	153	141	125	118	108	64
#lm	89	78	51	26	12	3	0
time (s)	0	0	0	1	0	0	0
	3-core	#c = 138		time < 1			
#lme	134	129	118	109	102	95	56
#lm	75	64	42	23	12	0	0
time (s)	0	0	0	0	0	0	0
	5-core	#c = 62		time < 1			
#lme	62	60	57	51	48	47	31
#lm	31	23	12	4	2	1	0
time (s)	1	0	0	0	1	0	0
	7-core	#c = 37		time < 1			
#lme	37	37	36	34	33	32	19
#lm	27	22	17	5	3	2	0
time (s)	0	0	0	0	0	0	0

core subgraph is smaller than the 1-core subgraph and may have better local modularity, which happens in the Delicious case.

Regarding the CPU times, we observe a considerable decrease using appropriate local modularity thresholds for both LastFM and Delicious which is especially important for weaker (graph-)constraints, i. e., with respect to the applied k-cores. Using the appropriate modularity thresholds the runtime can be considerably decreased which enables new approaches already for medium

Table 5 MinerLSD #lm, #lme and execution time - DBLP.C and DBLP.P, compared to #c of MinerLC for same core constraints.

DBLP.C	1-core	#C = 14820		time = 31			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	179	66	24	16	9	7	1
#lm	144	42	14	3	2	1	0
#time (s)	41	36	31	30	25	25	17
	2-core	#c = 1991		time = 20			
#lme	101	35	19	10	6	5	1
#lm	78	29	11	4	2	1	0
#time (s)	23	22	21	20	18	19	15
	3-core	#c = 319		time = 11			
#lme	46	23	11	5	4	2	1
#lm	39	15	5	3	2	1	0
#time (s)	12	11	11	11	10	10	9
	5-core	#c = 20		time = 2			
#lme	8	3	2	2	2	1	1
#lm	7	3	2	2	1	1	0
#time (s)	3	3	3	3	3	3	3
	7-core	#c = 2		time = 1			
#lme	1	1	1	1	1	1	1
#lm	1	1	1	1	1	1	0
#time (s)	1	1	0	1	1	1	0
DBLP.P	1-core	#c = 2396		time = 9			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	34	22	15	9	7	5	3
#lm	28	20	7	4	3	3	0
time (s)	42	42	42	40	38	37	33
	2-core	#c = 661		time = 7			
#lme	31	21	12	9	7	5	3
#lm	25	19	7	4	3	3	0
time (s)	38	39	39	38	37	37	33
	3-core	#c = 261		time = 7			
#lme	27	20	10	7	6	5	3
#lm	21	12	5	4	3	3	0
time (s)	32	33	34	34	32	33	30
	5-core	#c = 84		time = 6			
#lme	12	9	7	7	5	4	5
#lm	12	9	6	4	4	3	0
time (s)	20	21	20	20	19	19	17
	7-core	#c = 42		time = 5			
#lme	10	8	7	4	4	4	2
#lm	10	7	5	4	4	3	0
time (s)	12	12	12	12	11	11	10

sized datasets, e. g., concerning pattern exploration. Specifically, if we compare the extra computation performed by MinerLSD for computing the estimate, in the Delicious case, the benefit is immediately obvious: MinerLSD is always much faster than MinerLC. The LastFM dataset shows a somewhat different picture: with weaker core-constraints and at local modularity level of 0.01 MinerLC (which does not consider local modularity) is (slightly) faster than MinerLSD. This is not that surprising, since MinerLSD has to compute local modularity estimates and local modularities for all the developed patterns during search. However, first this happens only for weak constraints, and second, when using MinerLC all these computations (in fact much more as there is no pruning), would have to be made anyway in post-processing fashion for obtaining the patterns according to a local modularity threshold. Furthermore, the runtime behavior of LastFM here is similar to DBLP.P and can also be explained by the smaller number of labels compared to Delicious. Overall, this shows that if we consider appropriate local modularity thresholds MinerLSD already allows the analysis of larger datasets, especially in terms of larger sizes of the labels, while comparable results (with respect to MinerLC) are usually obtained for weak (graph-)constraints. However, the efficient pruning of MinerLSD is important, e. g., for exploration, and also for the processing of larger datasets, as we will also discuss in the next section for large datasets. Detailed results are presented in Table 6 which also displays the #lme numbers.

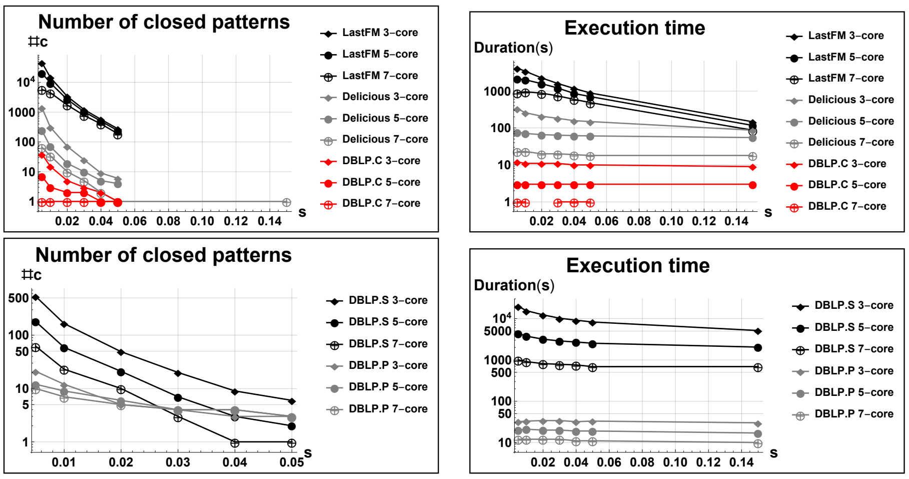

Fig. 6 Number of patterns and execution time of MinerLSD on the DBLP.C, DBLP.P, Delicious and LastFM datasets with 3-cores, 5-cores and 7-cores and local modularity thresholds ranging from 0.01 to 0.15. The Y-axis of the topmost figure represents the number of closed patterns outptut by MinerLSD while the bottom figure displays the CPU time. Both Y-axis are displayed using a logarithmic scale.

