Yearly Measurement of Boardex Graphs

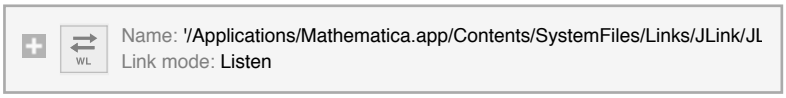
Report 1 for ID 2

In this version we use the largest connected subgraph

by dara@lossofgenerality.com
Jan 9th 2017

Default memory size does not suffice for Mathematica, use these functions to increase the memory to 32768 MB or over 32GIG to read and process one file :

In[1]:=
```
Needs["JLink`"]
ReinstallJava[JVMArguments → "-Xmx32768m"]
```

Out[2]=
```
LinkObject[
```


```
]
```

Read the 2008 file, Dara's Desktop:

In[3]:=
```
data2 =
  Import["/Users/darashayda1xfer/Desktop/MODULES/Ryerson 2016/BOZA RESEARCH/Task
     2008 NA - SMDEs Network - 2.xlsx"];
```

Read the 2008 file, log02 server:

```
data2 =
  Import["/var/www/uploader/uploads/Task 2008 NA - SMDEs Network - 226.xlsx"];
```

See what is inside data in terms of types:

In[4]:=
```
data2[[1]][[1]]
```

Out[4]=
```
{DirectorID*, Linked DirectorID*, Connected CompanyID*,
 Connected Company, Connected Company Type, Index, Sector,
 Date of overlap, Beginning of OverLap, End of OverLap,
 Overlapping Person's Role Title, ED/NED/SM, Individual's Role Title, ED/NED/SM}
```

Dimensions for the data structure

In[5]:=
```
Dimensions@data2
```
Out[5]= {1, 498 307, 14}

The actual data is accessed at fist index value 1:

In[6]:=
```
data = data2[[1]];
```

Connection type

```
data[[1006]][[5]]
Partnership
```

Build a graph that links “DirectorID*”⟶”Linked DirectorID*” :

In[7]:=
```
graph = Table[If[data[[i]][[5]] == "Quoted",
    data[[i]][[1]] -> data[[i]][[2]], Nothing], {i, 1 + 1, Length@data}];
```

Compute the disconnected subgraphs, clearly shows that there is a large super sub-graph and a few tiny disconnected graphs:

In[8]:=
```
graph2 = Table[If[data[[i]][[5]] == "Quoted",
    data[[i]][[1]] ⟞ data[[i]][[2]], Nothing], {i, 1 + 1, Length@data}];
components = ConnectedComponents[graph2];
Length@components
Length /@ components
```
Out[10]= 2

Out[11]= {72 678, 16}

Compute the largest connected subgraph

In[12]:=

```
superconnected = Subgraph[graph2, First@components];
ConnectedGraphQ[superconnected]
```

Out[13]= True

Count the vertices:

In[14]:=
```
VertexCount[superconnected]
```

Out[14]= 72 678

Number of edges

In[15]:=
```
EdgeCount[superconnected]
```

Out[15]= 318 000

Test and see if is a Simple Graph:

In[16]:=
```
SimpleGraphQ[superconnected]
```

Out[16]= False

Loop Free:

In[17]:=
```
LoopFreeGraphQ[superconnected]
```

Out[17]= True

It is not a Connected Graph, so it is comprised of multiple disjoint graphs:

In[18]:=
```
ConnectedGraphQ[superconnected]
```

Out[18]= True

it is not Planar Graph

In[19]:=
```
PlanarGraphQ[superconnected]
```

Out[19]= False

Maximum Vertex Degree

In[20]:=
```
Max@VertexDegree[superconnected]
```

Out[20]= 8044

Max In Degree

In[21]:=
```
Max@VertexInDegree[superconnected]
```
Out[21]=
```
8044
```

Max Out Degree

In[22]:=
```
Max@VertexOutDegree[superconnected]
```
Out[22]=
```
8044
```

FIXME: Mean Graph Distance

TBD: for these large graphs MeanGraphDistance requires hugest memory and running time.

```
N@MeanGraphDistance[graph]
```

```
$Aborted
```

I coded a brute force but highly parallelized version to find the stats on path length, after 12 hours on 64 cpu still did not finish:

```
n = VertexCount[graph];
sum = ConstantArray[0, n];
v = VertexList[graph];
tmp = 0;

CloseKernels[];
LaunchKernels[64];
d = ParallelTable[
    Table[sum[[j]] = sum[[j]] + If[i > j,
        If[(tmp = GraphDistance[graph, v[[i]], v[[j]], Method → "UnitWeight"]) == ∞,
         0, tmp], 0], {i, 1, n}];
    sum[[j]], {j, 1, n}];
CloseKernels[];
```

```
$Aborted
```

Random Sample Mean Graph Distance

In[23]:=
```
n = VertexCount[superconnected];
sum = ConstantArray[0, n];
v2 = VertexList[superconnected];


CloseKernels[];
LaunchKernels[4];

trials = 5000;

ints = Range@n;
d = ParallelTable[
    {i, j} = RandomChoice[v2, 2];
    GraphDistance[superconnected, i, j, Method → "UnitWeight"], {trials}
  ];
CloseKernels[];
```

Sampled Mean Graph Distance

In[32]:=
```
N@Mean@d
N@Variance@d
```

Out[32]= 4.2022

Out[33]= 0.62584

Mean Clustering Coefficient

How tightly clustered

In[34]:=
```
N@MeanClusteringCoefficient[superconnected]
```

Out[34]= 0.244719

In[35]:=
```
N@GlobalClusteringCoefficient[superconnected]
```

Out[35]= 0.00795814

Assortativity

For a graph with m edges and adjacency matrix entries $a_{i,j}$, the assortativity coefficient is given by $\left(\sum_{ij}\left(a_{i,j}-\frac{d_i\,d_j}{2\,m}\right)f_{i,j}\right)\Big/\left(\sum_{ij}\left(d_i\,\delta_{i,j}-\frac{d_i\,d_j}{2\,m}\right)f_{i,j}\right)$, where d_i is the out-degree for the vertex v_i and $\delta_{i,j}$ is 1 if there is an edge from v_i to v_j and 0 otherwise.

https://en.wikipedia.org/wiki/Assortativity

Correlation between vertices of different degree:

In[36]:=
```
N@GraphAssortativity[superconnected]
```

Out[36]=
```
-0.631756
```

We might also conclude that the graph of 2008 needs to be further broken down to relevant subgraphs which indicate some sort of community clustering features.

Communities

FindGraphCommunities finds communities with many edges joining vertices of the same community and comparatively few edges joining vertices of different communities.

Possible settings for the `Method` option include:

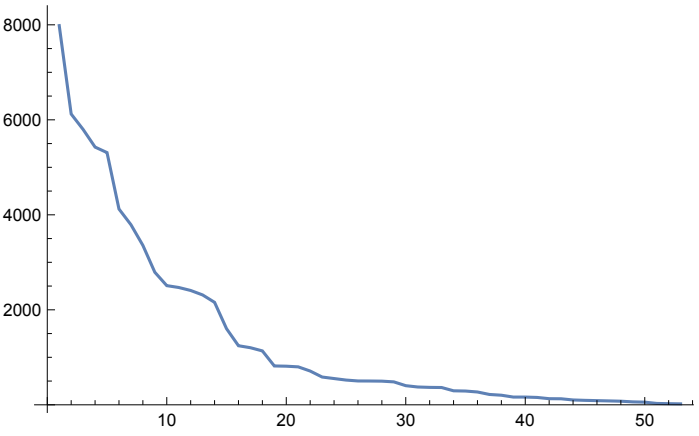
"Modularity"	modularity–based clustering
"Centrality"	centrality–based clustering
"CliquePercolation"	clique percolation–based clustering
"Hierarchical"	hierarchical–based clustering
"Spectral"	spectral–based clustering

In[37]:=
```
communities = FindGraphCommunities[superconnected];
```

In[38]:=
```
Length /@ communities
ListLinePlot[Length /@ communities]
```

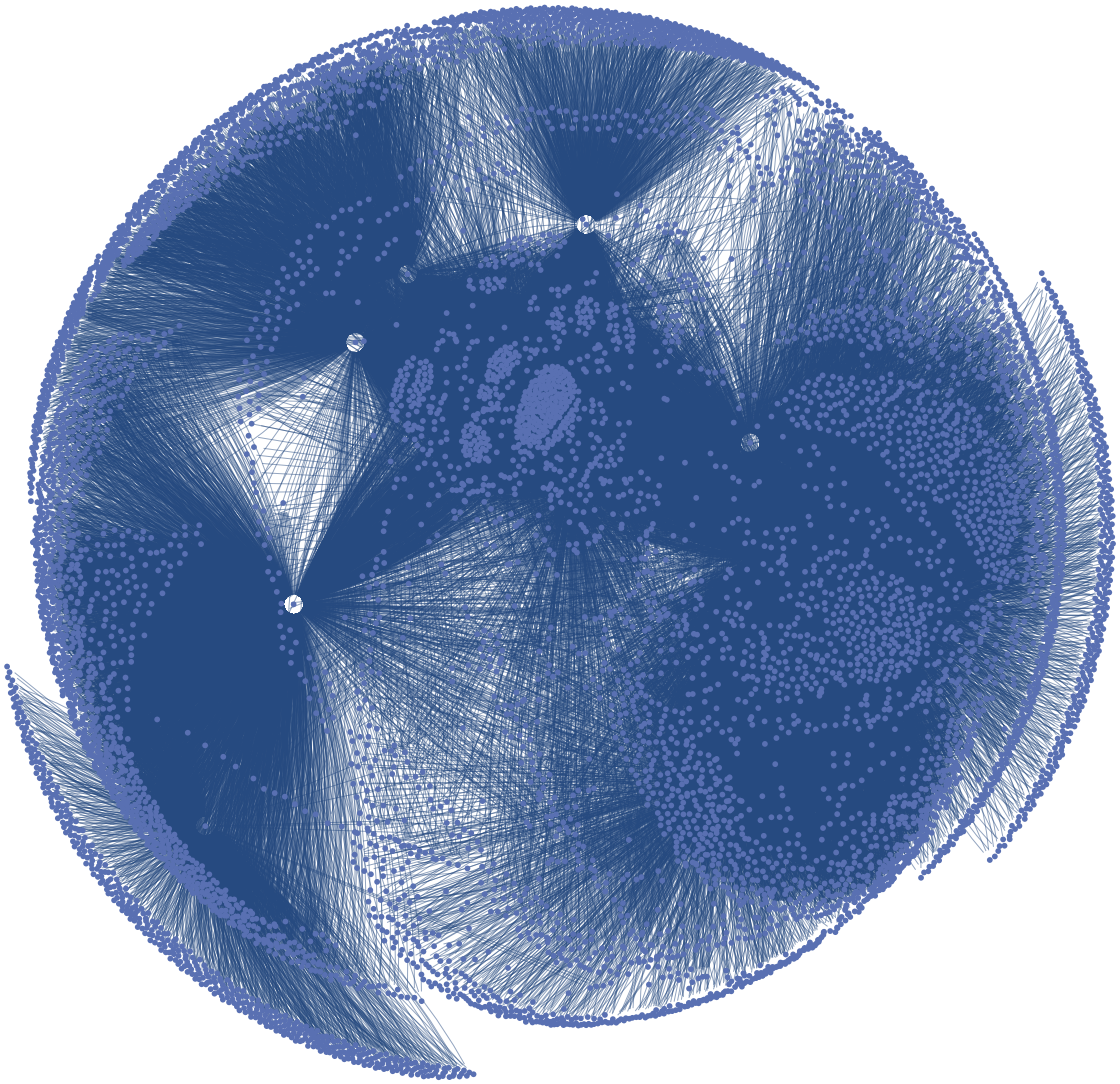
Out[38]=
```
{7983, 6122, 5797, 5426, 5310, 4122, 3792, 3357, 2791, 2509, 2470,
 2407, 2313, 2157, 1603, 1244, 1202, 1135, 818, 813, 799, 711, 585,
 553, 521, 502, 501, 498, 483, 403, 375, 367, 364, 295, 290, 270, 218,
 202, 163, 163, 155, 129, 127, 102, 94, 88, 82, 76, 62, 56, 29, 24, 20}
```

Out[39]=

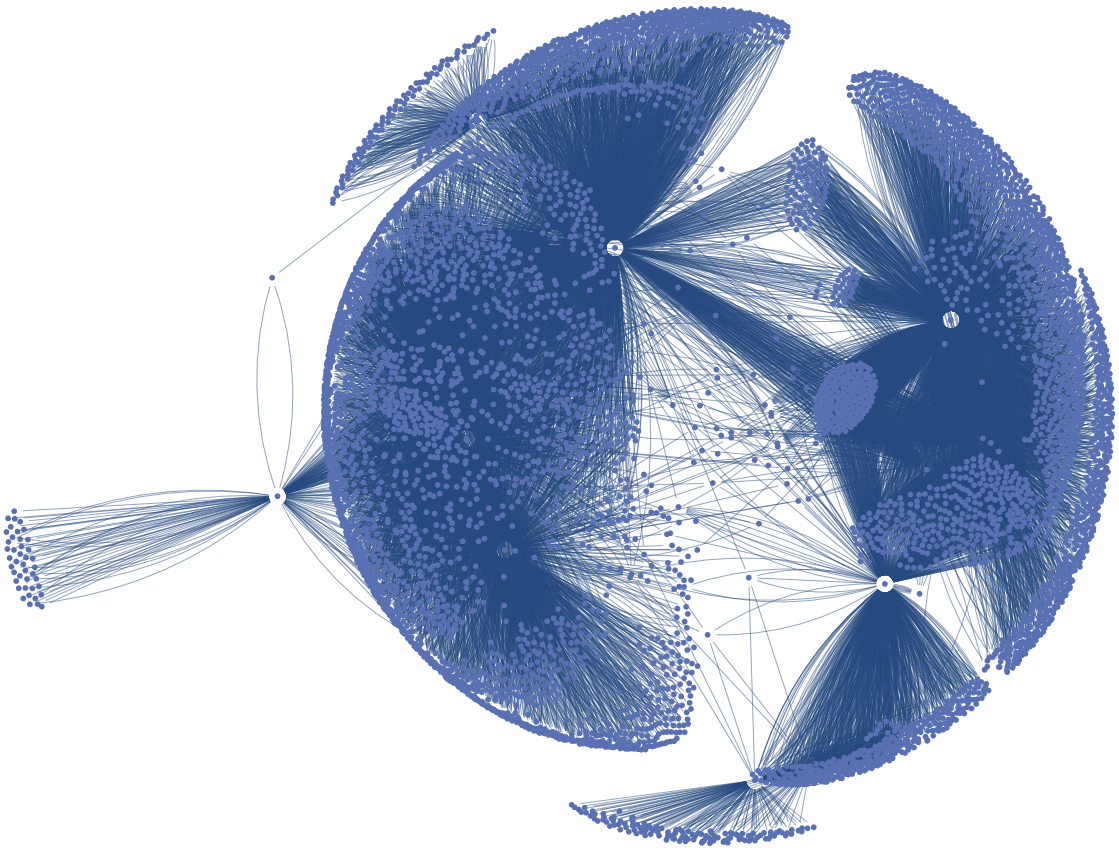


In[40]:=
```
Table[community = Subgraph[superconnected, communities[[i]]];
 Graph[community, ImageSize → 600],
 {i, 1, Length@communities}]
```

