2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Unexpected Attributed Subgraphs: a Mining Algorithm

Simon Delarue
LTCI, Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
simon.delarue@telecom-paris.fr

Tiphaine Viard
i3, Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
tiphaine.viard@telecom-paris.fr

Jean-Louis Dessalles
LTCI, Télécom Paris
Institut Polytechnique de Paris
Palaiseau, France
jean-louis.dessalles@telecom-paris.fr

***Abstract*—Graphs are ubiquitous in real-world data, ranging from the study of social interactions to bioinformatics or the modelling of physical systems. These *real-world* graphs are typically sparse, possibly large and frequently contain additional information in the form of attributes, making them a complex object to understand. Graph summarization techniques can help facilitate the discovery of hidden patterns in underlying data by providing an *interesting* subset of the interactions and available attributes, which we broadly call a *pattern*. However, determining what is considered interesting in this context is not straightforward. We address this challenge by designing an interestingness measure based on the information-theoretic measure of *Unexpectedness*, linking the concepts of relevance and Kolmogorov complexity. We design a pattern mining algorithm to provide a summary of the initial data in the form of a set of *unexpected patterns*, that is, patterns for which there is a drop between their expected complexity and the observed complexity. Experimental results on five real-world datasets with state-of-the-art methods demonstrate that our method exhibits a small number of diversified patterns, providing a human-readable summary of the initial attributed graph; we show that our summaries quantitatively outperforms attribute-only and interaction-only baselines as well as other pattern mining methods, reinforcing the need for methods dealing with attributed graphs. We visualize summaries extracted with our method, in order to qualitatively validate their readability.**

***Index Terms*—Complex networks, Pattern Mining, Information Theory**

I. Introduction

Graphs, *i.e.* sets of nodes linked with edges between them, provide a powerful framework for modelling interactions among entities. It is common to encounter such graphs in the real-world: Web or Wikipedia pages u and v are linked together if u contains a hyperlink to v, individuals u and v are linked if they interacted together (for example, on a social network), etc. In this setting, it is natural to model the data as an attributed graph, *i.e.* a graph where the nodes are enriched with additional information: web pages have contents, individuals have personal chracteristics such as their gender, their tastes, etc. However, studying such large attributed graphs becomes complex without the aid of methods that enable the summarization of the information they contain. Finding summaries that facilitate information discovery while remaining computationally tractable is a difficult problem. In his work, Miller argues that "good" explanations tend to be contrastive, selected (including in a biased manner), causal rather than probabilistic, and *dialogic*, in that they represent a transfer of knowledge [1]. Graphs, as high-dimensional objects representing relations at different scales, aptly model causal information and can be explored interactively through the selection of relevant subgraphs, making them an object of choice for data exploration [2]. We are interested in the problem of mining such summaries, in the form of subgraphs.

We are interested in the problem of mining such summaries, in the form of subgraphs. Our objective is to offer a practical tool to facilitate the discovery of meaningful graph representations. The applications of such tools span a wide range, including anomaly detection (*e.g.* for the fight against money laundering), enhancing machine learning explainability, and conducting social network analysis.

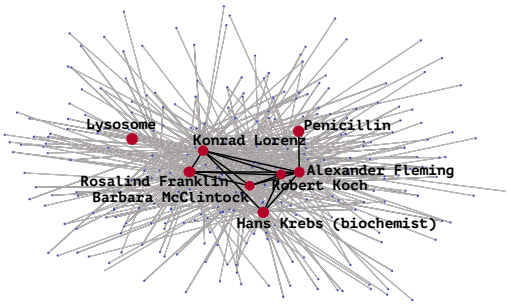


Fig. 1: Within a graph of Wikipedia pages, the pattern highlighted in red (nodes) and black (links) is unexpected: article `Lysosome` is not directly linked to the other nodes in the pattern, yet it shares numerous common characteristics (neighbours and attributes) with them. *In addition to the pattern, only the neighbours of nodes in the pattern (small and blue) and the links to them (grey) are displayed*

ASONAM '23, November 6-9, 2023, Kusadasi, Turkey

ACM ISBN 979-8-4007-0409-3/23/11…$15.00
https://doi.org/10.1145/3625007.3627314

Relying on the *homophily* assumption [3], *i.e.*entities that share attributes are more likely to be connected in the graph, numerous studies have focused on the development of community detection algorithms in attributed graphs [4]–[6] to provide subgraphs that form a partition of the original data. However, the communities obtained are often dense and there is nothing to prevent redundancy among attributes. In contrast, pattern mining techniques [7], [8] search for small relevant subgraphs for a given task: compression, exploration, summarization, etc. Since there can be at most an exponential number of patterns, multiple strategies have been developed to make the problem tractable, *e.g.* pruning the search space using *support* constraints [9] or density thresholds [7], but also using *interestingness* measures for ranking and selecting relevant patterns from a user standpoint. Defining a measure of interest that reduces the redundancy of results while maintaining an informative summary of the original data is challenging in itself. Moreover, following previous works [10], [11], we argue that there is more to interestingness than subgroup density, such as deriving from a surprising event.

In this work, we propose to summarize attributed graphs considering both the homophily assumption and an interestingness measure relying on the information-theoretic metric of *Unexpectedness* [12] defined within the Simplicity Theory (ST) [13], [14]. ST claims that at the human level, unexpectedness arises when an event seems overly simple compared to *normal* behaviour. In this context, the unexpectedness of an event is defined as the difference between its expected complexity, or the length of the minimal description in number of bits, under *normal* circumstances and the complexity to properly describe this event.

