Efficient Mining of Frequent Rooted Continuous Directed Subgraphs

Sreenivasa G J [1]
[1] *Dept of Mathematical and Computational Sciences*
Sreenivasa_nitk@rediffmail.com

Ananthanarayana V S [2]
[2] *Dept of Information Technology*
anvs@nitk.ac.in

National Institute of Technology Karnataka Surathkal, Srinivas nagar (post), Karnataka, INDIA, Pin Code- 575025

Abstract

Mining frequent rooted continuous directed (RCD) subgraphs is very useful in Web usage mining domain. We formulate the problem of mining RCD subgraphs in a database of rooted labeled continuous directed graphs. We propose a novel approach of merging **like** RCD subgraphs. This approach builds a Pattern Super Graph (**PSG**) structure .This PSG is a compact structure and ideal for extracting frequent patterns in the form of RCD subgraphs. The PSG based mine avoids costly, repeated database scans and there is no generation of candidates. Results obtained are appreciating the approach proposed.

1. Introduction

Frequent Structure Mining (FSM) refers to an important class of exploratory mining tasks, namely those dealing with extracting patterns in massive databases representing complex interactions between entities. FSM is not only encompasses mining techniques like associations [1] and sequences [2], but it also generalizes to more complex patterns like frequent trees and undirected graphs [3, 4]. Such patterns typically arise in applications like bioinformatics, web mining, and so on. As one increases the complexity of the structures to be discovered, one extracts more informative patterns.

Consider the web usage mining (WUM) [5] problem. Given a database of web access logs at a popular site, one can perform several mining tasks. The simplest is to ignore all link information from the logs, and to mine only the frequent sets of pages accessed by users. The next step can be to form for each user the sequence of links they followed, and to mine the most frequent user access paths (set of pages and their order). It is possible to look at the entire forward accesses of a user, and to mine the most frequently accessed subtrees at that site [6].

Web user's behavior pertaining to a web log session is as shown in Figure 1 (b) and (c) is represented in graphs. Here each node represents web page and edge represents link between source page and destination page. Many of the web usage mining algorithms till now formulated log data in the form of trees by eliminating cycles in the graphs.

But user sessions of the log data forms graph. For simplicity or in order to reduce the complexity of the structure they form trees from graphs. Suppose we convert Figure 1 (b) and (c) to tree by removing loops (or cycles) then both gives same output, as shown in Figure 1 (a). Cycle's information is very much useful in understanding how web users are taking transition from one set of pages to another set of pages. From this one can know where most of the users are leaving the site and accordingly one can restructure the site.

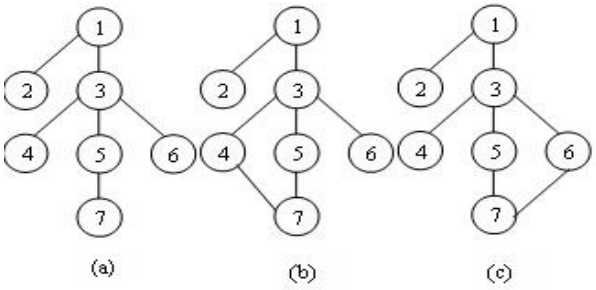

Figure 1: Tree and Graphs

In MoFa [7], gSpan [8], FFSM [9], and Gaston [10] undirected graph mining algorithms are discussed as a solution to web usage mining or bioinformatics.

Directed graphs are having explicit information than undirected graphs. So directed graphs are useful for analyzing web user traversal patterns. In Figure 2 (a), we have an edge between vertex 2 and 5. But we do not know whether it represents an edge from node 2 to 5 or from node 5 to 2 or both. Figure 2 (b) and (c) represents two different user behaviors with respect to directed graph mining. But with respect to undirected graph mining both Figure 2 (b) and (c) are treated as same and they are equivalent to Figure 2 (a). Directed graphs helps in web site personalization and caching of next possible pages.

Web users' complete behavior can be captured by directed graph. Frequent path discovery among web pages visited by users helps in making strong recommendations for how to retain and increase the visitors.