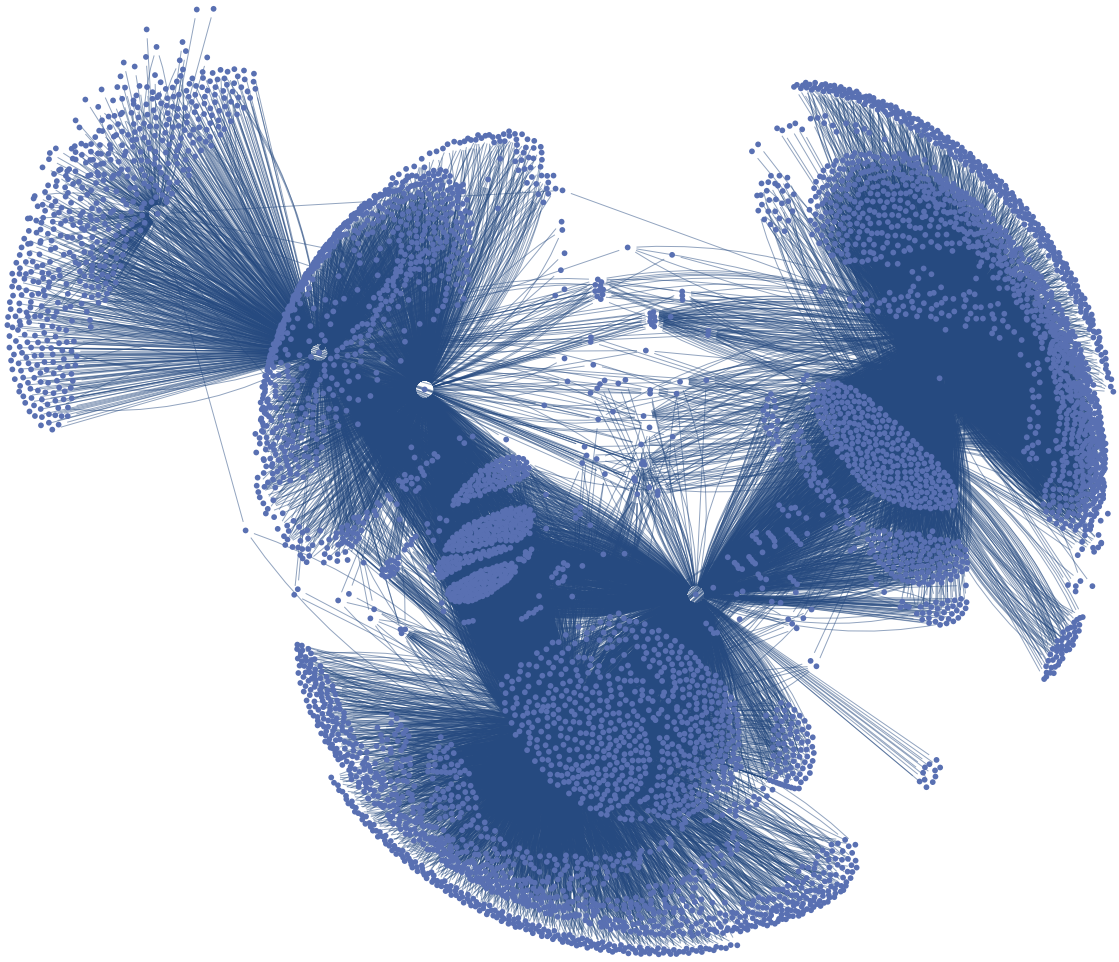
Out[40]= {



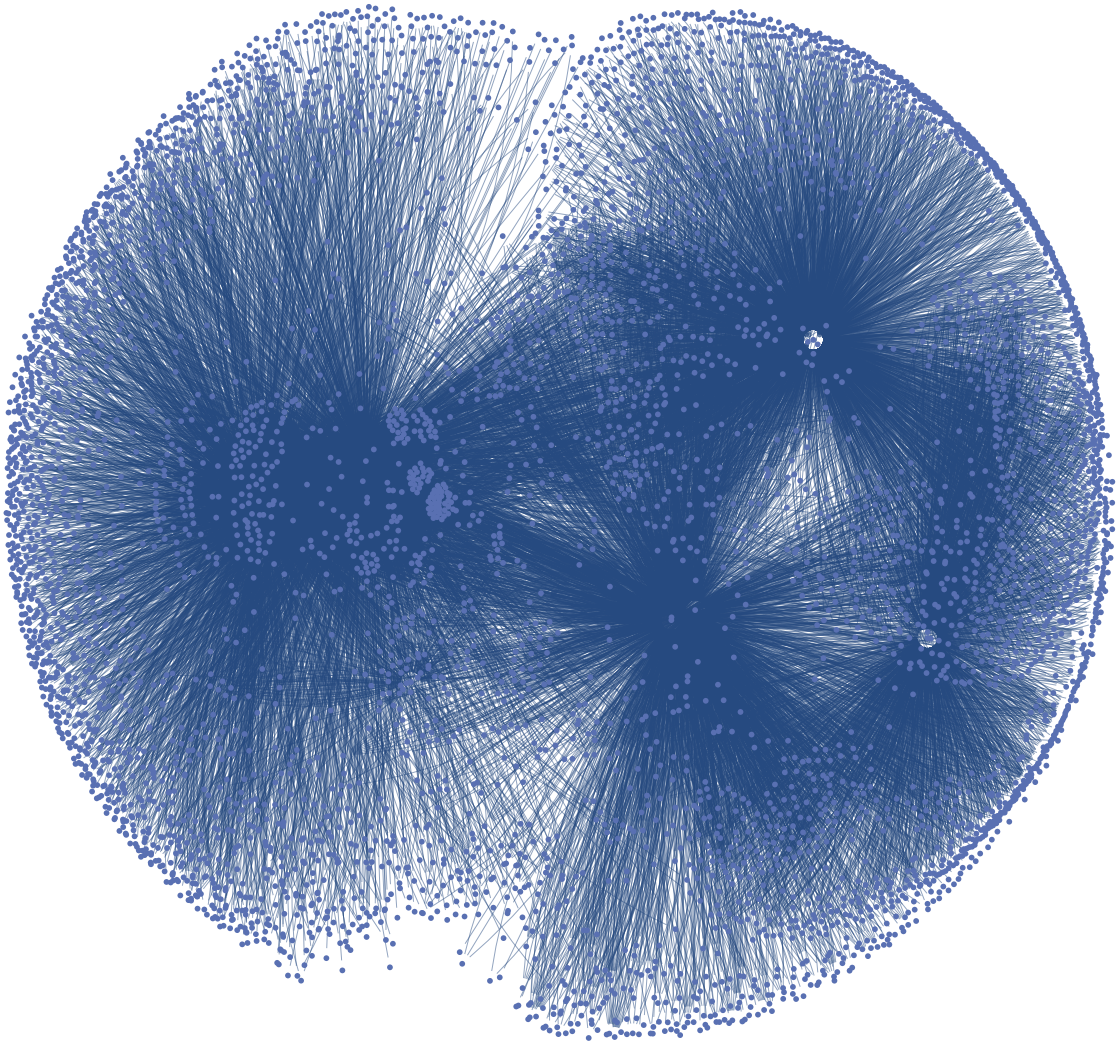
,



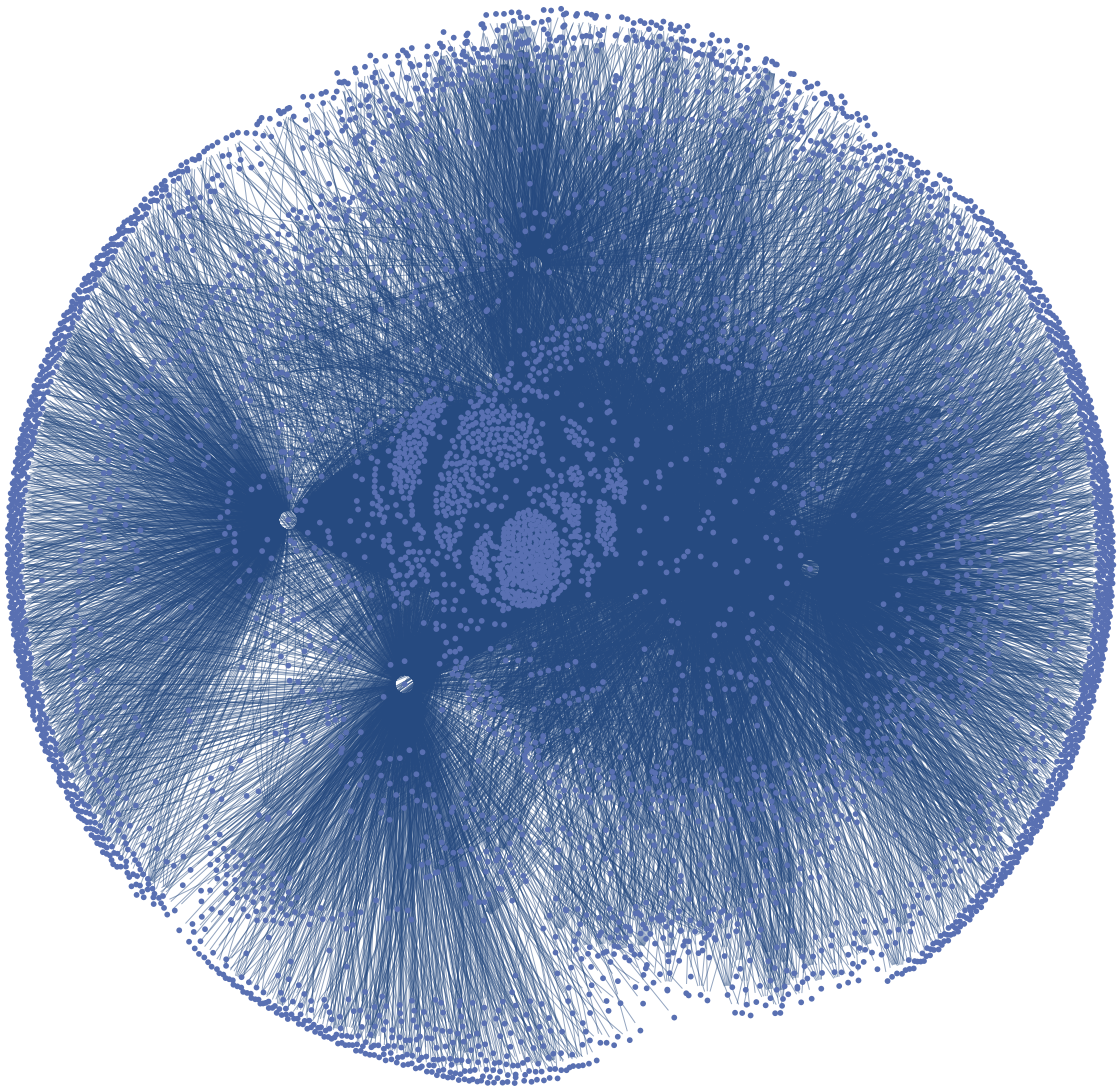
,



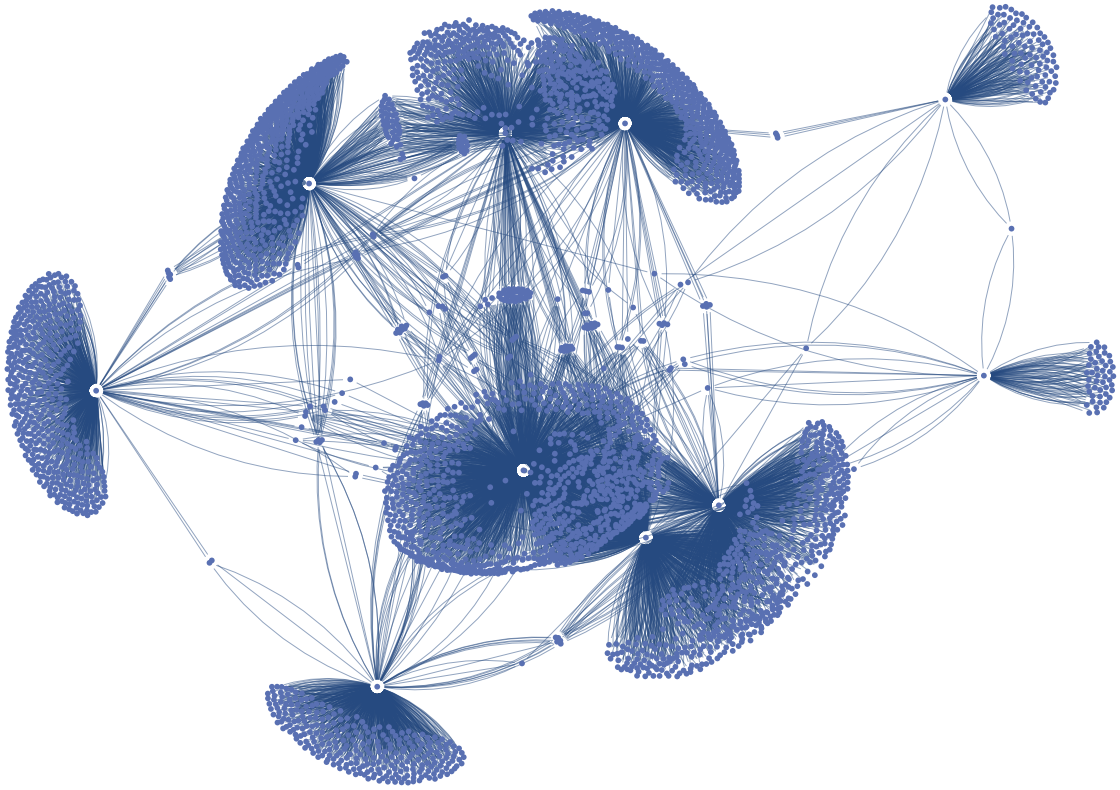
,

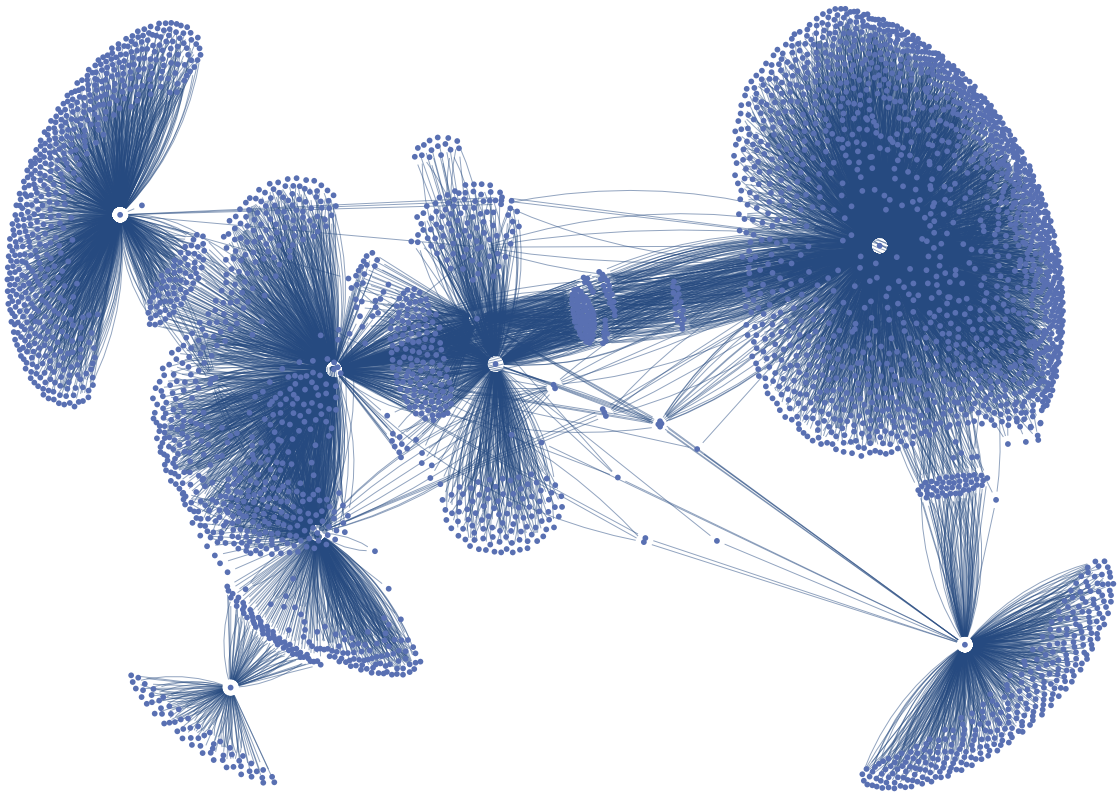


,

