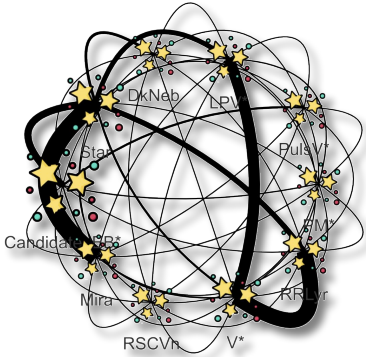
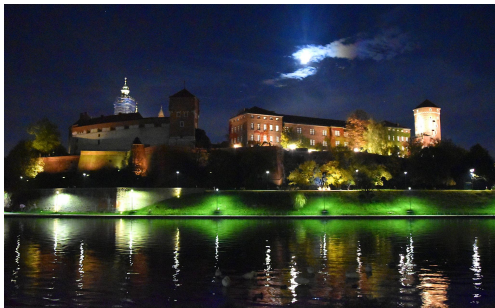
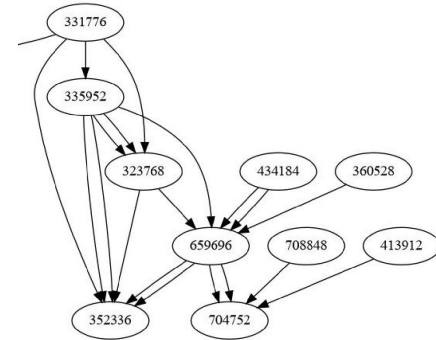
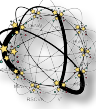


Using and Visualizing Graphs and Graph Algorithms

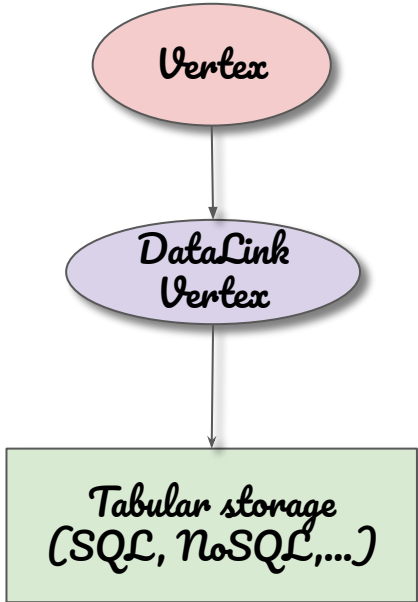


- Hybrid Database Architecture
- Graph Analyses Ecosystem
 - Lomikel
 - Grapher
- **Fink Classification Graphs**
 - Overlaps
 - Neighborhood
 - PCA-based Classification
- Graph Visualisation



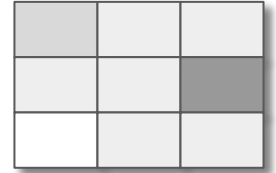


Hybrid Database Architecture



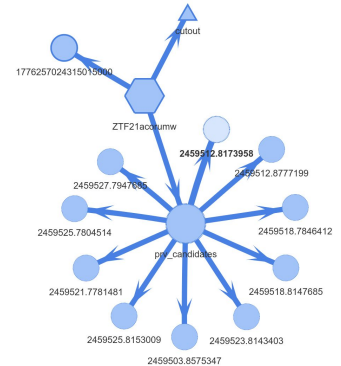
➤ **Data in tabular storage (HBase)**

- Stable
- Fast search
- Fast processing
- Fast injection
- Suitable for batch processing
- Contains all data
 - Static (rarely modified)



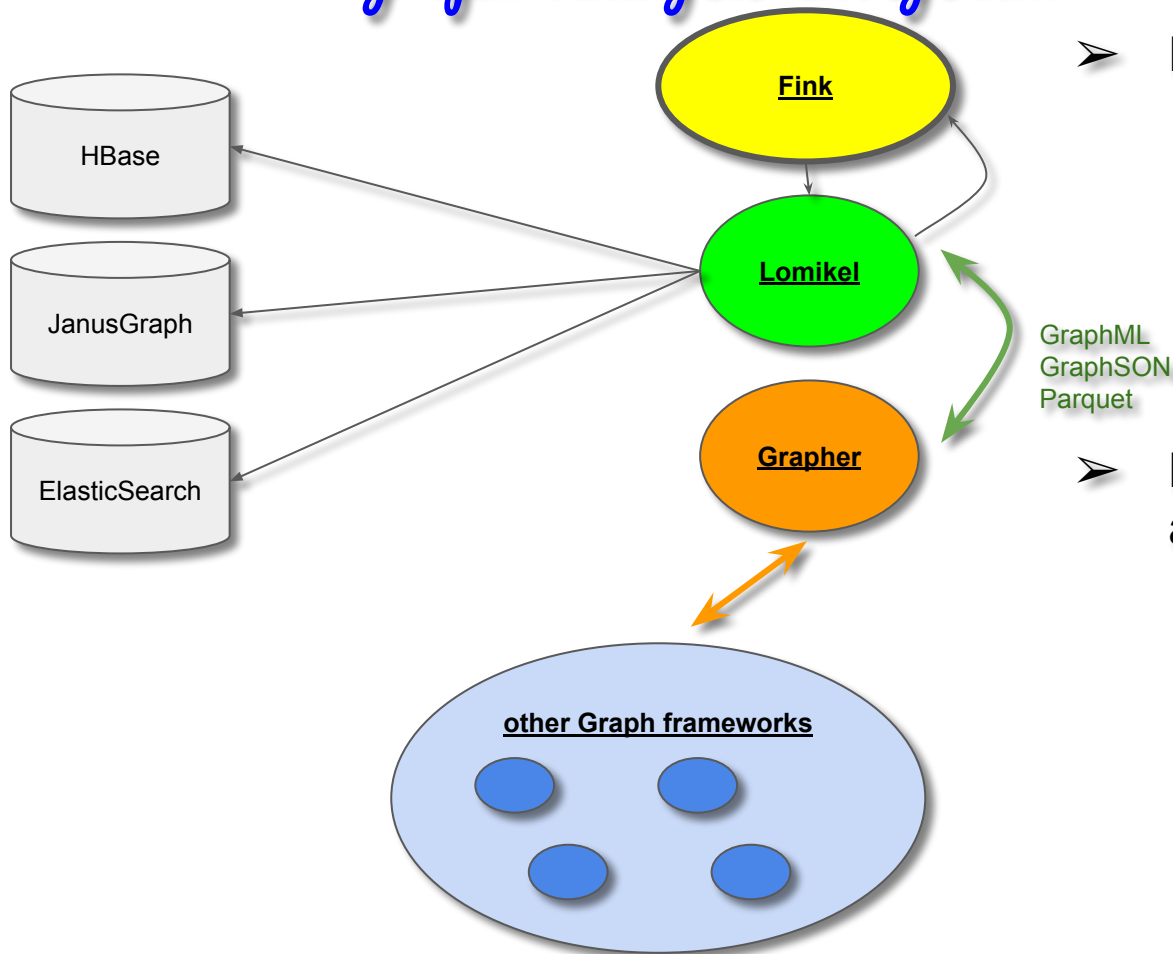
➤ **Relations in graph storage (JanusGraph)**

- Expressive
- Fast navigation
- Suitable for algorithmic search
- Contains structures/relations
 - Dynamic

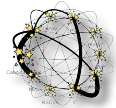


presented @CHEP2023

Graph Analysis Ecosystem



- **Lomikel, Grapher** exist as
 - standalone programs (jar or exe), scriptable in Java, Python, Groovy
 - libraries to be included in other Java, Python, Groovy, C,... programs
 - interactive Web Service
- Both can handle Graph algorithms
 - **Lomikel** is more database-oriented (i.e. useful for algorithms heavy on searching & navigation)
 - **Grapher** is better for pure Graph algorithms heavy on processing



Lomikel

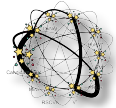
- Toolkit interacting with databases (SQL, NoSQL, Graph)
- Unified API for different technologies (as far as possible)
- Many useful utilities, especially for HBase and Gremlin-based Graph databases

```
$ lomikel -h
usage: java -jar Lomikel.exe.jar [-a ] [-b] [-g] [-h] [-n] [-q] [-s <file>] [-w]
-a,--api <language> cli language: [bsh|groovy|python]
                               (otherwise taken from source extension, default is groovy)
-b,--batch                      run in a batch
-g,--gui                        run in a graphical window
-h,--help                      show help
-n,--notebook                  run in an notebook
-q,--quiet                    minimal direct feedback
-s,--source <file>           source script file (init. is also sourced)
-w,--web                      run as a web service
```

```
$ lomikel -b -s script.bsh
$ lomikel -b -s script.py -api python
$ lomikel -b -s script.groovy -api groovy
$ lomikel
$ lomikel -gui
```

```
import com.Lomikel.JanusUser.JanusClient;
import com.astrolabsoftware.FinkBrowser.JanusUser.FinkGremlinRecipiesG;

// 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');
```



