ORIGAMI: Mining Representative Orthogonal Graph Patterns

Mohammad Al Hasan[1], Vineet Chaoji[1], Saeed Salem[1], Jeremy Besson[2], and Mohammed J. Zaki[1]
[1]Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 12180
[2]INSA de Lyon, LISI/LIRIS Batiment Blaise Pascal F-69621 Villeurbanne Cedex, France
[1]{alhasan, chaojv, salems, zaki}@cs.rpi.edu, [2]jeremy.besson@insa-lyon.fr

Abstract

In this paper, we introduce the concept of α-orthogonal patterns to mine a representative set of graph patterns. Intuitively, two graph patterns are α-orthogonal if their similarity is bounded above by α. Each α-orthogonal pattern is also a representative for those patterns that are at least β similar to it. Given user defined $\alpha, \beta \in [0, 1]$, the goal is to mine an α-orthogonal, β-representative set that minimizes the set of unrepresented patterns.

We present ORIGAMI*, an effective algorithm for mining the set of representative orthogonal patterns.* ORIGAMI *first uses a randomized algorithm to randomly traverse the pattern space, seeking previously unexplored regions, to return a set of maximal patterns.* ORIGAMI *then extracts an α-orthogonal, β-representative set from the mined maximal patterns. We show the effectiveness of our algorithm on a number of real and synthetic datasets. In particular, we show that our method is able to extract high quality patterns even in cases where existing enumerative graph mining methods fail to do so.*

1 Introduction

Increasingly, today's massive data is in the form of complex graphs or networks. Examples include the physical Internet, the world wide web, social networks (including blogs, chat rooms, phone networks, and networking web-sites), biological networks (including protein interactions networks and bio-chemical compounds). Mining such databases for graph patterns has attracted a lot of interest in recent years.

Typical graph mining methods follow the combinatorial pattern enumeration paradigm, and aim to extract *all* frequent subgraphs, perhaps subject to some constraints. In many real-world applications arising in bioinformatics and social network analysis, the complete enumeration of all patterns is practically infeasible, due to the combinatorial explosion in the number of mined subgraph patterns. For example, on a set of six proteins taken from the HOMSTRAD database of homologous protein structures (see dataset PS in Section 5), a typical enumerative graph mining method (we used gSpan [14]) did not finish running in even 2 days. These six graphs contain common motifs of size over 50-60 residues, thus any method that tries to enumerate all subgraphs is simply unable to mine this dataset. In fact, mining only the closed or even the maximal patterns in such domains can be untenable.

Aborting the mining process prematurely does not help either, as there is no guarantee that the resulting set of patterns is representative in any sense. Typically, one can expect that the patterns cover only a small region of the output search space (e.g., a breadth-first search approach will have seen patterns only up to some level, and a depth-first method may have seen patterns covering branches up to some point). For example, we ran a depth-first graph mining algorithm [5] on a protein-interaction dataset consisting of three graphs (see dataset PI in Section 5), each graph having 2154 nodes, and on average 81607 edges, with total database size 3MB. The mining process was aborted after a day of running, at which point it had generated a 7GB output file containing over 8 million subgraphs. The largest mined graph had only 22 edges; there were 57 such subgraphs, but these had a similarity of over 95% (differing in only a few edges), indicating that only a small fraction of the possible output space had been seen.

Note also that in many real-world cases, enumerating all frequent patterns is not necessarily the primary objective. Rather, mined patterns are likely to be used as inputs for a subsequent analysis/modeling step, and as such, a relatively small representative set of patterns may suffice. For example, mining frequent motifs in protein structures sets the stage to solve problems like structural alignment, homology detection, etc. Recurring patterns in a social network can be used for link prediction, de-duplication, hidden group identification, etc. Frequent patterns obtained from network log data can be used to build classification model that can predict network intrusion and other anomalous behavior. None of these applications requires the entire set of frequent patterns. Note also, that the lack of interpretability and the curse of dimensionality due to a large set of redundant patterns can cause problems for subsequent steps like clustering and classification. Many successful applications of pattern mining for solving real-life problems thus require

the result-set to be a *summary*, rather than a *complete set* of the frequent pattern space.

In this paper, our goal is to address all of the above limitations that prevent graph mining to be applied in real-world problems. Instead of enumerating all graph patterns, we aim to mine a relatively small set of representative patterns that share little similarity with each other. More specifically, given user-defined parameters $\alpha, \beta \in [0, 1]$, our goal is to find an optimal α-orthogonal β-representative set of patterns. Two patterns are said to be α-orthogonal if their similarity is at most α, and a pattern is said to be a β-representative for another pattern if their similarity is at least β. Instead of enumerating the entire set of subgraph patterns, we employ a randomized (but principled) search over the partial order of subgraph patterns, to obtain a representative sample of the possible output space (of maximal patterns). The aim is to cover, or traverse, different unexplored parts of the partial order yielding potentially representative patterns. In a second step, a locally optimal orthogonal representative pattern set is extracted from the output sample. The main contributions of our paper are as follows:

- We propose a new paradigm for mining a summary representation of the set of frequent graphs. Unlike previous techniques, that focus on the distance in the transaction space to obtain representatives, our approach captures representatives by considering the distances in the pattern space.
- We introduce a randomized approach for mining maximal subgraph patterns. The method is designed to cover the partial order of subgraphs, so that orthogonal maximal patterns are obtained quickly.
- We formulate the α-orthogonal β-representative set finding as an optimization problem. We show that the optimization problem is NP-Hard and we thus propose a local optimization solution that is efficient and practically feasible.

Our algorithm that finds the α-orthogonal β-representative set is called ORIGAMI (which stands for **O**rthogonal **R**epresentat**I**ve **G**r**A**ph **MI**ning). We demonstrate the effectiveness of ORIGAMI on a variety of synthetic and real dataset, and show that it is able to mine good quality orthogonal representative sets, especially for datasets where traditional enumerative methods fail completely.

2 Related Work

Many recent methods have been proposed for graph mining; these include [4, 8, 9, 14, 6, 10]. The focus of these methods is to mine all frequent subgraph patterns, rather than finding orthogonal or representative patterns. There is also an increasing interest in using the mined graph patterns for indexing [16].

There are several works guided towards finding a subset of frequent patterns that are most informative, compressed, discriminative and non-redundant [1, 13, 12, 3]. However, all these previous works handle itemset patterns only. In the graph domain, we did not find any work on compressed frequent patterns, except, works on closed frequent graphs [15] and maximal frequent graphs [11, 7]. Even though these two approaches generate a smaller set of patterns, the number of patterns in both cases can still be very large. Moreover, many patterns in the resulting sets can be very similar, hence, they may not be appropriate as a summary or representative pattern set.

We present a set of frequent graphs that are representative of the entire frequent graph partial order. Each element in representative set is more than α distant from the others. Moreover, since graphs represent the most general type of patterns, a solution to this problem in the graph setting automatically covers the other pattern types like itemsets, sequences and trees.

3 Problem Formulation

Graphs and Subgraphs: A graph $G = (V, E)$, consists of a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$, and a set of edges $E = \{(v_i, v_j) : v_i, v_j \in V\}$. Let L_V and L_E be the set of vertex and edge labels, respectively, and let $\mathcal{V} : V \rightarrow L_V$ and $\mathcal{E} : E \rightarrow L_E$ be the labeling functions that assign labels to each vertex and edge. The *size* of a graph G, denoted $|G|$ is the cardinality of the edge set (i.e., $|G| = |E|$). A graph of size k is also called a k-graph. A graph is *connected* if each vertex in the graph can be reached from any other vertex. All graphs we consider are undirected, connected and labeled.

