International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-021-00292-y

REGULAR PAPER

Mining subgraph coverage patterns from graph transactions

A. Srinivas Reddy[1] · P. Krishna Reddy[1] · Anirban Mondal[2] · U. Deva Priyakumar[1]

Received: 23 February 2021 / Accepted: 24 October 2021

Abstract
Pattern mining from graph transactional data (GTD) is an active area of research with applications in the domains of bioinformatics, chemical informatics and social networks. Existing works address the problem of mining frequent subgraphs from GTD. However, the knowledge concerning the coverage aspect of a set of subgraphs is also valuable for improving the performance of several applications. In this regard, we introduce the notion of subgraph coverage patterns (*SCPs*). Given a GTD, a subgraph coverage pattern is a set of subgraphs subject to relative frequency, coverage and overlap constraints provided by the user. We propose the **S**ubgraph **I**D-based **F**lat **T**ransactional (*SIFT*) framework for the efficient extraction of *SCPs* from a given GTD. Our performance evaluation using three real datasets demonstrates that our proposed *SIFT* framework is indeed capable of efficiently extracting *SCPs* from GTD. Furthermore, we demonstrate the effectiveness of *SIFT* through a case study in computer-aided drug design.

Keywords Graph mining · Subgraph mining · Subgraph coverage patterns · Bio-informatics

1 Introduction

A complex graph can be built from pieces of knowledge based on the relationships among various entities. Such a graph contains new kinds of interesting and useful knowledge structures. Hence, it can be extremely valuable for opening up new avenues for enhancing applications in several domains. In this regard, graph mining [10,33] has become an active area of research for mining knowledge from graph representations in bio-informatics, chemical informatics, social networks, computer vision, video indexing, text retrieval and web analysis. Mining the knowledge of frequent subgraphs from graph transactional data (GTD) is an important and active area of graph mining [14,17,22,23,40,42]. It has been demonstrated in [29,34] that frequent subgraph mining can extract *interesting patterns* from GTD for providing valuable knowledge in the domain of bio-informatics.

Knowledge concerning the coverage aspect of a set of subgraphs can be valuable for improving the performance of several applications. In the literature, the notion of *coverage* has been well explored in set theory and graph theory [7, 11–13,15,19,27,39,47] as well as the extraction of coverage patterns from transactional data [18,36]. *However, none of the existing works have investigated the issue of extracting the coverage-related knowledge of patterns from GTD.* We believe that the *coverage*-related knowledge in the form of subgraph patterns can be used in improving the performance of applications in chemical, biological and social network domains.

Computational biology approaches have become ubiquitous in the process of discovering new drug molecules that can treat diseases. In the process of drug design, careful and systematic changes in the molecular structure, which can maximize the interactions between the drug molecule and the protein relevant to the given disease, are crucial. Methods that help us to understand and quantify such intermolecular interactions between proteins and drug molecules will open up significant avenues for research in this direction. In the literature, frequent subgraph mining (FSM) techniques have been applied in [29,34] to study protein–ligand interac-

✉ A. Srinivas Reddy
srinivas.annappalli@research.iiit.ac.in

P. Krishna Reddy
pkreddy@iiit.ac.in

Anirban Mondal
anirban.mondal@ashoka.edu.in

U. Deva Priyakumar
deva@iiit.ac.in

[1] Kohli Centre on Intelligent Systems, IIIT, Hyderabad, India

[2] Department of Computer Science, Ashoka University, Delhi, India

Published online: 02 December 2021

tions by analyzing the interaction patterns of different drug molecules with the target protein.

Consider a scenario, where we have identified a number of low-binding drug molecules. The objective is to improve the structure of the molecules to increase the binding affinity so that the drug would work at lower dosages. Existing FSM techniques are not capable of extracting coverage-related knowledge of patterns. However, if we develop approaches for discovering coverage-related knowledge patterns, such methods can be effectively used to help in optimizing the binding affinity of the drug molecules w.r.t. the selected protein, thereby making the drug design process more efficient.

A subgraph coverage pattern (*SCP*) is a set (or pattern), whose elements are the subgraphs of GTD. This work addresses the problem of extracting all *SCPs*, which cover a given percentage of the graph transactions of GTD. Notably, in addition to the coverage aspect, we have to consider the aspect of *overlap*, which arises as a consequence of considering the coverage of a set of subgraphs. The sets of graph transactions covered by the corresponding subgraphs of an *SCP* may contain the common transactions, which we refer to as *overlap*. In addition to coverage, we consider an *SCP* is interesting if there is a minimal overlap among the graph transactions covered by subgraphs of an *SCP*.

Given a GTD, the issue is to extract all of the possible *SCPs* subject to the constraints associated with *coverage* and *overlap*. A brute-force approach would be to first extract all of the possible subgraphs of GTD by employing a frequent subgraph extraction algorithm [9,22,25,42] and then determine the *coverage* and *overlap* for each possible pattern consisting of subgraphs. Intuitively, such an approach would be prohibitively expensive because the number of possible subgraphs in a given GTD typically explodes. Consequently, the number of candidate patterns to be considered for the extraction of *SCPs* also explodes.

For efficiently extracting *SCPs* from a given GTD, we introduce the model of *SCPs* and present the framework to extract *SCPs*. In the proposed model, we define the notion of *coverage* and *overlap* and present the problem to extract *SCPs*. The problem is to extract all *SCPs* from a given GTD by satisfying the threshold values of given *coverage* and *overlap*. We present a framework to extract *SCPs*, which we designate as subgraph-identifier-based flat transactional (*SIFT*) framework. As a part of *SIFT* framework, we propose an approach to extract all subgraphs of the graph transactions in GTD and assign a unique subgraph identifier (*SID*) to each subgraph. Next, each graph transaction in GTD is transformed into the corresponding flat transaction, which consists of the corresponding *SIDs*. We also propose an approach to extract *SCPs* from the flat transactional dataset. The problem is similar to that of coverage pattern extraction [36] from flat transactional databases subject to *coverage* and *overlap* constraints. Incidentally, *overlap* follows the sorted closure property [28], which we shall use in this work for facilitating effective pruning. Hence, we extend the existing pattern extraction approach [36], which employs pruning strategy based on *overlap*, for the efficient extraction of *SCPs*. Observe that by forming a set of flat transactions, complex computationally intensive graph operations associated by *coverage* and *overlap* are replaced with the corresponding simpler and relatively fast set-based operations.

The key contributions of this work are three-fold:

1. We introduce the notion of *subgraph coverage patterns (SCPs)* for GTD.
2. We propose the *SIFT* framework for *efficiently* extracting *SCPs* from a given GTD.
3. We conduct an extensive performance study using three real datasets to demonstrate that it is indeed feasible to *efficiently* extract *SCPs* using our proposed *SIFT* framework. We also demonstrate the effectiveness of *SIFT* through a case study in computer-aided drug design.

To the best of our knowledge, this is the first work to consider the extraction of subgraph coverage patterns from graph transactional data. The remainder of this paper is organized as follows. Section 2 reviews related works and background information. Section 3 discusses the proposed framework of the problem. Section 4 presents our proposed *SIFT* framework. Section 5 reports our performance evaluation. Section 6 presents our case study. Finally, we conclude in Sect. 7.

2 Related work and background

This section discusses related works and background.

2.1 Related work

Research efforts, such as the *gindex* approach [44], have designed indexes for extracting subgraphs by considering *line, cycle* and *star* as basic graph query structures. Moreover, the *GString* semantic-based approach [24] indexes chemical compounds databases.

Graph mining techniques have also been applied in GTDs. The work in [22] used an apriori-based algorithm for discovering frequent subgraphs in GTDs. Moreover, the work in [25] discussed the frequent subgraph mining algorithm, which incorporates canonical labelling in conjunction with sparse graph representation to reduce both time and space complexity. The work in [42] proposed the *gSpan algorithm* for discovering frequent subgraphs without candidate generation. In particular, *gSpan* uses a lexicographic ordering for mapping each graph to a unique minimum depth-first-search code as its canonical label. A good survey on frequent sub-

graph mining techniques for GTDs can be found in [23]. An algorithm to extract the top-k frequent subgraphs has been proposed in [17].

The work in [26] proposed REAFUM, an approximate subgraph mining framework that constructs a list of representative graphs. It extracts frequent representative subgraphs, allows approximate matches and extracts consensus patterns. Another work in [48] proposed HOS-PLOC, a local clustering framework that extracts a small high-order conductance cluster, which largely preserves the user-specified network structures.

The work in [9] proposed an algorithm to mine molecular fragments based on association rule mining. These molecular fragments help in discriminating drug classes. Researchers have made efforts to model protein–ligand complex (PLC) as graph structures and extracted frequently occurring subgraph patterns. The works in [34,35] proposed GReMLIN (graph mining strategy to infer protein–ligand interaction patterns), which is a methodology to search for conserved protein–ligand interactions in a group of related proteins-ligand complexes. They use frequent subgraph mining to recognize structural patterns relevant to protein–ligand interaction. The work in [29] modeled PLC as a bipartite graph and used graph topological properties like degree, closeness, communicability, eccentricity, node betweenness and edge betweenness to summarize and extract frequent patterns. The work in [40] proposed FERRARI, which is a visual exploratory subgraph search framework. In particular, it employs two index structures VACCINE and ADVISE for indexing frequent and infrequent subgraphs to improve efficiency and scalability.

The notion of coverage has been well explored in set theory in the form of the set cover problem [12] and the hitting set problem [15]. In graph theory, the notion of coverage has been explored in the form of the minimum vertex cover problem [11,39], hitting set problem in hypergraph, traversals of a hypergraph [20], clique cover problem [19] and influence maximization problem [27,47]. The notion of coverage has been used in hypergraphs in the form of the minimum hitting set problem [7], which involves the extraction of the set of vertices that have a non-empty intersection with every hyperedge. The work in [31] states that a set of k-mers is a universal hitting set if every possible L-long sequence contains a k-mer from a given DNA/RNA sequence dataset.

Clique cover is another important problem with applications in compiler optimization, computational geometry and applied statistics [19]. Information coverage maximization in social networks [27,47] also uses coverage. An approach was proposed in [46] to select a set of influential users. More recently, the work in [41] proposed TOPKLS, a local search algorithm for finding diversified top-k cliques from a given graph. The notion of coverage has been well explored to solve the maximum coverage problem in facility location [6]. However, these works explore the notion of coverage for a single graph as opposed to a GTD, which is our focus. Moreover, the works in [18,36] find coverage patterns in transactional data using pattern-growth and level-wise pruning approaches, respectively.

Notably, all of the existing works have addressed the issue of GTDs for graph search and mining of frequent subgraphs related knowledge with applications in chemical and biological areas. The issue of extracting the knowledge related to coverage from GTD has not been addressed. In contrast with the existing works [29,40,42], in this paper, we have made an effort to propose an approach to extract coverage-related knowledge from GTDs.

2.2 Background information

We shall now discuss the model of subgraph discovery from graph transactions and coverage patterns.

2.2.1 Model of subgraph discovery

In the literature, several efforts [4,8] are being made to discover the knowledge of subgraphs from the given set of graph transactions. In particular, in the area of bioinformatics, research efforts [22,42] have been made by modeling chemical compounds as graph transactions. By considering a given chemical compound as a single unit, the corresponding graph transaction represents chemical elements as vertices and chemical bonds among them as edges. We first present the notion of graph transactions and briefly explain the subgraph discovery approach, which was presented in [42].

