EPJ Web of Conferences **337**, 01352 (2025) https://doi.org/10.1051/epjconf/202533701352
*CHEP 2024*

# Using and Visualizing Graphs and Graph Algorithms

*Julius* Hřivnáč[1,*]

[1]Université Paris-Saclay, CNRS/IN2P3, IJCLab, 91405 Orsay, France

**Abstract.** Representing HEP and astrophysics data as graphs (i.e., networks of related entities) is becoming increasingly popular. These graphs are not only useful for structuring data storage but are also widely used within various machine learning frameworks.
Despite their rising popularity, many opportunities remain underutilized, particularly regarding the application of graph algorithms and intuitive visualization techniques.
This presentation introduces a comprehensive graph framework designed for handling HEP and astronomical data. The framework supports the storage, manipulation, and analysis of graph data, facilitating the application of fundamental graph algorithms. Additionally, it enables the export of graph data to specialized external toolkits for advanced processing and analysis.
A key feature of this framework is its highly interactive, web-based graphical front-end. This interface provides users with deep insights into the graph structures of their data, enabling interactive analysis and multi-faceted visualization of graph properties. It also offers integration capabilities with other related frameworks.
The framework's practical application is demonstrated through its use in analyzing relationships between astronomical alerts, specifically from the Zwicky Transient Facility (ZTF) and the Rubin Observatory. By leveraging the collective properties and relationships within these data, the framework facilitates comprehensive analyses and provides recommendations based on object similarities and neighborhood characteristics. This approach paves the way for novel insights and methodologies in data-driven research.

## 1 Introduction

Efficient data processing and analysis in scientific computing have led us to the development of hybrid architectures that combine tabular (SQL or NoSQL) databases with graph-based data structures. Tabular storage is useful for efficiently storing large amounts of data, while graph databases provide a more flexible way to represent complex relationships.

Graph algorithms play a crucial role in applications such as classification, clustering, and pattern detection. This paper introduces a framework that integrates both approaches, enabling fast processing and meaningful data representation.

Unlike tabular databases, which store structured information in predefined tables, graph-based storage provides a more intuitive way to model relationships. This advantage is particularly significant in applications such as astronomical alert classification.

*e-mail: Julius.Hrivnac@cern.ch

Graph-based analysis offers enhanced efficiency in large-scale data processing, particularly in cases where relational databases struggle to manage intricate relationships. The ability to dynamically navigate data structures and apply graph-based search algorithms makes this approach highly versatile for handling big data challenges.

The whole system is an integral part of the Fink Broker[1] handling alerts created by Vera C. Rubin Observatory[2].

## 2 Hybrid Database Architecture

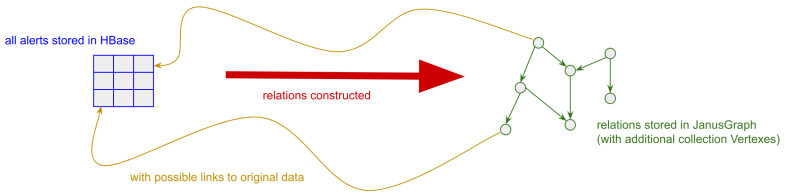


**Figure 1.** Hybrid database: construction of relationships on top of tabular data.

The developed system consists of two major components:

- **Tabular NoSQL storage (HBase[3])** provides stability, fast search, and batch processing capabilities. It is optimized for large-scale data storage and retrieval, making it an excellent choice for handling massive datasets.

- **Graph Storage (JanusGraph[4])** dynamically captures relationships and supports efficient navigational queries. It enables rapid retrieval of connected information and the application of advanced graph algorithms for analysis.

This architecture ensures that all data remains accessible while relationships are efficiently managed within a graph framework. By integrating both storage mechanisms, the system benefits from the structured organization of tabular storage and the flexibility of graph-based relationships.

