

LCGMiner: Levelwise Closed Graph Pattern Mining from Large Databases

Aihua Xu, Hansheng Lei

Computer Science and Engineering

University at Buffalo, State University of New York

Buffalo, NY 14260, USA

aihuaxu@cse.buffalo.edu, hlei@buffalo.edu

Abstract

LCGMiner (Levelwise Closed Graph Pattern Miner) is proposed to improve CloseGraph[2] in discovering frequent closed subgraphs. Frequent closed edgesets with the same extended vertexsets are expanded in pattern generation compared to one edge or one vertex in traditional methods. Experiments on synthetic datasets as well as a real NIH dataset demonstrates that our algorithm outperforms closeGraph and gSpan.

1 Introduction

CloseGraph[2] gives an efficient way to mine frequent closed graph patterns. However it requires multiple scans to generate child patterns and collect the supports. LCGMiner transforms edges into itemsets and reduces unnecessary patterns to be generated by mining closed itemsets and extending multiple edges of one level. The *support* of a graph pattern g , denoted $\sigma(g)$, is the number of graphs in which it occurs as a subgraph. A graph pattern is *closed* if there exists no proper superpattern $G \supset g$ with $\sigma(g) = \sigma(G)$.

2 Canonical Form

Vertices of graphs are traversed in BFS(Breadth First Search) order and grouped into multiple *levels*. We represent \circ as one BFS^+ traversal and denote *level*- k vertexset $\mathcal{V}_{(\circ,k)}$. Thus, $0 \leq k_2 - k_1 \leq 1$ for any edge $e = (v_i, v_j)$ where $v_i \in \mathcal{V}_{(\circ,k_1)}$ and $v_j \in \mathcal{V}_{(\circ,k_2)}$, i.e., edges only exist in the consecutive levels or the same level of vertices. If $k_2 - k_1 = 1$, we call e a *front edge*; a *cross edge* otherwise. An *level*- k edgeset $\mathcal{E}_{(\circ,k)} = \{(v_1, v_2) \mid v_1 \in \mathcal{V}_{(\circ,k)} \wedge v_2 \in \mathcal{V}_{(\circ,k+1)} \vee v_1, v_2 \in \mathcal{V}_{(\circ,k+1)}\}$.

Edge $e = (v_i, v_j)$ is mapped to a five-element tuple (i, j, l_i, l_e, l_j) [3] where l_i, l_e and l_j are labels of vertex v_i , edge e and vertex v_j respectively. Each BFS traversal produces a list of the tuples, called a *BFS Code*. A linear order

\prec is defined on each pair of edges (e_1, e_2) : $e_1 \prec e_2$ iff e_1 is a front edge and e_2 is a cross edge or $e_1 < e_2$ in lexicographic order. The minimum BFS code is unique and immutable, being suitable as the canonical form (graph representation) in the algorithm.

3 Pattern Generation

Let g be a pattern of database D with $level_{\circ} = k$ and S be an extendable edgeset, i.e., g is extendable with S in the order of \circ , denoted $g \oplus S$. S is an *extended edgeset* of g and $\mathcal{V}_{(g,f)} = \mathcal{V}_{(\circ_{g \oplus S}, k)}$ is a *frontier vertexset* of g . $\mathcal{V}_{(g,S,x)} = \mathcal{V}_{(\circ_{g \oplus S}, k+1)}$ is an *extended vertexset* of g . If $g \subseteq G \in D$, $\mathcal{V}_{(g,f)}|G = \{f(v) \mid v \in \mathcal{V}_{(g,f)}\}$ is an *frontier vertexset* of g in G , where f is an injective function from g to G . The **extended vertexset in G** , denoted $\mathcal{V}_{(g,x)}|G$, is the set of neighbor vertices of $\mathcal{V}_{(g,f)}|G$ but not in $\{f(v) \mid v \in \mathcal{V}_{(g,f)}\}$. The **extended edgeset in G** , denoted $\mathcal{E}_{(g,x)}|G$, is the set of edges (v_1, v_2) where $v_1 \in \mathcal{V}_{(g,f)}|G$ and $v_2 \in \mathcal{V}_{(g,x)}|G$ or $v_1, v_2 \in \mathcal{V}_{(g,x)}|G$. Projected database $D_g = \{\mathcal{E}_{(g,x)}|G \mid G \supseteq g \wedge G \in D\}$.

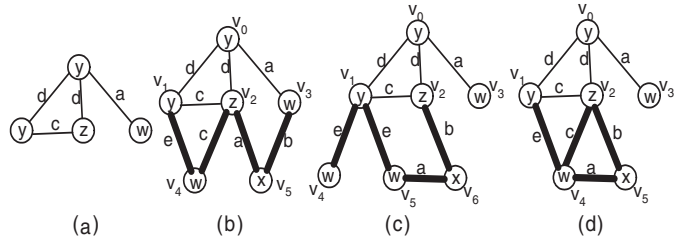


Figure 1. Pattern Transformation.

Theorem 1 (Non-closure pruning) Let g be any pattern with $level_{\circ} = k$. Let S' and S be two extended edgesets with $S' \subset S$. If $level_{\circ_{g \oplus S}} = level_{\circ_{g \oplus S'}} = k + 1$, $\mathcal{V}_{(g,S,x)} = \mathcal{V}_{(g,S',x)}$ and $\sigma(S) = \sigma(S')$ in D_g , then any descendant pattern of $g \oplus S'$ is not closed.

