End of B.I.R presentation

Lowagie Célia

Université de Mons

UMONS Université de Mons

Faculté des Sciences

4 septembre 2024

Goal

The main goal was to identify the different collaboration between the different organisations of NumFocus.

Goal

The main goal was to identify the different collaboration between the different organisations of NumFocus.
A sub-goal was to create a graph of all the interactions between the contributors.

What is NumFocus ?

An organisation that gathers and finances important open-source project related to data science.

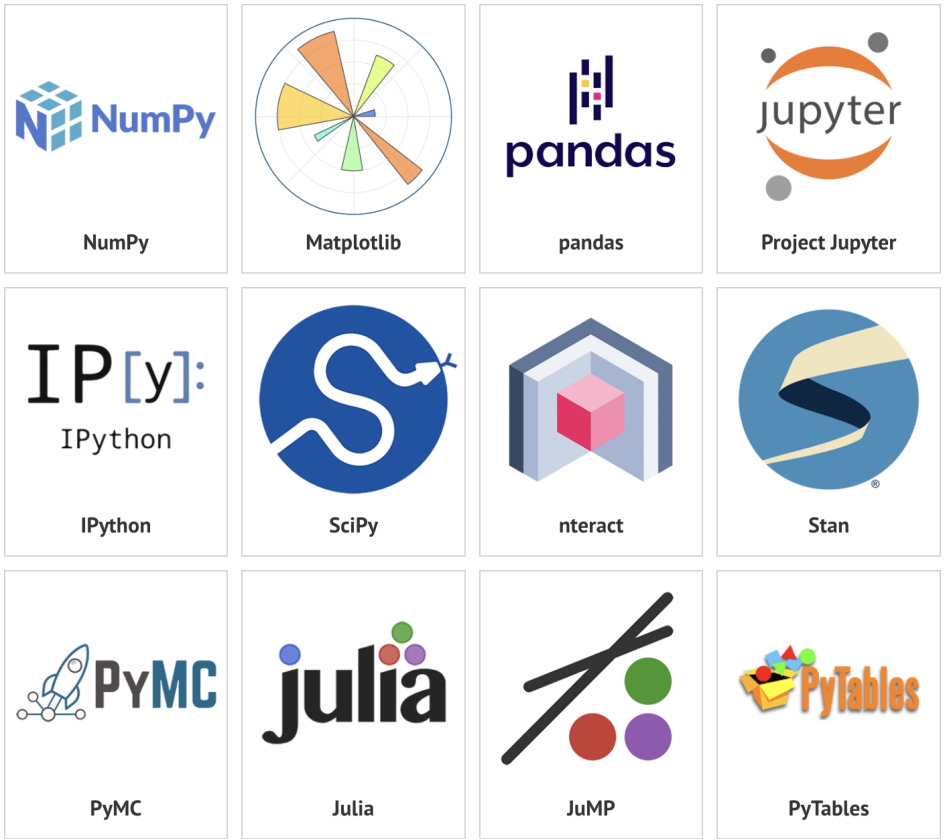


Figure – Example of projects

First approach

The first idea to simplify the graph :

- Nodes = contributors
- Edge = contributed same organisation

First approach

The first idea to simplify the graph :

- Nodes = contributors
- Edge = contributed same organisation

Creation of the graph

- Preprocessing the data :
  1. Removing organizations (conda-forge, Bioconductor, openjournals)
  2. Merging all the data extracted by Youness into a large dataframe

First approach

The first idea to simplify the graph :

- Nodes = contributors
- Edge = contributed same organisation

Creation of the graph

- Preprocessing the data :
  1. Removing organizations (conda-forge, Bioconductor, openjournals)
  2. Merging all the data extracted by Youness into a large dataframe
- Adding all the contributors from the dataframe as nodes

First approach

The first idea to simplify the graph :

- Nodes = contributors
- Edge = contributed same organisation

Creation of the graph

- Preprocessing the data :
  1. Removing organizations (conda-forge, Bioconductor, openjournals)
  2. Merging all the data extracted by Youness into a large dataframe
- Adding all the contributors from the dataframe as nodes
- For each organisation, linking the contributors of the organisation together (for each organisation saving the new file)

Results and graph

Here is the graph obtained for the data from June 2024

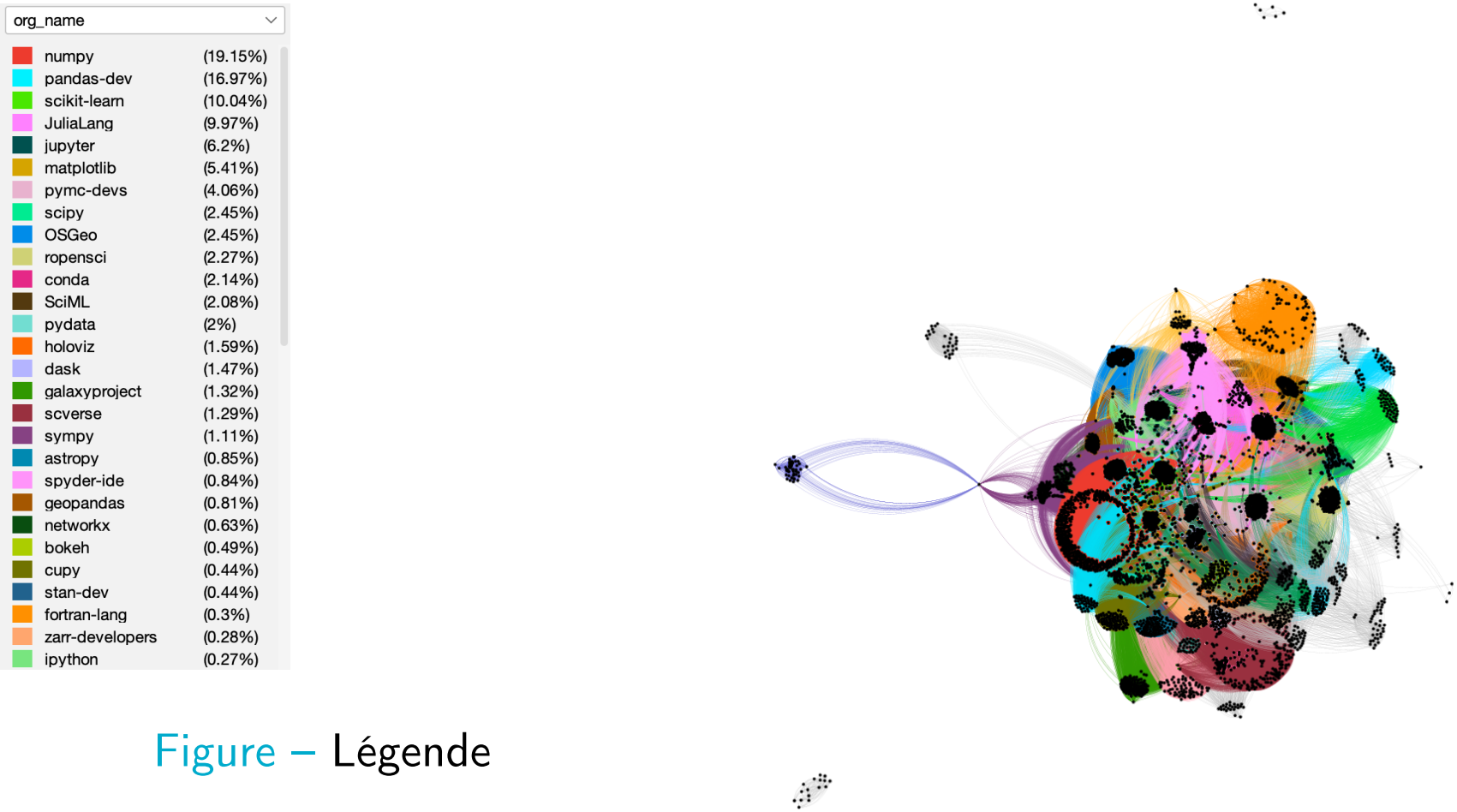


Figure – Légende

Results and issues

Observations :

- Two organisations are not connected at all to others : petsc and open-MBEE (toolkit for engineer)
- For one month of data, we reach 1 672 781 edges and the graph takes 409 MB of storage.

Results and issues

Observations :

- Two organisations are not connected at all to others : petsc and open-MBEE (toolkit for engineer)
- For one month of data, we reach 1 672 781 edges and the graph takes 409 MB of storage.

Issues encountered

- Crash due to lack of memory
- Crash due to lack of storage (fixed by only saving the last to files)
- Impossible to finish the graph with all the data, I could add the edges for only 16 organisation on 58.
- Edges too general and not giving enough information about events

Second approach

To fix the generality problem and try to fix the storage/memory issue, the next approach is :

- Nodes = contributors (that have at least one edge)
- Edge = contributed in the same event

How to determine which event(s) to use as edge

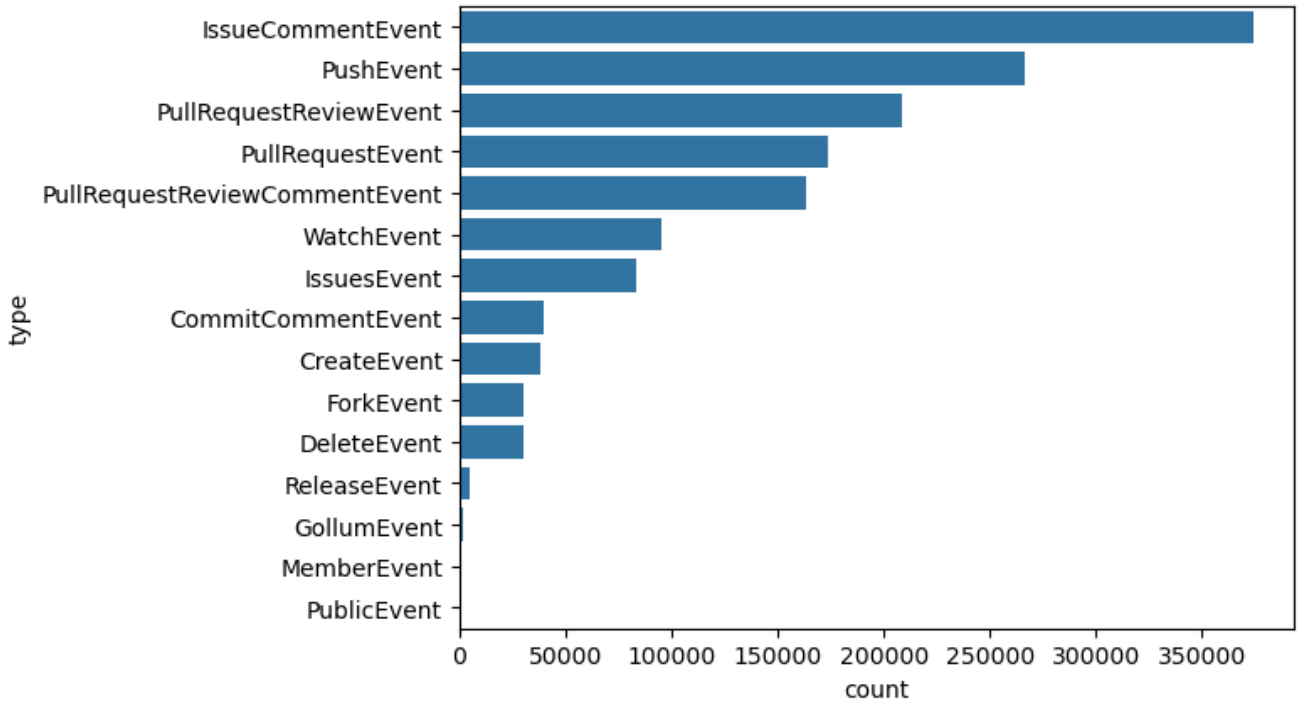


Figure – Event types in the whole dataset

To identify the relevant event types, I read the Github doc on events and explored a bit their payload

How to determine which event(s) to use as edge

I only kept events related to issue or pull request or review.