Large Datasets In this section, we present experiments of MinerLSD on two large datasets, namely DBLP.S and DBLP.XL (see Table 1 for their characteristics) to further explore the scalability of MinerLSD when using both k-core and local modularity constraints. Again we do not use any threshold on the pattern supports.

In Table 7, we report the results on DBLP.S and DBLP.XL with the same local modularity thresholds as in the previous section and applying $k = 1, 2, 3, 5, 7$ and 7 k-core constraints, respectively. The scalability of MinerLSD depends obviously on the size and density of the network but also

Table 6 MinerLSD #lm, #lme and execution time compared to #c of MinerLC for same core constraints.

LastFM	1-core	#c=1555292		time=2874			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme		59528	16163	6817	3475	1920	52
#lm		17627	3633	1238	575	276	0
time (s)		5816	3400	2252	1605	1187	196
	2-core	#c = 471546		time = 2320			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme		50507	14752	6464	3349	1856	52
#lm		16751	3646	1252	583	282	0
time (s)		4668	2915	1995	1452	1073	178
	3-core	#c = 161764		time = 1878			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	87211	39127	12694	5753	3039	1720	50
#lm	46400	14637	3377	1219	572	276	0
time (s)	4149	3422	2262	1596	1174	885	147
	5-core	#c = 26312		time = 1069			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	24807	18103	8224	4272	2352	1412	46
#lm	20562	9507	2680	1035	496	239	0
time (s)	2148	2013	1580	1206	857	706	117
	7-core	#c = 5859		time = 531			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	5854	5620	4031	2533	1517	994	39
#lm	5814	4482	1737	775	402	189	0
time (s)	902	953	877	738	594	486	87
Delicious	1-core	#c=11833577		time=121934			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme		5655	776	255	121	71	4
#lm		2214	165	31	6	1	0
time (s)		5296	2018	1173	825	643	179
	2-core	#c = 130458		time = 1845			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	7251	1421	288	116	65	37	3
#lm	5440	879	138	39	11	6	0
time (s)	1499	920	569	426	358	298	129
	3-core	#c = 11076		time = 269			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	1729	430	114	51	25	17	1
#lm	1419	311	71	25	9	6	0
time (s)	331	259	208	182	158	149	87
	5-core	#c = 576		time = 68			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	296	89	25	14	7	6	1
#lm	241	70	19	10	5	4	0
time (s)	77	71	66	64	62	61	55
	7-core	#c = 77		time = 21			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	67	41	13	7	4	1	1
#lm	66	34	10	5	2	1	1
time (s)	23	23	20	20	19	18	18

heavily depends on the size of the attribute set and on the average number of labels per vertex. DBLP.XL is then a real challenge as it is a large network made of 929,937 vertices related by 3,461,697 edges and described by more than 90,000 items, with an average number of 10.16 labels per vertex. The efficiency of the optimistic pruning is then of primary importance.

As can be seen in the results table, optimistic estimate pruning using local modularity is quite effective in achieving an efficient pattern mining approach. For both datasets, we observe large reductions in the number of patterns, while focussing on the interesting ones according to the applied local modularity interestingness measure and the utilized local modularity thresholds. In particular, the results for DBLP.S indicate the enormous pruning efficiency - here the dataset for weaker constraints cannot be handled by MinerLC at all, where the computation did not terminate after 36 hours. The DBLP.XL results indicate the same trend. Overall, this indicates the huge impact of optimistic estimate pruning using local modularity as provided by MinerLSD for handling large datasets.

Table 7 MinerLSD #lm, #lme and execution time compared to #c of MinerLC for same core constraints

DBLP.S	1-core	$\#c \geq 3457143$		time = STOPPED AFTER 36h			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	1150	351	103	50	26	18	1
#lm	778	230	68	25	12	6	0
time (s)	59989	37645	24906	20634	17299	16167	8332
	2-core	$\#c \geq 3584834$		time = STOPPED AFTER 36h			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	958	303	94	44	24	16	1
#lm	722	218	64	24	12	6	0
time (s)	36302	25949	19065	16068	13869	12907	7073
	3-core	$\#c = 1576164$		time = 45720			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	621	208	72	28	17	9	1
#lm	533	165	49	20	9	6	0
time (s)	19799	15531	12329	10221	9149	8276	5143
	5-core	$\#c = 44345$		time = 3791			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	200	71	26	10	6	4	1
#lm	180	59	21	7	3	2	0
time (s)	4410	3760	3173	2877	2709	2533	2044
	7-core	$\#c = 5659$		time = 881			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	62	24	10	4	2	1	1
#lm	62	23	10c	3	1	1	0
time (s)	1005	908	812	784	756	687	689
DBLP.XL	7-core	$\#c = 9206$		time = 93906			
l	0.005	0.01	0.02	0.03	0.04	0.05	0.15
#lme	10	5	4	3	2	1	1
#lm	9	5	3	1	1	1	0
time (s)	113790	111079	110142	107967	103363	97326	96811

6.2.6 Structural Pattern Set Analysis

In the following, we analyze the results of the proposed pattern mining method *MinerLSD* in more detail, focussing on different graph statistics. We report exemplary results on three datasets with different characteristics as outlined in Section 5, i. e., the Lawyers, the CoExp, and the DBLP.C datasets. We consider all patterns above a given local modularity threshold, combined with different core abstractions. For computing the graph statistics, we analyze the respective induced subgraph W of each pattern, and consider the following: (1) the vertex count N_W, (2) the edge count E_W, (3) the scaled density (cf., [45]) of subgraph W, i. e., the ratio of E_W divided by the number of edges of a complete graph with the same number of vertices as W and multiplied (scaled) by the total number of vertices; this measure approximately estimates the average degree of the nodes contained in the community, cf., [45]. (4) Furthermore, we also consider the fraction of outgoing edges, i. e., the edges connecting nodes contained in the pattern with others not being part of the pattern subgraph, to the set of edges E_W. The results are shown in Tables 8-11.

Table 8 Vertex Counts: Mean and standard deviation (in brackets) of the number of vertices of the pattern support, i. e., of the induced pattern subgraphs, for different values of k and the local modularity threshold lm.