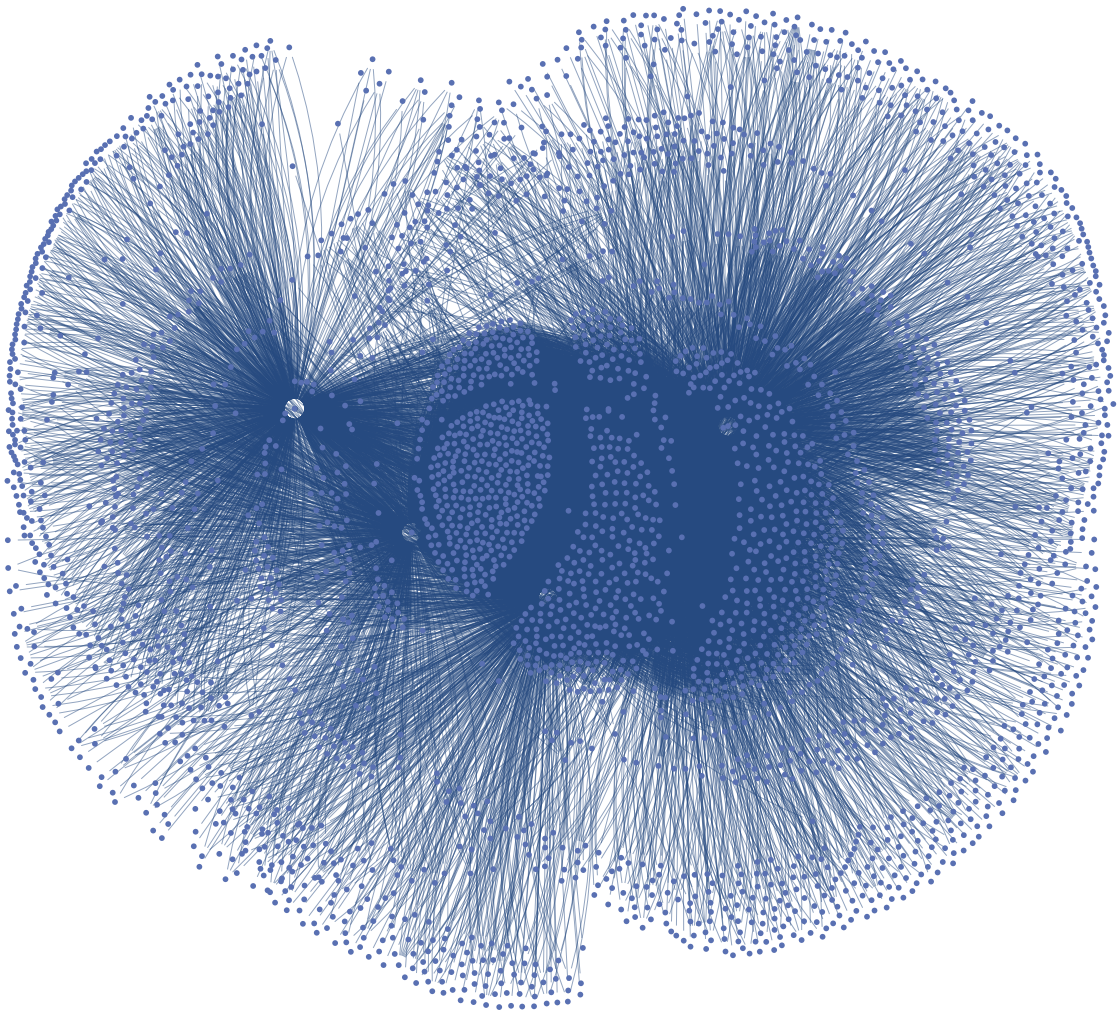


,

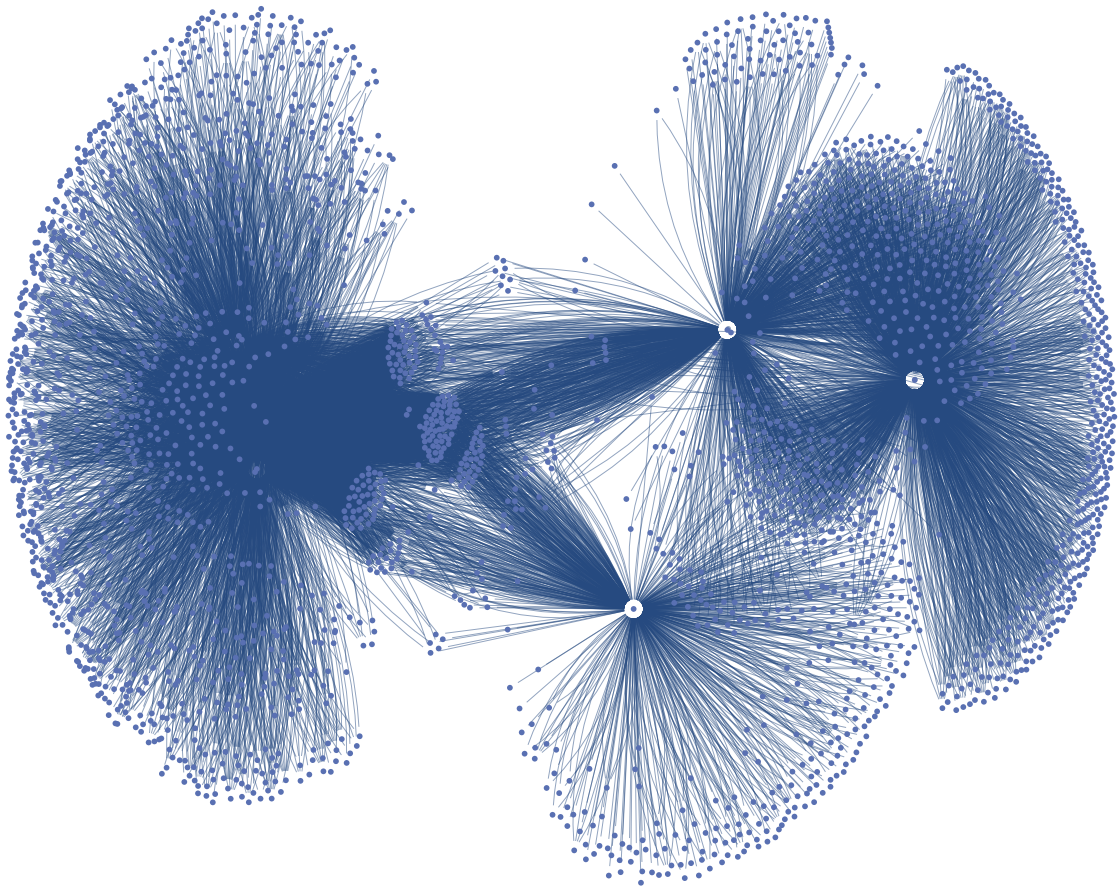

,



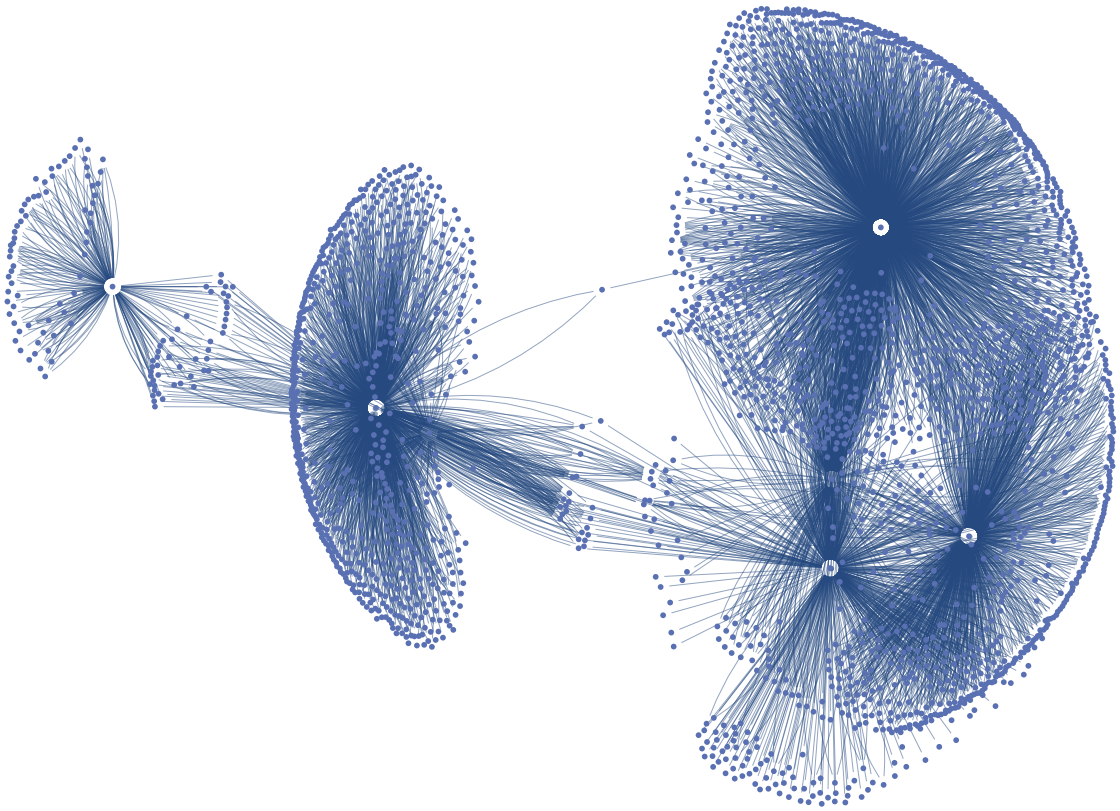
,



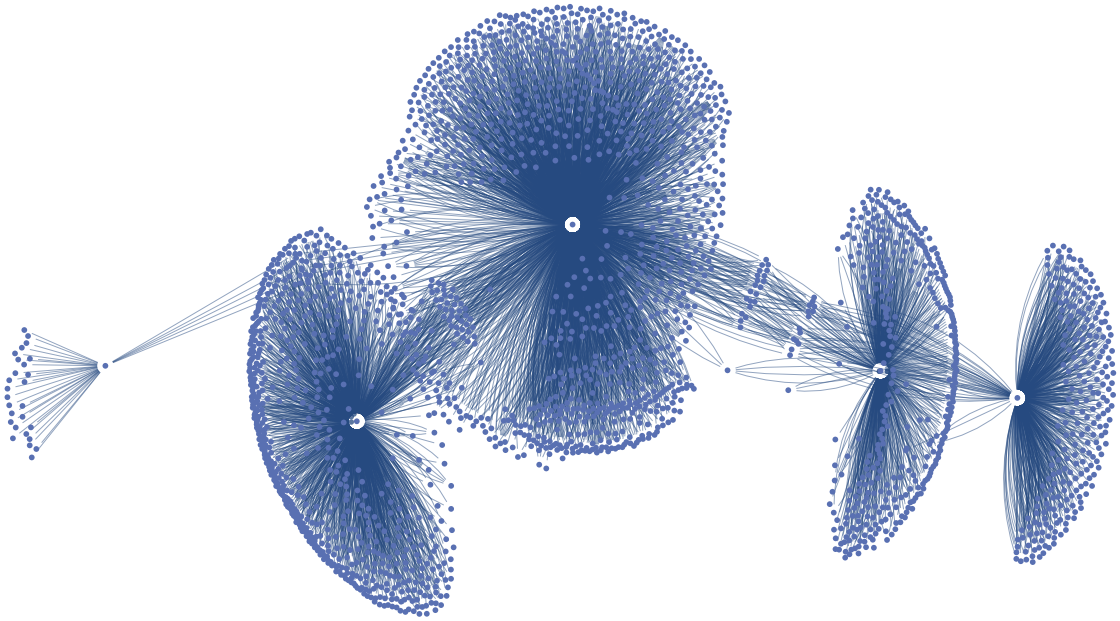
,



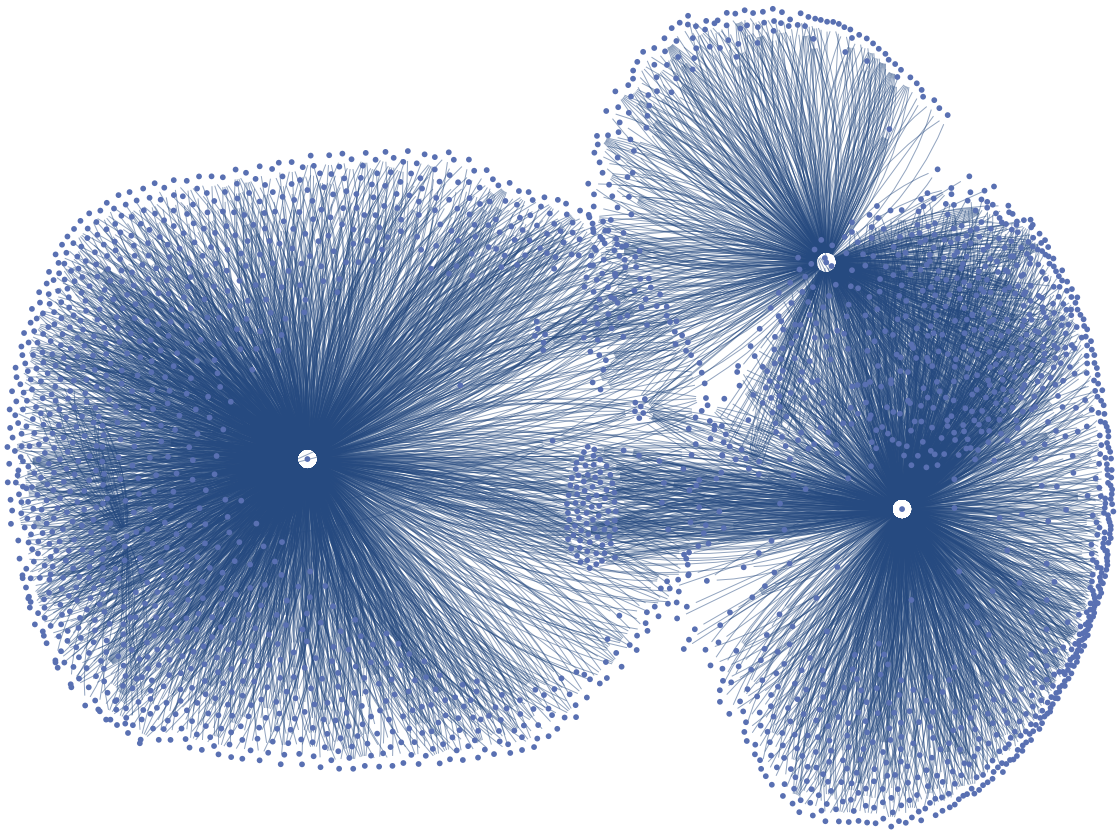
,



