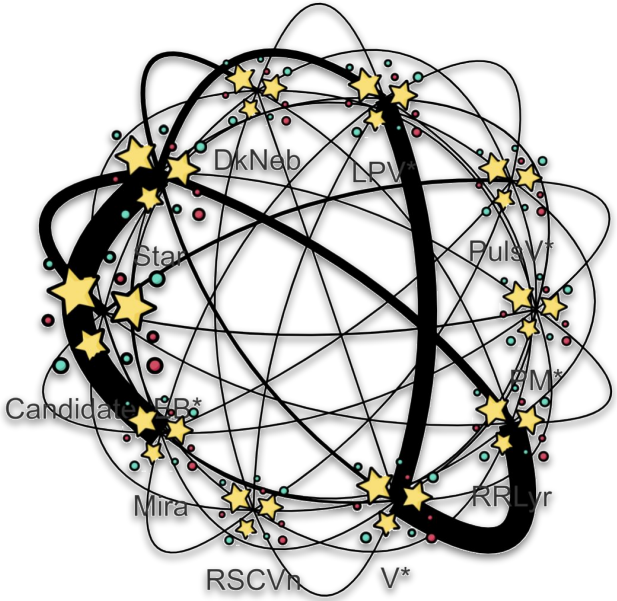
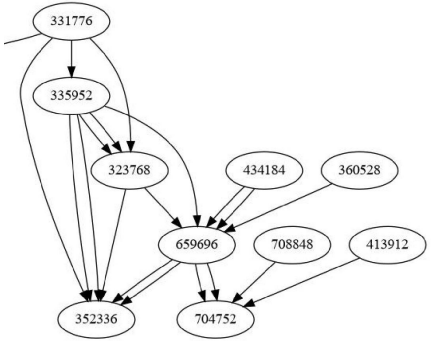
Graphs in the Sky

using graph techniques to investigate alerts

exploring possibilities to handle relational and collective alerts properties



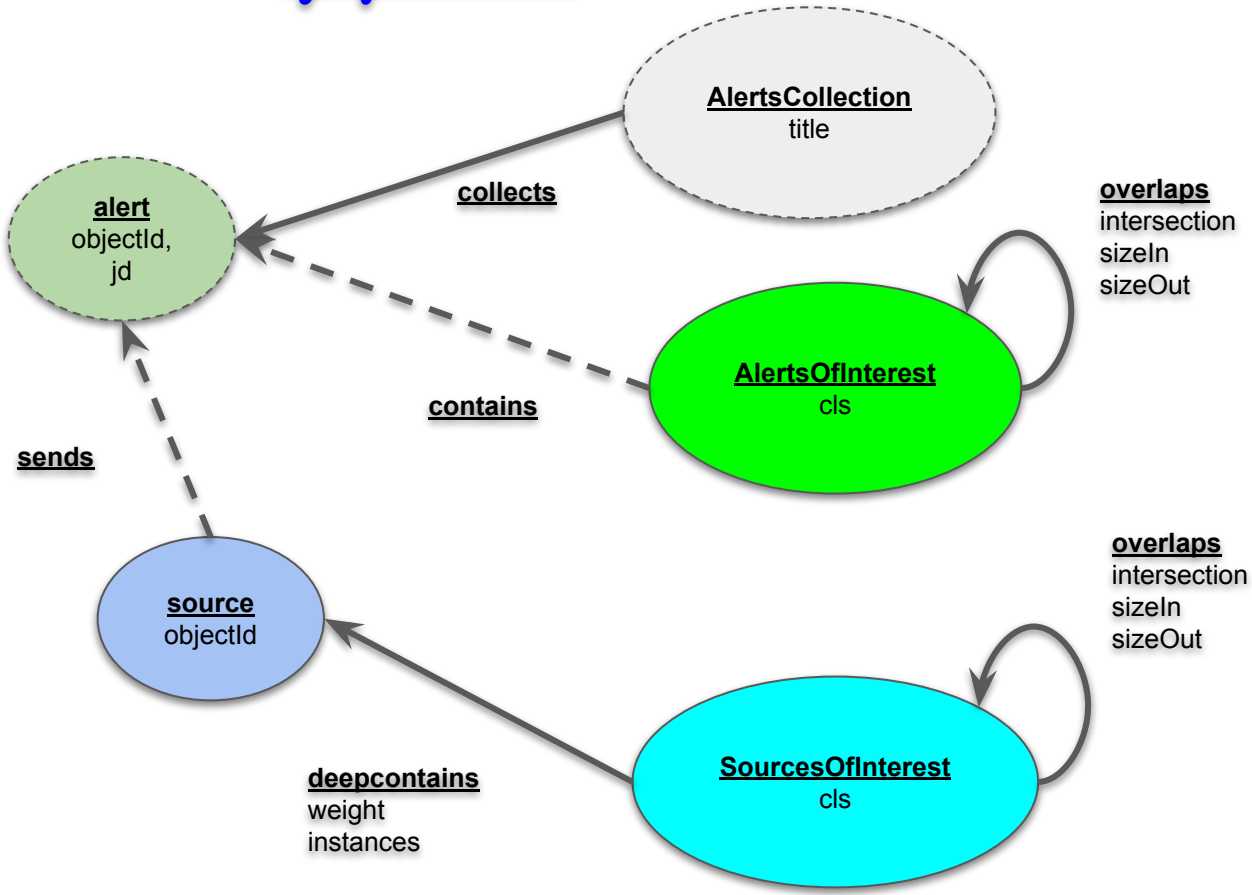
- **Classification graphs**
  - Schema
  - Construction
  - Overlaps
  - Neighbourhood
- Exporting and analysing graphs