A graph $G_1 = (V_1, E_1)$ is a *subgraph* of another graph $G_2 = (V_2, E_2)$, denoted $G_1 \subseteq G_2$, if there exists a 1-1 mapping $f : V_1 \rightarrow V_2$, such that $(v_i, v_j) \in E_1$ implies $(f(v_i), f(v_j)) \in E_2$. Further, f preserves vertex labels, i.e., $\mathcal{V}(v) = \mathcal{V}(f(v))$), and preserves edge labels, i.e., $\mathcal{E}(v_1, v_2) = \mathcal{E}(f(v_1), f(v_2))$. f is also called a *subgraph isomorphism* from G_1 to G_2. If $G_1 \subseteq G_2$, we also say that G_2 is a super-graph of G_1. Note also that two graphs G_1 and G_2 are *isomorphic iff* $G_1 \subseteq G_2$ and $G_2 \subseteq G_1$. Let $\mathcal{D}$ be a set of graphs, then we write $G \subseteq \mathcal{D}$ if $\forall D_i \in \mathcal{D}, G \subseteq D_i$. G is said to be a *maximal common subgraph* of $\mathcal{D}$ iff $G \subseteq \mathcal{D}$, and $\nexists H \supseteq G$, such that $H \subseteq \mathcal{D}$.

Graph Support: Let $\mathcal{D}$ be a database (a set) of graphs, and let each graph $D_i \in \mathcal{D}$ have a unique graph identifier. Denote by $\mathbf{t}(G) = \{i : G \subseteq D_i \in \mathcal{D}\}$, the *graph identifier set (gidset)*, which consists of all graphs in $\mathcal{D}$ that contain a subgraph isomorphic to G. The *support* of a graph G in $\mathcal{D}$ is then given as $\pi(G, \mathcal{D}) = |\mathbf{t}(G)|$, and G is called *frequent* if $\pi(G, \mathcal{D}) \geq \pi^{\min}$, where $\pi^{\min}$ is a user-specified minimum support (*minsup*) threshold. A frequent graph is *closed* if it has no frequent super-graph with the same support. A frequent graph is *maximal* if it has no frequent super-graph.

Denote by $\mathcal{F}, \mathcal{C}, \mathcal{M}$ the set of all frequent, all closed frequent, and all maximal frequent subgraphs, respectively. By definition, $\mathcal{F} \supseteq \mathcal{C} \supseteq \mathcal{M}$. The set of all maximal frequent subgraphs $\mathcal{M}$ is also known as the *positive border*. Note that the set of all (frequent) subgraphs forms a partial order with respect to the subgraph relationship, and associated with each graph in the partial order is its gidset.

Orthogonal and Representative Graphs: Define $sim : \mathcal{F} \times \mathcal{F} \rightarrow [0, 1]$ to be a symmetric binary function that returns the *similarity* between two graphs. For example, the similarity based on the maximum common subgraph [2] is given as: $sim(G_a, G_b) = \frac{|G_c|}{\max(|G_a|, |G_b|)}$, where G_c is the maximum common subgraph of G_a and G_b.

Given any collection of graphs $\mathcal{G}$, and given a similarity threshold $\alpha \in [0, 1]$, we say that a subset of graphs $\mathcal{R} \subseteq \mathcal{G}$ is α-**orthogonal** [1] with respect to $\mathcal{G}$ iff for any $G_a, G_b \in \mathcal{R}$, $sim(G_a, G_b) \leq \alpha$ and for any $G_i \in \mathcal{G} \setminus \mathcal{R}$ there exists a $G_j \in \mathcal{R}$, $sim(G_i, G_j) > \alpha$.

Given a collection of graphs $\mathcal{G}$, an α-orthogonal set $\mathcal{R} \subseteq \mathcal{G}$, and given a similarity threshold $\beta \in [0, 1]$, we say that $\mathcal{R}$ **represents** a graph $G \in \mathcal{G}$, provided there exists some $G_a \in \mathcal{R}$, such that $sim(G_a, G) \geq \beta$. Let $\Upsilon(\mathcal{R}, \mathcal{G}) = \{G \in \mathcal{G} : \exists G_a \in \mathcal{R}, sim(G, G_a) \geq \beta\}$, then we say that $\mathcal{R}$ is a β-**representative** set for $\Upsilon(\mathcal{R}, \mathcal{G})$.

Finally, given $\mathcal{G}$, and its α-orthogonal, β-representative set $\mathcal{R}$, define the **residue set** of $\mathcal{R}$ to be the set of unrepresented patterns in $\mathcal{G}$, given as $\Delta(\mathcal{R}, \mathcal{G}) = \mathcal{G} \setminus \{\mathcal{R} \cup \Upsilon(\mathcal{R}, \mathcal{G})\}$. The *residue* of $\mathcal{R}$ is defined to be the cardinality of its residue set, $|\Delta(\mathcal{R}, \mathcal{G})|$. Define the *average residue similarity* as follows: $ars(\mathcal{R}, \mathcal{G}) = \frac{\sum_{G_b \in \Delta(\mathcal{R}, \mathcal{G})} \max_{G_a \in \mathcal{R}} \{sim(G_a, G_b)\}}{|\Delta(\mathcal{R}, \mathcal{G})|}$.

Lemma 1 $\alpha < ars(\mathcal{R}, \mathcal{G}) < \beta$.

PROOF: For any $G_a \in \Delta(\mathcal{R}, \mathcal{G})$, we have $sim(G_a, G_b) < \beta$ for all $G_b \in \mathcal{R}$. Furthermore, by definition, for any $G_b \in \mathcal{G} \setminus \mathcal{R}, \exists G_a \in \mathcal{R}$, such that $sim(G_a, G_b) > \alpha$. Thus the numerator is always in the range (α, β). ■

Problem Definition: In this paper we are interested in finding the α-orthogonal, β-representative set for the set of all maximal frequent subgraphs, i.e., when $\mathcal{G} = \mathcal{M}$. In general, one can find orthogonal representative sets for any collection of patterns $\mathcal{G}$. Since the maximal patterns provide a synopsis of the frequent patterns, and since they are generally a lot fewer than the sets of all frequent and closed frequent patterns, it seems reasonable to try to find a orthogonal representative set among those. However, since even mining all the maximal graphs can be infeasible in many real-world domains, we try to find orthogonal representative sets for a subset of the maximal patterns $\widehat{\mathcal{M}} \subseteq \mathcal{M}$.

Given a graph database $\mathcal{D}$, user-defined similarity thresholds $\alpha, \beta \in [0, 1]$, and a minimum support threshold $\pi^{\min}$, the problem of mining α-orthogonal β-representative graph patterns can now be formulated as follows:

1. Mine a (diverse) sample of maximal frequent patterns $\widehat{\mathcal{M}} \subseteq \mathcal{M}$.
2. Mine an α-orthogonal β-representative set $\mathcal{R}$, that minimizes the residue $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})|$.

Note that an alternative objective can be to maximize $ars(\mathcal{R}, \widehat{\mathcal{M}})$. In this paper we focus on minimizing the residue ($|\Delta(\mathcal{R}, \widehat{\mathcal{M}})|$).