A *graph transaction* $G = (V,E,L,l)$ is a labeled, connected, simple and undirected graph, where V is a set of vertices, $E \subseteq V^2$ is a set of edges, L is a set of labels and $l : V \cup E \rightarrow L$, where l is a function for assigning labels to vertices and edges. A graph transactional dataset (GTD) D comprises n such graph transactions, where the value of n is typically large. Notably, a vertex/edge may belong to multiple graph transactions in D. A portion S of a graph transaction G is called a *subgraph* of G. Given $G = (V, E)$ and $S = (V_s, E_s)$, we say that S is a subgraph of G or S exists in G (denoted as $S \subseteq G$), iff $V_s \subseteq V$, $E_s \subseteq E$, $\forall(u, v) \in E_s \longrightarrow u, v \in V_s$.

Example 1 Consider a sample chemical compound shown in Fig. 1a and its equivalent graph transaction G=(V,E,L,l) depicted in Fig. 1b. Here, V={v_1, v_2, . . . ,v_{13}}, E={(v_1,v_3), (v_2,v_3), . . . , (v_9,v_{13})} and L= {C,F,H,N,O,1,2,3}. A mapping function l maps the vertices v_1,v_2,v_3,. . . ,v_{13} to H,N,N,. . . ,F and the edges (v_1,v_3), (v_2,v_3), . . . , (v_9,v_{13}) to 1, 3,. . . ,1, respectively.

Given a GTD, a subgraph is a potential subgraph if it is present in certain percentage of graphs in GTD. We can

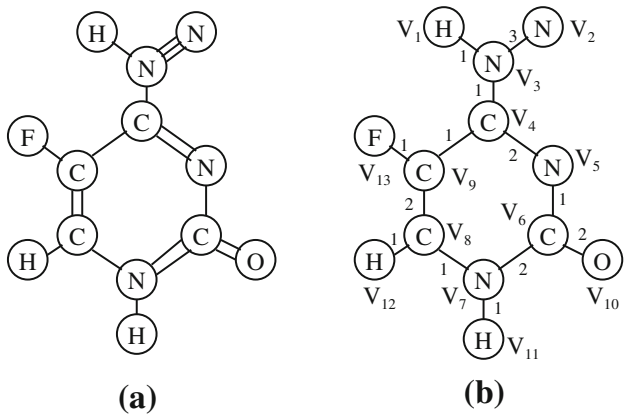


Fig. 1 **a** Sample chemical compound, **b** equivalent graph model

employ a subgraph discovery algorithm (e.g., *gSpan* [42]) for extracting candidate subgraphs from D. The *gSpan* algorithm employs a depth-first search (DFS) strategy to extract all subgraphs without candidate generation. It uses a pattern-growth approach to build a hierarchical search tree called the *DFS Code Tree*. In the *DFS Code Tree*, every node represents a subgraph/graph, and any subgraph/graph in GTD can find its node in the *DFS Code Tree*. Each node in the tree is assigned with a lexicographical canonical label called the DFS code and one subgraph can have multiple DFS codes. The first DFS code in pre-order traversal over the *DFS Code Tree* is called the *minimum DFS code* and is assigned as the *canonical label* to the subgraph. Moreover, *gSpan* also prunes all the nodes that contain non-minimum DFS code, thereby reducing the size of the *DFS Code Tree*. A depth-first search over the *DFS Code Tree* extracts all *minimum DFS codes* of all candidate subgraphs in D. The performance of *gSpan* improves drastically due to the merging of isomorphism test and subgraph growth into one procedure.

2.2.2 Model of coverage patterns

We shall now explain the concept of coverage patterns [18,36]. Coverage patterns are characterized by the notions of relative frequency, coverage support and overlap ratio. Given a transactional database D, each transaction is a subset of a set I of m items $\{i_1, i_2, i_3, \ldots, i_m\}$. T^{i_k} denotes the set of transactions in which item i_k is present. The fraction of transactions containing a given item i_k is designated as *Relative Frequency* of i_k ($RF(i_k)$). Hence, $RF(i_k)=\frac{|T^{i_k}|}{|D|}$. A given item is considered as *frequent* if its relative frequency is greater than that of a threshold value, which we designate as *minRF*. A pattern P is a subset of items in I, i.e., $P \subseteq I$ where $P=\{i_p, i_q, \ldots, i_r\}$, where $1 \leq p, q, r \leq$ m. The *Coverage Set* (*CSet(P)*) of a pattern P is the set of all the transactions that contain at least one item from the pattern P, i.e., $CSet(P)=T^{i_p} \cup T^{i_q} \cup \ldots \cup T^{i_r}$. The *Coverage Support* of a pattern P (*CS(P)*) is the ratio of the size of *CSet(P)* to the size of D, i.e., $CS(P)=\frac{|CSet(P)|}{|D|}$. In order to add a new item to the pattern P such that the coverage support increases significantly, the notion of *overlap ratio* is introduced. (This is possible in the case when the number of transactions, which are common to the new item and the pattern P, is low.) Given a pattern P, the notion of overlap ratio of P satisfies the sorted closure property [28], when the items in P are sorted in decreasing order of their relative frequencies, i.e., $1 \leq RF(i_p) \leq RF(i_q) \leq \ldots \leq RF(i_r)$. The *Overlap Ratio* of a pattern P (*OR(P)*) is the ratio of the number of transactions that are common between $CSet(P-i_r)$ and T^{i_r} to the number of transactions in T^{i_r}, i.e., $OR(P)=\frac{|CSet(P-i_r) \cap (T^{i_r})|}{|T^{i_r}|}$. A high value of coverage support indicates more number of transactions and a low value of overlap ratio means less repetitions among the transactions. A pattern is *interesting* if its coverage support is greater than or equal to the user-specified minimum Coverage Support threshold value (*minCS*) and its overlap ratio is less than or equal to the user-specified maximum Overlap Ratio threshold value (*maxOR*). Given the values of *minRF*, *minCS* and *maxOR*, a pattern $P=\{i_p, i_q, \ldots, i_r\}$ is considered as a *coverage pattern* if $RF(i_k) \geq minRF \; \forall \; i_k \in P$, $CS(P) \geq minCS$ and $OR(P) \leq maxOR$. By exploiting the sorted closure property of *overlap ratio*, a level-wise apriori-based approach has been proposed in [36] and a pattern-growth-based approach has been proposed in [18] for extracting all coverage patterns from D, given *minRF*, *minCS* and *maxOR* values. Additionally, a MapReduce-based algorithm to extract coverage patterns has been proposed in [32].

To extract the knowledge of coverage patterns, the following heuristics can be followed for setting *minRF*, *minCS* and *maxOR* values. Normally, the coverage patterns with maximum coverage (100%) and minimum overlap ratio (0%) are interesting. Moreover, the coverage patterns having items with less relative frequency are not interesting. So, *minRF* threshold value can be set based on the characteristics of the application. In the beginning, as a heuristic, coverage patterns can be extracted by setting *maxOR* at 0 and *minCS* at 1 and *minRF* can be set to 50% of *minCS* value. Then, based on the requirement of the number of coverage patterns, *maxOR* can be increased gradually, while *minCS* and *minRF* can be decreased gradually.

3 Proposed framework of the problem

Consider a graph transactional dataset (GTD) D, a minimum relative frequency threshold $minRF_g$, a minimum coverage support threshold $minCS_g$ and a maximum overlap threshold $maxO_g$. Our proposed *SIFT* framework returns the set of all subgraph coverage patterns (*SCPs*) subject to the $minRF_g$, $minCS_g$ and $maxO_g$ constraints. Note that we employ the

Table 1 Summary of notations

Notation	Description
G_i	Graph transaction
D	Graph transactions dataset (GTD)
Ψ	Universe of subgraphs
SP	Subgraph pattern
O_h	Subgraph identifier (SID)
f_i	Flat transaction
D_f	Flat transactional dataset
X	Set of *SIDs* or pattern
SCPs	Subgraph coverage patterns

notations $minRF_g$, $minCS_g$ and $maxO_g$, (i.e., we add a subscript g to *minRF*, *minCS* and *maxO*), to represent minimum relative frequency, minimum coverage support and maximum overlap thresholds concerning a graph transactional dataset. Now we shall explain the relevant terminology to present the framework of the problem. Table 1 depicts the summary of notations used in this paper.

3.1 Subgraph pattern, cover and cover set

Recall the notions of *graph transaction*, *graph transactional dataset* and *subgraph* from the discussions in Sect. 2.2. Given a GTD D and the set Ψ of all possible subgraphs over D, a subgraph pattern (SP) is a set of subgraphs belonging to Ψ. Consider a graph transaction $G_i \in D$. A subgraph S_j is said to *cover* G_i if S_j exists in G_i. We define $cover(S_j, G_i)$ as follows:

$$cover(S_j, G_i) = \begin{cases} 1 & if\, S_j \subseteq G_i \\ 0 & otherwise \end{cases} \tag{1}$$

Computation of $cover(S_j, G_i)$ involves solving the subgraph isomorphism problem [25], which is NP-complete [16]. The *gSpan algorithm* [42] uses a canonical labeling system called *DFS lexicographical order*, which assigns *minimum DFS code* to each graph as the canonical label. We compute $cover(S_j, G_i)$ based on DFS codes. The *Cover Set* of S_j ($CSet_g(S_j, D)$) is defined as the set of all graph transactions covered by S_j. Formally, $CSet_g(S_j, D) = \{G_i | cover(S_j, G_i) = 1 \ \& \ G_i \in D\}$. The *Cover Set* of SP ($CSet_g(SP, D)$) is a set of all graph transactions, which are covered by at least one subgraph of SP. It is equal to the union of all graph transactions covered by all the subgraphs in SP. Hence, $CSet_g(SP, D) = \bigcup_{\forall S_j \in SP} CSet_g(S_j, D)$.

3.2 Relative frequency RF_g of a subgraph

Given D and S_j, we denote the percentage of graph transactions in D covered by S_j as relative frequency RF_g of S_j. We compute $RF_g(S_j, D)$ as follows:

$$RF_g(S_j, D) = \frac{|CSet(S_j, D)|}{|D|} \tag{2}$$

Here, $0 \leq RF_g(S_j,\mathrm{D}) \leq 1$. We can extract subgraphs of interest from D based on user-specified *minimum relative frequency* ($minRF_g$) threshold.

Example 2 Consider a sample graph transactional dataset D comprising of 10 graph transactions G_1 to G_{10}, shown in Fig. 2a. Three subgraphs S_1, S_2 and S_3 are shown in Fig. 2b. Here, S_1 is a subgraph of G_1, G_6 and G_{10}; S_2 is a subgraph of G_5, G_7 and G_8; and S_3 is a subgraph of G_4 and G_7. The subgraph S_1 is said to *cover* G_1 since $S_1 \subseteq G_1$. Hence, $cover(S_1, G_1)$=1. Moreover, $CSet(S_1, D) = \{G_1, G_6, G_{10}\}$ and $RF_g(S_1, D) = \frac{|CSet(S_1, D)|}{D} = \frac{3}{10} = 0.3$. Similarly, RF values of S_2 and S_3 are 0.3 and 0.2, respectively.

