MinerLSD: Efficient Local Pattern Mining on Attributed Graphs

Martin Atzmueller
Tilburg University
Department of Cognitive Science and Artificial Intelligence
Warandelaan 2, 5037 AB Tilburg
The Netherlands
m.atzmuller@uvt.nl

Henry Soldano
LIPN UMR-CNRS 7030
Université Paris 13,
Villetaneuse, France, and
ISYEB UMR 7205, Museum
National d'Histoire
Naturelle Paris, France
henry.soldano@mnhn.fr

Guillaume Santini,
Dominique Bouthinon
LIPN UMR-CNRS 7030
Université Paris 13,
Villetaneuse, France
guillaume.santini@lipn.univ-paris13.fr
db@lipn.univ-paris13.fr

***Abstract*—Local pattern mining on attributed graphs is an important and interesting research area combining ideas from network analysis and graph mining. In this paper, we present *MinerLSD*, a method for efficient local pattern mining on attributed graphs. 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 Modularity metric with optimistic estimates, and 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 graphs.**

***Index Terms*—complex networks, mining attributed graphs, closed pattern mining, community detection**

I. Introduction

The analysis of complex networks is an important task for investigating structural properties, identifying interesting patterns, and ultimately enabling an understanding of phenomena and structures on those networks in various contexts, e. g., [1]–[16]. In this context, data mining on such networks represented as attributed graphs has emerged as a prominent research topic recently, e. g., [5], [8], [10], [13]–[17]. 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., [18], [19].

Local pattern mining is an important approach for identifying communities, e. g., [5], [9], [10], [13], [14], [17], [20], focusing on the identification of dense substructures in a graph that are captured by specific patterns composed of the given attributes. In this context, different interestingness measures for identifying communities have been utilized, ranging from simple graph measures like focusing on cliques and quasi-cliques to more elaborate community quality indices like the Modularity measure introduced by Newman [21], [22], for which also variants focusing on local modularity structures have been introduced [14]. The latter then directly connects to local pattern mining approaches.

In this paper, we present *MinerLSD* a method for efficient local pattern mining on attributed graphs, focusing both on (local) community detection using the Modularity metric, as well as graph abstraction that reduces graphs to k-core subgraphs [13]. In order to prevent the typical pattern explosion in pattern mining, we employ closed patterns for focusing the pattern exploration both for the structural as well as the compositional perspective. In addition, we exploit optimistic estimates for the local modularity for pruning the pattern space. Essentially, the optimistic estimate technique provides two advantages: First, it enables a very efficient pattern exploration approach. Second, it neglects the importance of a minimal support threshold which is typically applied in pattern mining. As we will show below, given a suitable threshold for the local modularity, efficient pattern mining is enabled. Then, this threshold can of course alternatively be entirely eliminated in a top-k approach.

We perform experiments on several standard datasets, using two baselines for local pattern mining, in relation to our proposed novel pattern mining approach. We demonstrate on various datasets the efficacy of our presented novel method MinerLSD for local pattern mining on attributed graphs.

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 base 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 II discusses related work. Then, Section III presents the considered methods including the novel MinerLSD method. Next, Section IV introduces the applied datasets. Sections V-VI discuss our experimental results. Finally, Section VII concludes with a summary and interesting directions for future work.

Preprint of: Martin Atzmueller, Henry Soldano, Guillaume Santini and Dominique Bouthinon (2018)
MinerLSD: Efficient Local Pattern Mining on Attributed Graphs.
In: Proc. 2018 IEEE International Conference on Data Mining Workshops (ICDMW). Data Mining in Networks.

II. Related Work

The detection of local patterns is a prominent approach in knowledge discovery and data mining, e. g., [23]–[25]. Below, we specifically discuss related work in the areas of local pattern mining and community detection on attributed graphs.

A. Local Pattern Mining

In general, local pattern mining, e. g., [23]–[30] 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. 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. Closed Pattern Mining (see for instance [31]) 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., [32], [33]). Various algorithms and methodologies using closure operators have also been proposed in the domain of Formal Concept Analysis [34], which goes further than the enumeration alone, being interested in the lattice structure of the set of closed patterns [35].

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 [36]. 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 [37]. 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 consists, for example, in extracting k-communities [38] that are extracted from the connected components of a graph derived from the original graph.

Combining both ideas, recently an extension of the *closed pattern mining* methodology to attributed graphs has been proposed. It relies on the reduction of the support set X of a pattern to the core of the pattern subgraph G_X [39]. This results in less and larger classes of equivalent patterns, and hence less closed pattern. MinerLC is a generic method to enumerate the set of such core closed patterns [40].

Similar to the approaches discussed above, the proposed MinerLSD approach also utilizes closed patterns, and graph abstractions, i. e., core subgraphs, relying on the MinerLC method as a foundation. However, it extends this using optimistic estimate pruning using an interestingness measure for (local) community detection, as described in the next section. We perform an extensive evaluation of the impact of closed patterns, optimistic estimate, and core structures on the effort for mining attributed graphs.