## 3 Graph Analysis Ecosystem

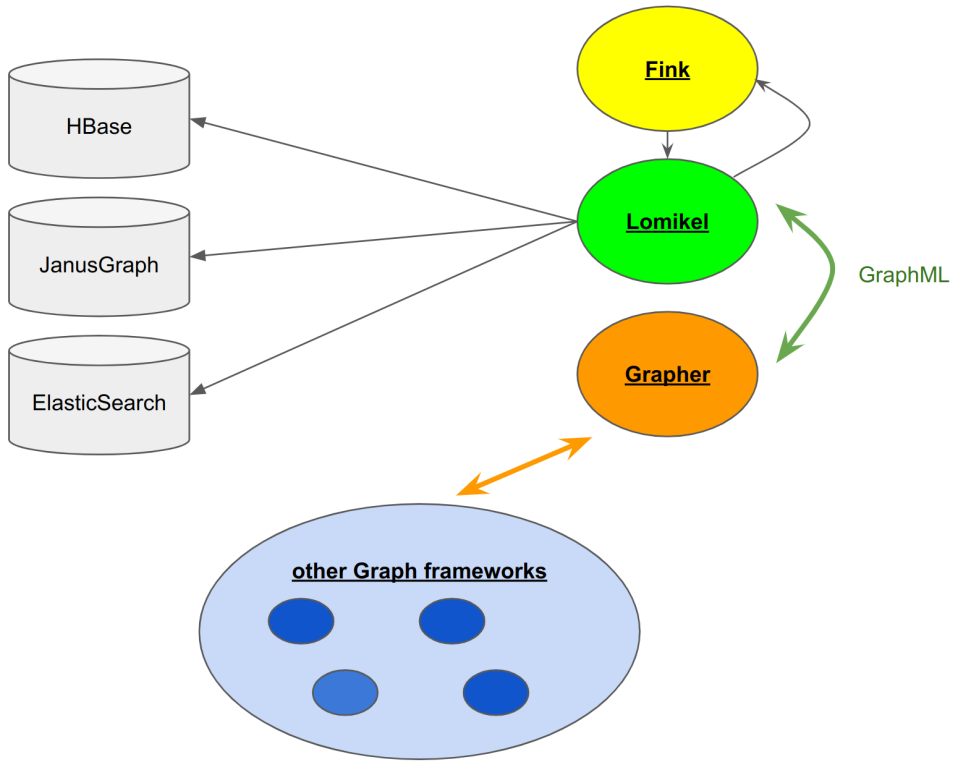


**Figure 2.** Graph Architecture: Involved packages (Fink, Lomikel, Grapher) and databases (HBase, Janus Graph, Elastic Search).

Several tools are integrated into the ecosystem, as shown in Figure 2.

### 3.1 Lomikel Package

Lomikel[5] is a toolkit for interacting with databases, supporting SQL, NoSQL, and graph storage. It offers scripting capabilities in Java, Python, and Groovy, allowing researchers and developers to customize workflows for data analysis.

Lomikel enables real-time querying of graph data, facilitating the traversal of large datasets. It also provides visualization features to help interpret results by dynamically rendering graph relationships.

```
// an example of a Lomikel script to analyse graph data
// connect to JanusGraph database
jc = new JanusClient("IJCLab.properties");
// access specific Fink utilities
gr = new FinkGremlinRecipiesG(jc);
// execute Gremlin (graph database API) request
jc.g().V().limit(1).valueMap().next();
// find closest sources
```

```
gr.sourceNeighborhood("ZTF17aaawgky", null, null, 10);
// get source classification
gr.classification("ZTF17aaawgky");
// get all overlaps
gr.overlaps();
// do some statistics
gr.standardDeviationE('deepcontains', ['weight']);
// export overlaps to GraphML file
gr.exportAoISoI("Overlaps.graphml");
```

### 3.2 Grapher Package

Grapher[6] focuses on pure graph algorithms and provides functionalities such as clustering, connectivity analysis, and distance metrics. These capabilities are essential for detecting relationships within large datasets.