Lawyers: $n = 71$ $m = 556$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	12.7 (9.6)	17.2 (9.5)	20.11 (9.6)	24.4 (9.5)	29.8 (4.59)
2	15.3 (10.0)	16.8 (9.4)	19.35 (9.3)	23.4 (9.2)	27.7 (5.6)
3	18.0 (10.6)	18.0 (9.5)	19.49 (9.2)	22.9 (9.0)	26.8 (6.3)
5	23.2 (11.9)	21.7 (10.1)	21.50 (9.0)	23.7 (8.3)	27.3 (5.6)
CoExp: $n = 151$, $m = 1849$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	59.5 (44.0)	51.9 (35.8)	51.5 (32.8)	54.6 (31.6)	53.9 (21.9)
2	56.9 (41.5)	49.2 (31.9)	51.9 (29.4)	57.2 (25.9)	63.0 (18.5)
3	54.7 (37.9)	47.2 (31.6)	47.4 (28.8)	52.2 (24.7)	60.2 (18.3)
5	50.0 (36.2)	48.3 (34.2)	51.6 (33.1)	54.9 (28.9)	85.5 (9.2)
DBLP.C: $n = 3140$, $m = 10689$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	8.3 (30.8)	124.0 (106.1)	231.3 (144.0)	373.9 (163.6)	631.0 (63.6)
2	12.5 (68.3)	124.6 (325.9)	159.5 (103.9)	248.0 (119.0)	434.5 (64.4)
3	21.7 (126.4)	117.3 (350.3)	250.7 (549.6)	604.6 (906.4)	1256.5 (1366.8)
5	62.6 (199.6)	164.9 (327.7)	351.3 (480.6)	511.0 (555.8)	904.0 (0.0)

Considering the results shown in Tables 8-9 we observe that, as expected, increasing numbers of k tend to focus on larger communities, which is especially the case for weaker core constraints and larger local modularity thresholds. In particular, we observe those trends for the local modularity for the Lawyers and the DBLP.C datasets, while this is also pronounced for CoExp regarding stronger constraints. For the DBLP.C network, in particular, we observe a rather strong effect. Overall, with no constraints quite small patterns are detected. When the k-core constraint and the local modularity threshold are increased, then larger patterns are detected which are also considerably denser than those with no constraints. This can clearly be observed in Table 10

Table 9 Edge Counts: Mean and standard deviation (in brackets) of the number of edges of the pattern support, i. e., of the induced pattern subgraphs, for different values of k and the local modularity threshold lm.

Lawyers: $n = 71$ $m = 556$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	38.2 (58.0)	61.4 (61.4)	78.4 (64.0)	103.2 (67.1)	25.6 (63.9)
2	53.8 (66.3)	62.7 (61.9)	78.3 (64.4)	104.1 (67.3)	126.1 (68.1)
3	73.2 (75.4)	72.9 (65.5)	82.2 (64.4)	104.5 (65.0)	124.8 (67.9)
5	121.7 (95.5)	109.9 (77.0)	108.2 (67.7)	123.7 (62.6)	135.7 (59.8)
CoExp: $n = 151$, $m = 1849$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	365.4 (473.9)	323.1 (512.8)	327.1 (497.0)	392.0 (579.7)	448.6 (653.1)
2	404.7 (488.8)	317.2 (492.9)	325.0 (483.7)	352.2 (492.9)	377.4 (529.4)
3	445.1 (496.5)	385.1 (535.0)	375.7 (525.1)	394.6 (529.5)	481.5 (598.9)
5	525.4 (499.6)	551.8 (563.1)	625.0 (548.9)	729.3 (544.9)	1495.0 (132.9)
DBLP.C: $n = 3140$, $m = 10689$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	7.4 (91.4)	164.9 (192.5)	347.3 (282.5)	627.8 (340.3)	1278.0 (356.4)
2	22.8 (240.5)	316.81 (1183.3)	349.2 (271.3)	577.7 (328.6)	1126.0 (357.8)
3	66.7 (531.01)	419.7 (1487.5)	945.3 (2350.3)	2393.0 (3924.2)	5229.0 (5897.3)
5	347.3 (1246.9)	950.4 (2066.5)	2101.3 (3055.2)	3085.0 (3586.5)	5621.0 (0.0)

Table 10 Scaled Graph Densities: Mean and standard deviation (in brackets) of the scaled densities of the pattern support, i. e., of the induced pattern subgraphs, for different values of k and the local modularity threshold lm.

Lawyers: $n = 71$ $m = 556$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	4.59 (2.63)	6.34 (2.51)	7.26 (2.46)	8.13 (2.61)	8.98 (3.27)
2	5.92 (2.56)	6.63 (2.47)	7.50 (2.38)	8.50 (2.42)	9.43 (2.82)
3	7.23 (2.47)	7.42 (2.32)	7.92 (2.27)	8.80 (2.27)	9.45 (2.67)
5	9.89 (2.28)	9.74 (2.10)	9.82 (1.96)	10.38 (1.80)	11.38 (1.95)
7	12.23 (2.08)	12.15 (2.05)	11.99 (1.86)	12.20 (1.40)	13.14 (1.20)
CoExp: $n = 151$, $m = 1849$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	9.58 (8.34)	9.38 (9.54)	10.14 (9.83)	11.17 (11.12)	12.19 (13.90)
2	11.15 (8.37)	9.78 (9.09)	10.14 (9.18)	10.41 (9.85)	10.19 (11.46)
3	13.07 (8.60)	12.02 (9.66)	12.15 (9.83)	12.22 (10.44)	13.22 (13.20)
5	17.92 (8.25)	18.40 (9.00)	20.68 (8.50)	25.22 (5.72)	35.42 (0.71)
DBLP.C: $n = 3140$, $m = 10689$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	1.73 (0.57)	2.60 (0.82)	2.93 (0.75)	3.38 (0.65)	4.02 (0.73)
2	3.15 (0.68)	4.15 (0.98)	4.33 (0.72)	4.66 (0.68)	5.13 (0.89)
3	4.73 (0.87)	5.90 (0.78)	6.35 (0.79)	6.74 (1.14)	7.89 (0.80)
5	7.30 (1.57)	8.55 (1.94)	10.16 (2.02)	10.92 (2.17)	12.45 (0)

for increasing k-core and local modularity threshold values. Furthermore, when we consider the ratio of outgoing edges vs. in-edges of a pattern shown in Table 11, then we also observe the trend that the proposed approach focuses on selecting denser pattern subgraphs with a stronger connectivity structure in terms of the links within the subgraph, i. e., the in-edges. This is especially obvious for higher k-core and local modularity threshold values, as exemplified by the CoExp and DBLP.C datasets, e. g., for $k = 5$ and $lm = 0.04$ where the number of in-edges strongly "dominates" the number of outgoing edges.