B. Community Detection on Attributed Graphs

Community detection on attributed graphs connects pattern mining and community detection according to several interestingness measures or optimization criteria. Moser et al. [17], 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. [41] combine subspace clustering and dense subgraph mining, also interleaving quasi-clique and subspace construction, e. g., focusing on the densities of quasi-cliques concerning the graph structure.

Galbrun et al. [10] 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. [5] 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. [9] includes a two-step process of community detection and community description. A heuristic approach is applied for discovering the top-k communities. Pool et al. utilize 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. [14], [42] applies an adapted subgroup discovery [30], [43], [44] approach for community detection on attributed graphs. 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 [21], [22]. 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 pattern mining resulting in the MinerLSD algorithm. The result is a combination of efficient closed pattern mining with different selection strategies according to local Modularity and graph abstractions, as we will show below.

III. Two attributed network pattern mining methods

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 the standard terminology, q is a *pattern* that *occurs* in all element of W which is also called the *support set* or *extension* $\mathrm{ext}(q)$ 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 of the *minimum support* constraint that requires the size of $\mathrm{ext}(q)$ to be above some min_sup 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 $\mathrm{ext}(q) = \mathrm{ext}(q')$. 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., $\mathrm{ext}(f(q)) = \mathrm{ext}(q)$. Furthermore, since we consider as objects 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. MinerLC, described below, uses both closed pattern mining and core operators to reduced the solution set.

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 interestingness $m(W)$ above some threshold l. However such measures, for example, the local modularity, 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. This is the basis of COMODO, the second method described hereunder.

A. Mining Closed Patterns to Enumerate Core Subgraphs

MinerLC enumerates pairs (c, W) where G_W is the core subgraph of pattern c i. e., $W = p \circ \mathrm{ext}(c)$ 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 such 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.

Core closed pattern mining: The operator f that returns for any pattern q the closed pattern $f(q)$ is a *closure operator* (see below) defined by $f(q) = \mathrm{int} \circ p \circ \mathrm{ext}(q)$, for which the operators are defined as follows:

- The intersection operator $\mathrm{int}(X)$ 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 [45]. (2) Furthermore, core definitions rely on a monotone property of a vertex within an induced subgraph [46]. 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. 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 MinerLC [40]: it performs 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. An algorithmic description following this scheme is given in Section VI-A, where we present the novel *MinerLSD* algorithm.

B. Pruning Subgroup Discovery Using Optimistic Estimates

The COMODO algorithm[1] presented in [14] focuses on *description-oriented community detection* for discovering the top-k communities. Essentially, COMODO is based on an adapted subgroup discovery approach [42], [48], and also tackles typical problems that are not addressed by standard approaches for community detection such as pathological cases like small community sizes. COMODO utilizes optimistic estimates [44], [49], which are efficient to compute, in order to prune the search space significantly. For that, a number of standard community evaluation functions have been applied using optimistic estimates for an efficient approach. In summary, COMODO enumerate pairs (c, W) where G_W is the subgraph of pattern c. It selects top k subgraphs according to an interestingness measure m of the subgraph and uses 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.

One particular quality function is the Modularity [21], [22]. In the following, we summarize the main features of optimistic estimate pruning for graph structure interestingness measures

[1] http://www.vikamine.org [47]

in the context of community detection. When introducing these, we adopt the notation of [14] for the main concepts.

Overall, 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 $G = (V, E)$, for which we use the following notation:

- $n := |V|$, $m := |E|$,
- $m_W := |\{\{u, v\} \in E : u, v \in W\}|$ – the number of *intra-edges* of W, and
- $\bar{m}_W := |\{\{u, v\} \in E : |\{u, v\} \cap W| = 1\}|$, resulting in the number of *inter-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. 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., [14].

Overall, the *Modularity* MOD [21], [22], [50] 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 a pattern) can then be computed (cf., [14], [50], [51]) as follows:

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

For the above, an optimistic estimate for the *local modularity contribution* has been introduced in [14]. It can be derived based only on the number of edges m_W within the community:

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

For a detailed discussion, the derivation of the local measure, and the respective proofs, we refer to [14].

C. Similarities and Differences in Selected Patterns

Both the considered 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 it is necessary to remove isolated nodes, which is enabled by applying a 1-core graph abstraction.
- In the basic algorithm, COMODO does not enumerate closed patterns, the same subgroup may then be associated to several patterns. Therefore, 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 postprocessing is one of the standard postprocessing 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 should be run with 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.

Furthermore, the two methods 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 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.