How to determine which event(s) to use as edge

I only kept events related to issue or pull request or review.
The other event were :

- either not directly collaborative (e.g. WatchEvent := someone starting a repo)

How to determine which event(s) to use as edge

I only kept events related to issue or pull request or review.
The other event were :

- either not directly collaborative (e.g. WatchEvent := someone starting a repo)
- either not giving enough information to easily access the collaborators (e.g. MemberEvent := adding/deleting someone to/from a repo)

How to identify the collaborators in a given event

By reading the payload, I found that issue related event had an issue_id.
Pull request and review also have a specific id.

How to identify the collaborators in a given event

By reading the payload, I found that issue related event had an issue_id. Pull request and review also have a specific id.

Definition

Two people collaborates on the same event if they both took part in an event that either have the same issue id or pull request id or review id

Information stored in the graph

We can either store information on the edges or on the nodes.
Nodes :

- The contributor name (actor_name)
- The contributor type and confidence given by a file from Youness (user_type, confidence)

Information stored in the graph

We can either store information on the edges or on the nodes.
Nodes :

- The contributor name (actor_name)
- The contributor type and confidence given by a file from Youness (user_type, confidence)
- The number and name of events, repositories and organisations where the contributor made an event related to pull request, review or issue (excluding those where no one else contributed)

Information stored in the graph

We can either store information on the edges or on the nodes.

Nodes :

- The contributor name (actor_name)
- The contributor type and confidence given by a file from Youness (user_type, confidence)
- The number and name of events, repositories and organisations where the contributor made an event related to pull request, review or issue (excluding those where no one else contributed)

Edges :

- The organisation name and repository name (org_name, repo_name)
- The event type, event url, event id and id type (either, pull_request_id, review_id or issue_id)

Information stored in the graph

We can either store information on the edges or on the nodes.

Nodes :

- The contributor name (actor_name)
- The contributor type and confidence given by a file from Youness (user_type, confidence)
- The number and name of events, repositories and organisations where the contributor made an event related to pull request, review or issue (excluding those where no one else contributed)

Edges :

- The organisation name and repository name (org_name, repo_name)
- The event type, event url, event id and id type (either, pull_request_id, review_id or issue_id)
- The event date and time

Information stored in the graph

We can either store information on the edges or on the nodes.

Nodes :

- The contributor name (actor_name)
- The contributor type and confidence given by a file from Youness (user_type, confidence)
- The number and name of events, repositories and organisations where the contributor made an event related to pull request, review or issue (excluding those where no one else contributed)

Edges :

- The organisation name and repository name (org_name, repo_name)
- The event type, event url, event id and id type (either, pull_request_id, review_id or issue_id)
- The event date and time
- A weight defined by the number of directed interaction on a same event between two given contributors

How to construct the graph

Methodology :

1. Pre-processing the data and adding three columns, one for each id type found

How to construct the graph

Methodology :

1. Pre-processing the data and adding three columns, one for each id type found
2. For each id type, we iterate on the event id
    1. We exclude events where only one contributor has participated

How to construct the graph

Methodology :

1. Pre-processing the data and adding three columns, one for each id type found
2. For each id type, we iterate on the event id
    1. We exclude events where only one contributor has participated
    2. We add as nodes all the contributors related to this specific event and the related information (if they weren't already in the graph. It's verified effectively with a dictionary stored as a property of the graph).

How to construct the graph

Methodology :

1. Pre-processing the data and adding three columns, one for each id type found
2. For each id type, we iterate on the event id
    1. We exclude events where only one contributor has participated
    2. We add as nodes all the contributors related to this specific event and the related information (if they weren't already in the graph. It's verified effectively with a dictionary stored as a property of the graph).
    3. We link contributors with the same event id following a specific strategy and add information on the edge

Linking strategies

We explored three different strategies to link the nodes together.

1. The naïve one : We define a directed edge as someone answering to one and only one person. So for each events, a person is linked to the directly previous one

Linking strategies

We explored three different strategies to link the nodes together.

1. The naïve one : We define a directed edge as someone answering to one and only one person. So for each events, a person is linked to the directly previous one
2. The hyper-connected one : We connect all the contributors that worked on the same event together

Linking strategies

We explored three different strategies to link the nodes together.

1. The naïve one : We define a directed edge as someone answering to one and only one person. So for each events, a person is linked to the directly previous one
2. The hyper-connected one : We connect all the contributors that worked on the same event together
3. The semi-connected one : We define a directed edge as someone answering to all the previous people on a given event

Strategies example

Here is a table of people contributing on the same event. Let's see how different their graph will be depending on the strategy.

Person	A	B	C
Event Date	6 September	5 September	4 September

Strategies example

Here is a table of people contributing on the same event. Let's see how different their graph will be depending on the strategy.

Person	A	B	C
Event Date	6 September	5 September	4 September

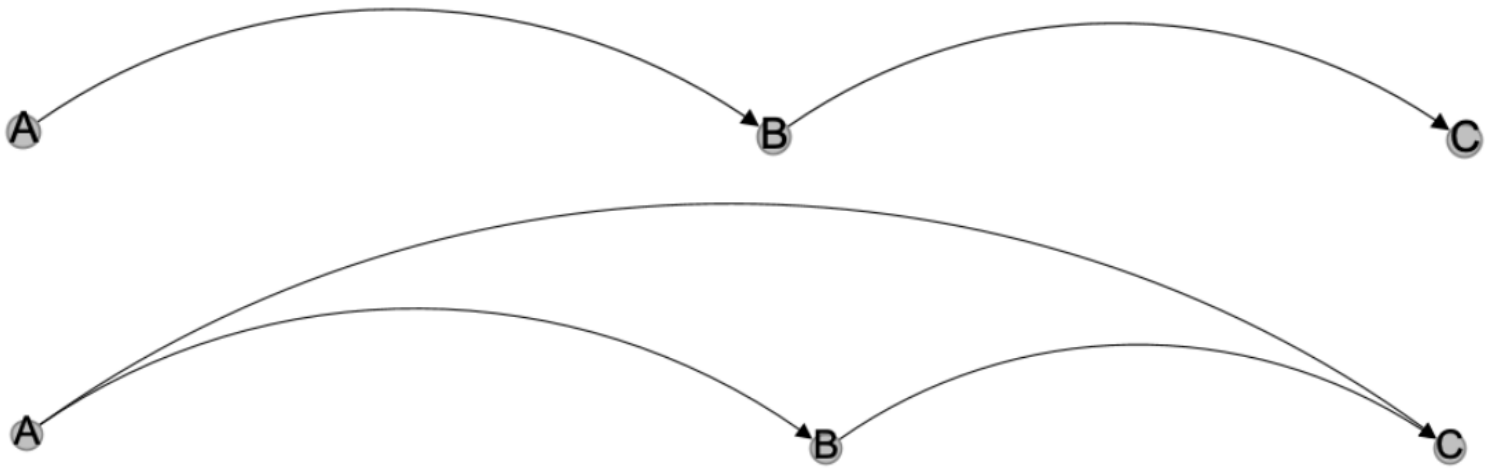


Figure – Naive and semi-connected strategy

Strategies example

Here is a table of people contributing on the same event. Let's see how different their graph will be depending on the strategy.

Person	A	B	C
Event Date	6 September	5 September	4 September

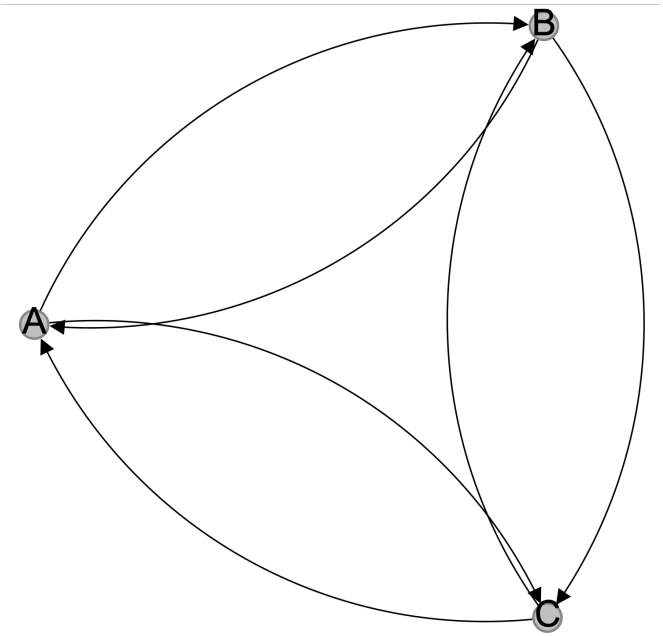


Figure – hyper-connected strategy

Questions about strategies

- Is there a strategy that is more optimal ?
- Is there a strategy that shows better the interaction between the contributors ?