Table 11 Ratio of outgoing edges to edges in the pattern subgraph (in-edges): Mean and standard deviation (in brackets) of that ratio of the pattern support, i. e., of the induced pattern subgraphs, for different values of k and the local modularity threshold lm.

Lawyers: $n = 71$ $m = 556$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	10.99 (9.07)	4.90 (2.55)	3.68 (1.72)	2.68 (1.19)	1.80 (0.65)
2	6.58 (4.40)	4.81 (2.53)	3.67 (1.72)	2.64 (1.16)	1.81 (0.65)
3	4.62 (2.81)	4.14 (2.18)	3.46 (1.58)	2.58 (1.10)	1.84 (0.66)
5	2.74 (1.62)	2.82 (1.53)	2.69 (1.32)	2.19 (0.89)	1.56 (0.52)
CoExp: $n = 151$, $m = 1849$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	4.95 (8.72)	2.46 (1.59)	2.13 (1.36)	1.62 (1.11)	0.69 (0.34)
2	3.88 (4.76)	2.69 (1.79)	2.48 (1.52)	2.24 (1.35)	1.69 (1.10)
3	3.58 (4.48)	2.83 (2.33)	2.58 (1.77)	2.41 (1.49)	1.82 (1.14)
5	3.75 (4.12)	2.89 (2.75)	2.24 (2.04)	1.57 (1.07)	0.16 (0.10)
DBLP.C: $n = 3140$, $m = 10689$					
k	No lm	lm=0.005	lm=0.01	lm=0.02	lm=0.04
1	14.23 (13.72)	6.85 (1.94)	5.28 (1.19)	3.94 (0.80)	2.39(0.41)
2	11.29 (10.37)	5.99 (2.09)	4.88 (1.17)	3.78 (0.77)	2.44 (0.50)
3	11.44 (8.04)	5.96 (1.90)	4.57 (1.64)	2.96 (1.82)	1.27 (1.68)
5	10.99 (8.22)	5.99 (3.07)	3.13 (2.53)	1.99 (2.22)	0.42 (0)

6.2.7 Pattern Selection and K-Core Abstraction

In this section, we provide examples of patterns demonstrating the benefits of pattern selection using local modularity and k-core abstraction. In particular, we discuss illustrative examples from two different datasets – the Lawyers and the (larger) DBLP.C dataset.

Lawyers Dataset In order to demonstrate the effectiveness of the pattern exploration and selection methodology using abstract closed pattern with k-cores and local modularity, we exemplify that with the two patterns shown in Figure 7. Here, we show two similar patterns in terms of Jaccard similarity (0.52) considering the nodes of the respective pattern-induced subgraphs. While the patterns are very similar regarding the overlap and their size, they have quite different local modularity values referring to their connectivity structure. The left pattern described by $35 < Age \leq 65$ AND $Seniority < 5$ AND $Status = Partner$, with a $size = 24$ of the set of nodes in its subgraph, is considerably denser with a local modularity of MODL = 0.058, compared to the pattern on the right; the latter is described by $Age < 40$ AND $Seniority \leq 30$, with a $size = 23$ of the pattern support and a local modularity of only MODL = 0.013. Therefore, while both patterns are abstract closed patterns according to similar support criteria and the 5-core abstraction, a higher modularity threshold, e. g., MODL ≥ 0.05 would only select the first (left pattern in Figure 7) instead of the right pattern. From the description, we can also observe that the selected (left) pattern is more interesting, since it provides a more precise description. In the figures, we depict in red the edges and the vertices in the pattern subgraph; in gray, we show the out-edges of the pattern (i. e., one vertex of a gray edge is contained in the pattern extension and the other vertex is not); in light gray we depict the rest of the graph.

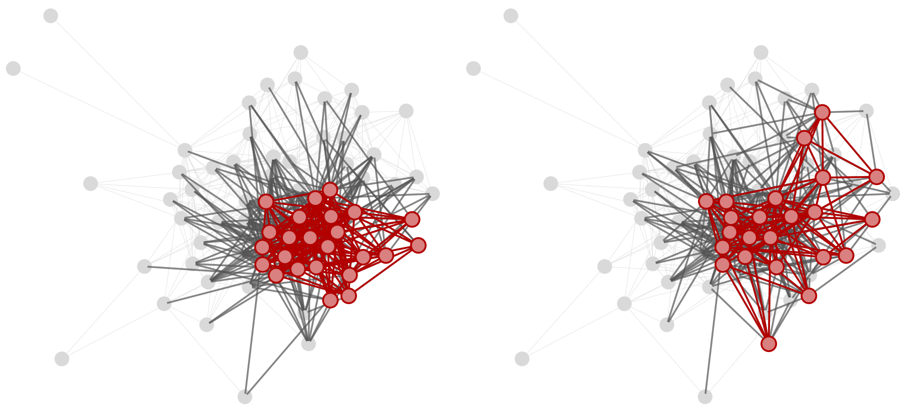

Fig. 7 Example patterns from the Lawyers dataset: Both patterns are similar 5-cores, with a Jaccard similarity considering the nodes of the respective pattern-induced subgraphs of 0.52. The pattern on the left (described by $35 < Age \leq 65$ AND $Seniority < 5$ AND $Status = Partner$, with $size = 24$) is considerably denser with a local modularity of MODL = 0.058, compared to the pattern on the right (described by $Age < 40$ AND $Seniority \leq 30$, with $size = 23$) which only has a local modularity of MODL = 0.013. In the figures, we depict in red the edges and the vertices in the pattern extension, in gray the out-edges of the pattern (i. e., one vertex of a gray edge is contained in the pattern subgraph and the other vertex is not) and in light gray the rest of the graph.