External toolkits such as JGraphT[7], NetworkX[8], Snap[9], and SageMath[10] allow for enhanced interoperability. Grapher ability to convert between different graph file formats enables interoperability with various external tools, making it a versatile component of the ecosystem. The ability to incorporate these external libraries extends the range of analytical techniques available to researchers. Graph algorithms are applied to detect patterns and analyze relationships. The integration of multiple algorithms ensures adaptability across different research domains.

```
// an example of a Grapher script to apply graph algorithms to graph data
// convert GraphML file into GraphViz Dot file
cli.setInfile("AoI.graphml");
cli.setOutfile("AoI.dot");
convertor = new Convertor(cli);
convertor.read();
convertor.convert();
// fill data into Analyser
analyser = new Analyser(cli);
analyser.fill(convertor.read());
// apply various Graph algorithms
analyser.applyStrongConnectivity();
analyser.applyConnectivity(10);
analyser.applyClustering("GirvanNewman", 30);
analyser.applyClustering("LabelPropagation", 30);
analyser.applyClustering("KSpanningTree", 30);
```

## 4 Application: Fink Classification Graphs

The framework has been applied to classify astronomical alerts in the LSST[2] (Vera C. Rubin Observatory). The Fink[1] broker processes up to 10 million alerts per night, constructing relations between alerts and sources.

### 4.1 Fink Browser and Searching for Relations

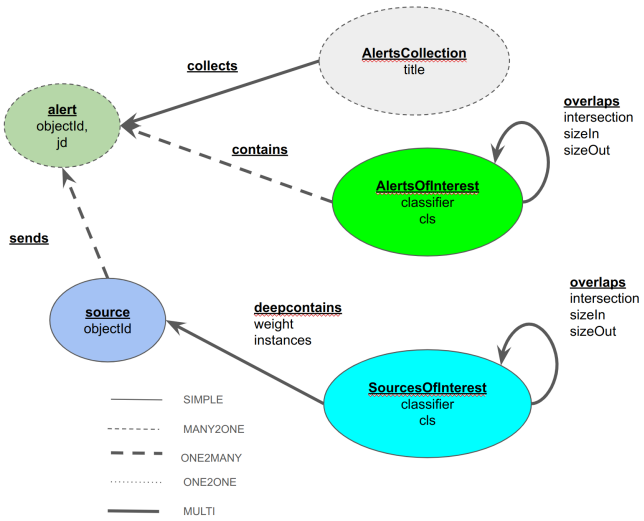


**Figure 3.** Graph database schema: Vertices and Edges.

The framework enables scientists to identify previously unknown connections in astronomical data, improving the classification of transient events and anomalies. The framework provides an interactive environment for exploring astronomical alert classifications. It allows users to identify clusters of similar objects using PCA-based classification and search for overlapping relationships among different sources.

### 4.2 Overlaps in Graphs

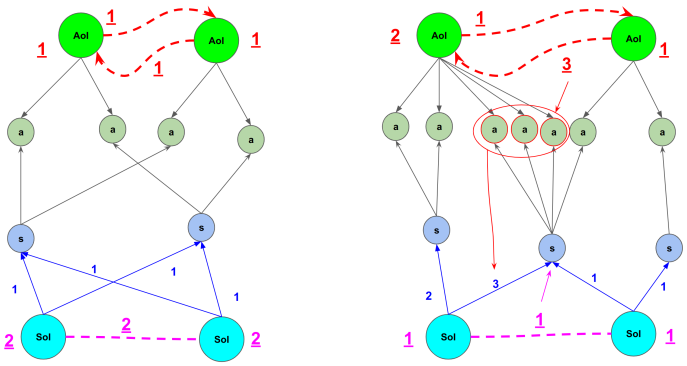


**Figure 4.** Overlaps between varius collection Vertices (AoI - *Alerts of Interest* or SoI - *Sources of Interest*) captures number of alerts (a) or alerts sources (s) belonging to them simultaneously

Overlapping relationships in graphs are essential for understanding how different alert sources are connected. Overlaps can be quantified by measuring the number of alerts that belong to

multiple classification categories as illustrated in Figure 4. This is achieved by calculating edge weights between nodes to determine overlap strength, using statistical measures to analyze the frequency of shared alerts and employing clustering techniques to group alerts with high overlap rates.

Figure 5 illustrates how overlapping alerts can be visualized within the graph structure.