The changes in gSpan to handle directed graph mining are discussed in [11]. This is the most recent algorithm in directed graph mining. But it involves costly operations like DFS-code

1-4244-0716-8/06/$20.00 ©2006 IEEE.

extension, code adjustment and child generation for directed graph along with usual gSpan operations.

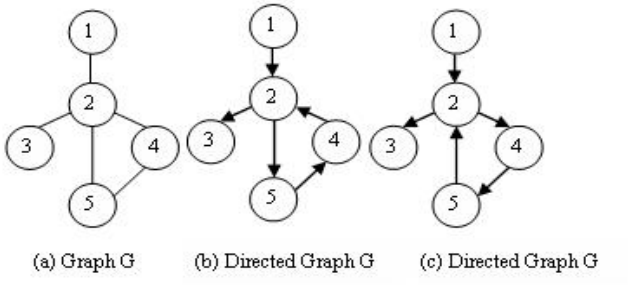

Figure 2: Graph and Directed Graphs

Till now most of researchers have concentrated on directed graph mining. But we are focusing on RCD graph mining. Rooted Continuous Directed (RCD) graph is an ideal data structure to capture the information in user web log sessions with respect to an organization. This is because each user has to start from the home page of an organization (i.e. root) and there is a continuity in terms of visiting next page with respect to the current web page.

In this paper, we propose a RCD graph mining algorithm which avoids operations like DFS-code extension, code adjustment and child generation for directed subgraph mining. These operations are more costlier than simple arithmetic count increment operations which is the core part of our RCD graph mining algorithm.

RCD graph mining approach involves merging 'like subgraphs' together and properly maintaining their importance in the form of a 'count' which is used for extracting frequent patterns. This activity involves a single scan of database.

The major contributions of this paper are as follows: (i) A novel and compact data structure called Pattern Super Graph (PSG) is constructed, which is an extended graph structure for storing crucial, quantitative information about patterns. In this paper, we have discussed how this structure is used for efficient mining of web logs. (ii) A notion and importance of frequent RCD subgraph is introduced and efficient mining of such subgraphs are discussed. (iii) We formulated a modified version of string encoding format proposed by Zaki [6] to represent the RCD graph which is space-efficient.

The rest of the paper is organized as follows. Notion of frequent RCD subgraphs with respect to RCD graph is discussed in section 2. Section 3 deals RCD graph representation in string encoding format. Design and construction of PSG which is a compact representation of the web log database is discussed in section 4. Section 5 deals with the method of extracting frequent patterns from PSG (PSG-mine). Results obtained from experimental work are discussed in section 6. Finally, section 7 provides concluding remarks.

2 Definitions

Most of the web users are visiting a set of pages in a web site more often. It is interesting to develop a novel and efficient method of finding users behavior by extracting highly visited set of pages with the order of visit. The order of visit is captured using RCD graphs. Rest of the section gives the related definitions with respect to RCD graphs.

2.1 RCD graph

A RCD graph G consists of a set of vertices V = $\{v_1, v_2, v_3 \ldots v_n\}$, a set of edges E = $\{e_1, e_2, e_3 \ldots e_m\}$, and a mapping φ that maps every edge onto some ordered pair of vertices (v_i, v_j). A vertex is represented by a point and an edge by a line segment between v_i and v_j with an arrow directed from v_i to v_j. A RCD graph is one which is having a unique vertex (called root node) and if that RCD graph is having n number of edges, then the first edge should start from root node to any node (say v_2) and next edge should be from v_2 to any node and so on for the remaining edges. Figure 3, 2 (b) and (c) are different RCD graphs.

RCD graph is an ideal data structure to capture the information in user web log sessions with respect to an organization. This is because each user has to start from the home page of an organization (i.e. root) and there is a continuity in terms of visiting next page with respect to the current web page.

2.2 RCD subgraph

A graph g is said to be a RCD subgraph of RCD graph G, if (i) all vertices and all set of the edges (which is non-empty) of g are in G, and all edges of g must have same sequence as that of G or (ii) g has only root vertex. Figure 3 shows the directed subgraphs of the graph as shown in Figure 2 (c).