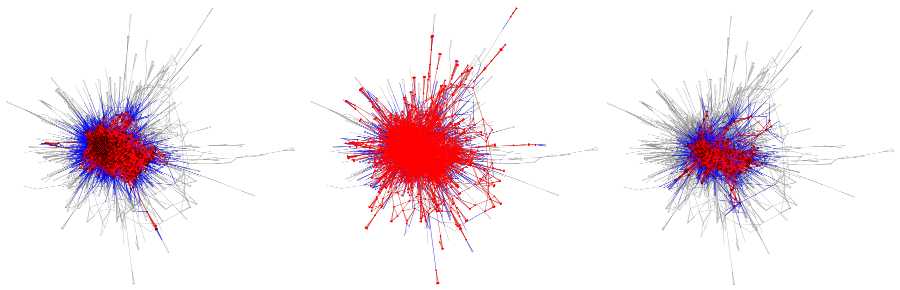

Fig. 8 Illustrative patterns (DBLP.C). Left: 5-core empty pattern with a local modularity of MODL = 0.1223; middle: 3-core empty pattern with a local modularity of MODL = 0.0430; right: 3-core "mine" pattern with a local modularity MODL = 0.0503. In the plots, red color indicates the core graph (i. e., the in-edges of the pattern), blue color shows the edges incident to the nodes of the core graph, gray depicts the edges of the rest of the graph.

DBLP.C Dataset In order to show the impact of pattern selection and k-core abstraction, we first consider the local Modularities on k-cores with increasing k. For analyzing the impact of the k-cores we firstly consider the empty pattern, thus only focussing on the abstraction by the applied k-core. For the local modularity values of the empty pattern, for $k = 2, 3, 4, 5$ we observe MODL $= 0.0075, 0.0430, 0.0915, 0.1223$, respectively. Thus, we observe the clear trend that increasing k yields patterns with higher connectivity structures as shown by the increasing local modularity values; similar trends are obtained for the other datasets. This complements our results in the last section, where we discussed, how increasing k for the k-core abstraction together with increasing local modularity thresholds focuses on larger and more "interesting" patterns as measured by the local modularity quality function.

Figure 8 illustrates these findings: The two left graphs show examples of the k-cores for the empty pattern, specifically, for the 5-core with the highest local modularity, and the corresponding 3-core pattern. Areas in red indicate the core graph – both vertices and edges, blue color shows the remaining edges incident to the nodes of the core graph, while gray depicts the edges of the rest of the graph. It is easy to see that both the 3-core (2223 vertices and 9399 edges) as well as the 5-core (904 vertices and 5621 edges) demonstrate a considerably strong connectivity structure. Finally, the graph plotted on the right of Figure 8 shows a specialization of the empty pattern on the 3-core, i.e. the pattern given by the label "mine". This pattern is obviously smaller (covering 290 vertices and 1059 edges) than the empty pattern, while its modularity structure is slightly better (MODL = 0.0503). The left plot in Figure 9 shows the "mine" pattern in detail, as a "zoom-in" focussing on all edges incident to nodes contained in the pattern subgraph.

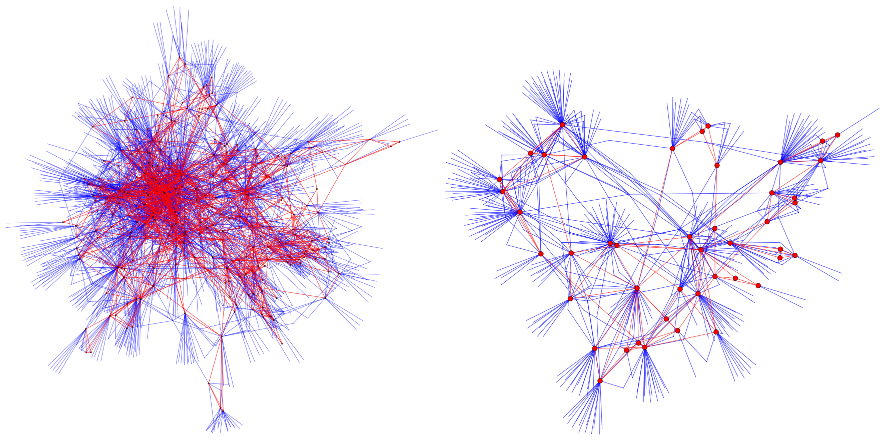
Fig. 9 Detailed view: "mine" pattern (left), with local modularity MODL = 0.0503 vs. the lower-quality "algorithm" pattern (right), MODL = 0.0072. In-edges (red), out-edges (blue).

Figure 9 illustrates the selection process for different 3-core patterns in detail, providing the "mine" pattern (covering 290 vertices and 1059 edges, MODL = 0.0503) that is selected according to a local modularity threshold $lm = 0.04$ and the "algorithm" pattern (covering 45 vertices and 93 edges, MODL = 0.0072) which is a further specialization of the 3-core empty pattern. As we can clearly observe for the "mine" pattern, its structure is more interesting concerning its connectivity – i. e., its distributional unusualness compared to the expectation modeled by the null-model. This is a representative illustration, how the proposed approach using local modularity pruning achieves a better pattern selection method for the same core constraint(s).

7 Conclusions

In this paper, we have proposed the novel MinerLSD method for efficient local pattern mining on attributed networks. It enumerates local patterns and associated subgroups in attributed networks, utilizing different pattern and graph mining techniques. In particular, MinerLSD is based on three main basic ideas: First, enumerating only closed patterns, which is particularly beneficial whenever items have dependencies. This occurs as soon as some attributes, either numeric or hierarchical, have to be translated into various items to express interesting patterns, e. g., interrelated intervals and hierarchical dependencies. Second, we focus on reducing pattern subgraphs to core subgraphs which allows both to strongly reduce the number of patterns and to focus on essential parts of graphs. Third, we select cohesive subgraphs during the search according to topological quantities as local modularity and, above all, to allow pruning by using optimistic estimates of the local modularity measure.

