Graphium Chrysalis: Exploiting Graph Database Engines to Analyze RDF Graphs

Alejandro Flores, Maria-Esther Vidal, Guillermo Palma

Universidad Simón Bolívar, Caracas, Venezuela
{aflores,mvidal,gpalma}@ldc.usb.ve

Abstract. We present Graphium Chrysalis, a tool to visualize the main graph invariants that characterize RDF graphs, i.e., graph properties that are independent of the graph representation such as, vertex and edge counts, in- and out-degree distribution, and in-coming and out-going h-index. Graph invariants characterize a graph and impact on the cost of the core graph-based tasks, e.g., graph traversal and sub-graph pattern matching, affecting time and space complexity of main RDF reasoning and query processing tasks. During the demonstration of Graphium Chrysalis, attendees will be able to observe and analyze the invariants that describe graphs of existing RDF benchmarks. Additionally, we will show the expressiveness power of state-of-the-art graph database engine APIs (e.g., Neo4j or Sparksee[1]), when main graph invariants are computed against RDF graphs.

1 Introduction

Graph invariants are properties that remain the same under two isomorphic graphs and any representation of the graph. These graph measures not only characterize bound complexity of some core graph-based tasks, but also some of them provide remarkable information on the graph topology, which allows to uncover hidden properties of the relationships modeled in the graph. For example, vertex and edge counts, adjacency distribution, in- and out-degree distributions, treewidth, betweenness centrality, reciprocity, and h-index are well-known invariants. Based on these invariants, complexity and graph topology properties can be formally described [1]. Computing graph invariants requires to traverse the whole graph to aggregate vertex and edge statistics as well as to identify certain patterns between these objects. Although some RDF query and reasoning complexity problems have been defined in terms of graph invariants [3], because existing RDF triple stores rely on tailored data structures to manage sub-graph pattern matching queries, they usually do not offer support to implement the computation of these measures efficiently. Different engines have been developed to manage, store and query graph databases (e.g., Neo4j [4] or Sparksee [2]). Each graph database engine implements general data structures and usually relies on indices to speed up execution time; additionally, some engines make available APIs comprised of methods to solve core graph-based tasks that facilitate the implementation of graph invariants. We present Graphium Chrysalis, a visualization tool that exploits different graphical representations to report on the results of evaluating a variety of RDF graphs, and graph invariants

[1] Sparksee was previously known as DEX.

implemented on top of Neo4j and Sparksee. Visualization techniques used in Graphium Chrysalis facilitate the understanding of graph invariants and the structure of RDF graphs generated by state-of-the-art benchmarks, e.g., the Berlin Benchmark. During the demonstration attendees will go through the visualization of different patterns in the measurements of the studied RDF graphs that will allow them to uncover the properties of existing benchmarks, as well as to identify the impact that these properties may have on the complexity of typical RDF data management tasks. Graphium Chrysalis is part of the Graphium project which has the goal of defining benchmarks for graph database engines. The demo is available at `http://graphium.ldc.usb.ve/chrysalis/`.

2 The Graphium Chrysalis Architecture

Graphium Chrysalis is built on top of existing graph database engine APIs to compute graph invariants on-the-fly efficiently; additionally, Graphium Chrysalis exploits visualization services implemented in JavaScript, to identify patterns between the values of different graph properties and the impact that they may have on main core graph-based tasks. Furthermore, we publish a Java library at the Graphium project website[2], thus, users will be able compute graph invariants locally. Figure 1 shows the Graphium Chrysalis GUI. In the area enclosed in red rectangle number 1, users can select different RDF graphs, and upload other graph results computed with our tool. This will allow users to understand: *i*) characteristics of a graph that impact on the values of the invariants; *ii*) time and space required to load a graph; and *iii*) time and space complexity of main core graph-based tasks, e.g., k-hops or shortest paths. Results are visualized in the area enclosed by the blue rectangle number 2; additionally, we plot the distributions of invariants such as in- and out-degree, and vertex and edge counts. Graphium Chrysalis exploits visualization capabilities of the HeatMaps[3] to compare in- and out-degree distributions and the cost of core graph-based tasks. Thus, attendees will be able to observe the relationships between the cost and the invariant measurements of an RDF graph.

3 Demonstration of Use Cases

We consider different RDF graphs generated by the Berlin SPARQL Benchmark[4]; additionally, attendees will be able to upload any other graph results during the demonstration. The goal of the demonstration is to visualize patterns that can be found in the invariants of these RDF graphs and the relationships between these measures and the cost of core graph-based tasks, e.g., graph adjacency, sub-graph pattern matching, graph traversals, and query processing.

