International Journal of Computer Applications (0975 – 8887)
Volume 137 – No.7, March 2016

Pushing Constraints to Generate Top-K Closed Sequential Graph Patterns

K. Vijay Bhaskar
GITAM University
Visakhapatnam, 530045
India

K. Thammi Reddy, PhD
GITAM University
Visakhapatnam, 530045
India

S. Sumalatha
National Institute of Technology
Warangal, 506004
India

ABSTRACT

In this paper, the problem of finding sequential patterns from graph databases is investigated. Two serious issues dealt in this paper are efficiency and effectiveness of mining algorithm. A huge volume of sequential patterns has been generated out of which most of them are uninteresting. The users have to go through a large number of patterns to find interesting results. In order to improve the efficiency and effectiveness of the mining process, constraints are more essential. Constraint-based mining is used in many fields of data mining such as frequent pattern mining, sequential pattern mining, and subgraph mining. A novel algorithm called CSGP (Constraint-based Sequential Graph Pattern mining) is proposed for mining interesting sequential patterns from graph databases. CSGP algorithm is revised to mine top-k closed patterns and named as TCSGP (Top-k Closed constraint-based Sequential Graph Pattern mining).

General Terms

Data mining, Graph mining, Constraint-based mining.

Keywords

Sequential patterns, Closed patterns, Constraints.

1. INTRODUCTION

Sequential pattern mining and Constraint-based sequential pattern mining [4,12,13,18,20,27] are an interesting research area in the field of data mining. The aim of constraint based data mining [9,10,15,17] is to provide the user more interesting patterns. Constraints make the mining more effective and efficient. In order to reduce a very large input search space, constraints were pushed deep into the mining process. The importance of graph mining has increased in the past few years. Graphs are one of the important data structures in computer science. Graphs are very much useful in representing the relationships among objects. There are many subgraph mining algorithms [3] to find the frequently occurring subgraphs from the graph database. In literature only much less work is done on constraint based graph mining [5,25,26]. Two problems were found while generating sequential patterns from graphs.

1.A Huge volume of sequential patterns is generated out of which most of them are uninteresting. The users have to go through a large number of patterns to find interesting patterns.
2. The time taken to generate the sequential patterns is exponential time.

The following issues are addressed in this paper.
1. Generating sequential patterns from graph databases.
2. Improving the effectiveness and efficiency of the mining process by pushing constraints deep into the mining process.

The rest of the paper is organized in the following manner. Section 2 describes the related work. Section 3 describes the problem statement and introduces various constraints. Section 4 describes proposed algorithm. Experimental results are shown in section 5 and the work is concluded in section 6.

2. RELATED WORK

Constraint based graph-pattern mining improves the effectiveness and efficiency of the mining process. Constraints drive the mining algorithm towards more interesting patterns. In [20], a family of novel algorithms was presented for frequent sequential pattern mining. These algorithms make use of user-specified regular expression constraints. Interactive process of running mining algorithms by repeatedly changing the constraints is very time consuming. An efficient method was proposed in [8] to speed up the mining process by utilizing the previous mining results. The method in [8] was implemented on FP-tree and Tree projection algorithms.

An approach [1] for automatically relaxing constraints has been developed. In [1], the authors defined two new operators to detect monotone or antimonotone constraints. Pushing constraints deep inside the mining algorithm reduces the search space of patterns and achieves high performance. A method for pushing convertible constraints deep inside frequent pattern-growth mining was introduced in [11].

Two algorithms CLOSECUT and SPLAT were proposed in [26] to mine closed relational graphs with connectivity constraints. CLOSECUT was a pattern-growth approach and SPLAT was a pattern-reduction approach. In [5], the authors developed novel algorithms to prune the search space for both data and patterns. In addition to frequency constraint, two new concepts compactness and recency are considered in [27]. Frequency and recency constraints ensure patterns that occur in the long run and in recent time periods. Compactness ensures that patterns occur within a reasonable time period.

An algorithm [18] was developed to perform constraint-based mining of sequential patterns in the presence of consecutive repetitions. CabGin framework [2] was developed to push graph-based constraints into mining algorithm. Large search space was effectively pruned by pushing graph-based constraints into mining algorithm. Data-Peeler [14] algorithm extracts all closed n-sets from n-ary relation. It [14] is a depth first approach that computes closed n-sets satisfying (anti) monotonic constraints. The idea of soft constraints was introduced in [22]. In this paper [22], the authors introduced soft constraints, the constraints which are not boolean functions. Instead of dividing patterns into two classes, namely interesting and not interesting, soft constraints based paradigm generates an order of patterns in which one pattern is more interesting than others. Constraints can also be pushed into the mining process by means of data reduction techniques. ExAminer [6] was a level-wise Apriori-like framework based on data reduction techniques.

CONQUEST [7] was an exploratory pattern discovery system. Interactive extraction of interesting knowledge is possible with CONQUEST, user-defined constraints define the pattern interestingness. WCloset [24] was developed to mine closed frequent patterns with weight constraints. The authors of [24] introduced lossless closed weighted frequent pattern mining and proved that there will be an information loss in weighted closed frequent pattern mining.