Julius Hrivnac, IJCLab

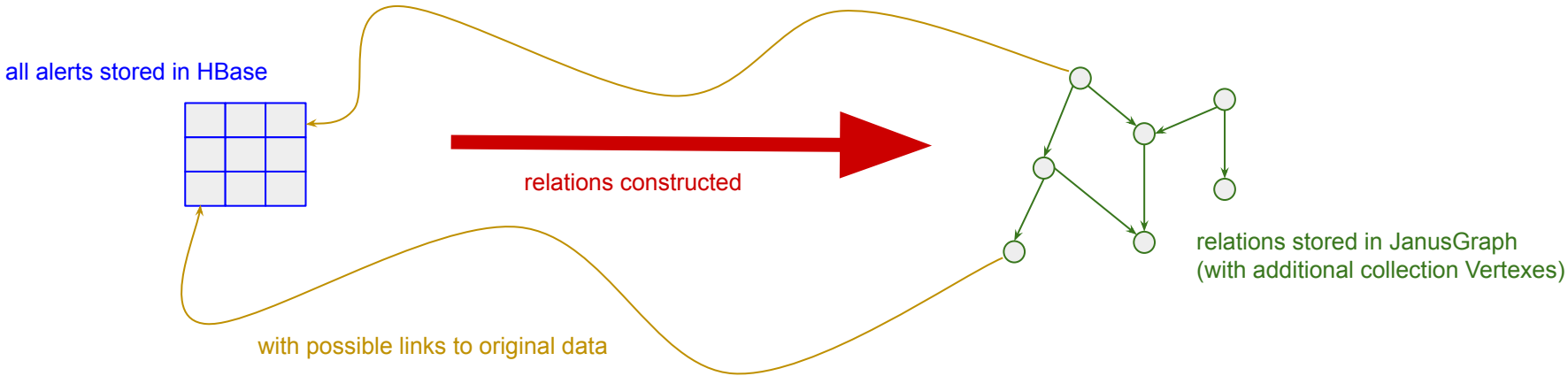
Fink-Brazil Workshop
Centro Brasileiro de Pesquisas Fisicas
6/May/2024

Graph Schema



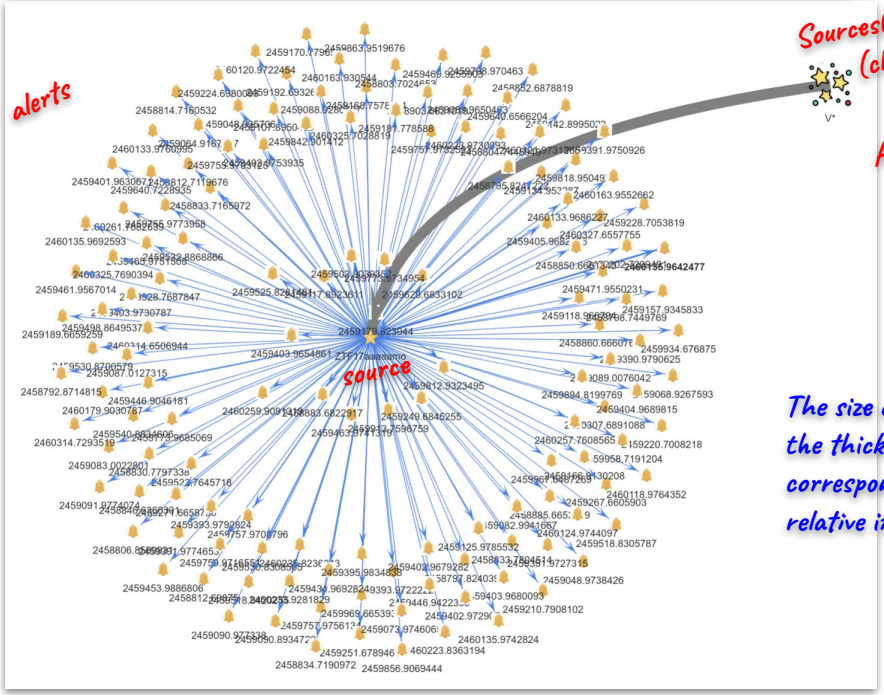
Constructing the Graph

- Import new sources (or update)
  - select interesting (latest, classified as anomaly,...) sources (from HBase or Fink Portal)
  - import them into graph
  - expand to alerts (optional)
  - fill with properties from HBase (optional)
  - construct SoI and AoI collections for alert classes (using classification from Fink Portal)
- Calculate all overlaps (after new imports, cca 1 min)
- Calculate sources similar to one source (per request, cca 1 min)



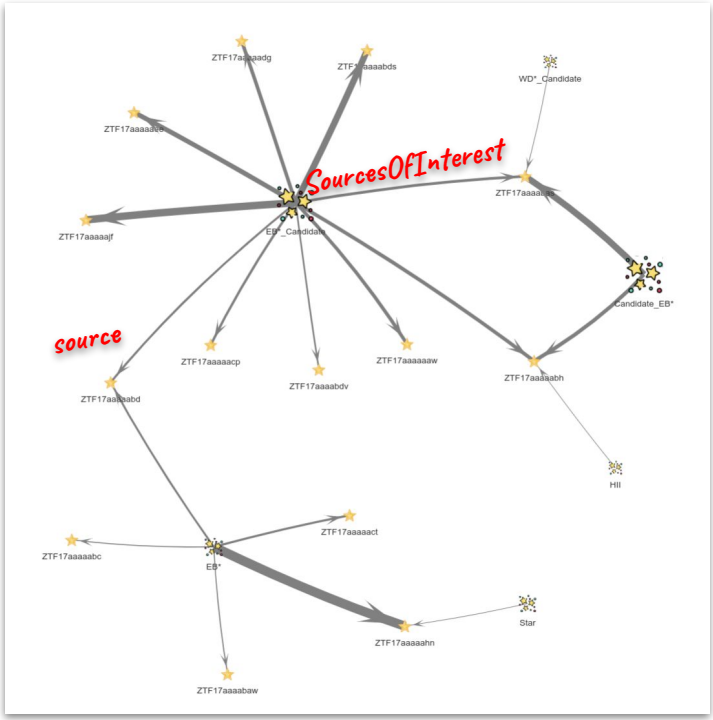
The Graph

One source of one SourcesOfInterest (V*) and all contained alerts



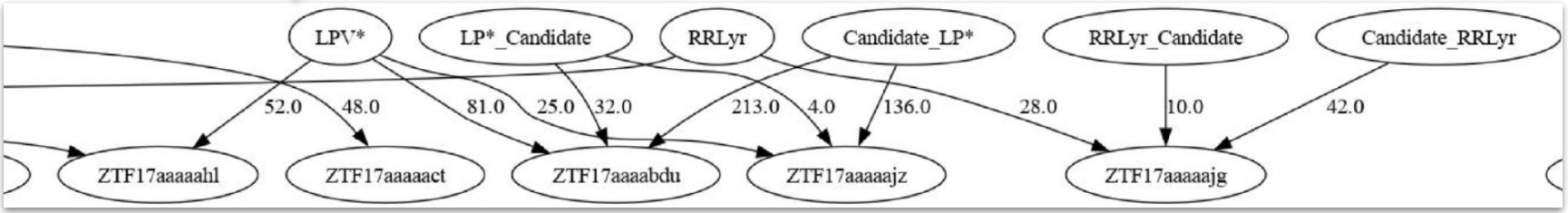
The size of vertexes and the thickness of edges correspond to their relative importance