3.3 Coverage support

Given D and a subgraph pattern SP, the coverage support of SP ($CS_g(SP, D)$) is the percentage of graph transactions in D covered by at least one subgraph in SP. We compute $CS_g(SP, D)$ as follows:

$$CS_g(SP, D) = \frac{|CSet(SP, D)|}{|D|} \tag{3}$$

Here, $0 \leq CS_g(SP, D) \leq 1$. Notably, $CS_g(SP, D) = 1$ when all of the graph transactions in D are covered by SP. Conversely, $CS_g(SP, D) = 0$ when none of the graph transactions are covered by SP. A pattern SP is interesting w.r.t coverage perspective if $CS_g(SP, D) \geq minCS_g$, where $minCS_g$ is a user-defined *minimum coverage support threshold* for graph transactions.

3.4 Overlap

A pattern SP, which satisfies a given $minCS_g$ constraint, may not be interesting if there is significant overlap among the sets of transactions covered by subgraphs of SP. In several applications, an SP with maximum coverage support and minimum overlap could be interesting. We now explain the notion of $overlap_g$ for capturing the overlap associated with graph transactions.

In the literature, the concept of overlap between sets is most often described using Euler diagrams [5]. Consider two sets A and B in a universe of objects. The overlap of A and B is computed by $\frac{A \cap B}{A \cup B}$. This equation does not consider the

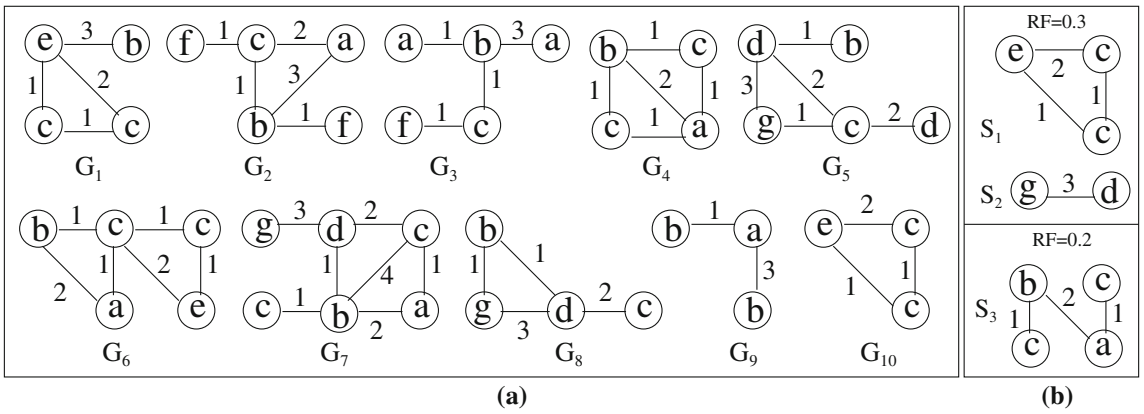


Fig. 2 **a** Sample of 10 graph transactions, **b** candidate subgraphs with $minRF_g = 0.2$

number of times an object is appearing in either A or B. As a result, it is not possible to attach a physical meaning unless we know the nature of repetition of objects in A and B. In this paper, we present the notion of overlap by considering the average number of times an object can appear in the given multi-set (contains duplicate elements). Let $M(SP, D)$ be the multi-set, which contains all transactions (with duplicate entries) covered by each subgraph in SP. We define the value of $overlap_g$ as the average number of times a transaction is repeated in $M(SP, D)$. We define $overlap_g$ as follows:

$$overlap_g(SP, D) = \left(\frac{|M(SP, D)|}{|CSet(SP, D)|} - 1 \right) \cdot 100 \qquad (4)$$

Observe that the value of $overlap_g$ can exceed 100% as the size of $|M(SP, D)|$ could be unbounded. In this paper, we restrict the size of $|M(SP, D)|$ by considering that a transaction can only appear twice in $|M(SP, D)|$. Hence, the notion of $overlap_g$ denotes the average number of times a subgraph appears at most twice in D. Here, $overlap_g$=1 if every transaction appears twice in $M(SP, D)$, i.e., the maximum value of $|M(SP, D)|$=2·$|CSet(SP, D)|$. Conversely, $overlap_g$=0 if every transaction appears only once in $M(SP, D)$, i.e., the minimum value of $|M(SP, D)|$=$|CSet(SP, D)|$. Hence, $0 \leq overlap_g(SP, D) \leq 1$. A pattern SP is interesting if $overlap_g(SP, D) \leq maxO_g$, where $maxO_g$ is a user-defined *maximum Overlap threshold* for graph transactions.

In essence, there can be different ways of computing overlap based on the application requirement. In case of applications, in which a transaction appears more than twice, say k times, in $|M(SP, D)|$, Equation 4 is modified as $overlap_g(SP, D) = \frac{1}{k-1} \left(\frac{|M(SP,D)|}{|CSet(SP,D)|} - 1 \right) \cdot 100$, where the maximum value of $|M(SP, D)|$ equals $k \cdot |CSet(SP, D)|$.

3.5 Subgraph coverage pattern

We consider an SP as interesting if the cover set of all subgraphs of SP satisfies the $minRF_g$ threshold, overlap of SP satisfies the $maxO_g$ threshold and coverage support of SP satisfies the $minCS_g$ threshold. We designate such SPs as subgraph coverage patterns (*SCPs*). The definition of *SCP* is given below.

Definition 1 (*Subgraph coverage pattern (SCP)*) Consider D and a pattern SP. We call SP as a subgraph coverage pattern if $CS_g(SP, D) \geq minCS_g$ and $overlap_g(SP, D) \leq maxO_g, \forall S_j \in SP, RF_g(S_j, D) \geq minRF_g$.

Example 3 In Fig. 2b, let SP be the set $\{S_1, S_2, S_3\}$. The RF values of S_1, S_2 and S_3 are 0.3, 0.3, and 0.2, respectively. The coverage set of SP, $CSet(SP,D)$= {(G_1,G_4,G_5,G_6,G_7,G_8, G_{10}}. The coverage support of SP, $CS_g(SP, D) = \frac{|CSet(SP,D)|}{|D|}$ = $\frac{7}{10}$ = 0.7. The multi-set of transactions covered by pattern SP, $M(SP, D)$= {(G_1,G_6,G_{10}),(G_5,G_7,G_8),(G_4,G_7)}. Therefore, the overlap among transactions covered by subgraphs of SP, $overlap_g(SP, D) = \left(\frac{|M(SP,D)|}{|CSet(SP,D)|} - 1\right)$=($\frac{8}{7}$ − 1) = 0.142. Given the values of $minRF_g = 0.2$, $minCS_g = 0.7$ and $maxO_g$=0.5, the pattern $SP = \{S_1, S_2, S_3\}$ is an *SCP*.

3.6 Problem statement

Given a graph transactional dataset D, and the values of user-defined constraint parameters $minRF_g$, $minCS_g$ and $maxO_g$, the problem is to extract all *subgraph coverage patterns* satisfying these user-defined constraints.

It can be noted that the objective is to extract *SCPs* with high coverage value for a given application scenario. Normally, the *SCPs* having subgraphs with low relative frequency value are not interesting. So, there will be significant number of *SCPs*, which cover small portion of GTD. As *minCS* threshold increases, the number of *SCPs* will reduce. Similarly, as *minRF* increases, the number of *SCPs* will

reduce. Regarding overlap, we consider that the *SCPs* with minimum overlap will be interesting. Therefore, in a dense data set scenario, a smaller number of *SCPs* will be returned for lower value of overlap. As overlap threshold increases, the number of *SCPs* explodes.

4 Proposed *SIFT* framework

This section discusses our proposed *SIFT* framework.

4.1 Basic idea

Given a GTD D and the threshold values of $minRF_g$, $minCS_g$ and $maxO_g$ as input, the goal is to extract all the *SCPs* from D.

A brute-force approach would be to extract all of the possible subgraphs of D based on $minRF_g$, and then determine CS_g and $overlap_g$ for each combination of subgraphs by computing the corresponding $CSet_g$ values of the given pattern. Each combination of subgraphs of D could be a candidate *SCP*. The number of candidate *SCPs* formed by the subgraphs of even a small number of graph transactions would essentially explode, thereby making the extraction of *SCPs* extremely challenging and difficult to scale.

The basic idea is as follows. We convert the given graph transactions into the corresponding flat transactions. For this purpose, we extract all subgraphs from GTD and assign unique *S*ubgraph *ID*entifiers (*SID*) to each subgraph. Next, we convert each graph transaction into flat transaction by including the corresponding *SID*. Next, we propose an efficient methodology to extract *SCPs* from flat transactional dataset. We shall henceforth refer to this framework as subgraph ID-based flat transactional (*SIFT*) framework.

We can intuitively understand that *SIFT* provides opportunities for efficient determination of candidate sets. Further, it provides efficient way to compute coverage and overlap for each candidate set. This is because by considering each graph transaction as a set of *SIDs*, the coverage and overlap of a given subgraph pattern can be calculated through a set-based operation. Thus, we are essentially replacing complex and computationally expensive graph-based operations by set-based operations, which are typically faster by several orders of magnitude. Hence, the problem of extracting *SCPs* becomes the problem of extracting combinations of *SIDs* from the set of flat transactions. Thus, we propose a pattern mining-based extraction method by exploiting an overlap-related pruning heuristic, which we shall discuss now.

Incidentally, CS_g and $overlap_g$ threshold constraints do not satisfy the *downward closure property* [21]. However, we can exploit the overlap ratio measure proposed in [36] for extracting coverage patterns from a flat transactional dataset. The overlap ratio constraint satisfies *sorted closure property* [28]. Consider a candidate pattern $SP = \{S_p, S_q, \ldots, S_r\}$, where the subgraphs in SP are sorted in descending order of their relative frequencies. When overlap ratio of SP fails to satisfy the maximum overlap ratio threshold, any superset of SP cannot possibly satisfy the maximum overlap threshold. Hence, we can avoid generating supersets of SP. We use this heuristic in our proposed approach for effective pruning of candidate patterns. The steps to extract *SCPs* from flat transaction are as follows. First, we sort all of the candidate subgraphs in descending order of their *relative frequencies*. Then, starting from individual candidate subgraph as SP, we continue to generate candidate SP of progressively larger sizes, while using the pruning heuristic based on *sorted closure property* of *overlap ratio* to efficiently prune candidate SP.

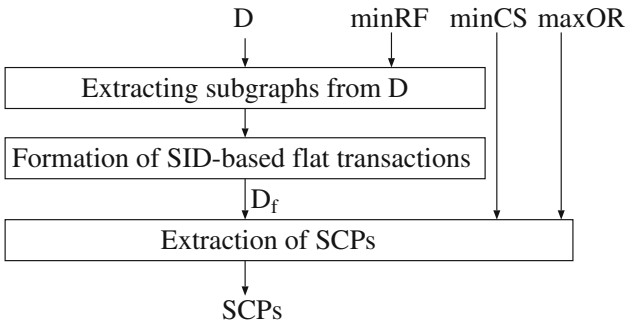


Fig. 3 Details of the *SIFT* framework

4.2 Details of the SIFT framework

Given D, $minRF_g$, $minCS_g$ and $maxO_g$, our proposed *SIFT* framework extracts *SCPs* from D. Figure 3 depicts the details of the *SIFT* framework. *SIFT* framework consists of the following steps (Algorithm 1):

(i) Extracting subgraphs from D
(ii) Formation of *SID*-based flat transactions and
(iii) Extraction of *SCPs*

We shall now explain these steps.