A solution to the above problem provides a small set of maximal frequent graph patterns that are non-redundant or orthogonal (for the α constraint) and also representative (for the β constraint). Depending on the value of β, the following two cases make interesting variants of the problem:

case I ($\beta \leq \alpha$): By definition of α-orthogonal set, for any $G_i \in \widehat{\mathcal{M}} \setminus \mathcal{R}$, there exists $G_j \in \mathcal{R}$, such that $sim(G_i, G_j) > \alpha \geq \beta$. This implies that each $G_i \in \widehat{\mathcal{M}} \setminus \mathcal{R}$ is represented by some $G_j \in \mathcal{R}$. Immediately have $\Upsilon(\mathcal{R}, \widehat{\mathcal{M}}) = \widehat{\mathcal{M}} \setminus \mathcal{R}$, which in turn implies that $\Delta(\mathcal{R}, \widehat{\mathcal{M}}) = \emptyset$. Thus, when $\beta \leq \alpha$ the residue of any α-orthogonal set $\mathcal{R}$ is 0, implying that every α-orthogonal set is optimal w.r.t the residue.

case II ($\beta > \alpha$): This is the general case, for which the α-orthogonal set $\mathcal{R}$, may not be a β-representative for some maximal frequent graphs in $\widehat{\mathcal{M}}$. In other words when $\beta > \alpha$, the residue $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| \geq 0$; thus an optimal solution is a set of orthogonal patterns that minimizes the residue. A special case of $\beta > \alpha$ occurs when $\beta = 1$. In this case each element in the α-orthogonal represents only itself, and the residue is $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| = |\widehat{\mathcal{M}} \setminus \mathcal{R}|$.

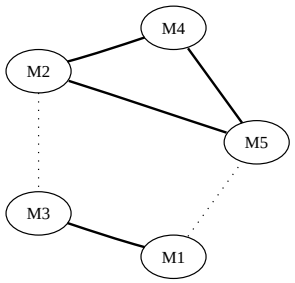
As an example, assume that we are given the pair-wise similarities between a set of graphs $\widehat{\mathcal{M}}$, as shown in Figure 1. If $\alpha = 0.2$, then there are two possible α-orthogonal sets, namely $\mathcal{R}_1 = \{M_1, M_3\}$ and $\mathcal{R}_2 = \{M_2, M_4, M_5\}$ as illustrated in Figure 1(b). If $\beta \leq \alpha$, both of these will be optimal in terms of the residue. However, if $\beta = 0.6$, then $\Upsilon(\mathcal{R}_1, \widehat{\mathcal{M}}) = \{M_2, M_5\}$, which gives $|\Delta(\mathcal{R}_1, \widehat{\mathcal{M}})| = |\{M_4\}| = 1$. This is illustrated in Figure 1(b), which shows that M_4 remains unrepresented by $\mathcal{R}_1$. For $\mathcal{R}_2$, $\Upsilon(\mathcal{R}_2, \widehat{\mathcal{M}}) = \{M_1, M_3\}$, yielding $|\Delta(\mathcal{R}_2, \widehat{\mathcal{M}})| = |\emptyset| = 0$. Thus in this case $\mathcal{R}_2$ is the optimal α-orthogonal β-representative set.

The intuition behind our definition of α-orthogonal β-representative set should now be clear. The orthogonality constraint ensures that the resulting set of frequent patterns has controlled redundancy. For a given α, several sets of (maximal) patterns qualify as feasible α-orthogonal sets. Besides redundancy control, we also want to achieve representativeness, i.e., for every maximal frequent patterns not

[1] This is inspired by linear algebra, where two vectors are said to be orthogonal if their similarity (dot product) is 0. We extend this notion to say that two graphs are α-orthogonal if their similarity is at most α. When $\alpha = 0$, it gives the usual sense of orthogonality.

	M_1	M_2	M_3	M_4	M_5
M_1	1.0	0.3	0.18	0.4	0.7
M_2	-	1.0	0.7	0	0.1
M_3	-	-	1.0	0.4	0.5
M_4	-	-	-	1.0	0.15
M_5	-	-	-	-	1.0

(a) Similarity Matrix



(b) Similarity Graph

Figure 1. Similarity Matrix & Graph: In the graph, $sim \leq \alpha = 0.2$ is denoted by bold edges, and $sim \geq \beta = 0.6$ by dotted edges.

reported, we want it to have a representative similar to it (based on the β threshold). Some patterns may still remain unrepresented, which make up the residue set. For a given α and β, the size of the residue set becomes an objective function to minimize when choosing the orthogonal representative sets.

4 The ORIGAMI Approach

ORIGAMI($\mathcal{D}, \pi^{\min}, \alpha, \beta$):
1. EM = Edge-Map ($\mathcal{D}$)
2. $\mathcal{F}_1$ = Find-Frequent-Edges ($\mathcal{D}, \pi^{\min}$)
3. $\widehat{\mathcal{M}} = \emptyset$
4. **while** stopping_condition() $\neq$ **true**
5. M = Random-Maximal-Graph($\mathcal{D}, \mathcal{F}_1, EM, \pi^{\min}$)
6. $\widehat{\mathcal{M}} = \widehat{\mathcal{M}} \cup M$
7. $\mathcal{R}$ = Orthogonal-Representative-Sets ($\widehat{\mathcal{M}}, \alpha, \beta$)

Figure 2. ORIGAMI Algorithm

ORIGAMI has two distinct steps to mine the orthogonal representative patterns. The first step finds a subset of frequent maximal patterns $\widehat{\mathcal{M}}$. The second step refines $\widehat{\mathcal{M}}$ to obtain an orthogonal representative set. The pseudo-code for ORIGAMI is shown in Figure 2. The algorithm accepts a graph database $\mathcal{D}$, a minimum support value $\pi^{\min}$, and values for the parameters α and β. ORIGAMI first computes two global data structure that are used to generate maximal frequent patterns (lines 1-2). The edge-map (EM) stores for each vertex label l_{v_a} a pair (l_{v_b}, l_e), if (v_a, v_b) is an edge with edge label l_e in some graph in $\mathcal{D}$. $\mathcal{F}_1$ stores the set of all frequent 1-graphs (i.e., single edges). ORIGAMI then computes an approximation or sample of the set of maximal patterns $\widehat{\mathcal{M}}$, by generating random maximal graphs until the stopping condition is met (lines 4-6). The stopping condition mainly ensures that the partial order of frequent graph patterns has been sufficiently explored. Once $\widehat{\mathcal{M}}$ is obtained, ORIGAMI computes one or several α-orthogonal β-representative sets (line 7). Details of the various steps appear below.

4.1 Mining Random Maximal Graphs

The first step in ORIGAMI finds a sample $\widehat{\mathcal{M}}$ of the set of all maximal frequent graphs $\mathcal{M}$. Our goal is to find a sample that itself has as diverse a collection of maximal patterns as possible. In other words we want to avoid generating maximal patterns that are very similar to other maximal patterns already found. This necessitates a deviation from traditional enumerative pattern mining approaches.