In this paper, two algorithms are developed, namely, CSGP and TCSGP to deal with the issues of pushing constraints deep into the mining process to generate most interesting sequential patterns from graph databases.

3. PROBLEM STATEMENT

In this section, the basic concepts of graphs, various constraints, the problem of constraint-based sequential graph pattern mining and Top-k closed constraint-based sequential graph pattern mining are defined.

3.1 Basic definitions

3.1.1 Labeled directed graph

A Labeled directed graph is defined as a tuple, G= (V, E, L, F) where V denotes a set of vertices, E denotes a set of directed edges, L denotes a set of labels and F is a labeling function that assigns labels to the vertices and edges. Every edge is a 4 tuple <s, d, g, l>, where s is a source vertex, d is a destination vertex, g is a graph id and l is an edge label. Every edge in a graph is assigned a unique timestamp as an edge label. A sample Graph data set is shown in Figure 1

3.1.2 Graph sequential pattern

An ordered collection of edges in a graph is called a graph sequence. A Graph sequence can be represented as,

GS=$<v_a v_b \ldots v_n>$ where v_a, v_b, v_n are the vertices of a graph. Sequence of graph edges in the input graph dataset is shown in Table 1. A frequent graph sequence that satisfies the given minimum support threshold is called a Graph sequential pattern. Length of the Graph sequential pattern is the number of edges in the sequence.

3.1.3 Canonical form of a graph

A standard way of representing a graph is known as its canonical form. Table 2 shows the Canonical form of a graph data set in Figure 1. Given a graph G_i with n vertices, m edges and an ordered sequence of edges $<v_1,v_2,\ldots v_n>$, Canonical form of G_i is denoted as $(v_1,v_2,i,1),(v_2,v_3,i,2),\ldots(v_{n-1},v_n,i,m)$

3.1.4 Projected database of a vertex

Given a Graph Database GD = $\{G_1, G_2,\ldots G_n\}$, The projected database of a vertex V is the set of subsequences whose prefix is V. For example, in the given graph database, the sequence that starts with vertex ‘a’ as the source forms projected database of ‘a’. In Figure 1, edge 1 of Graph 1, edge 1 of Graph 2, edge 3 of Graph 3, edge 2 of Graph 4, edge 1 of Graph 5 and edge 2 of graph 6 forms the sequences starting with v. Instead of storing the projected database separately for every vertex, considering the space constraints, store only the <Graph-Id, Edge-Id> pairs. Hence a-projected database is <(1,1), (2,1), (3,3), (4,2), (5,1), (6,2) >.

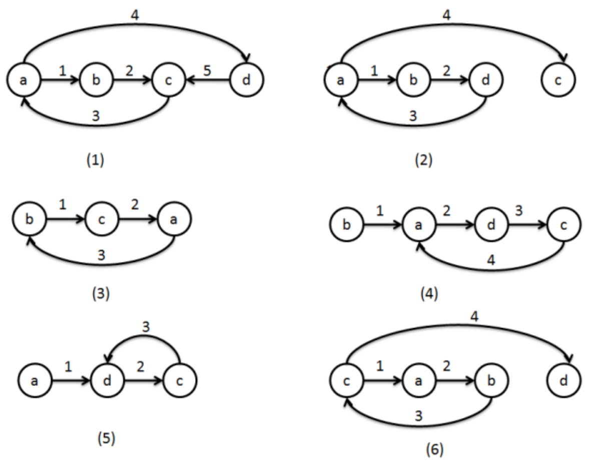

Fig. 1. An example graph data set

Table 1. Sequence of graph edges

(1)	(2)	(3)	(4)	(5)	(6)
a→b	a→b	b→c	b→a	a→d	c→a
b→c	b→d	c→a	a→d	d→c	a→b
c→a	d→a	a→b	d→c	c→d	b→c
a→d	a→c		c→a		c→d
d→c					

3.2 Constraints

A constraint C is a boolean function C (S_g) on the set of all sequential graph patterns. A graph sequential pattern S_g satisfies a constraint if and only if C (S_g) is true.
Seven constraints are used in the proposed algorithms. These constraints are classified into two classes as shown in Table3.

1. Constraints to improve the efficiency of the mining process.
2. Constraints to improve the effectiveness of the mining result.

Table 3 shows the constraints and their performance level to improve the efficiency and effectiveness of mining.

3.2.1 Constraint1: Average Constraint

Average constraint is defined as the average number of edges per graph. Let the E_{G1}, E_{G2}, $E_{G3},\ldots E_{Gn}$ be the number of edges in graph1, graph2, graph3 and so on respectively. The average constraint threshold is calculated as follows:
avg = $\lceil \sum E_{Gi} \rceil$ / n
Where avg is an average constraint threshold, n is the total number of graphs in input database.