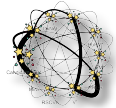
Grapher

- Toolkit for Graph manipulation
- Conversion between various file formats (to communicate with other toolkits)
 - graphml, dot, mat, g6, csf, json
- Algorithms for graph analyses and manipulation
 - Clustering
 - Connectivity, isolation, immersion
 - Adding distance between vertices (calculated in different ways)
 - Adding vertex inclusiveness

```
$ alias grapher='java -jar Grapher.jar'
$ grapher -h
usage: java -jar Grapher.exe.jar
  -a,--alg          apply algorithm (instead of just converting)
                   [sc = Strong Connectivity | cl = Clustering | ad = adding distances]]
                   several algorithms can be separated by ;
                   algorithm arguments can be supplied after ,
  -e,--noedge      ignore input edges
  -h,--help        show help
  -i,--in          input file name [.graphml]
  -o,--out         output file name [.dot|.mat|.g6|csv|json|graphml]
  -q,--quiet       minimal direct feedback
  -s,--script      script to run (ignores all other options) [.groovy|.py]
  -v,--novertex   ignore output edge-less vertices
  -w,--show        show in graphical window (instead of converting)
```

```
import com.Grapher.Convertors.Convertor;
Import com.Grapher.Analysis.Analyser;

// 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);
```

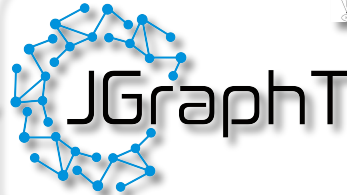


External Graph Toolkits

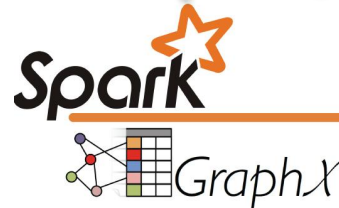
included in Lomikel



included in Grapher

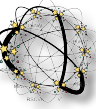


GraphML...



- Using **JGraphT** toolkit for graph manipulations
 - User-friendly, supports many exchange data formats and protocols
 - Contains many standard graph analysis algorithms
 - It's simple to add new algorithms
 - Pure Java, with Python interface (made by GraalVM)
- But many other toolkits (all inter-operable with **JGraphT**):
 - **NetworkX**: well integrated in Python environment
 - **Snap**: very advanced framework from Stanford (C++ & Python)
 - **SageMath**: very rich Math framework (Python-ish)

Fink Classification Graphs



- LSST/Fink
- **Overlaps**
- **Neighbourhood**
- **PCA-based Classification**

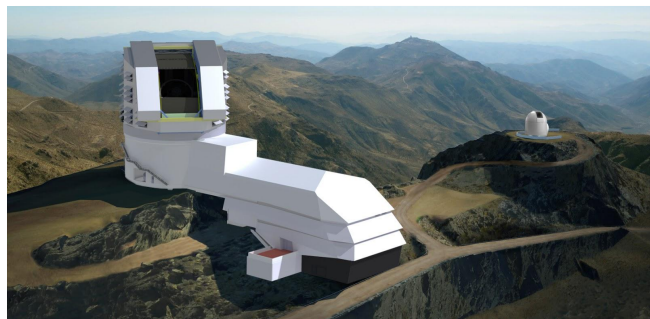
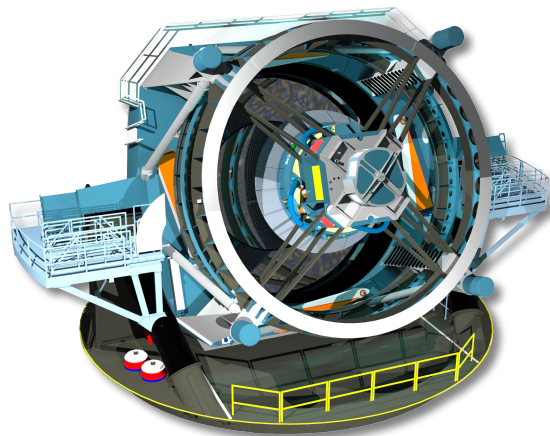
concrete examples of Graph algorithms usage:

Classification of astronomical alerts



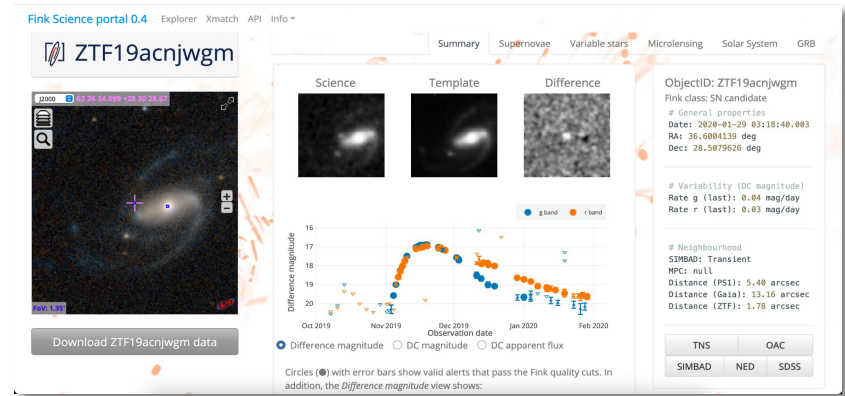
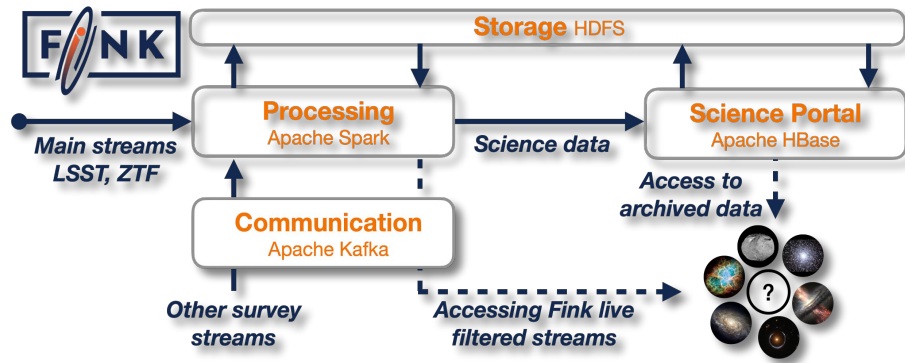
LSST

- **Vera C. Rubin Observatory Project for Legacy Survey of Space and Time (LSST)**
- Searching for new or changing objects
- 8.4 m, 3.2 Gpixel camera in Chile
- Up to **10 million alerts**, 20 TB of data **each night**
- 500 PB of data will be accumulated in 10 years of operation
- Alerts sent worldwide via the network of 'brokers'
- First real observations next year
 - Framework tested on ZTF (Zwicky Transient Facility) data

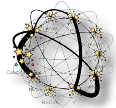




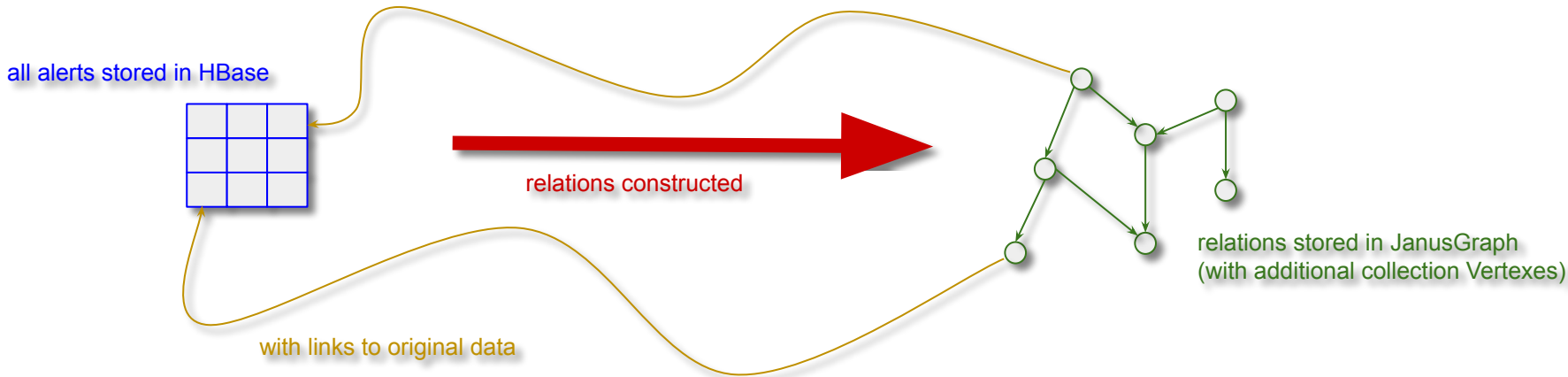
- Incoming **Alerts** Management
- Using Big Data / NoSQL databases
- One of the official alert brokers of the LSST