We extend *Unexpectedness* to attributed subgraphs and use this definition to identify the patterns most likely to interest the user. Informally, if a pattern can be described in a shorter manner than what was expected, it means that a compression of the information it contains, either on its structure or attributes, was possible and thus reveals an objective interest (see Fig. 1 for an example). Implemented in a subgraph discovery algorithm, this measure also allows us to efficiently prune the search space as soon as adding new elements to a pattern does not come with a gain in unexpectedness. This approach enables us to detect patterns that are both smaller in size and connectivity compared to state-of-the-art pattern mining approaches. Finally, leveraging the work from [15], [16], we build a metric that reflects *expressiveness* of patterns through commonly used criteria: diversity, coverage, and size of subgraphs. We show that our approach performs well in summarizing large attributed graphs in a non-redundant, concise and human-friendly way.

II. Related Work

Many works in graph mining initially focused on community discovery [17]. Several methods have since been developed to take advantage of the availability of attributes in graphs. This is the case of the mining of *cohesive patterns* [7], *i.e.* subgraphs that are neither a partition nor a cover of the entire graph but that are sufficiently dense and share common attributes. However, this method provides no guarantee regarding the redundancy of found patterns. To tackle this challenge, authors in [18] combine subspace clustering and dense subgraph mining and handle redundancy by excluding subgraphs if they are less qualitative in terms of density and size. In our work, we do not limit ourselves to the notion of density for identifying subgraphs; instead, we select them based on their relevance to users' interest.

Various works have followed this direction and focused on mining *exceptional* subgraphs rather than homogeneous [19]–[22]. To better include user prior knowledge about data, researchers in [23], [24] suggested that interestingness measures should be *subjective*, as interesting patterns are often driven by predefined assumptions from the user. More recent work has built on this idea, including the design of background distributions using information content and the Minimal Description Length (MDL) of patterns to model analysts' prior beliefs [25]–[29]. The main difference in our approach is that we opt for an objective definition of a user's interest, *i.e.* we consider a pattern to be interesting in absolute terms. Despite the link between our method and the MDL approach, we do not seek here to evaluate combinations of patterns to find a compression model, but rather score individual patterns according to a complexity drop. Recently, authors in [30] proposed an objective interestingness measure to detect patterns in heterogeneous networks based on structure and attribute pre-defined conditions, *e.g.* diversity applied to handcrafted subgraphs. However, we argue that for maximum objectivity, a metric of interest should not rely on user-defined rules. Therefore, in this work we solely employ diversity to evaluate our results.

At the intersection of community detection and pattern mining, Focal Structure Analysis methods [31] aim to extract key sets of entities that have the most influence in social networks. However, these approaches only consider custom subjective centrality metrics and do not include node attributes in their analysis.

Alongside these research efforts, work has been carried out to generate attributed graph summaries, *i.e.* concise and understandable representations of large and complex datasets [32]. These methods have much in common with pattern mining research, notably the use of measures of interest for the construction of summaries. However, they emphasise the explainability and comprehensibility of the results. Naturally, some work has been done to bridge the gap between the two fields and numerous methods have been developed in order to mine interesting patterns while providing concise descriptions of the results [16], [33], [34]. In all cases, the authors mainly focus on metrics such as diversity or coverage of the attributed graph for interestingness and only consider dense summaries as relevant to the user. In this paper, we consider a similar problem but we define a broader metric of interest than subgraph density, and use diversity and coverage (among others) only as a means of evaluating our results.

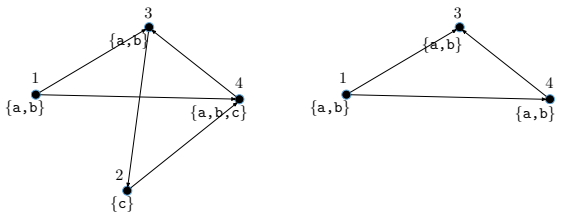


Fig. 2: On the left, an attributed graph $\mathcal{G} = (V, E, A)$, with $V = \{1, 2, 3, 4\}$, $E = \{(1, 3), (4, 3), (1, 4), (3, 2), (2, 4)\}$ and $A = \{a, b, c\}$. On the right, the pattern $(\{1, 3, 4\}, \{(1, 3), (3, 4), (1, 4)\}, \{\texttt{a}, \texttt{b}\})$.

III. Mining Unexpected Patterns

A. Problem Definition

We consider attributed graphs as sets of nodes with existing relations and associated features, such as vectors of attributes holding auxiliary information. For instance, a graph of citations could have nodes as articles and attributes as the words in the articles' abstracts. More formally, an *attributed graph* is a tuple $\mathcal{G} = (V, E, A)$ where V is a set of vertices, $E \subseteq V \times V$ a set of directed edges, and A a set of attributes (see Fig. 2 for an example). For any node $v \in V$, we denote by $a(v) \subseteq \mathcal{P}(A)$ the set of attributes carried by the node v, and by D_A the density of relations between nodes and attributes, *i.e.* $D_A = \frac{\sum_{v \in V} |a(v)|}{|V| \cdot |A|}$. Furthermore, for a set of attributes $B \subseteq A$, we define its *extension* as the set of nodes sharing all attributes in B: $ext(B) = \{v \in V : \forall i \in B, i \in a(v)\}$. Similarly, for a set of nodes $X \subseteq V$, we define its *intension* as the set of attributes shared by all nodes in X, $int(X) = \{i : \forall v \in X, i \in a(v)\}$. To simplify notation, we use $ext(b)$ instead of $ext(\{b\})$ for a single attribute $b \in A$, and $int(v)$ instead of $int(\{v\})$ for a node $v \in V$. In this context, we give the formal definition of a *pattern* in definition 1 and provide an example in Fig 2.