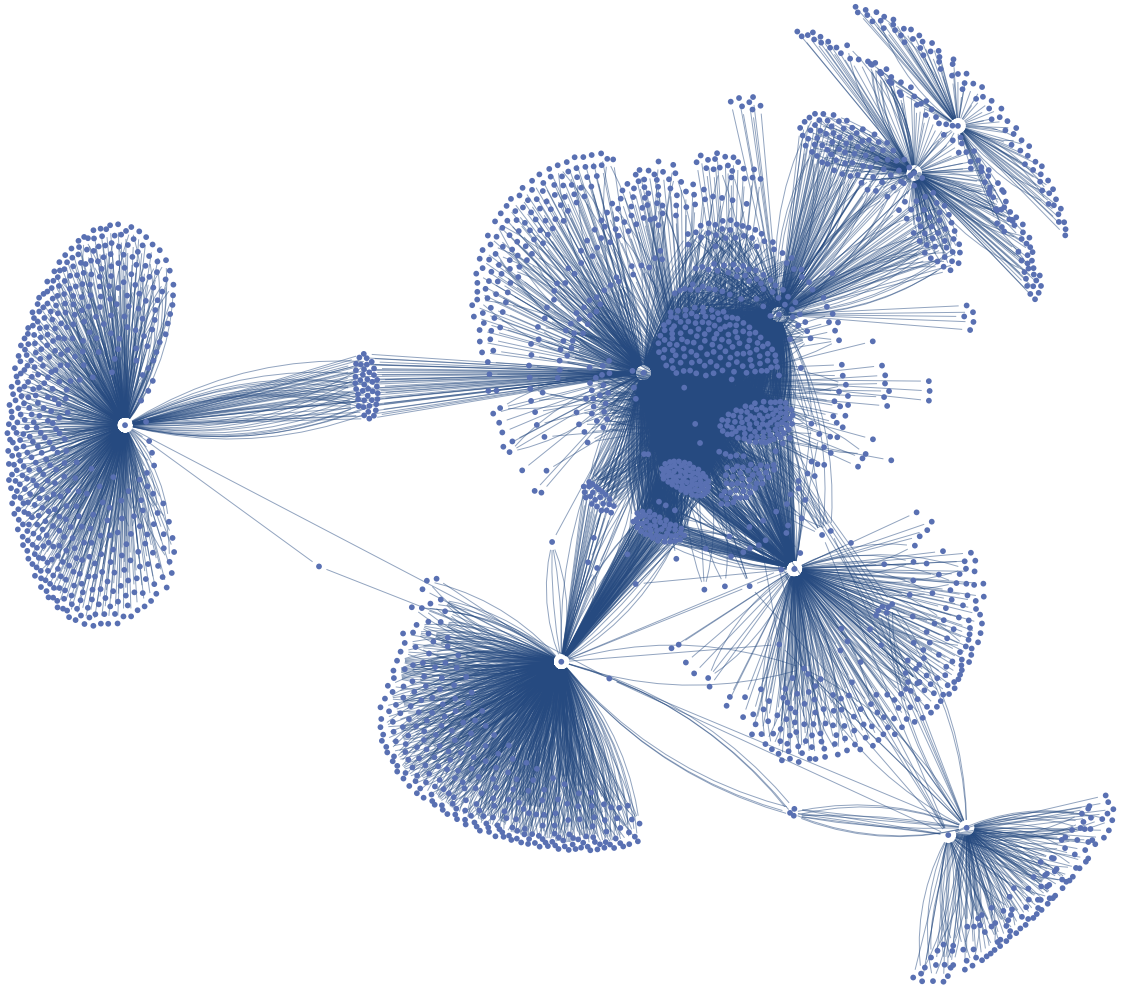
,



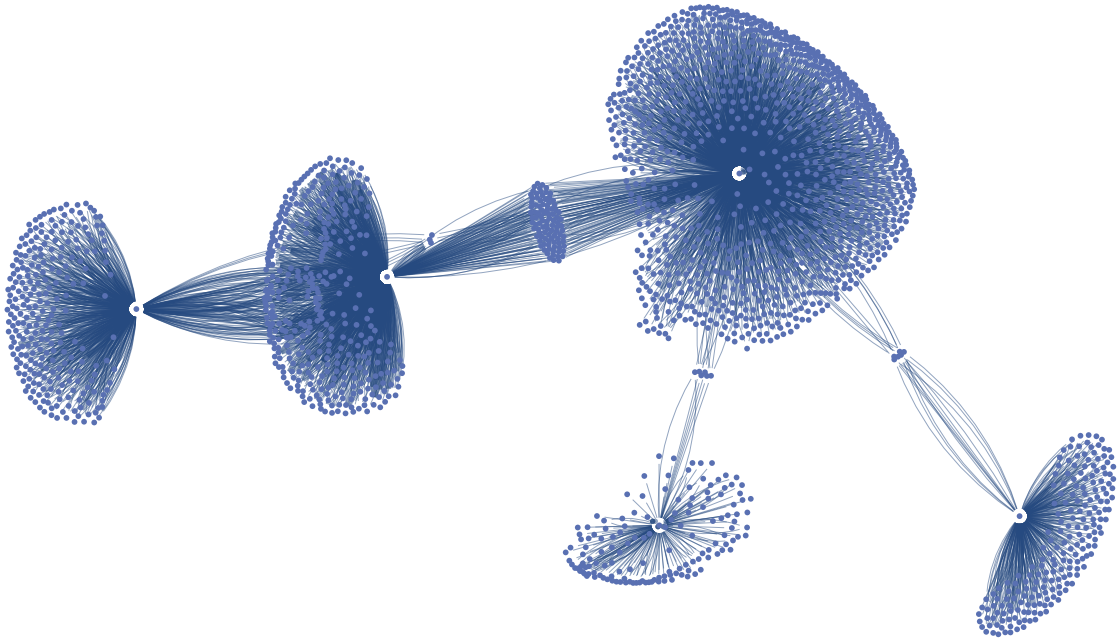
,



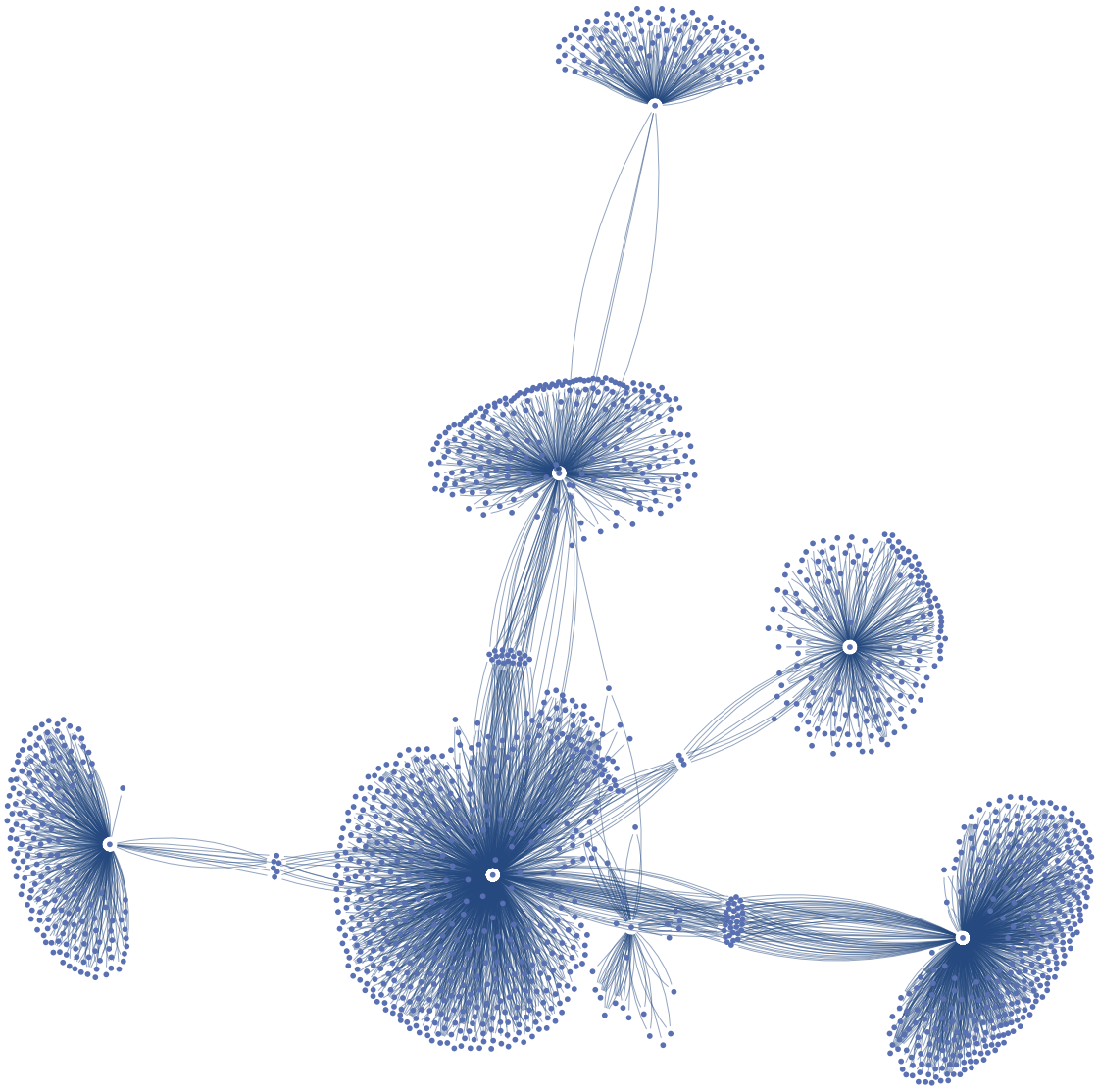
,



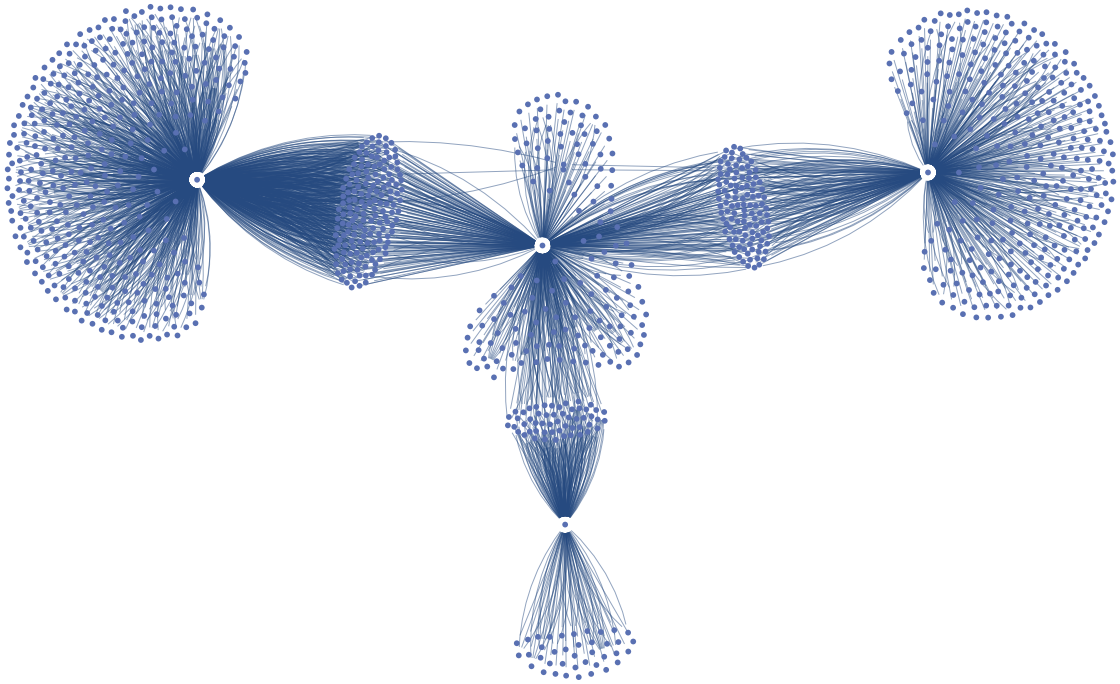
,



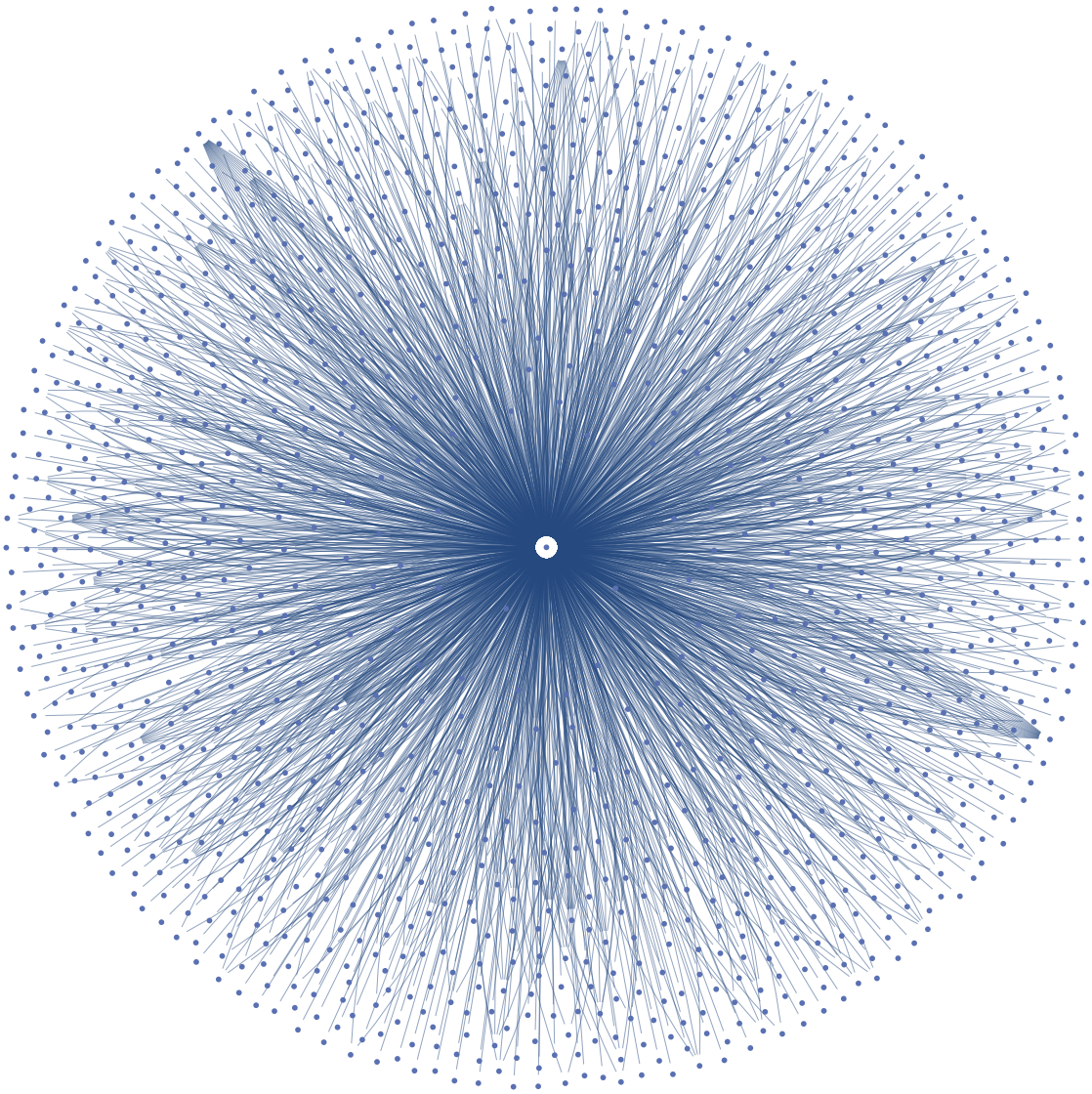
,



