ORIGAMI: A Novel and Effective Approach for Mining Representative Orthogonal Graph Patterns[†]

Vineet Chaoji[1], Mohammad Al Hasan[1], Saeed Salem[1], Jeremy Besson[2] and Mohammed J. Zaki[1]

[1] *Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 12180, USA*

[2] *INSA-Lyon, LIRIS UMR5205, F-69621 Villeurbanne, France*

Received 31 October 2007; revised 07 February 2008; accepted 13 February 2008
DOI:10.1002/sam.10004
Published online 9 April 2008 in Wiley InterScience (www.interscience.wiley.com).

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. © 2008 Wiley Periodicals, Inc. Statistical Analysis and Data Mining 1: 67–84, 2008

Keywords: graph mining; randomized algorthims; maximal pattern mining

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 because of 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 6), a typical enumerative graph mining method (we used gSpan [19]) did not finish running in even two days. These six graphs contain common motifs of size over 50–60 residues, and therefore 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 [9] on a protein-interaction dataset consisting of three graphs (see dataset protein interaction (PI) in Section 6), each graph having 2154 nodes, and on average 81607 edges, with total database size 3 MB. The mining process was aborted after a day of running, at which point it had generated a 7 GB 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

Correspondence to: Mohammed J. Zaki (zaki@cs.rpi.edu)
[†] The first two authors contributed equally for this research.

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 such as 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 a 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 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, which 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 [5, 12,13,19,10,15]. 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 [21].

There are several works guided towards finding a subset of frequent patterns that are most informative, compressed, discriminative and non-redundant [1,18,17,4]. 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 [20] and maximal frequent graphs [16,11]. 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 the 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 such as itemsets, sequences and trees.

3. ORIGAMI: PROBLEM FORMULATION AND OVERVIEW

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 $\mathcal{V} : V \to L_V$ and $\mathcal{E} : E \to L_E$ 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 $f : V_1 \to V_2$, 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.

Graph Similarity Measures: Similarity between graphs can be measured by using features in the pattern space or in the transaction space (the gidset) or a combination of both. 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 such as graphs, the computation is usually costly. On the other hand, the similarity in the gidset space is very easy to compute. The ratio of intersection-set and union-set can be used as a similarity measure. 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 rudimentary measure for similarity since two very different patterns can have a very similar set of transactions. We do not use this measure in this work. But, for simpler patterns, such as itemsets, it plays an important role in finding distances between patterns.

In this work, we used the graph similarity measure proposed by Bunke et al. [3,2], which computes the similarity between two patterns by finding the relative size of their common sub-patterns. 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 subgraph between these two graphs, then the following equation computes the similarity:

$$sim_{mc}(G_1, G_2) = \frac{|G_{mc}|}{\max(|G_1|, |G_2|)} \tag{1}$$

Orthogonal and Representative Graphs: Define $sim : \mathcal{F} \times \mathcal{F} \to [0, 1]$ to be a symmetric binary function that returns the *similarity* between two graphs. For instance, the similarity based on the maximum common subgraph [3] is given above. 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

[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.

an 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 $G_j \in \mathcal{R}$. We 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 $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| = |\widehat{\mathcal{M}} \setminus \mathcal{R}|$.

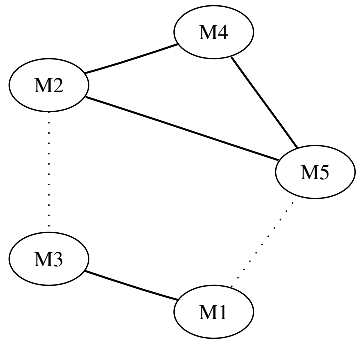
As an example, assume that we are given the pairwise similarities between a set of graphs $\widehat{\mathcal{M}}$, as shown in Fig. 1. If $\alpha = 0.2$, then there are two possible α-orthogonal sets, namely $\mathcal{R}_1 = \{M_1, M_3\}$ and $\mathcal{R}_2 = \{M_2, M_4, M_5\}$ as illustrated in Fig. 1(b). If $\beta \leq \alpha$, both 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 Fig. 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.

(a)

	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

SimilarityMatrix

(b)



Similarity Graph

Fig. 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.

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 pattern not 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.

The ORIGAMI Approach: 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 Fig. 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 structures 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). The first step of ORIGAMI is described in Section 4 and the second step is elaborated in Section 5.

4. 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 because of 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 patterns it enumerates will be very similar.

4.1. 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 (Fig. 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 super-graph 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, taken together, produce an approximate set of maximal patterns $\widehat{\mathcal{M}}$.

```
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 ($\widehat{\mathcal{M}}, \alpha, \beta$)
```

Fig. 2 ORIGAMI Algorithm.

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 us 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:

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

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}, which can be easily shown by induction.

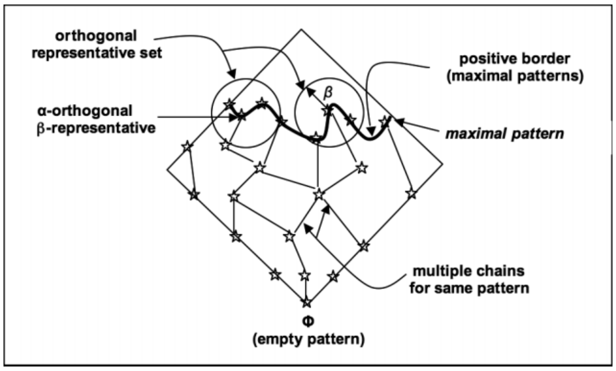

orthogonal representative set
positive border (maximal patterns)
α-orthogonal β-representative
β
maximal pattern
multiple chains for same pattern
Φ (empty pattern)

Fig. 3 Frequent graph partial order.

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:

$$\sum_{(e_1 e_2 \cdots e_n) \in ES(M)} P[(e_1 e_2 \cdots e_n)] \tag{3}$$

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 Eq. 2, if a graph grows larger, the probability of an edge sequence gets smaller, though a larger graph typically has more chains leading to it.

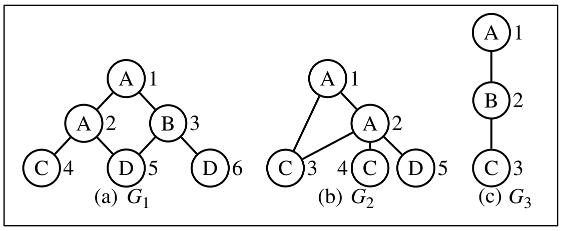
Termination Condition: The iterative loop (Fig. 2, line 4) that generates the maximal graphs terminates when an appropriate stopping condition is satisfied. The simplest case is to stop after a given number of walks k. We also implemented a dynamic termination condition, based on an estimate of the *collision 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 the signature of the intermediate patterns in a bounded-size hash-table. 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, the terminating condition can be triggered, since a collision rate exceeding ϵ implies that parts of the partial order are being revisited. An advantage of this dynamic approach is that the user need not explicitly specify k (though ϵ is now the new parameter).

