 Copyright © Objectivity, Inc. 2017

Objectivity ThingSpan Subgraph Operations

1. Background

Objectivity ThingSpan has an advanced pathfinding capability for operating on graphs and subgraphs. The DO declarative graph and conventional query processing language provides both navigational, pathfinding and vertex/edge specific queries. However, there are many other subgraph operations that could be provided. Mathematical packages, such as Mathlab™, offer a wide variety of such operations, but they generally only work if the subgraph(s) can all fit in memory, which limits their scalability. Other packages, such as Apache Spark GraphX, have been designed primarily for social network analysis, but they are generally hampered by being based on tabular structures rather than networks of objects.

2. Description of the Problem

There are two significant problems:

a. Although Objectivity ThingSpan is closely integrated with Spark GraphX, we need a way to perform operations on subgraphs without having to rely on Spark, which won't be available in many common configurations.
b. Complex queries can return a set of objects or a subgraph. The results can be stored as a collection, but there is no simple way to compare the results of two or more queries.

3. How is this problem being solved now and why isn't that acceptable?

Users have no option other than developing their own algorithms, many of which won't be as efficient as having the Objectivity/DB kernel embedded within ThingSpan use low level structures to speed up some of the operations, such as counting the number of links to an object without having to visit the related vertices. There is also the possibility that common customer application development tasks will be missed, leading to wasted effort.

4. Overview of the Required Features

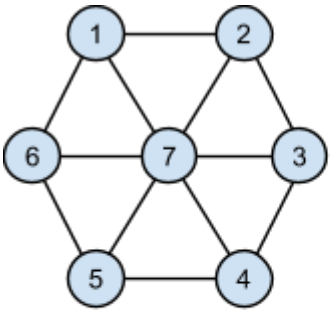
The following sets of subgraph capabilities are required:

- Subgraph Definition
- Commonality, Subtraction and Addition
- Similarity and Differences
- Span, Betweenness and Centrality
- Cliques (smallest and largest) and Subsets
- Combining and Pruning.

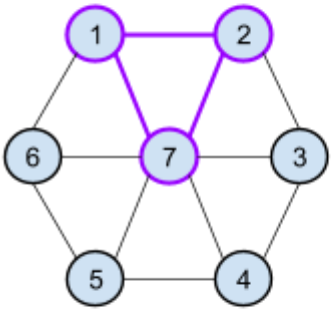
Copyright © Objectivity, Inc. 2017
Commercial-in-Confidence

5. Description of the Requested Features

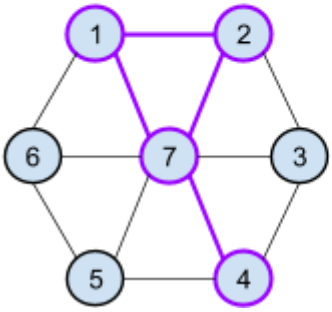
5.1 Subgraph Definition



Graph = {1,2,3,4,5,6,7} // Specify vertices
Graph = {1-2,2-3,3-4,4-5,5-6,6-1,7-1,7-2,7-3,7-4,7-5,7-6} // Edges
Graph = {7+ || 1} // 7 To a depth of 1
Graph = {1 + || 2} // 1 (or 2 to 6) to a depth of 2



Subgraph = {1,2,7} // Vertices
Subgraph = {1-2, 2-7,7-2} // Edges
Subgraph = {7-1,7-2}+ || 1} // Edges extended to a depth of 1



Subgraph = {1,2,7,4} // Vertices
Subgraph = {1-2,2-7,7-1,7-4} //Edges

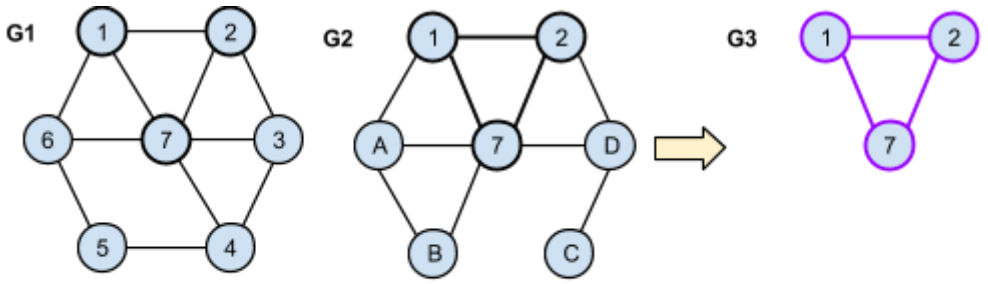
a. It must be possible to define graphs or subgraphs in at least these ways:
    i. As a list (collection) of vertex objects.
    ii. As a list (collection) of edge objects
    iii. As a subgraph collection (vertices and edges),
    iv. As a collection of vertices plus vertices linked to vertices in the subgraph extended out to N hops. As an example, extend(7, 1) in the diagram above results in the complete graph, involving all seven vertices.
    v. As a collection of edges plus vertices and edges linked to them extended out to N hops. As an example, extend(1-2, 2-7, 7-2) in the diagram above also results in the complete graph, involving all seven vertices.