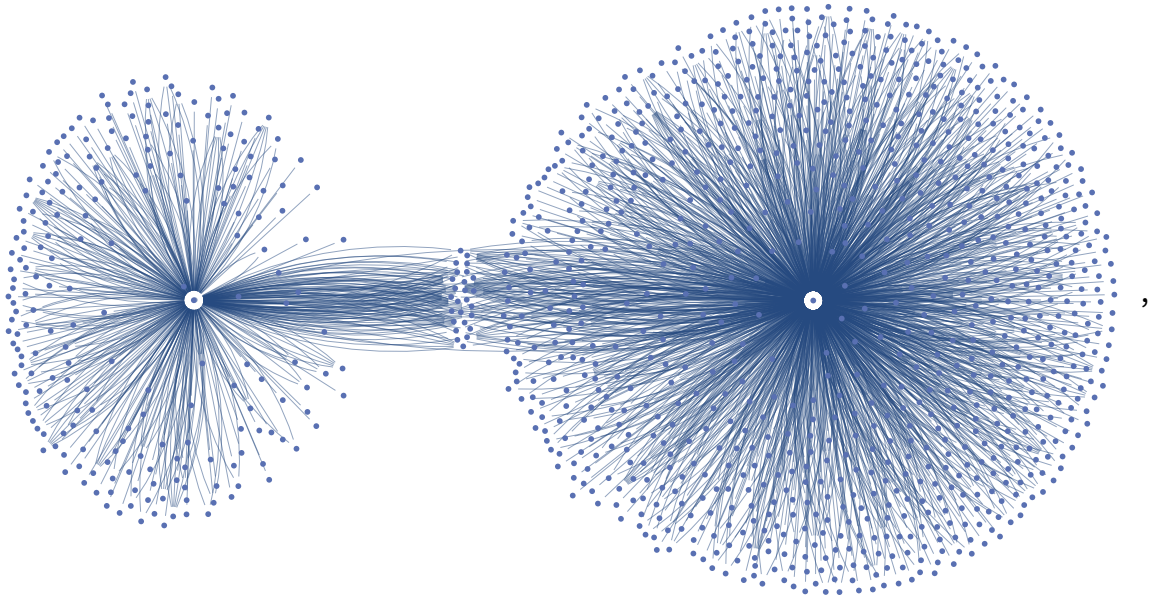
,



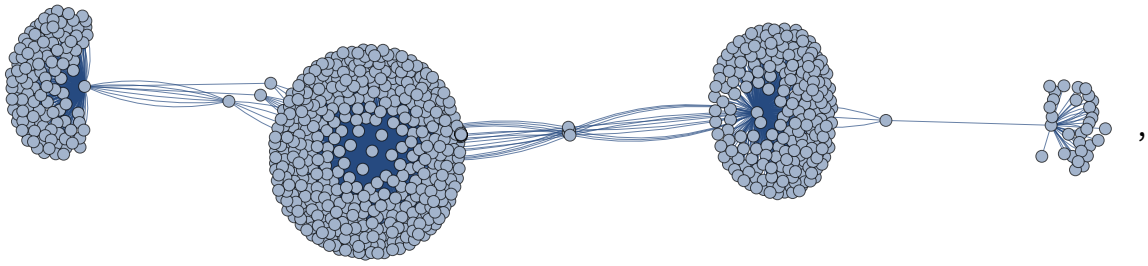
,



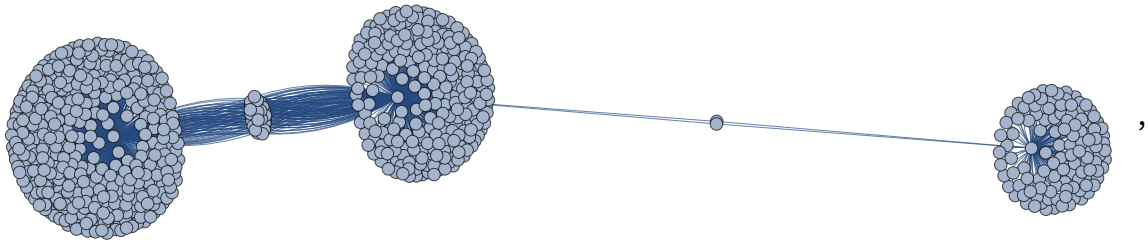
,



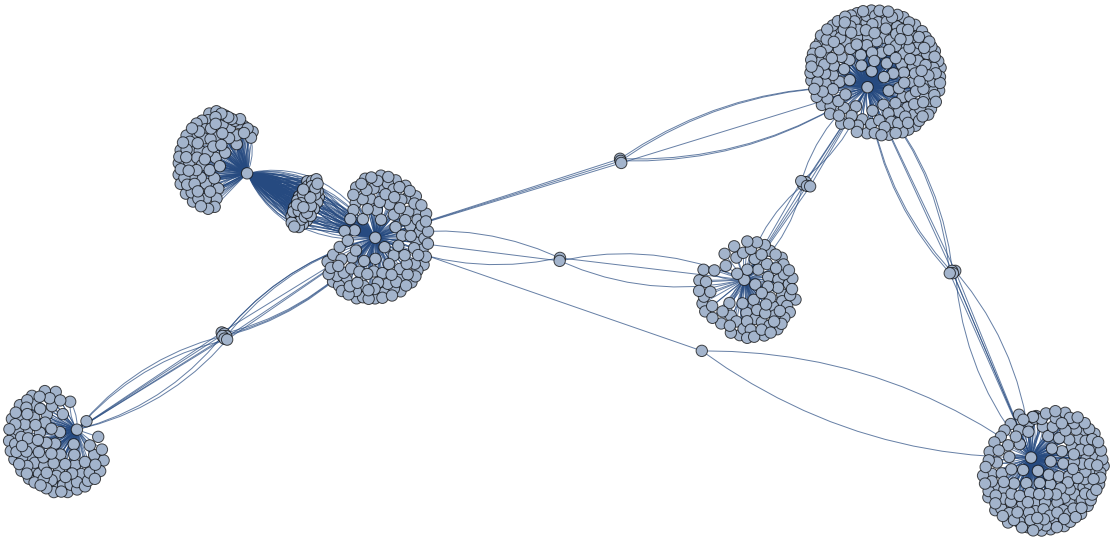
,



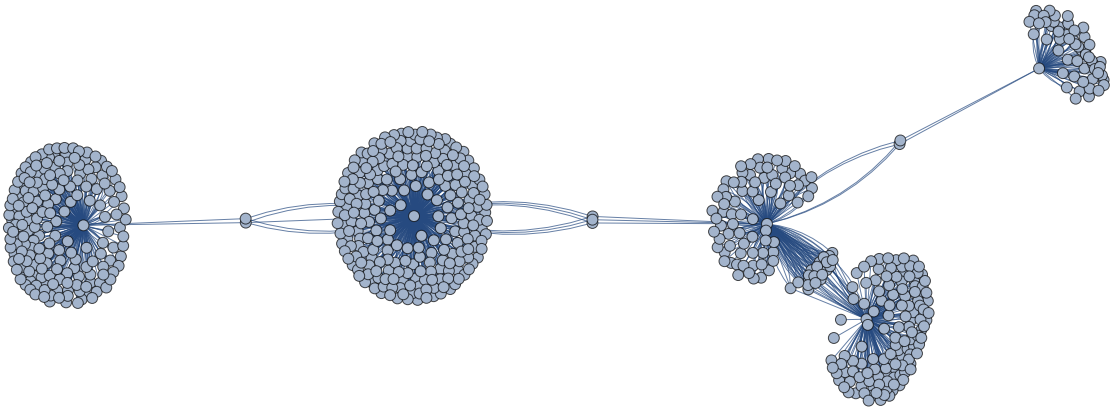
,



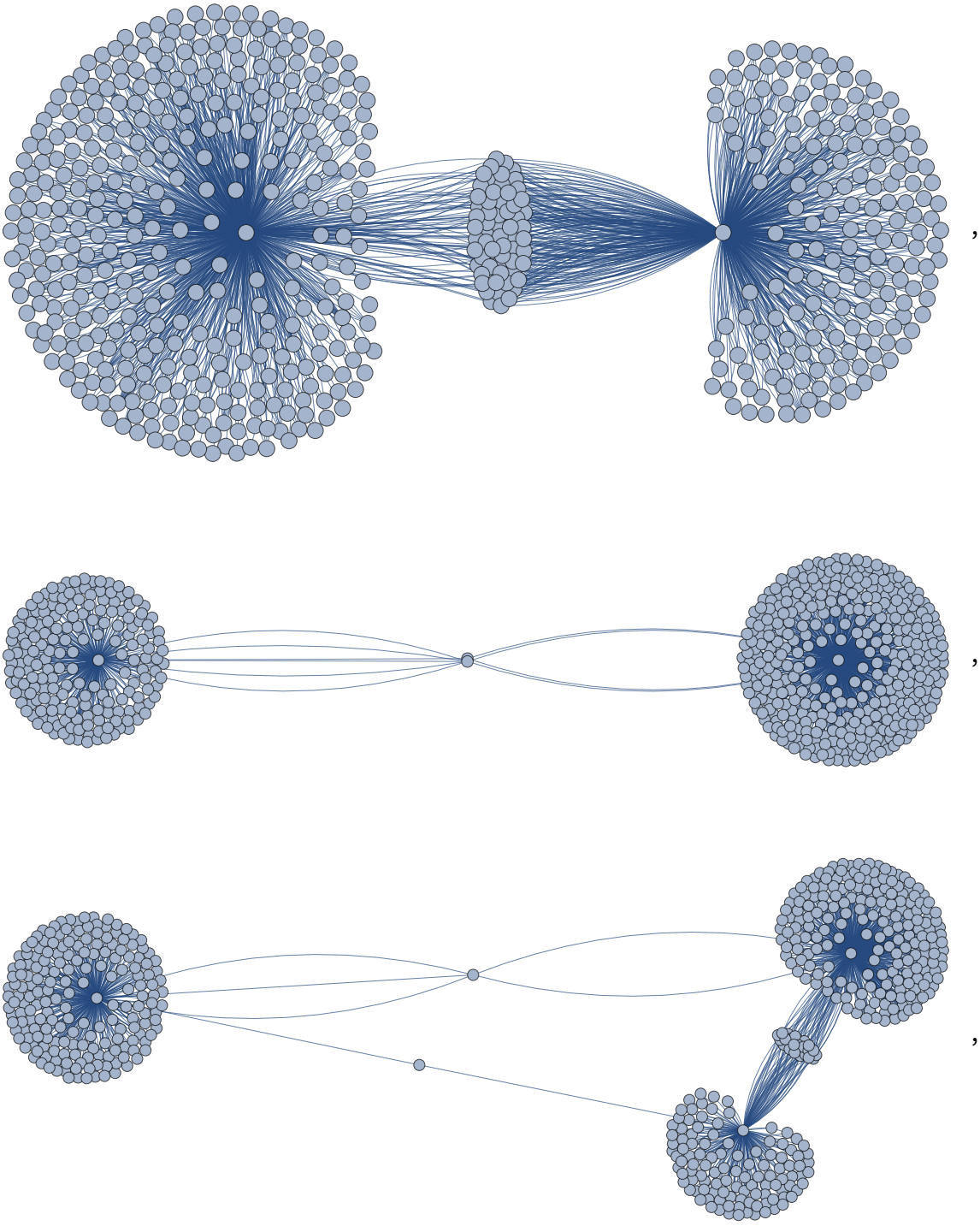