Comparing the strategies

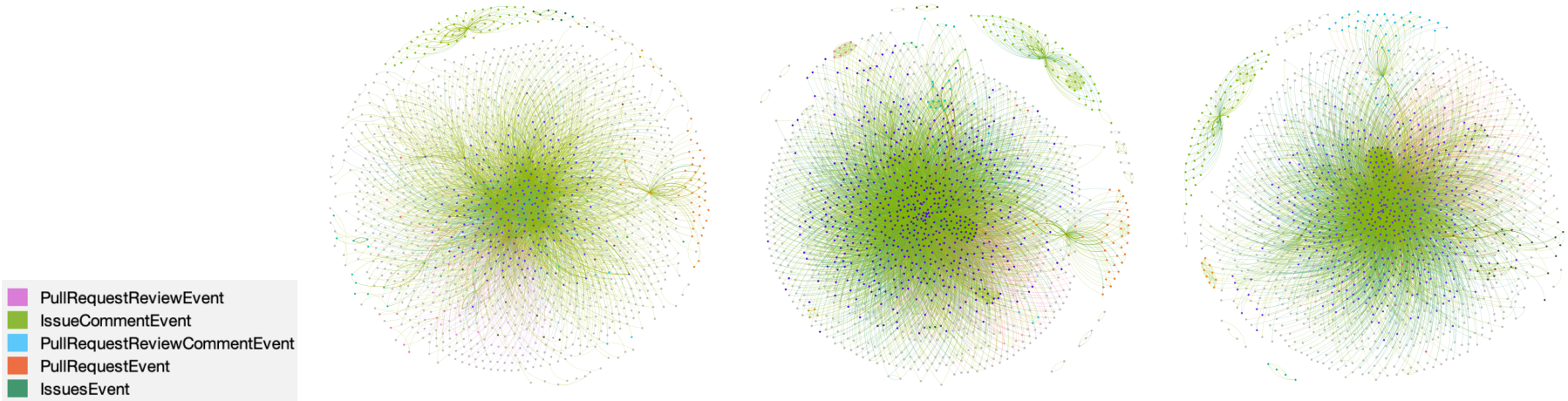


Figure – Naive, hyper-connected and semi-connected strategy

Explanation on graph density

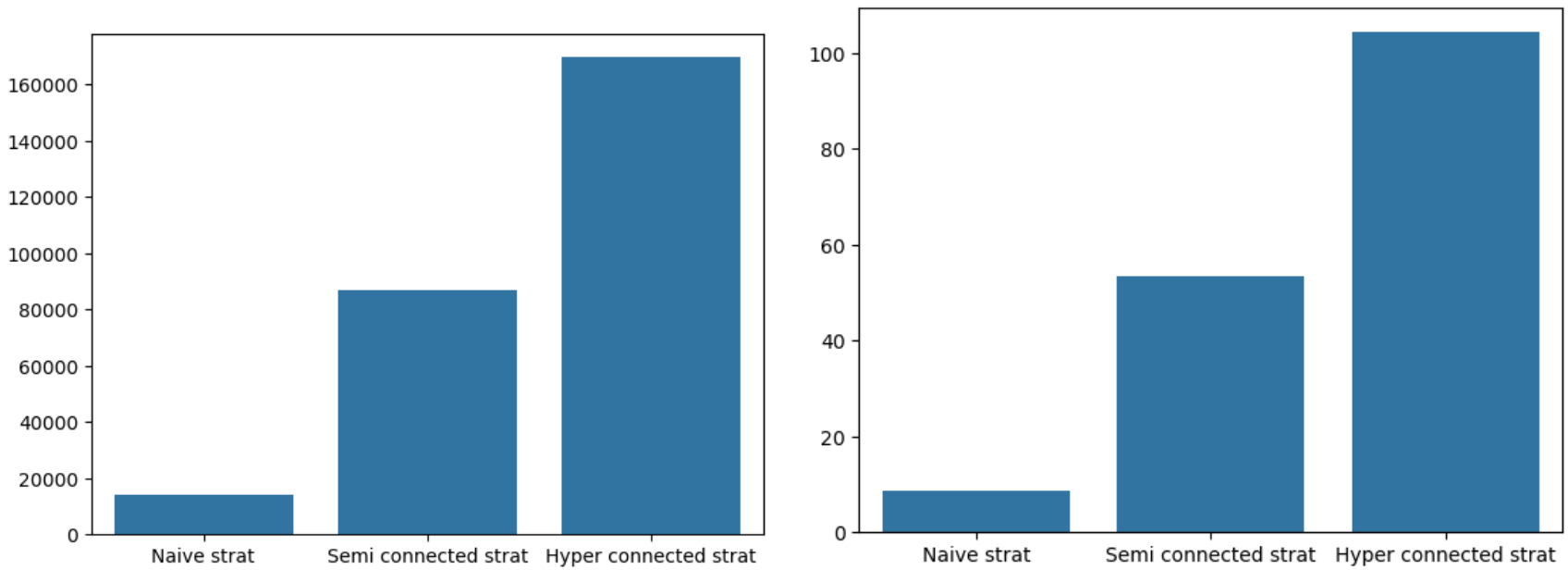


Figure – Number of edges and average node degree depending on the strategy

Explanation on graph density

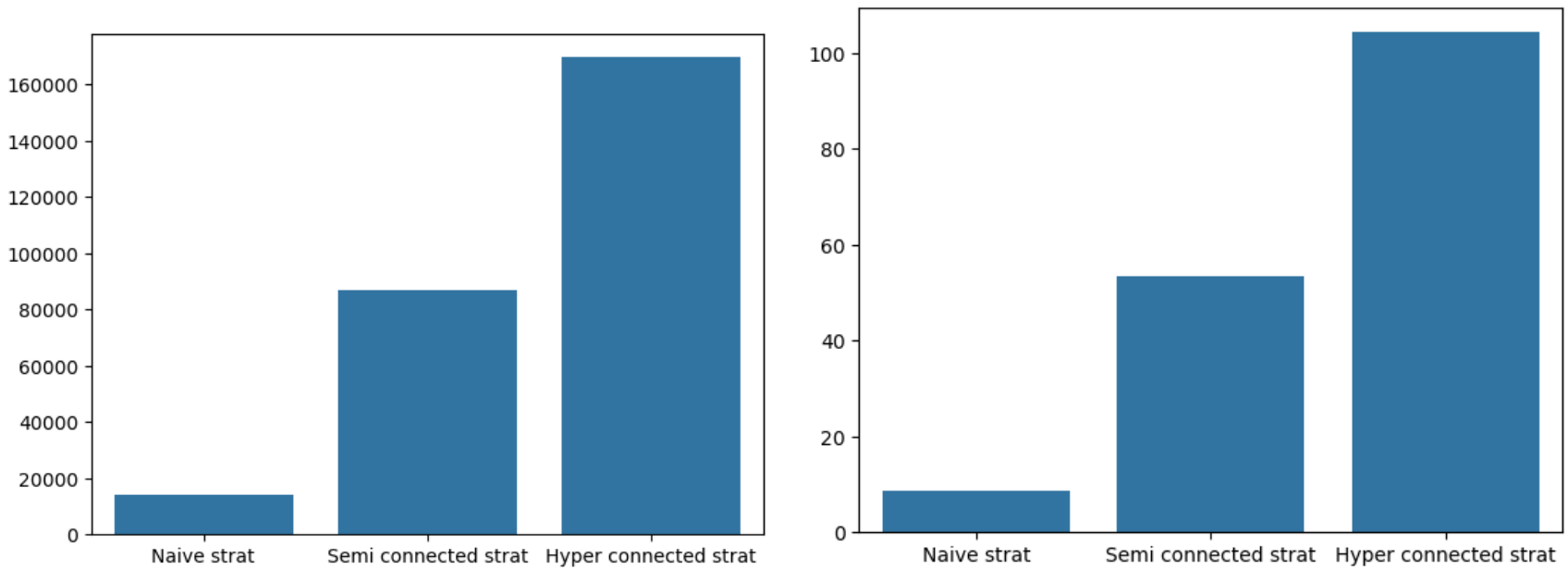


Figure – Number of edges and average node degree depending on the strategy

New question : Do we gain relevant information the more edges there are ?

Comparing strategies with metrics

Now, we're going to use three nodes metrics to compare these strategies.

Comparing strategies with metrics

Now, we're going to use three nodes metrics to compare these strategies.

Definition of shortest path

The shortest path between two nodes is the minimum number of edges used to go from one node to the other

Comparing strategies with metrics - Betweenness centrality

Definition of betweenness centrality

The betweenness centrality of a node is the number of shortest path that passes through the given node. (Measure the importance of a node)

Comparing strategies with metrics - Betweenness centrality

Definition of betweenness centrality

The betweenness centrality of a node is the number of shortest path that passes through the given node. (Measure the importance of a node)

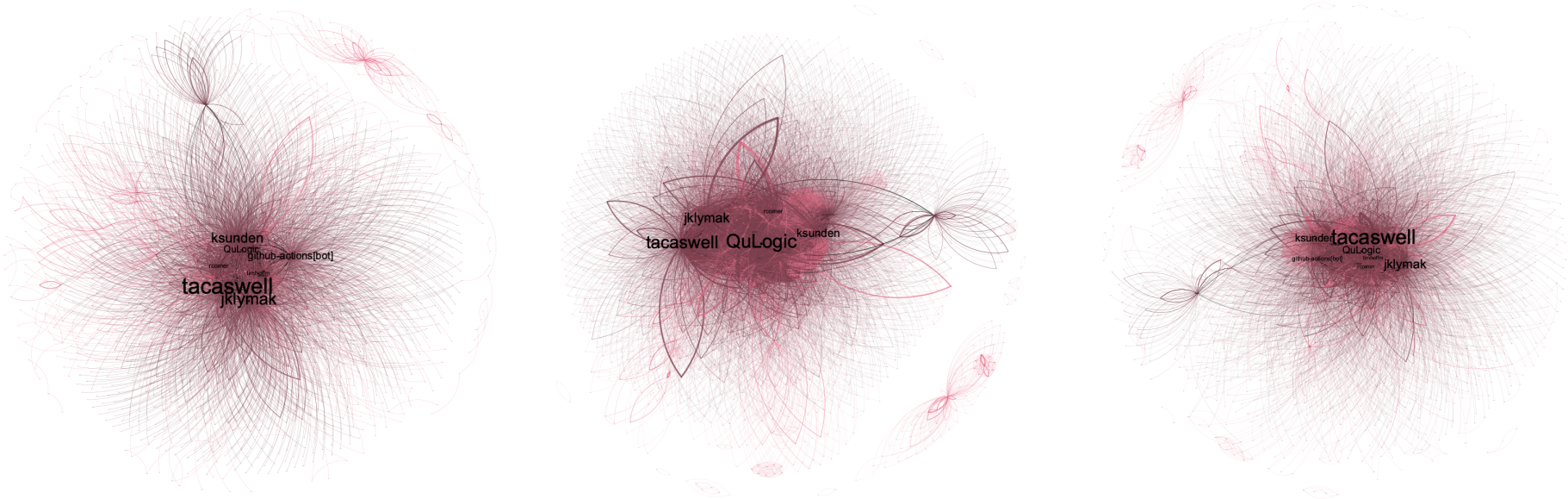


Figure – Naive, hyper-connected and semi-connected strategy

Comparing strategies with metrics - Closeness centrality

Definition of closeness centrality

The closeness centrality of a node is the reciprocal of the sum of all the shortest path between the node and all the other. (Measure how close a node is to all the other it's connected to)

Comparing strategies with metrics - Closeness centrality

Definition of closeness centrality

The closeness centrality of a node is the reciprocal of the sum of all the shortest path between the node and all the other. (Measure how close a node is to all the other it's connected to)

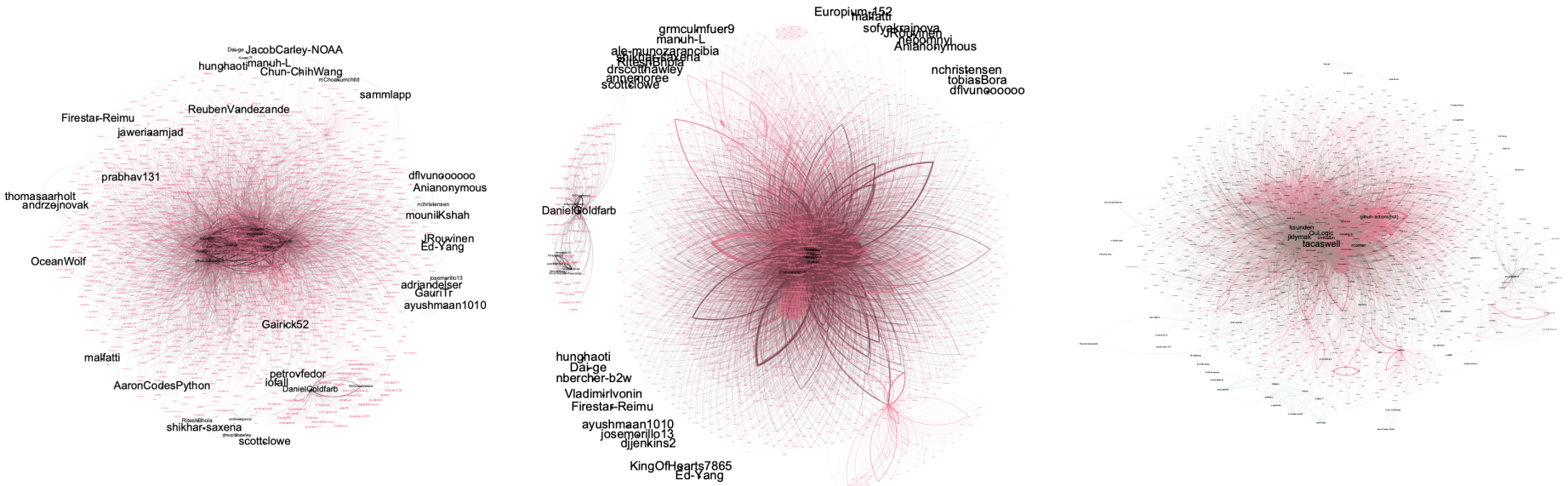


Figure – Naive, hyper-connected and semi-connected strategy

Comparing strategies with metrics - Closeness centrality

Problem : Nodes that are connected to only one person have also a maximum closeness centrality. So we only kept nodes that have at least contributed 3 times with different people or on different event.

Comparing strategies with metrics - Closeness centrality

Problem : Nodes that are connected to only one person have also a maximum closeness centrality. So we only kept nodes that have at least contributed 3 times with different people or on different event.