Table 2. Canonical representation of graphs

Graph id	Canonical form
1	<(a,b,1,1),(b,c,1,2),(c,a,1,3),(a,d,1,4),(d,c,1,5)>
2	<(a,b,2,1),(b,d,2,2),(d,a,2,3),(a,c,2,4)>
3	<(b,c,3,1),(c,a,3,2),(a,b,3,3)>
4	<(b,a,4,1),(a,d,4,2),(d,c,4,3),(c,a,4,4)>
5	<(a,d,5,1),(d,c,5,2),(c,d,5,3)>
6	<(c,a,6,1),(a,b,6,2),(b,c,6,3),(c,d,6,4)>

Table 3. Constraint classification

Sno.	Constraint name	Class1: Improves Efficiency	Class2: Improves Effectiveness
1	Average measure, C_a	Low	Low
2	Support of a vertex, C_{sv}	Low	Low
3	Support of Prefix of a sequence, C_{sp}	High	Low
4	Support of a sequence extension, C_{se}	High	Low
5	Closed sequential pattern, C_c	Low	Low
6	Minimum length constraint, C_m	Low	High
7	Top-k sequential pattern, C_k	High	High

Table 4. Projected database of vertices

Vertex v	Projected database of v	Support of v	Vertices reachable from v
a	<(1,1),(2,1),(3,3), (4,2), (5,1),(6,2) >	6	<b:6, c:6, d:6>
b	<(1,2),(2,2),(3,1),(4,1),(6,3)>	5	<a:5, c:6, d:4>
c	<(1,3),(3,2),(4,4),(5,3),(6,1)>	5	<a:4, b:2, d:3>
d	<(1,5),(2,3),(4,3),(5,2)>	4	<a:2, c:4>

Example1: In figure1, avg = (5+4+3+4+3+4)/6 = 4.

Graphs 3 and 5 do not satisfy the Average constraint and they are pruned from the graph database.

3.2.2 Constraint2: Support of a vertex

A directed graph may contain multiple sequences of edges starting at a vertex v. If a graph G contain one or more edges from v to any other vertex, then support count of v is 1 with respect to the graph G. Support of a vertex v is defined as the total number of edge sequences that start from vertex v with respect to all the graphs.

Example2: Support of vertex 'a' in Figure1 is 6

All the 6 graphs contain at least one edge starting from 'a'. Hence its support is 6. The support of all vertices in the input graph database is given in Table 4.

3.2.3 Constraint3: Support of a prefix

The set of first occurrence of (graphid, edgeid) pairs whose prefix is S with respect to all the graphs is called the projected database of a sequence S.The count of such (graphid, edgeid) pairs for a sequence S is known as the support of prefix.

Example3: Projected database of a sequence <ac> is

calculated as follows:

Scan the projected database of vertex 'a',

<(1,1),(2,1),(3,3),(4,2),(5,1),(6,2)>

Scan graph 1 and find the first occurrence of 'c' and it is found to be 3. Scan graph 2 and find the first occurrence of ‘c’ and it is found that there is no sequence starting with ‘c’ as the source vertex. Similarly, there is no sequence in graph 3. In graph 4 the first occurrence of ‘c’ as source is 4, similarly in

graph 5 it is 3 and in graph 6 it is 4. Hence, projected database of <ac> is < (1,3), (4,4),(5,3),(6,4) > and support of prefix <ac> is 4, that is the total number of (graphid, edgeid) pairs in its projected database.

3.2.4 Constraint4: Support of a sequence extension

The number of frequent sequences generated from the given sequence is known as the support of sequence extension.

Example4: Support of a sequence extension of <ac> is

calculated as follows:

The frequent sequences generated from the sequence <ac> are <acd> and <aca>. The support of the sequence extension is therefore 2.

3.2.5 Constraint5: Closed sequential pattern constraint

A sequential pattern S is said to be closed if there exists no proper super sequential pattern with the same support as S. Example5: If a Sequential pattern <aba> and its super pattern <abac> are having the same support count 2 then <aba> is not closed sequential pattern.

3.2.6 Constraint6: Minimum length constraint

Minimum length constraint specifies the minimum length of a sequential pattern. The length of a sequence is defined as the number of edges in the sequence.

Example6: The length of sequences <ac>, <ad>, <ab> are 1. If the minimum length threshold is set as 2 then the sequences <ac> <ad> <ab> are not in the result even though they are frequent.

3.2.7 Constraint7: Top-k Sequential pattern

A sequential pattern is said to be Top-k sequential pattern of minimum length min_len if there exists no more than (k-1) sequential patterns whose length is at least min_len.

3.2.8 Constraint-based Sequential graph pattern mining

Given a graph database GD, Minimum support threshold min_sup and a set of constraints C, the problem of Constraint-based sequential graph pattern mining is to find the set of interesting sequential graph patterns satisfying C.

3.2.9 Top-k Constraint-based closed sequential graph pattern mining

Given a graph database GD, Minimum support threshold min_sup and a set of constraints C, the problem of Top-k Constraint-based closed sequential graph pattern mining is to find the top-k closed sequential graph patterns satisfying C.

4. PROPOSED ALGORITHMS

In this section, two algorithms CSGP and TCSGP are proposed, CSGP generates frequent sequential graph patterns based on constraints and TCSGP generates top-k frequent sequential graph patterns based on constraints.