,

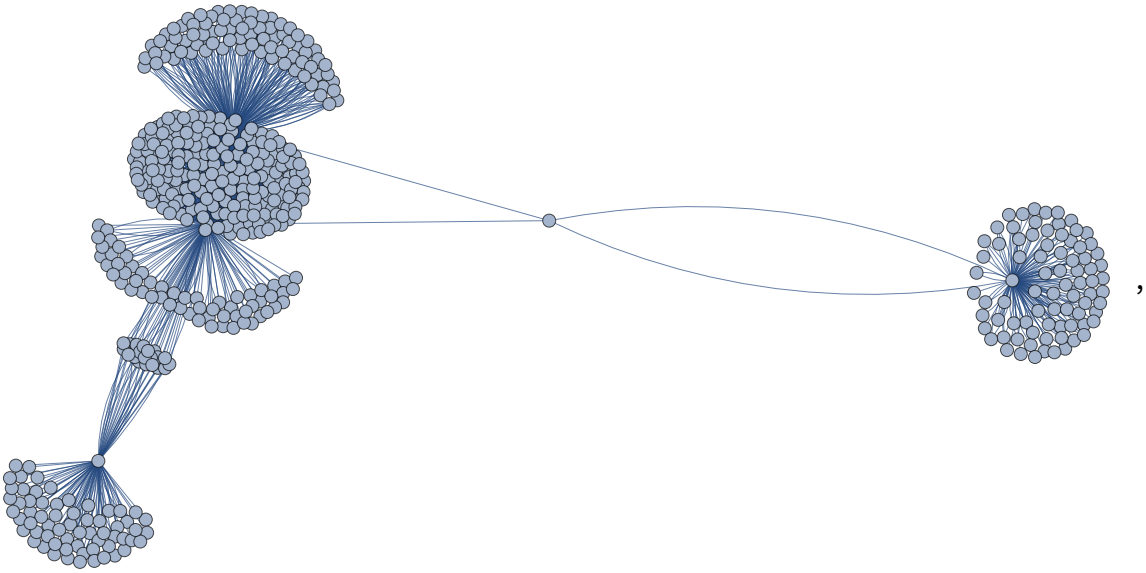


,

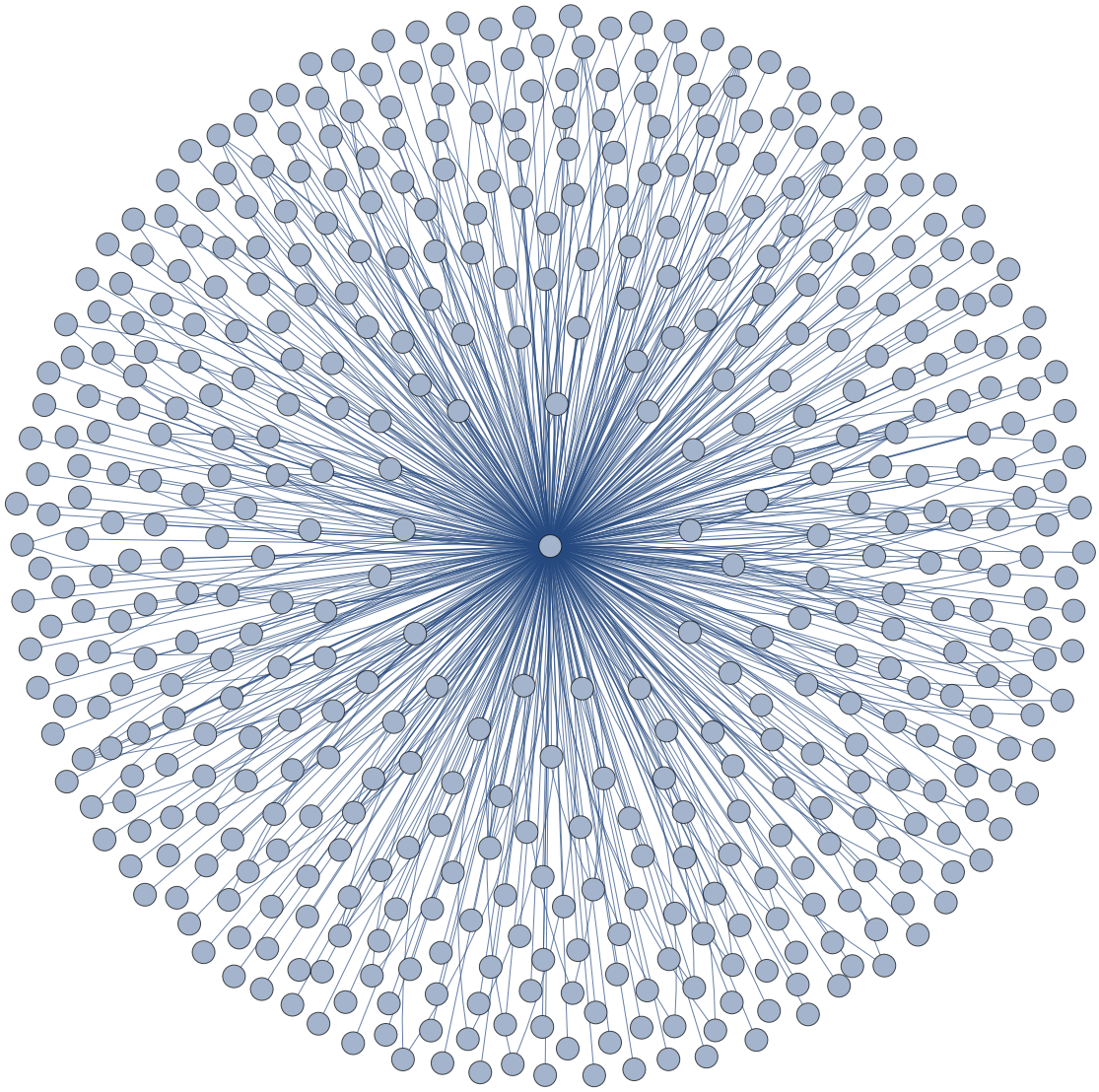


,

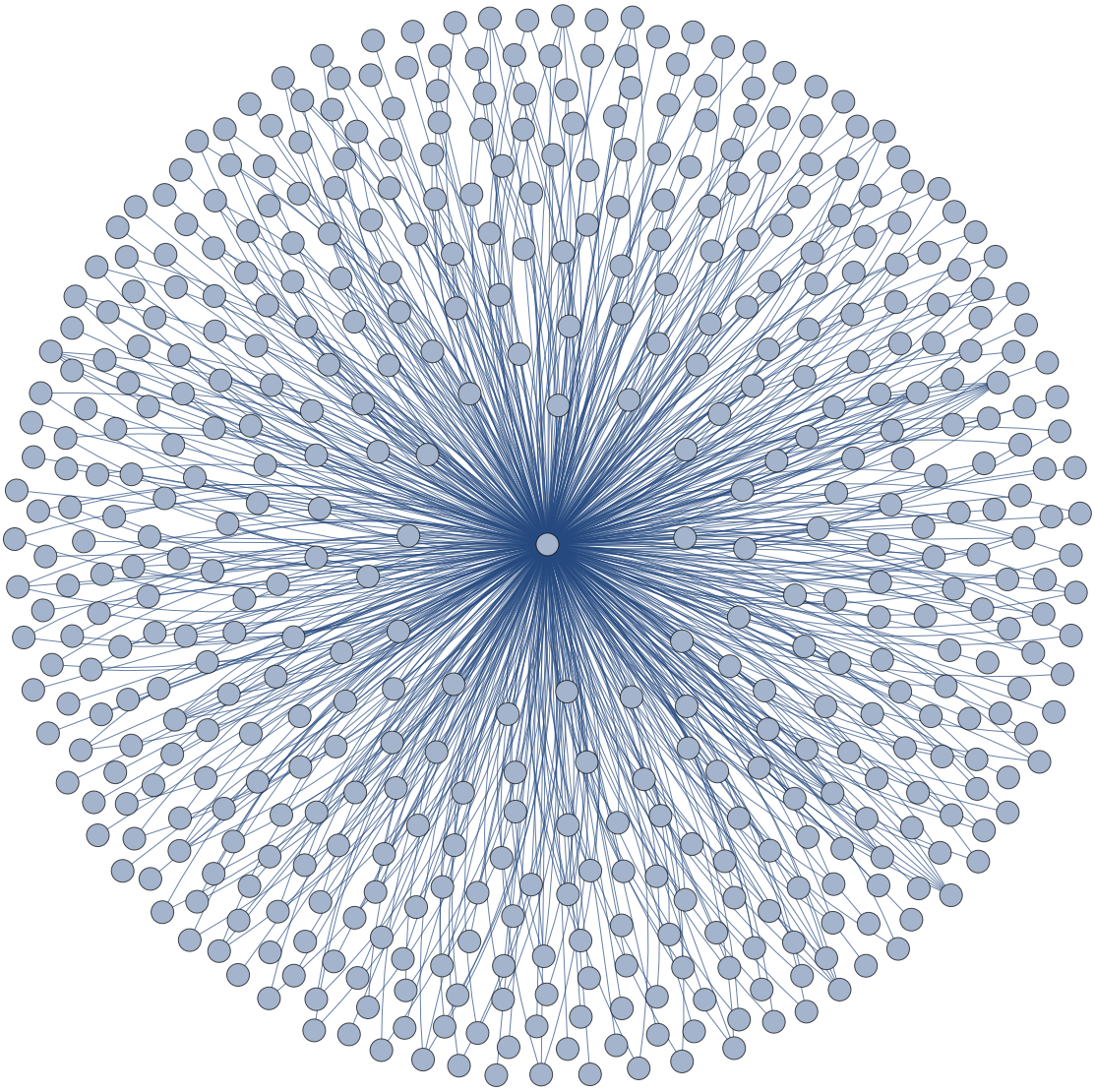




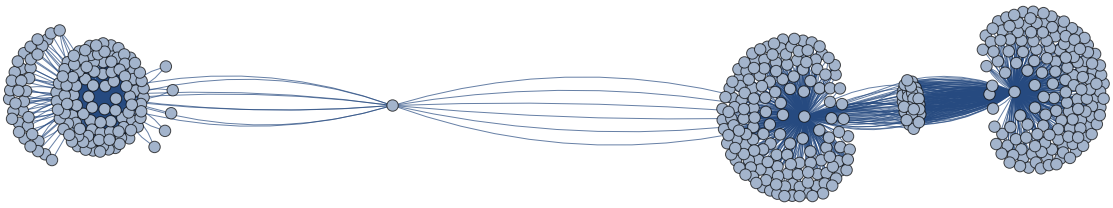
,



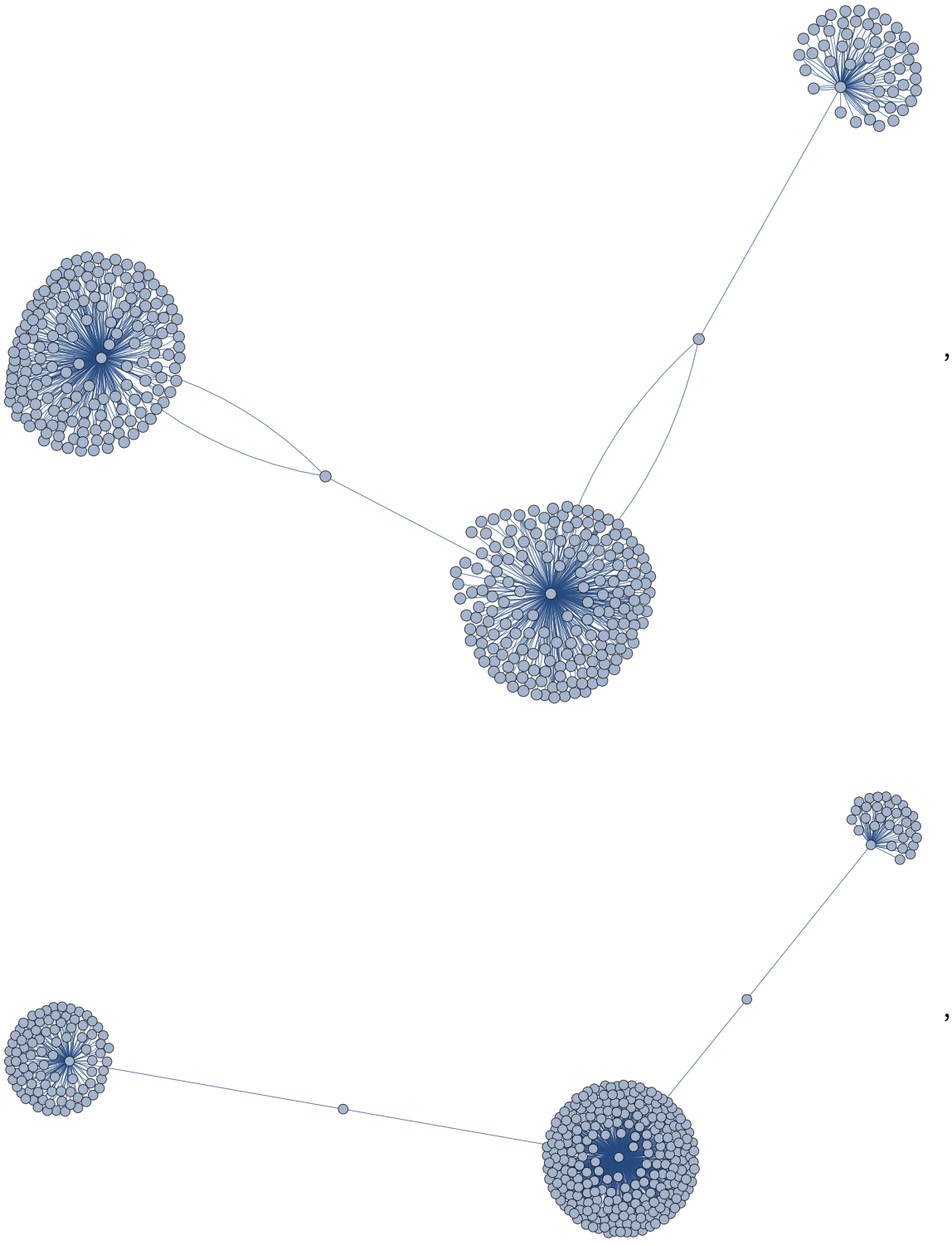