We performed a set of experiments in order to estimate the impact of the investigated approaches, for which we included two baseline methods, i. e., MinerLC and COMODO for comparison. The purpose was then to investigate i) the pruning efficiency of MinerLSD using the local modularity estimate as implemented in COMODO, ii) the impact of searching for closed patterns (as implemented in MinerLC) and therefore enumerating only the cohesive subgraph associated to the patterns, and iii) the added potential for pattern selection based on the combination of both k-core abstraction and local modularity selection. The latter allows to strongly reduce the number of patterns while focussing on essential parts of the graph which leads to more interesting high quality patterns. For our experiments we used a number of datasets with different characteristics, also ranging from small to large datasets in order to estimate the scalability of MinerLSD. Overall the result indicated effects that were always positive, and sometimes even crucial, for allowing to handle even rather complex and large datasets with reasonable pattern set sizes and computational effort – without using any minimum support threshold. Specifically, the results of our experiments show the efficiency of the presented method. Furthermore, we have presented exemplary results showing the benefit of pattern selection and abstraction which demonstrate the efficacy of the proposed *MinerLSD* approach. Overall, by implementing the different ideas. and techniques summarized above in the novel *MinerLSD* method, i. e., utilizing closed patterns, graph abstractions, optimistic estimate pruning using local modularity), we obtain a very flexible tool that allows to handle large graphs with adequate constraints on the subgroups and patterns to discover.

For future work, we intend to characterize the attributed graphs in terms of which pruning method is especially efficient, and to investigate other measures than local modularity in order to estimate their pruning efficiency. Furthermore, we aim to investigate other core definitions than k-cores as well. Also, focussing on sets of (local) patterns, and their relations, in order to obtain, e. g., the most diverse, representative, interesting, and relevant results, cf., [13, 42, 49, 79] is a further interesting research direction to consider.

Declarations

Availability of Data and Material

Datasets and the implementation of MinerLSD can be found at the following website: `https://lipn.univ-paris13.fr/MinerLC/`

Competing Interests

None of the authors have any competing interests in the manuscript.

Funding

This work has been partially supported by the German Research Foundation (DFG) project "MODUS" (under grant AT 88/4-1). Furthermore, the research leading to these results has received funding from the Project Chistera Adalab (ANR-14-CHR2-0001-04).

Authors' Contributions

MA and HS conceived of the idea and study, interpretation of the data and drafted the manuscript. MA and GS ran the experiments. DB and GS implemented MinerLSD and related software. All authors read and approved the final manuscript.

Acknowledgements

Martin Atzmueller was supported in part by Université Sorbonne Paris Cité as a visiting professor.

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. VLDB, pp. 487–499. Morgan Kaufmann (1994)
2. Almendral, J.A., Oliveira, J., López, L., Mendes, J., Sanjuán, M.A.: The Network of Scientific Collaborations within the European Framework Programme. Physica A: Statistical Mechanics and its Applications **384**(2), 675 – 683 (2007)
3. Atzmueller, M.: Data Mining on Social Interaction Networks. JDMDH **1** (2014)
4. Atzmueller, M.: Subgroup Discovery. WIREs DMKD **5**(1), 35–49 (2015)
5. Atzmueller, M.: Onto Explicative Data Mining: Exploratory, Interpretable and Explainable Analysis. In: Proc. Dutch-Belgian Database Day. TU Eindhoven, NL (2017)
6. Atzmueller, M.: Compositional Subgroup Discovery on Attributed Social Interaction Networks. In: Proc. International Conference on Discovery Science. Springer (2018)
7. Atzmueller, M.: Onto Model-based Anomalous Link Pattern Mining on Feature-Rich Social Interaction Networks. In: Proc. WWW 2019 (Companion). IW3C2 / ACM (2019)
8. Atzmueller, M., Doerfel, S., Mitzlaff, F.: Description-Oriented Community Detection using Exhaustive Subgroup Discovery. Information Sciences **329**, 965–984 (2016)
9. Atzmueller, M., Lemmerich, F.: Fast Subgroup Discovery for Continuous Target Concepts. In: Proc. 18th International Symposium on Methodologies for Intelligent Systems (ISMIS 2009), *LNCS*, vol. 5722, pp. 1–15. Springer, Berlin/Heidelberg, Germany (2009)
10. Atzmueller, M., Lemmerich, F.: VIKAMINE - Open-Source Subgroup Discovery, Pattern Mining, and Analytics. In: Proc. ECML/PKDD. Springer (2012)
11. Atzmueller, M., Mitzlaff, F.: Towards Mining Descriptive Community Patterns. In: Workshop on Mining Patterns and Subgroups. Lorentz Center, Leiden, NL (2010)