4.2.1 Extracting subgraphs from *D*

Based on the $minRF_g$ threshold, a subgraph discovery algorithm *gSpan* [42] is used to extract all the subgraphs from D subject to $minRF_g$ constraint (refer Sect. 2.2 for *gSpan algorithm*). We construct the set SG of subgraphs using *gSpan* algorithm, where each subgraph S_j is of the form $<Clabel, CSet>$, where *Clabel* represents *canonical label* of S_j and *CSet* consists of all *GIDs* of graph transactions that contains subgraph S_j.

Algorithm 1 : SIFT($D, minRF_g, minCS_g, maxOR_g$)

Inputs: D: Graph transactional dataset;
$minRF_g$: Minimum relative frequency threshold;
$minCS_g$: Minimum coverage support threshold;
$maxOR_g$: Maximum overlap ratio threshold
Output: *SCPs:* Set of *SCPs*
Variables: D_f: Set of flat transactions; S_j:<*Clabel, CSet*> A subgraph, where *Clabel* refers to canonical label of subgraph and *CSet* is a set of graph identifiers that contain S_j; SG: Set of subgraphs

1: $SG \leftarrow$ Extract all subgraphs from D using *gSpan algorithm* that satisfy $minRF_g$
2: D_f = Compute_Flat_Transactions(SG)
3: *SCPs* = Compute_*SCPs*($D_f, minCS_g, maxOR_g$)

4.2.2 Formation of SID-based flat transactions

The input to this step is a set of subgraphs SG of the form <*Clabel, CSet*>. In this step, we form the flat transaction for each graph transaction in D. The flat transaction contains the *SIDs* corresponding to *GID*. The details are given in Algorithm 2. In Algorithm 2, we maintain two hashmaps *SubList*: <*SID, Clabel*> and D_f:< f_i, *S(SID)*>, where f_i represents i^{th} flat transaction identifier and *S(SID)* represents set of $SIDs$ corresponding to the i^{th} graph transaction. For every subgraph S_j in SG, we check if the canonical label of S_j exists in *SubList.Clabel*. If it does not exist, we assign a new *SID* to S_j and insert *SID, Clabel* into *SubList*. Otherwise, we assign the *SID* to S_j corresponding to *Clabel* of S_j in *SubList.Clabel* (see Lines 2-9). In both the cases, for each subgraph S_j and for each *GID* in *CSet* of S_j, we insert *SID* of S_j into set $S(SID)$ of flat transaction identifier f_i corresponding to *GID* (see Lines 9–10). The set $< f_i, S(SID) >$ forms the *SID*-based flat transactional dataset D_f (see Line 14)).

The mapping of subgraphs in graph transaction G_i to *SIDs* in flat transaction f_i is a bijective function $\mathcal{F}$ represented as follows:

$$\forall S_j \in G_i, \forall O_h \in f_i, \mathcal{F} : S_j \rightarrow O_h$$

where O_h is an *SID* of S_j. When G_i has no subgraphs, f_i={ϕ}. Note that there are no duplicate *SIDs* in any flat transaction. The elements in each flat transaction are nothing but *SID* of subgraphs extracted from D subjective to the $minRF_g$ constraint. The constructed flat transactions do not represent all the features of original graph transaction, but represent only the subgraphs which satisfy $minRF_g$ constraint.

Let Ψ_{SID}= { $O_1, O_2, \ldots, O_m$ } be the set of m distinct *SIDs* in D_f. Let D_f={ $f_1, f_2, f_3, \ldots, f_n$ }, $\forall f_i \in D_f$, $f_i \subseteq \Psi_{SID}$, the set D_f forms the flat transactional dataset, where f_i is corresponding flat transaction of G_i, $\forall i = 1$ to n.

Algorithm 2 : Compute_Flat_Transactions(SG)

Input: SG: Set of subgraphs
Output: D_f: $< f_i, S(SID) >$ A flat transactional dataset, where f_i represents i^{th} flat transaction identifier and *S(SID)* represents set of *SIDs* corresponding to i^{th} graph transaction
Variables: S_j: <*Clabel, CSet*> a subgraph, where *Clabel* refers to canonical label of subgraph and *CSet* is a set of graph identifiers containing subgraph; *SubList*: <*SID, Clabel*> a hashmap, where *SID* is a subgraph identifier and *Clabel* is a canonical label of subgraph; *x, count*: Integers

1: $count = 0, D_f, SubList \leftarrow \phi$
2: **for** each subgraph $s \in SG$ **do**
3: **if** $s.Clabel$ not in $SubList.Clabel$ **then**
4: Insert $< count, s.Clabel >$ into $SubList$
5: $x \leftarrow count$
6: $count++$
7: **else**
8: $x \leftarrow SubList.SID$
9: **end if**
10: **for** each graph transaction i in $s.CSet$ **do**
11: Insert x into the set $D_f.S(SID)$ of corresponding flat transaction identifier f_i in D_f
12: **end for**
13: **end for**
14: **return** D_f

4.2.3 Extraction of SCPs

After converting GTD into *SID*-based flat transactional dataset, our objective is to extract *SCPs* subject to the constraints of $minRF_g, minCS_g$ and $maxO_g$. In this section, we explain the process to extract *SCPs* subject to the $minCS_g$ and $maxO_g$ constraints.

Under a brute-force approach, we would need to compute the values of $CS_g(SP, D)$ and $overlap_g$ for a prohibitively large number of candidate patterns formed by all *SIDs*. This is because the $minCS_g$ and $maxO_g$ constraints do not satisfy the *downward closure property* [21]. However, we can exploit the overlap ratio measure proposed in [18] for extracting coverage patterns from a flat transactional dataset. The overlap ratio measure satisfies the *sorted closure property* [28]. As explained in Sect. 2.2, the coverage pattern mining algorithm extracts coverage patterns subject to the constraints of minimum relative frequency (*minRF*), minimum coverage support (*minCS*) and maximum overlap ratio (*maxOR*).

Now, we explain the equivalence between the *minRF*, *minCS* and *maxOR* constraints for flat transactions (presented in Sect. 2.2 as defined in [18]) and $minRF_g$, $minCS_g$ and $maxO_g$ constraints associated with *SCPs* (defined in Sect. 3).

Recall that for flat transactions, the notion of relative frequency $RF(i_k)$ of an item i_k is the percentage of transactions, which contain i_k. In case of GTD, $RF_g(S_j, D)$ denotes the percentage of graph transactions, which contain a subgraph S_j. Furthermore, for flat transactions, the notion of coverage support $CS(X)$ of a pattern X is the percentage of the union of transactions covered by each item of X. In case of

GTD, $CS_g(SP, D)$ of a subgraph pattern SP denotes the percentage of the union of graph transactions covered by each subgraph of SP.

Regarding the overlap aspect, we have defined $overlap_g$ concept and $maxO_g$ constraint for GTD. First, we explain the overlap ratio (OR) constraint, which has been defined to extract coverage patterns for flat transactions [36]. Next, we explain how to employ the OR constraint to extract *SCPs* subject to the $maxO_g$ constraint.

Given a pattern X, and if the *elements* in X are sorted in the descending order of their relative frequency values, *Overlap Ratio* ($OR(X)$) of a pattern X, which satisfy the *sorted closure property*. We shall explain the *sorted closure property* after defining the *overlap ratio* of the pattern. The notion of $CSet$ of a pattern has been explained in Sect. 2.2.

Definition 2 (*Overlap ratio of a pattern X*) Let $X = \{O_p, O_q, \ldots, O_r, O_s\}$ be a pattern such that $RF(O_p) \geq RF(O_q) \geq \cdots \geq RF(O_r) \geq RF(O_s)$. (Here, the notations O_p, O_q, O_r and O_s represent *SIDs*.) The overlap ratio of a pattern X is defined as the ratio of the number of transactions common in $CSet(X - \{O_s\})$ and $CSet(O_s)$ to $CSet(O_s)$. It is defined as follows:

$$OR(X) = \frac{|CSet(X - \{O_s\}) \cap CSet(O_s)|}{|CSet(O_s)|}$$

For a pattern X, $0 \leq OR(X) \leq 1$. A pattern X is interesting if $OR(X) \leq maxOR$, where *maxOR* is a user-defined *maximum Overlap Ratio* threshold. A pattern X is said to be *non-overlap pattern* if $OR(X) \leq maxOR$ and $RF(O_h) \geq minRF, \forall O_h \in X$. Incidentally, it can be observed that the *maxOR* constraint follows the sorted closure property, which is explained below.

Definition 3 (*Sorted closure property*) Let the pattern $X=\{O_p, O_q, \ldots, O_r, O_s\}$, $1 \leq p \leq q \leq r \leq s \leq m$ such that the items in X are sorted in the descending order of their relative frequency values, i.e., $RF(O_p) \geq RF(O_q) \geq \cdots \geq RF(O_r) \geq RF(O_s)$. If $OR(X)$ is less than or equal to *maxOR*, i.e., $OR(X) \leq maxOR$, all the non-empty subsets of X containing O_s will also have OR less than or equal to *maxOR*.

Suppose, we extract the set S of coverage patterns from a given GTD with $OR(X) \leq \alpha$. We can compute $overlap_g(X)$ for all $X \in S$ and extract coverage patterns with $overlap_g(X) \leq \alpha$. For a given pattern X, the relationship between OR and $overlap_g$ is given in Theorem 1.

Theorem 1 *Consider a coverage pattern* $X=\{O_1, O_2, \ldots, O_p\}$ *with* $OR(X) \leq \alpha$. *Then,* $overlap_g(X) \leq \alpha$, *when* $p \leq \left(\frac{1+\alpha}{\alpha}\right)$.

Proof From the definition of $overlap_g$ in Sect. 3,

$$overlap_g(X) = \left(\frac{|M(X)|}{|CSet(X)|} - 1\right) \tag{5}$$

As X is a coverage pattern, $RF(O_1) \geq RF(O_2) \geq \cdots \geq RF(O_p)$. We consider a worst case scenario and assume that $|CSet(O_1)| = |CSet(O_2)| = \ldots = |CSet(O_p| = t$. So, $M(X) = p.t$ and $CSet(X) = p.t - (p-1)\alpha t$. Substituting the values of $M(X)$ and $CSet(X)$ in Equation 5,

$$\left(\frac{p.t}{p.t - (p-1)\alpha t} - 1\right) = \frac{\alpha(p-1)}{p - \alpha(p-1)} \tag{6}$$

By equating the above equation to α and solving for p,

$$\frac{\alpha(p-1)}{p - \alpha(p-1)} = \alpha; \implies p = \frac{1+\alpha}{\alpha}$$

Thus, we conclude that for a pattern X, when $OR(X) \leq \alpha$, $overlap_g(X) \leq \alpha$ if $p \leq \left(\frac{1+\alpha}{\alpha}\right)$. □

Algorithm 3 : Compute_*SCPs*(D_f, *minCS*, *maxOR*)

Inputs: D_f: Set of flat transactions;
minCS: Minimum coverage support;
maxOR: Maximum overlap ratio
Output: *SCPs*: Set of subgraph coverage patterns
Variables: NO_l: Set of l-size non-overlap patterns; C_1: Set of l-size candidate patterns; X: Pattern; l: Integer

```
1: NO_1 ← Set of frequent elements sorted in decreasing order of their
   relative frequencies
2: l=2, SCPs ← φ
3: while NO_{l-1} not empty do
4:    C_l ← NO_{l-1} ⋈ NO_{l-1}
5:    for each pattern X ∈ C_l do
6:       if OR(X, D_f) ≤ maxOR then
7:          NO_l ← NO_l ∪ X
8:          if CS(X, D_f) ≥ minCS then
9:             SCPs ← SCPs ∪ X
10:         end if
11:      end if
12:   end for
13:   l++
14: end while
15: return SCPs
```

Notably, we have employed two notions ($overlap_g$ and *overlap ratio*) to capture the notion of *overlap*. The notion of $overlap_g$ is intuitive from the user perspective, whereas *overlap ratio* (and *maxOR*) was employed as a pruning measure for efficient extraction of *SCPs*. For extracting *SCPs*, we can employ an existing coverage pattern algorithm such as a level-wise pruning based approach [32,36] or a pattern-growth approach [18], with the value of *minCS* equal to $minCS_g$ and the value of *maxOR* equal to $maxO_g$.