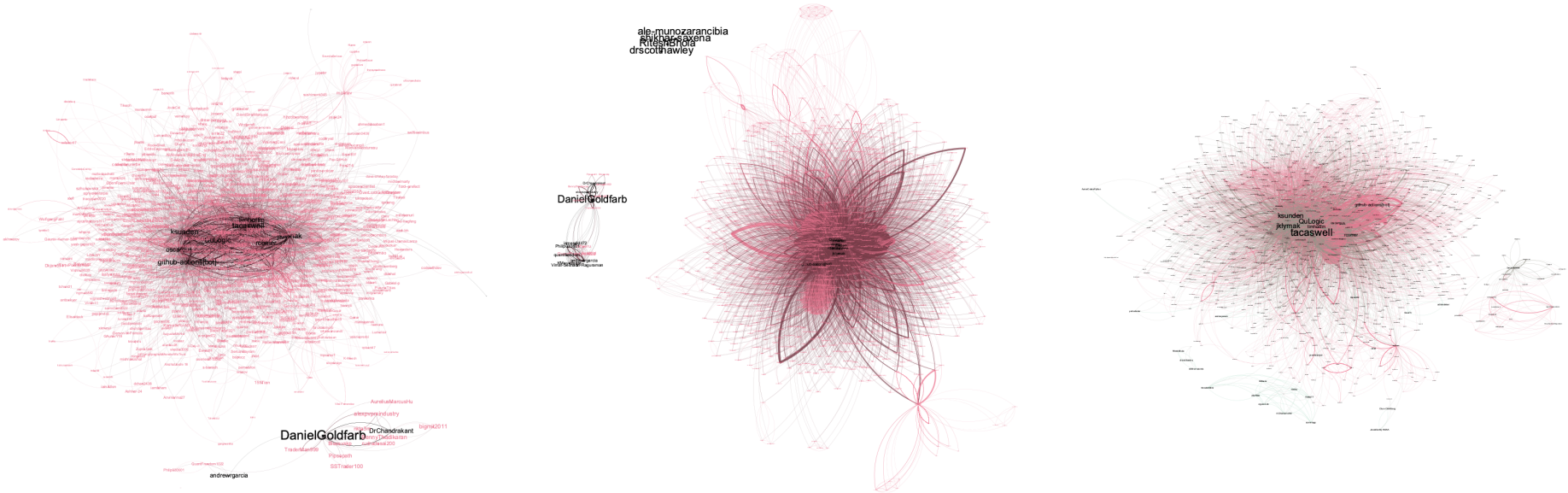


Figure – Naive, hyper-connected and semi-connected strategy

Comparing strategies with metrics - In degree

Definition of in degree

The in degree of a node is the number of edges that are directed to the node. (Measure how much other contributors contribute with the given contribuor)

Comparing strategies with metrics - In degree

Definition of in degree

The in degree of a node is the number of edges that are directed to the node. (Measure how much other contributors contribute with the given contribuor)

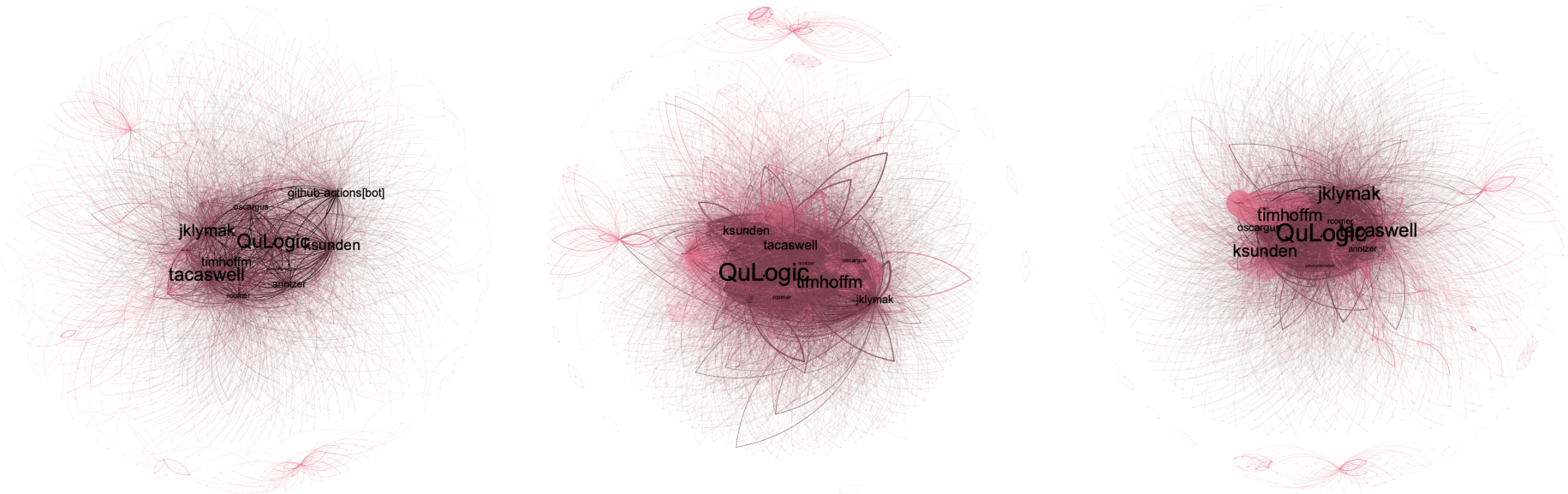


Figure – Naive, hyper-connected and semi-connected strategy

Comparing strategies with metrics - Out degree

Definition of out degree

The out degree of a node is the number of edges that leave the node. (Measure how much a contributor contribute with other)

Comparing strategies with metrics - Out degree

Definition of out degree

The out degree of a node is the number of edges that leave the node. (Measure how much a contributor contribute with other)

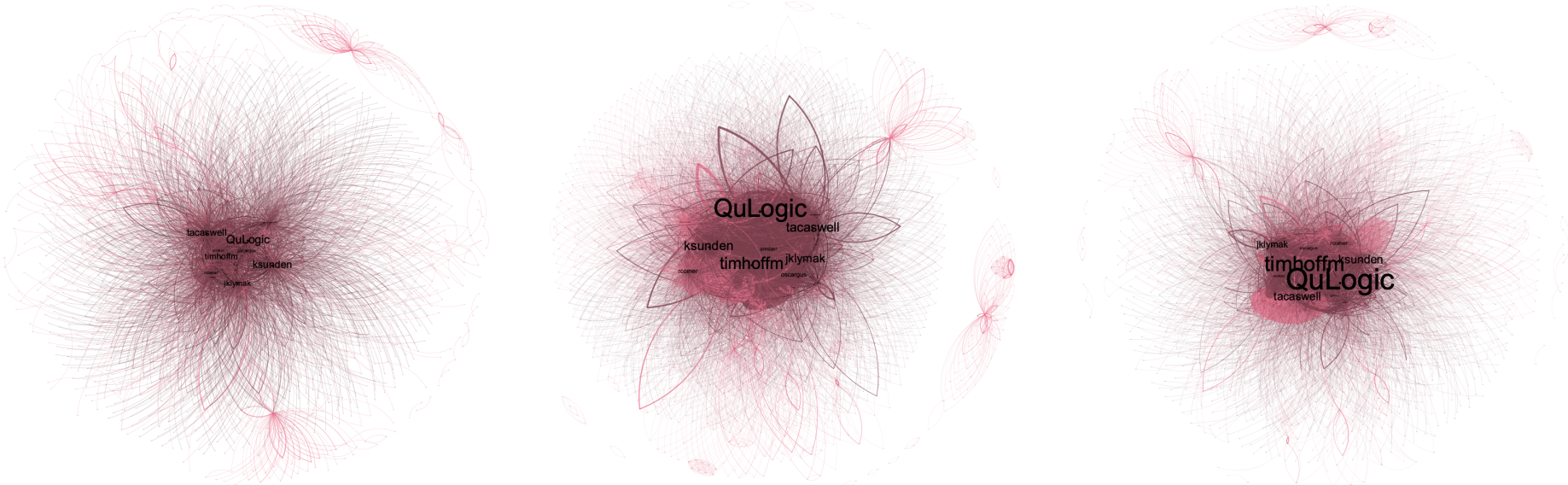


Figure – Naive, hyper-connected and semi-connected strategy

Conclusion of the comparaison

- In term of node metrics, the strategies aren't so different

Conclusion of the comparaison

- In term of node metrics, the strategies aren't so different
- The first strategy makes more readable graph
- The first strategy is less heavy in space and memory due to less edge

Conclusion of the comparaison

- In term of node metrics, the strategies aren't so different
- The first strategy makes more readable graph
- The first strategy is less heavy in space and memory due to less edge

So we're going to prefer the first strategy compared to the others.

Introduction

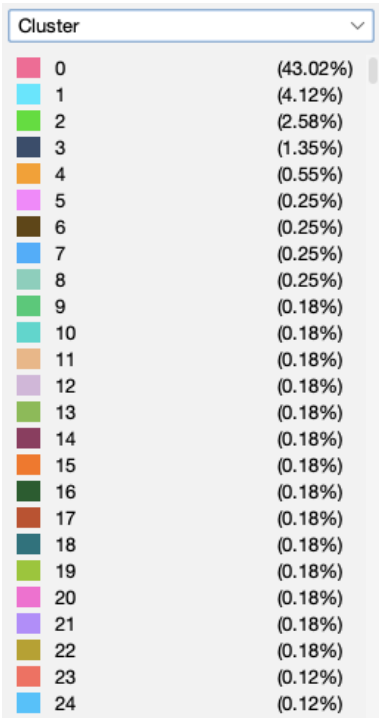
One of the remaining question is what does the Leiden algorithme (a community detection algorithm) mark as cluster.

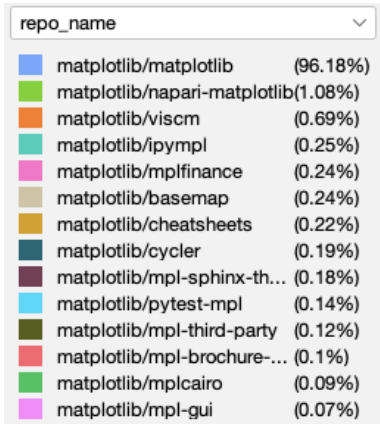
Introduction

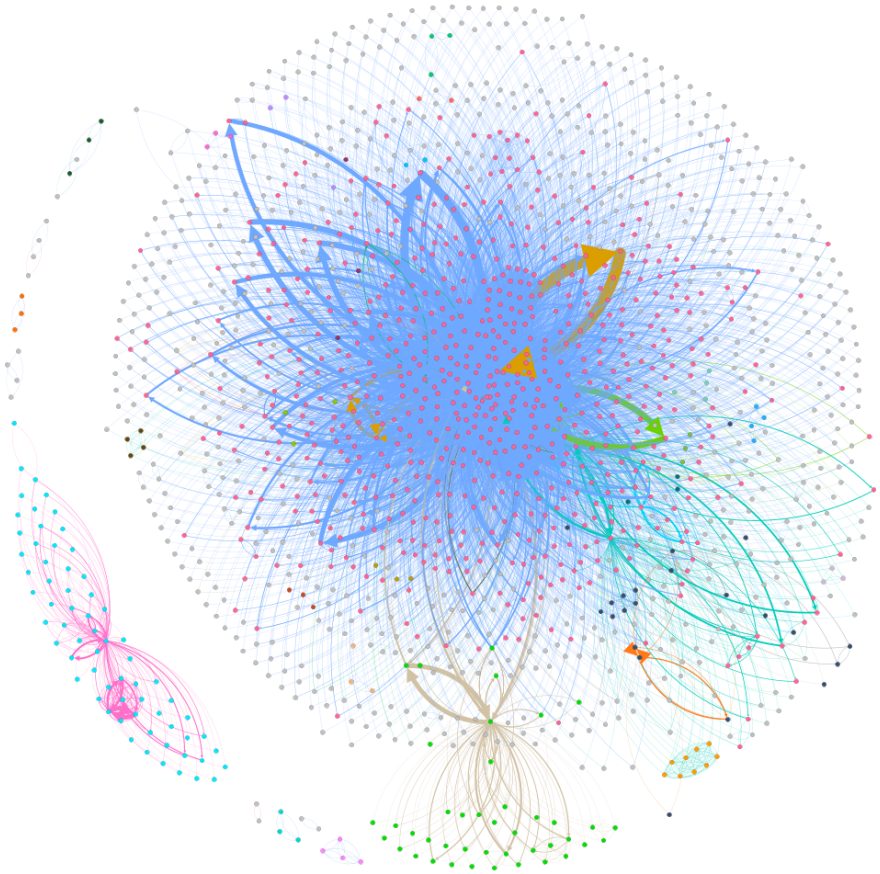
One of the remaining question is what does the Leiden algorithme (a community detection algorithm) mark as cluster. To answer this, I analysed :