4.2. Random Maximal Graph Generation

Let us take a closer look at the Random Maximal Graph method that 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 source vertex, with id v and label i, from where an extension will be attempted. Then, a random destination vertex label j is chosen from the EM, out of all $e(i, j)$ node pairs that form the two ends of an edge. The edge can, optionally, have an edge label. If no such e is found, no extension is possible from vertex v, and v is inserted in a list of *expired* vertices. When all vertices in the intermediate graph S_k are expired, the loop breaks and the pattern $S_k = M$ is a maximal pattern. But, if such edges are found, we randomly choose one of them. Note that there can be multiple edges with one end at vertex v and the other end at a vertex with label j. If such an edge is already in the graph S_k, it is called a backward extension: otherwise, it is a forward extension. An edge is added between node v and this node, resulting in the candidate pattern S_{k+1}. 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 again. Details of the actual support counting via a vertical data representation are essentially the same as the graph mining method in data mining template library (DMTL) [9].

Figure 4 demonstrates 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 EM (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, the edge label is recorded along with the vertex label of the other end. 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, consider the candidate frequent graph A—A—C, which is not maximal (Fig. 4 (e,f)). 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 (g)). 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).

Support Counting using Vertical Data Representation: ORIGAMI uses a vertical data format for support counting. In this approach, a data structure is maintained for each pattern, which captures all its embedding in every transaction graph. We called this the vertical attribute table (VAT) of the pattern. At the beginning, the VAT structures of all the frequent single edge patterns are created (Fig. 2 step 2). While a pattern is extended, its VAT is intersected with the VAT of the new edge (that is added to the pattern) to obtain the VAT of the new pattern. Figure 4(e) shows the VAT for the pattern A–A–C. Since this graph has two edges, we have two rows in the VAT table, one for each of its edges. The pattern appears in G_1 and G_2, and the corresponding columns in the VAT table record all its embedding. Note that in G_2 the pattern has totally three embedding, of which two embedding share the edge A(vid 1)–A (vid 2).



EDGE MAP with Max. Frequency

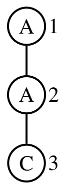
Vertex label	*Possible Other Label*
A	A(1), B(1), C(3), D(1)
B	A(1), C(1), D(2)
C	A(3), B(1)
D	A(1), B(2)

(d)

VAT of pattern in figure 4(f)

Edge	*G*1	*G*2
A – *A*	1 – 2	1 – 2
A – *C*	2 – 4	2 – 3 2 – 4
		1 – 3

(e)



(f)

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

(g)

Fig. 4 (a–c) A graph database with three graphs. (d) The EM data structure that shows possible extensions, with the maximal edge count. (e) An example VAT for the pattern in figure (f). (f–g) 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.

4.3. Collision Assisted Random Walk over Chains

As noted earlier, during a random walk previously explored graphs could be visited. Each repeated visit to an intermediate graph is considered as a collision (in a hash context). The repeated collisions during the random walks can also be used for controlling the traversal over the graph partial order. The advantage of such a biased traversal is two-fold. First is the reduction in the number of repeated maximal patterns generated by the random walks. Second, an evenly spread traversal over the graph partial order could lead to a smaller residue set (equivalently, a larger $\Upsilon(\mathcal{R}, \mathcal{G})$ set).

As discussed in Section 4.1, we maintain a hash containing a signature for the graph along with an integer value. The integer value captures the number of times a given graph is generated across all random walks. The signature is inserted the first time a graph is generated. During a random walk, each generated graph is checked against the hash. A sequence of k contiguous collisions prompts the algorithm to take a corrective action such that further collisions can be averted and the partial order explored uniformly. The intuition is that a sequence of collisions is indicative of a previously explored portion of the partial order. Once we determine that we have ventured into a previously explored region, various corrective actions can be taken. One option would be to completely discard the current random walk and start afresh. This approach leads to wasted effort, as we do not leverage the intermediate graphs leading up to the sequence of collisions. Another strategy that we apply is to backtrack along the current chain till we find an 'appropriate' intermediate graph, from which the random walk is restarted. The following criteria can be used to select such an intermediate graph from the current random walk.

- **Number of collisions** - Backtrack to an intermediate graph in the current path that has had the minimum number of collisions. This idea is driven by the logic that the partial order around such a graph has not been sufficiently explored.
- **Drop in support** - The graph that has the highest drop in support value as compared to its successor in the chain. Such a graph has the potential of multiple paths leading to the border of the partial order, since the difference between its support and the minimum support is large.

One can think of other global approaches as compared to the two path-based criteria discussed above. In a global approach, nodes are ranked by a certain criterion that represents their probability of leading to newer maximal patterns. For instance, a graph in the partial order that has a larger number of super-graphs which are maximal has a higher chance, as compared to a graph that has paths leading to fewer maximal graphs. A set of patterns $\mathcal{P}$ with the highest estimated probability is stored at any time. As the random algorithm proceeds, it is likely that all maximal super-graphs of a particular graph g are explored. This implies that the probability of g leading to newer maximal graphs will go to zero. Nodes are removed or inserted into set $\mathcal{P}$ based on changes in their estimated probability. When a random traversal encounters k collisions, it can restart its traversal from the node that has the highest probability of reaching newer maximal patterns. The above approaches are designed with the aim of providing a uniform and a wider traversal of the positive border. Experiments in Section 6.3 compare the collision-directed approach with the random approach, both in terms of the computation time and the quality of the selected representatives.

4.4. Convergence Rate of Random Walk

ORIGAMI generates maximal graphs by randomly traversing the partial order structure. With a suitable terminating condition, we get a collection of maximal patterns $\widehat{\mathcal{M}} \subseteq \mathcal{M}$; where, $\mathcal{M}$ is the true set of maximal patterns. If t is the number of iterations and $(\widehat{\mathcal{M}})_t$ is the set of maximal patterns after t iterations, we have, $\lim_{t\to\infty}(\widehat{\mathcal{M}})_t = \mathcal{M}$. This is obvious since every pattern has a generation probability strictly greater than 0. Now, the convergence rate of the above limit depends on the dataset: more precisely, on the probability of generating each maximal pattern. If the generation probability of some patterns are very small compared to some other patterns, then the convergence rate could be low. The best convergence can be achieved when all the patterns have an equal probability to be generated. Recall that ORIGAMI is not intended to be a complete maximal graph mining algorithm. It is better suited for large datasets where the traditional maximal graph mining algorithms fail to finish, so the user is willing to sacrifice the completeness and is content with a set of representative frequent patterns.