Definition 1 (Pattern)**.** *Given an attributed graph $\mathcal{G} = (V, E, A)$, a* pattern *$p = (X, E \cap (X \times X), Q)$ is an attributed subgraph, with $X \subseteq V$ a set of nodes and $Q \subseteq A$ a set of attributes such that $ext(Q) = X$,* i.e. *all nodes of X share all attributes in Q.*

Notice that we do not require that $int(X) = Q$ to define a pattern, and so this object differs from subgraphs induced by closed itemsets as studied in the field of Formal Concept Analysis. Given an attributed graph $\mathcal{G}$ and definition 1, our goal is to find a set of patterns that summarize $\mathcal{G}$ in an explainable and concise manner. We do not restrict ourselves to non-overlapping subgraphs, nor do we require that the patterns cover the entire graph.

B. Unexpectedness As Interestingness Measure

In order to mine *interesting* patterns, we introduce a measure of interest that allows comparison among patterns, using the notion of *Unexpectedness* [12] defined within Simplicity Theory (ST) [13], [14] (see definition 2). It claims that an event seems unexpected to a human if it appears *"too simple"*, *i.e.* there is a drop between its expected complexity and its observed complexity (both in number of bits). To illustrate this theory, [14] offers the following example: one can think about how unexpected a sequence from 1 to 6 would be as an output of a lottery draw (although the probabilities of each draw are the same). ST explains this surprise by comparing the expected complexity of a random draw (high) to the complexity to describe the sequence (low, thanks to the obvious pattern it contains).

Definition 2 (Unexpectedness)**.** *The Unexpectedness U of an event is the difference between its expected complexity C_{exp} and its observed complexity C_{obs}. Formally, $U = C_{exp} - C_{obs}$.*

In definition 2, C_{exp} is the complexity, or the length of the minimal description, that can be expected from an event, having some prior knowledge of the context in which this event takes place. Observed complexity C_{obs} on the other hand, is the amount of information required by an observer to describe the event uniquely among others.

In our approach, we consider patterns as events and use *Unexpectedness* to select the most interesting ones. We introduce two *compressors* to express both pattern graph structure and attributes using the length of their minimal descriptions. In the following, we use the superscripts $(g), (a)$ and (p) to denote the notions from definition 2 used on graph structure (*i.e.* nodes and edges), attributes or whole pattern respectively.

1) Graph Compressor: As per [35], we define the observed complexity of an arbitrary graph $G = (V, E)$ as the number of bits required to describe its nodes and edges, $C^{(g)}_{obs}(G) = \log_2 |V| + \log_2(b + 1) + \sum_{v \in V} \log_2 \left(\binom{|V|}{d_v}\right)$ with d_v the degree of node $v \in V$ and $b = \max_{v \in V} d_v$. The first term is the amount of bits required to describe all nodes. The second term describes the encoding of d_v values and the last term is used to encode the rows of G's adjacency matrix, where each row is encoded by a string of length $|V|$ containing d_v 1s and $(|V| - d_v)$ 0s. This is relevant for graphs where $d_v \ll (|V| - d_v)$, which is typically the case for real-world graphs [36].

The expected complexity refers to what a user can expect in terms of size and connectivity for a subgraph sampled from $\mathcal{G}$. For a given number of nodes, the expected complexity follows a concave function: it is minimal if there are no edges, grows until its apex, when the nodes are connected at random, and decreases as all nodes form a clique. Given a number of nodes n and a set of sampled graphs $\mathbb{G}_n$ of size n, we define this complexity as the average length of the minimal description of subgraphs of size n sampled in $\mathcal{G}$, $C^{(g)}_{\text{exp}}(n) = \sum_{G \in \mathbb{G}_n} C^{(g)}_{\text{obs}}(G)/|\mathbb{G}_n|$, where $\mathbb{G}_n = \{(X, E \cap (X \times X)) : \exists Q \subseteq A, X = ext(Q), |X| = n\}$. In the context of itemset mining, constructing $\mathbb{G}_n$ involves identifying all sets of attributes with a support size of exactly n. Finally, we define the unexpectedness of a graph G of size n as:

$$U^{(g)}(G) = C^{(g)}_{exp}(n) - C^{(g)}_{obs}(G) \tag{1}$$

2) Attribute Compressor: Considering a set of presumed independent attributes, denoted as A, our approach involves an iterative process of selecting elements from A to form a subset $Q \subseteq A$. This subset is then used to derive patterns (further detailed in III-C). At each step, the user can expect some complexity from the selection of $q \in A$, which is related to the frequency of q among all nodes in V. We define this complexity as $C^{(a)}_{exp}(Q \cup \{q\}) = C^{(a)}_{exp}(\{q\}) = -\log_2(f_q)$ with f_q the frequency of attribute q among all $v \in V$.

On the other hand, the complexity of observing the set of attributes $Q \cup \{q\}$ can be seen as the complexity of observing changes in its corresponding extension, *i.e.* adding q to Q eventually makes the induced subgraph of the new set more specific. Thus, the observed complexity of $Q \cup \{q\}$ is the ratio between the number of nodes in its extension, before and after adding q, $C^{(a)}_{\text{obs}}(Q \cup \{q\}) = -\log_2(\frac{|ext(Q \cup \{q\})|}{|ext(Q)|})$. In other words, describing a set of attributes that induced a significant decrease in the size of its extension is more complex than if this extension remained unchanged. The unexpectedness of adding q to Q is then:

$$U^{(a)}(Q \cup \{q\}) = U^{(a)}(Q) + C^{(a)}_{\text{exp}}(Q \cup \{q\}) - C^{(a)}_{\text{obs}}(Q \cup \{q\}) \quad (2)$$

In the following, adding an uncommon attribute (*i.e.* high expected complexity) to a pattern, without significantly decreasing the size of its extension (*i.e.* low observation complexity), will be considered unexpected as it contradicts the hypothesis of independence of attributes. Conversely, adding a frequent attribute to a pattern, with minimal change in the induced graph is not deemed interesting.