Several SourcesOfInterest and some of their sources



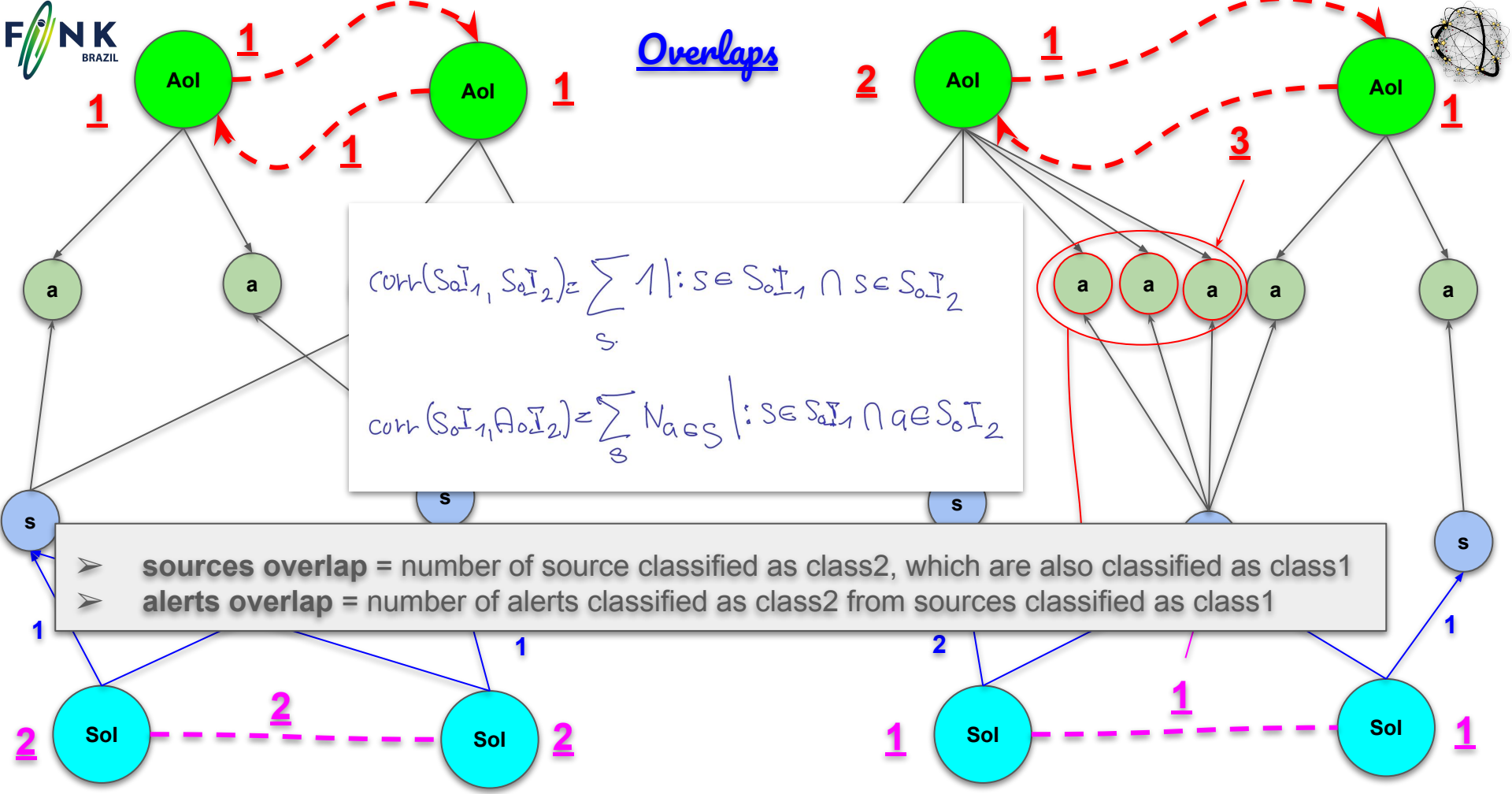
Graph (exported to GraphViz)

SourcesOfInterest (classes)