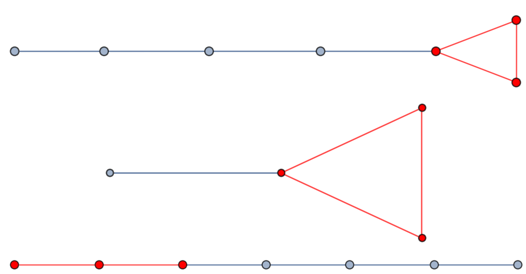

Fig. 1. Three graphs (top, center, bottom) each with a subgraph displayed in red. The two topmost subgraphs are 2-cores while the central subgraph has an empty 2-core.. The top and bottom graphs have local modularity above 0.15 while the central one as negative modularity -0.15.

IV. Datasets

We performed our experiments in a variety of attributed graphs ranging from small to medium graphs with small to large sets of items. Table I depicts the main characteristics of these datasets (see also [10]), 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[2] used in a previous work about graph abstractions [39]. 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 [39] 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 [52]. It concerns 71 attorneys (partners and associates) of this firm who are the vertices of four networks. In the resulting data[3], 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 ask 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 [53] program. In the graph representing the network, the vertices are co-regulators and they are linked if they share a common set of target genes. The vertices are labeled with their expression profile along a metabolic transition of the organism. Each influence value represents the regulation activity of the considered co-regulator. The influence is high whenever the level of expression of targeted genes is coherent with the role (activator or inhibitor) and level of expression of the considered co-regulator.
- LastFM and DBLP.C.ICDM (or DBLP.C for short) were used in Galbrun [10]. The first dataset models the social network of last.fm where individuals are described by the artists or groups they have listened to. The second 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.P was used in Bechara-Prado [54]. 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 corresponding to three clusters have been added to the original dataset based on a thematic partitioning 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 [55] at Recsys 2011.

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

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

TABLE I
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	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert L\rvert$	$\overline{deg(v)}$	$\overline{\lvert l(v)\rvert}$
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
DBLP.C	3140	10689	4588	6.81	15.02
DBLP.P	45131	228173	32	10.11	2.15
Delicious	1867	7664	52800	8.21	123.47

V. Experimental Comparisons on Local Modularity

A. Parameters and Datasets

We considered several rather small datasets using no minimal support parameters, a 1-core abstraction in MinerLC and parameters that do not limit the enumeration in COMODO. A post-processing step was added to MinerLC, resulting in MinerLC+P in order to select and count vertex subgroups whose induced subgraphs satisfy a local modularity threshold l. We also used a post-processing step of COMODO for the resulting pattern set in order to keep only the subset of closed patterns. The latter subset is obtained by considering all pairs (c, e) with same (vertex) subgroup e and only keeping the most specific ones. With this postprocessing COMODO returns exactly the same patterns as those output by MinerLC.

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

- #c the number of pairs (c, e)
- #lme: the number of pairs (c, e) such that $lme(e) \geq l$
- #nec: the number of pairs (c, e) a top-down search has to consider to ensure that no pair with $lm(e) \geq l$ is lost.
- #lm the number of pairs (c, e) such that $lm(e) \geq l$
- #lmeSD: the number of pairs (c, e) such that $lme(e) \geq l$ generated by COMODO.

B. Pruning: Efficiency of the local modularity estimate

The first experiment investigates how the local modularity constraint affects the number of output pairs. As lme is an optimistic estimator, we may consider the best possible optimistic estimator which would only develop the $\#nec$ nodes

[3]https://grouplens.org/datasets/hetrec-2011/

that have at least a descendant (c, e) with local modularity $lm(e) \geq l$. 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 lm that is much better than lme.

By computing these numbers, we can then state separately for each dataset whether the lme 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.

Below, Figure 2 depicts the results of the applied five datasets. Overall, we observe contrasted results. For instance, in the Lawyers dataset, MinerLC finds #c=3221 patterns at level l=0.005 and most of them, 2929, have an lme 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 with local modularity lm above 0.005. There is then a slow decrease of #lme while the decrease of #nec and #lm is much faster. In contrast, in the DBLP.C dataset, of the total #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 of datasets S50 and CoExp. The DBLP.C dataset displays a very efficient optimistic pruning and DBLP.P displays a similar behavior. Detailed results are shown in Table II.

TABLE II
NUMBER OF PATTERNS TOTAL, DEVELOPED, NECESSARY AND WITH REQUIRED LOCAL MODULARITY (ACCORDING TO THE RESPECTIVE THRESHOLD 0.005 ... 0.15).