Finally, we define the unexpectedness of a pattern built by adding q to Q, $p = (X, E \cap (X \times X), Q \cup \{q\})$ as the sum of the unexpectedness of its graph structure and its attribute set:

$$U^{(p)}(p) = U^{(g)}(G) + U^{(a)}(Q \cup \{q\}) \quad (3)$$

where $G = (X, E \cup (X \times X))$. To summarize, we anticipate a high unexpectedness in the graph structure for patterns with low or conversely, high connectivity compared to a sampled subgraph. Similarly, patterns that possess a substantial number of unusual attributes compared to what users would typically anticipate based on the original data will be considered unexpected.

C. Mining Algorithm

In order to enumerate unexpected patterns from attributed graphs, we design UNEXPATTERNS algorithm (see Algorithm 1). It is an adaptation of the Formal Concept Analysis (FCA) algorithm INCLOSE [37] – which enumerates conceptual structures, or formal concepts, from a binary table between sets of attributes and objects. For our purpose, we consider the set of attributes to be the set of graph attributes A and the set of objects to be the graph node set V. In general, FCA algorithms are enumeration algorithms, *i.e.* they seek all concepts. Typically, they are initialised with the concept that has no attributes (and its corresponding extension), then recursively build a lattice by adding one, two, ..., k attributes. The number of concepts in the lattice varies according to the different parameters, but in general it is bounded by the powerset of the set of objects, *i.e.* there are at most $2^{|V|}$ concepts. In Algorithm 1, we do not seek full enumeration anymore, but rather focus on exploring the search space to exhibit a few chosen elements – the unexpected patterns.

Algorithm 1: UNEXPATTERNS

Data: Attributed graph $\mathcal{G}$, Current pattern index r, Attribute index y, Unexpectedness constraint δ, Support constraints s and β
Result: Set of unexpected nodes and attributes

for $j = y$, $j < |A|$, j++ **do** ▷ A initially ordered on attribute frequency
  $X' \leftarrow \mathbf{X}_r \cap ext(a_j)$
  if $|X'| \geq s$ **then**
    if $|int(X')| \geq \beta$ **then**
      $Q' \leftarrow \mathbf{Q}_r \cup \{a_j\}$
      $u \leftarrow U(X', E \cap (X' \times X'), Q')$
      if $|X'| - |\mathbf{X}_r| == 0$ **then** ▷ Update current pattern
        if $u - \mathbf{U}_i \geq \delta$ **then**
          $\mathbf{Q}_r \leftarrow Q'$
          $\mathbf{U}_i \leftarrow u$
        else break ;
      else
        if ISCANONICAL$(\mathcal{G}, Q', \mathbf{X}_r, j-1)$ **then** ▷ Create new pattern
          if $u - \mathbf{U}_i \geq \delta$ **then**
            add X' to $\mathbf{X}$, Q' to $\mathbf{Q}$, u to $\mathbf{U}$
            $i \leftarrow i + 1$
            if $B(\sigma(|\mathbf{U}|))$ **then** ▷ Reduce pattern redundancy
              SHUFFLE$(A, j{+}1)$
            UNEXPATTERNS$(\mathcal{G}, |\mathbf{X}|{-}1, j{+}1)$
          else break ;
remove last element from $\mathbf{U}$
$i \leftarrow i - 1$
return $(\mathbf{X}, \mathbf{Q})$

Using definition 1, we recursively build patterns by iterating over graph attributes in a depth-first search manner until condition on pattern unexpectedness is no longer met, *i.e.* when making a pattern more specific does not come with a gain of unexpectedness greater than a threshold δ. We also control the depth of the search with the parameters β and s to further prune our search space: we only keep patterns that have at least β attributes, involving at least s nodes. Since we actively prune the search space, a risk inherent to our method is that we miss interesting patterns by finding large patterns first (in terms of number of nodes), making the threshold unreachable. To avoid this problem, we order the set of attributes by increasing frequency, *i.e.* frequent attributes will be examined last. However, this does not solve the problem of redundancy between patterns found. To maximise the chance of exploring different areas of the search space, we shuffle the list of attributes with a probability proportional to the depth of the recursion. In other words, if the number of variations around the same pattern becomes too large, we increase the probability that our algorithm will explore another part of the

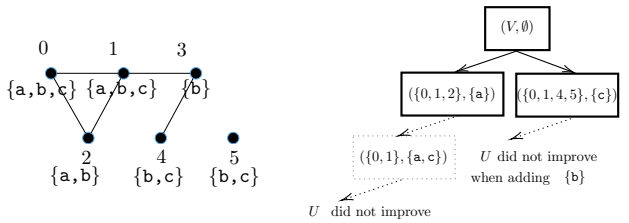


Fig. 3: An attributed graph (left) with details of the UNEXPATTERNS algorithm applied (right). Adding attribute $\{c\}$ modifies an existing pattern's extension, resulting in a new candidate pattern. However, it is discarded due to the unexpectedness loss, as the graph structure transitions from a 3-clique to a more common 2-nodes structure. The addition of attribute $\{b\}$ to another pattern does not change the graph structure, but since this attribute is shared by all nodes, it lacks sufficient interesting information and is not retained in the pattern.

search space. For an illustration, see Fig 3.

In Algorithm 1, global variables $\mathbf{X}$ and $\mathbf{Q}$ denote sets of pattern nodes and attributes (*i.e.* they are lists of patterns extensions and intensions, respectively), and we denote with $\mathbf{X}_r$ (resp. $\mathbf{Q}_r$) the set of nodes (attributes) of the r^{th} pattern. The index of the pattern being investigated is denoted by r. $\mathbf{U}$ stores all pattern unexpectedness values, and the index $\mathbf{U}_i$ refers to the unexpectedness of the i^{th} pattern. σ is a smoothing function, B a Bernoulli variable and SHUFFLE method shuffles all elements from a list starting at a specific index. Initially, $\mathbf{X}_0 = V$, $\mathbf{Q}_0 = \emptyset$, $\mathbf{U}_0 = 0$ and UNEXPATTERNS is called with $r = y = i = 0$. We verify that the pattern is new using ISCANONICAL, unchanged from [37].