source

Overlaps

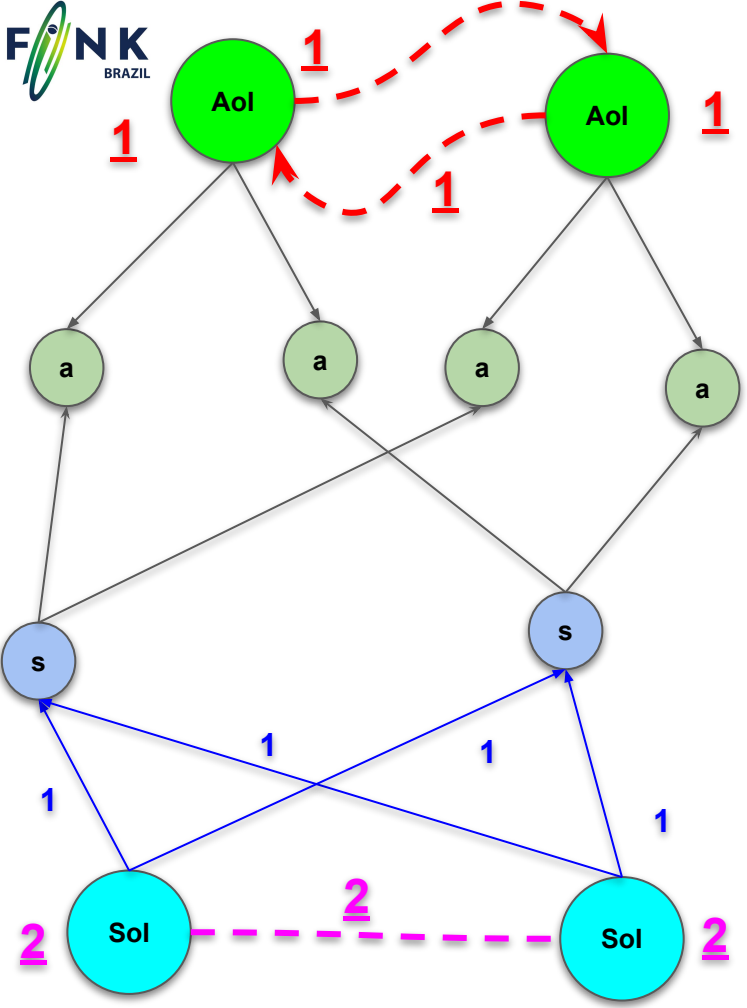


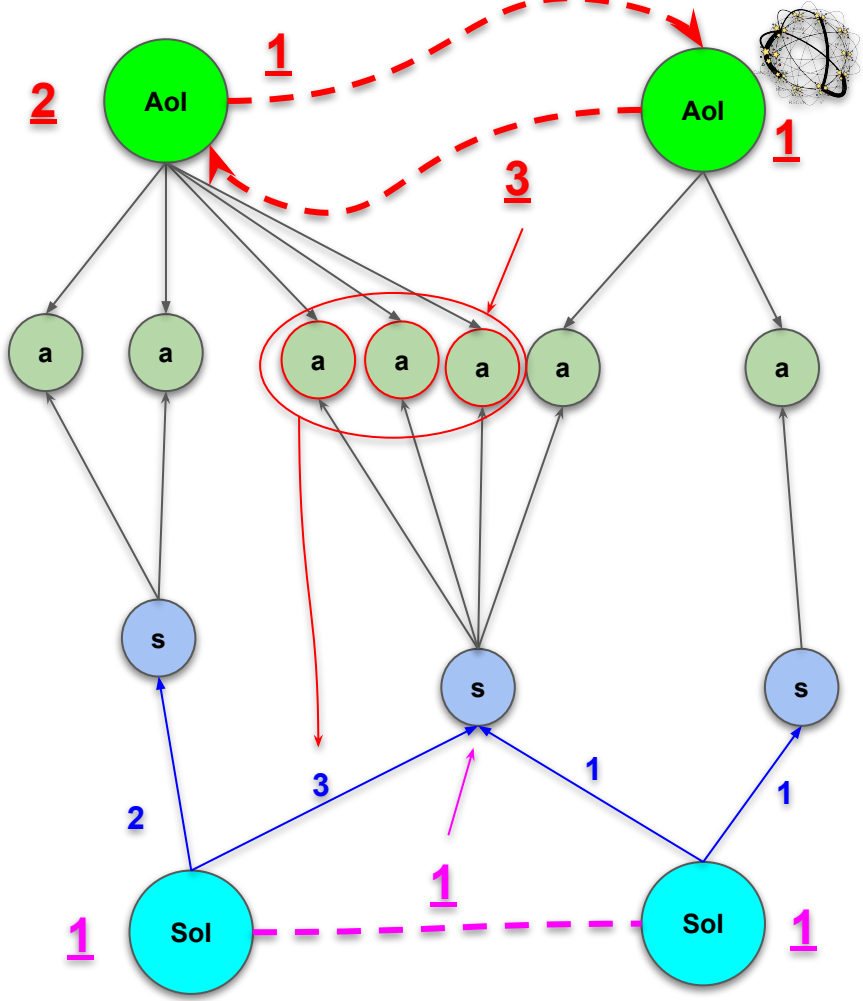
$$corr(SoI_1, SoI_2) = \sum_{s} 1 \;|\; s \in SoI_1 \cap s \in SoI_2$$

$$corr(SoI_1, AoI_2) = \sum_{s} N_{a \in s} \;|\; s \in SoI_1 \cap a \in SoI_2$$

- **sources overlap** = number of source classified as class2, which are also classified as class1
- **alerts overlap** = number of alerts classified as class2 from sources classified as class1

