2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC) | 978-1-6654-1016-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/HIPC53243.2021.00016

A Model of Graph Transactional Coverage Patterns with Applications to Drug Discovery

A. Srinivas Reddy
IIIT Hyderabad
Hyderabad, India
srinivas.annappalli@research.iiit.ac.in

P. Krishna Reddy
IIIT Hyderabad
Hyderabad, India
pkreddy@iiit.ac.in

Anirban Mondal
Ashoka University
Haryana, India
anirban.mondal@ashoka.edu.in

U. Deva Priyakumar
IIIT Hyderabad
Hyderabad, India
deva@iiit.ac.in

***Abstract*—Facilitating the discovery of drugs by combining diverse compounds is becoming prevalent, especially for treating complex diseases like cancers and HIV. A drug is a chemical compound structure and any sub-structure of a chemical compound is designated as a *fragment*. A chemical compound or a fragment can be modeled as a graph structure. Given a set of chemical compounds and their corresponding large set of fragments modeled as graph structures, we address the problem of identifying potential combinations of diverse chemical compounds, which cover a certain percentage of the set of fragments. In this regard, the key contributions of this work are three-fold: First, we introduce the notion of *Graph Transactional Coverage Patterns (GTCPs)* for any given graph transactional dataset. Second, we propose an efficient model and framework for extracting *GTCPs* from a given graph transactional dataset. Third, we conduct an extensive performance study using three real datasets to demonstrate that it is indeed feasible to *efficiently* extract *GTCPs* using our proposed *GTCP*-extraction framework. We also demonstrate the effectiveness of the *GTCP*-extraction framework through a case study in computer-aided drug design.**

***Index Terms*—Graph mining, Graph transactions, Coverage patterns, Drug discovery**

I. Introduction

Facilitating the discovery of drugs by combining diverse compounds is becoming prevalent, especially for treating complex diseases like cancers and HIV [1]. Researchers have identified *drug cocktails* (i.e., a drug with multiple compounds) that show high efficacy, less drug resistance and less side-effects as compared to that of a single drug [2]–[4]. Generally, identification of the right combination of drugs requires conducting a huge number of experiments as the number of available drugs increases. This problem is exacerbated when the experiments are expensive and the material for testing is rare. Therefore, it becomes imperative to use computational methods for efficient drug combination discovery. In this paper, we address the issue of facilitating the generation of drug cocktails by employing data mining methods.

A drug is a chemical compound or molecule consisting of two or more chemically bonded chemical elements. A sub-compound of a chemical compound is designated as its *fragment*. Fragments are important sub-structures of a chemical compound that define the nature of the chemical compound. Incidentally, fragment-based drug design is being widely explored in the literature [5]–[9]. In these works, molecular fragments have been considered as features in structure-based drug discovery process. Moreover, research efforts [2], [3], [10], [11] have been made for finding optimal drug combinations for drug cocktail discovery. The work in [1] describes that in the early stages of drug discovery, it is essential not only to identify compounds with high binding affinity, but also to maximize/diversify the types of possible fragments of these drug compounds with the target protein. In addition, the work in [12] highlights the importance of diversity in drug combination. However, due to the combinatorial explosion of possible drug combinations, the computational cost of these methods is prohibitively expensive. In this paper, we investigate an improved method for identifying potential drug combinations by employing the notions of coverage and overlap. In addition, we propose a pattern mining-based approach to extract the potential drug combinations.

Given a set of chemical/drug compounds and an identified set of fragments (features), a pharmacologist may be interested in drug combinations that contain a pre-specified percentage of fragment set. This paper introduces the problem of mining potential chemical compounds, which cover a user-specified percentage of fragment set. For this, we model each chemical compound as a graph structure with chemical elements as vertices and bonds as edges. Given a set of chemical compounds, the set of all such graph structures forms the Graph Transactional Dataset (GTD). A sub-compound of a chemical compound is designated as its *fragment* and can also be modeled as a subgraph structure. A set of all such fragments belonging to a GTD forms the fragment set S_f. In essence, we first model chemical compounds and sub-compounds as graph transactional data and fragment set. Then the graph transactional combinations, which cover a user-specified percentage of the fragment set, will form the potential drug combinations.

In this paper, we address the problem of extracting all graph transactional patterns that cover a user-specified percentage of the fragment set. These graph transactional patterns are designated as *Graph Transactional Coverage Patterns* (*GTCPs*). To extract the fragment set from a given GTD, we employ an existing subgraph mining algorithm [13]. Given a set of GTD and fragment set S_f, a naïve approach would be to first extract all possible combinations of graph transactional patterns and check whether each pattern satisfies the user-specified coverage of fragment set. Intuitively, such an approach would be

2640-0316/21/$31.00 ©2021 IEEE
DOI 10.1109/HiPC53243.2021.00016

prohibitively expensive because the number of possible graph transactional patterns of a given GTD typically explodes.

To address the issues of combinatorial explosion and diversity, we exploit the notion of overlap. In particular, we develop an approach, which extracts all potential *GTCPs* subject to fragment set coverage and overlap constraints. In the pattern mining area, the works in [14], [15] used the *overlap ratio* constraint to address the issue of combinatorial explosion and extracted coverage patterns from a transactional dataset. Furthermore, the *overlap ratio* constraint guarantees the extraction of diverse drug patterns subject to the user-specified *overlap ratio threshold* constraint. Incidentally, *overlap ratio* satisfies the *sorted closure property* [15], [16], which enables the pruning of a significant number of non-potential candidate patterns in an apriori manner and provides the opportunity to extract potential *GTCPs*.

The key contributions of this work are three-fold:

1) We introduce the notion of *Graph Transactional Coverage Patterns (GTCPs)* for a given graph transactional dataset.
2) We propose an efficient model and framework for extracting *GTCPs* from a given graph transactional dataset.
3) We conduct an extensive performance study using three real datasets to demonstrate that it is indeed feasible to *efficiently* extract *GTCPs* using our proposed *GTCP*-extraction framework. We also demonstrate the effectiveness of our proposed *GTCP*-extraction framework through a case study in computer-aided drug design.

To the best of our knowledge, this is the first work to introduce the model of *GTCPs* and propose a framework to extract *GTCPs*. The remainder of this paper is organized as follows. Section II reviews related works. Section III discusses the proposed framework of the problem. Section IV presents our proposed *GTCP*-extraction framework. Section V reports our performance evaluation. Section VI presents our case study. Finally, we conclude in Section VII.

II. Related work

The drug combination discovery method in [10] modeled the drug combination search space as a trellis-like structure without repetitions. In particular, it proposed three algorithms, which were originally developed for digital communication. However, this method needs to compute a biological score for each candidate drug combination. Moreover, the works in [1], [12] made the observation that the chances of success in drug development are high when the chemical compounds are more diverse in nature. *None of the existing works have investigated the issue of extracting potential drug combinations with graph coverage-related knowledge and pattern mining-based approach*. However, we explore graph coverage-related knowledge to mine potential drug combinations subject to a fragment set using a pattern mining-based approach.