Algorithm 1 operates as follow. Given a graph $\mathcal{G} = (V, E, A)$, we consider a pattern as a candidate if adding attribute $a_j \in A$ to its current attribute set and $ext(a_j)$ to its current node set enables it to satisfy the support constraints s and β (*lines 2 to 4*). In that case, if adding attribute a_j does not change the candidate pattern's number of nodes but still improves its unexpectedness, we expand the attribute set of the candidate pattern to include a_j (*lines 7 to 11*). Otherwise, we may start a new recursion branch to build the next pattern. To avoid enumerating the same patterns multiple times, we use ISCANONICAL [37] function (*line 13*). To prevent the algorithm from getting stuck with variations of the same pattern, we randomise the order of the remaining attributes using a Bernoulli variable with probability $p = \sigma(|\mathbf{U}|)$, where σ is a sigmoid function centered at a fixed value, *e.g.* $|\mathbf{U}| = 10$ (*lines 14 to 19*). Finally, at the end of each recursive call, we narrow down the unexpectedness storage and return pattern details (*lines 21 to 23*).

Complexity. Notice that at most $2^{|V|}$ patterns can be enumerated. At worst, the time complexity of one call of UNEXPATTERNS is $\mathcal{O}(nm^3)$, with $n = |V|$ and $m = |A|$ the number of nodes and attributes. This cost essentially stems from the computation of ISCANONICAL procedure, which incurs a cost of $\mathcal{O}(nm^2)$ and is repeated m times. In comparison, the computation cost of unexpectedness is primarily determined by the calculation of pattern node degrees. For graphs characterised by sparsity, typically real-world networks, this computation only costs $\mathcal{O}(k)$, with $k \ll n^2$ the number of edges in the graph. In practice, the efficiency of UNEXPATTERNS is maintained because graph sparsity limits patterns from containing both a large number of nodes and attributes.

TABLE I: Datasets statistics

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert A\rvert$	D_A
Wikivitals	10011	824999	37845	3.6×10^{-3}
Wikivitals-fr	9945	558427	28198	3.1×10^{-3}
Wikischools	4403	112834	20527	5.2×10^{-3}
SFCrimes	898	2172	39	0.57
Ingredients	2400	7932	20	0.45

D. Pattern Summaries

Even considering the different parameters and the attribute reordering mentioned above, the number of patterns found using UNEXPATTERNS can remain high enough in practice to be difficult to navigate through from a human level (*e.g.* thousands of patterns). To overcome this issue, we propose to merge the patterns found if they share common nodes in order to form *pattern summaries*. Therefore, a pattern summary is an attributed subgraph whose nodes (respectively attributes) are the union of the nodes (attributes) of the patterns that make it up. As an example, given a pattern $p_1 = (\{1,2,3\}, \{(1,2),(2,3),(3,1)\}, \{a,b\})$ and another pattern $p_2 = (\{2,4\}, \{(2,4)\}, \{c\})$, the resulting pattern summary will be given by the object $s = (\{1,2,3,4\}, \{(1,2),(2,3),(2,4),(3,1)\}, \{a,b,c\})$ since p_1 and p_2 share the node $\{2\}$. This step makes it possible to keep the results easy to visualise and relatively understandable, by identifying the components of the pattern summary.

IV. EXPERIMENTS

We evaluate our approach on five real-world networks. In our experiments, we address the following research questions:
RQ1 What is the impact of UNEXPATTERNS on the number of outputted patterns?
RQ2 How good is our method at summarizing graphs compared to other approaches?
RQ3 How informative are our patterns from a human point of view?
We make all of source code available [1].

A. Experimental Settings

Datasets. Wikipedia-based datasets[2] contain Wikipedia articles as nodes, with links between them if they are referencing each other. Each article comes with a feature vector containing

[1] https://github.com/unexpatterns/UnexPatterns
[2] https://netset.telecom-paris.fr/.

the count of words it contains. `Wikivitals` focuses on a selection of Wikipedia's "vital articles". `Wikivitals-fr` contains the vital articles written in French, and `Wikischools` contains articles related to material taught in schools. `SFCrimes` dataset[3] records 12 years of criminal activity in San Francisco. `Ingredients` dataset[4] links ingredients with high similarity in the recipes they appear in, and uses the number of recipes by nationality as attributes[5]. Dataset statistics are summarized in Table I.

Evaluation. In order to assess the benefits of our method, we design multiple baselines. We evaluate our approach against methods that leverage either the graph structure alone, the attributes alone, or both structure and attributes. Furthermore, we compare ourselves to two graph summarization methods from the literature, CENERGETICS and EXCESS [21]. We excluded SIAS-MINER [28] from our comparisons because depending on the dataset, the number of patterns returned was too large (hundreds of thousands) or the computation time too long (hours). Given k pattern summaries outputted by our method, our baselines are the following. With `louvain`, we apply the LOUVAIN algorithm [38] on the graph with a *resolution* tuned to generate k communities. In `spectral` we combine spectral embedding of the nodes with a KMEANS algorithm [39] to detect k communities. In `doc2vec` we consider each node as a document containing a bag-of-attributes and use DOC2VEC model [40] to learn representations of these documents. KMEANS is applied on top of these embeddings to generate k communities. With `GNN`, we use a Graph Convolutional Network [41] that embeds graph nodes while also considering their attributes. We apply a KMEANS algorithm to detect k communities among the node representations. Finally, we use CENERGETICS and EXCESS [21], which derive from the same algorithm and whose aim is to discover exceptional attributed subgraphs based on the frequency of some characteristics in patterns compared to the rest of the graph. The latter approach scales with the number of attributes by introducing a time budget.