,



,

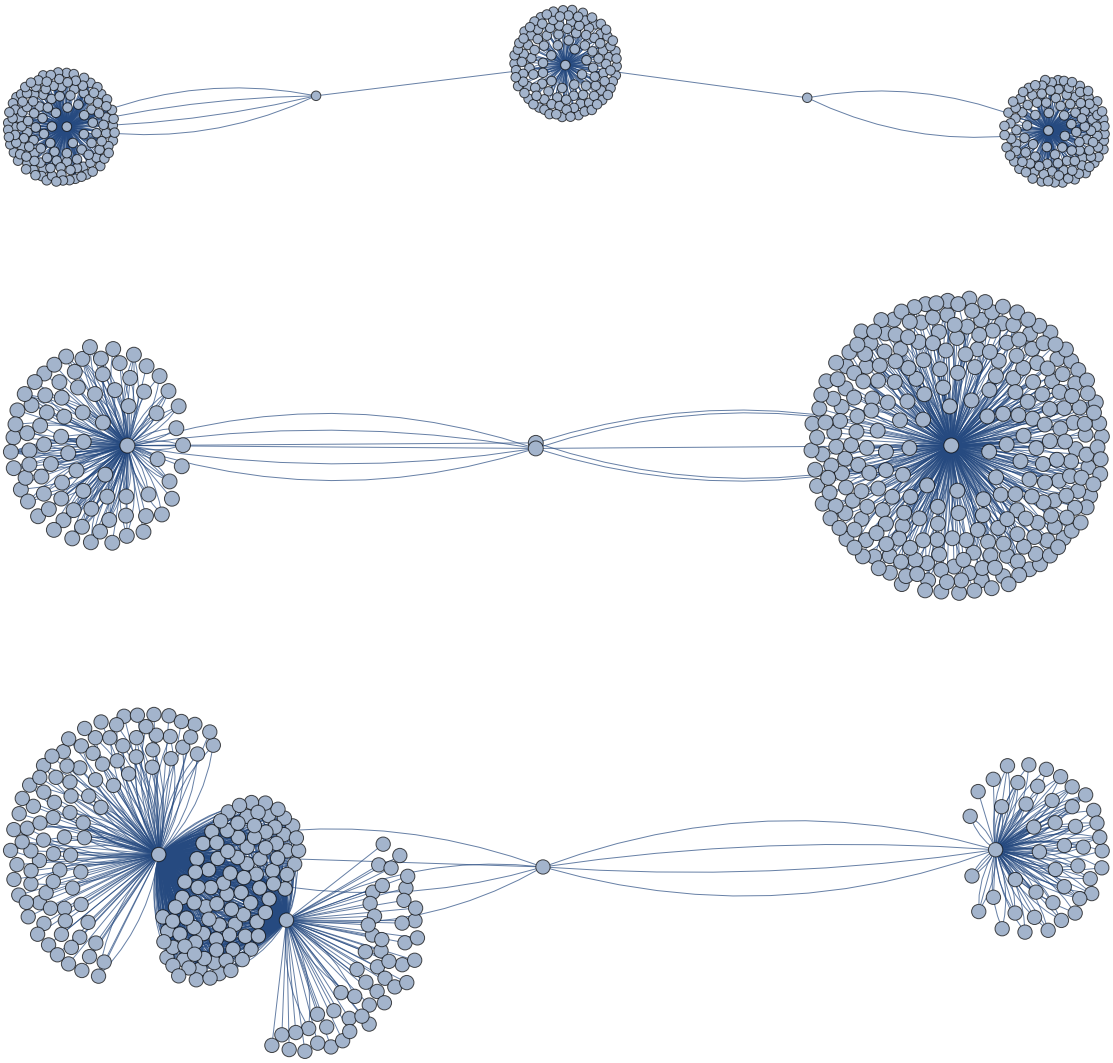


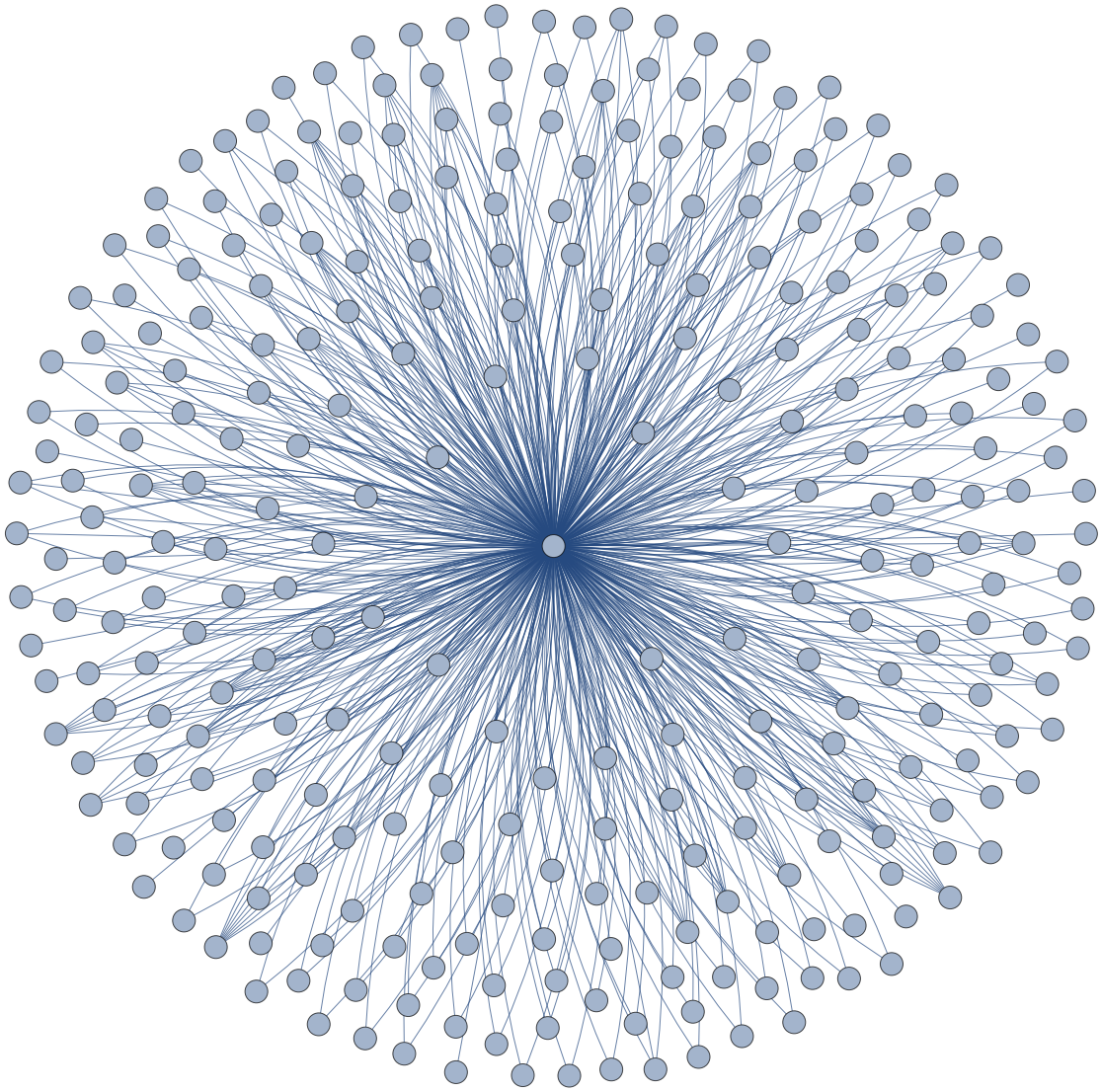
,



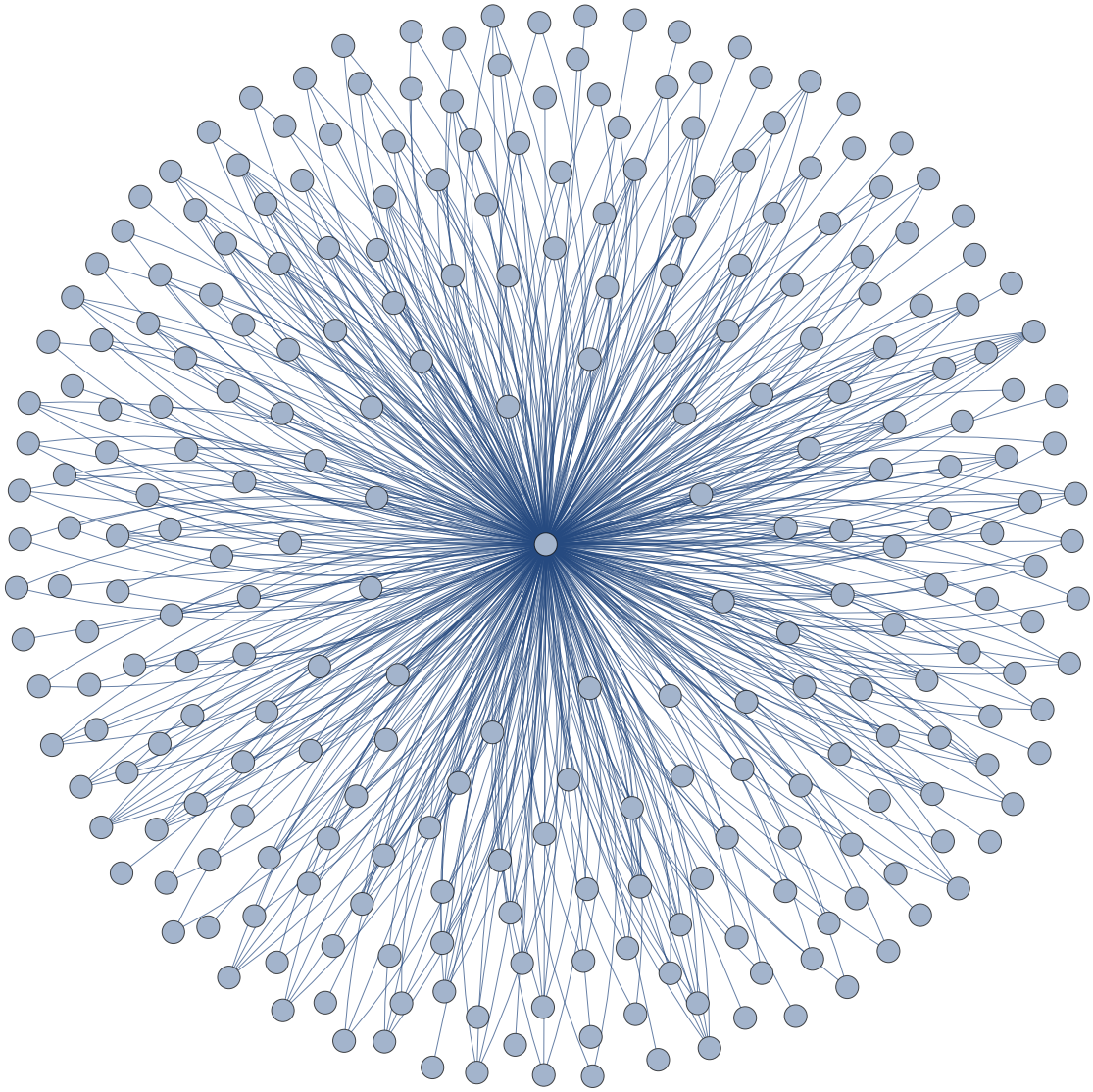
,

,

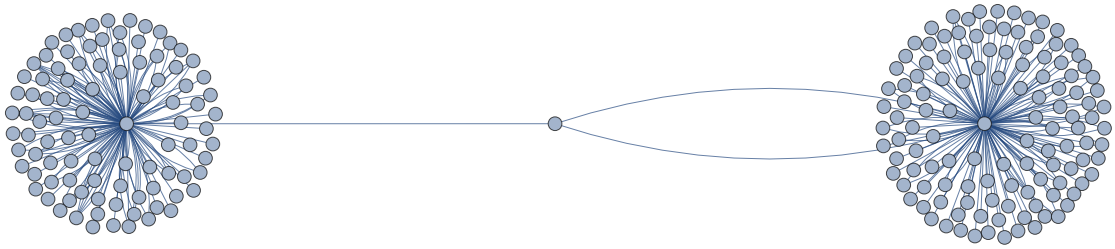




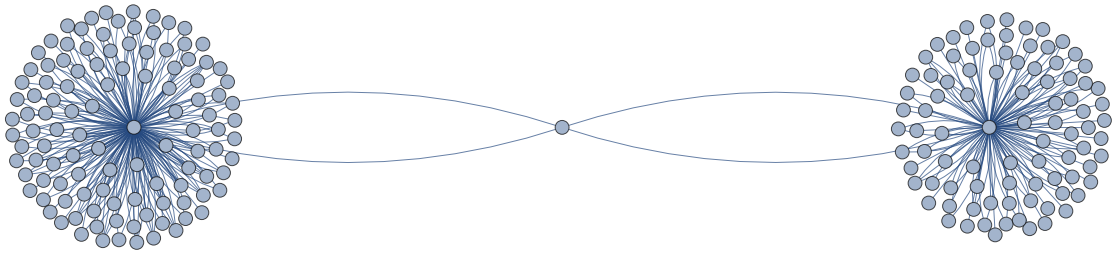