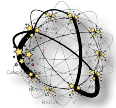


Constructing the Graph



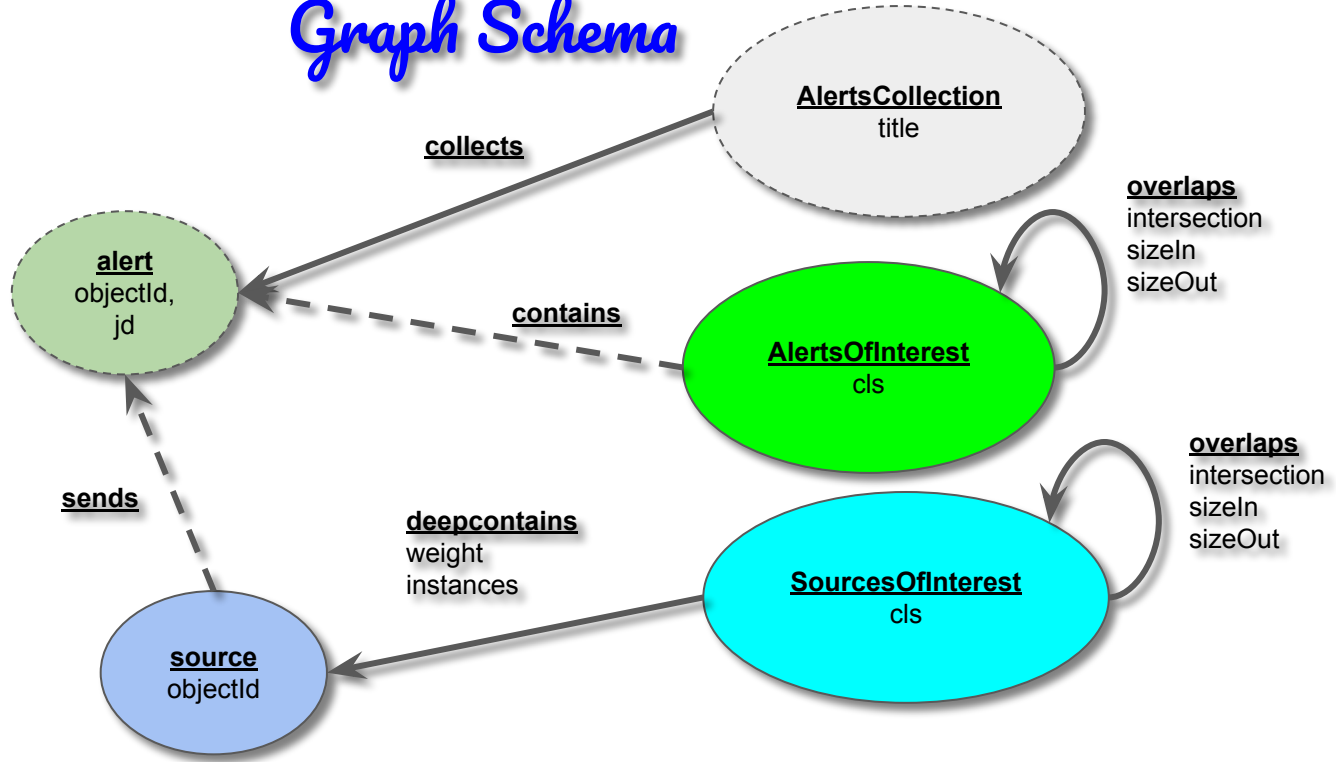
- Import new sources (or update)
 - select interesting (latest, classified as anomaly,...) sources (from HBase)
 - import them into graph
 - construct collections for alert classes
- Calculate relations





Graph Schema

- SIMPLE
- - - MANY2ONE
- - - ONE2MANY
- ONE2ONE
- MULTI



alert = something new happens or something changes

source = the origin of the alert

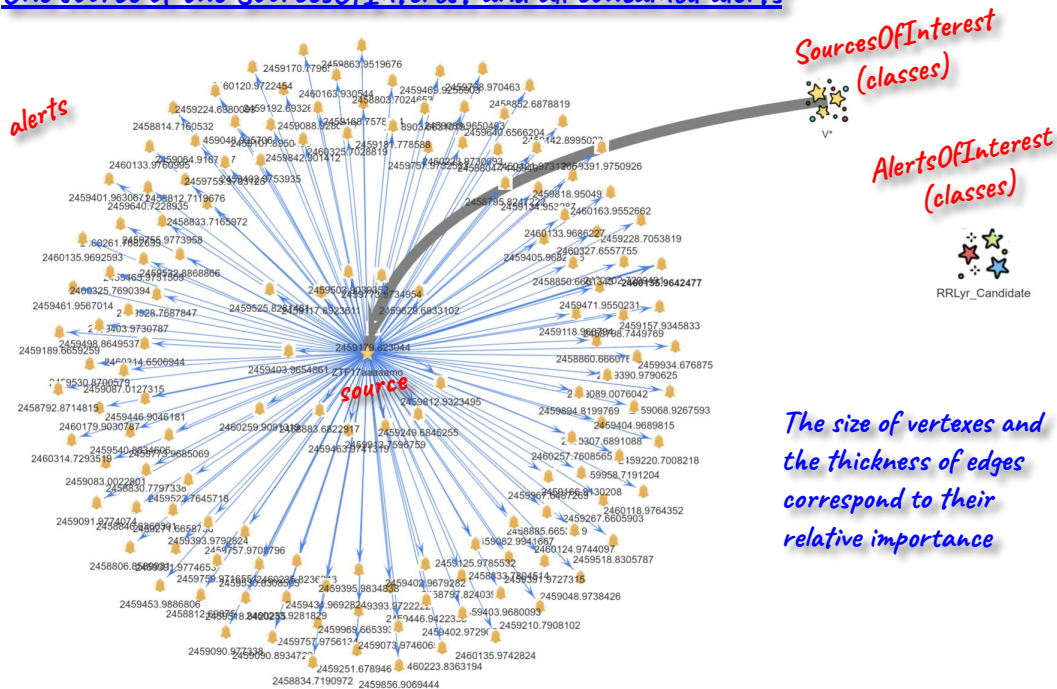
Alerts/SourcesOfInterest = alerts/sources are classified wrt possible types (SuperNova, Asteroid, Alien Spaceship,...)