Copyright © Objectivity, Inc. 2017
Commercial-in-Confidence

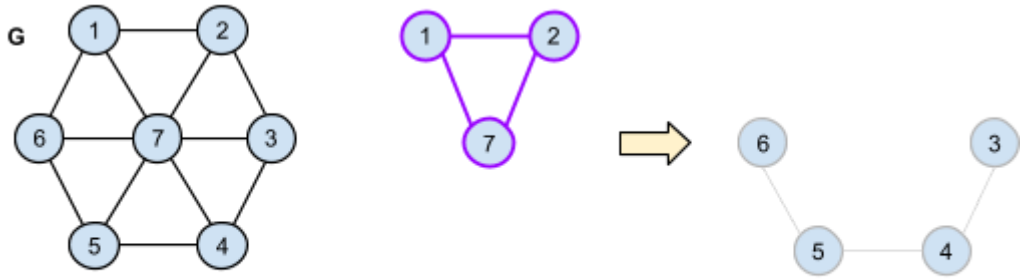
5.2 Subgraph Commonality, Subtraction and Addition

The new feature is to provide at least the following algorithms or results, where subgraph can also represent a complete graph:

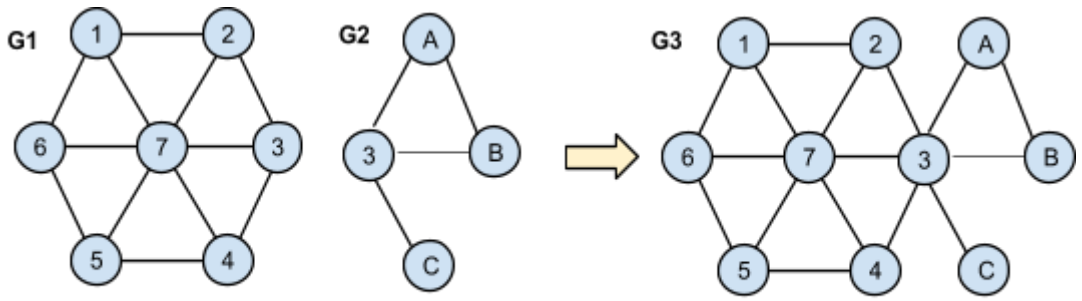
a. Find common subgraphs of two subgraphs.
b. Subtract subgraph A from subgraph B. Subgraph A must have an exact match in subgraph B. The Difference operation does not have this restriction.
c. Add a subgraph to a subgraph, merging identical vertices and edges.