The rate of convergence can be formally analyzed by answering the following question:

Is there a probabilistic bound on the number of random walks required to capture a certain number (or fraction) of the set of maximal patterns ($\mathcal{M}$)?

For a database $\mathcal{D}$ of graphs and a given minimum support $\pi^{\min}$, let us assume that the set of maximal patterns $\mathcal{M}$ is known. Let the probabilities of arriving at each of these maximal graphs be $p_1, p_2, \ldots, p_{|\mathcal{M}|}$, such that $\sum_{i=1}^{i=|\mathcal{M}|} p_i = 1$. With these assumptions, bounds for the number of random walks to capture a certain number maximal patterns can be provided by using the results for the generalized Coupon Collector's problem. The basic Coupon Collector's problem [14] is defined as follows.

DEFINITION 1 (Coupon Collector's Problem) Given u distinct coupons, what is the expected number of attempts required for picking d distinct coupons, when coupons are drawn at random with replacement. Each coupon is equally likely (i.e. $p_1 = p_2 = \ldots = p_{|\mathcal{M}|}$) to be picked and each attempt is independent of the others.

If X_d is the random variable for the number of attempts to obtain d distinct coupons, then the expected value for X_d is given by Lemma 2.

LEMMA 2: [14] $E[X_d] = u(H_u - H_{u-d})$; where, H_c is the c-th harmonic number, given by $H_c = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{c}$. ■

For, $d = u$, we have, $E[X_u] = O(u \ln u)$. In ORIGAMI setting, the number of maximal patterns is the number of distinct coupons and the number of iterations is the number of attempts to pick coupons. However, in our formulation, the probability of each maximal graph (coupon) is not uniform. The generalized version of the Coupon Collector's problem provides a closed-form solution for $E[X_d]$ when the probabilities are not uniformly distributed [7]. The generalized version of the Coupon Collector's problem resembles the ORIGAMI setting. The result derived in [7] is reproduced in Theorem 1.

THEOREM 1: In the case of non-uniform probabilities, represented by the probability distribution $(p_1, p_2, \ldots, p_{|\mathcal{M}|})$, the expected value of X_d is given by

$$\mathbf{E}[X_d] = \sum_{q=0}^{q=d-1} (-1)^{d-1-q} \binom{u-q-1}{u-d} \sum_{|J|=q} \frac{1}{1-P_J} \tag{4}$$

where, $P_J = \sum_{j\in J} p_j$ ■.

The above expression in Eq. 4 reaches its minimum with respect to the probability values when the term inside the second summation reaches its minimum value, which occurs when all the probabilities are equal.[2] The outer summation has d terms. Each term in the outer summation contributes to the number of attempts (on an average) required to obtain the q^{th} unique coupon.

The above analysis has the following intuitive explanation. Note that the maximal patterns in a dataset are of different sizes. Consider that the smallest and the largest maximal pattern have s and l edges, respectively. To simplify the analysis, consider that all the edges have the same probability, (say μ), of being selected at any time. Hence, the probability that a graph of length s and l will be generated is of the order of μ^s and μ^l. Note that although there are more valid edge-sequences for larger graphs, for sparse larger graphs the count of the edge-sequences does not grow exponentially with number of edges. On the other hand, for a clique of the same size, the number of valid edge sequences does grow exponentially. The convergence rate is inversely proportional to the range of the probability values, i.e. $P_{range} = \mu^l - \mu^s$. Section 6.3 confirms this observation empirically.

5. MINING ORTHOGONAL REPRESENTATIVE SETS

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

[2] $\frac{1}{1-x_1} + \frac{1}{1-x_2}$ is minimum when, $x_1, x_2 = \frac{1}{2}$.

THEOREM 2: 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 conditions 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 pairwise 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 3: Finding the optimal α-orthogonal β-representative that minimizes the residue is NP-hard.

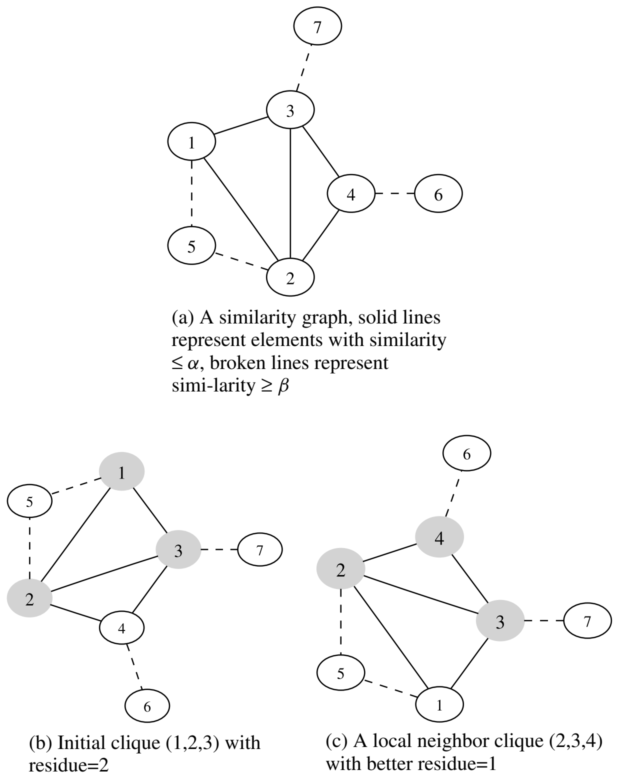
PROOF: This is easy to show, since the general problem contains an NP-hard subcase. For $\beta = 1$, each element in the α-orthogonal set represents only itself, giving the residue for any $\mathcal{R}$ as $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| = \widehat{\mathcal{M}} \setminus \mathcal{R}$. Thus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

5.1. Clique Finding

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 on the basis of the residue size.

Clique Refinement: We also designed a refinement to the above heuristic approach that 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. Figure 5 shows an example state transition for the local-optimal algorithm. In Fig. 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), which 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.

Other Refinements for Clique Finding: In the above discussion on finding a clique, we computed the similarity by using a maximal graph mining algorithm [11] that takes two graphs as input and mines patterns with 100% support. The frequent graph of maximum size is used to compute the size of the maximal common subgraph in the similarity equation.



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

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

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

Fig. 5 Local optimization example.

However, computing the exact similarity by solving the maximal common subgraph can be costly. Moreover, 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: For graphs G_1, G_2, 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: $sim_{em}(G_1, G_2) = \frac{|E_{G1} \cap E_{G2}|}{\max(|E_{G_1}|, |E_{G_2}|)}$. The following lemma always holds.

LEMMA 3: $sim_{em} \geq sim_{mc}$.