Overlaps



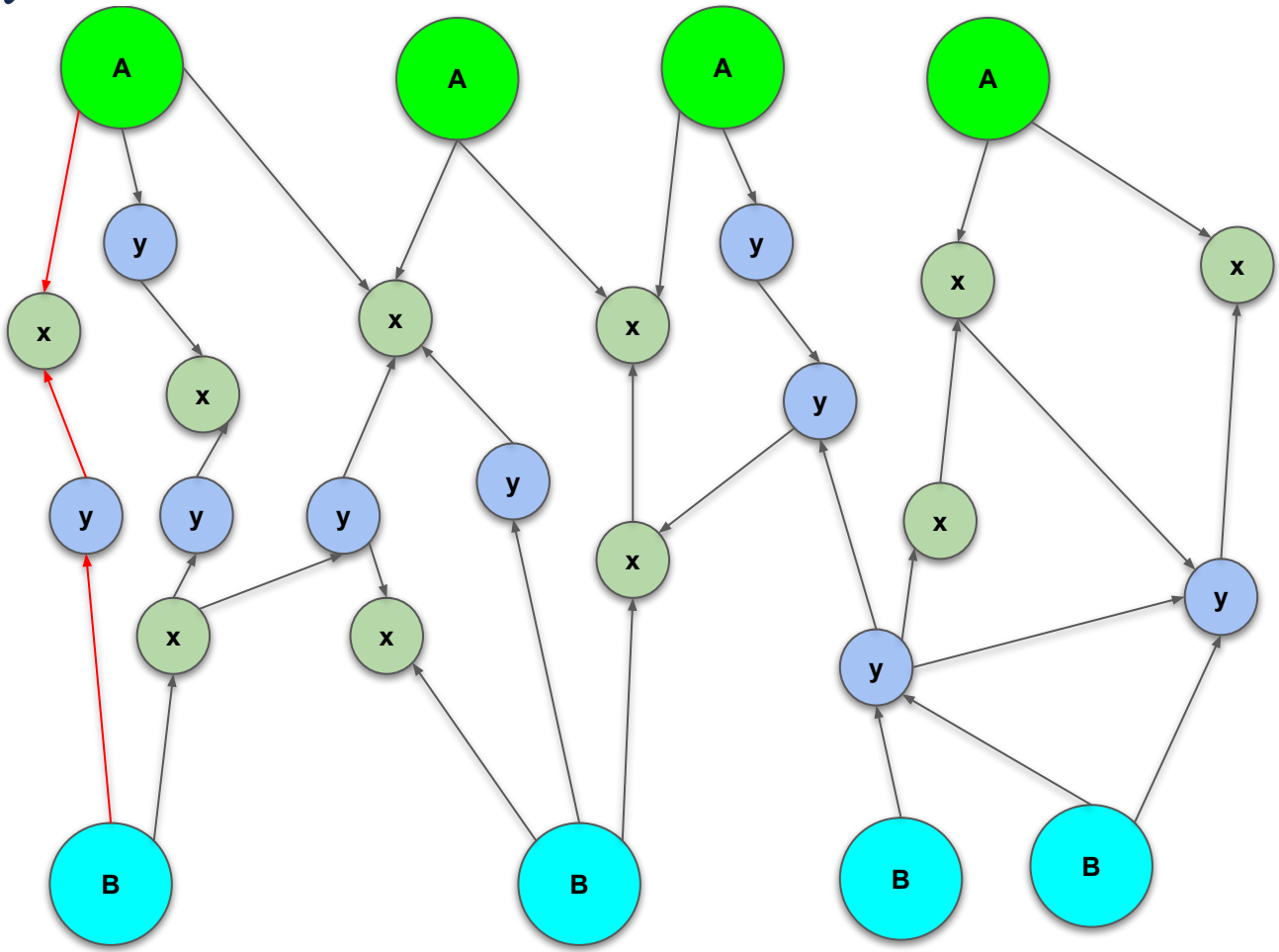


weight = number of **a** connecting **Sol** and **Aol** via **s**

number of **s** common to two **Sol**

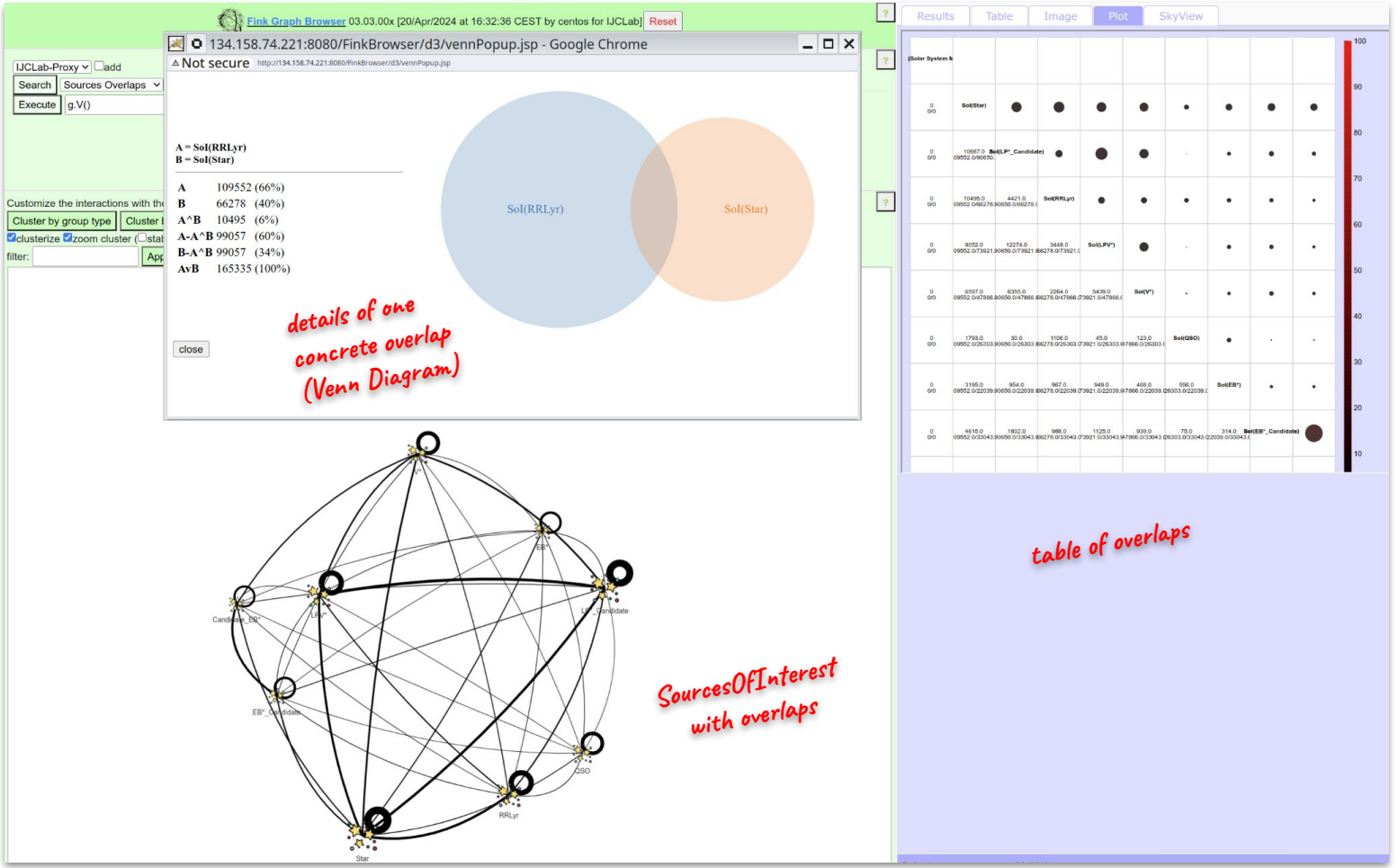
number of **a** from the other **Sol**

Generalisation of Overlaps

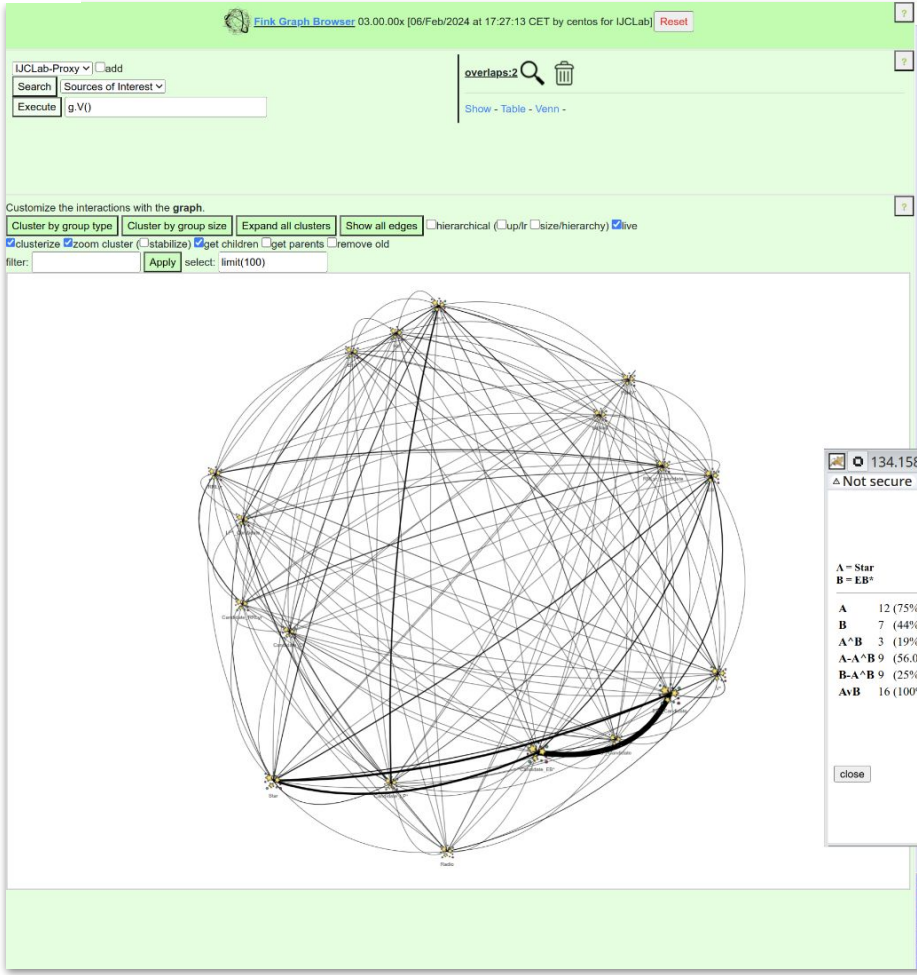


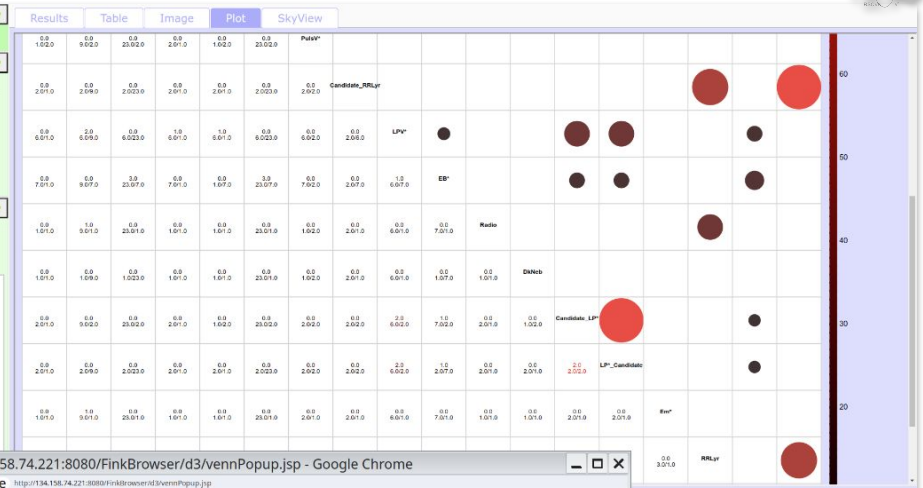
- find overlaps between A and B with respect to z
- how to take into account Edge weights ?
- how to include multiple-paths to the same x ?
- should work for A == B
- should work for oriented and unoriented Edges