4.1 CSGP

Algorithm 1 (CSGP) Constraint-based sequential graph pattern mining

Input:

Global data: A graph database GD, Minimum support threshold min_sup, Constraints: Average measure C_a, Support of a vertex C_{sv}, Support of a prefix C_{sp}, Support of sequence extension C_{se}**.**

Local data: Projected database of each vertex PD_v, Priority queue Q, Sequences generated from each vertex S.

Output: Frequent sequential graph patterns S_g.

Method:

```
Step1.   Find the average number of edges per graph
         and set it as a threshold for average measure
         constraint.
Step2.   Prune the graphs from the input database that do not
         satisfy the average constraint.
Step3.   Call the suprocedure1 FPDV (GD) to scan the
         pruned graph database and to find the projected
         database of each vertex and let it be $PD_v$.
Step4.   Prune the infrequent vertices that do not satisfy the
         threshold $C_{sv}$ and $C_{sp.}$ Let the new vertex set be V.
Step5.   Sort the frequent vertex set V in descending order of
         their support counts and insert them in a priority
         queue Q.
Step6.   Repeat
Step7.   Remove the vertex v from Q and find the
         sequences S that can be grown from v
         whose support count is not less than
         min_sup.
Step8.   Add the frequent sequences to $S_g$.
         $S_g \leftarrow S_g \cup S$
Step9.   ConstraintSequenceMining ($PD_v$, S)
Step10.  Until Q is empty
```

In step1, find the average measure constraint and push it into the mining algorithm. In step2, prune the graphs for which, the number of edges is less than the average measure constraint. Scan the pruned graph database and find the projected database of each vertex in step3. In step4, push the constraints C_{sv} and C_{sp} to prune the infrequent vertices that don’t satisfy support of a vertex and support of prefix constraints. Sort the new frequent vertex set in descending order of their support count and insert them in a priority queue. Repeat steps to 9 until the queue is empty. Find the sequences that can be grown from each vertex and call the recursive procedure ConstraintSequenceMining (PD_v, S). This procedure recursively finds the length 2, length 3 frequent patterns and so on.

Subprocedure1. (FPDV) FindProjectedDatabaseofVertex(GD)
Input: Pruned graph database GD

Output: Projected database of vertices $PD_{v,}$ Support count of vertices SC_v.

Method:

```
Step1.   For each graph i in pruned graph database GD
Step2.   For each edge j in graph i
         Step 2.1  Examine the edge $v \rightarrow u$.
         Step 2.2  if v is the first occurrence as a source in
                   edge j then
         Step 2.3  $PD_v \leftarrow PD_v \cup (i,j)$
```

```
        Step 2.4   $SC_v \leftarrow SC_v + 1$
Step3.  End for
Step4.  End for
```

Subprocedure1 scans each edge exactly once and finds the (graphid, edgeid) pair for each vertex as source. Whenever a new (graphid, edgeid) pair is found for a vertex, then SC_v is incremented by 1. SC_v holds the support count of vertex v.

Subprocedure2. ConstraintSequenceMining (PD_v, S)

Input: Projected database of a vertex PD_v, Set of graph sequences S.

Output: Frequent sequential graph patterns S_g.

Method:

```
Step1.  Repeat step 2 until S is empty.
Step2.  For every sequence $S_i$ in S
        Step 2.1  Call the subprocedure3 FPDS($PD_v$,S) to
                  find the projected database of the
                  sequence $S_i$ and let it be $PD_s$.
        Step 2.2  if the projected database count of $S_i$
                  satisfies $C_{sp}$ then scan the projected
                  database of $S_i$ and find the sequences
                  generated from $S_i$.
        Step 2.3  Add the new sequences to S′ only if they
                  satisfy the min_sup threshold.
        Step 2.4  if S′ is not empty then
                  $S_g \leftarrow S_g \cup S'$.
        Step 2.5  if number of frequent sequences in S′
                  satisfy the constraint $C_{se}$, then call the
                  recursive procedure ConstraintSequence
                  Mining ($PD_s$, S′).
Step3.  End for
```

In subprocedure2, push the constraints C_{sp} and C_{se}. C_{sp} is used to reduce the number of scans to projected database. Only the projected database of the sequences that satisfy C_{sp} are scanned to extend the sequences to length 2 , length 3 and so on. C_{se} is used to reduce the number of recursive calls

Table 5. Comparison between the constraints 1 to 4

	No Constraint	**C_a**	**C_{sv}**		**C_{sp}**		**C_{se}**		**$C_a \cup C_{sv}$**		**$C_a \cup C_{sp}$**		**$C_a \cup C_{se}$**		**$C_{sp} \cup C_{se}$**		**$C_a \cup C_{sp} \cup C_{se}$**	
		4	4	5	2	3	2	3	4/3	4/4	4/2	4/3	4/2	4/3	2/2	3/3	4/2/2	4/3/3
x	31	29	31	29	31	2	31	30	29	**23**	29	**23**	29	26	31	30	29	26
y	19	19	19	17	15	11	15	6	15	12	15	10	15	6	11	6	11	**5**
z	32	30	32	30	14	8	26	16	30	24	12	**7**	26	9	14	8	12	**7**