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 3, sim_{mc} is also smaller than α, and the corresponding patterns satisfy the α-orthogonal constraints. Otherwise, we compute sim_{mc}.

Earlier, a randomized approximate solution was proposed to obtain the set of α-orthogonal β-representatives. The end result of this algorithm depends on the initial random element $M \in \widehat{\mathcal{M}}$. Starting from a bad choice of the initial element can hamper the chances of obtaining a smaller residue set even with the local optimization strategy. We propose two heuristics that improve upon the existing strategy.

Heuristic 1: The first approach is based on finding a maximal clique in $\Gamma_{em}(\widehat{\mathcal{M}})$, where $\Gamma_{em}(\widehat{\mathcal{M}})$ is the graph with $V = \widehat{\mathcal{M}}$ and $E = \{(M_a, M_b) | sim_{em}(M_a, M_b) \leq \alpha\}$. $\Gamma_{mc}(\widehat{\mathcal{M}})$ can be defined similarly. Since sim_{em} is relatively cheaper to compute, generating $\Gamma_{em}(\widehat{\mathcal{M}})$ is much easier than generating $\Gamma_{mc}(\widehat{\mathcal{M}})$. The rationale behind this approach can be explained with the following lemma.

LEMMA 4: Every maximal clique $\mathcal{C}_{em}$ in $\Gamma_{em}(\widehat{\mathcal{M}})$ is a subset of some maximal clique $\mathcal{C}_{mc}$ in $\Gamma_{mc}(\widehat{\mathcal{M}})$.

PROOF: Proof follows from lemma 3. Since $sim_{em} \geq sim_{mc}$, every edge in $\Gamma_{em}(\widehat{\mathcal{M}})$ is an edge in $\Gamma_{mc}(\widehat{\mathcal{M}})$. Hence a maximal clique in $\Gamma_{em}(\widehat{\mathcal{M}})$ is a subset of some maximal clique in $\Gamma_{mc}(\widehat{\mathcal{M}})$. ■

Starting from the maximal clique obtained from Γ_{em} $(\widehat{\mathcal{M}})$,[3] nodes can be added iteratively to this maximal clique to generate the corresponding clique in $\Gamma_{mc}(\widehat{\mathcal{M}})$. The algorithm is similar to the random algorithm, except that instead of starting from a single random node, we start with the maximal clique. The performance of this heuristic depends on how well $\Gamma_{em}(\widehat{\mathcal{M}})$ approximates $\Gamma_{mc}(\widehat{\mathcal{M}})$. The better the approximation, the better the result.

Heuristic 2: The second heuristic also utilizes $\Gamma_{em}(\widehat{\mathcal{M}})$. In this case, the vertices of $\Gamma_{em}(\widehat{\mathcal{M}})$ are sorted in ascending order of degree. The vertex with the lowest degree is chosen as the initial node for the random algorithm followed by the local optimization. This heuristic is based on the assumption that the vertex in $\Gamma_{em}(\widehat{\mathcal{M}})$ with the least degree has a higher chance of being similar to a larger number of graphs in $\widehat{\mathcal{M}} \setminus \mathcal{R}$. We talk in terms of chance because $\Gamma_{em}(\widehat{\mathcal{M}})$ is a subset of $\Gamma_{mc}(\widehat{\mathcal{M}})$ and the absence of an edge in the former does not necessarily imply absence of an edge in the latter.

6. EXPERIMENTS

6.1. 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.[4] The dataset is classified into three subsets of compounds: confirmed active (CA), confirmed moderately

[3] Cliquer (http://users.tkk.fi/~pat/cliquer.html) was used to obtain maximal graphs.

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

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 $t = 7\AA$). 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 three large graphs, with an average of 2154 vertices and 81, 607 edges per graph.[5] 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.

6.2. Empirical Results

All experiments were run on a 2.75 GHz PowerPC G5 Machine with 4 GB Memory and 400 GB disk. Since ORIGAMI is randomized, we perform several runs (typically between three to five). 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 such gSpan [19], and DMTL [9] (which mine all frequent subgraphs) and SPIN [11] (which mines maximal graph patterns). The local optimization algorithm was used only in the result that compares against the random maximal clique algorithm.

6.2.1. *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 [9] 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 7 GB output (from an initial 3 MB 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 edge-sets. At $\pi^{\min} = 100\%$ this yields 90 maximal frequent graphs.

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 300 s 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. $\widehat{\mathcal{M}} = \mathcal{M}$).

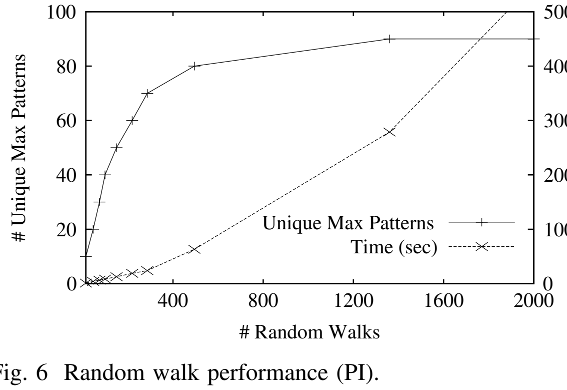


Fig. 6 Random walk performance (PI).

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

Table 1. 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

6.2.2. *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 eight 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 Fig. 7.

6.2.3. *Chemical Compound Mining*

Next we mined the chemical compound datasets. Note that neither gSpan nor SPIN was able to run on the full 40 942 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. The results are shown in Fig. 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 αs. We see that, as expected, bigger cliques are found for larger αs. Figure 8(c) shows the effect of β on the 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.

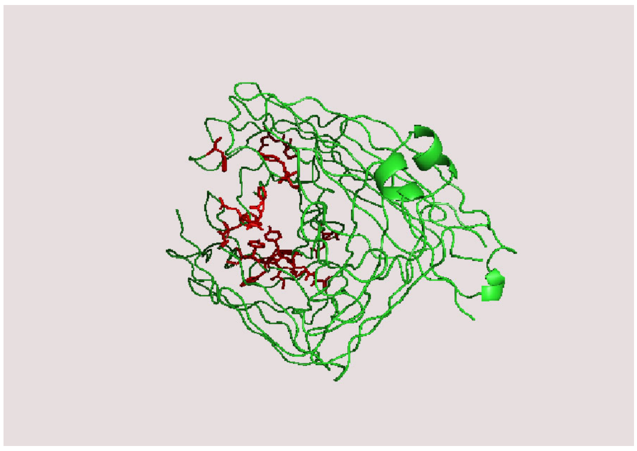


Fig. 7 Low entropy motif (in Red/Black).

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 181 s. 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 the 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.