3.1 Graph Invariants and Property Statistics

Among the graph invariants to be considered are the following:
Vertex and Edge Counts: outputs the number of vertices (i.e., graph order) and number of edges (i.e., graph size) of each RDF graph; vertices are categorized as URIs,

[2] `http://www.graphium.ldc.usb.ve`
[3] `http://bl.ocks.org/tjdecke/5558084`
[4] http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

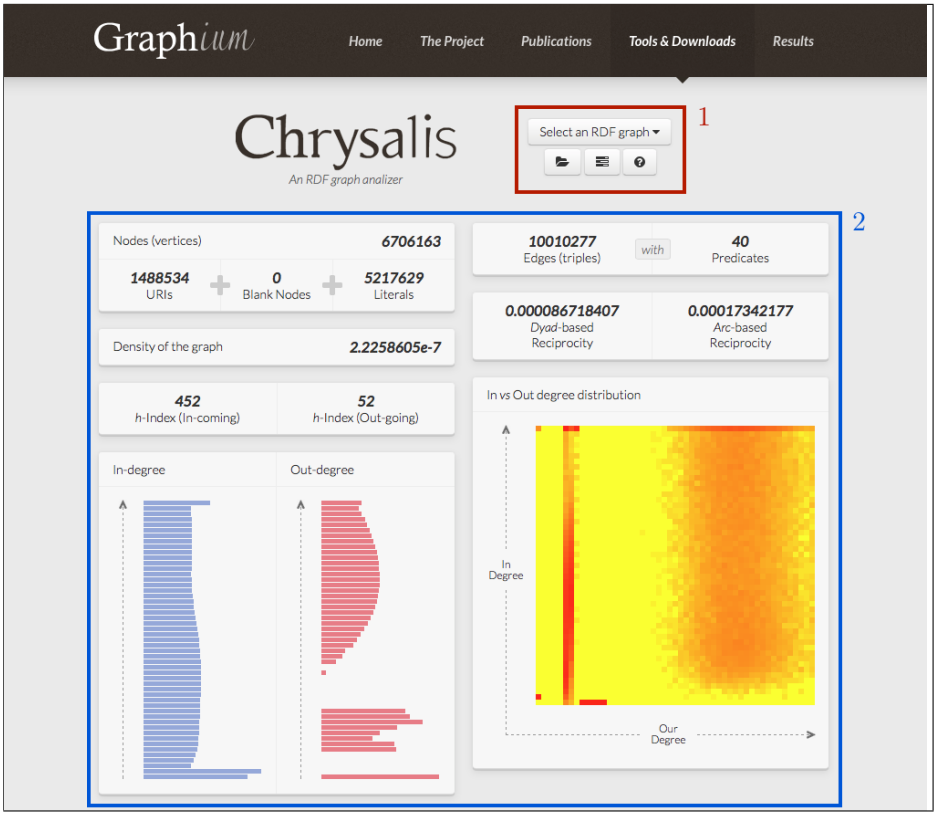

Fig. 1. The Graphium Chrysalis GUI. 1-Selection Area: RDF Graphs can be selected. 2-Visualization Area: Report of Graph Invariant Measurements: Node and Edge Count, In- and Out-Degree Distributions, In-coming and Out-going h-index, Graph Density, and Dyad- and Arc-based Reciprocity. Reported measurements for Berlin10M.

BlankNodes, or Literals. Edges are also discriminated in terms of different predicates.

Graph Density: corresponds to the number of edges in the graph divided by the number of possible edges in a complete digraph. Highly dense RDF graphs are comprised of a small number of resources (i.e., vertices) versus the number of triples (i.e., edges).

Reciprocity (Dyad-based): reciprocity measures the extend to which a triple that relates resources A and B is reciprocated by a another triple that relates B with A too. In general, graphs with high values of reciprocity are highly dense. A value of dyadic reciprocity reflects the proportion of dyads (pairs) with reciprocated triples among all possible adjacent dyads, i.e., a dyadic reciprocity of an RDF graph G corresponds to the number of mutual triples in G divided by the number of mutual triples plus the number of asymmetric triples. A triple t=*(A p B)* in G is mutual if there is another triple t'=*(B p' A)* in G, while t is asymmetric if there is no such triple t' in G. The number of mutual triples corresponds to the number of dyads, i.e., if t and t' exist in G, then they comprise

only one dyad. A value of dyadic reciprocity close to 1.0 indicates that a large number of triples are mutual, and this impacts on the cost of reasoning services such as instance reasoning and query answering.

Reciprocity (Arc-Based): reciprocity can also be measured in terms of the number of reciprocated triples among all the triples in a graph, i.e., arc-based reciprocity corresponds to the number of symmetric triples divided by the number of triples in G, where the number of symmetric triples corresponds to the double of the number of dyads.