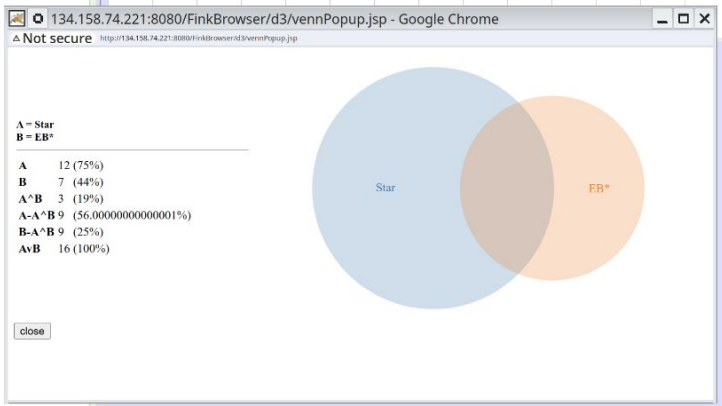
Visual Overlaps



Visual Overlaps







Access to Overlaps

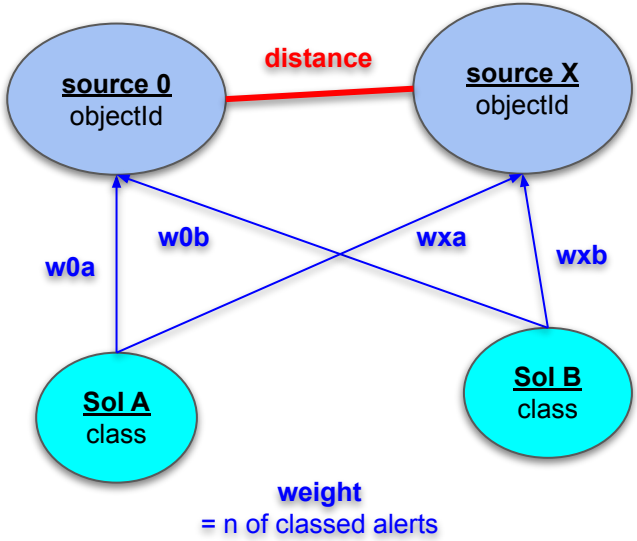
script

```
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
...
```

Source Neighborhood



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

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

Access to Source Neighborhood

script

```
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, ZTF21aaahkto=0.008679829271707878,
ZTF18abdlapp=0.011986363351189971}
```