6.2.4. *Implanted Seed Mining*

The goal of this experiment was 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 $\mathcal{R}$ was mined at $\alpha = 0.6$, which yielded a maximal clique of size $|\mathcal{R}| = 9$. These nine 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 nine 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

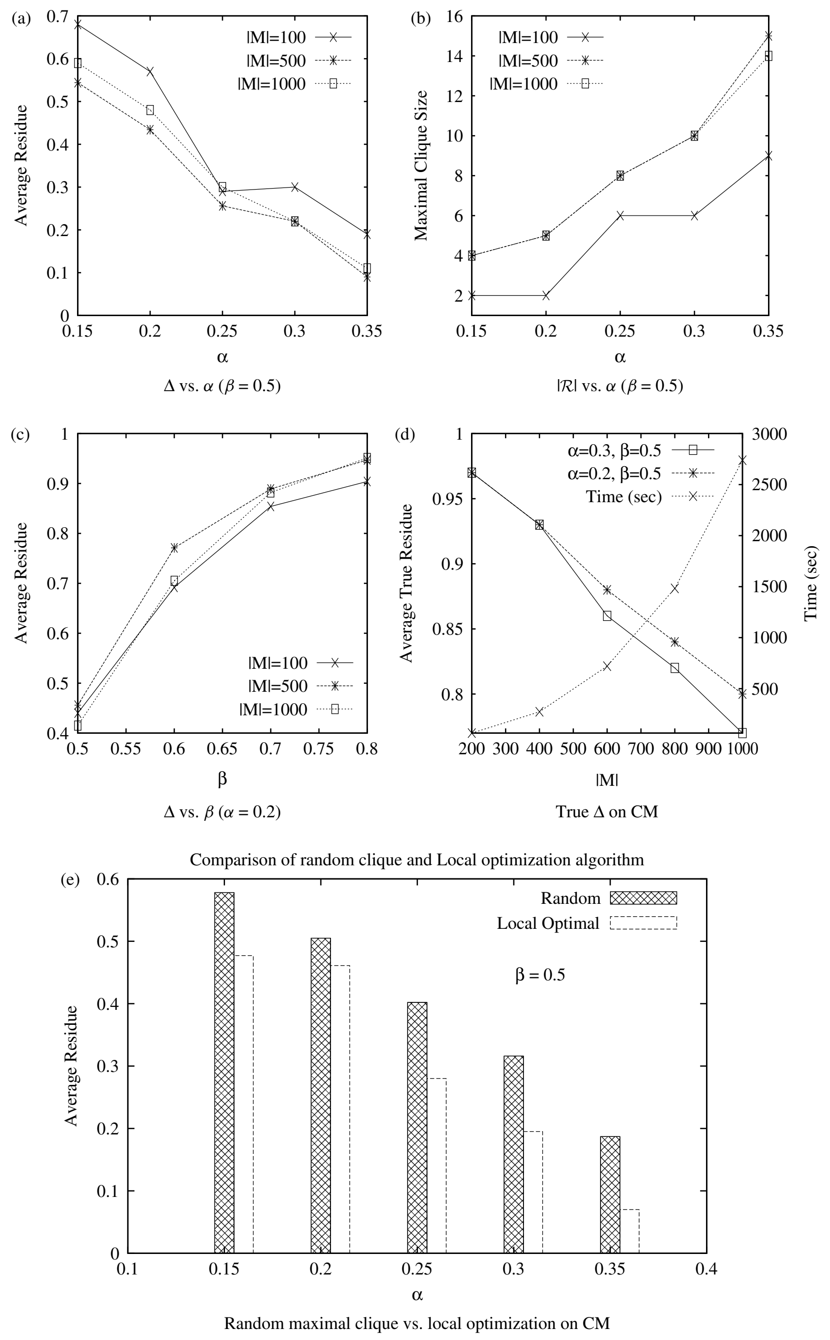


Fig. 8 Performance on DTP and CM.

size increases, the number of seeds in a graph increases, resulting in greater number of seeds being captured.

6.2.5. Scalability Results

Figure 9 shows the effect of varying dataset sizes for three datasets—DTP, Clique and Random. As the name indicates, the *Clique* dataset consists of complete graphs. The *Random* dataset consists of graphs generated under the Erdős-Renyi model. The datasets are generated such that the average number of vertices and edges per graph is nearly the same for the DTP and the Random datasets. For the Clique dataset, since it is not possible to match both the number of vertices and edges, the number of vertices was reduced to match the number of edges.

The time on the *y*-axis indicates the time taken to generate 100 maximal graphs by the random-walk-based

Table 2. Results on implanted seeds.

$\|\mathcal{D}\|$	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

graph mining algorithm. For each dataset and size, the mean execution time over 10 runs of the algorithm is shown. In order to truly understand the scalability of the algorithm, we need to suppress the effect of randomization across each run of the algorithm. This is achieved through a combination of two effects. First, larger datasets are generated by replicating copies of the smaller datasets. For instance, the 10K (10000 graphs) and 15K datasets are generated by merging two and three copies of the 5K (5000 graphs) dataset, respectively. This ensures that the same 100 maximal graphs are generated for datasets of varying sizes, provided the minimum support is kept the same. Second, the seed for the random number generator within the algorithm is kept the same for datasets of same size.

Figure 9(a) shows the mean run time for the DTP dataset along with the error bar. The two ends of the error bar represent the maximum and minimum run times over 10 executions of the algorithms, thus capturing the variation in run time with different random seeds. The figure shows that the run time increases almost linearly with the increase in the dataset size. The large difference between the maximum and minimum run times indicates that this dataset is very sensitive to the seed, which in turn points out that the maximal patterns are not uniformly distributed over the partial order. Both the Clique (Fig. 9(b)) and the Random dataset (Fig. 9(c)) show a similar linear relationship between run time and the dataset size. But both these datasets differ from the DTP dataset, in that they exhibit very little variation in run times. For the Random dataset, the variation is so small that it is hardly visible in Fig. 9(c). The small variation indicates that the maximal patterns are more uniformly distributed over the partial order, for these two datasets.