12. Atzmueller, M., Mitzlaff, F.: Efficient Descriptive Community Mining. In: Proc. FLAIRS, pp. 459 – 464. AAAI Press (2011)
13. Atzmueller, M., Mueller, J., Becker, M.: Mining, Modeling and Recommending 'Things' in Social Media, chap. Exploratory Subgroup Analytics on Ubiquitous Data. No. 8940 in LNAI. Springer, Berlin/Heidelberg, Germany (2015)
14. Atzmueller, M., Puppe, F.: SD-Map - A Fast Algorithm for Exhaustive Subgroup Discovery. In: Proc. PKDD, pp. 6–17. Springer, Berlin/Heidelberg, Germany (2006)
15. Atzmueller, M., Soldano, H., Santini, G., Bouthinon, D.: MinerLSD: Efficient Local Pattern Mining on Attributed Graphs. In: Proc. 2018 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE Press, Boston, MA, USA (2018)
16. Balasubramanyan, R., Cohen, W.W.: Block-LDA: Jointly modeling entity-annotated text and entity-entity links. In: Proc. SDM, pp. 450–461 (2011)
17. Batagelj, V., Zaversnik, M.: Generalized Cores. CoRR **cs.DS/0202039** (2002)
18. Batagelj, V., Zaversnik, M.: Fast Algorithms for Determining (Generalized) Core Groups in Social Networks. Adv. Data Analysis and Classification **5**(2), 129–145 (2011)
19. Bechara Prado, A., Plantevit, M., Robardet, C., Boulicaut, J.F.: Mining Graph Topological Patterns: Finding Co-variations among Vertex Descriptors. IEEE Transactions on Knowledge and Data Engineering **25**(9), 2090–2104 (2013)
20. Belfin, R., Bródka, P., et al.: Overlapping Community Detection using Superior Seed Set Selection in Social Networks. Computers & Electrical Engineering **70**, 1074–1083 (2018)
21. Bendimerad, A.A., Plantevit, M., Robardet, C.: Unsupervised Exceptional Attributed Subgraph Mining in Urban Data. In: Proc. ICDM, pp. 21–30. IEEE (2016)
22. Boley, M., Horváth, T., Poigné, A., Wrobel, S.: Listing Closed Sets of Strongly Accessible Set Systems with Applications to Data Mining. TCS **411**(3), 691–700 (2010)
23. Bothorel, C., Cruz, J.D., Magnani, M., Micenkova, B.: Clustering Attributed Graphs: Models, Measures and Methods. Network Science **3**(03), 408–444 (2015)
24. Cantador, I., Brusilovsky, P., Kuflik, T.: 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec). In: Proc. RecSys. ACM (2011)
25. Combe, D., Largeron, C., Géry, M., Egyed-Zsigmond, E.: I-louvain: An attributed graph clustering method. In: Proc. IDA. Advances in Intelligent Data Analysis, pp. 181–192 (2015)
26. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery: An Overview. In: U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (eds.) Advances in Knowledge Discovery and Data Mining, pp. 1–34. AAAI Press (1996)
27. Fortunato, S.: Community Detection in Graphs. Phys. Rep. **486**(3-5), 75–174 (2010)
28. Fortunato, S., Castellano, C.: Encyclopedia of Complexity and System Science, chap. Community Structure in Graphs. Springer, Heidelberg, Germany (2007)
29. Freeman, L.: Segregation In Social Networks. Sociological Methods & Research **6**(4), 411 (1978)
30. Galbrun, E., Gionis, A., Tatti, N.: Overlapping Community Detection in Labeled Graphs. DMKD **28**(5-6), 1586–1610 (2014)
31. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer Verlag, Heidelberg (1999)
32. Ge, R., Ester, M., Gao, B.J., Hu, Z., Bhattacharya, B.K., Ben-Moshe, B.: Joint cluster analysis of attribute data and relationship data: The connected k-center problem, algorithms and applications. TKDD **2**(2) (2008)
33. Grosskreutz, H., Rüping, S., Wrobel, S.: Tight Optimistic Estimates for Fast Subgroup Discovery. In: Proc. ECML/PKDD, *LNCS*, vol. 5211, pp. 440–456. Springer, Berlin/Heidelberg, Germany (2008)
34. Günnemann, S., Färber, I., Boden, B., Seidl, T.: GAMer: A Synthesis of Subspace Clustering and Dense Subgraph Mining. KAIS **40**(2), 243–278 (2013)
35. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns Without Candidate Generation. In: Proc. ACM SIGMOD, pp. 1–12. ACM Press (2000)
36. Kanawati, R.: Seed-Centric Approaches for Community Detection in Complex Networks. In: International Conference on Social Computing and Social Media, pp. 197–208. Springer, Berlin/Heidelberg, Germany (2014)

37. Kanawati, R.: Yasca: an ensemble-based approach for community detection in complex networks. In: International Computing and Combinatorics Conference, pp. 657–666. Springer, Berlin/Heidelberg, Germany (2014)
38. Kaytoue, M., Plantevit, M., Zimmermann, A., Bendimerad, A., Robardet, C.: Exceptional Contextual Subgraph Mining. Machine Learning pp. 1–41 (2017)
39. Kibanov, M., Atzmueller, M., Scholz, C., Stumme, G.: Temporal Evolution of Contacts and Communities in Networks of Face-to-Face Human Interactions. Science China Information Sciences **57**(3), 1–17 (2014)
40. Klösgen, W.: Explora: A Multipattern and Multistrategy Discovery Assistant. In: Advances in Knowledge Discovery and Data Mining, pp. 249–271. AAAI Press (1996)
41. Knobbe, A.J., Cremilleux, B., Fürnkranz, J., Scholz, M.: From Local Patterns to Global Models: The LeGo Approach to Data Mining. In: From Local Patterns to Global Models: Proceedings of the ECML/PKDD-08 Workshop (LeGo-08), pp. 1 – 16 (2008)
42. Knobbe, A.J., Ho, E.K.: Pattern Teams. In: Proc. PKDD, pp. 577–584. Springer, Berlin/Heidelberg, Germany (2006)
43. Kumar, R., Novak, J., Tomkins, A.: Structure and Evolution of Online Social Networks. In: Proc. ACM SIGKDD, pp. 611–617. ACM (2006)
44. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the Overlapping and Hierarchical Community Structure in Complex Networks. New J. Phys. **11**(3) (2009)
45. Lancichinetti, A., Kivelä, M., Saramäki, J., Fortunato, S.: Characterizing the community structure of complex networks. PloS one **5**(8), e11,976 (2010)
46. Lazega, E.: The Collegial Phenomenon: The Social Mechanisms of Cooperation Among Peers in a Corporate Law Partnership. Oxford University Press (2001)
47. Lemmerich, F., Atzmueller, M., Puppe, F.: Fast Exhaustive Subgroup Discovery with Numerical Target Concepts. Data Mining and Knowledge Discovery **30**, 711–762 (2016)
48. Lemmerich, F., Becker, M., Atzmueller, M.: Generic Pattern Trees for Exhaustive Exceptional Model Mining. In: Proc. ECML PKDD, *LNCS*, vol. 7524, pp. 277–292. Berlin/Heidelberg, Germany (2012)
49. Lemmerich, F., Rohlfs, M., Atzmueller, M.: Fast Discovery of Relevant Subgroup Patterns. In: Proc. FLAIRS, pp. 428–433. AAAI Press, Palo Alto, CA, USA (2010)
50. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. CoRR **abs/0810.1355** (2008)
51. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical Comparison of Algorithms for Network Community Detection. In: Proc. WWW, pp. 631–640. ACM, New York, NY, USA (2010)
52. Mitzlaff, F., Atzmueller, M., Benz, D., Hotho, A., Stumme, G.: Community Assessment using Evidence Networks. In: Analysis of Social Media and Ubiquitous Data, *LNAI*, vol. 6904. Springer (2011)
53. Mitzlaff, F., Atzmueller, M., Hotho, A., Stumme, G.: The Social Distributional Hypothesis. SNAM **4**(216) (2014)
54. Mitzlaff, F., Atzmueller, M., Stumme, G., Hotho, A.: Semantics of User Interaction in Social Media. In: Complex Networks IV, *SCI*, vol. 476. Springer (2013)
55. Morik, K.: Detecting Interesting Instances. In: D. Hand, N. Adams, R. Bolton (eds.) Pattern Detection and Discovery, *LNCS*, vol. 2447, pp. 13–23. Springer (2002)
56. Morik, K., Boulicaut, J., Siebes, A. (eds.): Local Pattern Detection, International Seminar, Dagstuhl Castle, Germany, April 12-16, 2004, Revised Selected Papers, *LNCS*, vol. 3539. Springer (2005)
57. Moser, F., Colak, R., Rafiey, A., Ester, M.: Mining Cohesive Patterns from Graphs with Feature Vectors. In: Proc. SDM, pp. 593–604 (2009)
58. Newman, M.E., Girvan, M.: Finding and Evaluating Community Structure in Networks. Phys Rev E Stat Nonlin Soft Matter Phys **69**(2), 1–15 (2004)
59. Newman, M.E.J.: The Structure and Function of Complex Networks. SIAM Review **45**(2), 167–256 (2003)
60. Newman, M.E.J.: Detecting Community Structure in Networks. EPJ **38** (2004)
61. Newman, M.E.J.: Modularity and Community Structure in Networks. PNAS **103**(23), 8577–8582 (2006)
62. Nicolle, R., Radvanyi, F., Elati, M.: Coregnet: Reconstruction and Integrated Analysis of Co-Regulatory Networks. Bioinformatics (2015)