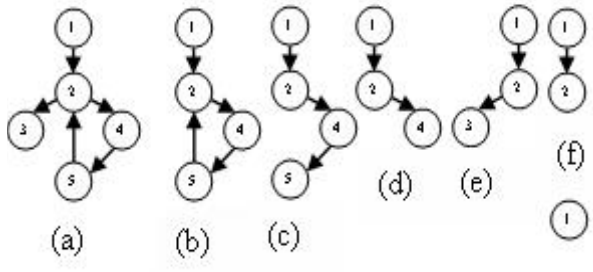

Figure 3: Subgraphs of the graph in Figure 2 (c)

2.3 Frequent RCD subgraph

Let DB = < T_1, T_2, T_3 … T_n > be a collection of web logs pertaining to n user web log sessions. Note that each T_i is a RCD graph (refer 2.1). A RCD subgraph S Є T_i, for any i ($1 < i < n$) is said to be frequent if number of occurrences of S in DB is greater than or equal to user defined minimum support value.

Let us consider the RCD graphs shown in Figure 4 (a), (b) and (c). Let us assume user specified minimum support = 3. Figure 4 (d) is a frequent RCD subgraph.

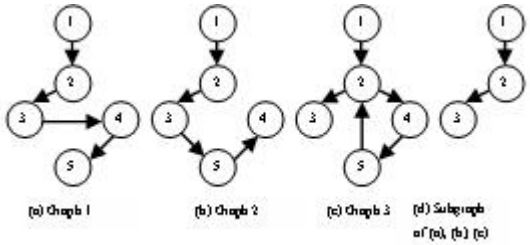

Figure 4: Directed graphs for subgraph mining

3 Representation of directed graphs as strings

We are proposing a modified version of string encoding format proposed by Zaki in [6]. This string encoding format constituted by referring user web log sessions this format is not only complete with respect to user web log sessions but also compact in size. This is an intermediate structure to construct PSG from user web log sessions which is an RCD graph.

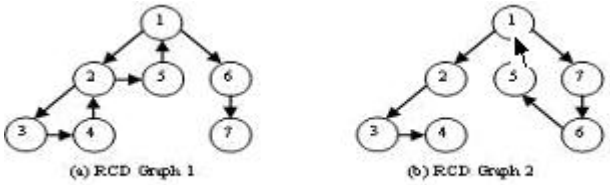

Figure 5: Examples for graph representation

The string encoding procedure for RCD graph is as follows. Start with the root node of RCD graph, insert it into the string, T. Since RCD graph is continuous and directed, insert subsequent nodes as they are visited. When ever there is a re-visit to a node, then add that node to the string with a – sign.

A directed graph is generally represented in one of the following three forms. They are (i) Adjacency Matrix which requires n^2 binary matrix, where n is the number of vertices, (ii)Edge Listing requires which twice the number of edges, (iii) Two Linear Arrays which requires 2 arrays of e size where e represents number of edges. Our approach requires only e+1 number of spaces where e is number of edges.

The technique proposed by M J Zaki in [6] is for tree representation. With some modifications we extended it for RCD graph representation. The string encoding format for Figure 5 (a) is 1 2 3 4 -2 5 -1 6 7 and for Figure 5 (b) is 1 7 6 5 -1 2 3 4. Zaki's approach is not applicable for RCD graphs. Even if we convert RCD graphs to trees then also it takes more space compare to our approach. Figure 6 (a) and (b) are tree representation of RCD graphs of Figure 5 (a) and (b) respectively. String encoding format for Figure 6 (a) and 6 (b) is respectively given as follows: 1 2 3 4 -1 -1 5 -1 -1 6 7 and 1 2 3 4 -1 -1 -1 -1 7 6 5. Even in worst case our approach takes e+1 number of spaces but in Zaki's approach, it is not depends on how much to backtrack. If backtrack is only one step at any node, then our approach is same as Zaki's approach.