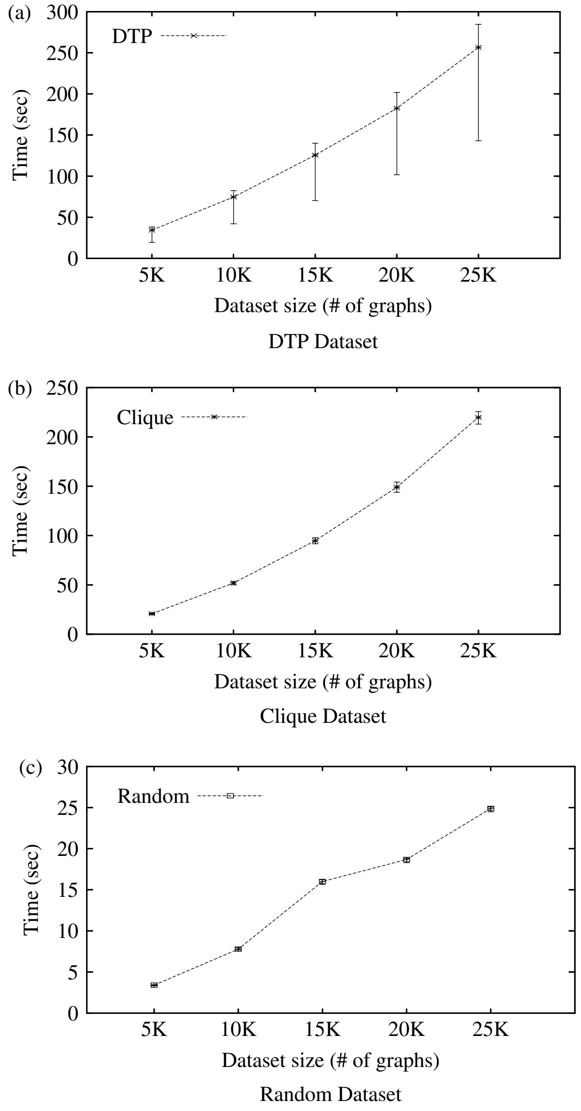
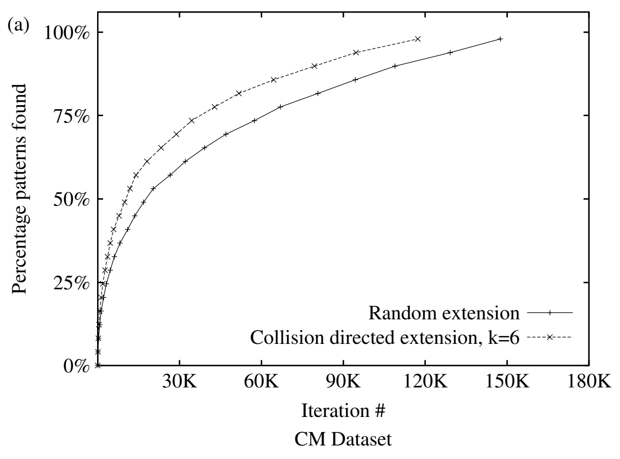


Fig. 9 Scalability on DTP, clique and random datasets.

6.3. Parameter Settings

Section 4.3 and 4.4 recommended alternatives to random walk and analyzed the expected number of random walks required to obtain a subset of the maximal graphs. In this section, we provide empirical evidence for the superior performance of the collision-directed traversal. We also highlight the influence of dataset characteristics on the number of random walks required to obtain a set of maximal graphs.

Collision-assisted Random Walk: Fig. 10(a), compares random traversal with the collision-directed traversal method for the CM dataset (described in Section 6.1). The entire set of maximal patterns $\mathcal{M}$ for this dataset (with $\pi^{\min} = 25$) is obtained from SPIN [11]. The y-axis in the plot is the percentage of $\mathcal{M}$ found upto certain number of random walks (on the x-axis). The x-axis is the count of the number of random walks along the partial order (indicated by *"Iteration #"* in the figure). Two metrics are used to compare the plain vanilla random walk with the collision-directed random walk. The first metric measures the number



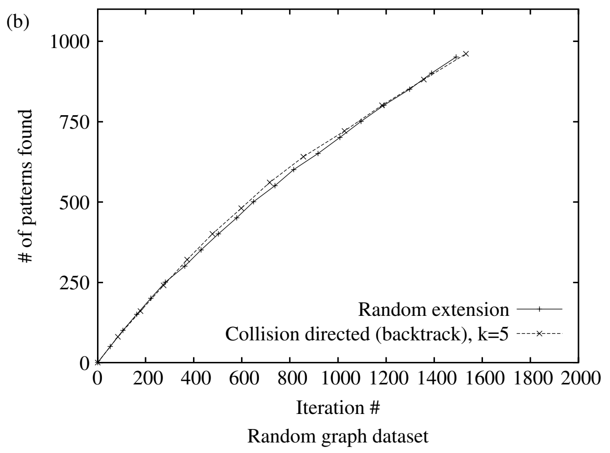


Fig. 10 Comparison between random extension and collision-directed extension.

of iterations required to obtain a certain fraction of maximal patterns. The number of iterations corresponds directly to the time required to generate the set of maximal graphs. As can be seen in Fig. 10(a), on the CM dataset, the collision-directed extension approach has a 25%–45% improvement in terms of the number of iterations needed to find the same number of maximal patterns. The CM dataset represents chemical compounds which are sparse; as a result the maximal graphs obtained are sparse, too. As noted in Section 4.4, sparse datasets have a larger P_{range} resulting is a faster convergence. On the other hand, Fig. 10(b) compares the two traversal techniques for a dataset of random graphs, generated under the Erdős-Renyi model. For such a dataset, the performance of random traversal is almost identical to the collision-directed approach. Notice that for this dataset, almost every random walk results in a unique maximal graph. This can be attributed to the uniform distribution of the maximal graphs over the partial order. For the same reason, the collision-directed approach is unable to perform better. For this dataset, SPIN could not generate the entire set of maximal patterns $\widehat{\mathcal{M}}$; as a result the plot for only the first 1000 maximal graphs is shown.

The second metric for evaluating the traversals compares the fraction of $\mathcal{M}$ covered by the $\widehat{\mathcal{M}}$ generated by each of the two traversals—henceforth referred to as $\widehat{\mathcal{M}}_{random}$ for the random traversal and $\widehat{\mathcal{M}}_{collision}$ for the collision-directed traversal. The premise behind this metric is that an efficient collision-directed traversal has larger $\frac{\Upsilon(\widehat{\mathcal{M}}_{collision},\mathcal{M})}{|\mathcal{M}|}$ ratio as compared to $\frac{\Upsilon(\widehat{\mathcal{M}}_{random},\mathcal{M})}{|\mathcal{M}|}$. To remind the reader, Υ is the set of represented graphs, i.e. those that have a similarity at least β with some graph in $\mathcal{R}$. We chose to set $\beta = 1 - \alpha$, with $\alpha = 0.3$. The *coverage ratio* for varying $|\widehat{\mathcal{M}}|$ is computed, where coverage ratio is given by the expression $\frac{|\Upsilon(\widehat{\mathcal{M}}_{random},\mathcal{M})|}{|\Upsilon(\widehat{\mathcal{M}}_{collision},\mathcal{M})|}$. Table 3 shows the comparison of the two traversals for the CM dataset.