Enumerative graph mining methods either explore the pattern space in a breadth-first (level-wise) or depth-first manner. The approaches work by extending an existing graph S of size k by adding one more edge to obtain a $(k+1)$-graph S'. The drawback of the breadth-first exploration of the pattern space is that longer patterns may never be reached, due to the combinatorial explosion in the number of subgraphs. On the other hand, depth-first exploration can produce some large maximal patterns, however it is likely to explore only a limited portion of the positive border, and most of the maximal pattern it enumerates will be very similar.

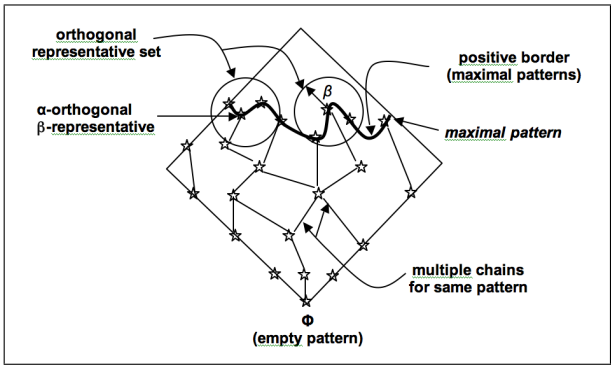


Figure 3. Frequent graph partial order

Random Walks over Chains: ORIGAMI adopts a random walk approach to enumerate a diverse set of maximal patterns from the positive border. Each run of Random-Maximal-Graph (Figure 2, line 5) outputs one random maximal pattern M by starting at the empty pattern and successively adding a random edge during each extension, until no extensions are possible. Each run of the method thus *walks a random chain* in the partial order (recall that a *chain* in a partial order is a path composed of subgraph to immediate supergraph edges). Figure 3 gives an illustration of this process. Each intermediate pattern is denoted by a star, and

there exists an edge between two graphs $G_a \subseteq G_b$ in the partial order if $|G_a| = |G_b| - 1$. The set of all maximal patterns or the positive border is denoted by the bold curve. Each random walk starts at the empty pattern $\emptyset$, and follows a random chain until it hits the positive border. Different runs of Random-Maximal-Graph produce an approximate set of maximal patterns $\widehat{\mathcal{M}}$.

Ideally the random chain walks would cover different regions of the partial order, and would produce dissimilar maximal patterns. However, in practice, this may not be the case, since duplicate patterns can be encountered in the following ways: (i) multiple iterations following overlapping chains, or (ii) multiple iterations following different chains, both leading to the same maximal pattern.

Let's consider a maximal frequent graph M of size n. Let $e_1 e_2 \cdots e_n$ be a sequence of random edge extensions, corresponding to a random chain walk, leading from the empty graph to the maximal graph M. Corresponding to the edge sequence is a series of intermediate graphs on the walk: $\emptyset = S_0 \rightarrow S_1 \rightarrow S_2 \cdots \rightarrow S_n = M$, where S_i is the intermediate obtained by extending S_{i-1} with e_i. The probability of a particular edge-sequence leading from $\emptyset$ to M is given as:

$$P[(e_1 e_2 \cdots e_n)] = P(e_1) \prod_{i=2}^{n} P(e_i | e_1 \cdots e_{i-1}) \qquad (1)$$

In general, any permutation, π, of an edge sequence, i.e., $(\pi(e_1)\ \pi(e_2) \cdots \pi(e_n))$ can also generate the same graph, M; however, all $n!$ permutations may not be valid, since we require all intermediate graphs to be connected. For example, for a k-edge star graph, the number of valid edge-sequences is $k!$, but for a linear k-edge graph (a sequence), the number of valid edge-sequences is 2^{k-1}.

Denote by $ES(M)$ the set of all valid edge-sequences for a graph M. The probability that a graph M is generated in a random walk is proportional to:

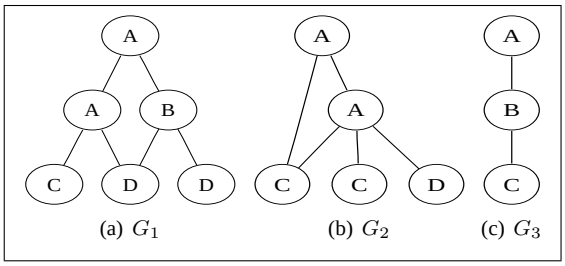
$$\sum_{(e_1 e_2 \cdots e_n) \in ES(M)} P[(e_1 e_2 \cdots e_n)] \qquad (2)$$

The probability of obtaining a specific pattern depends on the number of chains or edge sequences leading to that pattern and the size of the pattern. As we can see from Equation 1, if a graph grows larger, the probability of an edge sequence gets smaller, though a larger graph typically has more chains leading to it. So, our approach, in general, favors a maximal pattern of smaller size over a maximal pattern of larger size. We circumvent this problem by aborting a walk that is likely to generate duplicates or very similar patterns.

Termination Condition: The iterative loop (Figure 2, line 4) that generates the maximal graphs terminates when an appropriate stopping condition is satisfied. The simplest case is to stop after a given number of walks k. We also implemented a dynamic termination condition, based on an estimate of the *collision or hit rate* of the patterns. Intuitively the collision rate keeps track of the number of duplicate patterns seen within the same or across different random walks. As each chain is traversed, ORIGAMI maintains in a bounded-size hash-table, the signature of the intermediate patterns. As each intermediate or maximal pattern is seen, its signature is added to the hash-table and the collision rate is updated. If the collision rate exceeds a threshold ϵ, this information can be used in two different ways: i) Within a given random walk, we can abort further extensions along the current path and force the method to backtrack and choose another path (randomly). ii) Across different walks it can trigger the terminating condition, since a collision rate exceeding ϵ implies that same parts of the partial order are being re-visited. An advantage of this dynamic approach is that the user need not explicitly specify k (though ϵ is now the new parameter).

Random Maximal Graph Generation: As mentioned above the Random-Maximal-Graph method performs a random walk along a chain in the subgraph partial order. Starting from the empty pattern it adds random edges to obtain a succession of intermediate graphs leading to some maximal pattern $M \in \mathcal{M}$. To extend an intermediate pattern, say $S_k \subseteq M$, we first choose a random vertex, say with id v, from where the extension will be attempted. Then, we choose a random edge $e(i, j) \in EM$ from the edge-map, where i and j are the vertex labels of edge, e; the edge can, optionally, have an edge label. Note that i must be equal to the label of vertex v for a proper extension. The edge map data structure can provide all such edges efficiently. If no such e is found, no extension is possible from the vertex v and v is inserted in a list of *expired* vertices. When all vertices in an intermediate graph S_k are expired, the loop breaks and the pattern $S_k = M$ is a maximal pattern. But, if an edge e is found, we randomly choose the other end of this edge. If that is already in the graph S_k, this is called a back-extension, otherwise, it is a forward extension. An edge is added between this node, and the candidate pattern, S_{k+1} is built. Its support is then computed, and if the pattern is infrequent, we insert the following map entry, $(v \rightarrow e)$ in another data structure called the *failed-map*, to ensure that the edge e shall not be attempted at vertex v for extension of S_k any more in a later iteration. Details of the actual support counting via a vertical data representation are essentially the same as the graph mining method in DMTL [5].