Leveraging the work from [15], [16], we quantitatively evaluate our approach using three complementary criteria common to data mining, which we adapt to fit our definitions. We combine them in the same way as [16] did, in order to reflect the *expressiveness* of patterns.

We evaluate patterns on their *diversity* Δ: patterns should provide information on various themes from the original data, *i.e.* the outputs should not be redundant. To measure this, we learn representations of patterns using a pre-trained DOC2VEC model on their attribute set and then compute pairwise distances in the embedding space to evaluate their semantic proximity. More formally, given a set of patterns L we have $\Delta(L) = \frac{2 \cdot |\{pq : p,q \in L, d(\rho(p),\rho(q)) > \gamma\}|}{|L \times L|}$, with $pq \subseteq L \times L$ an unordered pair of patterns, ρ an embedding function, d a distance measure (*e.g.* Wasserstein distance) and γ a threshold.

[3]https://www.kaggle.com/c/sf-crime/data

[4]https://www.kaggle.com/datasets/kaggle/recipe-ingredients-dataset

[5]https://www.yummly.com

The *coverage* $K(L)$ is a measure of the cover of the pattern set on the graph, defined as $K(L) = \frac{|\bigcup_{p \in L} X|}{|V|}$, where $p = (X, E \cap (X \times X), Q)$. It measures the extent to which all nodes in the graph appear at least in one pattern.

The *width* $\Omega(L)$ measures the conciseness of a set of patterns, in terms of nodes and attributes. This is key to be readable and understandable by a user, and is defined as $\Omega(L) = |L| \cdot \sqrt{\bar{X} \cdot \bar{Q}}$, with $\bar{X}$ and $\bar{Q}$ the average number of nodes and attributes in patterns L. The square root operator reduces the penalty for methods without attribute filters, such as community search methods.

Therefore, we look for patterns with large diversity and coverage, but with small width and we define the *expressiveness* E of a set of patterns L as:

$$E(L) = \frac{\Delta(L) \cdot K(L)}{\Omega(L)} \tag{4}$$

B. Results

We evaluate the impact of the unexpectedness constraint on the number of patterns (RQ1) by running the UNEXPATTERNS algorithm with and without this constraint. In the latter case, it corresponds to the number of closed itemsets satisfying s and β. The results are reported in Table II with $\beta = 1$ for `Ingredients` and $\beta = 4$ otherwise. We observe that the greatest reductions occur for datasets with the largest density of attributes, namely `SFCrimes` and `Ingredients`. Under these circumstances, UNEXPATTERNS operates with high efficiency by effectively avoiding the creation of numerous variants of the same theme, which may have limited relevance to the user. Another advantage of using the unexpectedness constraint to reduce the search space is the significant reduction in computation time it allows. It takes UNEXPATTERNS a few hundreds seconds to mine the patterns from the largest dataset, namely `Wikivitals` with $s = 5$. In contrast neither SIAS-MINER nor CENERGETICS were able to produce results within a reasonable time frame of one hour.

TABLE II: Number of patterns found using UNEXPATTERNS (variation compared to algorithm without unexpectedness constraint). Reading key: on `Wikivitals` dataset, for support $s = 8$, 199 patterns are outputted, which is a reduction of 50.2% compared to not using the unexpectedness constraint.

	s							
Dataset	8		7		6		5	
`Wikivitals`	199	(-50.2%)	431	(-47.8%)	768	(-49.7%)	1169	(-53.9%)
`Wikivitals-fr`	50	(-25.4%)	87	(-33.1%)	138	(-43.2%)	361	(-32.5%)
`Wikischools`	247	(-69.7%)	697	(-73.0%)	1273	(-48.6%)	1357	(-77.8%)
`SFCrimes`	117	(-99.9%)	69	(-99.9%)	146	(-99.9%)	102	(-99.9%)
`Ingredients`	42	(-95.6%)	42	(-95.7%)	31	(-96.6%)	47	(-95.2%)

We then compare the *expressiveness* (4) of patterns (RQ2) (see Fig. 4). In the majority of datasets and parameters, pattern summaries built from UNEXPATTERNS demonstrate the highest level of expressiveness. This outcome can be attributed to two main factors. Firstly, our summaries exhibit a smaller *width* compared to the outputs from literature algorithms and baselines. The former is penalised by returning a large