- The matplotlib hyper-connected graph
- The graph containing all the events generated with the naive strategy

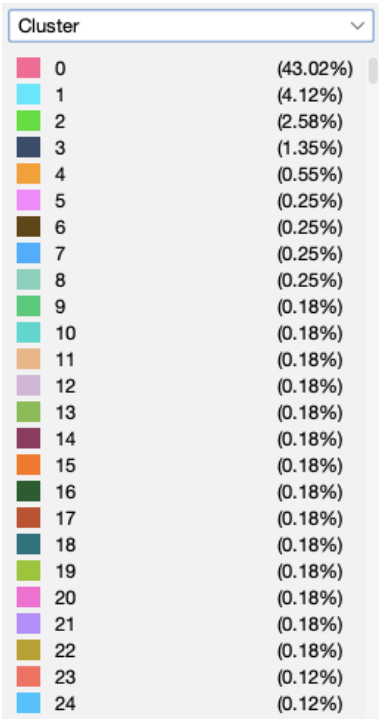
Matplotlib graph

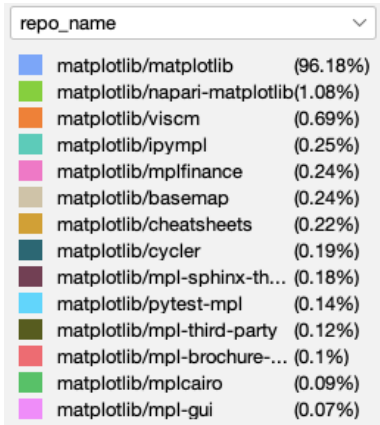


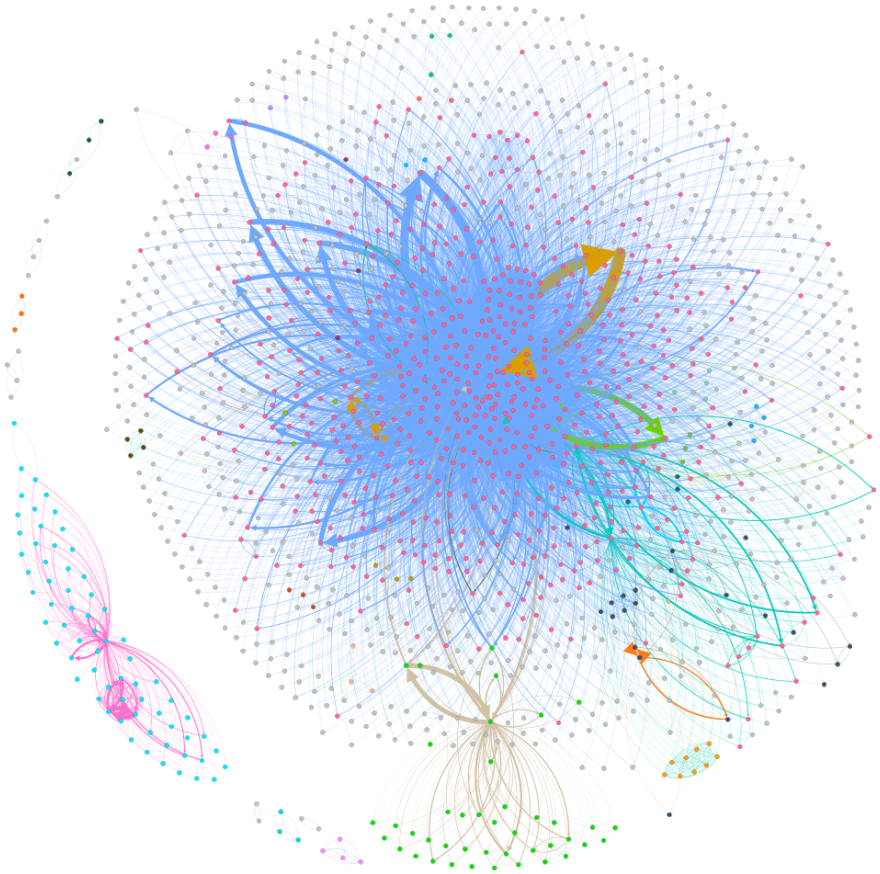




Matplotlib graph







Intuition : cluster = repository

Methodology

Methodology : generating the graph for a given cluster and a repository and side by side

Methodology

Methodology : generating the graph for a given cluster and a repository and side by side

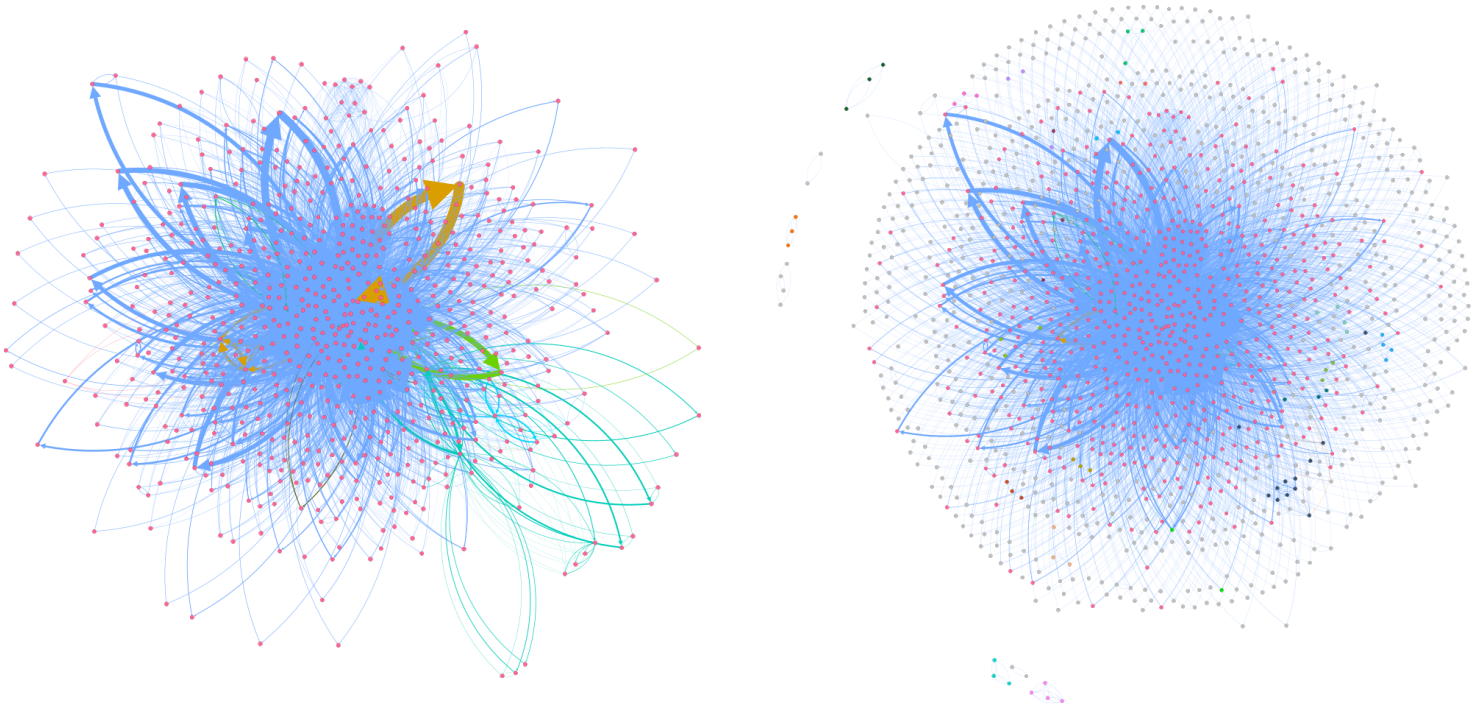


Figure – Cluster 0 and Principal repo

Methodology

Methodology : generating the graph for a given cluster and a repository and side by side

Figure – Cluster 1 and matplotlib/mplfinance repo

Methodology

Methodology : generating the graph for a given cluster and a repository and side by side

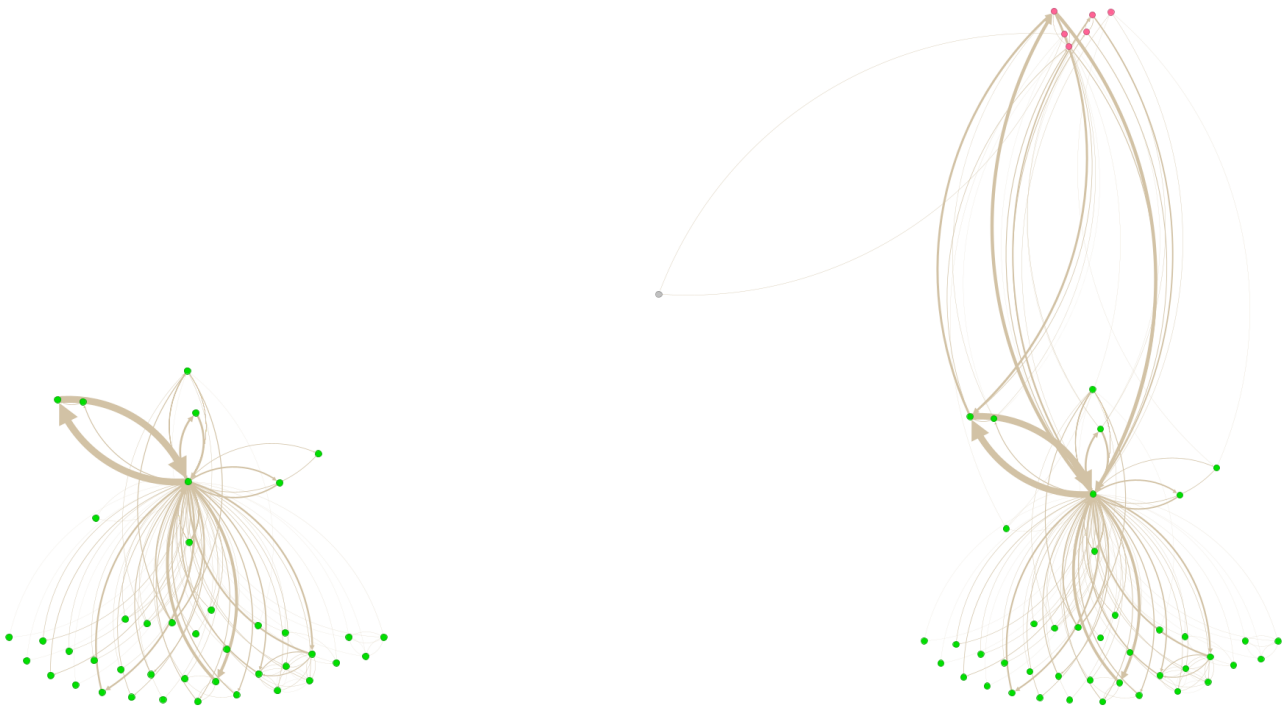


Figure – Cluster 2 and matplotlib/basemap

Methodology

Methodology : generating the graph for a given cluster and a repository and side by side

Figure – Cluster 4 and matplotlib/ipympl

Observations

The last graph confirms that cluster != repository.

Observations

The last graph confirms that cluster != repository.

I thought that the other contributors weren't detected because of either :

- other contributors contributed a lot to other repo

Observations

The last graph confirms that cluster != repository.