Proof. If $S' \subset S$ and $\sigma(S) = \sigma(S')$ in D_g , S' occurs where S exists in D_g , i.e., $g \oplus S$ and $g \oplus S'$ stay together in D . Further, $\mathcal{V}_{(g,S,x)} = \mathcal{V}_{(g,S',x)}$ implies that for any descendant pattern G of $g \oplus S'$, $G \subset G \oplus (S - S')$ and $\sigma(G) = \sigma(G \oplus (S - S'))$.

$\mathbf{R} = \{S \mid \text{same } \mathcal{V}_{(g,S,x)}\}$, i.e., \mathbf{R} is the set of edgesets where g has the same extended vertexset. If $S \in \mathbf{R}$ is not closed in \mathbf{R} , then $g \oplus S$ cannot lead to closed patterns. Thus, instead of enumerating edgesets in D_g as the naive method does, frequent closed ones in different \mathbf{R} s are extended and completeness is guaranteed.

Mining frequent closed edgesets in \mathbf{R} s reduces to mining frequent closed ones in D_g due to the following observation: let S be a frequent close edgeset in D_g , $\mathcal{V}_1 = \mathcal{V}_{(g,f)}$ and $\mathcal{V}_2 = \mathcal{V}_{(g,S,x)}$; for each $\mathcal{V}'_2 \subseteq \mathcal{V}_2$, necessary edgesets are obtained by enumerating possible edges in S between \mathcal{V}_1 and \mathcal{V}'_2 or among \mathcal{V}'_2 .

Extended edgesets are transformed to itemsets and CLOSET[4] is applied to mine closed edgesets. For each $S \in D_g$, edges $e = (v_1, v_2)$ are removed ('edge removal') where $v_1, v_2 \in \mathcal{V}_2$; conjunct vertex v_3 in \mathcal{V}_2 between edges $e_1 = (v_1, v_3)$ and $e_2 = (v_2, v_3)$ are broken ('vertex break'). Thus, D_g is transformed into D'_g .

CLOSET mines closed extended edgesets E'_g from D'_g and the frequent closed ones E_g in D_g is recovered by taking steps including vertex join, cross edge extension and pattern enumeration(closed in $D_g \Rightarrow$ closed in \mathbf{R}).

Child pattern set of g $C_g = \{g \oplus S \mid S \in E_g\}$. For patterns in C_g , pattern generation is called recursively to mine the next level of patterns.

4 Evaluations

Experiments were conducted to compare the performance of LCGMiner, closeGraph[2] and gSpan[3]. We are interested in finding frequent recurring substructures from the datasets which may be useful to characterize the classes and classify unknown data.

Synthetic datasets. We generated several datasets using the synthetic data generator in [1]. Fig. 2(a) shows the running time for different threshold values on dataset $D10kN40I16T20$ and fig. 2(b) displays the number of discovered frequent patterns compared to the number of frequent closed patterns. LCGMiner and CloseGraph both outperform gSpan by a fact of 0.2. Especially when the threshold is getting close to 3, LCGMiner runs faster by a fact of 0.5 than CloseGraph and gSpan. Also, as fig. 2(b) shows, the set of frequent closed patterns returned by LCGMiner and CloseGraph is much smaller than frequent ones.

NIH dataset. We also selected one real dataset – AIDS antiviral screen compound dataset from NCI/NIH¹. We per-

¹http://dtp.nci.nih.gov/docs/aids/aids_data.html

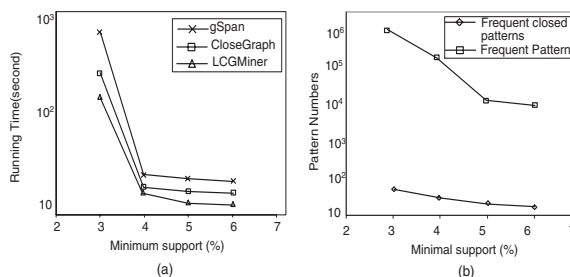


Figure 2. Performance on D10kN40I16T20

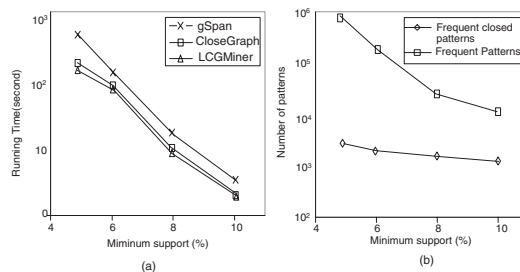


Figure 3. Performance for the NIH CA dataset

formed experiments on 422 CA (confirmed active) from March 2002 Release dataset. On average, CA compound has 40 vertices and 42 edges. The maximum one has 188 vertices and 196 edges. Fig. 3(a) shows running time with various minimum support and fig. 3(b) displays the number of frequent graph patterns and closed graph patterns. Fig. 3 shows LCGMiner outperforms gSpan and closeGraph.

5 Conclusions

LCGMiner extends closed patterns with the same extended vertexsets. Experiments demonstrate the efficiency compared with CloseGraph and gSpan.

References

- [1] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM*, 313–320, 2001.
- [2] X. Yan and J. Han. CloseGraph: Mining closed frequent graph patterns. In *KDD*, 2003.
- [3] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *ICDM*, 721–724, 2002.
- [4] J. Pei, J. Han and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *DMKD*, 21–30, 2000.