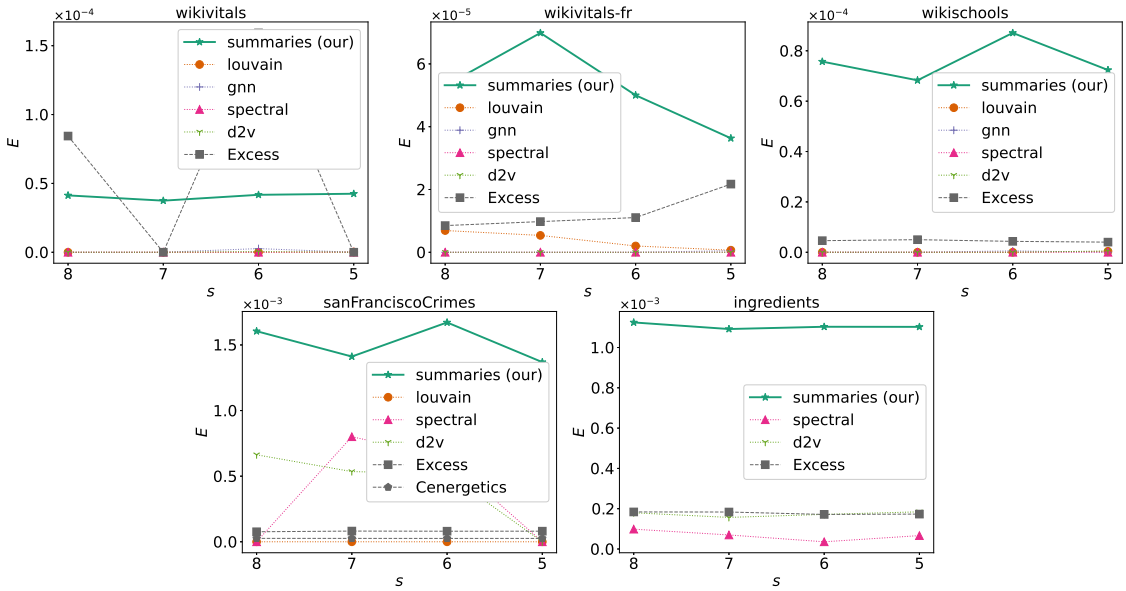
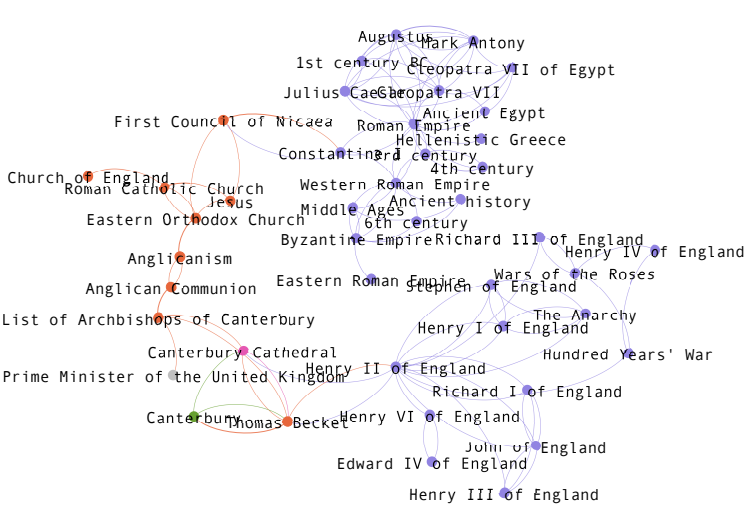
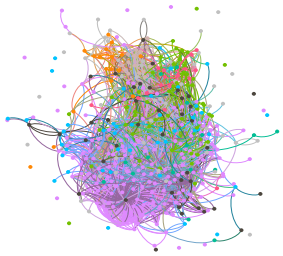


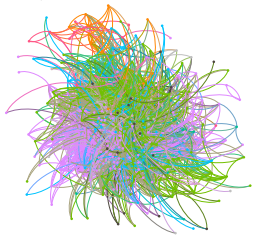
Fig. 4: *Expressiveness* E according to parameter s



(a) Pattern summary (43 nodes, 31 attributes)



(b) GNN (418 nodes, 8205 attributes)



(c) EXCESS (344 nodes, 710 attributes)

Fig. 5: A pattern summary (a) compared to its closest community or subgraph obtained with GNN (b) and EXCESS (c) methods. The colours of nodes and edges correspond to their respective categories in the Wikischools dataset.

number of patterns, while the latter lacks an attribute filter mechanism. Secondly, the *diversity* of our results compensates for their relatively low *coverage*. In contrast, community-based approaches maximise *coverage*, but their communities tend to be redundant as they encompass several different nodes and attributes. This redundancy makes it more challenging for users to comprehend the theme of each group.

Finally, we conduct a qualitative analysis of the results (RQ3) qnd compare a pattern summary from the Wikischools dataset with subgraphs obtained using the GNN and EXCESS methods (Fig. 5). We selected the methods that share the greatest number of nodes with our result. Our approach yields a subgraph containing 43 nodes (31 attributes), which is an aggregation of 60 patterns. In contrast, the respective comparison methods produce subgraphs with 418 and 344 nodes (8205 and 710 attributes respectively). Since our summary solely encodes connections between unexpected patterns without adding supplementary edges, it facilitates understanding of dependencies between node groups. Additionally, it allows for a scattered view of the results while preserving global understanding of interactions.

V. CONCLUSION

We introduced a novel approach for mining interesting patterns from attributed graphs by extending the concept of

Unexpectedness to attributed subgraphs. Experimental results on five real-world networks demonstrate that our method outperforms baseline and state-of-the-art approaches. It effectively reduces information redundancy while focusing only on interesting nodes and attributes. Furthermore, we illustrated how a simple aggregation of these patterns enables the generation of human-readable summaries that highlight some significant aspects of the original data.

Future perspectives for this work include an in-depth exploration of attributed graph Unexpectedness, possibly involving a user-study, to further validate its correlation with human interestingness. In terms of applications, UNEXPATTERNS could be used as an unsupervised approach for learning information from a transaction graph and detecting fraudulent activities. Furthermore, our approach holds promise in the field of explainability, where mining unexpected patterns in the proximity of predicted edges could provide insights into the relationship between algorithm choices and human-perceived points of interest.

REFERENCES

[1] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[2] A. Perer and B. Shneiderman, "Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 265–274, 2008.

[3] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, pp. 415–444, 2001.

[4] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 718–729, 2009.

[5] H. Cheng, Y. Zhou, and J. X. Yu, "Clustering large attributed graphs: A balance between structural and attribute similarities," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 2, 2011.

[6] P. Chunaev, "Community detection in node-attributed social networks: a survey," *Computer Science Review*, vol. 37, p. 100286, 2020.