63. Nicosia, V., Mangioni, G., Carchiolo, V., Malgeri, M.: Extending the Definition of Modularity to Directed Graphs with Overlapping Communities. J. Stat. Mech. (2009)
64. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society. Nature **435**(7043), 814–818 (2005)
65. Palla, G., Farkas, I.J., Pollner, P., Derenyi, I., Vicsek, T.: Directed Network Modules. New J. Phys. **9**(6), 186 (2007)
66. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Efficient Mining of Association Rules using Closed Itemset Lattices. Information Systems **24**(1), 25–46 (1999)
67. Peng, C., Kolda, T.G., Pinar, A.: Accelerating Community Detection by Using k-core Subgraphs. arXiv preprint arXiv:1403.2226 (2014)
68. Pernelle, N., Rousset, M.C., Soldano, H., Ventos, V.: ZooM: A Nested Galois Lattices-Based System for Conceptual Clustering. Journal of Experimental and Theoretical Artificial Intelligence **2/3**(14), 157–187 (2002)
69. Pool, S., Bonchi, F., van Leeuwen, M.: Description-driven Community Detection. ACM Transactions on Intelligent Systems and Technology **5**(2) (2014)
70. Seidman, S.B.: Network Structure and Minimum Degree. Social Networks **5**, 269–287 (1983)
71. Silva, A., Meira Jr., W., Zaki, M.J.: Mining attribute-structure correlated patterns in large attributed graphs. Proc. VLDB Endow. **5**(5), 466–477 (2012)
72. Silva, A., Meira Jr, W., Zaki, M.J.: Mining Attribute-Structure Correlated Patterns in Large Attributed Graphs. Proc. VLDB Endowment **5**(5), 466–477 (2012)
73. Smith, L.M., Zhu, L., Lerman, K., Percus, A.G.: Partitioning networks with node attributes by compressing information flow. CoRR **abs/1405.4332** (2014)
74. Soldano, H., Santini, G.: Graph Abstraction for Closed Pattern Mining in Attributed Networks. In: Proc. ECAI, *FAIA*, vol. 263, pp. 849–854. IOS Press (2014)
75. Soldano, H., Santini, G., Bouthinon, D.: Local Knowledge Discovery in Attributed Graphs. In: Proc. ICTAI, pp. 250–257. IEEE (2015)
76. Soldano, H., Santini, G., Bouthinon, D., Lazega, E.: Hub-Authority Cores and Attributed Directed Network Mining. In: Proc. ICTAI. IEEE, Boston, MA, USA (2017)
77. Steinhaeuser, K., Chawla, N.V.: Community detection in a large real-world social network. In: Social computing, behavioral modeling, and prediction, pp. 168–175. Springer (2008)
78. Uno, T., Asai, T., Uchida, Y., Arimura, H.: An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases. In: Proc. Discovery Science, pp. 16–31 (2004)
79. Van Leeuwen, M., Knobbe, A.: Diverse Subgroup Set Discovery. Data Mining and Knowledge Discovery **25**(2), 208–242 (2012)
80. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications, 1 edn. No. 8 in Structural analysis in the social sciences. Cambridge University Press (1994)
81. Wille, R.: Restructuring Lattice Theory. In: Symposium on Ordered Sets, pp. 445–470. University of Calgary, Boston (1982)
82. Wrobel, S.: An Algorithm for Multi-Relational Discovery of Subgroups. In: Proc. PKDD, pp. 78–87. Springer, Berlin/Heidelberg, Germany (1997)
83. Xie, J., Kelley, S., Szymanski, B.K.: Overlapping Community Detection in Networks: The State-of-the-art and Comparative Study. ACM Comput. Surv. **45**(4), 43:1–43:35 (2013)
84. Xie, J., Szymanski, B.K.: LabelRank: A Stabilized Label Propagation Algorithm for Community Detection in Networks. In: Proc. IEEE Network Science Workshop. West Point, NY (2013)
85. Xu, Z., Ke, Y., Wang, Y., Cheng, H., Cheng, J.: A model-based approach to attributed graph clustering. In: Proc. SIGMOD, pp. 505–516 (2012)
86. Yakoubi, Z., Kanawati, R.: Licod: Leader-driven approaches for community detection. Vietnam Journal of Computer Science **1**(4), 241–256 (2014)
87. Yang, J., Leskovec, J.: Defining and Evaluating Network Communities Based on Ground-truth. In: Proc. ACM SIGKDD Workshop on Mining Data Semantics, MDS '12, pp. 3:1–3:8. ACM, New York, NY, USA (2012)
88. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. PVLDB **2**(1), 718–729 (2009)
89. Zhu, L., Ng, W.K., Cheng, J.: Structure and attribute index for approximate graph matching in large graphs. Inf. Syst. **36**(6), 958–972 (2011)