- with uncertainties (one alert/source can belong to several classes)

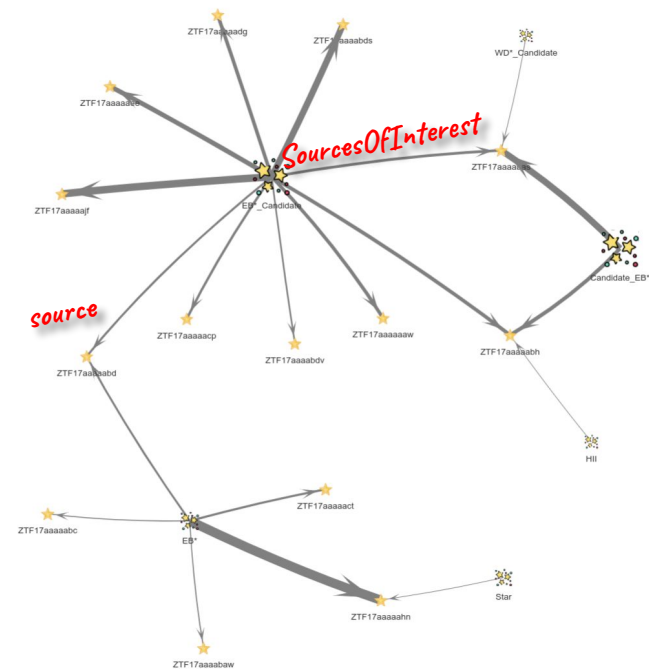
The Graph

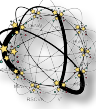


One source of one SourcesOfInterest and all contained alerts



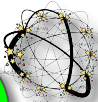
Several SourcesOfInterest and some of their sources



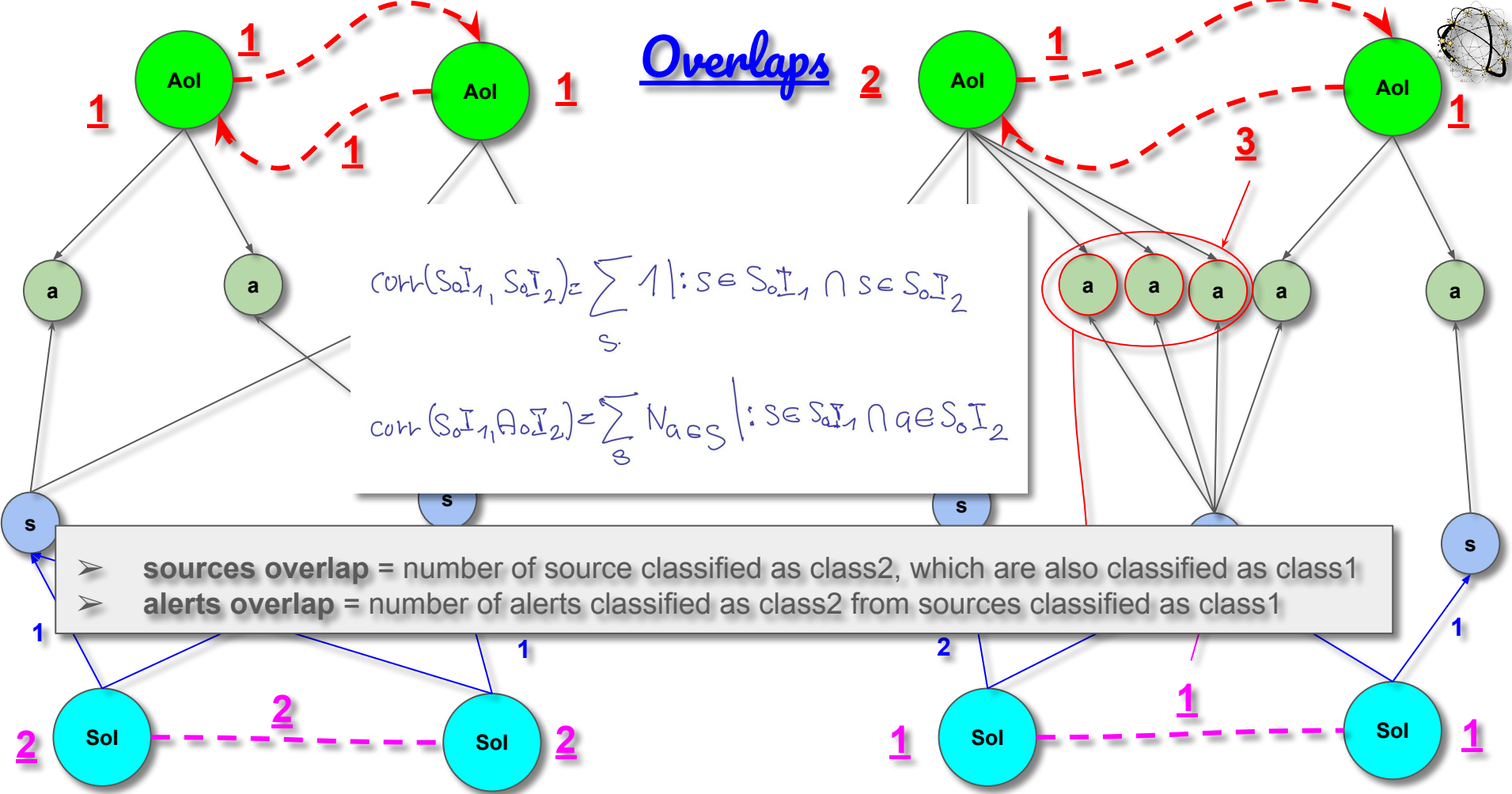


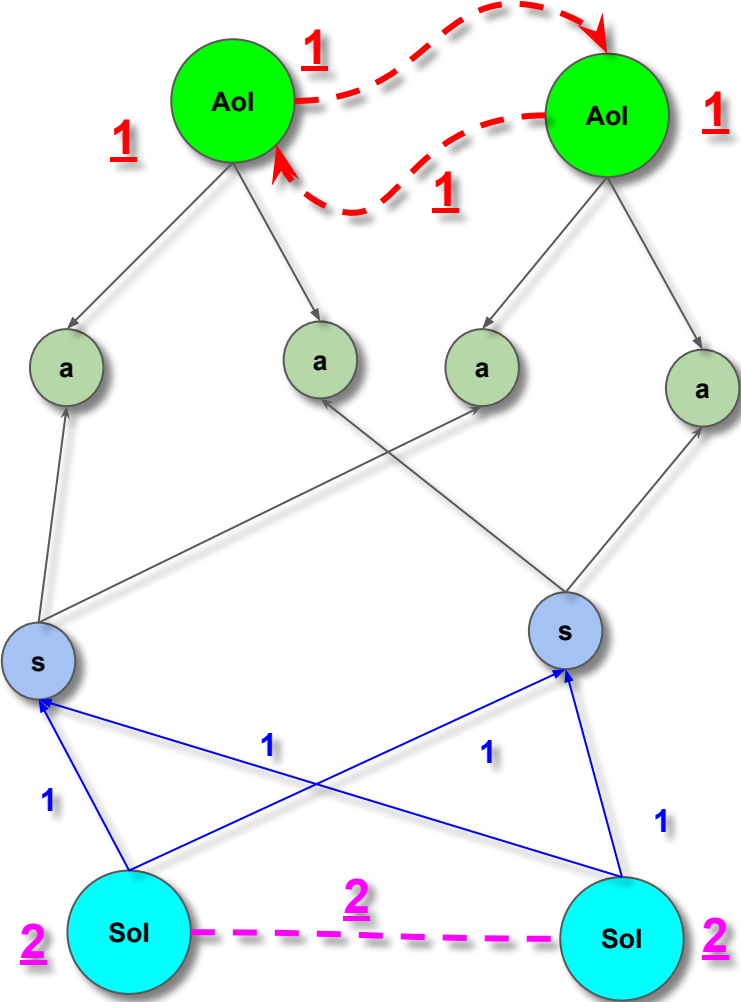
Interesting Relations to search for

- **Overlaps:** How many alerts are classified as several types (classes)
- **Neighborhood:** Which alerts are similar (wrt possible classifications) to an alert
- **PCA-based classification:** Classify new alerts with clusters of existing alerts