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

C. Closure & Efficiency: Impact of closed patterns in reducing the search space

MinerLC searches a space of closed patterns while COMODO searches the whole pattern space. Therefore, we

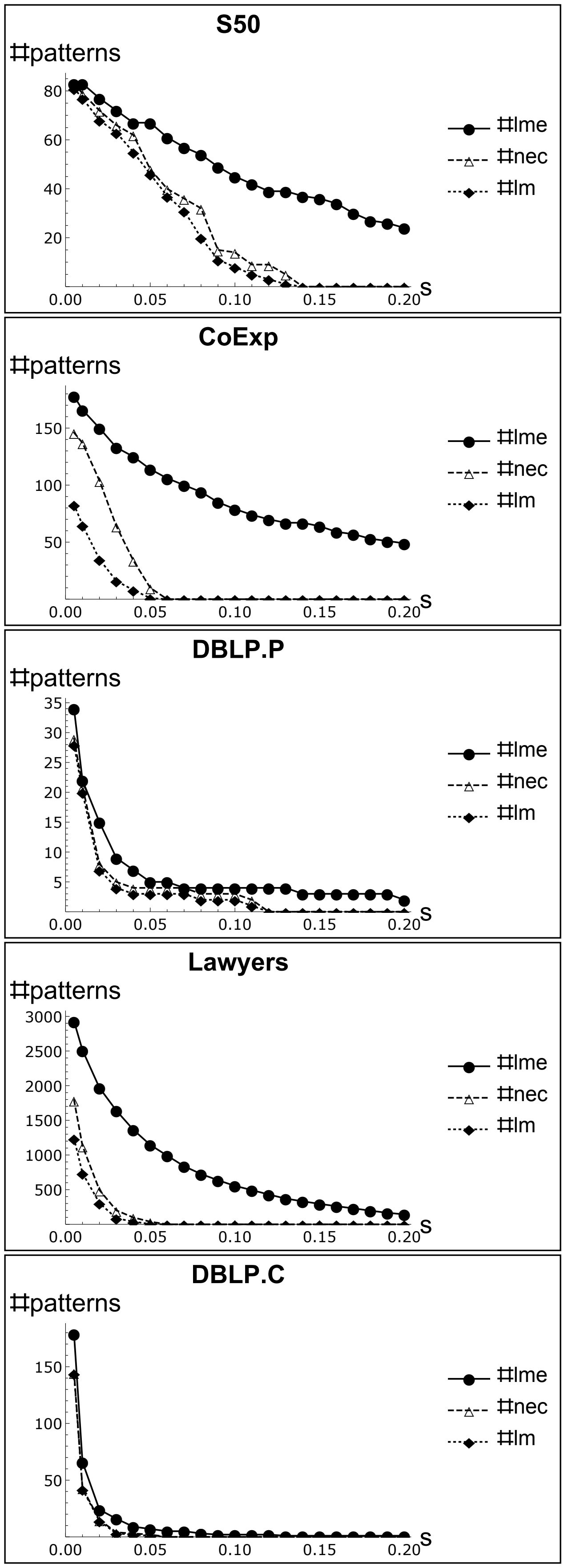

Fig. 2. Numbers of patterns with #lme, #nec and #lm values (on the Y-axis), above the local modularity threshold (on the X-axis) for 5 attributed networks.

will compare the impact of the closure reduction, for each local modularity threshold. For that, we consider the quantity #lme produced by MinerLC+post-processing to the quantity #lmeSD produced by COMODO. Table III reports #lme and #lmeSD for our datasets under investigation.

Again, we observe two very different situations. In the Lawyers and CoExp datasets there is a large difference between #lmeSD and #lme, while there are not so strong differences in the other datasets. 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 specialisation 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 #meSD are much closer, and even identical regarding the DBLP.C dataset.

TABLE III
NUMBER OF PATTERNS TO DEVELOP IN MINERLC (FROM POST-PROCESSING) AND COMODO (ACCORDING TO THE RESPECTIVE LOCAL MODULARITY THRESHOLD 0.005 ... 0.15).

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

D. Comparing k-core reduction to local modularity reduction

In the following, we investigate the effect of applying graph abstractions, i. e., a k-core abstraction to the local modularity in terms of reducing the exploited pattern space.

Basically, reducing the number of selected patterns is performed by MinerLC by applying the k-core constraint. For the COMODO algorithm, it is implemented by requiring the respective local modularity values of the patterns to exceed a given threshold. Both ideas result in strongly reducing the pattern set when applied together with strong constraints. We investigate hereunder two large datasets, namely the LastFM and Delicious datasets. Using a 1-core abstraction and no post-processing MinerLC returns a large number #c of closed patterns. Then, we run MinerLC applying k-core constraints and compare the size of the closed pattern to the size of the closed pattern set obtained by COMODO combined with a (postprocessing) selection of the closed patterns with a given local modularity thresholds l. When no constraint (outside the 1_core) is applied, MinerLC finds 1,555,292 and 11,833,577 closed patterns, respectively, and so as many subgraphs and subgroups.