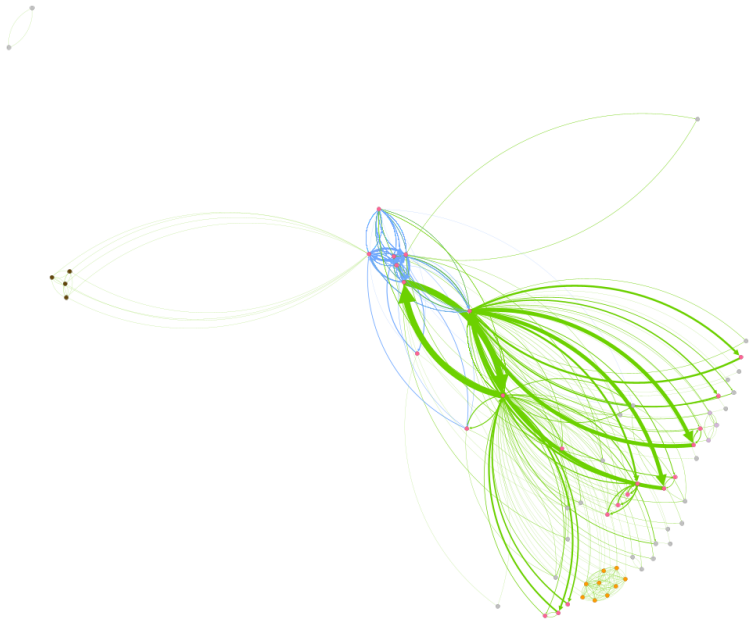
I thought that the other contributors weren't detected because of either :

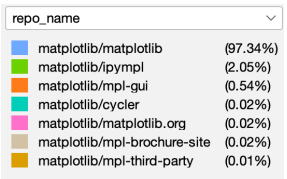
- other contributors contributed a lot to other repo
- this cluster = same event

Observations

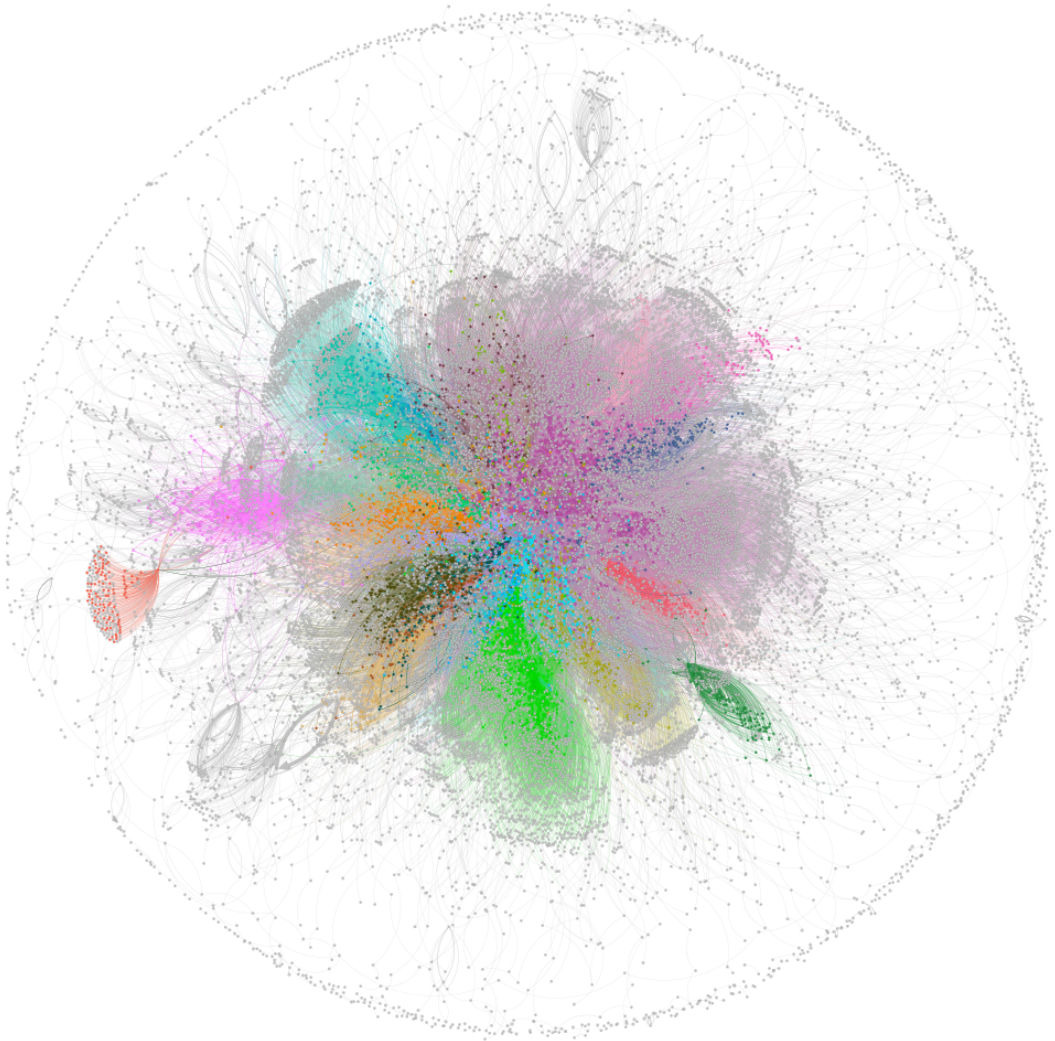
The last graph confirms that cluster != repository.
I thought that the other contributors weren't detected because of either :

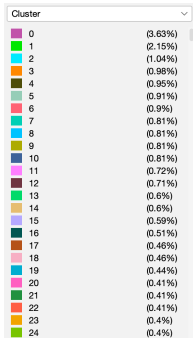
- other contributors contributed a lot to other repo
- this cluster = same event





Entire graph



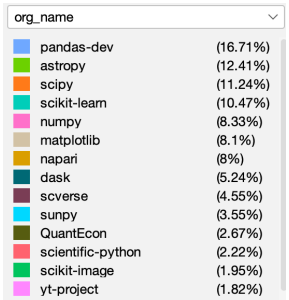


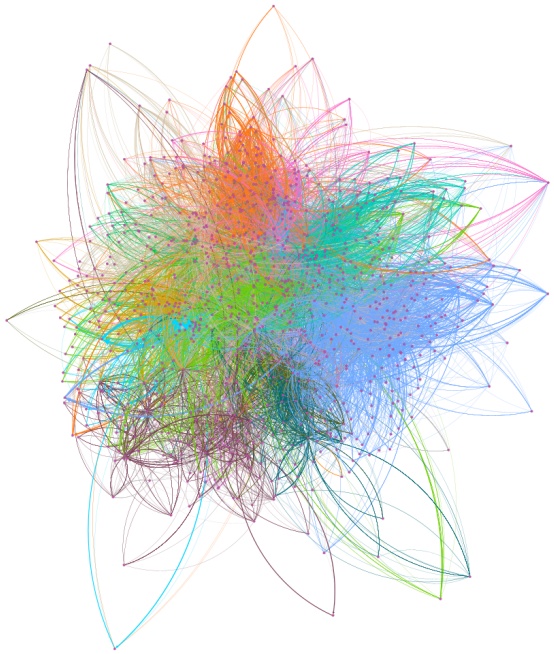
Methodology

For some cluster, to keep an ideea of the location on the entire graph, I generated two images :

- One that keeps the layout
- One where I re-run the layout to see the interactions