For extracting *SCPs* from flat transactional dataset, we employ coverage pattern mining algorithm proposed in [36]. Algorithm 3 depicts the coverage pattern mining algorithm. The inputs are flat transactional dataset D_f and user parameters *minCS* and *maxOR* values. Coverage pattern mining algorithm exploits apriori like level-wise search approach to find the l-size candidate patterns from (l-1)-size *non-overlap patterns* (see Lines 3–4). A *non-overlap pattern* is a pattern that satisfies *maxOR* constraint. It uses *sorted closure property* to prune the search space and extracts all *non-overlap patterns*, which become the candidates for the next iteration (see Lines 5–7). The considered *non-overlap patterns* that satisfy the *minCS* constraint are considered as the *SCPs* (see Lines 8–9). This process is repeated until no new *non-overlap* patterns are generated.

After extracting the set of *SCPs*, top-k *SCPs* can be listed by considering a ranking criteria based on CS_g or a combination of CS_g and $overlap_g$ values of *SCPs* based on the specific requirements of the application domain.

4.3 Time complexity

The time complexity of the proposed *SIFT* framework is equals:

$$O(kmn + rm) + O(mq) + \sum_{l=1}^{m} l(|C_{l-1}| \cdot |C_{l-1}|) \tag{7}$$

where $O(kmn + rm)$ is the complexity of subgraph extraction, $O(mn)$ is the time complexity of flat transactions modeling, and $\sum_{l=1}^{m} l(|C_{l-1}| \cdot |C_{l-1}|)$ is the time complexity of *SCPs* computation. The explanation of each term in Equation 7 is as follows:

First, in *SIFT* framework, we employ *gSpan* algorithm to extract subgraphs from GTD. As mentioned in [2], the complexity of *gSpan* algorithm to extract all subgraphs from GTD is $O(kmn + rm)$, where k is the maximum number of subgraph isomorphism tests, m is the number of frequent subgraphs, n is the number of graph transactions, and r is the maximum number of duplicate codes of the frequent subgraphs that grow from other minimum DFS codes. It can be noted that extraction of all subgraphs from a given graph transaction is an NP-complete problem [2]. To improve performance, the *gSpan* algorithm employs the notion of *minimum DFS code* and converting the subgraph extraction problem into a pattern mining problem through string comparison. In practical scenarios, the value of $m \leq n$, the value of r is much less than n and the value of k is small for sparse and diverse labels. Hence, the time complexity for subgraph extraction depends on the value of $m \times n$.

Second, the process to compute the flat transactional dataset from the set of <canonical label of a subgraph, set of the corresponding GIDs> produced in the preceding step consists of two steps. First, we assign an *SID* to each unique canonical label (*Clabel*) and compute an hashmap $< SID, Clabel >$. The search time for the existence of *Clabel* in the hashmap take $O(1)$. Second, after mapping the *Clabel* with unique *SID*, for each corresponding *GID*, we will insert *SID* into the corresponding flat transaction identifier. The search time to insert is $O(1)$. Consider that on average, each *SID* belongs to q number of transactions. The time complexity to compute the flat transactional dataset is bounded by $O(m \times q)$. Notably, $q \ll m$. Therefore, the time complexity to model flat transactions from graph transactions is proportional to m.

Third, in *SIFT* framework, we employ an iterative level-wise apriori based algorithm. The time complexity of an iterative pruning algorithm is $\sum_{l=1}^{m} l(|C_{l-1}| \cdot |C_{l-1}|)$, where $|C_{l-1}|$ is the number of candidate patterns of size l (Refer to Chapter 6 of [38]). In the proposed *SIFT* framework, the number of candidate patterns generated depends on the value of *overlap ratio* threshold *maxOR*. Normally, at lower values of *maxOR*, less number of candidate patterns are produced at each level.

Overall, the time complexity of proposed *SIFT* framework depends on the graph transactional dataset size n, number of subgraphs extracted m and number of candidate patterns generated. Note that the value of m depends on *minRF* threshold and the number of candidate patterns depends on *maxOR* threshold. By choosing proper values of *minRF* and *maxOR* threshold values, it is indeed capable of extracting subgraph coverage patterns from graph transactional dataset.

5 Performance evaluation

We conducted our experiments in the ADA cluster [1] (at IIIT Hyderabad), which consists of 42 Boston SYS-7048GR-TR nodes equipped with dual Intel Xeon E5-2640 v4 processors, providing 40 virtual cores per node. The aggregate theoretical peak performance of ADA is 47.62 TFLOPS. We have conducted experiments on 20 virtual machines. Each virtual machine is allocated with 2 GB memory. We also reported the experiments on the scalability aspect of our proposed approach by varying the number of virtual machines from 5 to 40. We implemented our proposed schemes in Python 3.0. The link to the code for the implementation is provided in the footnote.[1]

We used three real datasets, namely Yeast 167 (Yeast anticancer), P388 (Leukemia), from Pubchem [3,43] and Zinc dataset consisting of drug-like molecules [37]. The Yeast 167 and P388 datasets consist of chemical compounds, which are modeled as graph transactions. In these datasets, each

[1] https://github.com/srinivas2234/SCPs.

Table 2 Summary of the real datasets

Dataset	#graph transactions	Avg. density	#vertex labels	#edge labels	Avg. size of graph
Yeast	79601	0.0537	75	3	40.7
P388	41472	0.052	73	3	41.8
Zinc	4672	0.73	15	10	1.8

Table 3 Parameters of the SIFT framework performance evaluation

Parameter	Default		Variations	
	P388, Yeast	Zinc	P388, Yeast	Zinc
minRF	0.3	0.025	0.3-1 (step size=0.1)	0.025-1 (step size=0.05, 0.1)
maxOR	0.3	0.5	0-1 (step size=0.1)	0-1 (step size=0.1)
minCS	0.7	0.7	0.3-1 (step size=0.1)	0-1 (step size=0.1)
N_M	20	20	5-40 (step size=5)	NIL

chemical compound is modeled as graph, where chemical elements are represented as vertices and chemical bonds among them are represented as edges. We have reported our case study by considering the Zinc dataset. The Zinc dataset consists of drug-like molecules docked with 1WOF protein to form a protein–ligand complex. Table 2 summarizes the three datasets.

To the best of our knowledge, there exists no other approach for extracting SCPs from a GTD. As the number of subgraphs increases, the complexity of a naïve brute-force approach for extracting *SCPs* increases exponentially as it requires the determination of the coverage and overlap values based on prohibitively expensive graph-based computations. Hence, in the absence of any meaningful reference approach for comparison, we define the objective of our performance evaluation toward demonstrating the feasibility of the proposed *SIFT* framework in extracting *SCPs* from a given dataset.

We have conducted the experiments by implementing three components of the *SIFT* framework as follows. First, we employ the *gSpan algorithm* [42] for extracting all candidate subgraphs from a given GTD and assign *SIDs* to the extracted subgraphs. Second, we employ the proposed *SIFT* framework to form the transformed flat transactional dataset over the extracted *SIDs*. Third, to extract *SCPs* from the transformed flat transactions, we use the MapReduce-based coverage pattern mining algorithm [32], which was proposed to extract coverage patterns from flat transactions. Table 3 summarizes the parameters of our performance study.

The performance metrics for extracting *SCPs* are (i) processing time (T_S) to extract subgraphs, assign *SIDs* and form *SID*-based flat transactions, (ii) number of candidate subgraphs (N_S), (iii) average number of *SIDs* (AVG) in the *SID*-based flat transactions, (iv) processing time (T_{SCP}) to extract *SCPs* from flat transactions, (v) number of patterns (N_P) to be examined for extracting *SCPs* and (vi) number of *SCPs* (N_{SCP}). Here, T_S represents the processing time consumed to extract subgraphs by accessing the graph dataset from the disk. T_{SCP} is the processing time consumed for extracting *SCPs* from the *SID*-based transactional dataset, which resides on disk.

5.1 Effect of varying *minRF*

The results in Fig. 4 depict the effect of varying *minRF*. The results in Fig. 4a indicate the performance of T_S, while the results in Fig. 4b show the performance of N_S as we vary *minRF* for the P388 and Yeast datasets. The results in Fig. 4a show that when the value of *minRF* is low, the value of T_S is high. As *minRF* is increased, the value of T_S reduces exponentially due to the pruning effect of *minRF*. The value of T_S depends upon the number of subgraphs extracted from the dataset. The results in Fig. 4b show that the number of *SIDs* decreases with increase in the value of *minRF*. This occurs due to decrease in the number of subgraphs that satisfy the *minRF* constraint. The results in Fig. 4c depict the effect of varying *minRF* on *AVG*. The results show that when the value of *minRF* is low, the value of *AVG* is high, and as *minRF* increases, the value of *AVG* is decreased. This is because at lower value of *minRF*, there will be a large number of subgraphs, which satisfy the *minRF* constraint. As the value of *minRF* increases, the number of subgraphs decreases because less number of subgraphs satisfy the *minRF* constraint. The value of *AVG* depends on the number of subgraphs that are extracted. Therefore, at lower values of *minRF*, a transactional dataset with large *AVG* is extracted.

The results in Figs. 4d, e and f show that the values of T_{SCP}, N_P and N_{SCP} decrease with the increase in the value of *minRF*, respectively. The reason is that at lower value of *minRF*, there will be large number of subgraphs satisfying the *minRF* constraint. As *minRF* increases, the value of N_P decreases because less number of patterns satisfy the *minRF* constraint. Consequently, the values of T_{SCP} and N_{SCP} also decrease.