In Table IV, we show the size of the closed pattern sets obtained with various k and l parameters. We also report results from the Lawyers dataset which is smaller but denser than the previous ones. A first and expected remark is that these reductions depend on the dataset. When considering the same parameters, the reduction is always stronger in Delicious than in LastFM. A second remark is that we obtain strong reductions in pattern sets even with relatively mild requirements. Combining both reductions without any post-processing is appealing as it should allow to address larger and denser datasets without performing any post-processing. The two kinds of constraints are of different nature and in the last column of the table a post-processing is applied to the 6-core and 8-core results of MinerLC in order to select closed patterns with local modularity of at least 0.02.

Applying both constraints results in a stronger reduction for all datasets. This is observed in the last two columns of the table where a post-processing is applied to the 6-core and 8-core results of MinerLC in order to select closed patterns with local modularity of at least 0.02.

TABLE IV
REDUCTION ON CLOSED PATTERN SET SIZE WITH k-CORES AND LOCAL MODULARITY THRESHOLDS l

Data	k= 4	k= 6	k=8	6, 0.02	8, 0.02
Last.	1,555,292 61560	12066	3031	3085	2503
Deli.	11,833,577 2150	193	44	22	9
Law.	3221 800	274	83	104	37

VI. MINERLSD: K-CORE AND LOCAL MODULARITY CONSTRAINED SEARCH

In the following, we first outline our proposed novel approach *MinerLSD*. We introduce and discuss the algorithm in detail. After that, we present results of our experiments applying MinerLSD for k-Core and local modularity constrained pattern mining.

A. MinerLSD

The algorithm that we describe below is basically an adapted version of MinerLC; we extended this algorithm by adding optimistic estimate pruning according to lme and pattern selection according to lm. As input (parameters), it 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 $p(X) = W$ is associated 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 and a local modularity threshold l. 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.

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[4].

MinerLSD (G, I, D, p, s, l)
  #lme← #lm← 0
  $W \leftarrow p(V)$
  // also defines the associated core subgraph $C = G_W$
  if $\mid W \mid < s$ **or** $\mathrm{lme}(W) < l$ **then exit**
  $\mathrm{enum}(\mathrm{int}(W), C, \emptyset)$ // $\mathrm{int}(W)$ *is the closure of* $\emptyset$

Function $\mathrm{enum}(c, C, EL)$
ensure: outputs the frequent (c', W') pairs where $c' \supseteq c$ and contains no items of EL
  Increase #lme
  if $\mathrm{lm}(C) \geq l$ **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** $\mathrm{lme}(W) \geq l$ **and** $c \cap \mathrm{EL} = \emptyset$ **then**
      $\mathrm{enum}(c, C^x, EL)$
      // enumerate the subtree rooted on c
      $\mathrm{EL} \leftarrow \mathrm{EL} \cup \{x\}$
    end if
  end for

Function $\mathrm{int}(W)$
  return $\cap_{v \in W} D(v)$

As MinerLC, MinerLSD ensures that each pair (c, W) is enumerated once. It has been implemented starting from the *MinerLC* original sources[5]and therefore uses the same dataset reduction techniques to reduce the subgraphs during the depth-first traversal of the pattern space. This means that we may have a fair comparison regarding respective computation costs of MinerLC and MinerLSD.

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

B. Experiments

MinerLSD detects the same closed patterns as MinerLC with post-processing, but with the benefit of pruning using the $lme \geq l$ condition, i. e., only developing the #lme nodes according to Table III.

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 1-core and no local modularity thresholds – these were not investigated in Tables II and III, respectively.

We performed experiments using 1-cores, 2-cores and 3-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 3.

The benefit of applying local modularity constraints in the resulting number of closed patterns is, as expected, quite impressive. In the LastFM case there are no strong differences when using 1-cores, 2-cores and 3-cores while we know from Table IV that using 4-cores does have an important effect. 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-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, in the Delicious case, the benefit is obvious: even when not considering the post-processing costs MinerLSD is always much faster than MinerLC. The last.fm dataset shows a somewhat different picture: with 1-cores and at local modularity level of 0.01 MinerLC (which does not consider local modularity at all) 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 nodes. 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), have to be made anyway at the post-processing stage. Detailed results are presented in Table V which also displays the #lme numbers.

VII. Conclusions

In this paper, we have investigated approaches for efficient local pattern mining on attributed graphs. For that, we had a look at different options, including closed pattern mining, optimistic estimate pruning using local modularity, and applying graph abstractions. In particular, we have proposed a novel method called MinerLSD for enumerating new local patterns and associated subgroups in attributed graphs.