Common (G1, G2) => { 1,2,7 } => G3



Subtract (G, {1,2,7}) => {3,4,5,6}

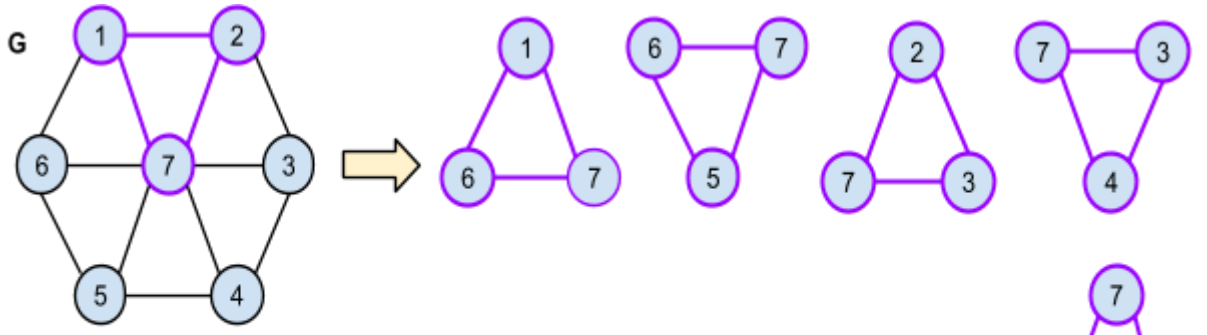


Add (G1, G2) => { G3 }

Copyright © Objectivity, Inc. 2017
Commercial-in-Confidence

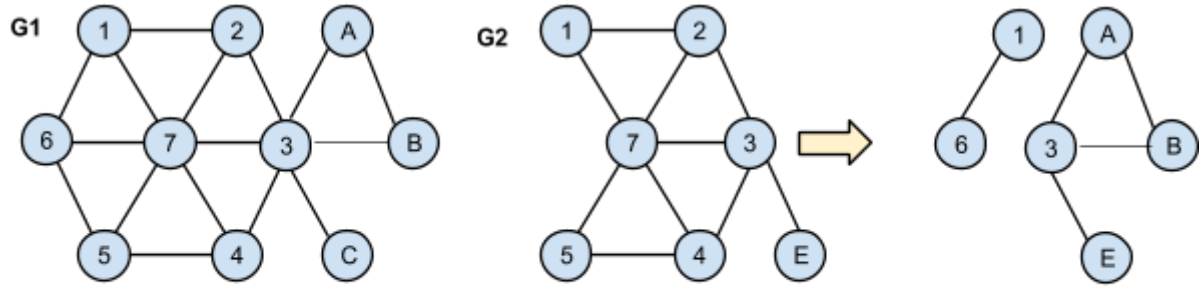
5.3 Subgraph Similarity and Differences

d. Find similar subgraphs in a graph, where similar means that there are vertex and edge combinations that have exactly the same topology. In the example below, the subgraphs all have 3 vertices that are each connected to exactly 2 other vertices.

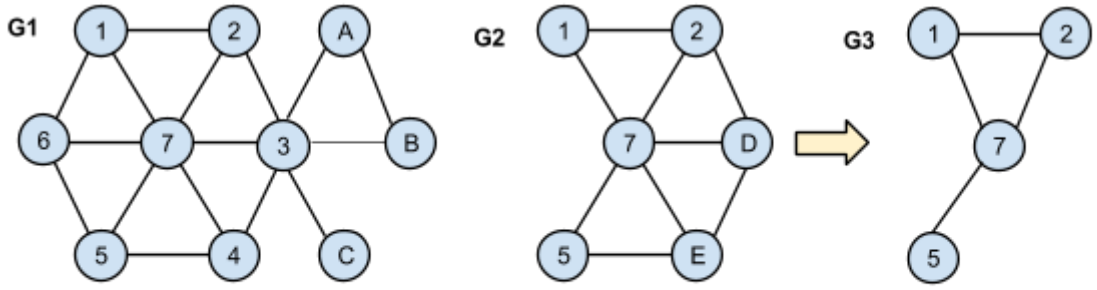


Like ({1,2,7}) => { {1,6,7}, {5,6,7}, {2,3,7}, {3,4,7}, {4,5,7} }
Like (7) => Null
Like (6) => { 1 }, { 2 }, { 3 }, { 4 }, { 5 }

e. Find the difference ("diff") between two subgraphs.
f. Find vertices/edges and subgraphs that two subgraphs have in common.



Diff (G1, G2) => { 1,6 }, { A,B,3,E }

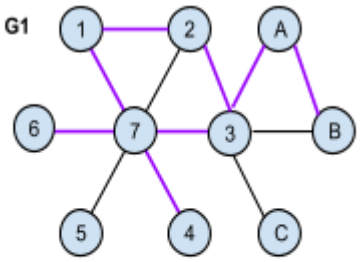


Common (G1, G2) => { 1,2,5,7 } => G3

Copyright © Objectivity, Inc. 2017
Commercial-in-Confidence

5.4 Subgraph Span, Centrality and Betweenness

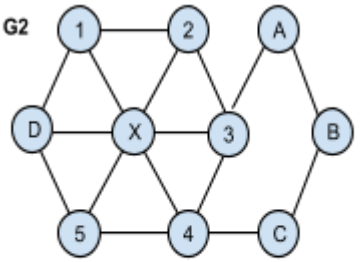
g. Find the minimum and maximum span of a subgraph, where the span is measured between the closest and most distant leaf vertices, using the edge counts on a path.
h. Find the degree centrality of a vertex.
i. Find the vertex with the highest centrality, e.g. for page ranking.
j. Find the Betweenness of two vertices.



Maximum Span = 6, i.e. 6 -> 7 -> 1 -> 2 -> 3 -> A -> B,

Minimum Span = 4, i.e. 6 -> 7 -> 3 -> C

The Betweeness of 6 and 3 is 2 and of 6 and B is 3.