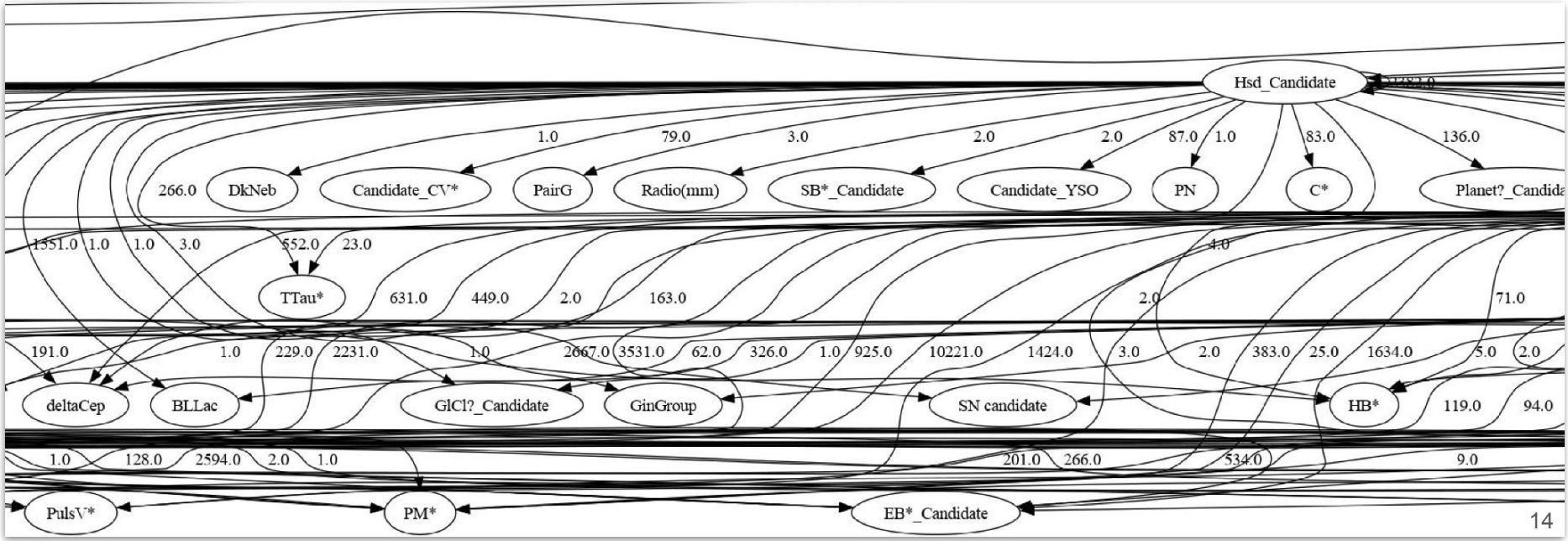
Exporting Overlaps (to GraphViz)

script

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

conversion

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



Analysing Exported Overlaps

script

```
> 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);
```

result

```
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

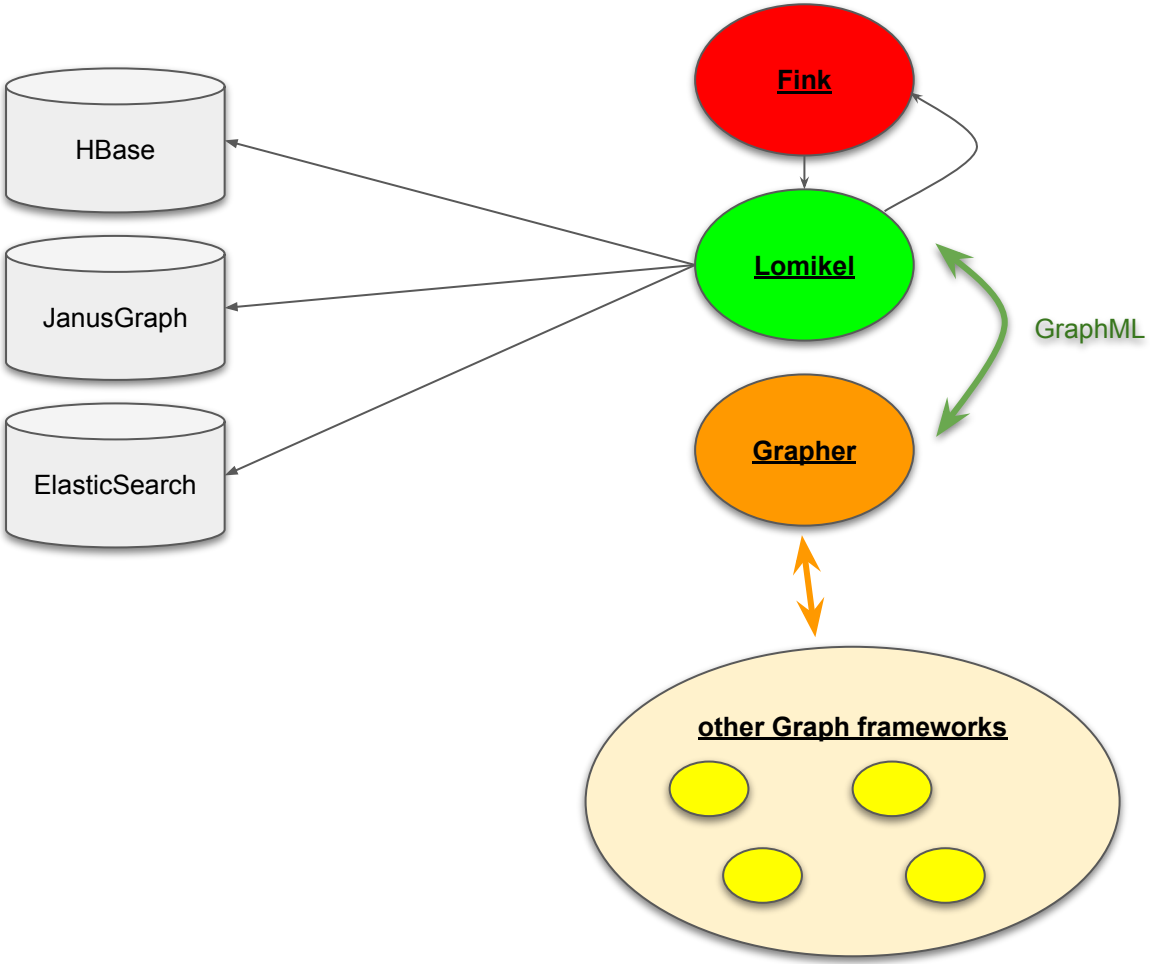
script

```
> 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);
```

result

```
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 algoritm
    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]
    ...
```

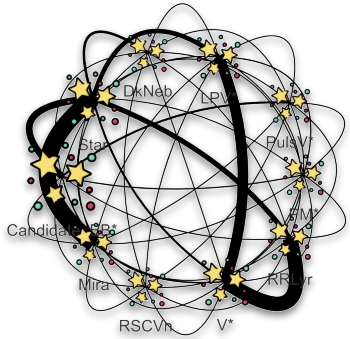
Ecosystem



- Lomikel, Grapher exist as
  - standalone programs, scriptable in Java, Python, Groovy
  - libraries to be included in other Java, Python, Groovy programs
  - interactive Web Service

How To

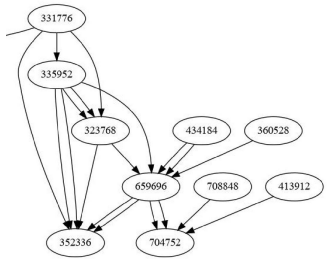
execution