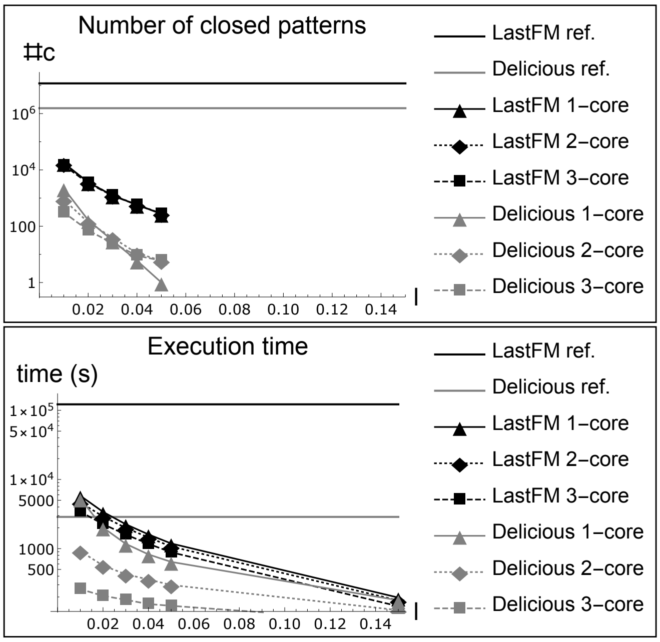

Fig. 3. Numbers of patterns and execution time of MinerLSD on Delicious and LastFM datasets with 1-cores, 2-cores and 3-cores and local modularity thresholds ranging from 0.01 to 0.15. The Y-axis of the topmost figure represents the number of closed patterns outptut by MinerLSD while the bottom figure displays the CPU time. Both Y-axis are displayed using a logarithmic scale. As a reference we also display on each figures horizontal lines representing the results of MinerLC with 1-cores.

TABLE V
MINERLSD #LM, #LME AND EXECUTION TIME COMPARED TO #C OF MINERLC FOR SAME CORE CONSTRAINT

LastFM	1-core	#c= 1555292		time= 2874		
l	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					
l	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					
l	0.01	0.02	0.03	0.04	0.05	0.15
#lme	39127	12694	5753	3039	1720	50
#lm	14637	3377	1219	572	276	0
time (s)	3422	2262	1596	1174	885	147
Delicious	1-core	#c= 11833577		time= 121934		
l	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					
l	0.01	0.02	0.03	0.04	0.05	0.15
#lme	1421	288	116	65	37	3
#lm	879	138	39	11	6	0
time (s)	920	569	426	358	298	129
	3-core					
l	0.01	0.02	0.03	0.04	0.05	0.15
#lme	430	114	51	25	17	1
#lm	311	71	25	9	6	0
time (s)	259	208	182	158	149	87

MinerLSD is based on two methods – MinerLC and COMODO: From MinerLC we kept the idea of 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. We also kept the idea of reducing pattern subgraphs to core subgraphs which allows both to strongly reduce the number of patterns and to focus on essential part of graphs. From COMODO, we borrowed the idea of selecting 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. First, we presented the results of several experiments using the two basic approaches, i. e., MinerLC and COMODO, where we applied some post-processing for an overall analysis. The purpose was then to investigate i) the pruning efficiency of MinerLC using the local modularity estimate as implemented in COMODO, ii) the impact of searching for closed patterns and therefore enumerating only the cohesive subgraph associated to patterns, and iii) the added selection potential obtained by combining both k-core reduction and local modularity selection.

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. Then, the results of these studies were above all a motivation to implement our novel proposed method, named MinerLSD, since it was built from the sources of MinerLC, that allows both closure, optimistic estimate pruning and k-core reduction during the search, and COMODO which is based on an adapted subgroup discovery (SD) approach.

The further experiments using MinerLSD show the efficiency of the presented method, and in particular that the extra computational steps which were added to MinerLC do not harm the overall computational costs, even when applying weak constraints, while strongly reducing the pattern set sizes. Overall, mixing these different ideas and constraints we obtain a very flexible tool that allows to handle large graphs with adequate constraints on the subgroups 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.

VIII. ACKNOWLEDGEMENTS

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

References

[1] M. E. J. Newman, "The Structure and Function of Complex Networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003.

[2] R. Kumar, J. Novak, and A. Tomkins, "Structure and Evolution of Online Social Networks," in *Proc. ACM SIGKDD*. ACM, 2006, pp. 611–617.

[3] J. A. Almendral, J. Oliveira, L. Lpez, J. Mendes, and M. A. Sanjun, "The Network of Scientific Collaborations within the European Framework Programme," *Physica A: Statistical Mechanics and its Applications*, vol. 384, no. 2, pp. 675 – 683, 2007.

[4] F. Mitzlaff, M. Atzmueller, D. Benz, A. Hotho, and G. Stumme, "Community Assessment using Evidence Networks," in *Analysis of Social Media and Ubiquitous Data*, ser. LNAI, vol. 6904, 2011.

[5] A. Silva, W. Meira Jr, and M. J. Zaki, "Mining Attribute-Structure Correlated Patterns in Large Attributed Graphs," *Proc. VLDB Endowment*, vol. 5, no. 5, pp. 466–477, 2012.

