International Journal of Computer and Communication Technology

Volume 1 | Issue 3 Article 5

July 2010

g-FSG Approach for Finding Frequent Sub Graph

Sadhana Priyadarshini
Department of Computer Applications ITER, SOA University, Bhubaneswar, ODISSA,
sadhana_pridarshini@rediffmail.com

Debahuti Mishra
Institute of Technical Education and Research, Siksha O Anusandhan University, Bhubaneswar, Odisha, India, debahuti@iter.ac.in

g-FSG Approach for Finding Frequent Sub Graph

Sadhana Priyadarshini
Department of Computer Applications
ITER, SOA University,
Bhubaneswar, ODISSA
sadhana_pridarshini@rediffmail.com

Debahuti Mishra
Department of Computer Sc. & Engineering
ITER, SOA University,
Bhubaneswar, ODISSA
debahuti@iter.ac.in

Abstract— **Informally, a graph is set of nodes, pairs of which might be connected by edges. In a wide array of disciplines, data can be intuitively cast into this format. For example, computer networks consist of routers/computers (nodes) and the links (edges) between them. Social networks consist of individuals and their interconnections (which could be business relationships or kinship or trust, etc.) Protein interaction networks link proteins which must work together to perform some particular biological function. Ecological food webs link species with predator-prey relationships. In these and many other fields, graphs are seemingly ubiquitous. The problems of detecting abnormalities (outliers) in a given graph and of generating synthetic but realistic graphs have received considerable attention recently. Both are tightly coupled to the problem of finding the distinguishing characteristics of real-world graphs, that is, the patterns that show up frequently in such graphs and can thus be considered as marks of realism. A good generator will create graphs which match these patterns. In this paper we present gFSG, a computationally efficient algorithm for finding frequent patterns corresponding to geometric sub graphs in a large collection of geometric graphs. gFSG is able to discover geometric sub graphs that can be rotation, scaling, and translation invariant, and it can accommodate inherent errors on the coordinates of the vertices.**

Keywords-Frequent Sub graph; Graph Isomorphisim; Geometric dscriptors

INTRODUCTION

Over the years, these frequent patterns have been used extensively to discover association rules, to extract prevalent patterns that exist in the data sets, and to build effective clustering and classification algorithms [4]. Nevertheless, as data mining techniques have been increasingly applied to non-traditional domains, such as scientific, spatial and relational data sets, situations tend to occur in which we can't apply existing item set discovery algorithms because these problems are difficult to be adequately and correctly modeled with the traditional market-basket transaction approaches [5].

RELATED WORK

Discovery of frequent patterns has been studied in a variety of data mining settings. In its simplest form, known from association rule mining, the task is to discover all frequent item sets[1-3]. The fundamental task of association rule and frequent set discovery has been extended in various directions, allowing more useful patterns to be discovered with special purpose algorithms.[4-6] .The graph isomorphism problem In graph theory, an isomorphism of graphs G and H is a bisections between the vertex sets of G and H such that any two vertices u and v of G are adjacent in G if and only if f(u) and f(v) are adjacent in H. This kind of bisections commonly called "edge-preserving bisections", in accordance with the general notion of isomorphism being structure-preserving bisections [8-10].

GOAL OF PAPER

This paper focuses on the problem of finding frequently occurring geometric patterns in geometric graphs-graphs whose vertices have 2-D or 3-D coordinates associated with them. These patterns correspond to geometric sub graphs that are embedded in a sufficiently large number of graphs[6]. Data sets arising in many scientific domains often contain such geometric information and any patterns discovered in them are of interest if they preserve both the topological and the geometric nature of the pattern. Here we present an algorithm called gFSG that is capable of finding frequently occurring geometric sub graphs in a large database of graph transactions[5].The key characteristic of gFSG is that it allows for the discovery of geometric sub graphs that can be rotation, scaling and translation invariant. Furthermore, to accommodate inherent errors on the coordinates of the vertices (either due to experimental measurements or floating point round-off errors), it allows for patterns in which the coordinates can match with some degree of tolerance. gFSG uses a pattern discovery framework, which follows the level-by-level approach made popular by the Apriori[1] algorithm, and incorporate numerous computationally efficient algorithms for (i) computing isomorphism between geometric subgraphs that are rotation, scaling and translation invariant, (ii) candidate generation, and