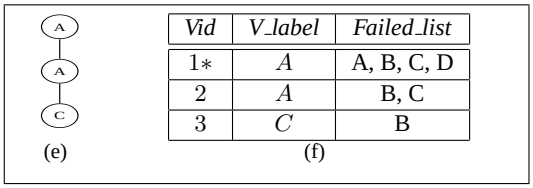
Figure 4 shows an example of the Random-Maximal-Graph algorithm, while finding a random maximal graph from a graph database of size 3 (Fig. 4 a-c) with $\pi^{\min} = 2$. The edge map (Fig. 4 d) records all the possible extensions for a given vertex label, recording the labels of the vertices on the other end of that edge. If the edges have labels, they also become part of the possible other labels. For simplicity, we ignore edge labels in this example. The edge map also remembers the highest frequency of an edge *within* any graph in the database, so that some candidates which are not frequent shall never be attempted. For instance, con-



EDGE MAP with Max. Frequency

Vertex label	*Possible Other Label*
A	A(1), B(1), C(3), D(1)
B	A(1), C(1), D(2)
C	A(3), B(1)
D	A(1), B(2)

(d)



(e)

Vid	*V_label*	*Failed_list*
1∗	A	A, B, C, D
2	A	B, C
3	C	B

(f)

Figure 4. (a-c) A graph database with 3 graphs. (d) The edge map data structure that shows possible extensions, with the maximal edge count. (e-f) A snapshot of the random extension process while mining with $\pi^{\min} = 2$. The failed-list table shows which edge extensions have been attempted and which failed; * denotes an expired vertex id.

sider the candidate frequent graph A—A—C, which is not maximal (Fig. 4 e). But, the graph already has one A—A edge, with vertex ids (vid) 1 and 2, respectively. Since the maximum frequency of the A—A edge is 1, the edge extension A—A shall never be attempted from vid 1 or 2. The failed list that we maintain along with every iteration of the maximal graph generation process is also shown (Fig. 4 f). Note that for vid 1, all possible labels for the other end are in the failed list, i.e., they had been attempted and found to produce infrequent graphs. So, vid 1 is marked as expired (denoted by *). When all the vertices are expired the process terminates and we obtain a maximal graph. For this particular example, adding an edge A—D at vid 2, yields the maximal graph with support 2 (in graphs G_1 and G_2).

4.2 Orthogonal Representative Sets

Given a set of maximal patterns $\widehat{\mathcal{M}}$, ORIGAMI extracts an α-orthogonal β-representative set from it.

Theorem 4.1 *Given $\widehat{\mathcal{M}}$, let $\Gamma(\widehat{\mathcal{M}})$ be the graph with $V = \widehat{\mathcal{M}}$ and $E = \{(M_a, M_b) | sim(M_a, M_b) \leq \alpha\}$. Then any α-orthogonal set $\mathcal{R}$ is a maximal clique in $\Gamma(\widehat{\mathcal{M}})$, and vice-versa.*

PROOF*:* If $\mathcal{R}$ is an α-orthogonal set, then for any $M_a, M_b \in \mathcal{R}$, $sim(M_a, M_b) \leq \alpha$, and for any $M_a \in \widehat{\mathcal{M}} \setminus \mathcal{R}$, there exists $M_b \in \mathcal{R}$, with $sim(M_a, M_b) > \alpha$. This implies that the α-orthogonal set must be maximal. Since $\Gamma(\widehat{\mathcal{M}})$ has edges only for α-orthogonal graphs, it follows that every maximal clique in $\Gamma(\widehat{\mathcal{M}})$ is α-orthogonal set, and vice-versa. ■

As mentioned earlier, the α-orthogonality controls the amount of redundancy allowed among the output patterns. For a given α, several maximal cliques can exist in the graph $\Gamma(\widehat{\mathcal{M}})$, each a feasible solution to the orthogonal set problem. The β-representative condition allows each element of the orthogonal to represent similar graphs, and also allows us to rank the maximal cliques in terms of their residue (or average residue similarity).

There are several challenges in finding the optimal α-orthogonal β-representative set. At the outset it should be noted that the orthogonal set is a representative set only for the sample $\widehat{\mathcal{M}}$. If sufficient number of maximal patterns were not sampled (for example, if the stopping condition were too restrictive), then $\widehat{\mathcal{M}}$ may not approximate the set of all maximal patterns $\mathcal{M}$ very well, and the quality of $\widehat{\mathcal{M}}$ would suffer. Another challenge is that, depending on the size of the maximal set $\widehat{\mathcal{M}}$, it may not be reasonable to compute the full pair-wise similarity matrix between all elements of $\widehat{\mathcal{M}}$, since it has $O(\widehat{\mathcal{M}}^2)$ time and space complexity. That is, it may not be reasonable to compute the full graph $\Gamma(\widehat{\mathcal{M}})$. Even if $\Gamma(\widehat{\mathcal{M}})$ were available, the challenge is that finding the optimal maximal clique that minimizes the residue is an NP-hard problem.

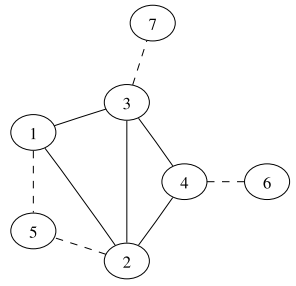
Theorem 4.2 *Finding the optimal α-orthogonal β-representative that minimizes the residue is NP-hard.*

PROOF*:*This is easy to show, since the general problem contains an NP-hard sub-case. For $\beta = 1$, each element in the α-orthogonal set represents only itself, giving the residue for any $\mathcal{R}$ as $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| = \widehat{\mathcal{M}} \setminus \mathcal{R}$. Thus minimizing the residue for $\beta = 1$ corresponds to solving the maximum clique problem, which is known to be NP-hard. ■

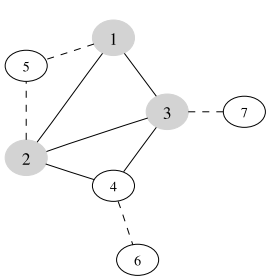
Given the hardness result, instead of enumerating the optimal maximal clique, we resort to approximate algorithms to solve the problem efficiently. Since, the optimal solution is a maximal clique of the similarity graph, we adopt maximal clique finding as a heuristic. Using this approach, ORIGAMI finds a maximal clique without computing the full similarity matrix. Given the set $\widehat{\mathcal{M}}$ it randomly selects one element $M \in \widehat{\mathcal{M}}$, and adds it to $\mathcal{R}$. The idea is to iteratively add one element from $\widehat{\mathcal{M}} \setminus \mathcal{R}$ to the current $\mathcal{R}$ set until no more elements can be added, which would yield a maximal clique. At any intermediate step, we compute the similarities for all $M_b \in \widehat{\mathcal{M}} \setminus \mathcal{R}$ to elements $M_a \in \mathcal{R}$. If there exists $M_b \in \widehat{\mathcal{M}} \setminus \mathcal{R}$, such that $sim(M_a, M_b) \leq \alpha$ for all $M_a \in \mathcal{R}$, we add M_b to $\mathcal{R}$. This process is repeated

until a maximal clique is obtained. The complexity of finding a single clique is $O(|\widehat{\mathcal{M}}||\mathcal{R}|)$, but in general we expect $|\mathcal{R}| \ll |\widehat{\mathcal{M}}|$, so that the time is closer to $O(|\widehat{\mathcal{M}}|)$. Finally, to obtain multiple cliques ORIGAMI simply starts with different initial maximal graphs. Finally the best clique is chosen based on the residue size.