Table 6. Comparison between the constraints 5 to 7

	C_c	**$C_a \cup C_c$**	**C_m** min_length=2	**$C_c \cup C_m$**	**$C_a \cup C_c \cup C_m$**	**C_k** k=10	**$C_k \cup C_c$**	**$C_a \cup C_k \cup C_c$**	**All constraints from 1 to 7**
x	28	26	20	18	17	10	10	10	10
y	12	10	19	12	10	7	7	7	6
z	32	30	32	32	30	16	17	16	6

Table 7. Comparison between all the constraints

S.No.		**Number of sequential patterns generated, x**	**Number of recursive calls, y**	**Number of scans to projected database, z**
1	No constraint	x	y	z
2	C_a	<x	= y	<z
3	C_{sv}	=x for threshold 4, <x for threshold 5	=y for threshold 4, <y for threshold 5	=z for threshold 4, <z for threshold 5
4	C_{sp}	=x for threshold 2, <x for threshold 3	<y	<<z
5	C_{se}	=x for threshold 2, <x for threshold 3	<y for threshold 2, <<y for threshold 3	<z
6	$C_a \cup C_{sv}$	<x	<y	<z
7	$C_a \cup C_{sp}$	<x	<y	<<z
8	$C_a \cup C_{se}$	<x	<y for threshold 2, <<y for threshold 3	<z for threshold 4/2 <<z for threshold 4/3
9	$C_{sp} \cup C_{se}$	=x for threshold 2/2, <x for threshold 3/3	<y for threshold 2/2, <<y for threshold	<z for threshold 2/2

			3/3	<<z for threshold 3/3
10	$C_a \cup C_{sp} \cup C_{se}$	<x	<y for threshold 2/2, <<y for threshold 3/3	<z for threshold 4/2/2 <<z for threshold 4/3/3
11	C_c	<x	<y	=z
12	$C_a \cup C_c$	<x	<y	<z
13	C_m	<x	=y	=z
14	$C_c \cup C_m$	<x	=y	=z
15	$C_a \cup C_c \cup C_m$	<x	<y	<z
16	C_k	<x	<<y	<z
17	$C_k \cup C_c$	<x	<<y	<z
18	$C_a \cup C_k \cup C_c$	<x	<<y	<z

to the subprocedure ConstraintSequenceMining (PD_s, S′). The subprocedure2 is recursively invoked only if the number of frequent sequences extended from S satisfies the constraint C_{se}.

Subprocedure3. (FPDS) FindProjectedDatabaseofSequence (PD,S)

Input: Projected database PD, Graph sequence S, where S=<v_a,v_b,……v_n>

Output: Projected database of a sequence PD_s.
Method:

```
Step1. For each graphid, edgeid pair (i,j) in PD
Step2.   Scan graph i from j to find k, where k is the first
         occurrence of $v_n$.
Step3.   $PD_s \leftarrow PD_S \cup (i,k)$
Step4. End for.
```

4.2 TCSGP

Algorithm 2 (TCSGP) Top-k Closed constraint-based Sequential Graph Pattern mining.

Input:

Global data: A graph database GD, Minimum support threshold min_sup, Constraints: Average measure C_a, Support of a vertex C_{sv}, Support of a prefix C_{sp}, Support of sequence extension C_{se}, Closed Sequential pattern C_c, Minimum length constraint C_m, Top-k sequential pattern constraint C_k .

Local data: Projected database of each vertex PD_v, Priority queue Q.

Output: Top-k closed frequent sequential graph patterns S_g.

Method:

```
Step1.  Push the average constraint $C_a$ on the input graph
        database. Prune the graphs that do not satisfy $C_a$.
Step2.  Find the projected database $PD_v$ of all the vertices
        and their support counts. Push the constraints $C_{sv}$
        and $C_{sp}$ and prune the infrequent vertices.
Step3.  Scan the projected database of frequent vertices and
        find the frequent sequences of length-1.
Step4.  Sort all frequent length-1 sequences in descending
        order of their support counts and insert them into a
        priority queue Q.
Step5.  Call the sub procedure Topk_ClosedSequences
        ($PD_v$, Q).
```

The constraints C_a , C_{sv}, C_{sp} are applied in step1 and step2 to prune the input graphs that do not satisfy the average measure and to prune the infrequent vertices. The step3 of algorithm2 finds all the frequent length-1 sequences and step4 of algorithm2 arranges the sequences in descending order of their support. Recursive procedure Topk_ClosedSequences (PD_v, Q) is called in step5 to find the top-k closed frequent sequential graph patterns based on constraints.

Subprocedure4. Topk_ClosedSequences (PD_v, S)

Input: Projected Database of vertices PD_v, Set of frequent graph sequences S.

Output: Top-k closed frequent graph sequences based on constraints TCS.

