gSpan: Graph-Based Substructure Pattern Mining

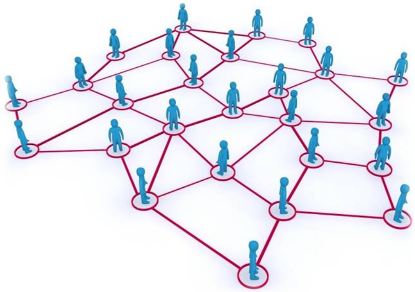
Xifeng Yan and Jiawei Han

University of Illinois at Urbana-Champaign

February 3, 2017

Agenda

- What motivated the development of gSpan?
- Technical Preliminaries
- Exploring the gSpan algorithm
- Experimental Performance Evaluation



Introduction and Motivation

Why are we interested in mining graphs?

- Social Network Data, Electronic Transactions (Fraud Detection)
- Chemical Compounds
- Computer vision, video indexing, text retrieval

Issues with Previous Graph Mining Approaches

- Previous benchmark approaches called AGM and FGS inefficient
- Take Apriori-like, level-wise approaches
- 1) Generate huge candidate sets
- 2) Require multiple scans of database
- 3) Difficult to mine long patterns

What does gSpan have to offer?

Apriori-based algorithms suffer from:

- Costly subgraph isomorphism test
- Subgraph candidate generation complicated and expensive

gSpan combats these issues by:

- **Eliminating candidate generation!**
- Applying Depth First instead of Breadth First search
- Introducing a new lexicographic canonical labeling system
- Combining isomorphism test and subgraph growth into one procedure

Preliminaries: Formal Problem Definition

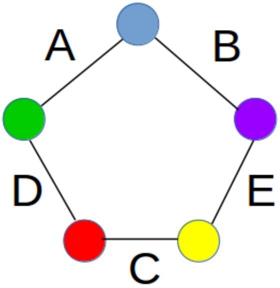
Given: A graph dataset $GS = \{G_i | i = 0, ..., n\}$, and *minsup*

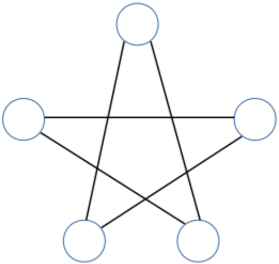
- Each G_i a labeled, undirected graph (V, E, L, l)
- *minsup* is the minimum support

Find: All g s.t. $\sum_{G_i \in GS} \zeta(g, G_i) \geq minsup$

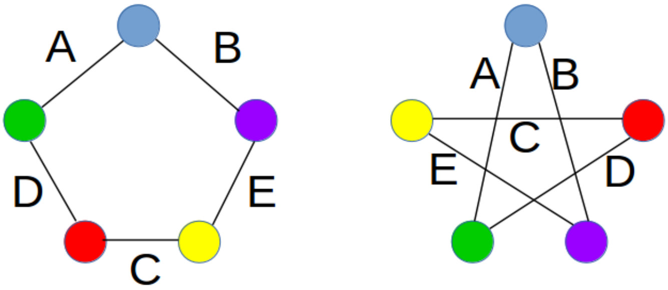
$$\zeta(g, G_i) = \begin{cases} 1 & \textit{if } g \textit{ isomorphic to a subgraph of } G, \\ 0 & \textit{if } g \textit{ not isomorphic to a subgraph of } G \end{cases}$$

Preliminaries: Graph Isomorphism





Are these graphs isomorphic?



Yes! An **NP**-complete Problem!

Preliminaries: DFS Lexicographic Ordering

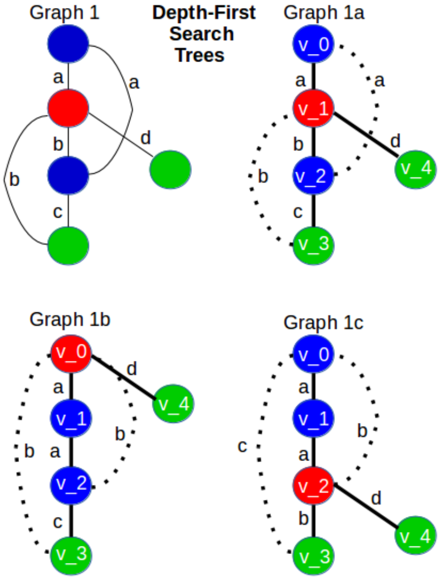
In class we have seen ordering used during itemset mining:

- Items in "baskets" lexicographically ordered A,B,...,F
- **Systematically** search for frequent itemsets
- Eliminates redundancy, improving efficiency

gSpan applies the same concept!!