Fragment-based drug discovery has been extensively studied in the literature. The work in [5] extracted chemical fragments and used them as molecular descriptors for the quantitative structure-activity relationship (QSAR) studies in drug design. The works in [6], [7] extracted discriminate fragments from drug compounds to classify whether a drug is toxic or non-toxic. The work in [8] proposed visGReMLIN (visual Graph Mining strategy to infer protein-Ligand INteraction patterns), which is a methodology to search for conserved protein-ligand interactions (potential fragments) in a group of related proteins-ligand complexes. The work in [9] used molecular fragments in structure-based drug discovery as feature vectors and proposed a Machine Learning framework for Enhanced MolEcular Screening (MEMES) based on Bayesian optimization for efficiently sampling molecules in the drug design process.

The notion of coverage has been applied in set theory as set cover problem [17] and hitting set problem [18]. Further, coverage has been well explored in applications like maximum covering location problem [19] and k-path covering in route planning [20]. In graph theory, coverage has been used to formulate the vertex cover problem and the clique cover problem [17], [21].

The work in [14] proposed the notion of *coverage patterns* for efficient banner advertisement placement in e-commerce websites. In particular, it proposed a new data mining technique called *Cmine* to extract coverage patterns from transactional datasets. It has been extended the work in [15], which proposed *Coverage Pattern Projection Growth (CPPG)* and *Enhanced Coverage Pattern Projection Growth (ECPPG)* algorithms for efficiently mining coverage patterns. *Notably, none of the existing approaches have addressed the issue of extracting graph transactional coverage patterns that cover user-specified percentage of fragment set*. Our work is also differentiated from the works in [14], [15], which extract coverage patterns from a transactional dataset, while we extract *graph transactional coverage patterns* from a GTD.

TABLE I: Summary of notations.

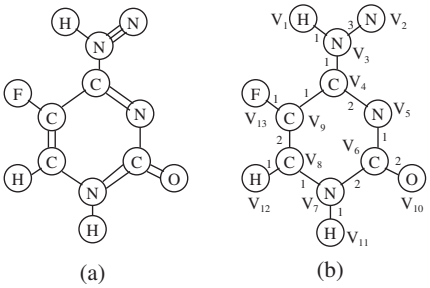
Notation	Description
G_i	Graph transaction
D	Graph transactions dataset (GTD)
S_f	Set of fragments
FID	Fragment Identifier
P	Graph pattern
P^l	l-size graph pattern
g_i	FID-based flat transaction
D_f	FID-based flat transactional dataset
GTCP	Graph transactional coverage pattern

III. Proposed Framework of the Problem

This section presents our proposed problem framework for computing *Graph Transactional Coverage Patterns* (*GTCPs*). Now we shall discuss some relevant terminology. Table I depicts the summary of notations used in this paper.

A. Graph Transaction and Subgraph

In the literature, efforts are being made to extract the knowledge of frequently occurring sub-compounds by modeling a



(a) (b)

Fig. 1: (a) Sample chemical compound, (b) Equivalent graph model.

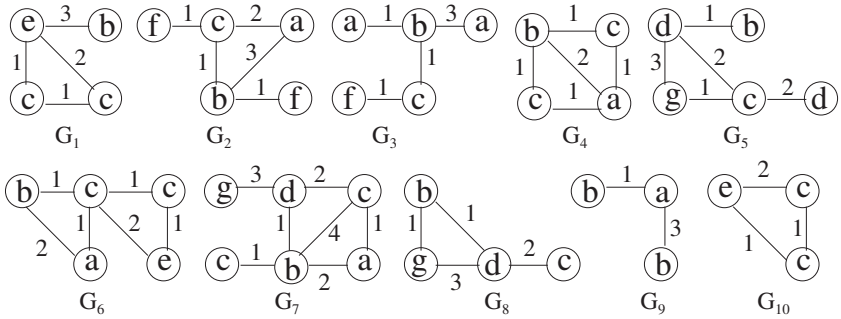
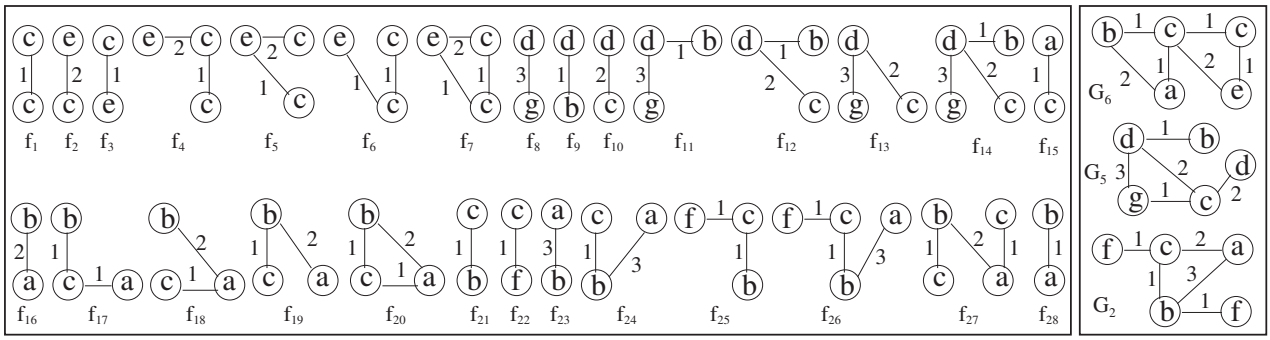


Fig. 2: Sample of 10 graph transactions.



(a) (b)

Fig. 3: (a) Set of fragments (f_1 to f_{28}) extracted from GTD D for $min\_sup = 0.2$. (b) GTCP $\{G_6,G_5,G_2\}$.

chemical compound as a graph transaction [22]. We define the notions of graph transaction and subgraph in a similar manner.

***Definition 1 (Graph Transaction)*:** A graph transaction $G = (V,E,L,l)$ is a labeled, connected and undirected graph, where V is a set of vertices, while $E \subseteq V^2$ is a set of edges. Here, 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*

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

***Definition 2 (Fragment/Subgraph)*:** A portion f of a graph transaction G is called a *fragment/subgraph* of G. Given $G = (V, E)$ and $f = (V_f, E_f)$, we say that f is a fragment/subgraph of G or f exists in G (denoted as $f \subseteq G$), iff $V_f \subseteq V$, $E_f \subseteq E$, $\forall(u, v) \in E_f \longrightarrow u, v \in V_f$ [13].

We shall henceforth use the terms *fragment* and *subgraph* interchangeably. We also use the terms *graph* and *graph transaction* interchangeably. In the literature, several algorithms have been proposed in [13], [23] to extract subgraphs from GTD subject to a *minimum support* (*min_sup*) threshold.

B. Graph Transactional Coverage (GTC)

Consider a GTD D=$\{G_1,G_2,\ldots,G_n\}$ of n graph transactions. Let S_f=$\{f_1,f_2,\ldots,f_m\}$ be the set of all fragments extracted from D. We model each graph transaction $G_i \in D$ as a subset of fragments set S_f ($G_i \subseteq S_f$). Therefore, each graph G_i in D is a set of fragments that it belongs to.

Given a set S_f, we define the notion of graph transactional coverage, which indicates the percent of fragments covered by a given graph transaction. We first define the notion of *cover*, *cover set* and then define the *graph transactional coverage*.