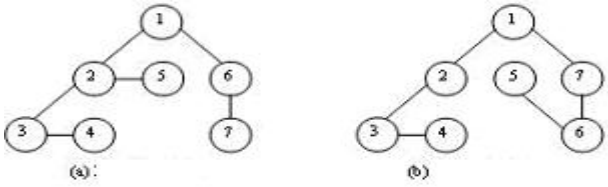

Figure 6: Example for trees

4 Pattern Super Graph (PSG): Design and Construction

PSG is constructed by merging string encoded format of different user's web log sessions with respect to an organization. PSG consists of two types of nodes namely Visit Node and Re-visit Node. Visit Node consists of the following fields: Label, Count, Child Pointer, Sibling Pointer, Re-visit Pointer and Back Pointer. The Label field is used to hold web page number. Count field is used hold the number of patterns till to this Visit Node from the first Visit Node where that pattern starts. Incrementing of Count field value is done at the time of merging string encoded format of different user's web log sessions. Child Pointer points to one of its child node which is either a Visit or Re-visit Node. Sibling Pointer of a Visit Node points to its sibling. Back Pointer points to a Visit Node where it is emerged. Each Re-visit Node consists of the following fields: Label, Count and Next Pointer. Label field is used to hold web page number of a visited node which is to be re-visited. Count field is used to hold the number of patterns till to this node from where that pattern starts. Next Pointer points to the next Re-visit Node if any. Re-visit Node provides the page number to be re-visited and the re-visiting is done with the help of Back Pointers. With the help of Re-visit Node and Back Pointer we are capturing the looping information. PSG structure is shown in Figure 7.

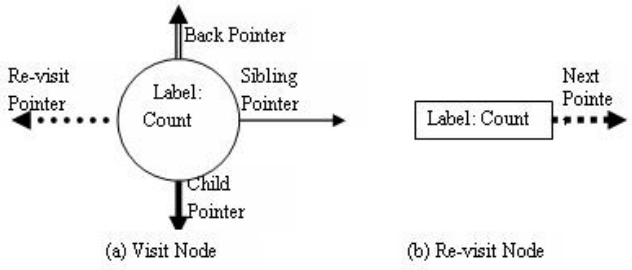

Figure 7: structures of PSG

We illustrate the process of PSG construction for a given user web logs sessions which are represented in string encoding format is given in Table 1.

Table 1: Database of user sessions

Session-Id	String encoded format of session Graph
1	1 2 4 -2 5 -2 6 -2 7 8
2	1 2 9 -2 5 -2 7 8 -2 6
3	1 3 4 5 -1 2 7 8 -1 2 9

Consider the first string encoded format of user web logs of a session graph (i.e. Session-Id = 1). The PSG corresponding to this is given in Figure 8. After merging the second user web log session of graph (i.e. Session-Id = 2) with the PSG as shown in Figure 8, the resultant PSG is shown in Figure 9. Note that updating Counter field of Visit Node and Re-visit Node may be done during merging of user web log sessions. Figure 10 shows the resultant PSG after merging user web log sessions (i.e. Session-Id = 3) to PSG shown in Figure 9. Note that the construction of the PSG from string encoded format of user web log sessions requires single scan on string encoded format of user web log sessions.