- Each graph has a unique, ordered label
- Efficiently discover frequent subgraphs
- Prune the search space

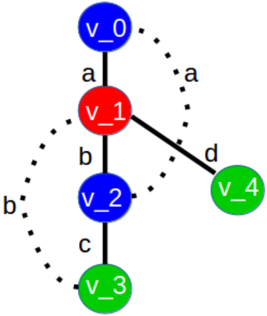
Preliminaries: DFS Lexicographic Ordering



Preliminaries: DFS Lexicographic Ordering

Depth-First Search Code

Graph 1a



Edge Order	Label
0	(0,1,B,b,R)
1	(1,2,R,b,B)
2	(2,0,B,a,B)
3	(2,3,B,c,G)
4	(3,1,G,b,R)
5	(1,4,R,d,G)

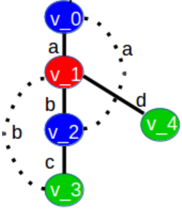
Let $e_1 = (i_1, j_1)$, $e_2 = (i_2.j_2)$

1) $e_1 \prec_T e_2$ if $i_1 = i_2$ and $j_1 < j_2$

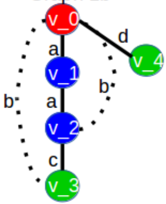
2) $e_1 \prec_T e_2$ if $i_1 < i_2$ and $j_1 = i_2$

Preliminaries: DFS Lexicographic Ordering

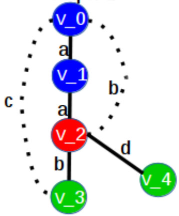
Graph 1a



Graph 1b



Graph 1c



Edge	Code 1a	Code 1b	Code 1c
0	(0,1,B,b,R)	(0,1,R,a,B)	(0,1,B,a,B)
1	(1,2,R,b,B)	(1,2,B,a,B)	(1,2,B,a,R)
2	(2,0,B,a,B)	(2,0,B,b,R)	(2,0,R,b,B)
3	(2,3,B,c,G)	(2,3,B,c,G)	(2,3,R,d,G)
4	(3,1,G,d,R)	(3,0,G,b,R)	(3,0,G,c,B)
5	(1,4,R,d,G)	(0,4,R,d,G)	(2,4,R,d,G)

Preliminaries: DFS Lexicographic Ordering

We have several DFS codes for the same graph!

- Fortunately, an ordering on DFS codes too!
- Specified by DFS Lexicographic Order
- In our case, Graph 1c $\prec$ Graph 1b $\prec$ Graph 1a

Graph 1c the **Minimum DFS Code** of Graph 1

- This is the unique, canonical label of Graph 1

Theorem 1

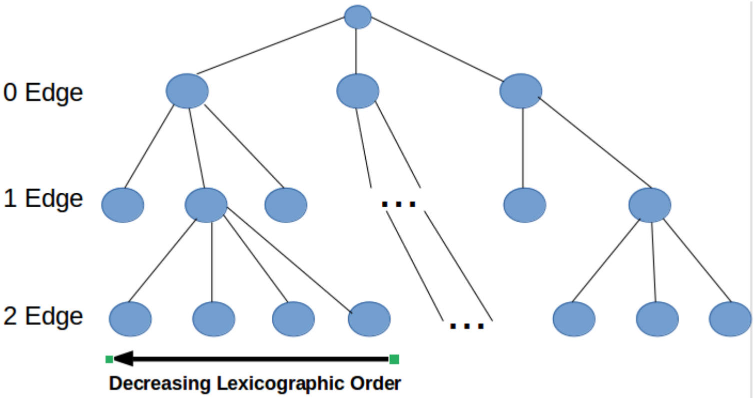
Given two graphs G and G', G is isomorphic to G' if and only if $min(G) = min(G')$.