1) Cover: Let G_i be a graph belongs to D and f_j be a fragment belongs to S_f, we say graph G_i covers fragment f_j if fragment f_j exists in G_i. Formally, *Cover* ($Cover(G_i, f_j)$) of f_j by G_i is defined as follows:

$$Cover(G_i, f_j) = \begin{cases} 1 & \textit{if } f_j \subseteq G_i \\ 0 & \textit{otherwise} \end{cases} \quad (1)$$

Computation of $Cover(G_i, f_j)$ involves solving the subgraph isomorphism problem, which is NP-complete [13]. The *gSpan* algorithm [13] uses a canonical labeling system called *DFS lexicographical order*, which assigns *minimum DFS code* to each graph as the canonical label. We compute $Cover(G_i, f_j)$ based on DFS codes.

2) Cover Set: Let S_f be a fragment set and $G_i \in D$ be a graph transaction, the *Cover Set* ($CSet(G_i, S_f)$) of a graph G_i is defined as set of all fragments in S_f covered by graph G_i.

Formally, $CSet(G_i, S_f) = \{f_j | Cover(G_i, f_j) = 1 \ \& \ f_j \in S_f\}$. The *Graph Transactional Coverage* ($GTC(G_i, S_f)$) of G_i is defined as the percentage of fragments in S_f covered by graph $G_i \in D$. It is defined as follows:

$$GTC(G_i, S_f) = \frac{|CSet(G_i, S_f)|}{|S_f|} \quad (2)$$

Here, $0 \leq GTC(G_i,S_f) \leq 1$. A graph transaction can be a *potential graph transaction* if $GTC(G_i, S_f) \geq minGTC$, where *minGTC* is a user-defined *minimum Graph Transactional Coverage threshold*.

***Example** 2:* Consider a sample graph transactional dataset D comprising 10 graph transactions G_1 to G_{10}, as shown in Figure 2. Let f_1 to f_{28} be the set of all fragments extracted from D with $min\_sup = 0.2$, as shown in Figure 3(a). Here, fragment f_4 exists in graph G_1. Therefore, $Cover(G_1, f_4)$=1. Moreover, $CSet(G_1, S_f)$={f_1,f_2,f_3,f_4,f_5,f_6,f_7} and $GTC(G_1, S_f)$=$\frac{|CSet(G_1,S_f)|}{|S_f|}$=$\frac{7}{28}$=0.25. Similarly, $GTC(G_6, S_f)$=0.5, $GTC(G_5, S_f)$=0.25 and $GTC(G_2, S_f)$=0.21 respectively.

C. Graph Transactional Pattern Coverage

Users may be interested in a graph if it contains more number of fragments. However, a single graph may not cover a user-specified percentage of fragments. Therefore, we find a set of graphs (say Pattern (P)) in D that collectively cover the user-specified percentage of fragments. We now define the notion of *Pattern Cover Set* ($PCSet(P, S_f)$) of a graph pattern P. Let P={G_p,G_q,...,G_r} be a graph pattern, the $PCSet(P, S_f)$ of a graph pattern P is a union of all fragments in S_f covered by graphs in P. Hence, $PCSet(P, S_f)$=$\bigcup_{\forall G_i \in P} CSet(G_i, S_f)$.

Given a GTD D, fragment set S_f and let P be a graph pattern, the *Graph Transactional Pattern Coverage* ($GTPC(P, S_f)$) of a graph pattern P is the percentage of fragments in S_f covered collectively by graphs in P. We compute $GTPC(P, S_f)$ as follows:

$$GTPC(P, S_f) = \frac{|PCSet(P, S_f)|}{|S_f|} \quad (3)$$

Here, $0 \leq GTPC(P, S_f) \leq 1$. Notably, $GTPC(P, S_f) = 1$ when all of the fragments in S_f are covered by P. Conversely, $GTPC(P, S_f) = 0$ when none of the fragments are covered by graphs in P. A graph pattern P is interesting with respect to coverage point of view if $GTPC(P, S_f) \geq minGTPC$, where *minGTPC* is a user-defined *minimum Graph Transactional Pattern Coverage threshold*.

***Definition** 3 (**Graph Transactional Coverage Pattern**):* Given D and extracted set of all fragment S_f, a graph pattern P is called as *Graph Transactional Coverage Pattern* if $GTPC(P, S_f) \geq minGTPC$, $\forall G_i \in P$ and $GTC(G_i, S_f) \geq minGTC$

***Example** 3:* Consider a pattern P={G_6,G_5,G_2} as shown in Figure 3(b). The GTC values of G_6, G_5, and G_2 are 0.5, 0.25, and 0.21 respectively. The pattern cover set of P, $PCSet(P,S_f)$={f_1,f_2,f_3,f_4,f_5,f_6,f_7,f_8,f_9,f_{10},f_{11},f_{12},f_{13},f_{14}, f_{15},f_{16},f_{17},f_{18},f_{19},f_{20},f_{21},f_{22},f_{23},f_{24},f_{25},f_{26}}.
The graph transactional pattern coverage of *P*, $GTPC(P, S_f)$=$\frac{|PCSet(P,S_f)|}{|S_f|}$=$\frac{26}{28}$=0.92. Given the values of *minGTC*=0.2 and *minGTPC*=0.9, the pattern P={G_6, G_5, G_2} forms the *Graph Transactional Coverage Pattern*.

***Definition** 4 (**Minimal Graph Transactional Coverage Pattern**):* A pattern P is called as *Minimal Graph Transactional Coverage Pattern* if there **does not** exist any graph pattern P', $P' \subset P$ and $GTPC(P', S_f) \geq minGTPC$, $\forall G_i \in P'$ and $GTC(G_i, S_f) \geq minGTC$.

Problem statement: Given D and having extracted set of fragments S_f, and user-defined parameters *minGTC* and *minGTPC*, the problem is to extract all minimal *GTCPs* subject to user-defined constraints *minGTC* and *minGTPC*. *It can be noted that we henceforth represent minimal graph transactional coverage patterns simply as graph transactional coverage pattern (GTCPs).*

IV. Proposed Approach

This section presents our proposed approach.

A. Basic Idea

Our goal is to extract *GTCPs* from graph transactional dataset subject to *minGTC* and *minGTPC* constraints.

Given n graph transactions, we need to examine 2^n-1 combinations of graph transactions to determine the candidate sets that satisfy the given constraints. As n increases, the candidate sets of graph transactional patterns to be examined would essentially explode. Hence, it becomes imperative to use heuristics for efficiently pruning the search space to minimize the number of candidate graph transactional patterns. Notably, the evaluation of each graph transactional pattern is in itself non-trivial because checking the constraints associated with *minGTC* and *minGTPC* requires computationally expensive graph-based computations.

The basic idea is as follows. We extract all potential fragments (or subgraphs) from each graph transaction and convert each graph transaction into the flat transaction with the corresponding set of *F*ragment *ID*entifiers (*FIDs*). We also form the complete set of fragments by combining *FIDs* of all graph transactions. Then, we propose an efficient methodology to compute the coverage for a given candidate pattern set in terms of percentage of given fragments covered based on the overlap-based pruning technique.

The proposed *GTCP*-extraction framework consists of three steps: extraction of potential fragments from GTD, formation of flat transactions and extraction of *GTCPs*.