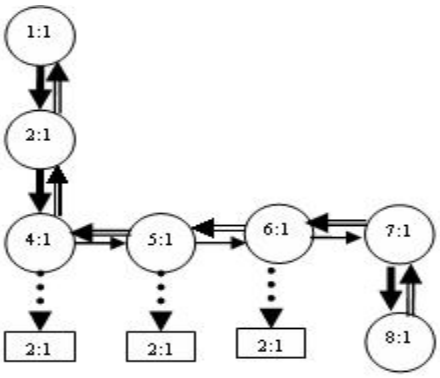

Figure 8: PSG for first user session

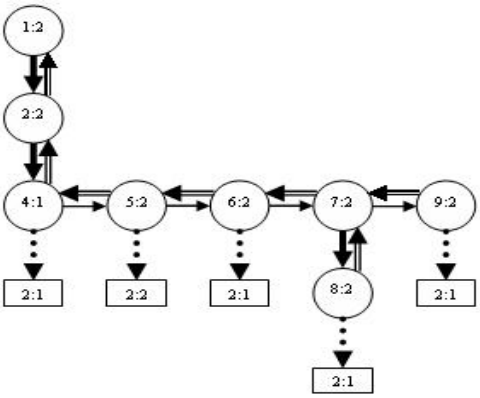

Figure 9: PSG of two user sessions

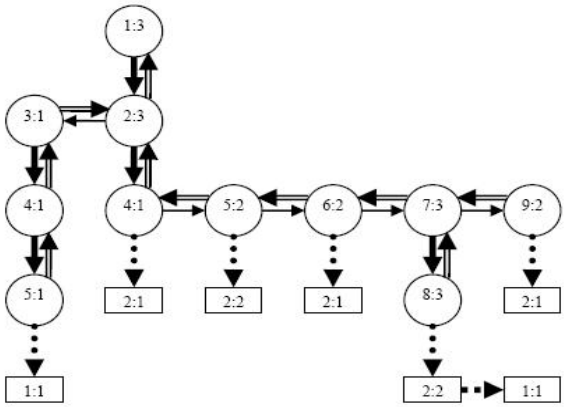

Figure 10: PSG of three user sessions

4.1 Completeness and compactness of PSG

There are several important properties of PSG that can be derived from the PSG construction process. PSG is the complete representation of RCD subgraphs of the database of the string encoded format of user web log sessions (say DB_2). The DB_2 is complete representation of the database of web logs (say DB_1). By transitive property, we can say that PSG is complete representation of the RCD subgraphs of DB1. Each RCD subgraph in DB_2 is mapped to one of the path in PSG. Different RCD subgraphs in PSG are distinctly placed. Suppose two or more RCD subgraphs are identical, and then the PSG maintains those set of identical RCD subgraphs as a single RCD subgraph and update its count of occurrence accordingly. So PSG is a compact representation of DB_1. The size of a PSG is bounded by sum of unique RCD subgraphs of the graphs of DB_1, and the height of the PSG is bounded by maximum height of the RCD subgraph among the DB_1. Results obtained (discussed in section 5) are showing the compactness of the PSG. As the number of sessions increases the compactness of PSG also increases. This is because as the number of users visiting the site increases, we can expect more similar kind of behavior of visiting the same set of pages.

The size of a PSG is bounded by the size and nature of its corresponding database. This is because there is often a lot of sharing of subgraphs among graphs pertaining to different user's web log sessions. The size of the PSG is usually much smaller than its original database, DB_1. Even in the worst case, maximum size of the PSG is less than that of DB_1. This is because at least root node is common to all RCD graphs.

5 Mining frequent RCD subgraph from PSG

In this section, we are exploring the compact information stored in the PSG. We propose a frequent RCD subgraphs extraction algorithm, PSG-mine, for mining the complete set of frequent RCD subgraphs from PSG.

In this mining process, we have to handle two categories of RCD subgraphs. First category is subgraphs having only one path. Another category is subgraphs having multiple paths i.e., the path sharing common prefixes. Extraction of the first category of subgraphs needs to traverse along that subgraph to its last node and check its support i.e., the Count field value the of last node (Visit or Re-visit Node). If support is more than user specified value, then extract that RCD subgraph and put it in frequent subgraph list, L. For RCD subgraphs having multiple paths, first move to the node where there is a diversion (i.e., to the node which has many child nodes). While moving, add nodes in the path to a temporary array, T in FIFO order. At diverted position, select non-considered path and prefix T to the nodes selected along that path. Let it be T1. The count of the last node in the path gives the support value for T1. If it is more than user specified support value then move it to L and drop T1. Repeat the above steps at diverted position for remaining paths. The complete set of frequent RCD subgraphs for the PSG, shown in the Figure 7 for different minimum support is given in the Table 2.

Table 2: RCD subgraphs with minimum support

Minimum support	RCD Subgraphs in string encoding format
1	1 3 4 5 -1, 1 2 4 -2, 1 2 5 -2, 1 2 6 -2, 1 2 7 8 -2, 1 2 7 8 -1, 1 2 9 -2
2	1 2 5 -2, 12 7 8 -2
3	1 2 7 8

The entire process of RCD frequent subgraph generation from user web log sessions of an organization can be depicted pictorically as shown in Figure 11.