Preliminaries: DFS Lexicographic Ordering

$$
\begin{array}{ll}
(i) & for\ some\ t, 0 \leqslant t \leqslant min\{m,n\},\ we\ have\ a_k = b_k\ for\ k < t,\ and^* \\
 & a_t < b_t = \begin{cases} true\ if\ a_t \in E_{\alpha,b}\ and\ b_t \in E_{\beta,f}. \\ true\ if\ a_t \in E_{\alpha,b},\ b_t \in E_{\beta,b},\ and\ j_a < j_b. \\ true\ if\ a_t \in E_{\alpha,b},\ b_t \in E_{\beta,b},\ j_a = j_b,\ and\ l_{(i_a,j_a)} < l_{(i_b,j_b)}. \\ true\ if\ a_t \in E_{\alpha,f},\ b_t \in E_{\beta,f},\ and\ i_b < i_a. \\ true\ if\ a_t \in E_{\alpha,f},\ b_t \in E_{\beta,f},\ i_a = i_b,\ and\ l_{i_a} < l_{i_b}. \\ true\ if\ a_t \in E_{\alpha,f},\ b_t \in E_{\beta,f},\ i_a = i_b,\ l_{i_a} = l_{i_b},\ and\ l_{(i_a,j_a)} < l_{(i_b,j_b)}. \\ true\ if\ a_t \in E_{\alpha,f},\ b_t \in E_{\beta,f},\ i_a = i_b,\ l_{i_a} = l_{i_b},\ l_{(i_a,j_a)} = l_{(i_b,j_b)},\ and\ l_{j_a} < l_{j_b}. \end{cases} \\
(ii) & a_k = b_k\ for\ 0 \leqslant k \leqslant m,\ and\ n \geqslant m.
\end{array}
$$

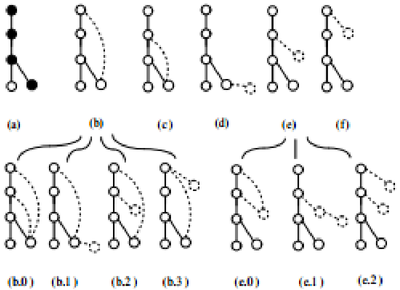
Determining the Minimum DFS Code Lexicographically

Preliminaries: DFS Lexicographic Ordering



DFS Code Tree: The "search space" of gSpan algorithm

Preliminaries: DFS Lexicographic Ordering



"Growing" a tree in DFS Lexicographic Order during search

Preliminaries: DFS Search Space

DFS Code Tree Covering

- The DFS Code Tree contains minimum DFS codes for all graphs

Frequency Antimonotone

- If graph G is frequent, any subgraph of G is frequent
- If graph G is infrequent, any graph containing G is not frequent

DFS Code Pruning

- Let the DFS Codes of G be α_0, α_1, ..., α_n with α_0 being $min(G)$. Pruning $\{\alpha_i : 1 \leq i \leq n\}$ preserves DFS Code Tree Covering.

Algorithm 2 GraphSet_Projection($\mathbb{GS}$,$\mathbb{S}$).

sort labels of the vertices and edges in $\mathbb{GS}$ by their frequency;
remove infrequent vertices and edges;
relabel the remaining vertices and edges in descending frequency;
$\mathbb{S}^1 \leftarrow$ all frequent 1-edge graphs in $\mathbb{GS}$;
sort $\mathbb{S}^1$ in DFS lexicographic order;
$\mathbb{S} \leftarrow \mathbb{S}^1$;
for each edge $e \in \mathbb{S}^1$ **do**
    initialize s with e, set $s.GS = \{g \mid \forall g \in \mathbb{GS}, e \in E(g)\}$; (only graph ID is recorded)
    Subgraph_Mining($\mathbb{GS}$, $\mathbb{S}$, s);
    $\mathbb{GS} \leftarrow \mathbb{GS} - e$;
    if $|\mathbb{GS}| < minSup$;
        break;

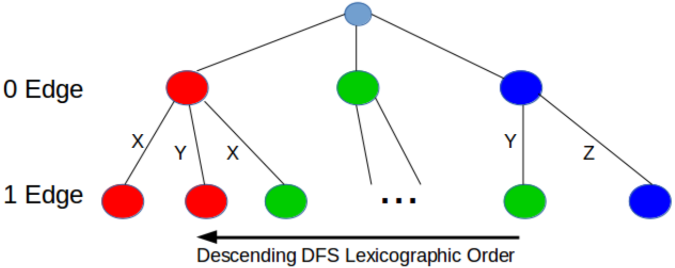
The gSpan Algorithm

Steps 1-2: Remove infrequent vertices and edges

- Begin pruning dataset early!
- Clearly if edge e or vertex v are ¡ *minsup*, any graph G containing e or v is guaranteed to be infrequent!