Results for top cluster





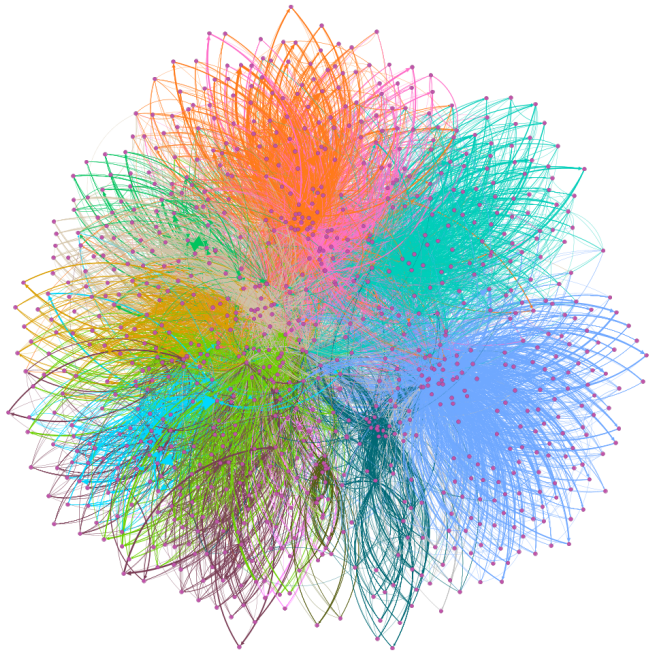


Figure – Cluster 0, original layout, re-run layout

Results for top cluster

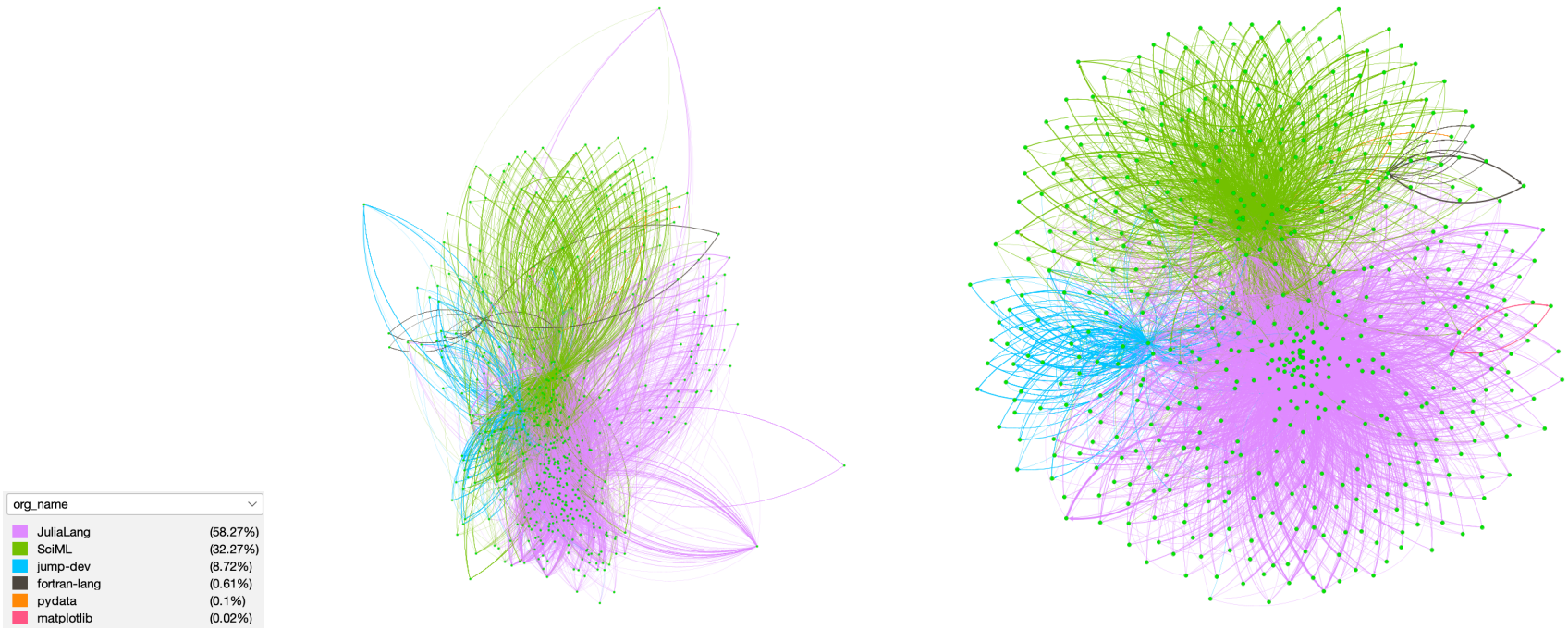


Figure – Cluster 1, original layout, re-run layout

Results for top cluster

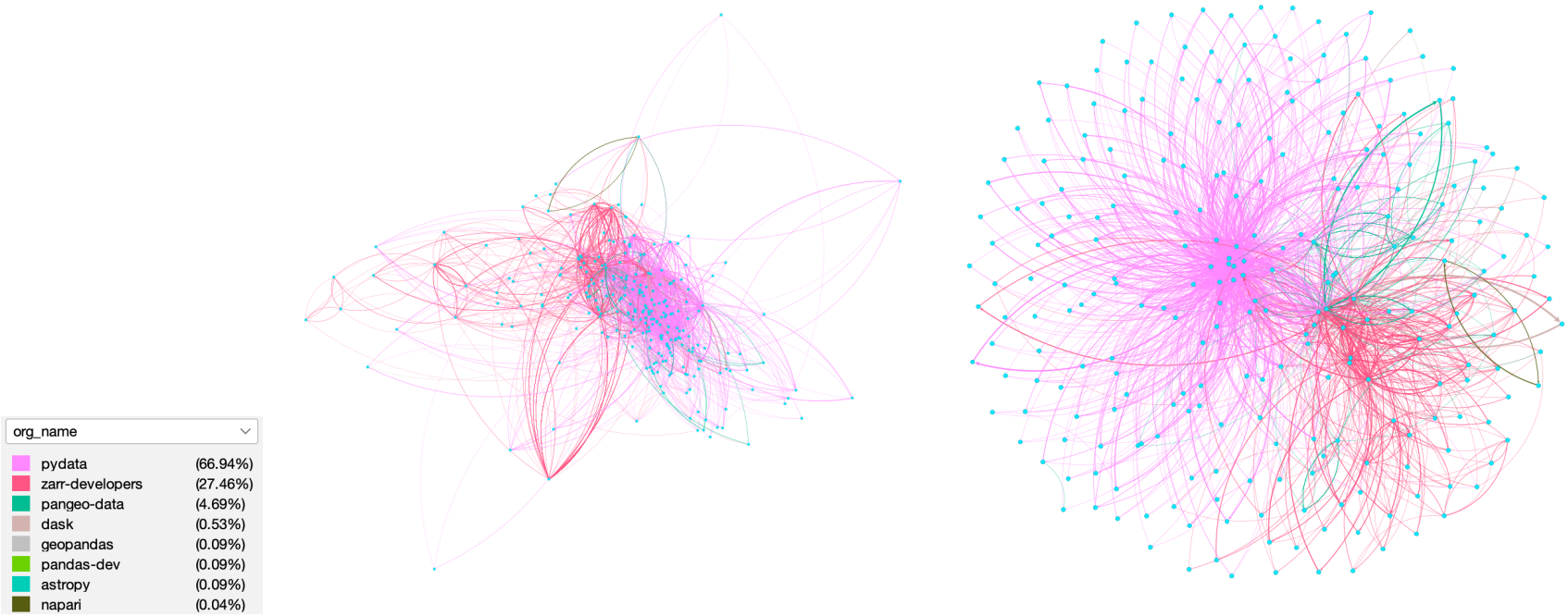


Figure – Cluster 2, original layout, re-run layout

Results for most of the clusters

As most of the other cluster only contains edges of one given organisation, we apply the previous methodology : displaying side by side a graph of the cluster and then a graph with only the organisation

Results for most of the clusters

As most of the other cluster only contains edges of one given organisation, we apply the previous methodology : displaying side by side a graph of the cluster and then a graph with only the organisation

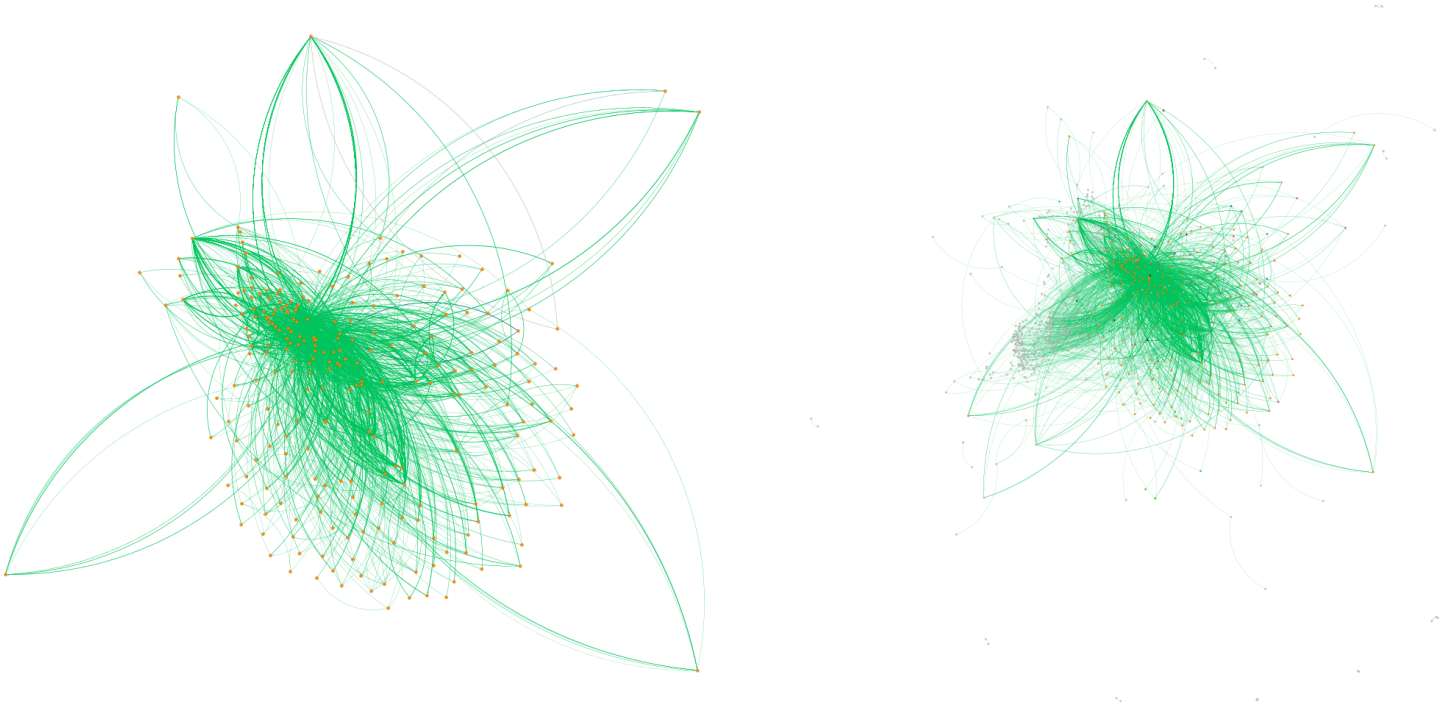


Figure – Cluster 3 and sympy

Results for most of the clusters

As most of the other cluster only contains edges of one given organisation, we apply the previous methodology : displaying side by side a graph of the cluster and then a graph with only the organisation

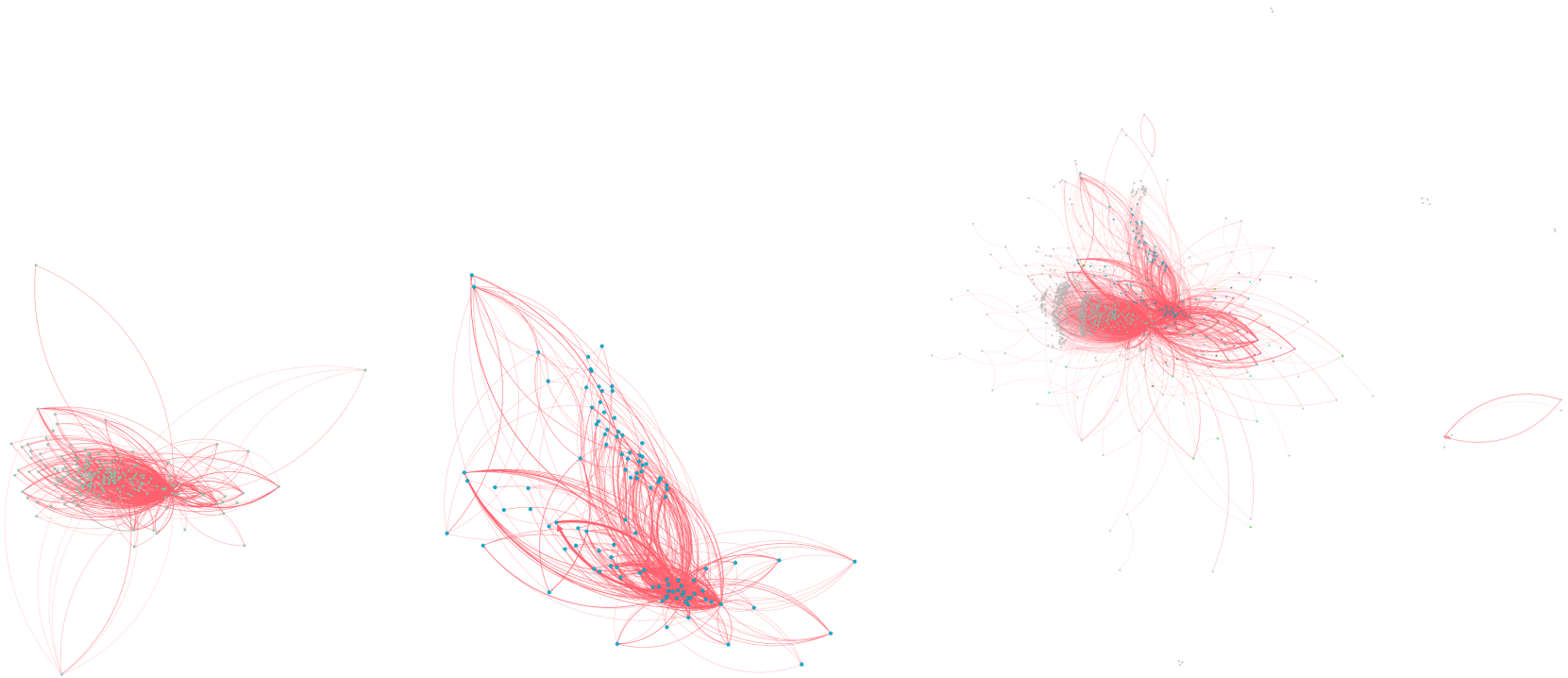


Figure – Cluster 5 and 19 and OSGeo

Results for most of the clusters

As most of the other cluster only contains edges of one given organisation, we apply the previous methodology : displaying side by side a graph of the cluster and then a graph with only the organisation