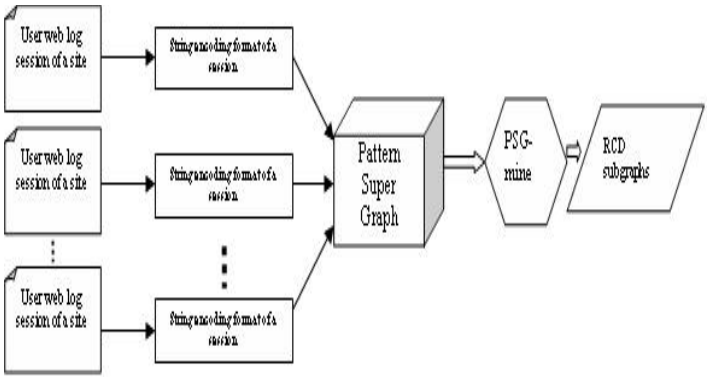

Figure 11: Process of frequent RCD subgraph mining

6 Results and discussion

We have conducted the experiments using datasets consists of web logs files collected over 1 month at an academic institution. The logs touched 381796 user requests, the number of sites covered is 2461and the number of sessions is 87085. We are interested in one site analysis at any time. The following discussion is based on web logs collected from a site, www.wwe.com. The site, www.wwe.com gives good number of user sessions say 110 to 5123 for different log files. The site consists of 1336 unique web pages. The average string encoding length for a user graph was 18.

We wrote a data preparation module to convert web logs into RCD graphs of a user sessions represented in string encoding format. The module performs following set of operations: (i) Select user request of the given site. (ii) Group the user requests having same Session-Id. (iii) Construct the session graphs as follows: For each Session-Id, separate the nodes when '/' is encountered from each URLs. For the first URL, move nodes to Session-Id. For remaining URLs, compare correspondingly these nodes with previous URL nodes where it differs then add '–' with the node previous to where it differs and add remaining nodes.

A PSG is usually much smaller than the size of string encoded format of user's web log sessions. The compactness of PSG is shown in two ways: (i) compare the number of nodes in PSG and the number of nodes in corresponding string encoded format of user web log session graphs. This is shown in Table 3 and the corresponding graph is given in Figure 12. it is clear from the Table (Graph) that, as the number of user web log sessions increases from 110 to 5123, the size of PSG in terms of nodes is reduced from 67% to 92% with respect to string encoded format of user web log sessions. (ii) Another way is to compare the number of distinct RCD subgraphs generated in PSG and string encoded format of user web log sessions. This is shown in Table 4 and corresponding graph is given in Figure 13. It is clear from Table (Graph) that as the number of user's web log session increases from 110 to 5123, the total number of RCD subgraphs in PSG is reduced from 34% to 77% with respect to the number of RCD subgraphs in string encoding format of user web log sessions. These two results clearly show the compactness of PSG with respect to user web log sessions pertaining to www.wwe.com.

7 Conclusions

The method of merging of user web log sessions of an organization is introduced in this paper. This is a promising approach for scalable frequent pattern mining. The other existing approach in directed graph mining is based on the candidate generation which is costlier in terms of computation and space. We have proposed a novel data structure, Pattern Super Graph (PSG) for storing compressed crucial information about RCD subgraphs. We have developed a pattern mining method, PSG-mine, for efficient mining of frequent patterns in large databases. This method avoids multiple database scans and candidate generation. We also formulated a modified version of string encoding of the RCD graph which is space-efficient.

7.1 Future Enhancements

The philosophy of database compression (merging pattern) and frequent-pattern mining can be extended to constraint-based mining and mining other kinds of frequent patterns, such as max-patterns, sequential patterns. Finally, we plan to apply our RCD subgraph mining techniques to other compelling applications, such as finding common subgraph patterns in bioinformatics, telecommunication and analyzing the executions of a buggy software program.

Table 3: Number of paths for sessions

Sl No.	No. of sessions	No. of nodes in session graphs	No. of nodes in PSG
1	110	1457	468
2	500	5628	1000
3	1000	10520	1654
4	2000	21694	3161
5	3000	32512	3768
6	3721	40052	3989
7	4645	54550	4998
8	5123	59934	5187