Method:

```
Step1.  Repeat step2 until S is empty.
Step2.  For every sequence $S_i$ in S
        Step2.1  call the subprocedure 3 FPDS ($PD_v$, $S_i$) to
                 find projected database $PD_{Si}$ .
        Step2.2  if projected database count of $S_i$ satisfies
                 $C_{sp}$ then scan $PD_{Si}$ and find the sequences
                 that can be grown.
        Step2.3  for every new sequence grown from $S_i$
        Step2.3.1 Add the new sequence to S′ only if the
                 support count of the new sequence
                 satisfies min_sup.
        Step2.3.2 End for
        Step2.4 if there is no new sequence with support
                 same as the support of $S_i$ and if the length
                 of the sequence $S_i$ is not less than min_len
                 then add $S_i$ to set of closed sequences.
                 $TCS \leftarrow TCS \cup S_i$.
        Step2.5  if the number of sequences in TCS is
                 equal to k then return.
```

Step2.6 if the number of frequent sequences in S′ satisfies C_{se} then call the recursive procedure Topk_ClosedSequences (PD_{Si}, S′).

Step3. End for

Step2.1 of subprocedure4 calls the subprocedure3 to find the projected database of the input sequence. Five constraints C_{sp}, C_k, C_m, C_c, and C_{se} are applied in subprocedure4. The recursive procedure is returned either if number of sequential graph patterns generated are k or if the input set of sequences is empty. Here top-k closed patterns of minimum length min_len are generated. In order to generate top-k patterns, in algorithm2, all frequent length 1 sequences are found and arranged in descending order of their support. First grow the sequences of the highest support to generate top-k patterns.

Example: Given the graph database with min_support=2, average measure threshold=4, support of vector=3, support of prefix=2, support of sequence extension=2, min_length=2, k=10, the top 10 closed graph sequential patterns as mined as follows:

1. Frequent length-1 sequences generated from TCSGP algorithm are as follows

<ac:5>,<bc:5>,<ad:4><ba:4>,<bd:4>,<ab:3>,<ca:3>,<dc:3>,<cd:2>,<da:2>.

2. The sequence with the highest support is first grown, i.e. <ac>. Frequent sequences extended from <ac> are <aca:2>,<acd:2>. Among these 3 sequences only <aca> and <acd> are added to the list of closed sequential patterns. The next sequence that is extended is <bc>. The sequences <bca>, <bcd> are the frequent sequences generated from the prefix <bc>. The procedure is repeated until top 10 closed graph sequences are obtained.

3. The result of TCSGP algorithm is <aca:2>,<acd:2>,<bca:2>,<bcd:2>,<adc:3>,<ada:2>,<bac:3>, <badc:2>,<bdc:3><bda:2>.

5. RESULTS

CSGP and TCSGP algorithms are implemented and tested on a synthetic dataset produced by a graph database generator [23]. It is based on the IBM Quest Synthetic Data Generation Code for Association and sequential patterns. Data sets are generated by the graph generator based on the five parameters as shown in Table 9.

In CSGP Algorithm, first four constraints are used as mentioned in Table 3.

1. Average measure C_a : This constraint prunes the input data prior to the mining process. This is used to remove the graphs that do not have an average number of edges. The idea behind this constraint is, the graphs that do not have sufficient number of edges cannot be used as input to generate the frequent sequences.
2. Support of a vertex C_{sv} : This constraint prunes the infrequent vertices. The vertex whose support count is less than the minimum support threshold is said to be infrequent .
3. Support of a prefix C_{sp}: In mining frequent sequential patterns, sufficient time is spent in scanning the projected database of prefixes.Prune infrequent prefixes whose support don't satisfy the constraint C_{sp}. The idea behind this constraint is to prune the unnecessary extension of frequent sequences.
4. Support of a sequence extension C_{se} : Once frequent vertices visited from a sequence are found, then new frequent sequences are obtained. Recursive procedure is called to find the projected database of new sequences and to extend them. In order to reduce the number of recursive calls, a constraint C_{se} is proposed.

Constraints 3 and 4 improves the efficiency of the CSGP algorithm by reducing the frequency of scans to projected database and by reducing the frequency of recursive calls to subprocedure2. The patterns generated by CSGP algorithm are shown in Table 8. In CSGP algorithm, the constraints C_a , C_{sv}, C_{sp} and C_{se} are used. These constraints are applied individually as well as all the four constraints are pushed at once into the mining process. The values of parameters y and z obtained are minimized when the constraints C_a , C_{sp} and C_{se} are combined as highlighted in Table 5. Increasing the threshold of constraint C_{sv} reduces the number of recursive calls and number of projected database scans, but it results in pruning of vertices that might result in frequent sequences.

Table 8. Frequent sequential graph patterns

Vertex	Frequent sequential length 1 patterns	Frequent sequential length 2 patterns	Frequent sequential length 3 patterns
a	<ac:5>,<ad:4>,<ab:3>	<aca:2>,<acd:2>,<adc:3>,<ada:2>	<abcd:2>,<abdc:2>,<abac:2>
b	<bc:5>,<ba:4>,<bd:4>	<bca:2>,<bcd:2>,<bac:3>,<bad:2>,<bdc:3>,<bda:2>	<badc:2>
c	<ca:3>,<cd:2>	<cad:2>,<cac:2>	-
d	<dc:3>,<da:2>	-	-

Table 9. Database parameters

Parameter	Meaning
D	Total number of graphs
V	Number of vertex labels
E	Number of edge labels
T	Average size of the graph
M	Average density of the graph