First, we extract the set of potential fragments (or subgraphs) from each graph transaction. A fragment is called as a potential fragment if it exists in the threshold percentage of graph transactions, which we call *minimum support* (min_sup). We can employ any of the subgraph extraction algorithm [23], such as *gSpan* algorithm [13] to extract all potential fragments. Second, we form the *FID*-based flat transactional dataset, where transaction consists of all *FIDs* of fragments corresponding to the given graph transaction.

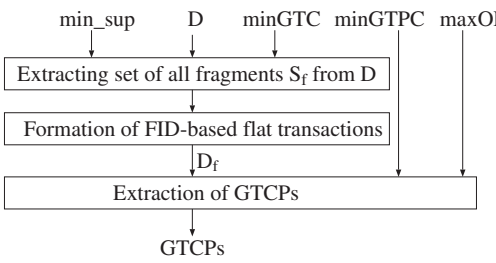


Fig. 4: *GTCP*-extraction framework.

Third, we employ pattern mining-based approach with apriori kind of properties to extract all *GTCPs*. To develop the proposed approach, we exploit the observation that the a set of drug compounds are interesting if they are more diverse in nature [12]. The notion of diversity can be captured by minimizing overlap among set of fragments covered by drug combinations. Incidentally, the overlap constraint satisfies the *sorted closure property* [15]. We exploit overlap-based heuristic for pruning the search space and propose the pattern mining-based approach to extract *GTCPs*. In the next section, we explain the details of these three steps.

B. GTCP-Extraction Framework

GTCP-extraction framework consists of three steps: (i) Extraction of potential fragments from GTD (ii) Formation of *FID*-based flat transactions (iii) Extraction of *GTCPs*. Figure 4 depicts the *GTCP*-extraction framework. Algorithm 1 depicts the steps in *GTCP*-extraction framework.

1) Extraction of Potential Fragments from GTD: The input to this step is GTD D and *minimum support* threshold (min_sup) and output is the fragment set S_f of all fragments/subgraphs that satisfy min_sup constraint. For this, we employ the *gSpan* algorithm [13] to extract all fragments with support greater than or equal to min_sup as depicted in Algorithm 1 (see line 1-2). The *gSpan* algorithm employs a pattern-growth approach and builds a hierarchical search tree called the *DFS Code Tree*.

In the *DFS Code Tree*, every node represents a subgraph, and every subgraph in GTD will have a respective 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. A depth-first search over the *DFS Code Tree* extracts all *minimum DFS codes* of all subgraphs in D. The set of fragments/subgraphs which satisfy min_sup constraint is returned as S_f. Each fragment in S_f is stored in the form of <*FID, Clabel, GIDs[ ]*>, where *FID* is a *F*ragment *ID*entifier, *Clabel* is the minimum *DFS* code assigned as a canonical label to the fragment and *GIDs[ ]* represents the set of all graph identifiers in which a fragment exists.

2) Formation of FID-based Flat Transactions: The input to this step is S_f and output is the set D_f of flat transactions, in which each flat transaction contains *FIDs*. Algorithm 2 depicts this procedure. In this step, two hash maps are maintained: S_f:<*FID, Clabel, GIDs[ ]*> and D_f:< g_i, *FIDs[ ]*>, where g_i represents i^{th} flat transactional identifier. For each fragment f_j in S_f and for each g_i in S_f.*GIDs[ ]*, we construct the *FID*-based flat transactional database D_f by inserting S_f.*FID* into D_f.*FIDs[ ]* of corresponding g_i i.e., D_f :< g_i, *FIDs[ ]*> (see lines 1-3). The set D_f :< g_i, *FIDs[ ]*> forms the *FID*-based flat transactional dataset (see line 4).

Algorithm 1 : GTCP-Extraction_Framework(*D*, *min_sup*, *minGTC*, *minGTPC*, *maxOR*)

Inputs: *D, min_sup,* $minGTC$, $minGTPC$, $maxOR$
Output: *GTCPs:* Set of all *GTCPs*.
Variables: Set of all fragments S_f:<*FID, Clabel, GIDs[ ]*>; *FID*-based flat transactional dataset D_f :< $g_i, FIDs[]$ >;

1: $S_f \leftarrow \phi$
2: $S_f \leftarrow$ Extract all fragments from D using *gSpan* algorithm that satisfy min_sup
3: D_f = Get_FID-Based_Flat_Transactions(S_f)
4: *GTCPs* = GTCPA(D_f, $minGTC$, $minGTPC$, $maxOR$)

Algorithm 2 : Get_FID-Based_Flat_Transactions(S_f)

Input: S_f
Output: D_f:< g_i, *FIDs[ ]*>

1: **for** each fragment $f_j \in S_f$ **do**
2: **for** each gid in f_j.*GIDs[ ]* **do**
3: Insert $f_j.FID$ into list *FIDs[ ]* of corresponding gid in D_f (D_f:< g_i, *FIDs[ ]*>)
4: **return** D_f

3) Extraction of GTCPs: Recall Section IV basic idea, which discuss the issue of candidate pattern explosion as n increases. To address this issue, we employ *sorted closure property* of *overlap ratio* to prune the search space.

About overlap ratio: Given a GTD D with n graph transactions, a naïve approach would need to compute the value of $GTPC$ for all 2^n-1 candidate patterns. To improve the performance, we employ the notion of *overlap ratio* as it satisfies the *sorted closure property* [15] when the graphs in the pattern are sorted in the descending order of their *GTC* values. If a pattern P fails to satisfy the user-specified maximum overlap threshold constraint, any superset of P cannot possibly satisfy the overlap threshold constraint. Hence, we can avoid generating supersets of P. We use this heuristic in our proposed *GTCPA* for efficient pruning of candidate patterns and extract all *GTCPs* from D. We now define the *overlap ratio* of a pattern and the *sorted closure property*.

***Definition 5 (Overlap Ratio of a Pattern*):** Let P=$\{g_p$,g_q,. . .,g_r,$g_s\}$ be a pattern such that $GTC(g_p, S_f) \geq GTC(g_q, S_f) \geq \cdots \geq GTC(g_r, S_f) \geq GTC(g_s, S_f)$. The *Overlap Ratio* ($OR(P, S_f)$) of a pattern P is defined as the ratio of the number of fragments in S_f common in

$PCSet(\{P-g_s\},S_f)$ and $CSet(g_s,S_f)$ to $CSet(g_s,S_f)$. It is defined as follows:

$$OR(P,S_f)=\frac{|PCSet(\{P-g_s\},S_f)\cap CSet(g_s,S_f)|}{|CSet(g_s,S_f)|}$$

For a pattern P, $0 \leq OR(P,S_f) \leq 1$. A pattern P is interesting if $OR(P,S_f) \leq maxOR$, where *maxOR* is a user-defined *maximum Overlap Ratio threshold*. Incidentally, it can be observed that the *maxOR* constraint follows the *sorted closure property*, which is explained below.