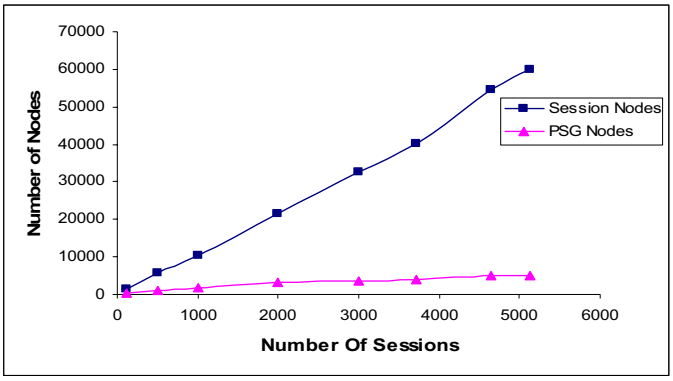

Figure 12: No. sessions verses No. nodes

Table 4: No. of paths for sessions

Sl No.	No. of sessions	No. of paths in session	No. of paths in PSG
1	110	363	238
2	500	1256	539
3	1000	2231	921
4	2000	4689	1787
5	3000	6761	2145
6	3721	8198	2284
7	4645	11997	2873
8	5123	13111	2973

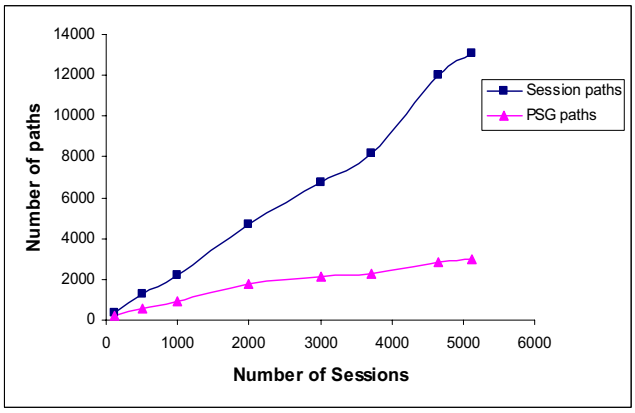

Figure 13: No. sessions versus No. paths

References

[1] R. Agrawal, et al. Fast discovery of association rules. Advances in Knowledge Discovery and Data Mining, pages 307-328. AAAI Press, Menlo Park, CA, 1996.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In 11th Intl. Conf. on Data Engg., 1995.

[4] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In 1st IEEE Int'l Conf. on Data Mining, November 2001.

[5] R. Cooley, B. Mobasher, and J. Srivastava. Web Mining: Information and Pattern Discovery on the World Wide Web. In 8th IEEE Intl. Conf. on Tools with AI, 1997.

[6] M. J. Zaki, Efficiently Mining Frequent Trees in a Forest: Algorithms and Applications, IEEE Transactions on Knowledge and Data Engg, Vol. 17, No. 8, pages 1021-1035 August 2005

[7] Christian Borgelt and Michael R. Berthold. Mining Molecular Fragments: Finding Relevant Substructures of Molecules. In Vipin Kumar, Shusaku Tsumoto, Ning Zhong, Philip S. Yu, and Xindong Wu, editors, Proc. IEEE Int'l Conf. on Data Mining ICDM, pages 51-58, Maebashi City, Japan, December 2002. IEEE Press.

[8] Xifeng Yan and Jiawei Han. gSpan: Graph-Based Substructure Pattern Mining. In Vipin Kumar, Shusaku Tsumoto, Ning Zhong, Philip S. Yu, and Xindong Wu, editors, Proc. IEEE Int'l Conf. on Data Mining ICDM, pages 721-723, Maebashi City, Japan, December 2002. IEEE Press.

[9] Jun Huan, Wei Wang, and Jan Prins. Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism. In Jude Shavlik, Xindong Wu, and Alex Tuzhilin, editors, Proc. of the 3rd IEEE Intl. Conf. on Data Mining ICDM, pages 549-552, Melbourne, FL, USA, November 2003. IEEE Press.

[10] Siegfried Nijssen and Joost N. Kok. A Quickstart in Frequent Structure Mining can Make a Difference. Technical report, LIACS, Leiden University, Leiden, The Netherlands, April 2004.

[11] Marc Sebastian WÄorlein. Extension and parallelization of a graph-mining-algorithm. A Ph.d Thesis. Submitted to Institut fur Informatik Lehrstuhl fur Informatik 2 Programmiersysteme Friedrich-Alexander-Universitat Erlangen-Nurnberg in March 2006.