The coverage ratio in Table 3 shows that for a smaller representative set the collision-directed traversal outperforms random traversal by more than 40%, although as the representative set size increases the gap decreases.

Convergence Rate: The intuitive argument for the rate of convergence provided in Section 4.4 is validated in the following discussion, by showing the convergence rate on different datasets, where their maximal frequent graphs have different range of generation probabilities.

In Fig. 11 we compare two different types of datasets for their convergence. The first is the CM dataset and the other is a synthetic dataset (referred to as the clique dataset) that is built by converting an itemset dataset into a graph dataset. An itemset dataset is converted into a graph dataset by representing each itemset as a clique with each item as a vertex of the clique. Note that all the maximal frequent graphs for the clique dataset are cliques and there is a one-to-one correspondence between the maximal itemsets mined on the itemset dataset and the maximal cliques. We also ensured that both the CM dataset and the clique dataset have an almost equal number of maximal graph patterns. However, the generation probability of each maximal pattern in the clique dataset is almost uniform, as the opposing tendencies in Eqs 2 and 3 cancel out. As a result, the range of the probability values, P_{range}, for the clique dataset is much smaller than the CM dataset. In the figure, the label on the y-axis side corresponds to the clique dataset and the right corresponds to the CM dataset. We can see that in 15 k iterations, around 98% of the maximal

Table 3. Coverage Ratio on the CM dataset for varying $|\widehat{\mathcal{M}}|$.

$\mid\widehat{\mathcal{M}}\mid$ (s)	Coverage Ratio
100	1.42
200	1.2
300	1.1

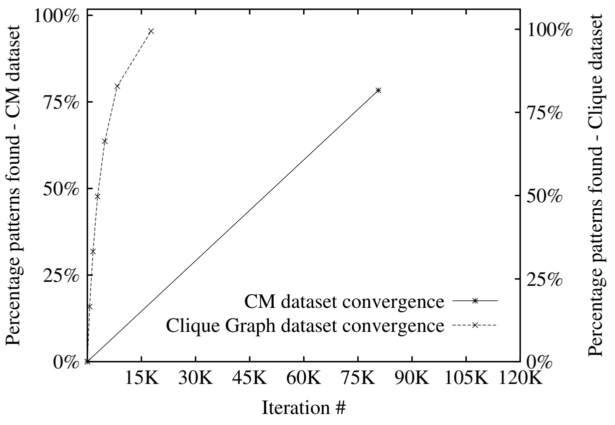


Fig. 11 Comparison between the convergence between a clique dataset and the CM dataset.

frequent graphs have been obtained for the clique dataset, whereas even in 90K iterations only 75% of the maximal graphs were obtained for the CM dataset.

7. REPRESENTATIVE SET MINING: DISCUSSION

The second step of ORIGAMI that uses a local-optimal algorithm to obtain an α-orthogonal β-representative set ensures strict orthogonality (two patterns are not more than α similar) and a locally optimal representativeness (residue patterns are at least β similar to a representative pattern). Interestingly, these two properties are contradicting criteria; orthogonality restricts the set $\mathcal{R}$ from growing, while representativeness encourages it to grow. In absence of β constraint (or setting $\beta = 0$), any clique of Theorem 2 is an optimal solution. In the absence of the α constraint, the entire $\widehat{\mathcal{M}}$ set is an optimal solution. For other choices of α and β, it may not be feasible to obtain $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| = 0$; so, we seek the best representative set that we can obtain, while enforcing strict orthogonality. This is exactly how ORIGAMI works. Since, no efficient global optimization algorithm exists, it minimizes the residue set by a local optimal algorithm. Several variations of ORIGAMI can be obtained by trading off orthogonality upto a certain extent for better representativeness.

Orthogonality as constraints with penalty: Representatives can be obtained by solving a typical data clustering problem, where the cluster centers can be taken as member of $\mathcal{R}$. However, for a user-defined β, we also want that the farthest element of a cluster to be at least β similar to its center so that $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| = 0$. To satisfy this, an appropriately large k should be chosen; otherwise the problem will be infeasible. Thus far, this formulation does not offer any orthogonality, to enforce that we can apply *cannot-link* constraints among the cluster centers if the similarity between them is greater than α. Thus, a constraint K-means clustering formulation similar to one proposed in [6] can be adopted as a variation of our approach. However, this formulation has three parameters: β, k, and a penalty value for not satisfying a cannot-link constraint. Also it does not enforce strict orthogonality, it may violate some cannot-link constraints. Furthermore, it will only produce local optimal solution. In fact, the authors in [6] have proved that feasibility problem for K-means clustering with cannot-link constraints is NP-complete.

Only β constraints: In this formulation, we have no α constraint (or $\alpha = 1$) and we want complete representativeness for some β. As mentioned before, an immediate solution would be to take the entire $\widehat{\mathcal{M}}$ set as the representative set. However, that is a trivial solution. For the purpose of summarization, we need to introduce another constraint that will control the summary set size. For instance, we can choose to have at most k representatives. However, feasibility of this problem for arbitrary value of β and k is not guaranteed; in fact, the following theorem holds:

THEOREM 4: Given $\alpha = 1$, the problem whether for an arbitrary β there exists a k such that $|\mathcal{R}| \le k$, with $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| = 0$, is NP-complete.

PROOF: In graph theory, a *dominating set* for a graph is a subset V' of the vertices of the graph such that each vertex not in V' has an edge to at least one member of V'. Given a graph and a positive integer k, the *dominating set problem* asks whether the graph has a dominating set of size k or less. We can reduce a dominating set problem (known NP-complete problem [8]) to the problem in Theorem 4. For a given graph $G(V, E)$, assume that each vertex $v \in V$ represents a pattern in $\widehat{\mathcal{M}}$, so $|V| = |\widehat{\mathcal{M}}|$. For every edge $(v_1, v_2) \in E$ we consider that the similarity between the corresponding pattern is greater than β, i.e. v_1 is a β-representative for v_2 and vice versa. Now, there exist a dominating set, R of size k or less in G if and only if, we have a representative set, $\mathcal{R}$, of size k or less such that the $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| = 0$. For any $v \in R$, the corresponding pattern in $\widehat{\mathcal{M}}$ is a representative pattern, because for any $v \notin R$, there exists a $u \in R$ adjacent to it. Correspondingly, for any pattern $q \in \widehat{\mathcal{M}} \setminus R$, there exists a pattern $p \in \mathcal{R}$ such that $sim(p, q) \ge \beta$. Hence, every pattern in $\widehat{\mathcal{M}}$ is either in the representative set or is covered by at least one representative, so we have $|\Delta(\mathcal{R}, \widehat{\mathcal{M}})| = 0$. Since, $\alpha = 1$, there is no restriction how we choose the elements for the set $\mathcal{R}$. ■

Thus, finding an optimal k is NP-hard and no efficient algorithm exists to solve this problem. This formulation has two parameters, β and k, and there is absolutely no guarantee regarding orthogonality.