Definition 6 (Sorted Closure Property): Let the pattern P=$\{g_p$,g_q,...,g_r,$g_s\}$, $1 \leq p \leq q \leq r \leq s \leq n$ such that the *GIDs* in P are sorted in the descending order of their *GTC* values i.e., $GTC(g_p,S_f) \geq GTC(g_q,S_f) \geq \cdots \geq GTC(g_r,S_f) \geq GTC(g_s,S_f)$. If $OR(P,S_f)$ is less than or equal to *maxOR*, i.e., $OR(P,S_f) \leq maxOR$, all the non-empty subsets of P containing g_s will also have *overlap ratio* less than or equal to *maxOR*.

Definition 7 (Non-overlap pattern): A pattern P is said to be non-overlap pattern if $OR(P,S_f) \leq maxOR$ and $GTC(g_i,S_f) \geq minGTC$, $\forall g_i \in P$

Example 4: Consider a pattern P=$\{G_6$,G_5,$G_2\}$ as shown in Figure 3(b). The *overlap ratio* of P, $OR(P,S_f)$=$\frac{|PCSet(\{P-g_s\},S_f)\cap CSet(g_s,S_f)|}{|CSet(g_s,S_f)|}$ = 0.16. Given the values of *maxOR*=0.3, the pattern P=$\{G_6$,G_5,$G_2\}$ forms the non-overlap pattern.

Algorithm 3 : GTCPA(D_f, *minGTC*, *minGTPC*, *maxOR*)

Inputs: D_f: *FID*-based flat transactional dataset, *minGTC*, *minGTPC*, *maxOR*

Output: L_l: Set of graph transactional coverage patterns

Variables: NO_l (*l*-size non-overlap patterns), C_l (*l*-size candidate patterns), *P* (pattern)

1: $NO_1 \leftarrow$ Set of all potential graphs sorted in decreasing order of their *GTC* values
2: $L_1 \leftarrow$ Set of all 1-size *GTCPs*
3: *l=2*;
4: **while** NO_{l-1} not empty **do**
5: $C_l \leftarrow NO_{l-1} \bowtie NO_{l-1}$
6: **for** each pattern $P \in C_l$ **do**
7: **if** $OR(P,S_f) \leq maxOR$ **then**
8: $NO_l \leftarrow NO_l \cup P$
9: **if** $GTPC(P,S_f) \geq minGTPC$ **then**
10: $L_l \leftarrow L_l \cup P$
11: $l++$
12: **return** L_l

Graph Transactional Coverage Pattern Algorithm (GTCPA): Algorithm 3 depicts the steps in *GTCPA*. The proposed *GTPCA* extracts *GTCPs* from D_f given *minGTC*, *minGTPC* and *maxOR*. It uses apriori like approach to find the l-size patterns from (l-1)-size patterns. It uses *sorted closure property* to prune the search space and extracts all non-overlap patterns subject to *minGTPC* and *maxOR* constraints.

Let C_l be a set of l-size candidate patterns, L_l be a set of l-size *GTCPs* and NO_l be a set of l-size non-overlap candidate patterns. The *GTCPA* begins by scanning the D_f and discover all 1-size candidate pattern set C_1. The set of all patterns in C_1 are sorted in decreasing order of their GTC values and form the 1-size non-overlap pattern NO_1 and any pattern that satisfies *minGTPC* value will form the *GTCP* L_1 (see line 1-2). Using NO_1 as seed set, we generate 2-size candidate pattern C_2 by self join operation over NO_1, i.e., C_2=$NO_1 \bowtie NO_1$. From C_2, the patterns that satisfies *maxOR* constraint are considered as 2-size non-overlap patterns NO_2 and that satisfies *maxOR* and *minGTPC* are considered as 2-size *GTCPs* L_2. At each l-stage, there are two phases consisting of *Join phase* and *Prune phase* as described below.

1) **Join Phase:** To generate l-size candidate pattern set C_l, we join two $(l-1)$-size, non-overlap patterns with itself i.e, C_l=$NO_{l-1} \bowtie NO_{l-1}$ (see lines 3-5). Note that two pattern in NO_{l-1} can be joined only when their first $(l-2)$ *GIDs* are the same.
2) **Prune Phase:** We use *sorted closure property* of *overlap ratio* to reduce the size of candidate pattern set C_l. When a l-size pattern P^l does not satisfying the *maxOR* constraint ($OR(P,S_f) > maxOR$), then any superset of P^l will also not satisfy *maxOR* constraint. Therefore, the pattern P^l is removed from C_l. Any pattern that satisfies *maxOR* and *minGTPC* constraints are considered as *GTCPs* (see lines 6-10).

This process continues until no new *GTCPs* are generated or no new candidate patterns can be generated. To reduce the memory space, we have used bitmap to store $D_f < g_i$, *FIDs[ ]*>, which enables efficient computation of $GTC(g_i,S_f)$, $GTPC(P,S_f)$ and $OR(P,S_f)$ using bitwise AND and OR operations. Conversion of set based AND and OR operations to bitwise AND and OR operations significantly reduce the execution time and memory space.

C. Time and Complexity Analysis

We discuss about the time complexity analysis of the proposed framework, which consists of three parts: extraction of fragment set, formation of *FID*-based flat transactions and extraction of *GTCPs*. Let n be the number of graph transactions in D and m be the number of fragments in S_f.

First, the extraction of fragments from GTD involves subgraph isomorphism problem, which is NP-complete problem. The *gSpan* algorithm [24] assigns *minimum DFS code* to each fragment and converts isomorphism problem into string comparison problem. In such a case, the complexity is measured in terms of number of fragments and/or graph isomorphism tests. The run-time can be bounded by $O(kmn+rm)$, where k is the maximum number of subgraph isomorphism tests, i.e., string comparisons, m is the number of fragments and r is the maximum number of duplicate codes of the fragments that grow from other *minimum DFS codes*. In practical scenarios, the value of $m \leq n$, the value of r is much less than the n and the value of k is small for sparse and diverse labels. Hence, the complexity depends on the value of n.

Second, for *FID*-based flat transactions formation, each *FID* in S_f has to be inserted into the flat transaction. The search

time for *FID* in S_f is $O(1)$. Thus, the time complexity comes to $O(mn)$.

Finally, the time complexity of computing *GTCPs* is measured in a manner similar to any pattern extraction algorithm, which employs candidate pruning [25]. The complexity is captured in terms of the number of candidate patterns generated which is equal to $\sum_{l=1}^{n} l(|C_{l-1}| \cdot |C_{l-1}|)$, where $|C_{l-1}|$ is the number of candidate patterns of size l. In our case, the number of candidate patterns depends on the value of *overlap ratio*.

V. Performance Evaluation

We conducted our experiments in the ADA cluster [26] (at IIIT Hyderabad) with 10 virtual machines. Each virtual machine is allocated with 2 GB memory. We implemented our proposed schemes in Python 3.0. The link to the implementation code is provided in the footnote[1].

TABLE II: 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	1000	0.07	7	4	35.6

TABLE III: Parameters of the performance evaluation.