(iii) frequency counting[3-6]. In addition, gFSG incorporates an iterative pattern shape optimization algorithm whose goal is to identify the geometric shape of patterns that lead to the highest support

G-FSG—FREQUENT GEOMETRIC SUBGRAPH DISCOVERY ALGORITHM

The gFSG was designed to operate on a database of geometric graphs (either 2D or 3D) and find all sub graphs accordingly FSG follows the level-by-level structure of the Apriori algorithm [1] and shares many characteristics with the previously developed frequent sub graph discovery algorithm for topological graphs[10]. The motivation behind this choice is the fact that the level-by-level structure of Apriori requires the smallest number of sub graph isomorphism computations during frequency counting, as it allows it to take full advantage of the downward closed property of the minimum support constraint and achieves the highest amount of pruning when compared with the most recently developed depth first-based approaches such as dEclat [3], Tree Projection [2], and FP-growth [4]. In fact, despite the extra overhead due to candidate generation that is incurred by the level-by-level approach, recent studies have shown that because of its effective pruning, it achieves comparable performance with that achieved by the various depth-first-based approaches, as long as the data set is not dense or the support value is not extremely small. At the same time, the relatively simple algorithmic structure of this approach allows us to focus on the non-trivial aspects of operating on geometric graphs.

The gFSG provides two basic approaches for constructing the shape of the frequent geometric pattern. The first approach uses an arbitrarily selected embedding of a graph as its representative geometric shape, whereas the second approach employs an iterative shape optimization phase that tries to greedily select as its representative, a geometric shape that will maximize the frequency of the corresponding sub graph. These two approaches provide different performance-coverage trade-offs. The first approach is faster but it may fail to identify some of the frequent sub graphs, whereas the second approach is somewhat slower, but in general, finds a larger number of frequent geometric sub graphs [7]. However, regardless of the method, due to their inherently heuristic nature, both of them may miss some of the patterns, especially as the value of r increases.

TABLE-I: The notation used in the algorithm

Notation	Notation
D	A data set of graph transactions
t	A graph transaction in D
k-(sub) graph	A (sub)graph with k edges
g^k	A k-sub graph
V(g)	A set of vertices of graph g
E(g)	A set of edges of graph g
C^k	A set of candidates with k edges
F^k	A set of frequent k-sub graph

Algorithm :GFSG(D,s,adjust_type,N)
F^1,F^2,F^3 ← all frequent geometric subgraphs of size 1,2,3 in D
K←4
while $F^{k-1}\neq\Phi$ do
for each candidate $g^k \in C^k$ do
$g^k.S$ ← COUNT_FREQUENCY(g^k,D)
if $|g^k.S| < s$ then
continue;
if adjust type ≠ None then
ADJUST_SHAPE ($g^{k,}$, D, adjust-type, N)
F^k ← {$g^k \in C^k | g^k.count > sD$}
K ← k +1
return $F^1,F^2,F^3.....F^{k-2}$
COUNT_FREQUENCY (g^k,D)
S ← Φ
for each transaction t ∈ D do
If candidate g^k is included in t then
S ← S U { t }
return S
ADJUST_SHAPE ($g^{k,}$, D, adjust-type, N)
for i=1 ..N do
compute the average of vertex coordinates of $g^k.S$.
if adjust-type = Simple then
$g^k.S$ ← COUNT_FREQUENCY (g^k,D)
else if adjust-type = Support then
S' ← COUNT_FREQUENCY (g^k,D)
If S' = $g^k.S$ then
return $g^k.S$ ← S'
else if adjust-type = DWC then
if g^k fails the downward closure check then
return $g^k.S$ ← COUNT_FREQUENCY (g^k,D)