[7] F. Moser, R. Colak, A. Rafiey, and M. Ester, "Mining cohesive patterns from graphs with feature vectors," in *Proceedings of the 2009 SIAM international conference on data mining*, pp. 593–604, SIAM, 2009.

[8] B. Boden, S. Günnemann, H. Hoffmann, and T. Seidl, "Mining coherent subgraphs in multi-layer graphs with edge labels," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1258–1266, 2012.

[9] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," in *Proceedings 2001 IEEE international conference on data mining*, pp. 313–320, IEEE, 2001.

[10] D. J. Hand, "Pattern detection and discovery," in *Pattern detection and discovery*, pp. 1–12, Springer, 2002.

[11] W. Klösgen and J. M. Zytkow, *Handbook of data mining and knowledge discovery*. Oxford University Press, Inc., 2002.

[12] J.-L. Dessalles, "Coincidences and the encounter problem: A formal account," in *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (B. C. Love, K. McRae, and V. M. Sloutsky, eds.), (Austin, TX), pp. 2134—2139, Cognitive Science Society, 2008.

[13] P. Maguire, P. Moser, R. Maguire, and M. T. Keane, "Seeing patterns in randomness: A computational model of surprise," *Topics in Cognitive Science*, vol. 11, no. 1, pp. 103–118, 2019.

[14] J.-L. Dessalles, "Algorithmic simplicity and relevance," in *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence: Papers from the Ray Solomonoff 85th Memorial Conference, Melbourne, VIC, Australia, 2011*, pp. 119–130, Springer, 2013.

[15] L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 3, 2006.

[16] N. Zhang, Y. Tian, and J. M. Patel, "Discovery-driven graph summarization," in *2010 IEEE 26th international conference on data engineering (ICDE 2010)*, pp. 880–891, IEEE, 2010.

[17] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[18] S. Günnemann, I. Färber, B. Boden, and T. Seidl, "Subspace clustering meets dense subgraph mining: A synthesis of two paradigms," in *2010 IEEE International Conference on Data Mining*, pp. 845–850, 2010.

[19] F. Lemmerich, M. Becker, P. Singer, D. Helic, A. Hotho, and M. Strohmaier, "Mining subgroups with exceptional transition behavior," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 965–974, 2016.

[20] M. Kaytoue, M. Plantevit, A. Zimmermann, A. Bendimerad, and C. Robardet, "Exceptional contextual subgraph mining," *Machine Learning*, vol. 106, no. 8, pp. 1171–1211, 2017.

[21] A. Bendimerad, M. Plantevit, and C. Robardet, "Mining exceptional closed patterns in attributed graphs," *Knowledge and Information Systems*, vol. 56, no. 1, pp. 1–25, 2018.

[22] A. A. Bendimerad, M. Plantevit, and C. Robardet, "Unsupervised exceptional attributed sub-graph mining in urban data," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, IEEE, 2016.

[23] T. De Bie, K.-N. Kontonasios, and E. Spyropoulou, "A framework for mining interesting pattern sets," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 92–100, 2011.

[24] A. Silberschatz and A. Tuzhilin, "On subjective measures of interestingness in knowledge discovery.," in *KDD*, vol. 95, pp. 275–281, 1995.

[25] T. De Bie, "An information theoretic framework for data mining," in *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 564–572, 2011.

[26] T. De Bie, "Subjective interestingness in exploratory data mining," in *International Symposium on Intelligent Data Analysis*, pp. 19–31, Springer, 2013.

[27] M. van Leeuwen, T. De Bie, E. Spyropoulou, and C. Mesnage, "Subjective interestingness of subgraph patterns," *Machine Learning*, vol. 105, no. 1, pp. 41–75, 2016.

[28] A. Bendimerad, A. Mel, J. Lijffijt, M. Plantevit, C. Robardet, and T. De Bie, "Sias-miner: mining subjectively interesting attributed subgraphs," *Data Mining and Knowledge Discovery*, vol. 34, no. 2, 2020.

[29] J. Deng, B. Kang, J. Lijffijt, and T. De Bie, "Mining explainable local and global subgraph patterns with surprising densities," *Data Mining and Knowledge Discovery*, vol. 35, no. 1, pp. 321–371, 2021.

[30] S. Dasgupta and A. Gupta, "Discovering interesting subgraphs in social media networks," in *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 105–109, IEEE, 2020.

[31] M. Alassad, B. Spann, and N. Agarwal, "Combining advanced computational social science and graph theoretic techniques to reveal adversarial information operations," *Information Processing & Management*, vol. 58, no. 1, p. 102385, 2021.

[32] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM computing surveys (CSUR)*, vol. 51, no. 3, 2018.

[33] Y. Wu, Z. Zhong, W. Xiong, and N. Jing, "Graph summarization for attributed graphs," in *2014 International conference on information science, electronics and electrical engineering*, vol. 1, pp. 503–507, IEEE, 2014.

[34] M. Atzmueller, S. Doerfel, and F. Mitzlaff, "Description-oriented community detection using exhaustive subgroup discovery," *Information Sciences*, vol. 329, pp. 965–984, 2016.

[35] D. J. Cook and L. B. Holder, "Substructure discovery using minimum description length and background knowledge," *Journal of Artificial Intelligence Research*, vol. 1, pp. 231–255, 1993.

[36] D. J. Watts, "Networks, dynamics, and the small-world phenomenon," *American Journal of sociology*, vol. 105, no. 2, pp. 493–527, 1999.

[37] S. Andrews, "In-Close, a Fast Algorithm for Computing Formal Concepts," in *International Conference on Conceptual Structures (ICCS)*, p. 15, 2009.

[38] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[39] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[40] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*, pp. 1188–1196, PMLR, 2014.

[41] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.