Parameter	Default	Variations
minGTC	0.3	0.3 - 1.0 (step size=0.1)
maxOR	0.3	0.0 - 1.0 (step size=0.1)
minGTPC	0.8	0.3 - 1.0 (step size=0.1)

We have used three real datasets, namely Yeast 167 (Yeast anti-cancer), P388 (Leukemia), from Pubchem [27] and Zinc dataset consisting of drug-like molecules [28]. The Yeast 167, P388 and Zinc datasets consist of chemical compounds, which are modeled as graph transactions. A sample graph transaction for a chemical compound is depicted in Figure 1. We have reported our case study by considering the Zinc dataset. The Zinc dataset consists of 250000 drug-like molecules docked with 4BTK protein using the AutoDock 4.2 and their binding affinity values are recorded. The top 1000 molecules with high binding affinity values are selected for out case study. Table II summarizes the three datasets.

To the best of our knowledge, there exists no other approach for extracting GTCPs from a GTD. As the number of graph transactions increases, the complexity of a naïve brute-force approach for extracting *GTCPs* increases exponentially due to large number of candidate patterns. Further, each candidate patterns require the computation of *graph transactional pattern coverage* and *overlap ratio*. Hence, in the absence of any meaningful reference approaches for comparison, we define the objective of our performance evaluation towards demonstrating the feasibility of the proposed *GTCP*-extraction framework in extracting *GTCPs* from a given graph dataset.

[1]https://github.com/srinivas2234/Graph-Transactional-Coverage-Patterns

Table III summarizes the parameters of our performance evaluation. The following performance metrics are employed.

- T_f: Represents the processing time consumed to form *FID*-based flat transactions from graph transactions that reside on the disk.
- N_f: Number of potential fragments extracted.
- AVG_{g_i}: Average number of *FIDs* in the *FID*-based flat transactions.
- T_{GTCP}: Processing time to extract *GTCPs* from flat transactions.
- N_P: Number of candidate patterns to be examined.
- N_{GTCP}: Number of *GTCPs* extracted from *FID*-based flat transactions.

A. Effect of Varying min_sup

The results in Figure 5 depicts the effect of varying *min_sup*. The results in Figure 5(a) shows that T_f increase exponentially with decrease in *min_sup*. A similar trend was also observed in N_f and AVG_{g_i} as depicted in Figure 5(b) and Figure 5(c). This occurs because, at lower values of *min_sup*, a large number of fragments possibly satisfied the *min_sup* constraints. Also, the value of T_f depends upon the number of fragments extracted from the dataset. As the value of *min_sup* increases, the value of T_S, N_f and AVG_{g_i} reduces exponentially due to the pruning effect of *min_sup*. In case of AVG_{g_i}, when the value of N_f decreases, the average number of fragments in each flat transaction also decreases.

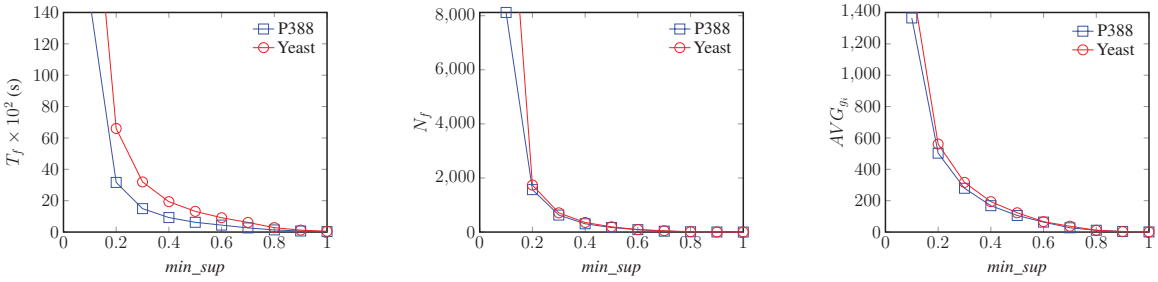
B. Effect of Varying minGTC

The results in Figure 6 depicts the effect of varying *minGTC*. The results in Figure 6(a), Figure 6(b) and Figure 6(c) depicts that T_{GTCP}, N_P and N_{GTCP} decrease with the increase in *minGTC* value. This is due to presence of large number of potential graph transactions at lower values of *minGTC* (Recall from Section III that a graph transaction with $GTC(g_i, S_f) \geq minGTC$ is called as *potential graph transaction*). At lower value of *minGTC*, a large number of candidate patterns N_P that satisfy the *maxOR* constraints are generated due to more number of potential graph transactions. As a result, more number of candidate patterns that satisfy *minGTPC* and *maxOR* (N_{GTCP}) constraint are generated.

As the value of *minGTC* increases, the number of potential graphs decreases, resulting in decrease in the number of candidate patterns N_P that satisfy *maxOR*. Further, the number of candidate patterns that satisfy *maxOR* and *minGTPC* (N_{GTCP}) also decreases. Note that we have reported our results starting from *minGTC*=0.1 as we could not experiment with *minGTC* less than 0.1 due to explosion in the number of potential graphs.

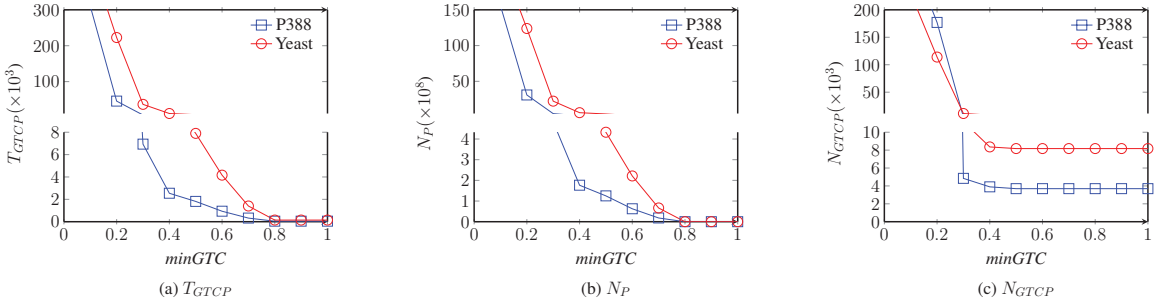
C. Effect of Varying maxOR

The results in Figure 7 shows the effect of varying *maxOR*. The results in Figures 7(a), 7(b), and 7(c) show that T_{GTCP}, N_P and N_{GTCP} increases with the increase in the value of *maxOR*. At lower values of *maxOR*, the number of candidate patterns N_P that satisfy the *maxOR* constraint are very less.



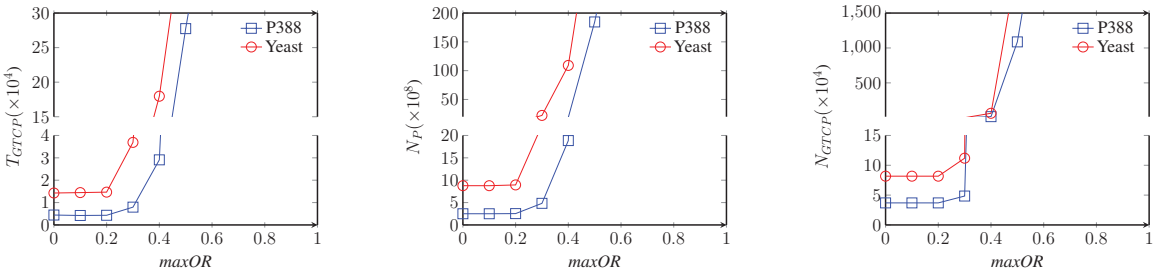
(a) T_f (b) N_f (c) AVG_{g_i}

Fig. 5: Effect of varying *min_sup*.