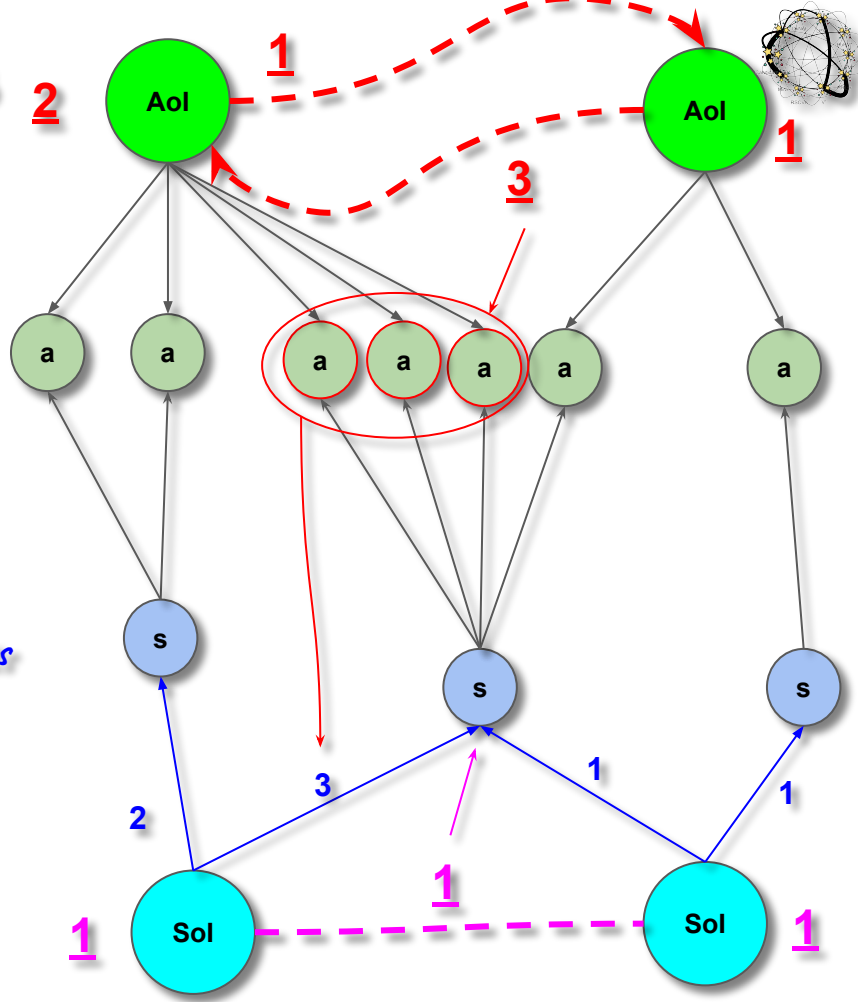


Overlaps





Overlaps

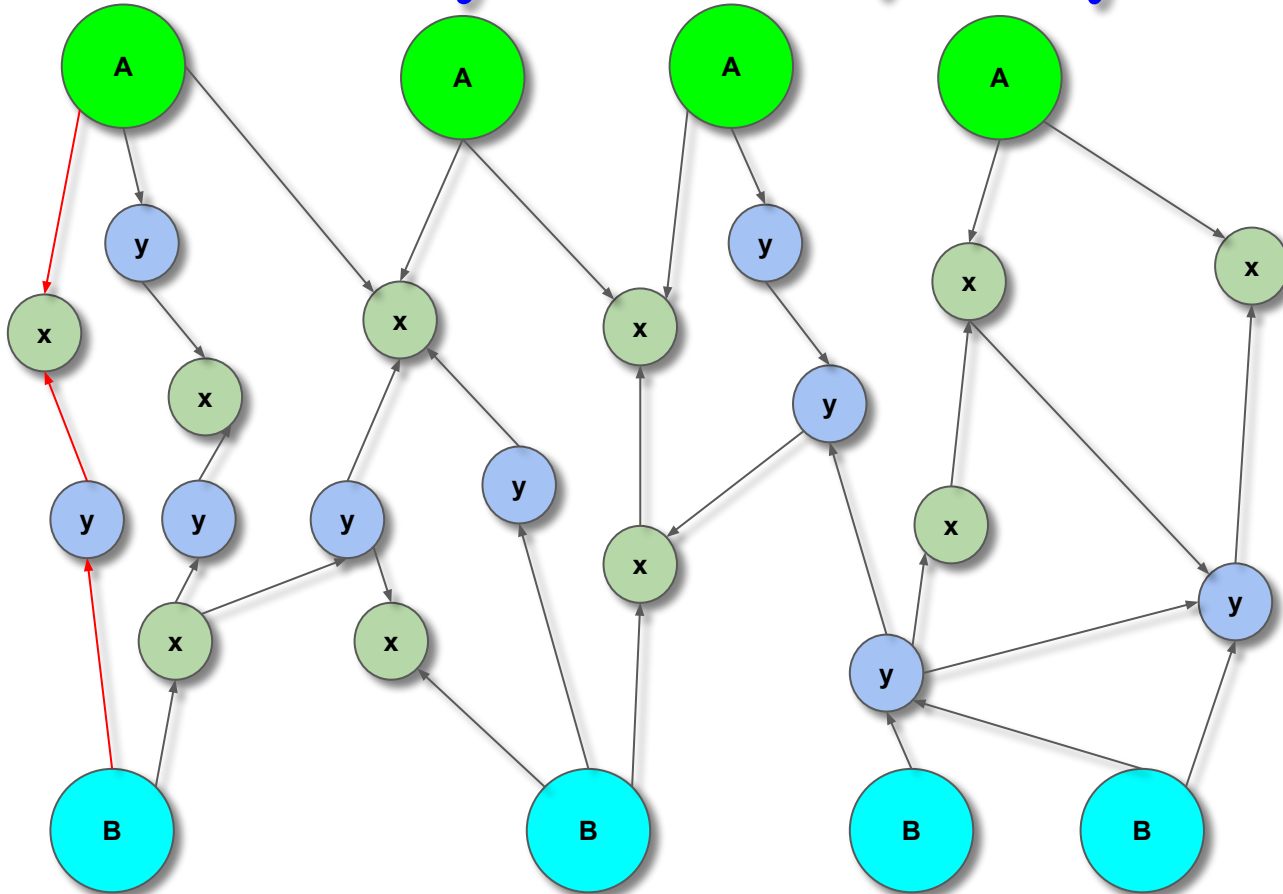
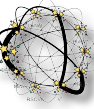


weight = number of a connecting Sol and Aol via s

number of s common to two Sol

number of a from the other Sol

Generalisation of Overlaps



- find overlaps between **A** and **B** with respect to **x**
- how to take into account Edge weights ?
- how to include multiple-paths via the same **x** ?
- should work for **C == B**
- should work for oriented and unoriented Edges

Not well handled by common graph analyses toolkits - as it contains vertices of different type !

Visual Overlaps



Fink Graph Browser 03.00.00x [06/Feb/2024 at 17:27:13 CET by centos for IJCLab]Reset

IJCLab-Proxy ▾

Sources of Interest ▾

Execute

g.V()

overlaps:2

Sources of Interest

Show - Table - Venn -

Customize the interactions with the graph.

Cluster by group type | Cluster by group size | Expand all clusters | Show all edges | hierarchical (up/lr | size/hierarchy) live

clusterize zoom cluster (stabilize) get children get parents remove old

filter: Apply select: limit(100)

SourcesOfInterest with overlaps

Results	Table	Image	Plot	SkyView
0.0 100.0 5.0 900.0 0.0 20.00.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 20.0 2.0 600.0 0.0 20.00.0 1.0 60.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0				
0.0 50.0 3.0 900.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 1.0 500.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				
0.0 100.0 0.0 1000.0 0.0 20.01.0 0.0 100.0 0.0 2000.0 0.0 2000.0 0.0 2000.0				

table of overlaps

134.158.74.221:8080/FinkBrowser/d3/vennPopUp.jsp - Google Chrome3.0 30% dMLr

Not secure <http://134.158.74.221:8080/FinkBrowser/d3/vennPopUp.jsp>

A = Star
B = EB*

A 12 (75%)
B 7 (44%)
A^B 3 (19%)
A-A^B 9 (56.000000000000001%)
B-A^B 9 (25%)
A^B 16 (100%)

details of one concrete overlap (Venn Diagram)

Select graph server and initial graph, then select an element to see possible actions.

Showing 10 new elements

Sending Gremlin request to //134.158.74.221:8080/FinkBrowser/Proxy.jsp?server=http://134.158.74.85:24445: g.V().limit(10)

Showing 10 new elements

Sending Gremlin request to //134.158.74.221:8080/FinkBrowser/Proxy.jsp?server=http://134.158.74.85:24445: g.V("245891072").valueMap("lb").toList()[0]

Access to Overlaps



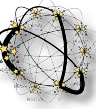
script

```
jc = JanusClient("IJCLab.properties");  
gr = FinkGremlinRecipiesG(jc);  
print(gr.overlaps('AlertsOfInterest'));
```

result

```
AlertsOfInterest:RRLyr * AlertsOfInterest:RRLyr=3666076.0  
AlertsOfInterest:V* * AlertsOfInterest:V*=1872160.0  
AlertsOfInterest:RRLyr * AlertsOfInterest:Star=1801680.0  
AlertsOfInterest:EB* * AlertsOfInterest:EB*=1160366.0  
AlertsOfInterest:QSO * AlertsOfInterest:QSO=1082817.0  
AlertsOfInterest:LPV* * AlertsOfInterest:LPV*=1079266.0  
AlertsOfInterest:Candidate_EB* * AlertsOfInterest:Candidate_EB*=1020285.0  
...  
AlertsOfInterest:Star * AlertsOfInterest:LPV*=189870.0  
AlertsOfInterest:Mira * AlertsOfInterest:LPV*=184988.0  
AlertsOfInterest:PulsV* * AlertsOfInterest:PulsV*=184247.0  
AlertsOfInterest:RRLyr * AlertsOfInterest:LPV*=179521.0  
AlertsOfInterest:LP*_Candidate * AlertsOfInterest:V*=175345.0  
AlertsOfInterest:RRLyr * AlertsOfInterest:V*=173963.0  
AlertsOfInterest:RRLyr * AlertsOfInterest:GinCl=172219.0  
AlertsOfInterest:V* * AlertsOfInterest:RRLyr=162798.0  
AlertsOfInterest:LPV* * AlertsOfInterest:Mira=158251.0  
...
```