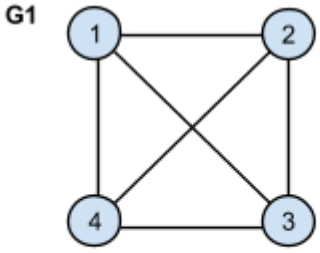
The Degree Centrality of node 1 is 3.

The Degree Centrality of node A is 2

Node X has the highest Degree Centrality (6)

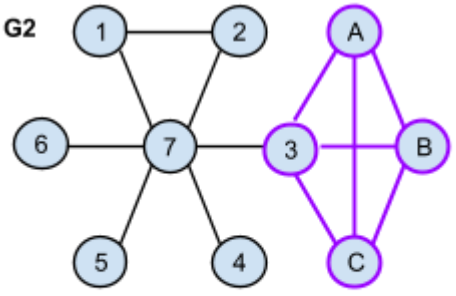
5.5 Cliques and Subsets

k. Determine if a subgraph is a clique, i.e. every vertex is connected to all other vertices.
l. Find the smallest (minimal) clique in a subgraph.
m. Find the largest (maximal) clique in a subgraph.



{1,2,4}, {1,2,3}, {2,3,4} and {1,3,4} are all minimal cliques, with a clique count of 3.

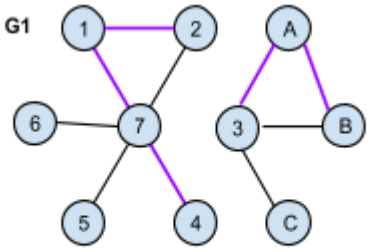
{1,2,3,4} is a maximal clique.



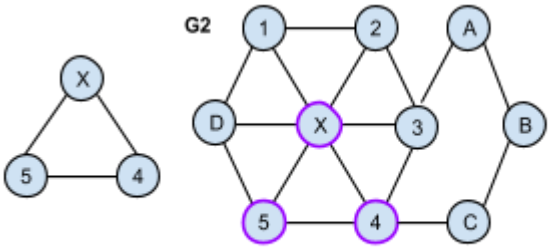
{A,B,C,3} is a maximal clique.

Copyright © Objectivity, Inc. 2017
Commercial-in-Confidence

n. Determine if a subgraph is actually a complete graph.
o. Determine whether a graph is a true subset of another subgraph.



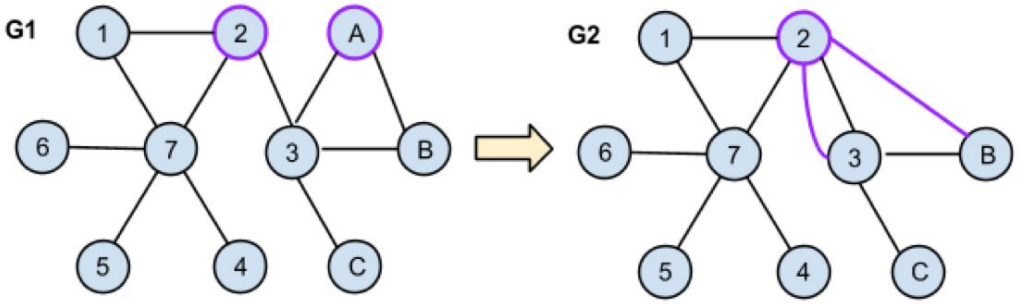
{3,A,B,C} is actually not a part of graph G1, it is a detached graph.



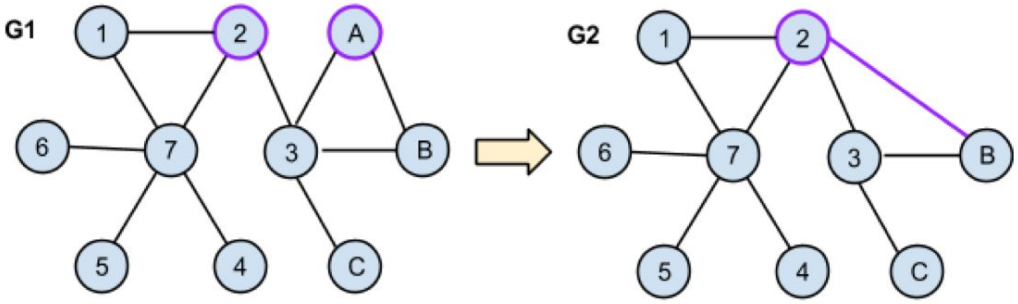
{5,X,4} is a subgraph of G2

5.4 Combining and Pruning Subgraphs

p. Combine two vertices, moving all edge connections to the new (or surviving) vertex.
q. Combine two vertices, eliminating any redundant edges.