Only α constraints: In this formulation, we only care about strict orthogonality, and no β constraint exists (or

$\beta = 1$). However, to obtain the best representative set, we like to obtain the largest summary set that respects the orthogonality. In the proof of Theorem 3 we have already shown that obtaining the optimal set for such a formulation is NP-Hard.

7.1. Experimental Comparison

On the basis of the discussion above, we can see that finding the optimal α-orthogonal β-representative set is very difficult; even different restrictive versions of this formulation lead to NP-Hard problems and they all have to be solved with some local optimization or approximation algorithms. The local optimization formulation of ORIGAMI tries to respect both orthogonality and representativeness. We experimentally compare the performances of our algorithm with a standard k-means algorithm, to show that our approach is considerably better. CM dataset is used for this comparison and size of $\widehat{\mathcal{M}}$ is fixed at 1000. For the sake of valid comparison we optimize a different performance measure *avgsim* (Average Similarity), since the size of residue ($|\Delta|$) is meaningless for the k-means formulation. *avgsim* is computed by finding the average similarity of a pattern to its representative. In k-means, each pattern belongs to some cluster, hence it is covered by the corresponding cluster representative. The β parameter is not required, and hence ignored for this performance metric. Now, for k-means we need to find a value for k. To provide the same k value for both our algorithm and the k-means, we first run our local optimal algorithm for some value of α, and from its result we record the size of the representative set and use that for k.

Figure 12(a) shows the *avgsim* value for different αs. Note that as α is increased, the non-redundancy constraint is relaxed and more patterns are inserted in the representative set, and naturally the value of *avgsim* increases for both the algorithms. We used the size of the clique from our algorithm as the value of k. So, for the k-means graph in 12(a) is showing the *avgsim* value for those k values. In Fig. 12(b) we show the clique sizes (i.e. the different k values) that are obtained for those different α values. The value of *avgsim* for k-means is better for smaller α, which is expected, since k-means has no constraints to choose the members of the $\mathcal{R}$ set, while our algorithm can only choose as members of the $\mathcal{R}$ those graphs that respect the pairwise α-orthogonality constraints. For higher α values, the orthogonality constraint is relaxed and our algorithm performs as good as k-means. In Fig. 12(b), we also plot the orthogonality of the k-means approach. *Orthogonality* is defined as the fraction of pairwise similarity among the k-representatives that satisfy the α constraint. It is evident that for low-α k-means shows very poor orthogonality. For example, for $\alpha = 0.25$ only 40% of the pairwise orthogonality constraints are honored out of $\binom{k}{2}$ constraints. Thus k-means is just trading off orthogonality to obtain better representativeness. But, our algorithm is guaranteed to be alpha-orthogonal (orthogonality=1) for all values of α. From Figs 12(a) and 12(b), we can see that for reasonable values of α, ORIGAMI guarantees orthogonality, and yet has a good average similarity score.

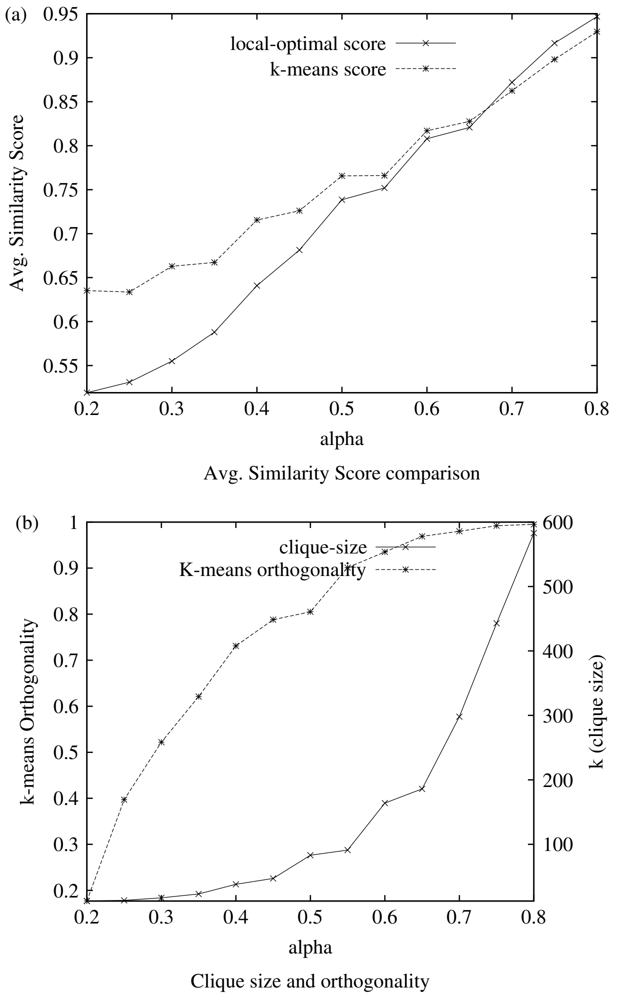


Fig. 12 Comparison between k-means and Local Optimal algorithm.

8. 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 also discuss the convergence properties, both theoretically and empirically. 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.

ACKNOWLEDGEMENTS

This work was supported in part by NSF CAREER Award IIS-0092978, and NSF grants EIA-0103708 and EMT-0432098.

REFERENCES

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

[2] H. Bunke, On a relation between graph edit distance and maximum common subgraph, Pattern Recognit Lett 18(9) (1997), 689–694.

[3] H. Bunke and K. Shearer, A graph distance metric based on the maximal common subgraph, Pattern Recognit Lett 19 (1998), 255–259.

[4] 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.

[5] D. Cook and L. Holder, Substructure discovery using minimal description length and background knowledge, J Artif Intell Res 1 (1994), 231–255.

[6] I. Davidson and S. S. Ravi, Clustering with constraints: feasibility issues and the $k-$means algorithm, In SIAM SDM Proceedings, Newport Beach, CA, 2005.

[7] P. Flajolet, D. Gardy, and L. Thimonier, Birthday paradox, coupon collectors, caching algorithms and self-organizing search, Discrete Appl Math 39(3) (1992), 207–229.

[8] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, New York, W. H. Freeman & Co., 1979.

[9] 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), San Diego, CA, 2005.

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

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

[12] A. Inokuchi, T. Washio, and H. Motoda, Complete mining of frequent patterns from graphs: mining graph data, Mach Learn 50(3) (2003), 321–354.

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

[14] R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, UK, 1995.

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

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

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

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

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

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

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