[6] M. Atzmueller and F. Lemmerich, "Exploratory Pattern Mining on Social Media using Geo-References and Social Tagging Information," *IJWS*, vol. 2, no. 1/2, pp. 80–112, 2013.

[7] F. Mitzlaff, M. Atzmueller, G. Stumme, and A. Hotho, "Semantics of User Interaction in Social Media," in *Complex Networks IV*, ser. SCI. Springer, 2013, vol. 476.

[8] M. Atzmueller, "Data Mining on Social Interaction Networks," *Journal of Data Mining and Digital Humanities*, vol. 1, June 2014.

[9] S. Pool, F. Bonchi, and M. van Leeuwen, "Description-driven Community Detection," *TIST*, vol. 5, no. 2, 2014.

[10] E. Galbrun, A. Gionis, and N. Tatti, "Overlapping Community Detection in Labeled Graphs," *DMKD*, vol. 28, no. 5-6, pp. 1586–1610, Sep. 2014.

[11] F. Mitzlaff, M. Atzmueller, A. Hotho, and G. Stumme, "The Social Distributional Hypothesis," *SNAM*, vol. 4, no. 216, 2014.

[12] M. Kibanov, M. Atzmueller, C. Scholz, and G. Stumme, "Temporal Evolution of Contacts and Communities in Networks of Face-to-Face Human Interactions," *Sci. Ch. Inf. Sci*, vol. 57, no. 3, pp. 1–17, 2014.

[13] H. Soldano, G. Santini, and D. Bouthinon, "Local Knowledge Discovery in Attributed Graphs," in *Proc. ICTAI*. IEEE, 2015, pp. 250–257.

[14] M. Atzmueller, S. Doerfel, and F. Mitzlaff, "Description-Oriented Community Detection using Exhaustive Subgroup Discovery," *Information Sciences*, vol. 329, pp. 965–984, 2016.

[15] A. A. Bendimerad, M. Plantevit, and C. Robardet, "Unsupervised Exceptional Attributed Subgraph Mining in Urban Data," in *Proc. ICDM*. Washington, DC, USA: IEEE, 2016, pp. 21–30.

[16] M. Kaytoue, M. Plantevit, A. Zimmermann, A. Bendimerad, and C. Robardet, "Exceptional Contextual Subgraph Mining," *Machine Learning*, pp. 1–41, 2017.

[17] F. Moser, R. Colak, A. Rafiey, and M. Ester, "Mining Cohesive Patterns from Graphs with Feature Vectors," in *Proc. SDM*, 2009, pp. 593–604.

[18] M. Atzmueller, "Compositional Subgroup Discovery on Attributed Social Interaction Networks," in *Proc. International Conference on Discovery Science*, 2018.

[19] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, 1st ed., ser. Structural analysis in the social sciences. Cambridge University Press, 1994, no. 8.

[20] M. Atzmueller, "Detecting Community Patterns Capturing Exceptional Link Trails," in *Proc. IEEE/ACM ASONAM*. Boston, MA, USA: IEEE Press, 2016.

[21] M. E. J. Newman, "Detecting Community Structure in Networks," *Europ Physical J*, vol. 38, 2004.

[22] M. E. Newman and M. Girvan, "Finding and Evaluating Community Structure in Networks," *Phys Rev E Stat Nonlin Soft Matter Phys*, vol. 69, no. 2, pp. 1–15, 2004.

[23] K. Morik, "Detecting Interesting Instances," in *Pattern Detection and Discovery*, ser. Lecture Notes in Computer Science, D. Hand, N. Adams, and R. Bolton, Eds. Springer, 2002, vol. 2447, pp. 13–23.

[24] K. Morik, J. Boulicaut, and A. Siebes, Eds., *Local Pattern Detection, International Seminar, Dagstuhl Castle, Germany, April 12-16, 2004, Revised Selected Papers*, ser. LNCS, vol. 3539. Springer, 2005.

[25] A. J. Knobbe, B. Cremilleux, J. Fürnkranz, and M. Scholz, "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)*, 2008, pp. 1 – 16.

[26] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in *Proc. VLDB*. Morgan Kaufmann, 1994, pp. 487–499.

[27] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns Without Candidate Generation," in *Proc. ACM SIGMOD*. ACM Press, 05 2000, pp. 1–12.

[28] M. Atzmueller, *Knowledge-Intensive Subgroup Mining – Techniques for Automatic and Interactive Discovery*. IOS Press, 2007.

[29] F. Lemmerich, M. Atzmueller, and F. Puppe, "Fast Exhaustive Subgroup Discovery with Numerical Target Concepts," *Data Mining and Knowledge Discovery*, vol. 30, pp. 711–762, 2016.

[30] M. Atzmueller, "Subgroup Discovery – Advanced Review," *WIREs: Data Mining and Knowledge Discovery*, vol. 5, no. 1, pp. 35–49, 2015.