Exporting Overlaps (to GraphViz)

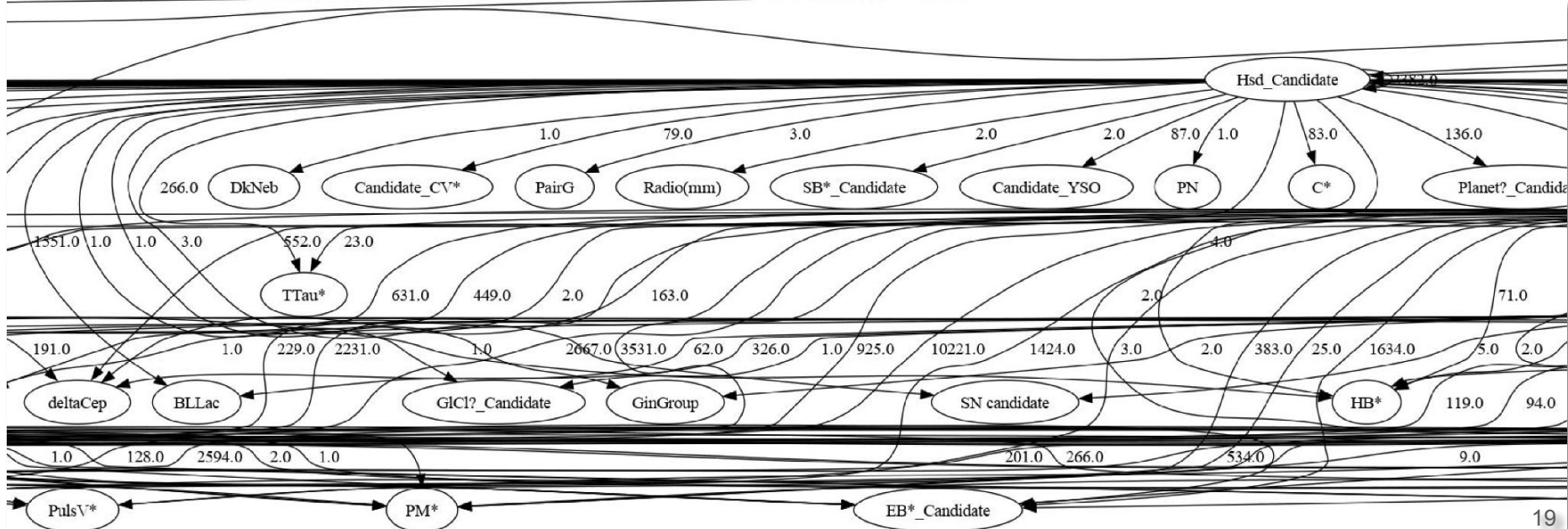


script

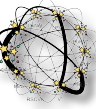
```
jc = JanusClient("IJCLab.properties");  
gr = FinkGremlinRecipiesG(jc);  
gr.exportAoISoI("/tmp/overlaps.graphml");
```

conversion

```
> grapher -i overlaps.graphml -o overlaps.dot
```



Analysing Exported Overlaps



result

Script searching for the most connected classes

- i.e. classes with most overlaps
- i.e. commonly mis-classified classes

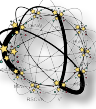
```
> grapher -s analyse.groovy
import com.Grapher.Convertors.Convertor;
import com.Grapher.Analysis.Analyser;

cli.setInfile("overlaps.graphml");
convertor = new Convertor(cli);

analyser = new Analyser(cli);
analyser.fill(convertor.read());
analyser.applyConnectivity(10);
```

```
Grapher initialised, version: 01.01.00x [23/Apr/2024 at 11:00:43 CEST by hrivnac]
Executing Groovy analyse.groovy ...
Reading overlaps.graphml
Generating weights ...
Imported graph: DefaultGraphType [directed=true, undirected=false, self-loops=true,
multiple-edges=false, weighted=true, allows-cycles=true, modifiable=true][211,
7503] from SoI.graphml
Applying Connectivity Algorithm ...
Most Connected:
SourcesOfInterest(532525064):Solar System MPC=2.001475751813528E-5
SourcesOfInterest(122896456):Candidate_EB*=1.9299584561486104E-4
SourcesOfInterest(246046768):Candidate_RRLyr=2.1192136461512155E-4
SourcesOfInterest(245780528):EB*_Candidate=2.1717508026100228E-4
SourcesOfInterest(163864680):RRLyr_Candidate=2.2113304520797968E-4
SourcesOfInterest(532557832):Candidate_LP*=2.5487725838755677E-4
SourcesOfInterest(245788784):V*=2.6911282233468034E-4
SourcesOfInterest(245780592):Mira=2.96411720247869E-4
SourcesOfInterest(245776432):LP*_Candidate=3.045124013188856E-4
SourcesOfInterest(532500488):LPV*=3.1365731058502976E-4
Least Connected:
SourcesOfInterest(670097464):Candidate_post-AGB*=0.16667572451461254
SourcesOfInterest(332038232):WR*_Candidate=0.187895248349279
SourcesOfInterest(291020976):Candidate_WR*=0.187895248349279
SourcesOfInterest(862470184):Possible_GrG=0.19446988682381575
SourcesOfInterest(179531848):Candidate_SG*=0.1952614379084967
SourcesOfInterest(453677288):SFregion=0.25003816356905695
SourcesOfInterest(499388640):Candidate_LMXB=0.3333394893009283
SourcesOfInterest(167751752):LMXB_Candidate=0.3333394893009283
SourcesOfInterest(616284304):Candidate_LensSystem=0.34090909090909094
SourcesOfInterest(177287240):Candidate_RSG*=0.35
```

Analysing Exported Overlaps



Script searching for groups of closely connected classes

- i.e. groups of most often overlapping classes*

result

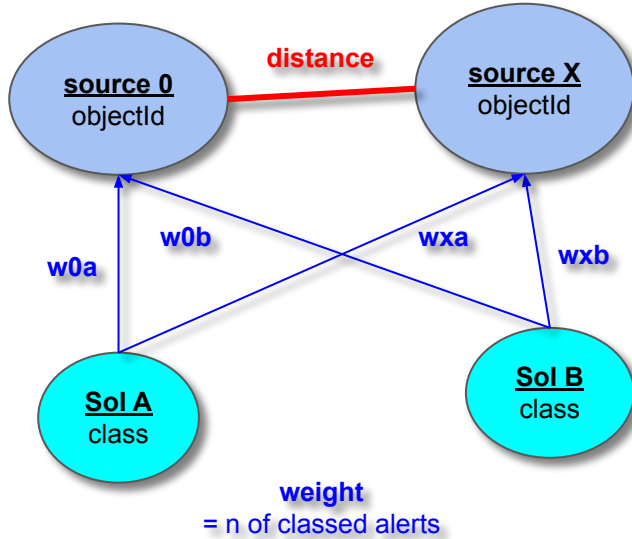
```
> grapher -s analyse.groovy
import com.Grapher.Convertors.Convertor;
import com.Grapher.Analysis.Analyser;

cli.setInfile("overlaps.graphml");
convertor = new Convertor(cli);

analyser = new Analyser(cli);
analyser.fill(convertor.read());
analyser.applyClustering("GirvanNewman", 10);
```

```
Grapher initialised, version: 01.01.00x [23/Apr/2024 at 11:00:43 CEST by
hrivnac]
Executing Groovy analyse.groovy ...
Reading overlaps.graphml
Generating weights ...
Imported graph: DefaultGraphType [directed=true, undirected=false,
self-loops=true, multiple-edges=false, weighted=true, allows-cycles=true,
modifiable=true][211, 7503] from SoI.graphml
Applying Clustering Algorithm ...
    usingt GirvanNewman algorithm
    searching for 10 clusters
Clusters:
...
    [SourcesOfInterest(81973328):Early SN Ia candidate,
SourcesOfInterest(122953800):Solar System candidate,
SourcesOfInterest(163905752):Ambiguous,
SourcesOfInterest(164077672):Kilonova candidate]
    [SourcesOfInterest(499388640):Candidate_LMXB,
SourcesOfInterest(167751752):LMXB_Candidate]
...
```