A. Geometric graph and sub graph isomorphism

Geometric graph isomorphism

In principle, a geometric isomorphism between two graphs g1 and g2 can be computed in two different ways. First, we can identify all topological isomorphism between g1 and g2, and then check each one of them to determine whether or not there is an allowable homogeneous geometric transformation that brings the corresponding vertices of the two graphs within an r distance from each other (where r is the coordinate matching tolerance). Alternatively, we can first identify the possible set of geometric transformations that map the vertices of g1 within an r distance of the vertices of g2, and then check each one of them to see if it preserves the topology (and the vertex and edge labels) of the two graphs.

Transform and map approach

Each geometric graph has its own coordinate system or reference frame. When we check the geometric isomorphism between g1 and g2, both should be in the same coordinate system. However, there can be infinitely many geometric configurations, especially when we consider rotation invariant isomorphism. Our algorithm limits this number by using a

subset of the edges of the graph to define the coordinate axes. In the two-dimensional space, it suffices to choose an edge and its direction to determine a local coordinate system and in the three-dimensional space, two connected non-collinear edges .

Using topological and geometric descriptors to speed up the computations

To further reduce the overall time spent in checking whether two graphs are geometrically isomorphic or not, g-FSG computes various descriptors that capture certain topological properties and geometric transform invariants. Geometric transform invariants are certain quantities computed from a geometric graph that remain the same no matter how the original graph is rotated, scaled, or translated[7]. The key idea behind this approach is to use these descriptors to quickly eliminate pairs of graphs that cannot be isomorphic to each other by simply checking to see whether their respective descriptors match or not. Since both the topological properties and the geometric transform invariants remain the same regardless of the principal configuration of a particular graph, these descriptors need to be computed only once[9-10].

B. Generating size one, two, and three frequent sub graphs

The first step in g-FSG is to determine the frequent size one, two, and three r-tolerant geometric sub graphs using a direct enumeration approach. This is done primarily for two reasons. First, direct enumeration, if done efficiently, can significantly reduce the amount of time required to find size two and three sub graphs over an approach that uses the general candidate-generation and frequency-counting framework[5].This is consistent with observations of previous studies performed in the context of frequent item sets and sequences. Second, g-FSG's candidate generation scheme obtains a candidate (k +1) sub graph g^{k+1} by joining two distinct frequent sub graphs g_i^{k} and g_j^{k} that share a common (k-1)-sub graph h^{k-1}. To uniquely determine the geometry of the candidate subgraph,h^{k-1} must represent a rigid body. Thus, h^{k-1} must have two non-collinear vertices for two-dimensional graphs, and three non-collinear vertices for 3D graphs [3-5]. Consequently, the size k sub graphs that are joined should at least of size two or three for two and three-dimensional, respectively.

C. Candidate Generation

Candidate geometric sub graphs of size k+ 1 are generated by joining two frequent geometric k-sub graphs. In order for two such frequent k-sub graphs to be eligible for joining they must contain the same geometric (k-1)-sub graph. We will refer to this common geometric (k-1) sub graph among two k-frequent sub graphs as their core .

D. Frequency counting

Once candidate sub graphs have been generated, g-FSG computes their frequency. In the on text of the original Apriori algorithm, the frequency counting is performed efficiently by storing the candidate item sets in a hash-tree data structure and then scanning each transaction to determine which of the item sets in the hash-tree it supports[6]. However, developing such an algorithm for frequent sub graphs is challenging (if not impossible) because there is no natural way to build a hash tree-like structure for graphs. For this reason, g-FSG's frequency counting approach considers one candidate sub graph at-a-time and tries to determine the transactions that it is contained in.

EXPERIMENTAL EVALUATION

The Sample isomorphic graphs are given below

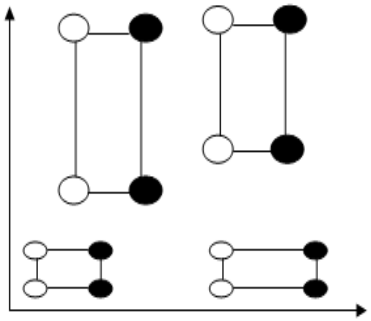


Figure 1: A triangle and its geometric configuration under rotation and translation.