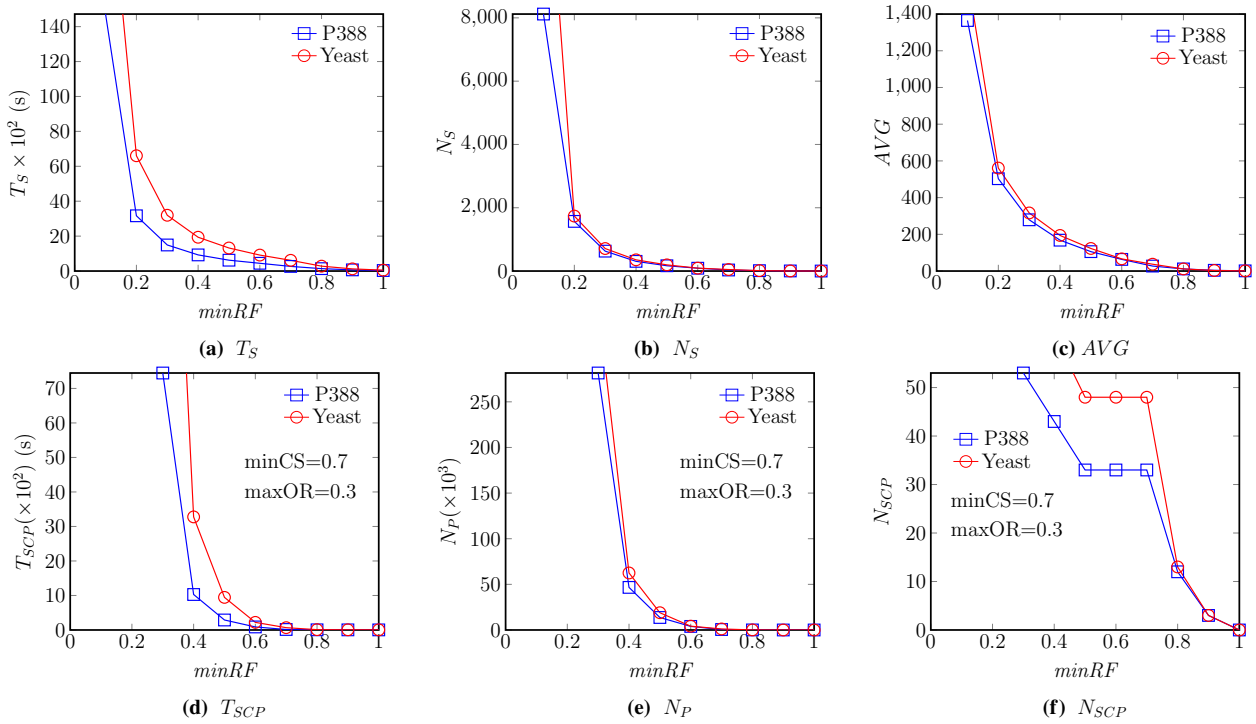


Fig. 4 Effect of varying *minRF*

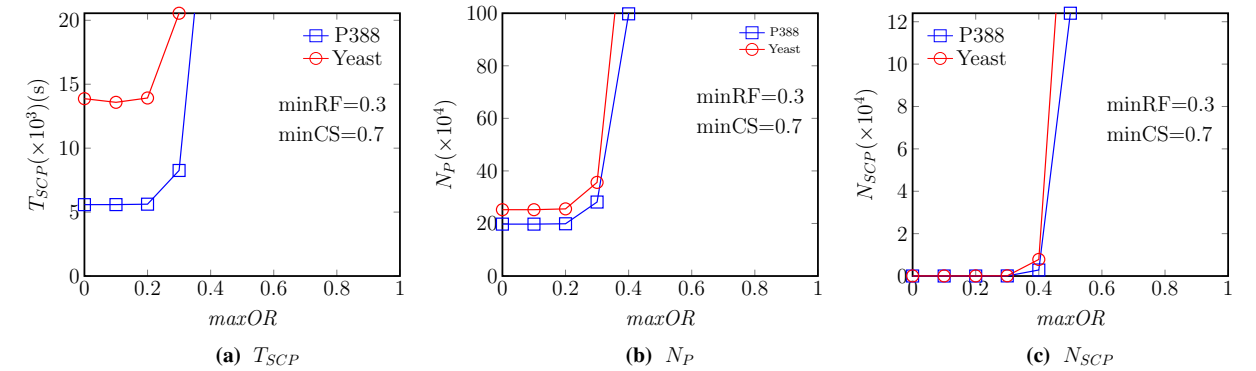


Fig. 5 Effect of varying *maxOR*

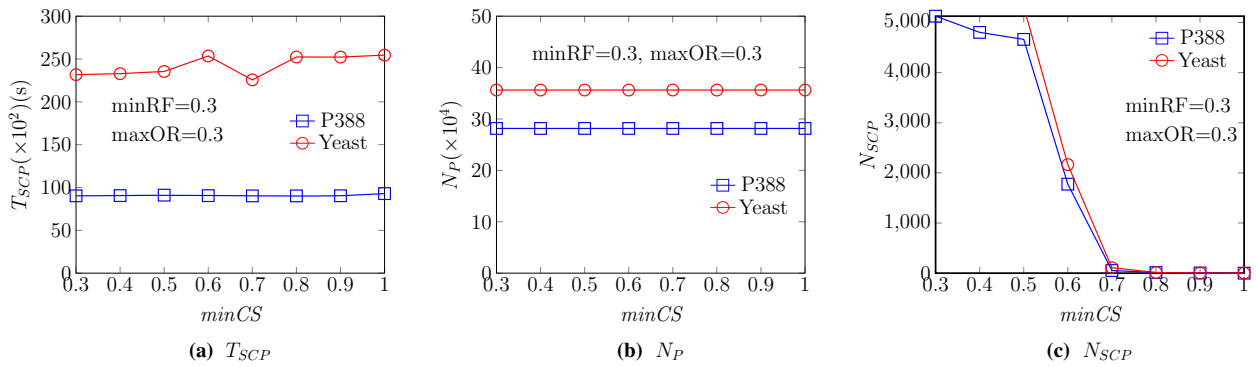
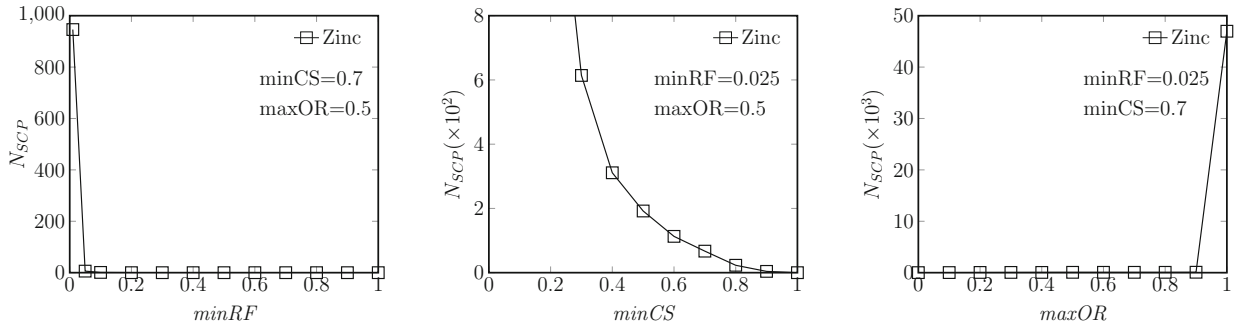
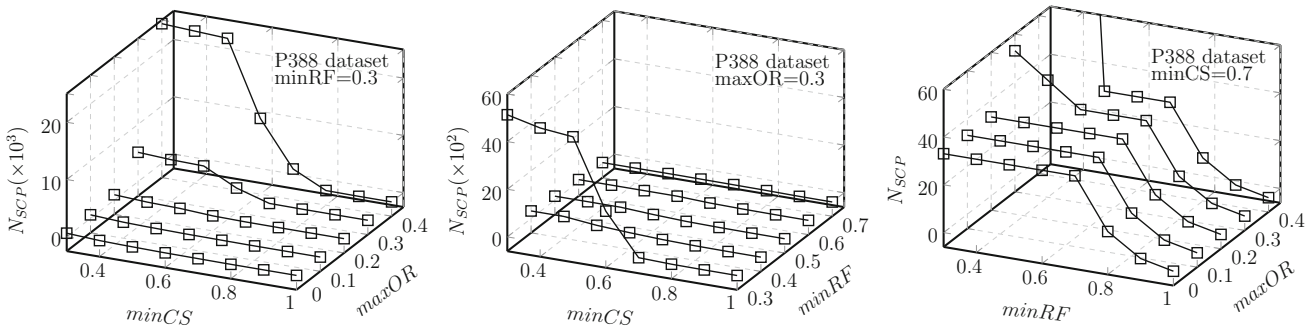


Fig. 6 Effect of varying *minCS*



(a) *minRF vs* N_{SCP} (b) *minCS vs* N_{SCP} (c) *maxOR vs* N_{SCP}

Fig. 7 Effect of varying *minRF, minCS and maxOR* on Zinc dataset



(a) *minCS vs maxOR vs* N_{SCP} (b) *minCS vs minRF vs* N_{SCP} (c) *minRF vs maxOR vs* N_{SCP}

Fig. 8 Effect of varying (**a**) *minCS* and *maxOR* **b** *minCS* and *minRF* and **c** *minRF* and *maxOR*

It can be observed that we have reported results starting from *minRF*=0.1 as we could not experiment with *minRF* less than 0.1 due to explosion in the number of patterns.

5.2 Effect of varying *maxOR*

The results in Fig. 5 depict the effect of varying the value of *maxOR*. The results in Figs. 5a, b and c show that T_{SCP}, N_P and N_{SCP} increase with the increase in *maxOR*, respectively. The reason is that at lower value of *maxOR*, there will be less number of patterns, which satisfy the *maxOR* constraint, thereby resulting in less value of T_{SCP} and N_{SCP}. As *maxOR* increases, the value of N_P increases because more patterns satisfy the *maxOR* constraint, which increases the value of T_{SCP} and N_{SCP}. It can be observed that even at *maxOR*=0, there are 48 *SCPs* in Yeast and 13 *SCPs* in P388. At higher values of *maxOR*, more number of *SCPs* can be extracted.

5.3 Effect of varying *minCS*

The results in Fig. 6 depict the effect of varying the value of *minCS*. The results in Fig. 6a and b indicate that the values of T_{SCP} and N_P remain comparable for all the values of *minCS* for both datasets. The reason is as follows. When the values of *minRF* and *maxOR* are fixed, the same number of candidate patterns is examined to extract the *SCPs*. Therefore, as expected, irrespective of variations in the value of *minCS*, both the values of T_{SCP} and N_P remain comparable.

The results in Fig. 6c indicate that the value of N_{SCP} decreases with increase in the value of *minCS*. Notably, in the proposed approach, after satisfying the *maxOR* constraint, we prune a candidate pattern if it does not satisfy the *minCS* constraint. At higher values of *minCS*, a candidate pattern will be pruned even though it satisfies the *maxOR* constraint. As a result, the value of N_{SCP} reduces as we increase the value of *minCS*.

The results in Figs. 7a–c depict the effect of varying the values of *minRF*, *minCS* and *maxOR*, respectively, for Zinc dataset. The results depict trends similar to P388 and Yeast datasets. The value of N_{SCP} is small due to the small size of Zinc dataset.

5.4 Performance results with 3D plots

The results in Fig. 8a depict the effect of varying the values of *minCS* and *maxOR*. The result shows that when the values of *minCS* and *maxOR* are low, the value of N_{SCP} is small due to candidate pruning based on *maxOR* constraint.

When the values of *minCS* are high and *maxOR* is low, the value of N_{SCP} decreases further because very few patterns satisfy high value of *minCS* and low value of *maxOR*. When the value of *minCS* is low and *maxOR* is high, the value of N_{SCP} is high because more number of patterns satisfy low value of *minCS* and high value of *maxOR*. However, when the values of *minCS* and *maxOR* are high, the value of N_{SCP} will decrease because there are few patterns that may satisfy the high value of *minCS*.