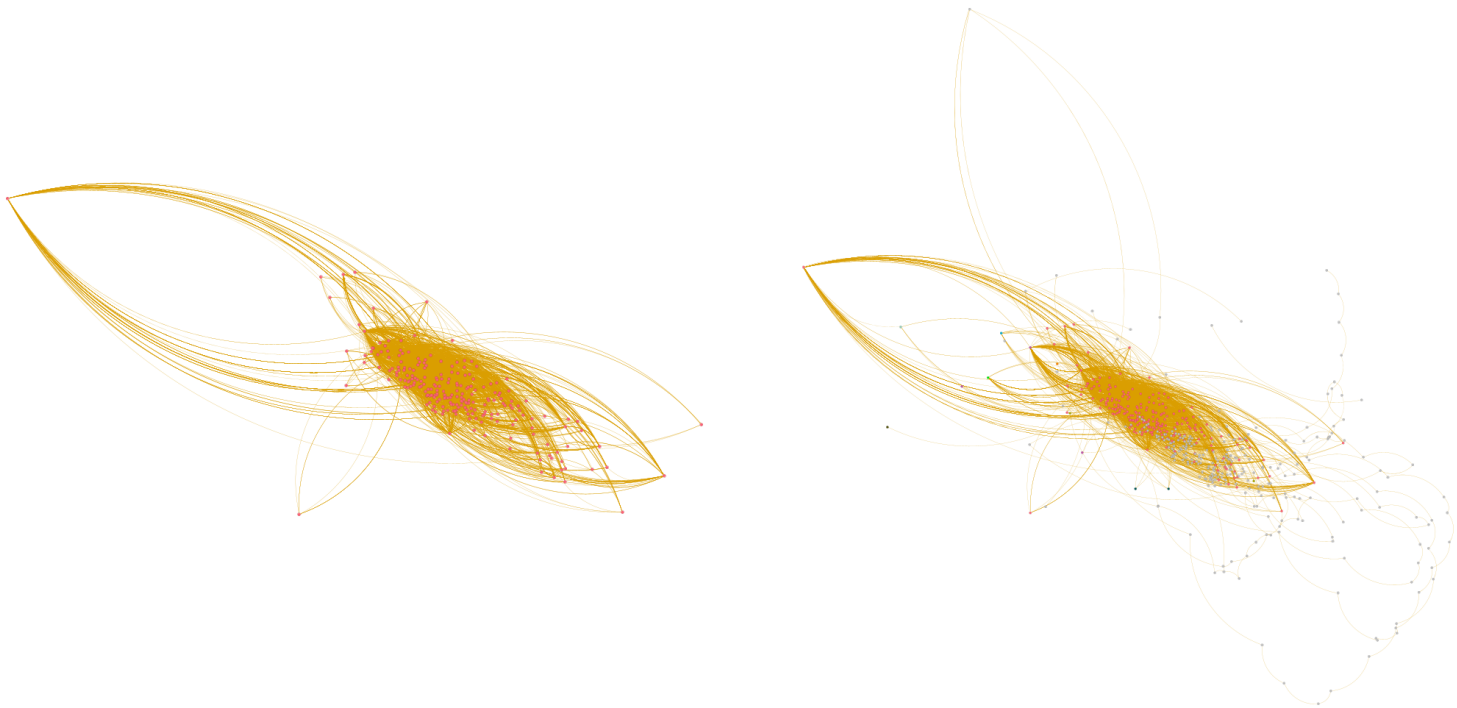


Figure – Cluster 6 and Galaxy Project

Results for most of the clusters

As most of the other cluster only contains edges of one given organisation, we apply the previous methodology : displaying side by side a graph of the cluster and then a graph with only the organisation

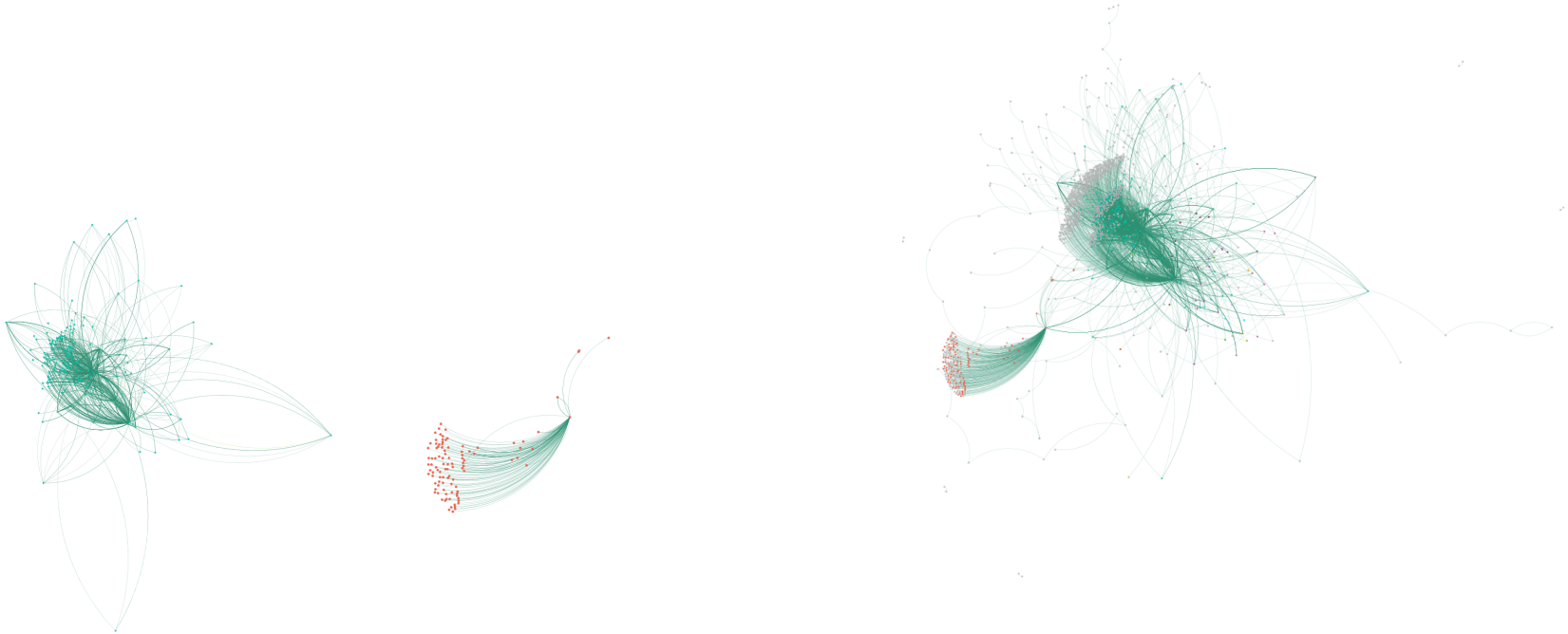


Figure – Cluster 7 and 22 and Spyder-ide

Results for most of the clusters

As most of the other cluster only contains edges of one given organisation, we apply the previous methodology : displaying side by side a graph of the cluster and then a graph with only the organisation

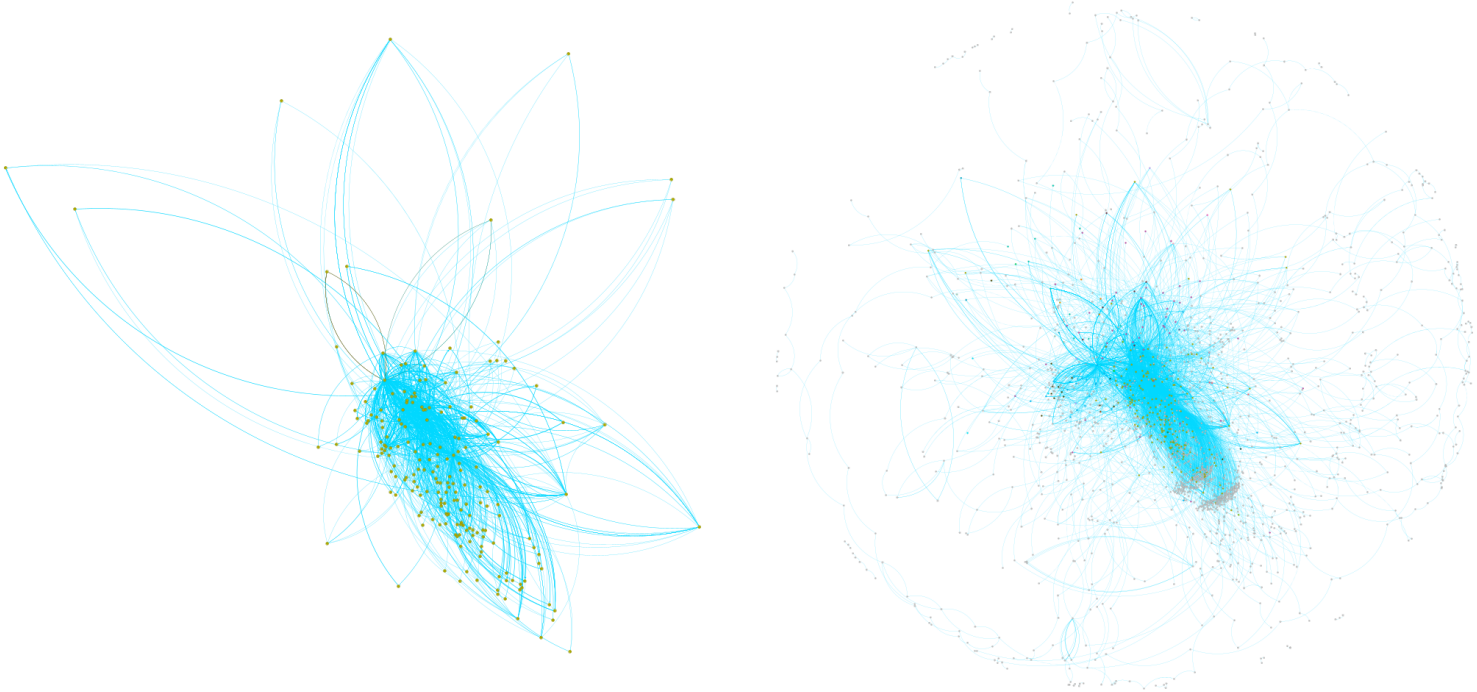


Figure – Cluster 9 and Conda

Results for most of the clusters

As most of the other cluster only contains edges of one given organisation, we apply the previous methodology : displaying side by side a graph of the cluster and then a graph with only the organisation

Figure – Cluster 9 and Matplotlib

Conclusion

The analyse of cluster gave some insights :

1. The cluster 0 gathers the most central contributors

Conclusion

The analyse of cluster gave some insights :

1. The cluster 0 gathers the most central contributors
2. Cluster can be useful to get relations between organisations and identify contributors that links organisation

Conclusion

The analyse of cluster gave some insights :

1. The cluster 0 gathers the most central contributors
2. Cluster can be useful to get relations between organisations and identify contributors that links organisation
3. Cluster can identify community in an organisation or repository (can be wrong)

Conclusion

The analyse of cluster gave some insights :

1. The cluster 0 gathers the most central contributors
2. Cluster can be useful to get relations between organisations and identify contributors that links organisation
3. Cluster can identify community in an organisation or repository (can be wrong)

However there is some remaining questions :

1. Are community in an organisation related to repository ?

Conclusion

The analyse of cluster gave some insights :

1. The cluster 0 gathers the most central contributors
2. Cluster can be useful to get relations between organisations and identify contributors that links organisation
3. Cluster can identify community in an organisation or repository (can be wrong)

However there is some remaining questions :

1. Are community in an organisation related to repository ?
2. How many clusters do we need to analyse to get enough information ?

Conclusion

The analyse of cluster gave some insights :

1. The cluster 0 gathers the most central contributors
2. Cluster can be useful to get relations between organisations and identify contributors that links organisation
3. Cluster can identify community in an organisation or repository (can be wrong)

However there is some remaining questions :

1. Are community in an organisation related to repository ?
2. How many clusters do we need to analyse to get enough information ?
3. Is the study of cluster useful to detect the collaboration between organisation ?

Remaining questions

1. If we merge all the directed edges going between two given contributors into a single directed edges, will it affect the results/metrics ?

Remaining questions

1. If we merge all the directed edges going between two given contributors into a single directed edges, will it affect the results/metrics ?
2. How does the graph evolve if we generate one graph per month ? Are there disappearance of important contributor ? Does time modify the link between organisations ?

Remaining questions

1. If we merge all the directed edges going between two given contributors into a single directed edges, will it affect the results/metrics ?
2. How does the graph evolve if we generate one graph per month ? Are there disappearance of important contributor ? Does time modify the link between organisations ?
3. What information could we gain if we generate a graph where the nodes are the events and the edges are the contributors ?

Remaining questions

1. If we merge all the directed edges going between two given contributors into a single directed edges, will it affect the results/metrics ?
2. How does the graph evolve if we generate one graph per month ? Are there disappearance of important contributor ? Does time modify the link between organisations ?
3. What information could we gain if we generate a graph where the nodes are the events and the edges are the contributors ?
4. Does the contributor type influence the metrics ? Can we note a difference visually or recognize a bot just by looking at the graph ?

Remaining questions

1. If we merge all the directed edges going between two given contributors into a single directed edges, will it affect the results/metrics ?
2. How does the graph evolve if we generate one graph per month ? Are there disappearance of important contributor ? Does time modify the link between organisations ?
3. What information could we gain if we generate a graph where the nodes are the events and the edges are the contributors ?
4. Does the contributor type influence the metrics ? Can we note a difference visually or recognize a bot just by looking at the graph ?
5. If we only keep the most active repository of each organisation, will it affect the metric/graph ?

Conclusion

Thanks for listening