Graph Algorithms in Neo4j using CYPHER

P.Balaghan & K.A.Hawick
Computer Science Department, University of Hull

UNIVERSITY OF Hull

Aim

Can complex Graph-Based Algorithms be implemented into a Graph Database such as Neo4j, using only a language created for the purpose of interacting with the database?

Introduction

Graph Databases can store data in a graph-like structure. This allows the opportunity to implement advance graph algorithms into the database to find new and interesting facts about the data. Traditional SQL databases could allow some graph-like functions by using the JOIN command to bring various tables together, however this is computationally expensive and does not scale well. (Robinson et al., 2013)

Neo4j and CYPHER

Neo4j is a JVM-based Graph Database and follows the Property Model. A Property Model Graph Database contains Nodes (N), Edges (E) and Properties (P) such that:

$$G = (N,E,P)$$

Properties are Key-Pair items which can be attached to both Nodes and Edges. (Angles and Gutierrez, 2008) CYPHER is a Domain Specific Query Language used to interact with a Neo4j Database. It is written in a combination of Scala and Java and uses ASCII-like art within its language to define graph queries.

```
MATCH (n)-[r]->()
RETURN n
```

Figure 1: An example CYPHER query to find all outgoing nodes from node "n". The parenthesis surrounding the 'n' represent a node. The square brackets around the 'r' represent relationships. The query would find all nodes and relationships within the graph

The query in fig. 1 essentially states "find all nodes which have a direct, outgoing relationship with node n". (Vukotic and Watt, 2015)

Algorithms Used

A selection of algorithms were used and they ranged from the very basic, such as the average degree of the graph, to the complex, such as finding the Minimum Dominating set of a Graph.
One of the most basic graph functions is finding the degree of a node. This is the amount of edges which are either going into, or out of a particular node. The average degree of a graph (Equ. 1) can be an indicator to how complex a graph is.

$$\langle k \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} k_i \tag{1}$$

Betweenness Centrality finds the most critical node in a graph. It has previously been used to assess power and water network robustness (Hawick, 2012a) (Hawick, 2012b). It can be calculated by finding all shortest paths within a graph, and if any immediate nodes within a path exist, they are given a weight. Equ. 1 shows a way of doing this. (Freeman, 1977)

$$g_k = \sum_{i \neq j} \frac{C_k(i,j)}{C(i,j)} \tag{2}$$

One of the more complex Algorithms for a Graph is finding the Minimum Dominating Set. This is the minimum amount of Nodes required to have an edge connection to every other node in the database. This can be done by using the greedy algorithm (Parekh, 1991) in Alg. 1.

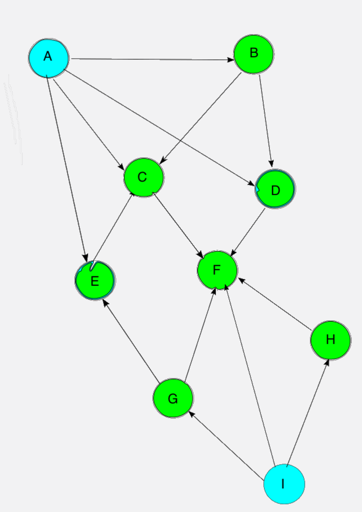

Figure 2: Minimum Dominating Set Of a Graph. The Blue nodes represent the dominating nodes, while the Green nodes represent the dominated node

Algorithm 1 Minimum dominating set of a Graph

$S := \emptyset$
while $\exists$ *white nodes* **do**
  choose $v \in \{x | w(x) = max_{u \in V}\{w(u)\}\}$
  $S := S \cup v$
end while

Other algorithms implemented into CYPHER include Amount of Triangles in a Graph, ShortestPath Algorithm, the allShortestPaths algorithm and the Graph Diameter.

Results

Some algorithms can be converted into CYPHER. One such algorithm is the average degree. In Fig. 1, it shows how this was done. The query gets a count of all of the nodes,and stores it in $countOfNodes$, then it gets the total count of relationships and stores it into $countOfRelationships$. The $countOfRelationships$ are then divided by the $countOfNodes$ to produce the Average Degree of the Graph.

```
MATCH(n)
WITH count(n) as countOfNodes
MATCH (n)-[r]-()
WITH count(r) as countOfRelationships,
countOfNodes
RETURN (tofloat(countOfRelationships) /
tofloat(countOfNodes))
```

Figure 3: The Average Degree of a Node in CYPHER for a Neo4J Graph

Once more complex algorithms where implemented into CYPHER, limitations were found. CYPHER struggled with the concept of recursive algorithms. While the queries themselves are recursive, algorithms such as the Graph Dominating Set require specific sections of code to be recursive. CYPHER was not able to do this. The table in fig.4 shows a break-down of the various algorithms used, and whether they were successfully implemented into CYPHER.

Algorithm	Successfully Implemented
Average Degree	Yes
Betweenness Centrality	Yes
Triangles in Graph	Yes
Shortest Path	Yes
All Shortest Paths	Yes
Graph Diameter	Yes
Graph Domination	No
Clustering Coefficient	Yes
Component Labelling	No

Figure 4: A table showing the final results of the experiment

Conclusion

The results show that some algorithms can successfully be converted into CYPHER and run on a Neo4j Graph Database. However, once more complex algorithms are introduced, it does not currently have the capability to implement these algorithms. These limitations prevent further complex algorithms from being implemented into CYPHER. Future work involves comparing the performance of these algorithms in CYPHER with those from a high level language,and comparing the complexity of these algorithms when ran through CYPHER.

References

Angles, R. and C. Gutierrez (2008). Survey of graph database models. *ACM Computing Surveys 40*(212), 1–39.

Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness.

Hawick, K. A. (2012a). Betweenness centrality metrics for assessing electrical power network robustness against fragmentation and node failure. *Proc. International Conference on Power and Energy Systems (EuroPES 2012)*, 186–193.

Hawick, K. A. (2012b). Water Distribution Network Robustness and Fragmentation using Graph Metrics Water Distribution Network Robustness and Fragmentation using Graph Metrics.

Parekh, A. K. (1991). Analysis of a greedy heuristic for finding small dominating sets in graphs. *Information Processing Letters 39*(5), 237–240.

Robinson, I., J. Webber, and E. Eifrem (2013). *Graph Databases*.

Vukotic, A. and N. Watt (2015). *Neo4j In Action* (First Edit ed.). Shelter Island: Manning Publications co.