The results in Fig. 8b depict the effect of varying the values of *minCS* and *minRF*. The result shows that at low values of *minCS* and *minRF*, the value of N_{SCP} is high. This is because, large number of candidate patterns will be generated at lower values of *minRF* and most of them satisfy the low value of *minCS* constraint. When *minCS* is high and *minRF* is low, the value of N_{SCP} low because, less number of candidate patterns satisfy the high value of *minCS* threshold. At low values of *minCS* and high values of *minRF*, the value of N_{SCP} is low, due to small number of candidate pattern generation. Further, when the values of *minRF* and *minCS* are high, the value of N_{SCP} decreases further.

The results in Fig. 8c depict the effect of varying the values of *maxOR* and *minRF*. The results show that N_{SCP} does not vary much at lower values of *minRF* and *maxOR*. When we increase the value of *minRF*, the value of N_{SCP} decreases due to decrease in number of candidate pattern. When the values of *minRF* and *maxOR* are high, the value of N_{SCP} is less, due to less number of candidate patterns. However, when the values of *minRF* and *maxOR* are high, the number of *SCPs* explodes due to large number of candidate pattern generation.

5.5 Effect of varying N_M

Figure 9 depicts the effect of varying the number N_M of machines. Observe that the value of T_{SCP} decreases with increase in the value of N_M. This is due to increase in the parallel extraction of *SCPs*. However, the change in the value of T_{SCP} decreases with increase in N_M and exhibits a saturation effect when more than 30 machines are used. This is due to communication overhead. The results show that the value of T_{SCP} can be reduced by employing additional resources.

Given a dataset, the processing time to extract *SCPs* equals the sum of the processing time to form *SID*-based graph transactions (as depicted in Figs. 4a) and the processing time to extract *SCPs* (as depicted in Fig. 9). Overall, the results demonstrate that it is feasible to extract the knowledge of *SCPs* by processing a reasonable size dataset of Yeast with 79601 graph transactions.

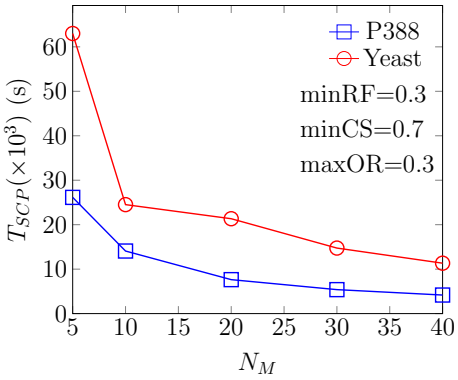


Fig. 9 Effect of varying $\mathbf{N_M}$

5.6 Discussion about setting thresholds in *SIFT*

In this approach, conversion of graph transactions into *SID*-based flat transactions is one time computation process. Once graph transactions are transformed to flat transactions, it is always possible to choose relative frequency thresholds greater than $minRF_g$ and compute *SCPs* for various values of $minCS_g$ and $maxOR_g$.

Now let us discuss how to set the values of the parameters such as $minRF_g$, $minCS_g$ and $maxOR_g$. The $minRF_g$ threshold value can be set to half the maximum $minRF_g$ value. Then, based on number of coverage patterns, $minRF_g$ can be decreased. The goal is to extract *SCPs* with maximum coverage, while minimizing the overlap to zero. Hence, as a heuristic, we could start with the coverage support value equal to 1 and then progressively keep decreasing the value of coverage support until a desired number of *SCPs* is obtained. Regarding $maxOR_g$, we can start with $maxOR_g$ equal to 0 and then progressively keep increasing the value of $maxOR_g$ until a desired number of *SCPs* can be obtained.

Based on the application, the domain expert can first extract *SCPs* by setting *maxOR*=0 and *minCS*=1. If the domain expert needs more number of *SCPs*, he can increase *maxOR* or decrease *minCS* progressively. Normally, the process of pattern mining is an iterative approach. As we have proposed a pattern mining model, the usual methodologies employed to set threshold values can be employed in this case also.

6 Case study: usefulness of *SCPs* in drug design

We demonstrate the feasibility of applying knowledge of *SCPs* in computer-aided drug design toward developing a drug for coronavirus. Corona viruses that include the SARS coronavirus 2 responsible for the COVID-19 pandemic are pathogens that cause various diseases that are

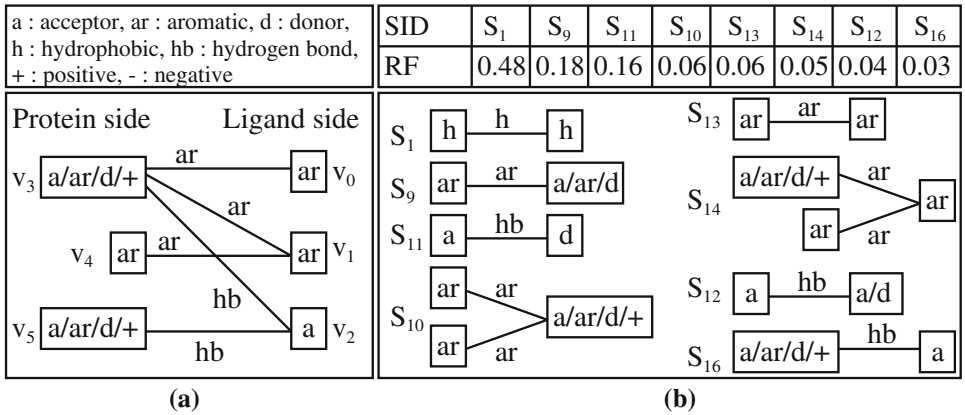


Fig. 10 **a** Sample graph modeling of protein–ligand complex, **b** candidate subgraphs and corresponding *minRF* values

Table 4 Top 8 *SCPs* extracted from PLC dataset

S.No	Subgraph coverage pattern	Coverage support	Overlap ratio
1	$\{S_1, S_9, S_{11}, S_{12}, S_{16}\}$	0.9	0.25
2	$\{S_1, S_9, S_{11}, S_{12}\}$	0.87	0.12
3	$\{S_1, S_9, S_{11}\}$	0.83	0
4	$\{S_1, S_{11}, S_{10}, S_{13}, S_{12}, S_{16}\}$	0.83	0.17
5	$\{S_1, S_{11}, S_{10}, S_{14}, S_{12}, S_{16}\}$	0.81	0.17
6	$\{S_1, S_{11}, S_{13}\}$	0.72	0.0
7	$\{S_1, S_{11}, S_{10}\}$	0.71	0.0
8	$\{S_1, S_{11}, S_{14}\}$	0.7	0.0

sometimes fatal in human beings. Coronavirus main protease enzymes (CoV-Mpro) are crucial for virus replication. Drugs designed to inhibit this class of enzymes help in treating coronavirus infection [45]. We consider Zinc database comprising 250000 drug-like molecules. We selected Mpro protein (PDB ID: 1WOF) using the Autodock 4.2 software program [30]. Molecular docking procedure takes each of the 250000 molecules, identifies the best binding mode with the protein, and gives the binding affinity and the protein–ligand bound complex (PLC) structure using a scoring function. Better the intermolecular interactions between the protein and the ligand, better is the binding affinity and better is the molecule for it to be a drug. The top-1000 molecules among the 250000 molecules in the initial dataset that yielded high binding affinity with the protein molecule were chosen for mining *SCPs*.

For our case study, protein–ligand complexes (PLCs) are converted to graphs transactions using GReMLIN [34]. A ligand can interact with the same protein at different sites, producing multiple graphs for the same protein and ligand, but different vertex and edge label sets. Here, vertices are amino acids of proteins and atoms of ligands and the edges are interactions between amino acids and ligands. Example 4 presents the modeling of PLC as a graph transaction.

Example 4 Consider a sample PLC modeled as a graph transactions $G=(V,E,L,l)$ shown in Fig. 10a. The left side part nodes belong to protein and the right side part nodes belong to ligand. Here, $V=\{v_0, v_1, \ldots, v_5\}$, $E=\{(v_0,v_3), (v_1,v_3), \ldots, (v_2,v_5)\}$, $L=\{$*aromatic*, *acceptor*, *acceptor/aromatic/donor/positive*, *aromatic bond* and *hydrogen bond*$\}$. A mapping function l maps the vertices $v_0, v_1, \ldots, v_5$ to *aromatic*, *aromatic*, ..., *acceptor/aromatic/donor/positive* and edges $(v_0,v_3), (v_1,v_3), \ldots, (v_3,v_5)$ to *aromatic bond*, *aromatic bond*, ..., *hydrogen bond*$\}$, respectively.

The top-1000 ligands interact with 1WOF protein and form 1000 protein–ligand complexes. GReMLIN generated 4672 graph transactions from these 1000 PLCs. We extracted *SCPs* by providing *minRF*=0.025, *minCS*=0.7 and *maxOR*=0.5. The time consumed to extract subgraph coverage patterns is about 10 seconds (3.52 seconds to model flat transaction from graph transactions and 6.03 seconds to extract subgraph coverage patterns from flat transactions). The top-8 candidate subgraphs along with their corresponding *RF* values are depicted in Fig. 10b, and the top-8 *SCPs* sorted by *coverage support* and their corresponding *overlap ratio* are provided in Table 4.

Consider an *SCP* $\{S_1, S_9, S_{11}\}$ that covers 83% of Zinc dataset with 0 overlap. Figure 11a depicts the overall structure of the Mpro protein and highlights the region, where a drug molecule could bind. We have analyzed the interactions among all residues that have interactions with at least one of the 1000 ligands in the dataset. The analysis regarding the utility of *SCPs* in understanding protein–ligand interactions and its possible inputs to drug design efforts is as follows.

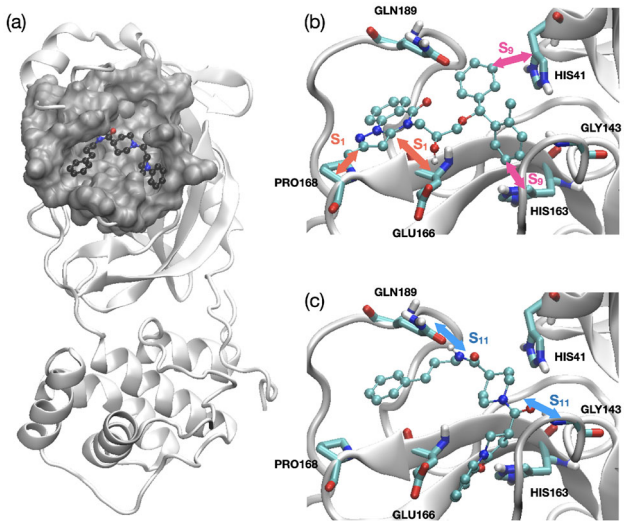


Fig. 11 **a** Structure of the MPro protein bound to a ligand. **b** Six selected protein amino acids and interactions with a ligand molecule corresponding to S_1 and S_9 subgraphs **c** Protein–ligand interactions corresponding to S_{11} subgraph

Figure 11b depicts an example of a ligand in which interactions corresponding to S_1 and S_9 subgraphs are possible (orange and pink arrows). As shown in Fig. 11b, it is evident that the method captured the hydrophobic interaction S_1 and aromatic interaction S_9, which contribute toward the favorable binding affinity between the protein and the ligand. On the other hand, Fig. 11c depicts the protein–ligand hydrogen bonding interactions involving another ligand that represent the S_{11} subgraph. Similar to the aromatic and hydrophobic interactions above, the approach captures the hydrogen bonded interaction efficiently. These three subgraphs have an overall coverage of 83% with overlap ratio as zero indicating their prevalence and hence importance for the molecules to bind to the protein. Such a new knowledge gives possible directions for improving the molecule by modifying the structure of these ligands so that multiple modes of interactions are possible and hence, improve the binding affinities. Therefore, the proposed *SIFT* framework not only helps in understanding the protein–ligand interactions, but also helps in designing better drugs by extracting the knowledge of subgraph coverage patterns.