Step 3: Relabel edges and vertices in descending order

- Need an ordering on labels to determine DFS Lexicographic Order on DFS Codes
- $V = \{$Red, Green, Blue$\}$, $E = \{$X,Y,Z$\}$



Steps 4-6: Sort frequent 1-edge graphs in DFS Lexicographic order

Steps 7-9: Loop over frequent 1-edge subgraphs

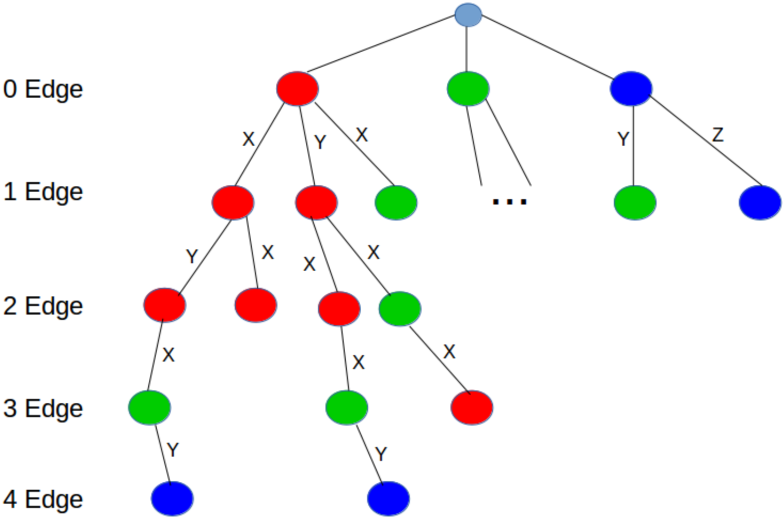
- Begin with least edge in lexicographic order
- Set s.GS $= G \in GS$ where $e \in E(G)$
- Call $SubgraphMining(GS, S, s)$;

The gSpan Algorithm

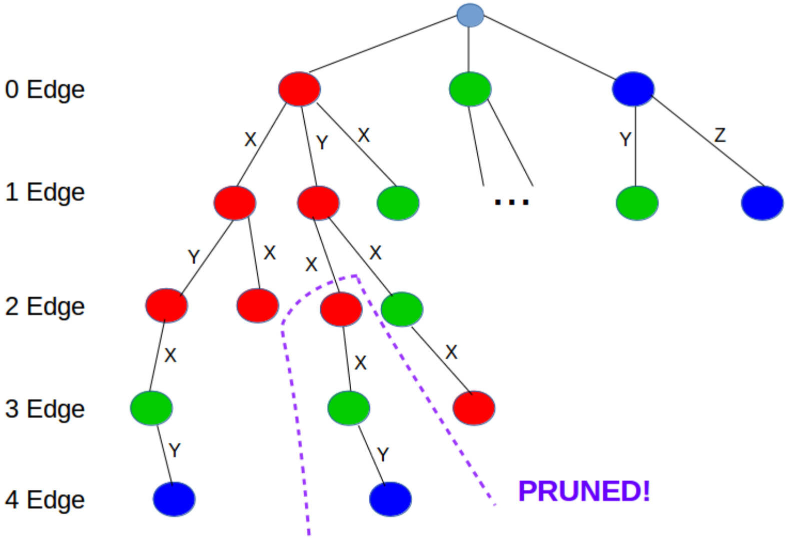
Subprocedure 1 Subgraph_Mining($\mathbb{GS}$, $\mathbb{S}$, s).

if $s \neq min(s)$
    return;
$\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$;
generate all s' potential children with one edge growth;†
Enumerate(s);
for each c, c is s' child **do**
    if $support(c) \geq minSup$
        $s \leftarrow c$;
        Subgraph_Mining($\mathbb{GS}$, $\mathbb{S}$, s);

The gSpan Algorithm



The gSpan Algorithm



Step 11: $\mathcal{D} \leftarrow \mathcal{D} - e$

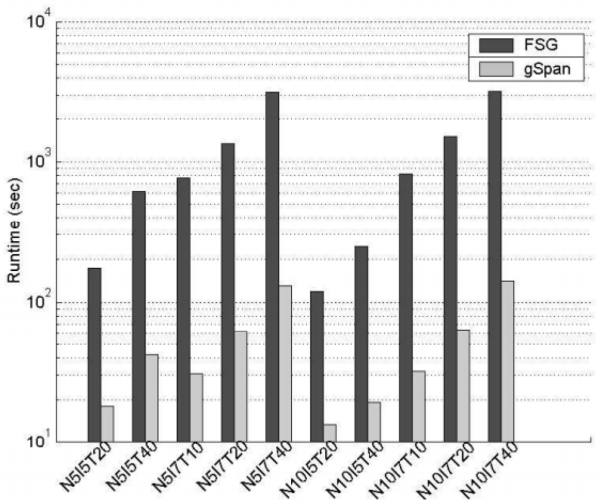
- Very important step!
- Because of Lexicographic DFS, have already mined ALL subgraphs containing edge e
- Remove all instances of edge e from graph dataset
- Mining process takes less time for later iterations
- Drastically improves efficiency!