Pushing the above mentioned four constraints into CSGP algorithm increased the performance of the algorithm compared to running the algorithm without constraints. In the experimental results, it is found that these constraints alone will not improve the effectiveness of the algorithm. The number of sequential patterns generated are still high as shown in Figure 2 and also found the importance of finding still more highly interesting patterns. This rise to the problem of improving effectiveness of the mining algorithm. Three more constraints are included and TCSGP algorithm is developed to deal with the problem of effectiveness of the mining process. C_m, C_c, C_k are the three additional constraints used in TCSGP algorithm. The comparison of the results obtained by introducing these 3 constraints is shown in Table 6 and Table 7. In the experimental evaluation, the performance of CSGP and TCSGP algorithm are compared as shown in Figure 3. Pushing all the constraints from 1 to 7 into TCSGP algorithm resulted in higher performance than CSGP algorithm.

6. CONCLUSION

In this paper, the problem of pushing constraints into the mining process is studied to generate top-k closed sequential graph patterns. Two algorithms are proposed, CSGP generates a constraint based sequential frequent graph patterns and TCSGP generates a constraint based top-k closed sequential frequent graph patterns. The constraints are classified into two classes, constraints that improve the efficiency of mining process and constraints that improve the effectiveness of the mining process. Both classes of constraints are pushed inside TCSGP algorithm and the algorithms are implemented on a synthetic database [23]. The constraint based top-k closed sequential graph pattern mining is more preferable to generate highly interesting graph patterns with high performance. Interactive process of running mining algorithms by repeatedly changing the constraints is very time consuming. As a future work, the mining process speed can be increased by utilizing the previous mining results. Data mining is an interactive and iterative process. The user must change the constraints and run the algorithm many times to obtain the final results. The interactive process of mining is time consuming. The proposed approach can be further extended to speed up the mining process by incrementally mining the graph data whenever constraints are changed.

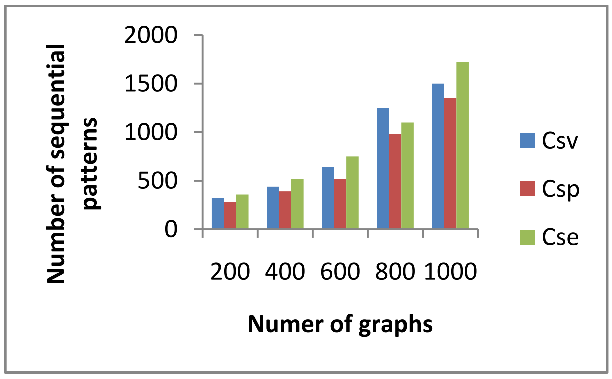

Fig. 2. Efficiency of constraints

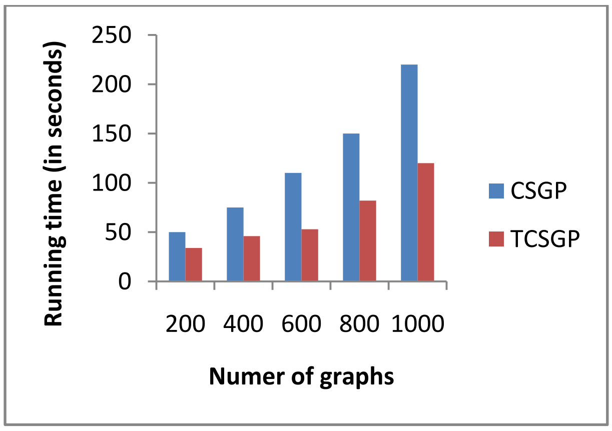

Fig. 3. Effectiveness of constraints

7. ACKNOWLEDGMENTS

We express our deepest gratitude to Prof. R. B. V. Subramanyam for his inspiration, comments and suggestions.

8. REFERENCES

[1] Arnaud Soulet, and Bruno Cremilleux, "Optimizing Constraint-Based Mining by Automatically Relaxing Constraints", Fifth IEEE International conference on Data mining, Nov 2005.

[2] Chen Wang, Yangtai Zhu, Tianyi Wu, Baileshi, "Constraint-Based Graph Mining in Large Database", Web Technologies Research and Development-ApWeb 2005, Lecture Notes in Computer Science, Volume 3399, 2005, 133-144.

[3] Chuntao Jiang, Frans Coenen and Michele Zito, "A survey of Frequent Subgraph Mining Algorithms", The Knowledge Engineering Review, Volume 28, Issue 01, Mar 2013, 75-105.

[4] F. Masseglia, P. Poncelet, and M. Teisseire, "Efficient mining of sequential patterns with time constraints: Reducing the combinations", Expert Systems with Applications, Elsevier, Volume 36, Issue 02, Mar 2009, 2677-2690.

[5] Feida Zhu, Xifeng Yan, Jiawei Han, and Philip S. Yu, "gPrune: A Constraint Pushing Framework for Graph Pattern Mining", Advances in Knowledge Discovery and Data Mining, Lecture notes in Computer Science, Springer, Volume 4426, 2007, 388-400.