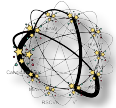
Neighborhood



$$\text{dist}(\text{source } \phi, \text{source } x)$$
$$= \frac{1}{\sqrt{2}} \sqrt{\sum_{i,j \in \text{SoI}} \left(\frac{|w_{\phi i} - w_{\phi j}|}{w_{\phi i} + w_{\phi j}} - \frac{|w_{xi} - w_{xj}|}{w_{xi} + w_{xj}} \right)^2}$$

distance between sources = similarity of sources with respect to classification of contained alerts

Access to Source Neighborhood



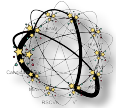
script searching for source closest to a source wrt a classification
“if you like this one, you will probably also like those”

```
jc = JanusClient("IJCLab.properties");  
gr = FinkGremlinRecipiesG(jc);  
print(gr.sourceNeighborhood("ZTF17aaawgky", None, ["Star", "Mira", "V*"], 5));
```

result

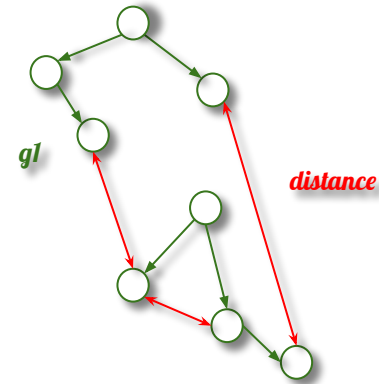
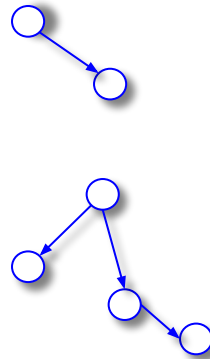
```
21683 INFO (Januser.FinkGremlinRecipies : 877) : calculating source distances from ZTF17aaawgky wrt [V*, Star, Mira] ...
```

```
{ZTF18ablxmzd=0.005986089152901986, ZTF18adjlqkl=0.007482611441127496, ZTF18aayebnh=0.007482611441127496, ZTF21aahkto=0.008679829271707878,  
ZTF18abdlapp=0.011986363351189971}
```



PCA-based Alert Classification

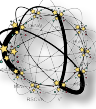
1. Find alert's **Primary Components** (Vertices)
2. Define **metric** capturing alerts similarity (Edges)
3. **Clusterize** (cluster = source class) using Graph algorithms
4. Use known alerts (classified with certainty) to define clusters classes
5. **Classify** new alerts wrt existing clusters
6. Execute iteratively



```
// Find all pairs of PCA Vertexes, which are close enough.  
// Connect them with the Edge 'distance' having a 'difference' property equal to  
// the calculated difference.  
variables = 'pca00 pca01 ... pca24'  
difference = 'qdistane(variables,...) // quadratic distance  
gr.structurise(g.V().has('lbl', 'PCA'), difference, variables, 'distance', 'difference', ...)  
// Get some statistics about newly created Edges.  
g.E().hasLabel('distance').values('difference').union(min(), max(), sum(), mean(), count())  
// Find clusters of 'close' PCAs.  
FinkBrowser.findPCAclusters(g, 'distance', 'difference', 2, 10.5)
```

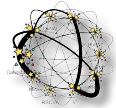
*while previous algorithms analysed existing alerts classifications,
this one helps to classify new alerts*

Graph Visualisation



Currently, there are no publicly available tools for the generic visualization of graph databases. Existing solutions are often proprietary, expensive, or specific to particular application domains.

We have developed an interactive, dynamic, and highly configurable web service designed for the visualization of graph database contents, queries, and analysis results.



Lomikel Browser

Link Data Explorer 02.00.00+ [06/May/2022 at 18:18:38 CEST by centos for UCLab] [Reset](#)

UCLab-Proxy Add
Search AstroLabNet
Execute [Show - Table -](#)

prv_candidate:2459512.8173958

Customize the interactions with the graph:
 Cluster by group type Cluster by group size Expand all clusters Show all edges Hierarchical Up/rl Size/hierarchy Blue
 Clusterize Zoom cluster Rotate Get children Get parents Remove old
filter: select: limit(10)

id label clrcoeff clrcount diffmaglim fid field jd magzpsci magzpscirms magzpsclunc nid pdiffmaglimname
20844949584 prv_candidate -0.0689774 3.70181E-5 20.1174 1 299 2459527.7947685 26.02 0.0342466 2.67963E-5 1773 ztf/archive/sci/2021/1109/294768/ztf_20211109294768_000299_zg_c11_o_q4_scinreldiffmag.fits.gz
id: 20844949584
label: prv_candidate
clrcoeff: -0.0689774
clrcount: 3.70181E-5
diffmaglim: 20.1174
fid: 1
field: 299
jd: 2459527.7947685
magzpsci: 26.02
magzpscirms: 0.0342466
magzpsclunc: 2.67963E-5
nid: 1773
pdiffmaglimname: ztf/archive/sci/2021/1109/294768/ztf_20211109294768_000299_zg_c11_o_q4_scinreldiffmag.fits.gz
pid: 1773294764315
programid: 1
program: Kulkarni
rversion: 117_fs_c3
rcid: 43
lbl: prv_candidate
+ 21992865832 prv_candidate 0.121702 2.35149E-5 20.1729 2 299 24
+ 21992869928 prv_candidate -0.0435896 4.32017E-5 19.6978 1 299 24
+ 21992874024 prv_candidate -0.0866524 3.42425E-5 20.5701 1 299 24
+ 23462101224 prv_candidate -0.062068 3.16767E-5 20.7444 1 299 24
+ 23462105320 prv_candidate 0.120364 2.04689E-5 20.6909 2 299 24
+ 23462109416 prv_candidate -0.0794315 2.80797E-5 20.657 1 299 24
+ 23463370056 prv_candidate 0.134517 2.73896E-5 19.568 2 299 24
+ 26003374152 prv_candidate 0.125216 2.13962E-5 20.5773 2 299 24
+ 27194933248 prv_candidate 0.121286 2.34516E-5 20.5467 2 299 24

Showing 1 to 10 of 10 rows

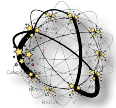
Select graph server and initial graph, then select an element to see possible actions.
Sending Gremlin request to //134.158.74.85:24445: g.V().has('lbl', 'AstroLabNet').limit(10)
Sending Gremlin request to //134.158.74.221:8080:LinkDataExplorerProxy.jsp?server=http://134.158.74.85:24445: g.V().has('lbl', 'Alert').limit(10)
Showing 10 new elements

Lomikel Browser is a Web Service to visualize and any graph with Gremlin API.

Written in JSP, JavaScript, Groovy and Gremlin.

Objects (Vertices and Edges) can be manipulated and interrogated.

It works out-of-the box with the default style, but can be heavily customized by visualization stylesheets and plugins.



Lomikel Browser

Graph context-sensitive operations
And introspection

Other ways of graph analyses
(as plugins)

Other plugins can be called
For more detailed analyses

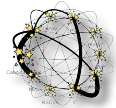
Selection of the Graph server
And initial request

Global operations
on Graph

Graph shown as
an interactive table

The screenshot shows the Lomikel Browser interface. On the left, there is a graph visualization with a central node labeled 'prc_candidate:245952.8173958' and several surrounding nodes connected by edges. The nodes are labeled with IDs like '2459527.7347076', '2459525.7804514', '2459521.7781481', '2459525.8153009', '2459523.8143403', '2459503.8575347', '2459518.8147685', '2459512.8777199', and '2459512.8173958'. On the right, there is a table with columns: 'id', 'label', 'scoreoff', 'scoreons', 'diffmayfm', 'fid', 'field', 'jid', 'maessens', 'maessensms', 'maessensims', 'nid', and 'editInfilecans'. The table contains multiple rows of data, with the first row being: '2044899594 prc_candidate -0.0689774 3.70181E-5 20.1174 1 209 2459527.7347685 26.02 0.032486 2.67903E-5 1773 /zstfarchw/ks/20211109204768uf_20211109'. Below the table, there is a section for 'Showing 1 to 10 of 10 rows' and a footer with instructions: 'Select graph server and initial graph. Then select an element for new context-sensitive actions.' and 'Sending Golem request to /134.158.74.85:24445 g VU has/IK/ /Admin/alter/ limit(10)'.