[31] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Efficient Mining of Association Rules using Closed Itemset Lattices," *Information Systems*, vol. 24, no. 1, pp. 25–46, 1999.

[32] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases," in *Proc. International Conference on Discovery Science*, 2004, pp. 16–31.

[33] M. Boley, T. Horváth, A. Poigné, and S. Wrobel, "Listing Closed Sets of Strongly Accessible Set Systems with Applications to Data Mining," *Theor. Comput. Sci.*, vol. 411, no. 3, pp. 691–700, 2010.

[34] R. Wille, "Restructuring Lattice Theory," in *Symposium on Ordered Sets*, University of Calgary, Boston, 1982, pp. 445–470.

[35] B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*. Springer Verlag, 1999.

[36] S. B. Seidman, "Network Structure and Minimum Degree," *Social Networks*, vol. 5, pp. 269–287, 1983.

[37] V. Batagelj and M. Zaversnik, "Fast Algorithms for Determining (Generalized) Core Groups in Social Networks," *Adv. Data Analysis and Classification*, vol. 5, no. 2, pp. 129–145, 2011.

[38] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek, "Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society," *Nature*, vol. 435, no. 7043, pp. 814–818, Jun 2005.

[39] H. Soldano and G. Santini, "Graph Abstraction for Closed Pattern Mining in Attributed Networks," in *Proc. ECAI*, ser. Frontiers in Artificial Intelligence and Applications, vol. 263. IOS Press, 2014, pp. 849–854.

[40] H. Soldano, G. Santini, D. Bouthinon, and E. Lazega, "Hub-Authority Cores and Attributed Directed Network Mining," in *Proc. ICTAI*. Boston, MA, USA: IEEE Computer Society, November 7-9 2017.

[41] S. Günnemann, I. Färber, B. Boden, and T. Seidl, "GAMer: A Synthesis of Subspace Clustering and Dense Subgraph Mining," in *KAIS*, vol. 40, no. 2. Springer, 2013, pp. 243–278.

[42] M. Atzmueller and F. Mitzlaff, "Efficient Descriptive Community Mining," in *Proc. FLAIRS*. AAAI Press, 2011, pp. 459 – 464.

[43] W. Klösgen, "Explora: A Multipattern and Multistrategy Discovery Assistant," in *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 249–271.

[44] S. Wrobel, "An Algorithm for Multi-Relational Discovery of Subgroups," in *Proc. PKDD*, 1997, pp. 78–87.

[45] N. Pernelle, M.-C. Rousset, H. Soldano, and V. Ventos, "ZooM: A Nested Galois Lattices-Based System for Conceptual Clustering," *J. Exp. Theor. Artif. Intell.*, vol. 2/3, no. 14, pp. 157–187, 2002.

[46] V. Batagelj and M. Zaversnik, "Generalized Cores," *CoRR*, vol. cs.DS/0202039, 2002.

[47] M. Atzmueller and F. Lemmerich, "VIKAMINE - Open-Source Subgroup Discovery, Pattern Mining, and Analytics," in *Proc. ECML/PKDD*. Springer, 2012.

[48] F. Lemmerich, M. Becker, and M. Atzmueller, "Generic Pattern Trees for Exhaustive Exceptional Model Mining," in *Proc. ECML/PKDD*, ser. LNCS, vol. 7524. Springer, 2012, pp. 277–292.

[49] H. Grosskreutz, S. Rüping, and S. Wrobel, "Tight Optimistic Estimates for Fast Subgroup Discovery," in *Proc. ECML/PKDD*, ser. LNCS, vol. 5211, 2008, pp. 440–456.

[50] M. E. J. Newman, "Modularity and Community Structure in Networks," *PNAS*, vol. 103, no. 23, pp. 8577–8582, 2006.

[51] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri, "Extending the Definition of Modularity to Directed Graphs with Overlapping Communities," *J. Stat. Mech.*, p. 03024, 2009.

[52] E. Lazega, *The Collegial Phenomenon: The Social Mechanisms of Cooperation Among Peers in a Corporate Law Partnership*. Oxford University Press, 2001.

[53] R. Nicolle, F. Radvanyi, and M. Elati, "Coregnet: Reconstruction and Integrated Analysis of Co-Regulatory Networks," *Bioinformatics*, 2015.

[54] A. Bechara Prado, M. Plantevit, C. Robardet, and J.-F. Boulicaut, "Mining Graph Topological Patterns: Finding Co-variations among Vertex Descriptors," *IEEE TKDE*, vol. 25, no. 9, pp. 2090–2104, Sep. 2013.

[55] I. Cantador, P. Brusilovsky, and T. Kuflik, "2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec)," in *Proc. RecSys*. New York, NY, USA: ACM, 2011.