[6] Francesco Bonchi, Claudio Lucchese, "Extending the state-of-the-art of Constraint-based pattern discovery", Data and Knowledge Engineering, Elsevier, Volume 60, Issue 02, Feb 2007, 377-399.

[7] Francesco Bonchi, Fosca Giannotti, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Roberto Trasarti, "A constraint-based querying system for exploratory pattern discovery", Information Systems, Elsevier, Volume 34, Issue 01, March 2009, 3-27.

[8] Gao Cong, and Bing Liu, " Speed-up Iterative Frequent Itemset Mining with Constraint Changes". In proceedings of 2002 IEEE International conference on Data mining, 2002, 107-114.

[9] Jean-Francois Boulicat and Baptiste Jeudy, "Constraint-Based Data Mining". The Data mining and Knowledge Discovery Handbook, Springer, 2005, 399-416.

[10] Jian Pei and Jiawei Han, "Can We Push More Constraints into Frequent Pattern Mining?". In Proceedings of the Sixth ACM SIGKDD international conference on knowledge discovery and data mining, 2000, 350-354.

[11] Jian Pei, Jiawei Han, and Laks V.S. Lakshmanan, "Mining Frequent Itemsets with Convertible Constraints". In proceedings of 1th international conference on Data Engineering, IEEE, April 2001, 433-442.

[12] Jian Pei, Jiawei Han, and Wei Wang, " Mining Sequential Patterns with Constraints in Large Databases". In Proceedings of CIKM'02 Eleventh International conference on Information and knowledge management, ACM, Newyork, 2002, 18-25.

[13] Jian Pei, Jiawei Han, and Wei Wang, "Constraint-based

sequential pattern mining: the pattern-growth methods", Journal of Intelligent Information Systems, Springer, Volume 28, Issue 02, April 200, 133-160.

[14] Loic Cerf, Jeremy Besson, Celine Robardet, Jean-Francois Boulicaut, "DATA-PEELER: Constraint-Based Closed Pattern Mining in n-ary Relations". In proceedings of the 2008 SIAM International conference on Data Mining, 2008, 37-48.

[15] Luc De Raedt, and Albrecht Zimmermann, "Constraint-Based Pattern Set Mining". In proceedings of the 2007 SIAM International conference on Data Mining, 2007.

[16] Luc De Raedt, Tias Guns, Siegfried Nijssen, "Constraint Programming for Itemset Mining". In Proceedings of KDD'08, ACM, Aug 2008, 204-212.

[17] Marek Wojciechowski and Maciej Zakrzewicz, "Dataset Filtering Techniques in Constraint-Based Frequent Pattern Mining". In Proceedings of the ESF Exploratory workshop on Pattern Detection and Discovery, Springer, 2002, 77-91.

[18] Marion Leleu, Christophe Rigotti, Jean-Francois Boulicaut, and Guillaume Euvrard, " Constraint-Based Mining of Sequential Patterns over Datasets with Consecutive Repetitions", Knowledge Discovery in databases: PKDD 2003, Lecture notes in Computer Science, Springer, Volume 2838, 2003, 303-314.

[19] Mehdi Khiari, Patrice Boizumault, and Bruno Cremilleux, "Combining CSP and Constraint-Based Mining for Pattern Discovery", Computational Science and its applications, ICCSA 2010 Lecture Notes in Computer science, Springer, Volume 601, Mar 2010, 432-447.

[20] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim, " SPIRIT: Sequential Pattern Mining with Regular Expression Constraints". In Proceedings of VLDB'99 25th International conference on very large databases, Morgan Kaufmann publishers, San Francisco, 1999, 223-234.

[21] Siegfried Nijssen, TiasGuns, and Luc De Raedt, "Correlated Itemset Mining in ROC space : A Constraint Programming Approach". In Proceedings of KDD'09, ACM, 2009, 647-656.

[22] Stefano Bistarelli, and Francesco Bonchi, "Soft constraint based pattern mining", Data and Knowledge Engineering, Elsevier, Volume 62, Issue 01, July 2007, 118-137.

[23] Synthetic graph generated by IBM Quest Synthetic Data Generation Code for Associations and Sequential Patterns. [http://www.7.ust.hk/graphgen/].

[24] Unil Yun, "Mining lossless closed frequent patterns with weight constraints", Knowledge-Based Systems, Elsevier, Volume 20, Issue 01, Feb 2007, 86-97.

[25] Wei Wang, Chen Wang, Yongtai Zhu, Baile Shi, Jian Pei, Xifeng Yan, and Jiawei Han, " GraphMiner: A Structural Pattern-Mining System for Large Disk-based Graph Databases and Its Applications". In proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, Newyork, 2005, 89-881.

[26] Xifeng Yan, X.Jasmine Zhou, and Jiawei Han, "Mining Closed Relational Graphs with Connectivity Constraints. In proceedings of KDD'05, ACM, Newyork, 2005, 324-333.

[27] Yen-Liang Chen, Ya-Han Hu, "Constraint-based sequential pattern mining: The consideration of recency and compactness", Decision Support Systems, Elsevier, Volume 42, Issue 02, Nov 2006, 1203-1215.