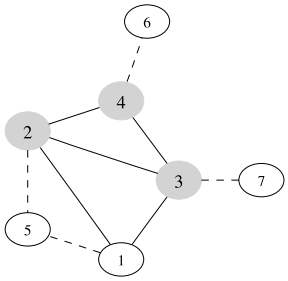
We also designed an approximate solution, which is better than the above heuristic approach and also guarantees local optimality. The neighborhood structure of the local optimal formulation uses maximal clique in a meta-heuristic approach. The algorithm starts with a random maximal clique. At each state transition, another maximal clique which is a local neighbor of the current maximal clique, is chosen. If the new state has a better solution, the new state is accepted as the current state and the process continues. The process terminates when all neighbors of the current state have equal or higher residue size. Two maximal cliques of size m and n (where, $m \geq n$) are considered neighbors, if they share exactly $n-1$ vertices. The state transition procedure selectively removes one vertex from the maximal clique of current state and then expands it to obtain another maximal clique, which satisfies the neighborhood constraints.



(a) A similarity graph, solid lines represent elements with similarity $\leq \alpha$, broken lines represent similarity $\geq \beta$



(b) Initial clique (1,2,3) with residue=2



(c) A local neighbor clique (2,3,4) with better residue=1

Figure 5. Local optimization example

Figure 5 shows an example state transition for the local-optimal algorithm. In Figure 5(a) we show a toy similarity graph, where the solid lines represent low similarity ($\leq \alpha$) and broken lines represent high similarity ($\geq \beta$) between corresponding elements. Figure 5(b) shows an initial clique $(1, 2, 3)$ which has residue = 2 (since element 4 and 6 are not covered). Figure 5(c) shows a neighboring clique $(2, 3, 4)$, having 2 common nodes (2 and 3), that has a better residue value (residue = 1; since only element 1 is not covered). Thus, the local optimal algorithm will accept the new clique in Figure 5(c) and will continue. For this toy example, this clique is also optimal. In the experimental section we show the performance superiority of the local optimal method over the random clique approach.

Different Similarity Measures: Computing similarity between graphs is one of the significant tasks in finding the α-orthogonal graph set. Similarity can be measured by using features in the pattern space or in the transaction space (the gidset) or a combination of both of the above. In the case of pattern space, the most common way to compute similarity is using the edit distance between two patterns. Depending on the pattern complexity, the cost of edit distance computation varies. For complex patterns like graphs, the computation is usually costly. On the other hand, the similarity in the gidset space is very easy to compute. A ratio of intersection-set and union-set can represent a similarity. For two patterns G_a and G_b, it can be computed as: $sim(G_a, G_b) = \frac{|\mathbf{t}(G_a) \cap \mathbf{t}(G_b)|}{|\mathbf{t}(G_a) \cup \mathbf{t}(G_b)|}$. This is a very crude measure for similarity since, two very different patterns can have a very similar set of transactions. We did not use this measure in our work. But, for simpler patterns, like itemsets, it plays an important role in finding distances between patterns.

We used the graph similarity measure proposed by Bunke et al. [2] that computes the similarity between two patterns by finding the relative size of their common subpatterns. For the case of graphs, this is equivalent to finding the relative size of the maximal common subgraph of two graphs. If G_1, G_2 are two graphs and G_{mc} is the maximum common sub-graph between these two graphs, then the following equation computes the similarity: $sim_{mc}(G_1, G_2) = \frac{|G_{mc}|}{\max(|G_1|, |G_2|)}$.

For our purpose, we computed the similarity by using a maximal graph mining algorithm [7], that takes two graphs as input and mines for maximal graph patterns with 100% support. The frequent maximal graph of maximum size is used to compute the size of the maximal common subgraph in the similarity equation.

However, computing the exact similarity by solving the maximal common subgraph can be costly, and for the α-orthogonal graph problem, most often, we can compute a lower bound on the graph-distance by considering a graph as a labeled edge-multiset. We define the edge-multiset similarity as follows: Let G_1, G_2 be graphs, and s be the similarity between them as computed using sim_{mc}. Let E_{G_1} and E_{G_2} be the edge-multiset where each edge is defined by an ordered triple of its vertex labels and edge label: $\langle v_{l1}, e_l, v_{l2} \rangle$. The edge multiset similarity is then given as: $sim_{em}(G_1, G_2) = \frac{|E_{G1} \cap E_{G2}|}{\max(|E_{G_1}|, |E_{G_2}|)}$. Now, the following lemma always holds.

Lemma 2 $sim_{em} \geq sim_{mc}$.

PROOF: $sim_{em} \geq sim_{mc}$, unless $|G_{mc}| > |E_{G_1}| \cap |E_{G_2}|$. But this is impossible, since all the edges in G_{mc} are present

in both the sets E_{G_1} and E_{G_2}. ■

In computing similarity between two patterns, we first compute the sim_{em}. If sim_{em} is smaller than α, according to Lemma 2, sim_{mc} is also smaller than α, and the corresponding patterns satisfy the α-orthogonal constraints. Otherwise, we compute sim_{mc}.

5 Experiments

5.1 Dataset Description

Chemical Compound Datasets (DTP and CM: The chemical dataset is obtained from the DTP AIDS Antiviral Screen test. The dataset can be retrieved from DTP website [2]. The dataset is classified into three subsets of compounds: confirmed active (CA), confirmed moderately active (CM) and confirmed inactive (CI). Each chemical compound is modeled as a graph where atoms represent the labeled vertices and bonds represent the labeled edges of the graph. There are 3 bond types and 61 vertex types. The full DTP database has 40942 graphs, with average graph size 45 edges and 43 vertices. The CM subset has 1084 graphs with average 31 vertices and 34 edges.

Protein Structure Dataset (PS): Given a protein structure, we create a protein graph as follows. Each amino acid residue is treated as a vertex (labeled by one of the 20 amino acids), and there exists an edge between two vertices v_i and v_j if $d(v_i, v_j) \leq t$, i.e., if the Euclidean distance between the C_α atom of the residues is at most t (we use $t = 7\mathring{A}$). We created a database of 100 proteins (10 structural families, with 10 proteins from each family), from the HOMSTRAD (`http://www-cryst.bioc.cam.ac.uk/~homstrad/`) database of structurally-aligned homologous proteins. The protein graphs have on average 165 nodes and 734 edges. The goal is to discover the orthogonal representative structural motifs for each protein family.

Protein Interaction Dataset (PI): Data on pairs of interacting proteins was collected from three different sources. This dataset contains only 3 large graphs, with an average of 2154 vertices and 81,607 edges per graph [3]. Each interaction graph is created using one source: the first graph has an edge if the proteins involved are known to interact (via biological experiments), the second graph has an edge if the proteins are part of a known pathway, and the third graph has an edge if the proteins have correlated gene expression values.

Synthetic Implanted Dataset (SI): We wrote a graph generator that accepts seed graphs and implants them in larger graphs to create a database $\mathcal{D}$. First, we restrict the seeds to be α-orthogonal. The $|S|$ α-orthogonal seeds can be generated randomly or they may be extracted from a real dataset. We generate $|\mathcal{D}|$ graphs with the average graph size taken from a Poisson distribution with mean T. Seeds are selected to be added to the current graph $D_i \in \mathcal{D}$ uniformly at random; as each seed is added we ensure that the graph D_i remains connected (by adding random edges). If the addition of a new seed to D_i would exceed size T, instead of adding the seed, we make up the differential by adding edges/vertices randomly to existing nodes in D_i. The vertex and edge labels are chosen randomly from L_V (the vertex labels) and L_E (the edge labels), respectively.