```
# download Lomikel-Janus.exe.jar
# and
# jython.jar (if Python scripting is required)
# if Python scripting is not required:
$ alias lomikel='java -jar Lomikel-Janus.exe.jar'
# if Python scripting is required:
$ alias lomikel='java -cp Lomikel-Janus.exe.jar:jython.jar com.Lomikel.Apps.LUC'
$ 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] (othewise taken from source extension, defauld 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 a
```

Lomikel

execution

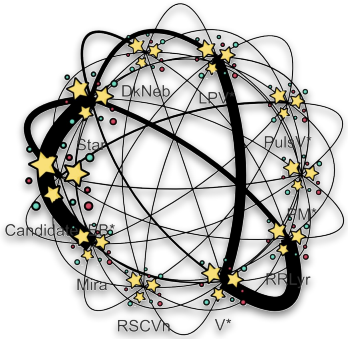



```
# download Grapher.exe.jar
$ alias grapher='java -jar Grapher.jar'
$ grapher -h
usage: java -jar Grapher.exe.jar
 -a,--alg   apply algorithm (instead of just converting)
                      [sc = Strong Connectivity | cl = Clustering | ad = adding distances]]
                      several algorithms 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)
```

Grapher

some Lomikel actions
require direct access to
HBase/JanusGraph/ElasticSearch databases

Lomikel Script

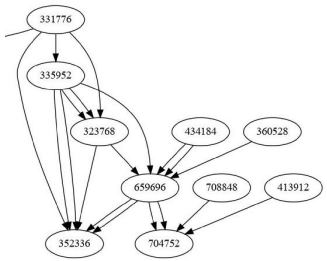


script

```
import com.Lomikel.Januser.JanusClient;
import com.astrolabsoftware.FinkBrowser.Januser.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');
```

Lomikel

Grapher Script

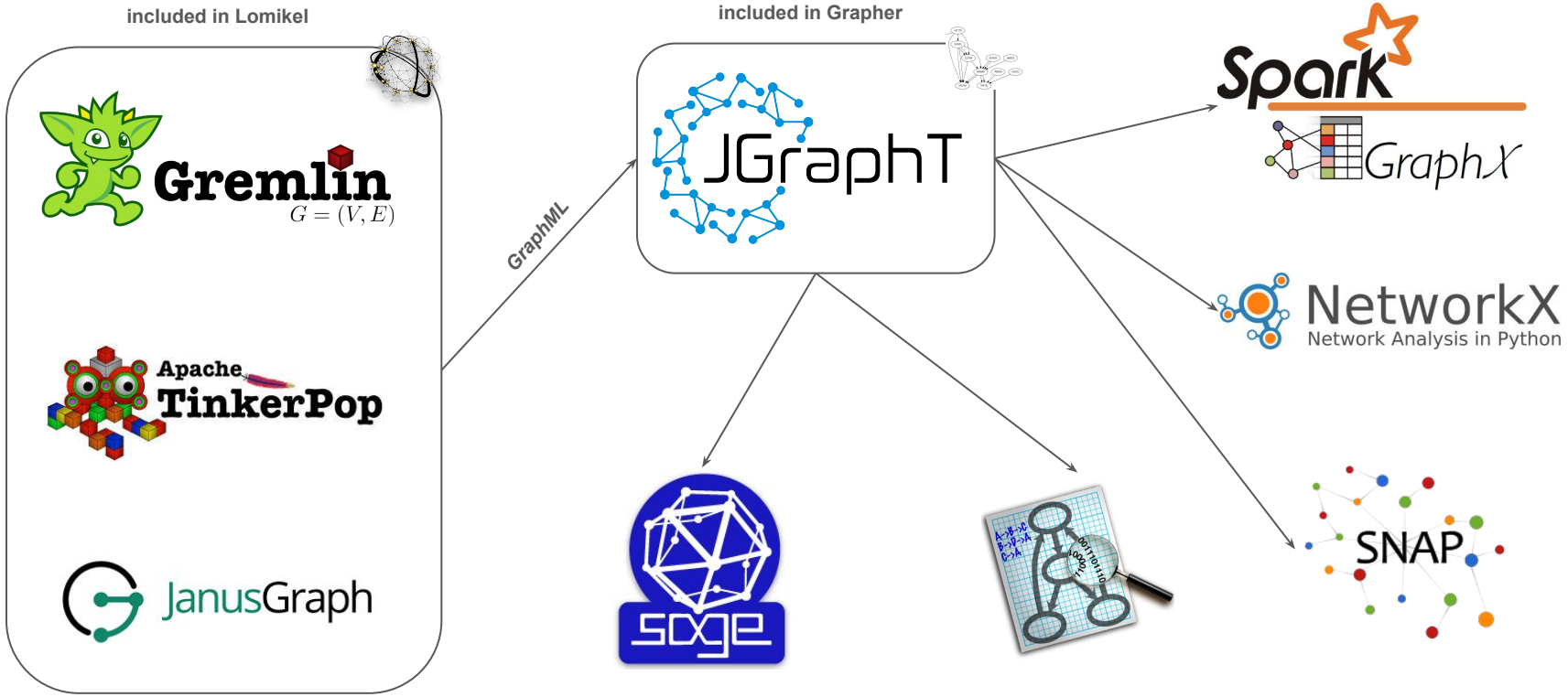


script

```
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);
```

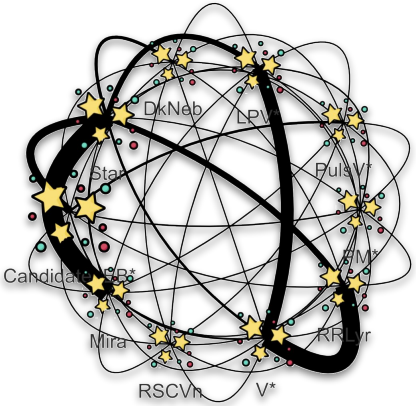
Grapher

Other Graph Frameworks



Graphs in the Sky

using graph techniques to investigate alerts



- **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
- Next steps:
  - simplify user access
  - integrate into Fink Portal (recommendation)
  - streamline alerts injection