7 Conclusion

Subgraph pattern mining is an active research area with applications in the domains of chemical, biological and social networks. Given graph transactional data, existing works have focused on the problem of extracting frequent subgraphs, but they have not considered the problem of extracting the knowledge of coverage-related subgraph patterns. Hence, we have introduced the concept of *subgraph coverage patterns*. In particular, we have proposed the *SIFT* framework for extracting subgraph coverage patterns from graph transactional data based on *minRF*, *minCS* and *maxO* constraints. Our performance evaluation with three real datasets demonstrates the effectiveness of the proposed scheme in terms of processing time and pruning efficiency. We have also demonstrated the feasibility of applying the knowledge of *SCPs* through a case study in the bioinformatics domain. To the best of our knowledge, this is the first work to consider the extraction of subgraph coverage patterns from graph transactional data. Given the prevalence of graph data modeling, the proposed model of *SCPs* has a potentially huge scope for opening up new avenues for the extraction of interesting knowledge from graph datasets in several important and diverse domains.

Funding The research of A Srinivas Reddy and P Krishna Reddy is supported by India-Japan Joint Research Laboratory Project entitled "Data Science based farming support system for sustainable crop production under climatic change (DSFS)," funded by Department of Science and Technology, India (DST) and Japan Science and Technology Agency (JST). The research of U Devapriya Kumar is supported by IHub-Data, IIIT Hyderabad.

Data Availability The datasets Yeast 167 (Yeast anti-cancer), P388 (Leukemia) are available at https://sites.cs.ucsb.edu/~xyan/dataset.htm

Declarations

Conflicts of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Code availability The source code is available at https://github.com/srinivas2234/SCPs.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. ADA. http://hpc.iiit.ac.in/wiki/index.php/Ada_User_Guide (Accessed in September 2021)
2. UIUC technical report, UIUCDCS-R-2002-2296. https://sites.cs.ucsb.edu/~xyan/papers/gSpan.pdf (Accessed in September 2021)
3. Pubchem. https://pubchem.ncbi.nlm.nih.gov/ (2021)
4. Aida, M., Pieter, M., Wout, B., Pieter, M., Boris, C., Bart Goethals, K.L.: Grasping frequent subgraph mining for bioinformatics applications. BioData Min. **11**(1), 1–20 (2018)
5. Alsallakh, B., Aigner, W., Miksch, S., Hauser, H.: Radial sets: interactive visual analysis of large overlapping sets. IEEE Trans. Visual Comput. Gr. **19**(12), 2496–2505 (2013)
6. Amiri, A., Salari, M.: Time-constrained maximal covering routing problem. OR Spectrum **41**(2), 415–468 (2019)
7. Andrew, G.D., Paola, V.L.: The minimal hitting set generation problem: Algorithms and computation. SIAM **31**(1), 63–100 (2017)

8. Ayed, R., Hacid, M.S., Haque, R., Jemai, A.: An updated dashboard of complete search FSM implementations in centralized graph transaction databases. J. Intell. Inf. Syst. **55**, 149–182 (2020)
9. Borgelt, C., Berthold, M.R.: Mining molecular fragments: finding relevant substructures of molecules. In: Proceedings of the ICDM, pp. 51–58 (2002)
10. Charu C, A., Haixun, W.: Managing and mining graph data, vol. 40. Springer (2010)
11. Chen, J., Lin, Y., Li, J., Lin, G., Ma, Z., Tan, A.: A rough set method for the minimum vertex cover problem of graphs. Appl. Soft Comput. **42**, 360–367 (2016)
12. Chvatal, V.: A greedy heuristic for the set-covering problem. Math. Oper. Res. **4**(3), 233–235 (1979)
13. Cormode, G., Karloff, H., Wirth, A.: Set cover algorithms for very large datasets. In: Proceedings of the ACM CIKM, pp. 479–488 (2010)
14. Dehaspe, L., Toivonen, H., King, R.D.: Finding frequent substructures in chemical compounds. In: Proceedings of the KDD, pp. 30–36 (1998)
15. Fazekas, K., Bacchus, F., Biere, A.: Implicit hitting set algorithms for maximum satisfiability modulo theories. In: Proceedings of the IJCAR, pp. 134–151 (2018)
16. Fortin, S.: The graph isomorphism problem: Technical report. Univ Alberta, Edmonton (1996)
17. Fournier Viger, P., Cheng, C., Lin, J.C.W., Yun, U., Kiran, R.U.: TKG: Efficient mining of top-*k* frequent subgraphs. In: Proceedings of the Big Data Analytics, pp. 209–226 (2019)
18. Gowtham Srinivas, P., Krishna Reddy, P., Trinath, A.V., Bhargav, S., Uday Kiran, R.: Mining coverage patterns from transactional databases. J. Intell. Inf. Syst. **45**, 423–439 (2015)
19. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Data reduction and exact algorithms for clique cover. ACM J. Exp. Algorithm. pp. 2.2–2.15 (2009)
20. Guevara, V.I.G., Calderon, S.G., Cabrera, E.A., Calvo, H.: Symbolic learning for improving the performance of transversal-computation algorithms. IEEE Access **7**, 19752–19761 (2019)
21. Han, J., Cheng, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Min. Knowl. Disc. **15**(1), 55–86 (2007)
22. Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery, pp. 13–23 (2000)
23. Jiang, C., Coenen, F., Zito, M.: A survey of frequent subgraph mining algorithms. Knowl. Eng. Rev. **28**(1), 75–105 (2013)
24. Jiang, H., Wang, H., Philip, S.Y., Zhou, S.: GString: A novel approach for efficient search in graph databases. In: Proceedings of the ICDE, pp. 566–575 (2007)
25. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: Proceedings of the ICDM, pp. 313–320 (2001)
26. Li, R., Wang, W.: REAFUM: Representative approximate frequent subgraph mining. In: Proceedings of the ICDM, pp. 757–765. SIAM (2015)
27. Li, Y., Fan, J., Wang, Y., Tan, K.L.: Influence maximization on social graphs: A survey. IEEE TKDE **30**(10), 1852–1872 (2018)
28. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proceedings of the ACM SIGKDD, pp. 337–341 (1999)
29. Medina, S.G., Fassio, A.V., de A. Silveira, S., da Silveira, C.H., de Melo-Minardi, R.C.: CALI: A novel visual model for frequent pattern mining in protein-ligand graphs. In: International Conference on Bioinformatics and Bioengineering, pp. 352–358 (2017)
30. Morris, G.M., Huey, R., Lindstrom, W., Sanner, M.F., Belew, R.K., Goodsell, D.S., Olson, A.J.: AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility. J. Comput. Chem. **30**(16), 2785–2791 (2009)
31. Orenstein, Y., Pellow, D., Marçais, G., Shamir, R., Kingsford, C.: Designing small universal *k*-mer hitting sets for improved analysis of high-throughput sequencing. PLoS Comput. Biol. **13**(10), 1–15 (2017)
32. Ralla, A., Siddiqie, S., Reddy, P.K., Mondal, A.: Coverage pattern mining based on MapReduce. In: Proceedings of the ACM IKDD CoDS-COMAD, pp. 209–213 (2020)
33. Rehman, S.U., Khan, A.U., Fong, S.: Graph mining: A survey of graph mining techniques. In: Proceedings of the International Conference on Digital Information Management, pp. 88–92 (2012)
34. Ribeiro, V.S., Santana, C.A., Fassio, A.V., Cerqueira, F.R., da Silveira, C.H., Romanelli, J.P.R., Patarroyo-Vargas, A., Oliveira, M.G.A., Gonçalves-Almeida, V., Izidoro, S.C., de Melo-Minardi, R.C., Silveira, S.d.A.: visGReMLIN: Graph mining-based detection and visualization of conserved motifs at 3D protein-ligand interface at the atomic level. BMC Bioinformatics **21**(2), 1–12 (2020)
35. Santana, C.A., Cerqueira, F.R., Da Silveira, C.H., Fassio, A.V., De Melo-Minardi, R.C., Silveira, S.d.A.: GReMLIN: A graph mining strategy to infer protein-ligand interaction patterns. In: IEEE International Conference on Bioinformatics and Bioengineering, pp. 28–35 (2016)
36. Srinivas, P.G., Reddy, P.K., Bhargav, S., Kiran, R.U., Kumar, D.S.: Discovering coverage patterns for banner advertisement placement. In: Proceedings of the PAKDD, pp. 133–144 (2012)
37. Sterling, T., Irwin, J.J.: ZINC 15 - Ligand discovery for everyone. J. Chem. Inf. Model. **55**(11), 2324–2337 (2015)
38. Tan, P.N., Steinbach, M., Karpatne, A., Kumar, V.: Introduction to Data Mining, 2*nd* edn. Pearson (2018)
39. Wagner, M., Friedrich, T., Lindauer, M.: Improving local search in a minimum vertex cover solver for classes of networks. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1704–1711 (2017)
40. Wang, C., Xie, M., Bhowmick, S.S., Choi, B., Xiao, X., Zhou, S.: FERRARI: an efficient framework for visual exploratory subgraph search in graph databases. VLDB J. pp. 1–26 (2020)
41. Wu, J., Li, C.M., Jiang, L., Zhou, J., Yin, M.: Local search for diversified top- k clique search problem. Computers & Operations Research **116**, 104867 (2020)
42. Xifeng Y., Jiawei H.: gSpan: Graph-based substructure pattern mining. In: Proceedings of the ICDM, pp. 721–724 (2002)
43. Yan, X., Cheng, H., Han, J., Yu, P.S.: Mining significant graph patterns by leap search. In: Proceedings of the ACM SIGMOD, pp. 433–444 (2008)
44. Yan, X., Yu, P.S., Han, J.: Graph indexing: a frequent structure-based approach. In: Proceedings of the ACM SIGMOD, pp. 335–346 (2004)
45. Yang, H., Xie, W., Xue, X., Yang, K., Ma, J., Liang, W., Zhao, Q., Zhou, Z., Pei, D., Ziebuhr, J., Hilgenfeld, R., Yuen, K.Y., Wong, L., Gao, G., Chen, S., Chen, Z., Ma, D., Bartlam, M., Rao, Z.: Design of wide-spectrum inhibitors targeting coronavirus main proteases. PLoS Biol. **3**(10), 1742–1752 (2005)
46. Zareie, A., Sheikhahmadi, A., Khamforoosh, K.: Influence maximization in social networks based on TOPSIS. Expert Syst. Appl. **108**, 96–107 (2018)
47. Zhefeng, W., Enhong, C., Qi, L., Yu, Y., Yong, G., Biao, C.: Information coverage maximization in social networks. Comput. Res. Repository arxiv:1510.03822 (2015)
48. Zhou, D., Zhang, S., Yildirim, M.Y., Alcorn, S., Tong, H., Davulcu, H., He, J.: A local algorithm for structure-preserving graph cut. In: Proceedings of the ACM SIGKDD, pp. 655–664 (2017)