Interactive view of a Graph
(Customisable by a stylesheet)



Customisation

Stylesheet

```
stylesheet.nodes.datalink = {
  properties:{gremlin:"valueMap('name', 'technology').toList()[0]"},
  graphics: {
    label:"name",
    title:"name",
    subtitle:"technology",
    group:" ",
    shape:"dot",
    image:"",
    borderRadius:"0",
    borderWidth:"1",
    borderDashes:[1,0],
    value:"0"
  },
  actions:[
    {name:"Link", url:{gremlin:"id().next().toString().replaceFirst(\"^\", \"DataLink.jsp?id=\")"}, target:"result" },
    {name:"Fits", url:{gremlin:"id().next().toString().replaceFirst(\"^\", \"DataLinkFits.jsp?id=\")"}, target:"result" },
    {name:"Show", url:{gremlin:"id().next().toString().replaceFirst(\"^\", \"Node.jsp?id=\")"}, target:"result" },
    {name:"Table", url:{gremlin:"id().next().toString().replaceFirst(\"^\", \"Nodes.jsp?id=\")"}, target:"table" }
  ]
}

stylesheet.nodes.alert = {
  properties:{gremlin:"valu
  graphics: {
    label:"lbl",
    title:"lbl",
    subtitle:" ",
    group:{gremlin:"values(
    shape:"hexagon",
    image:"",
    borderRadius:"0",
    borderWidth:"2",
    borderDashes:[1,1],
    value:{gremlin:"out().o
  },
  actions:[
    {name:"Show", url:{g
    {name:"Table", url:{g
  ]
}
```

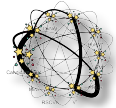
Stylesheet is a JSON document, describing possible Vertices and Edges, containing scripts in Gremlin or JavaScript.

Plugins can specify

- how Vertices/Edges are shown
- their context-sensitive operations as either call to internal plugins or external services.

Many **standard plugins** exist:

- correlations/overlaps (i.e. properties of Edges between Vertices) as table and Venn diagrams
- scatterplots for Vertex/Edge properties
- time dependence of Vertex/Edge properties (if time property defined)
- visualization of embedded data (pictures,...)
- navigation to connected databases (SQL, NoSQL or Graph)



Atlascope 01.02.00+ [24/Apr/2021 at 10:12:16 CEST by atdev for CERN] [Reset](#)

Search: ATLAS
Execute: g.V().has('lbl', 'canonical')

Actions:

Graphs: Image Plot

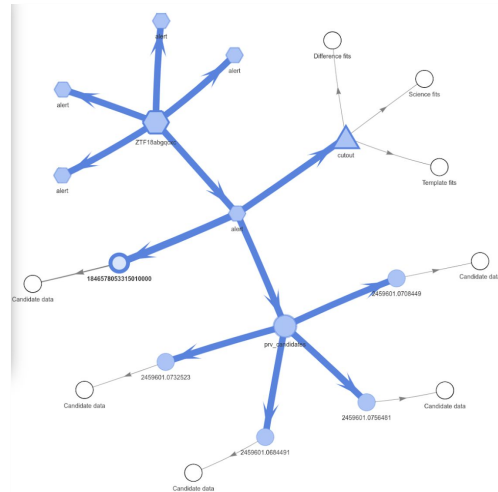
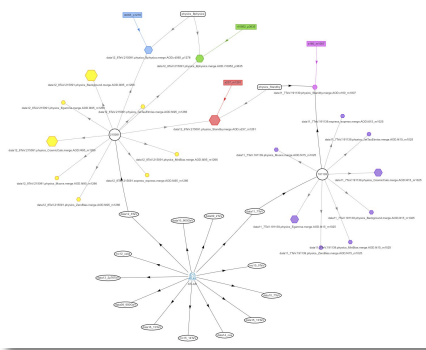
Customize the interactions with the graph:
 Cluster by group type | Cluster by group size | Expand all clusters | Show all edges | ...
 Clusterize | Zoom cluster | stabilize | Get children | Get parents | remove old
 filter: [min:10] Apply Select [min:10]

canonical-AOD
10221559 events

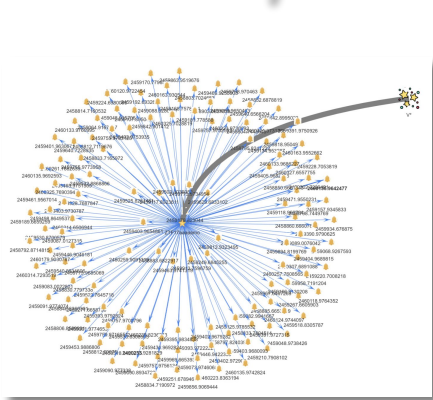
```

version: 1832 @ 1812
Pulsio: 1526716
project: data17_13TeV
streamers: physics_Pv1a
prodtop: morse
dataset: data17
displayid: 1832
986 12553
events: nuclo:10221559
pulsio: 25-Tue Mar 26 12:34:18 CET 2017
files: 742
events: 10221559
updated: at Tue Mar 30 09:29:07 CEST 2021
is_implicit: false
status: ZIPPED
has_rmc: true
has_trigger: true
prod: canonical
prodid: canonical
prodid: true
fullfill: true
  
```

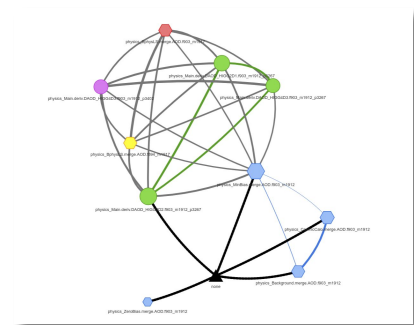
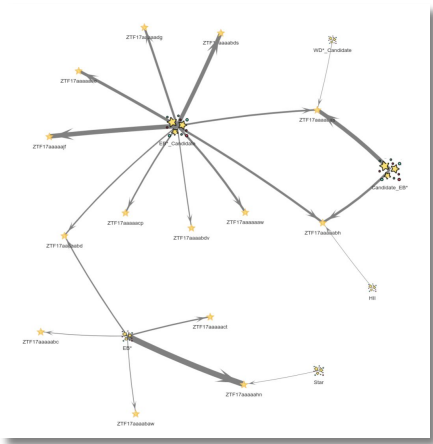
Examples



Vertex introspection

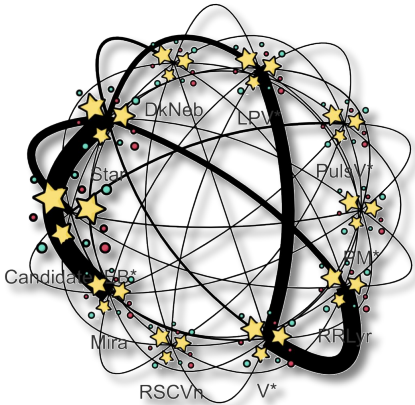


Graph with relations to data in external database (HBase in this case)



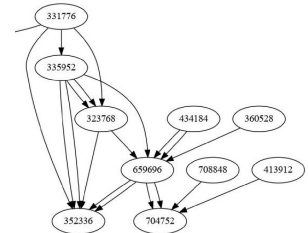
Using and Visualizing Graphs and Graph Algorithms

using graph techniques to investigate LSST alerts



1. Accumulate all data in classical database (SQL/NoSQL)
2. Extract **Vertices** into Graph, keeping backlink to full data
3. Create important **Edges** using appropriate **metric** capturing important relations
4. Derive new relations (like overlaps, neighborhoods,...)
- using graph DB utilities
5. Analyse relations (find clusters, isolated elements,...)
- using graph toolkits

Plans to integrate in user interface (Web Portal)
So that users can trigger Graph algorithms



➤ **Lomikel**

- Home: <https://hrivnac.web.cern.ch/hrivnac/Activities/Packages/Lomikel>
- Git: <https://github.com/hrivnac/Lomikel.git>

➤ **Grapher**

- Home: <https://hrivnac.web.cern.ch/hrivnac/Activities/Packages/Grapher>
- Git: <https://github.com/hrivnac/Grapher.git>