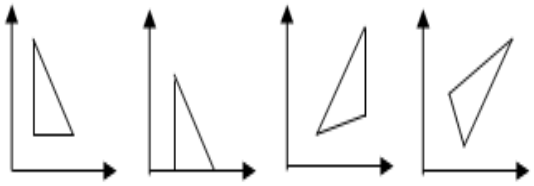
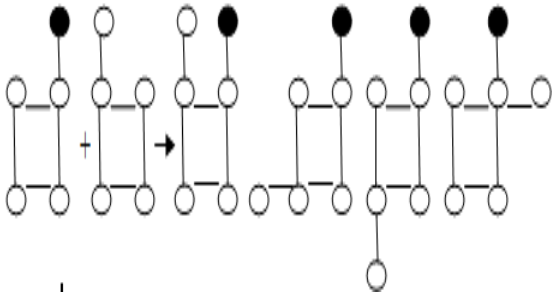


Figure 2: The core- a square of 4 vertices – has more than one auto morphism that result four in deferent candidates of size 6.



CONCLUSION AND FUTURE WORK

In this paper we presented an algorithm, gFSG, for finding frequently occurring in scientific, spatial, and relational data sets. These patterns can correspond to either exact occurrences or occurrences that are translation, rotation, and/or scaling invariant, and can accommodate a user-specified tolerance on how the coordinates of the various vertices are matched. In addition, we presented an iterative shape refinement framework that makes it possible to optimize the discovered patterns; thus, increasing their frequency and the number of patterns that get discovered. Our experimental evaluation showed that g-FSG can scale reasonably well to very large graph databases provided that graphs contain sufficiently many different labels of edges and vertices.

REFERNCES

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: J.B. Bocca, M. Jarke, C. Zaniolo (Eds.), Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann, Los Altos, CA, 1994, pp. 487–499.

[1] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Dallas, TX, 2000, pp. 1–12.

[2] X. Yan, J. Han, gSpan: graph-based substructure pattern mining, in: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM), 2002, pp. 721–724.

[3] X. Yan, J. Han, CloseGraph: mining closed frequent graph patterns, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003), 2003, pp. 286–295.

[4] J. Huan, W. Wang, J. Prins, Efficient mining of frequent subgraph in the presence of isomorphism, in: Proceedings of the 2003 IEEE International Conference on Data Mining (ICDM'03), 2003, pp. 549–552.

[5] X. Wang, J.T.L. Wang, D. Shasha, B.A. Shapiro, I. Rigoutsos, K. Zhang, Finding patterns in three dimensional graphs: algorithms and applications to scientific data mining, IEEE Trans. Knowl. Data Eng. 14 (4) (2002) 731–749.

[6] M.J. Zaki, Scalable algorithms for association mining, IEEE Trans. Knowl. Data Eng. 12 (2) (2000) 372–390.

[7] G. Cong, L. Yi, B. Liu, K. Wang, Discovrering frequent substructures from hierarchical semi-structured data, in: Proceedings of the Second SIAM International Conference on Data Mining (SDM-2002), 2002.

[8] K. Wang, H. Liu, Discovering structural association of semistructured data, IEEE Trans. Knowl. Data Eng. 12 (2000) 353–371.

[10] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, S. Arikawa, Efficient substructure discovery from large semistructured data, in: Proceedings of the Second SIAM International Conference on Data Mining (SDM'02),

[11] Sadhana Priyadarshini, ,Debahuti Mishra, An Insight into Graph Database, Proceeding of National Conference on Computational Biology,NCCB-09,(2009)Page 68-71.

[12] Sadhana Priyadarshini , Debahuti Mishra, An approach to Graph Mining using Apriori Algorithm Proceeding of First International Conference on Advance computing and Communication,ICACC-2010,Page 69-72.