In- and Out-degree Distribution: distribution of the number of in-coming and out-going edges of the vertices of a graph. These distributions allow to visualize if vertices with small degrees are more or less frequent. Looking at the logarithmic plots is useful to explain the low selectivity (resp., high selectivity) of the queries than involve the highly connected vertices (resp., low connected vertices).

In-coming and Out-going h-index of a Graph: h is the maximum number, such that h vertices have each at least h in-coming neighbors (resp., out-going neighbors). An RDF graph with a high value of in-coming h-index indicates that at least a large number h of vertices play the role of objects in h-stars, while a small value of out-going h-index represents that at least a small number of h resources play the role of subjects in h-stars. High values of h-index impact on the cost of reasoning and query processing tasks, as well as on the query selectivity.

3.2 Use Cases

We will demonstrate the following use cases:
Patterns and Effects of Vertex and Edge Counts: Attendees will be able to choose between diverse RDF graphs, and analyze different counts of edges and vertices as well as the ones that are resources, literals, or predicates. In the area enclosed by the blue rectangle number 2 in Figure 1, we can observe the relationships between the number of resources, literals, and predicates in a Berlin Benchmark RDF graph of 10 million-triples (Berlin10M). Additionally, attendees will be able to observe that these counts monotonically impact on the cost of core graph-based tasks and query processing; thus these statistics can be used during query optimization to identify good query plans.

Patterns and Effects of In- and Out-Degree Distributions: Attendees will observe distributions of both in-coming and out-going edges in RDF graphs, i.e., a distribution of the frequency of a resource playing the role of subject (out-degree), and the number of times a resource or literal is an object (in-degree). Additionally, comparison of both distributions can be analyzed. Blue rectangle in Figure 1 encloses the distributions of in-, out-degree, and the correlation of both distributions for Berlin10M. We exploit the properties of the HeatMaps to represent these correlations. In Figure 1, an entry (x, y) in the HeatMap represents the number of resources or literals that appear x times as subjects and y times as objects. Entries colored from yellow to red correspond to resources or literals with low to high frequency of appearance. We notice that there is one red point located at $(0, 1)$ which encloses all the literals of the RDF graph. Another interesting pattern that can be observed is that there is a clear difference between the resources that are highly or lowly connected. Based on these statistics, we could con-

clude the queries where triple patterns are instantiated to vertices colored in red are low selective, while those that refer to resources colored in light yellow are very selective.

Patterns and Effects of In-coming and Out-going h-index: We demonstrate the values of h-index for both in-coming and out-going edges, and distributions of different values of the vertices that comprise the studied graphs. Attendees will observe that in Berlin graphs, values of h-index increase as the size of the graphs. Furthermore, resources that represent products, and literals are part of the set of vertices that meet the condition of the h-index, i.e., they have at least h in-coming neighbors (resp., out-going neighbors). Thus, queries composed of triple patterns where a product is instantiated and the predicate is unbound will be low selective. On the other hand, queries with triple patterns bound to resources with low values of h-index will be highly selective. Additionally, traversal tasks in graphs with high values of h-index will be more expensive in time and space than in graphs with low values of this property. Finally, attendees will explore the resources which are in the set of the vertices that meet the h-index condition, and they will also observe the structural patterns that characterize these vertices.

Patterns and Effects of Reciprocity Values: We illustrate values of reciprocity that characterize the studied RDF graphs, and the distribution of both dyad- and arc-based reciprocity. Attendees will observe that values of reciprocity of the Berlin Benchmark graphs are very low, and remain almost the same as the size of the graphs increase.

4 Conclusions

Graphium Chrysalis allows to visualize patterns in the invariants of the RDF graphs, and the impact that the distributions of these values have on core graph-based tasks and reasoning services. Different configurations of RDF graphs will be analyzed, allowing the attendees to understand the graph invariants that characterize graphs generated by existing benchmarks. Attendees will learn the different characteristics of the resources that exhibit high values of the studied measures and what are the effects of using these resources in the different described graph tasks.

References

1. M. C. Lin, F. J. Soulignac, and J. L. Szwarcfiter. Arboricity, h-index, and dynamic algorithms. *Theor. Comput. Sci.*, 426:75–90, 2012.
2. N. Martínez-Bazan, V. Muntés-Mulero, S. Gómez-Villamor, J. Nin, M.-A. Sánchez-Martínez, and J.-L. Larriba-Pey. Dex: high-performance exploration on large graphs for information retrieval. In *CIKM*, pages 573–582, 2007.
3. R. Pichler, A. Polleres, F. Wei, and S. Woltran. drdf: Entailment for domain-restricted rdf. In *ESWC*, pages 200–214, 2008.
4. I. Robinson, J. Webber, and E. Eifrem. *Graph Databases*. O'Reilly Media, 2013.