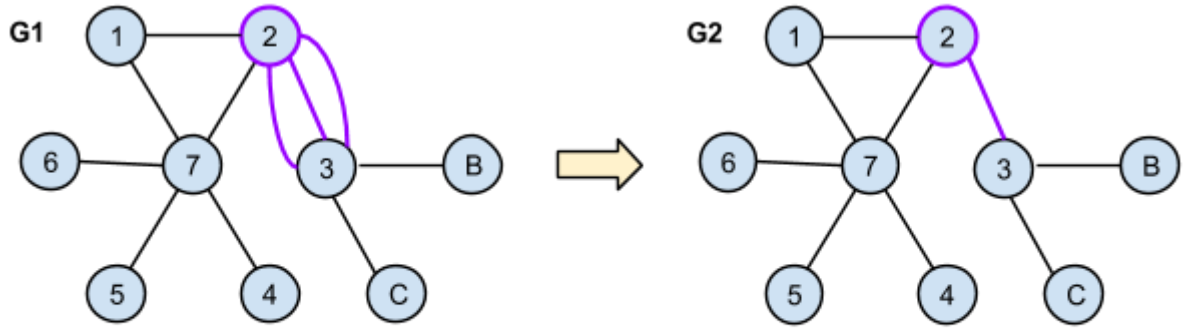
Combine vertices { 2 } and { A } => leaving only { 2 } but preserving all edges



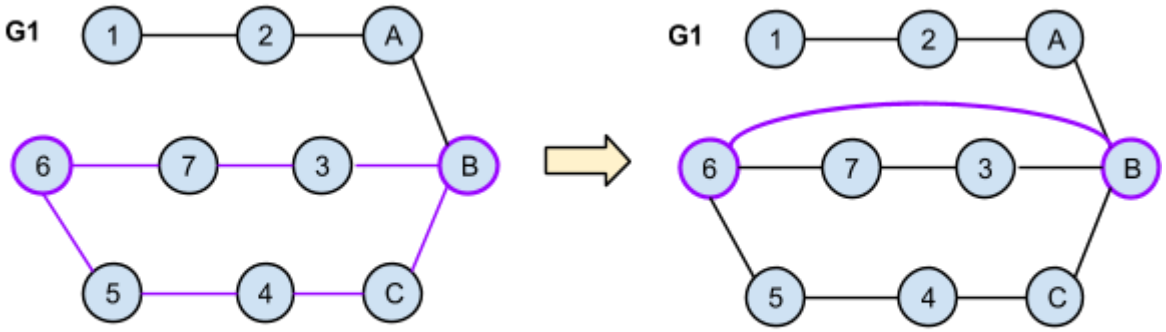
Combine vertices 2 } and { A } removing redundant edges => leaving only { 2 }

Copyright © Objectivity, Inc. 2017
Commercial-in-Confidence

r. Combine two or more edges connecting the same pair of vertices.
s. Combine two or more paths with the same endpoints and no tributary edges into a single edge.



Combine two or more edges connecting the same vertices (2 and 3)



Create a new edge representing two or more paths having the same endpoints

6. What languages must support this capability?

- Java

7. Which platforms must be supported?

- Linux
- Solaris (later)

8. Do any competitors already have this feature?

- Some graph databases support various combinations of these features, but not at scale.

Copyright © Objectivity, Inc. 2017
Commercial-in-Confidence

9. Customers who require this feature

- Any customer wanting to perform pattern analysis on a very large graph structure .

10. Related Material

We will also need to provide:

- Quality Assurance material.
- Documentation
- Professional Services
- Custom Support services

11. Implementation Guidelines

11.1 Goals

There are three primary goals:

- Safe, high quality, low cost, fastest time to market.
- High performance, massive scalability and optimal physical storage efficiency.
- Minimal disruption to the Objectivity/DB kernel and ThingSpan DO API.

11.2 Priorities

These operations can be implemented as a second deliverable:

d. Find similar subgraphs in a graph.
g. Find the minimum and maximum span of a subgraph
l. Find the smallest (minimal) clique in a subgraph.
m. Find the largest (maximal) clique in a subgraph.

11.3 Persistence

Some of the operations could optionally make permanent changes to the graph, specifically - subtraction, addition, combination and pruning.

12. Additional Notes

There is a separate Market Requirements Document covering features for finding "*islands*", i.e. subgraphs separated from the rest of a graph or subgraph.