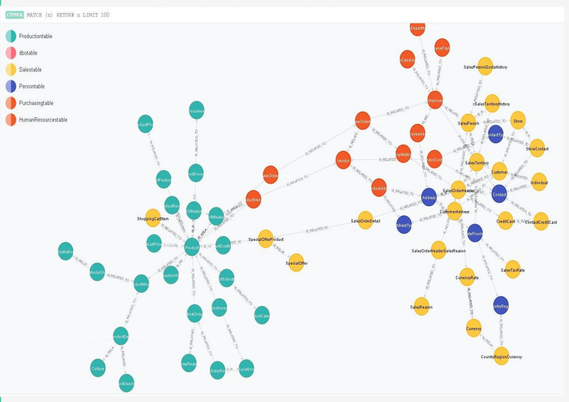
FREQUENT GRAPH MINING FOR neo4j

Roudolf Yves MBONYA NDUMBE
Politecnico di Torino, ITALY

Introduction

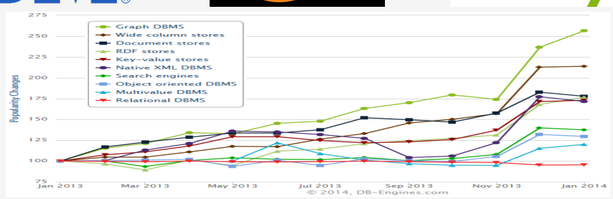
Frequent pattern mining has been an active research theme in data mining with many efficient and scalable techniques.



However many scientific and commercial applications require to find frequent patterns in large datasets represented as graphs. Hence there is a need for **frequent graph mining**.

To store graphs, one recent development is the use of graph databases. One such database is **neo4j**.

Neo4J is a graph database management system developed by Neo4j, inc in 2002. It is an **ACID –** compliant transactional database with native graph storage and processing, implemented in **Java** and accessible from software written in other languages using **Cypher Query Language** through a transactional HTTP endpoint, or through the binary 'bolt' protocol. Graph databases are gaining popularity and their integration with data mining algorithms could be interesting. Neo4j is used my many giant companies today such as *IBM, Amazon, ebay*, …etc to manage their database.

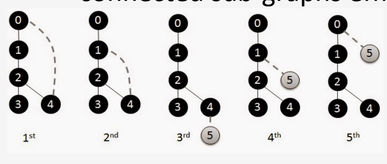


Frequent Graph Mining is the is the search for frequent graph-based patterns (sub-graphs) in graph datasets.

Procedure

- Selecting an efficient algorithm for mining frequent sub-graphs in **java**–**gSpan Algorithm.**

gSpan is a graph-based substructure pattern mining algorithm which discovers frequent substructures without candidate generation by building a lexicographic order, adopting a DFS strategy and following a growth priority to mine frequent connected sub-graphs efficiently.



- Creating graph databases (to obtain test cases on which to run **gSpan**) and using the java driver to access the **neo4j** server and manage it using its **Cypher Query Language**.
- Connecting the **gSpan** algorithm to the neo4j server to search for frequent sub-graph using the java driver. In other to do this I will need to manage some parameters, for instance;

Labeling

Vertex Symbol	Vertex Label	Edge Symbol	Edge Label
α	A	—	a
β	B	—	b
α	C	·····	c

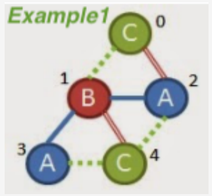
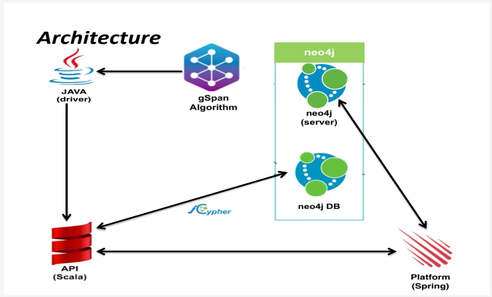


Table of Example1

Edge Number	DFS Code
0	(0,1,C,c,B)
1	(1,2,B,a,A)
2	(2,0,A,b,C)
3	(1,3,B,a,A)
4	(3,4,A,c,C)
5	(4,1,C,b,B)
6	(4,2,C,c,A)

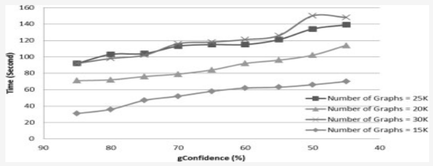
- Establishing the data used by gSpan, by using a DFS code to document the vertices and edges of the different graphs in tables by first labeling them for use by gSpan.
- Sorting the data in descending order

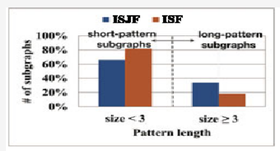
- Building a web interface to "talk" with the server and database using **Spring Data for Neo4j** – this is a programming template which builds the basis for interaction with the graph database.



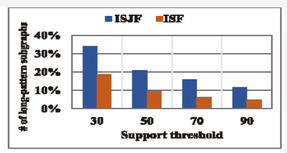
Expected Results

- A system that allows to run a graph mining algorithm on a graph database given a "language" to specify constraints.
- The system will be evaluated on how scalable the approach used is and the effect of t the constraints applied(min. support, pattern length) on the algorithm to its running time.





(i) # of subgraph patterns *w.r.t.* pattern length.



(ii) # of long-pattern subgraphs *w.r.t.* support threshold.

Conclusions

Graph databases nowadays are not the most popular databases because of the great complexity in implementing them but as time passes its popularity and use grows due to its efficiency and speed in managing large databases. Using **gSpan** to analyze graphs is promising, as it offers practical efficiency compared other graph miners.