[2] http://dtp.nci.nih.gov/docs/aids/aids_data.html

[3] This dataset was provided by Prof. Igor Kuznetsov at SUNY, Albany

5.2 Empirical Results

All experiments were run on a 2.75Ghz PowerPC G5 Machine with 4GB Memory and 400GB disk. Since ORIGAMI is randomized, we perform several runs (typically between 3 to 5). Each run generates an approximate maximal set $\widehat{\mathcal{M}}$. We next extract several orthogonal representative sets (typically 10) using our primary algorithm that reports the best clique found. All numbers reported in the experiments below are the averages over the best results over all the runs. Wherever possible we tried to run state-of-the-art graph mining methods like gSpan [14], and DMTL [5] (which mine all frequent subgraphs) and SPIN [7] (which mines maximal graph patterns). The local optimization algorithm was used only in the result that compares against the random maximal clique algorithm.

5.2.1 Protein Interaction Mining

First we evaluate our random walks approach to mining maximal patterns. As mentioned in the introduction, we ran a depth-first graph mining algorithm from DMTL [5] to mine the protein interaction dataset (PI), looking for frequent graphs at $\pi^{\min} = 100\%$ (3 out of 3). The method was running for over a day before we terminated it. During this time it had generate a 7GB output (from an initial 3MB database), containing 8 million subgraphs. SPIN was not able to run on this dataset; it terminated with a segment fault. Utilizing the fact that each protein appears only once in a given graph, we converted each graph into an itemset of edges, and we were then able to mine the maximal edgesets. At $\pi^{\min} = 100\%$ this yields 90 maximal frequent graphs.

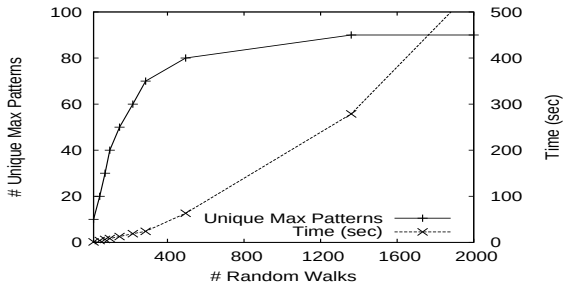


Figure 6. Random Walk Performance (PI)

Next, we ran ORIGAMI on the original PI dataset. Figure 6 shows the number of unique maximal patterns found

versus the number of random walks. The figure shows that *all* 90 maximal patterns were found after 1400 random walks, and it took under 300s running time! This illustrates the effectiveness of our random walks maximal pattern mining approach. In this particular example, it was able to return the exact set of maximal patters (i.e., $\widehat{\mathcal{M}} = \mathcal{M}$).

5.2.2 Protein Structure Mining

$\pi^{\min}$	Time(s)	$\lvert\widehat{\mathcal{M}}\rvert$	$\lvert\mathcal{R}\rvert$	$\mathcal{H}(\widehat{\mathcal{M}})$	$\mathcal{H}(\mathcal{R})$	$\max \mathcal{H}$
7	154.4	1002	213	1.092	1.039	1.946
8	75.5	1003	180	1.154	1.128	2.079
9	64.9	1007	190	1.217	1.195	2.197
10	50.4	1009	184	1.274	1.243	2.303

Table 1. Protein Structure Mining

Table 1 shows the time taken to mine the protein structure dataset at different values of minimum support. It also shows the number of maximal and α-orthogonal patterns found (for $\alpha = 0.2$). Also shown is the average entropy of the patterns in $\widehat{\mathcal{M}}$ and in $\mathcal{R}$. Note that for a set of graphs $\mathcal{G}$, the average entropy is given as $\mathcal{H}(\mathcal{G}) = \frac{\sum_{G \in \mathcal{G}} \mathcal{H}(G)}{|\mathcal{G}|}$, where $\mathcal{H}(G) = -\sum_i p_i \ln p_i$, where p_i is the fraction of occurrences of G in protein family i. For example, if $\pi^{\min} = 8$, and the protein subgraph appears in 8 different HOMSTRAD families, then its entropy will be $-8\frac{1}{8}\ln(\frac{1}{8}) = 2.079$. The maximum possible entropy for a pattern with support exactly $\pi^{\min}$ is also shown. We can see that in general ORIGAMI produces relatively good patterns that have about half the entropy compared to the maximum entropy. An example of a low entropy pattern in the Immunoglobulin family from HOMSTRAD is shown in Figure 7.

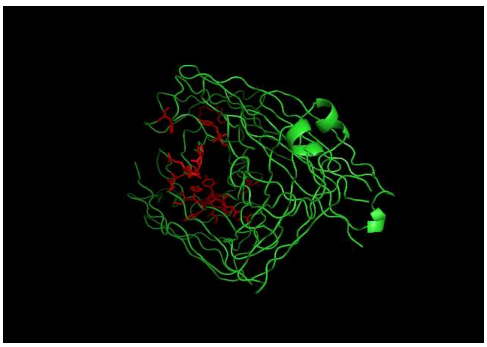
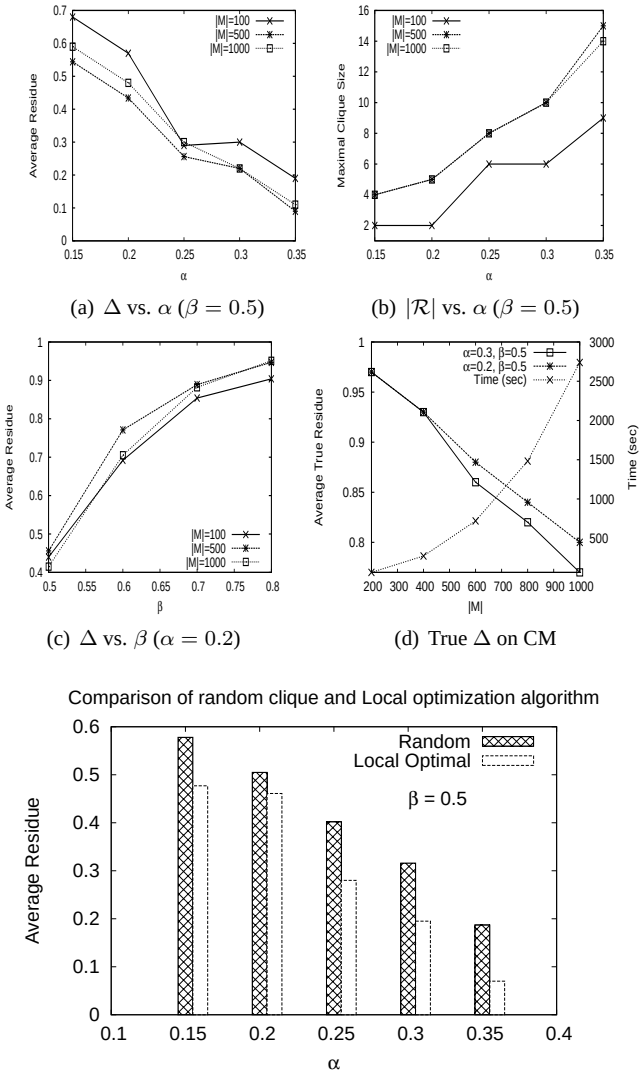


Figure 7. Low Entropy Motif (in Red)

5.2.3 Chemical Compound Mining

Next we mined the chemical compound datasets. Note that neither gSpan nor SPIN were able to run on the full 40942 graph DTP dataset. On the other hand, we were able to successfully run ORIGAMI on DTP, using as the stopping criteria for $\widehat{\mathcal{M}}$, the number of unique maximal patterns generated. We next extracted orthogonal representative sets for different values of α and β.