(a) T_{GTCP} (b) N_P (c) N_{GTCP}

Fig. 6: Effect of varying *minGTC*.



(a) T_{GTCP} (b) N_P (c) N_{GTCP}

Fig. 7: Effect of varying *maxOR*.

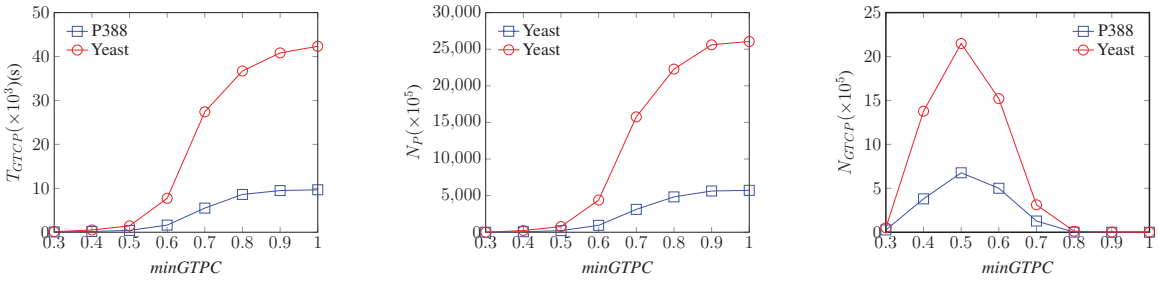
This is due to presence of common fragments in graph belongs to graph pattern and are pruned by *maxOR* constrained of overlap ratio. As a result, the value of T_{GTCP} is also low. When the value of *maxOR* increases from 0.0, the restriction imposed on patterns by *sorted closure property* of overlap ratio constraints are progressively relaxed. Therefore, the number candidate patterns N_P qualifying the *maxOR* constraints increases exponentially. Consequently, the value of T_{GTCP} increases exponentially. As a result of increase in N_P, the number of *GTCPs* N_{GTCP} increases with increase in *maxOR*. Observe that when *maxOR*=0, there are 3695 *GTCPs* in P388 dataset and 8175 *GTCPs* in Yeast dataset. At higher values of *maxOR*, more *GTCPs* can be extracted.

D. Effect of Varying minGTPC

The results in Figure 8 depicts the effect of varying *minGTPC*. The results in Figure 8(a) and Figure 8(b) indicate that T_{GTCP} and N_P increases with increase in *minGTPC* value. The reason being, when the value of *minGTPC* increases, *GTCPA* generated candidate patterns of bigger size (l) to satisfy the higher coverage requirement. Note that the number of candidate patterns is significantly lower as compared to 2^n-1 candidate patterns due to the pruning effect of *sorted closure property* of overlap ratio.

The results in Figure 8(c) show that the value of N_{GTCP} increases initially with increase in *minGTPC* due to increase in N_P. When the value of *minGTPC* increases further, the N_{GTCP} value starts decreasing due presence of more overlapping fragments in graph patterns. Consequently, most of the candidate *GTCPs* will be pruned by *maxOR* constraint.

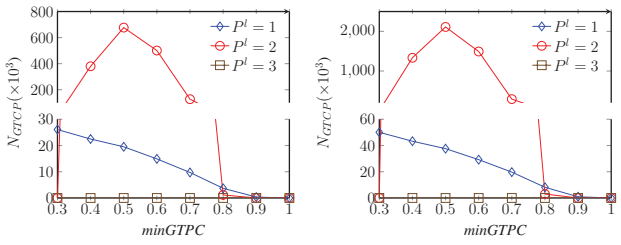
The results in Figure 9 depicts the number of *GTCPs* according to the pattern size (P^l). The results in Figure 9(a) depicts the number of patterns with P^l=1, 2, and 3 for P388



(a) T_{GTCP} (b) N_P (c) N_{GTCP}

Fig. 8: Effect of varying *minGTPC*.

datasets at different values of *minGTPC*. As expected, the results show that the number of *GTCPs* with P^l=1 decreases as *minGTPC* increases. However, it can be observed that the number of patterns with P^l=2 increases to a peak value and then decreases as the value of *minGTPC* increases. This is due to the fact that at lower values of *minGTPC*, the number of *GTCPs* increases due to less coverage threshold. As *minGTPC* increases, increased number of patters fail to meet the *minGTPC* constraint. As a result, the number of patterns reaches to a peak value and starts decreasing. For these data sets, we could not find any patterns with P^l=3 at the given *minGTPC* and *maxOR* settings. We observed similar trends in the results for the Yeast dataset, as depicted in Figure 9(b). The values differ due to difference in dataset sizes.



(a) N_{GTCP} (P388) (b) N_{GTCP} (Yeast)

Fig. 9: Effect on pattern size l of varying *minGTPC*.

E. Discussion

In the proposed approach, conversion of graph transactions into *FID*-based flat transactions is done only once at given *min_sup* threshold. Once graph transactions are transformed to flat transactions, one can vary $minGTC$, $minGTPC$ and $maxOR$ as per the application requirement.

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

VI. Case Study: Usefulness of *GTCPs* in Drug Design

In this section, we discuss the case study by considering the context of Structure-Based Drug Discovery (SBDD). SBDD is the design and optimization of a chemical structure with the goal of identifying candidate compounds/molecules suitable for clinical testing. It involves virtual screening of large libraries of molecules against the target protein to measure the inter-molecular interaction score (binding affinity). When the binding affinity between a molecule and target satisfies the threshold level, the molecule is called a lead molecule or active drug molecule. Recall that researchers have identified that drug cocktails with diverse drugs help therapies to target cancer and drug resistance more efficiently [1], [12]. In this regard, one needs to select molecules that have high binding affinity and are diverse in nature.

For this case study, we have chosen top 1000 drug molecules from Zinc dataset with highest binding affinity when docked against the 4BTK protein using the AutoDock 4.2 [28]. We have modeled them as graph transactions and extracted 6 interesting drug molecules combinations by setting *minGTC*=0.2, *minGTPC*=0.9 and *maxOR*=0.3. One of the drug molecules combination along with its potential fragments is presented in Figure 10. It depicts that selected fragments are unique to each of the two molecules and are common to both the molecules. All the 1000 top molecules have comparable binding affinity, but we need to explore all possible combinations to choose drug combination that exhibits maximum coverage and minimal overlap of fragments. The two drug molecules 436 and 270 have a binding affinity value of -8.5 kcal/mol and -8.7 kcal/mol respectively, indicating that both of them bind to the target protein strongly. While these two molecules exhibit 92% coverage of potential fragments, the overlap is only 14%. This indicates that though we started with 1000 drugs and over a million drug combinations, we are able to select a drug combination that exhibits high binding affinity and more diversity, thereby supporting our hypothesis.