## 5 Graph Visualization

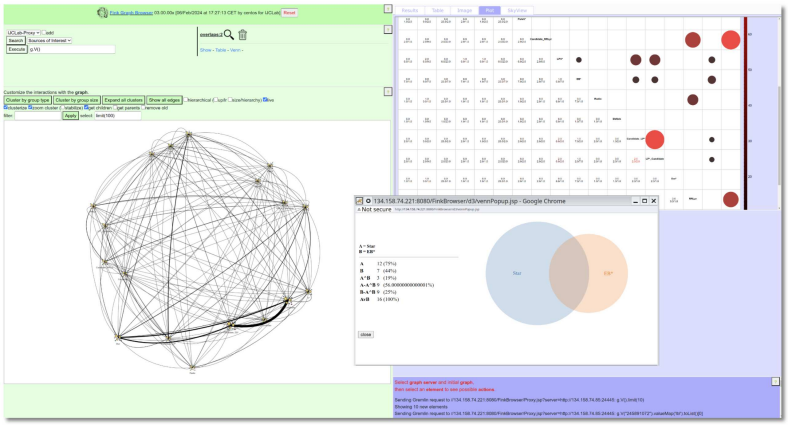

**Figure 5.** Lomikel/Fink Browser showing network of overlaps with details represented in a tabular and Venn diagrams.

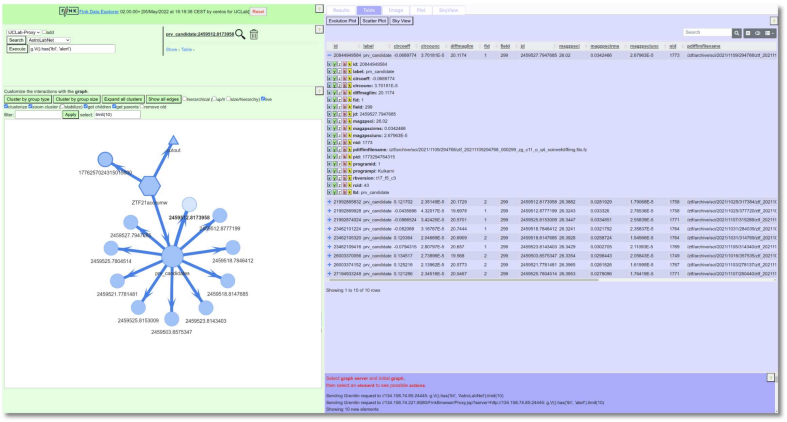

**Figure 6.** Lomikel/Fink Browser allows highly interractive access to graph data and their interpretaion in various ways (tabular,...).

Graph visualization remains a challenge due to the lack of generic tools. We have developped the Lomikel Browser, a web-based service enabling interactive graph exploration, allowing users to navigate complex datasets visually. Customizable visual stylesheets, which help

tailor the visualization to specific research needs. External plugins for deeper analysis can be included, providing extensibility for future research applications.

Visualization helps in identifying key patterns in data. Figure 6 demonstrates a graphical representation of classified alerts. Effective visualization methods enable researchers to interpret results efficiently and identify meaningful insights from the data.

## 6 Discussion and Future Work

Future work will focus on enhancing the efficiency of real-time querying, refining graph visualization techniques, and integrating machine learning methodologies to improve automated classification.

## References

[1] A. Muller, J. Peloton, E.E.O. Ishida, C. Arnault, E. Bachelet, T. Blaineau, D. Boutigny, A. Chauhan, E. Gangler, F. Hernandez et al., Fink, a new generation of broker for the LSST community, Monthly Notices of the Royal Astronomical Society **501**, 3272 (2020). 10.1093/mnras/staa3602

[2] LSST Science Collaboration, P.A. Abell, J. Allison, Anderson, LSST Science Book, Version 2.0, ArXiv e-prints arXiv:0912.0201 (2009), 0912.0201.

[3] Apache hbase, https://hbase.apache.org

[4] Janusgraph, https://janusgraph.org

[5] Lomikel, https://github.com/hrivnac/Lomikel

[6] Grapher, https://github.com/hrivnac/Grapher

[7] Jgrapht, https://jgrapht.org

[8] Networkx, https://networkx.org

[9] Snap, https://snap.stanford.edu/snap

[10] Sagemath, https://www.sagemath.org