(a) Δ vs. α ($\beta = 0.5$)

(b) $|\mathcal{R}|$ vs. α ($\beta = 0.5$)

(c) Δ vs. β ($\alpha = 0.2$)

(d) True Δ on CM

(e) Random maximal clique vs. local optimization on CM

Figure 8. Performance on DTP and CM

The results are shown in Figure 8. In (a)-(c) we plot three curves, corresponding to increasing number of unique maximal patterns found, i.e., for different $|\widehat{\mathcal{M}}|$ values. Figure 8(a) plots the effect of α on the average residue, which is defined as $\frac{|\Delta(\mathcal{R},\widehat{\mathcal{M}})|}{|\widehat{\mathcal{M}}|}$. As we can observe, as α increases the average residue shrinks to under 10% indicating that the α-orthogonal β-representative set has left unrepresented less than 10% of the mined maximal patterns $\widehat{\mathcal{M}}$. Figure 8(b) plots the size of the orthogonal representative set (or maximal clique) for different α. We see that, as expected, bigger cliques are found for larger α. Figure 8(c) shows the effect of β on average residue. As β increases, we find that average residue increases, since the more stringent (i.e., higher) the representativeness threshold, the fewer the patterns that are represented.

Whereas SPIN was not able to run on the full DTP

dataset, we were able to run it on the smaller 1084 CM dataset at a minimum support of $\pi^{\min} = 25/1084 = 2.3\%$. At this support level it output 1227 maximal patterns in about 181s. Thus for this smaller dataset we know the true set of maximal patterns $\mathcal{M}$. Figure 8(d) plots the average residue with respect to the true maximal set $\mathcal{M}$, and the time for mining as a function of the size of $\widehat{\mathcal{M}}$. The observed trend is that as $|\widehat{\mathcal{M}}|$ increases, the average true residue also decreases, since the orthogonal set is able to represent more true maximal graphs.

Figure 8(e) shows a comparison of random maximal clique method and the local optimization method for different α values using the CM dataset. In every case, the residue of the local optimal method is 30% to 50% smaller than that of the random maximal clique method.

5.2.4 Implanted Seed Mining

$\|\mathcal{D}\|$	T	$\pi^{\min}$	$\|\mathcal{R}\|$	Seeds Found
20000	50	200	25	4/9
30000	50	200	26	5/9
40000	50	200	27	8/9
20000	50	*400*	23	4/9
20000	50	*800*	19	4/9
20000	*20*	200	25	4/9
20000	*30*	200	21	5/9
20000	*40*	200	25	6/9
20000	*60*	200	24	7/9

Table 2. Results on Implanted Seeds

The goal of this experiment is to recover implanted seeds. We generated a set of seeds from the full DTP dataset as follows: Initially $|\widehat{M}| = 3550$ maximal patterns were generated from DTP using ORIGAMI. Next an orthogonal set $\mathcal{R}$ is mined at $\alpha = 0.6$, which yielded a maximal clique of size $|\mathcal{R}| = 9$. These 9 seed graphs, containing on average 6.4 edges and 7.4 nodes, were fed into the graph generator to create varying datasets, as shown in Table 2.

Once the datasets were generated we mined them at different $\pi^{\min}$ values, and used $\alpha = 0.8$ to extract the α-orthogonal sets. The first three rows show results for varying dataset size. As the dataset size increases the number of seeds found increases, since the odds of the graphs containing the 9 implanted seeds increases. Rows 4 and 5 in the table show results for varying the minimum support on a dataset with 20000 transactions. The number of seeds captured does not change in this case. Rows 6-9 show the effect of varying the average size (T) of a graph in $\mathcal{D}$. The results confirm our intuition that as the average size increases, the number of seeds in a graph increases, resulting in greater number of seeds being captured.

6 Conclusions

In this paper we proposed a new paradigm for mining a summary representation of the set of frequent graphs. This is a very difficult problem to solve, as it consists of individually hard problems: i) computing similarity between graphs, ii) random sampling from the set of frequent maximal graphs, and iii) finding maximal cliques. ORIGAMI employs effective techniques to tackle these challenges, as demonstrated empirically on a variety of datasets. Unlike previous techniques that focus on the distance in the transaction space to obtain representatives, our approach captures representatives by considering the distances in the pattern space. We introduced a randomized approach for mining maximal subgraph patterns. The method is designed to cover the partial order of subgraphs, so that orthogonal maximal patterns are obtained quickly. We formulated the α-orthogonal β-representative set finding as an optimization problem. We show that the optimization problem is NP-Hard and we thus propose a local optimization solution that is efficient and practically feasible. We demonstrate that ORIGAMI is able to mine good quality orthogonal representative sets, especially for datasets where traditional enumerative methods fail completely.

References

[1] F. Afrati, G. Gionis, and H. Mannila. Approximating a collection of frequent sets. In *SIGKDD*, 2004.

[2] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19:255–259, 1998.

[3] T. Calders, C. Rigotti, and J.-F. Boulicaut. A Survey on Condensed Representation for Frequent Sets. In *Constraint-Based Mining and Inductive DB (LNCS Vol. 3848)*, 2005.

[4] D. Cook and L. Holder. Substructure discovery using minimal description length and background knowledge. *Journal of Artificial Intelligence Research*, 1:231–255, 1994.

[5] M. Hasan, V. Chaoji, S. Salem, N. Parimi, and M. Zaki. DMTL: A generic data mining template library. In *Workshop on Library-Centric Software Design (w/ OOPSLA)*, 2005.

[6] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *ICDM*, 2003.

[7] J. Huan, W. Wang, J. Prins, and J. Yang. SPIN: Mining Maximal Frequent Subgraphs from Graph Databases. In *SIGKDD*, 2004.

[8] A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.

[9] M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. In *ICDM*, 2001.

[10] S. Nijssen and J. Kok. A quickstart in frequent structure mining can make a difference. In *SIGKDD*, 2004.

[11] L. Thomas, S. Valluri, and K. Karlapalem. MARGIN: Maximal Frequent Subgraph Mining. In *ICDM*, 2006.

[12] D. Xin, H. Cheng, X. Yan, and J. Han. Extracting Redundancy-Aware Top-k Patterns. In *SIGKDD*, 2006.

[13] D. Xin, J. Han, X. Yan, and H. Cheng. Mining Compressed Frequent-Pattern Sets. In *VLDB*, August 2005.

[14] X. Yan and J. Han. gSpan: Graph-Based Substructure Pattern Mining. In *ICDM*, 2002.

[15] X. Yan and J. Han. CloseGraph: Mining Closed Frequent Graph Patterns. In *SIGKDD*, 2003.

[16] X. Yan, P. S. Yu, and J. Han. Graph Indexing: A Frequent Structure-based Approach. In *SIGMOD*, 2004.