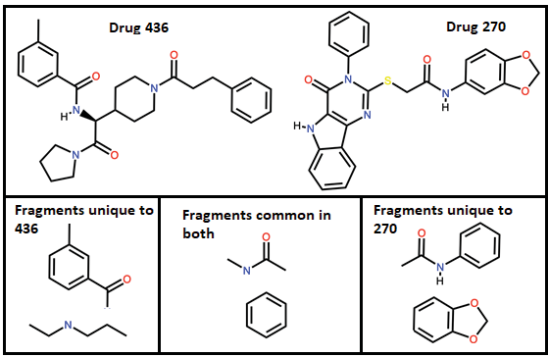


Fig. 10: Potential drug combination with drug molecules 436 and 270 along with its potential fragments.

VII. Conclusion

Mining of graph transactional coverage patterns (*GTCPs*) has significant applications in the domains of chemical and biological networks e.g., during the drug design and discovery process. We have considered the problem of extracting the coverage-related graph patterns subject to fragment set with a data mining-based approach. Given a set of chemical compounds modeled as graph transactions, we have introduced the concept of *GTCPs*, which represent potential drug combinations. We have also proposed a *GTCP*-extraction framework for efficiently extracting *GTCPs* from a given graph transactional dataset subject to *minGTC*, *minGTPC* and *maxOR* 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 *GTCPs* through a case study in drug design. To our knowledge, this is the first work to introduce the notion of *GTCPs* and consider the extraction of *GTCPs* from graph transactional data. Given the prevalence of graph data modeling, our proposed model of *GTCPs* has a potentially huge scope for opening up new avenues of research in the realm of extraction of interesting knowledge from graph transactional datasets in several important and diverse domains.

Acknowledgement 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.

References

[1] M. Snowden and D. V. Green, "The impact of diversity-based, high-throughput screening on drug discovery: chance favours the prepared mind," *Current opinion in drug discovery & development*, vol. 11, no. 4, pp. 553–558, 2008.

[2] A. Zimmer, A. Tendler, I. Katzir, A. Mayo, and U. Alon, "Prediction of drug cocktail effects when the number of measurements is limited," *PLOS Biology*, vol. 15, no. 10, pp. 1–16, 2017.

[3] H. Liu, W. Zhang, L. Nie, X. Ding, J. Luo, and L. Zou, "Predicting effective drug combinations using gradient tree boosting based on features extracted from drug-protein heterogeneous network," *BMC Bioinformatics*, vol. 20, no. 1, pp. 1–12, 2019.

[4] J. Tanwar, S. Das, Z. Fatima, and S. Hameed, "Multidrug resistance: an emerging crisis," *Interdisciplinary perspectives on infectious diseases*, vol. 2014, p. 541340, 2014.

[5] R. Khashan, W. Zheng, and A. Tropsha, "The development of novel chemical fragment-based descriptors using frequent common subgraph mining approach and their application in QSAR modeling," *Molecular Informatics*, vol. 33, no. 3, pp. 201–215, 2014.

[6] C. Borgelt and M. R. Berthold, "Mining molecular fragments: finding relevant substructures of molecules," in *Proc. ICDM*, pp. 51–58, 2002.

[7] C. Borgelt, M. R. Berthold, and D. E. Patterson, "Molecular fragment mining for drug discovery," in *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pp. 1002–1013, 2005.

[8] V. S. Ribeiro, C. A. Santana, A. V. Fassio, F. R. Cerqueira, C. H. da Silveira, *et al.*, "visGReMLIN: Graph mining-based detection and visualization of conserved motifs at 3D protein-ligand interface at the atomic level," vol. 21, no. 2, pp. 1–12, 2020.

[9] S. Mehta, S. Laghuvarapu, Y. Pathak, A. Sethi, M. Alvala, and U. D. Priyakumar, "Memes: Machine learning framework for enhanced molecular screening," *Chemical Science*, vol. 12, pp. 11710–11721, 2021.

[10] D. Calzolari, S. Bruschi, L. Coquin, J. Schofield, J. D. Feala, J. C. Reed, A. D. McCulloch, and G. Paternostro, "Search algorithms as a framework for the optimization of drug combinations," *PLoS Comput Biol*, vol. 4, no. 12, p. e1000249, 2008.

[11] D. Paul, G. Sanap, S. Shenoy, D. Kalyane, K. Kalia, and R. K. Tekade, "Artificial intelligence in drug discovery and development," *Drug Discovery Today*, vol. 26, no. 1, pp. 80–93, 2020.

[12] T. Meinl, C. Ostermann, and M. R. Berthold, "Maximum-score diversity selection for early drug discovery," *Journal of Chemical Information and Modeling*, vol. 51, no. 2, pp. 237–247, 2011.

[13] Xifeng Yan and Jiawei Han, "gSpan: Graph-based substructure pattern mining," in *Proc. ICDM*, pp. 721–724, 2002.

[14] P. G. Srinivas, P. K. Reddy, S. Bhargav, R. U. Kiran, and D. S. Kumar, "Discovering coverage patterns for banner advertisement placement," in *Proc. PAKDD*, pp. 133–144, 2012.

[15] P. Gowtham Srinivas, P. Krishna Reddy, A. V. Trinath, S. Bhargav, and R. Uday Kiran, "Mining coverage patterns from transactional databases," *J. Intell. Inf. Syst.*, vol. 45, pp. 423–439, 2015.

[16] B. Liu, W. Hsu, and Y. Ma, "Mining association rules with multiple minimum supports," in *Proc. of SIGKDD*, pp. 337–341, ACM, 1999.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[18] A. Gainer-Dewar and P. Vera-Licona, "The minimal hitting set generation problem: algorithms and computation," *SIAM Journal on Discrete Mathematics*, vol. 31, no. 1, pp. 63–100, 2017.

[19] O. Berman, J. Kalcsics, and D. Krass, "On covering location problems on networks with edge demand," *Comput. Oper. Res.*, vol. 74, pp. 214–227, Oct. 2016.

[20] S. Funke, A. Nusser, and S. Storandt, "On k-path covers and their applications," *Proc. VLDB Endow.*, vol. 7, pp. 893–902, June 2014.

[21] J. Wu, C. M. Li, L. Jiang, J. Zhou, and M. Yin, "Local search for diversified top-*k* clique search problem," *Computers & Operations Research*, vol. 116, p. 104867, 2020.

[22] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," in *Proc. IEEE ICDM*, pp. 313–320, 2001.

[23] M. Aida, M. Pieter, B. Wout, M. Pieter, C. Boris, and K. L. Bart, Goethals, "Grasping frequent subgraph mining for bioinformatics applications," *BioData mining*, vol. 11, no. 1, pp. 1–20, 2018.

[24] "UIUC technical report, UIUCDCS-R-2002-2296." https://sites.cs.ucsb.edu/~xyan/papers/gSpan.pdf, December Accessed in September 2021.

[25] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, *Introduction to Data Mining*. Pearson, 2^{nd} ed., 2018.

[26] "ADA." http://hpc.iiit.ac.in/wiki/index.php/Ada_User_Guide, October Accessed in September 2021.

[27] X. Yan, H. Cheng, J. Han, and P. S. Yu, "Mining significant graph patterns by leap search," in *Proc. ACM SIGMOD*, pp. 433–444, 2008.

[28] T. Sterling and J. J. Irwin, "ZINC 15 – Ligand discovery for everyone," *Journal of Chemical Information and Modeling*, vol. 55, no. 11, pp. 2324–2337, 2015.