Experimental Performance Results

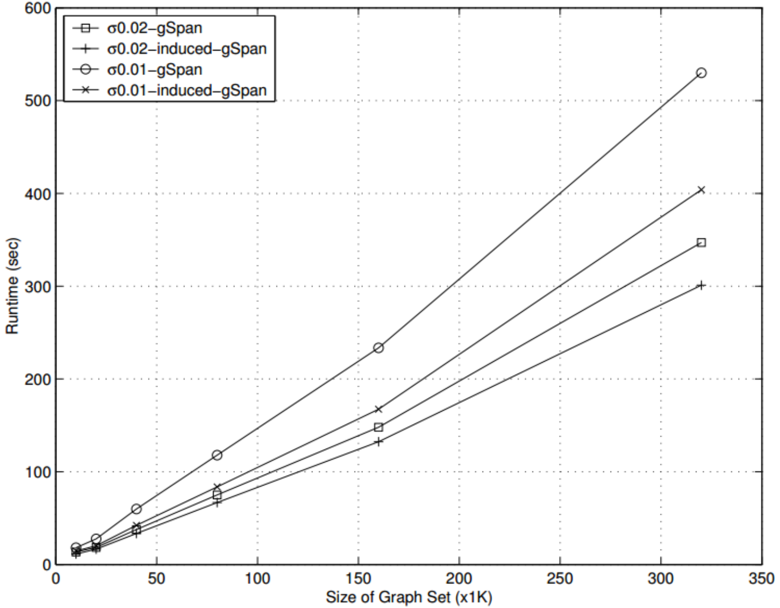
Tested on synthetic and real (Chemical Compounds) datasets

- Tested against previous mining benchmark FSG algorithm
- Performs 6-45x faster on synthetic data
- Performs **15-100x** better on chemical compound data

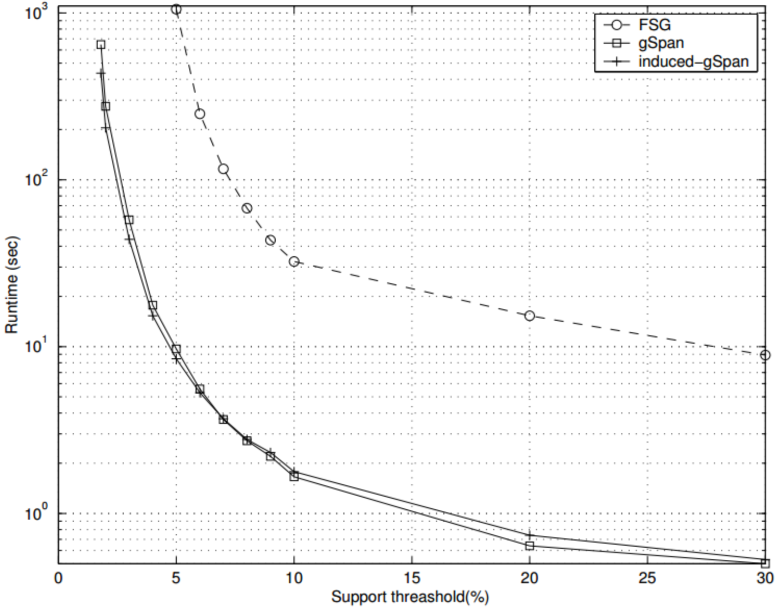
Experimental Performance Results



Experimental Performance Results



Experimental Performance Results



Conclusion

gSpan sets new benchmark for mining frequent graphs

- No candidate generation! Reduce expensive false candidate testing
- Introduce DFS Lexicographic order to systematically SEARCH and PRUNE search space
- Depth-first as opposed to Breadth-first search
- Possible can fit in main memory, reduce IOs
- Shrink graph dataset with each iteration!