,



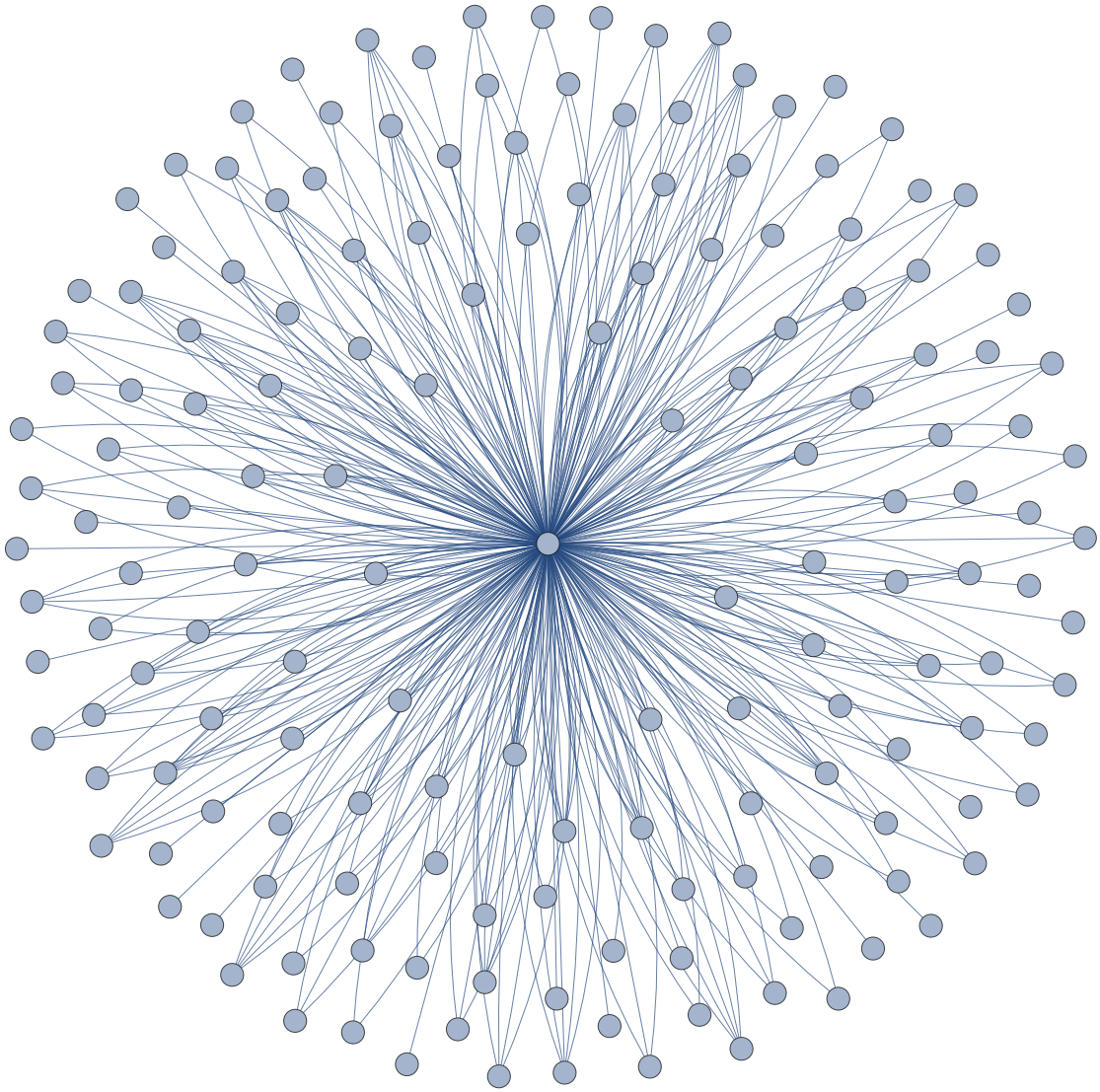
,



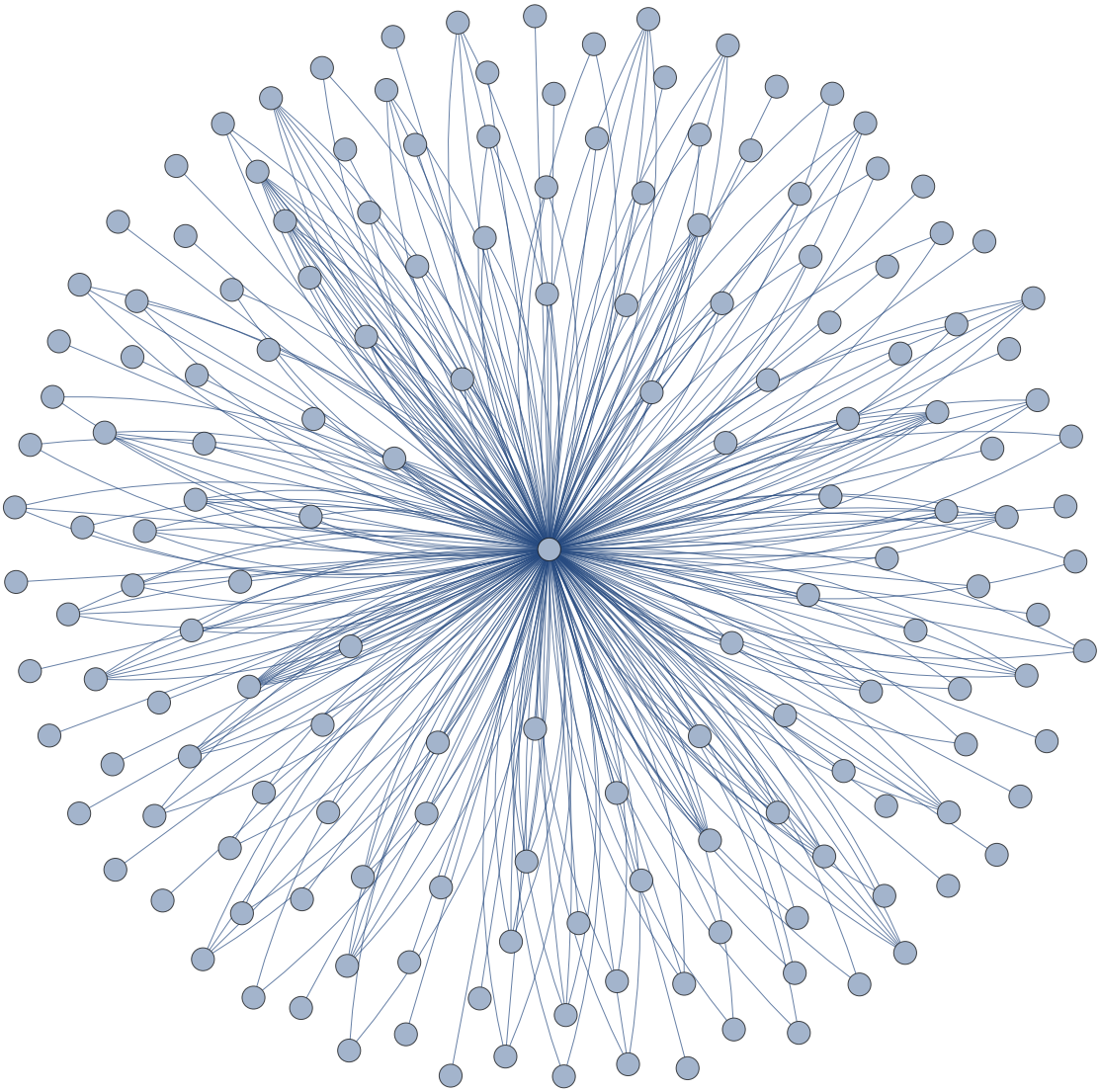
,



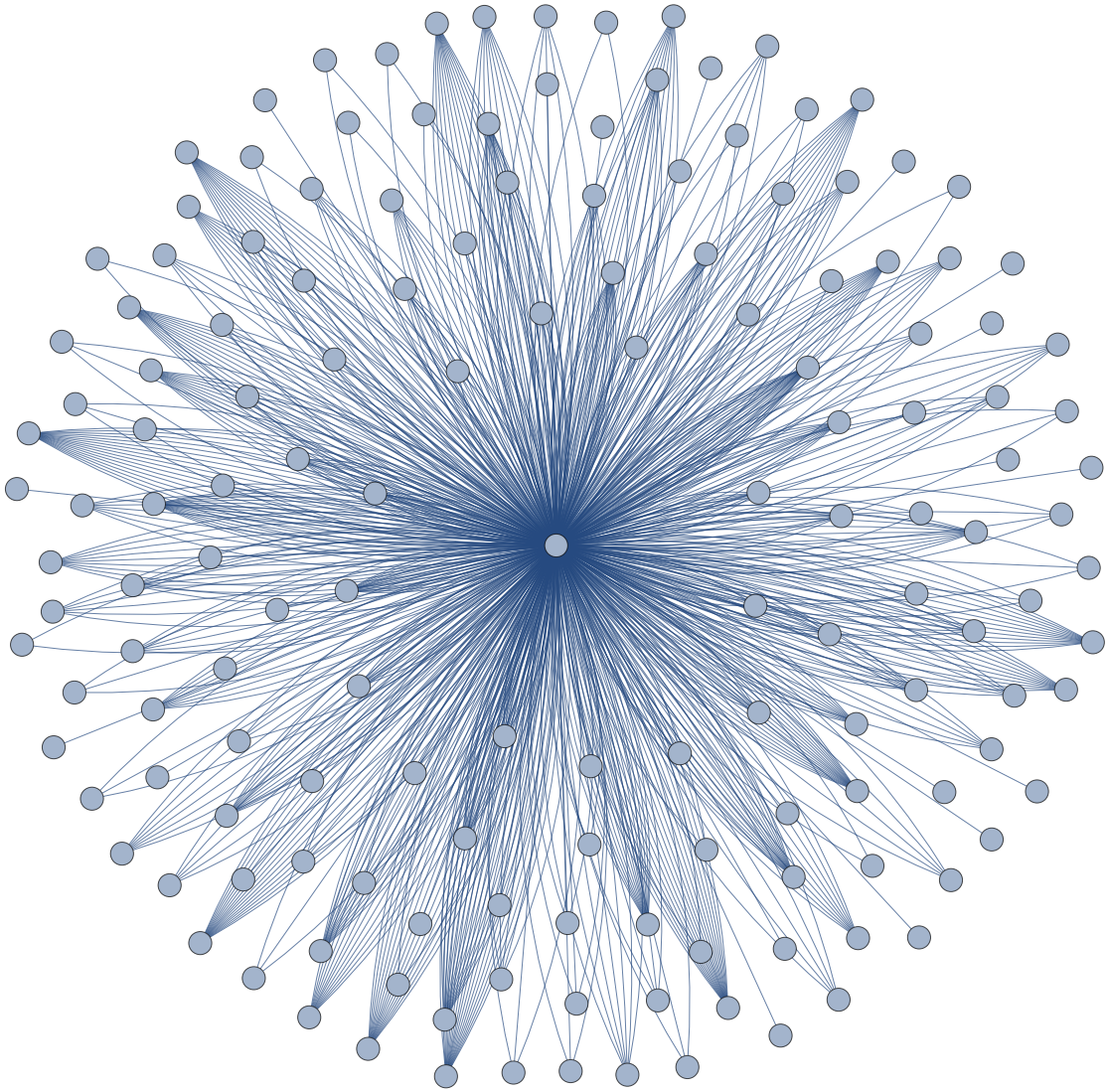
,



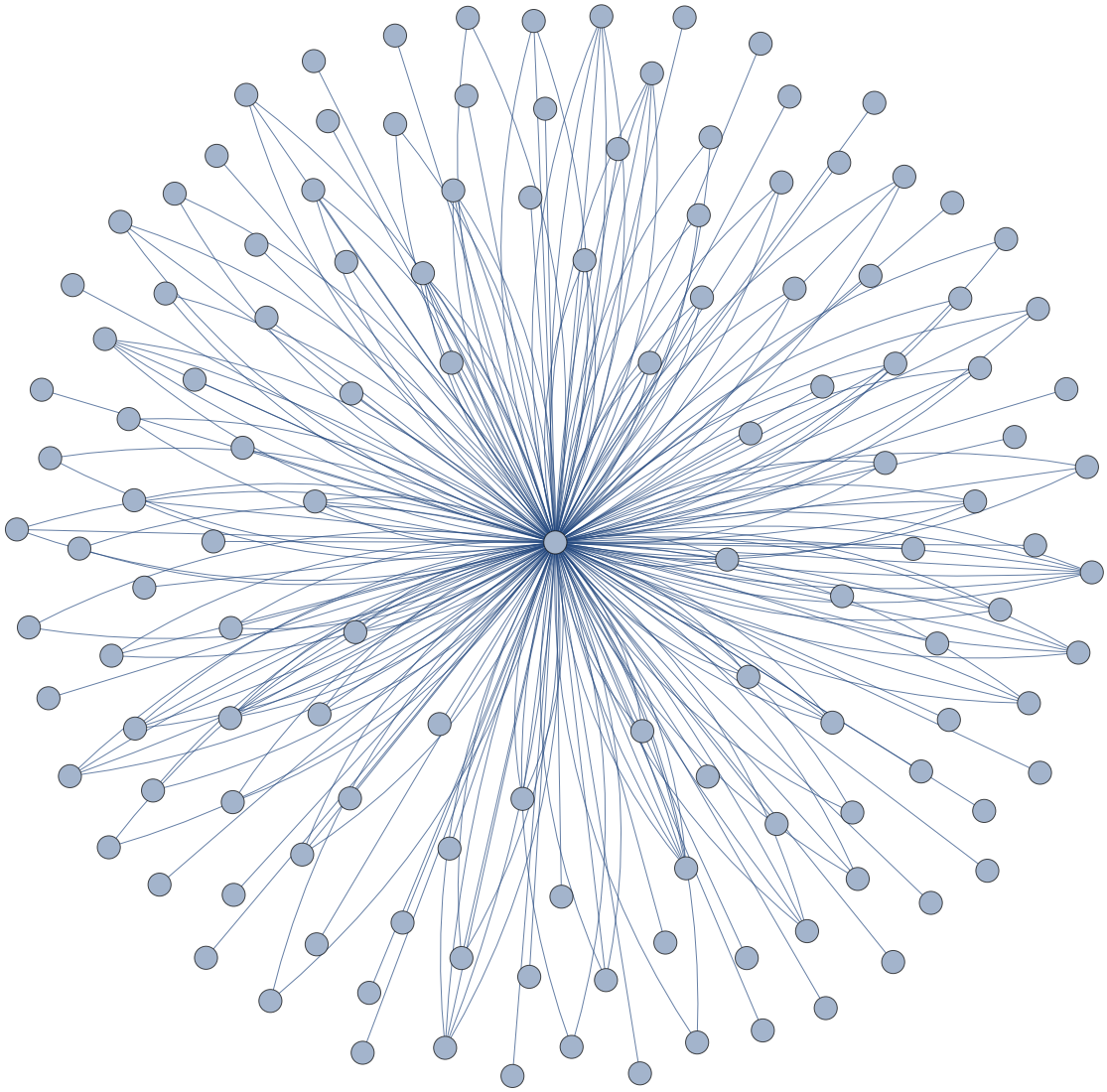
,



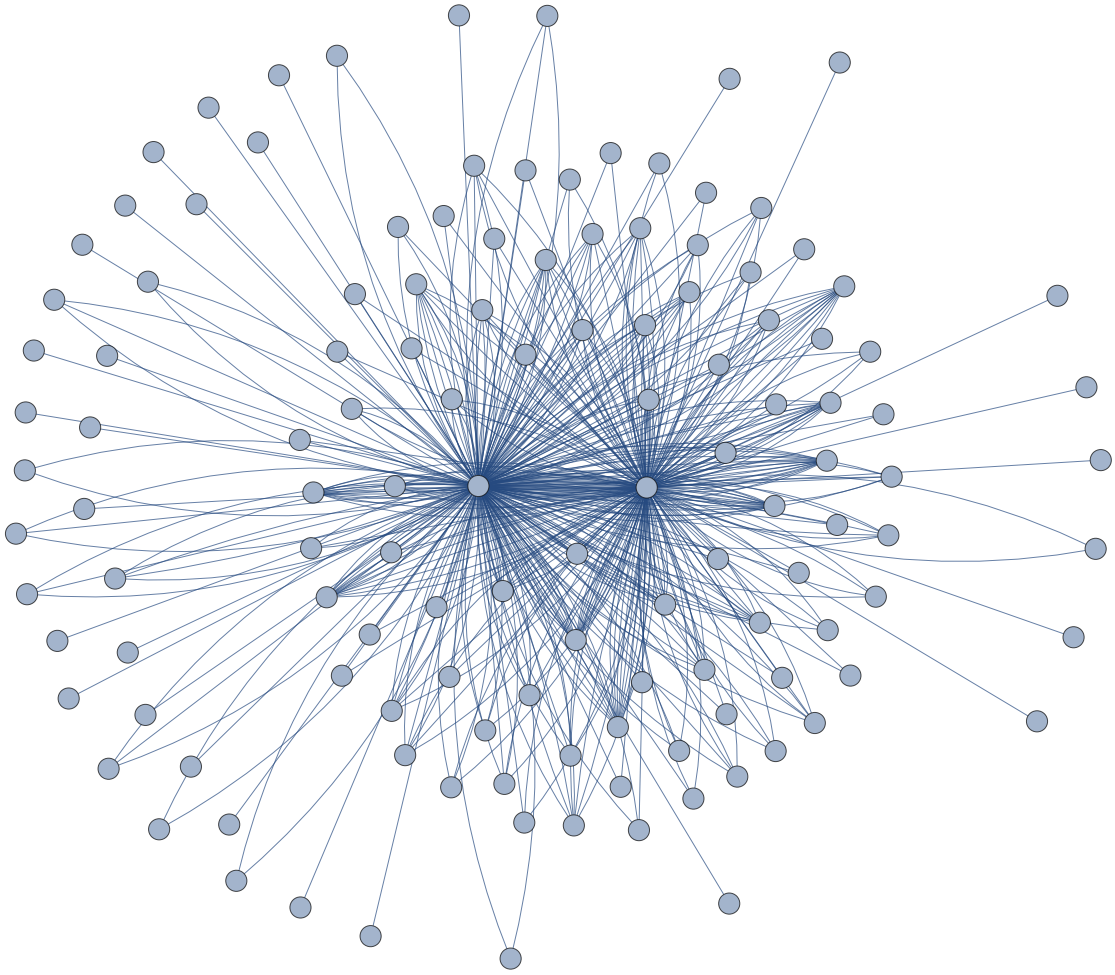
,



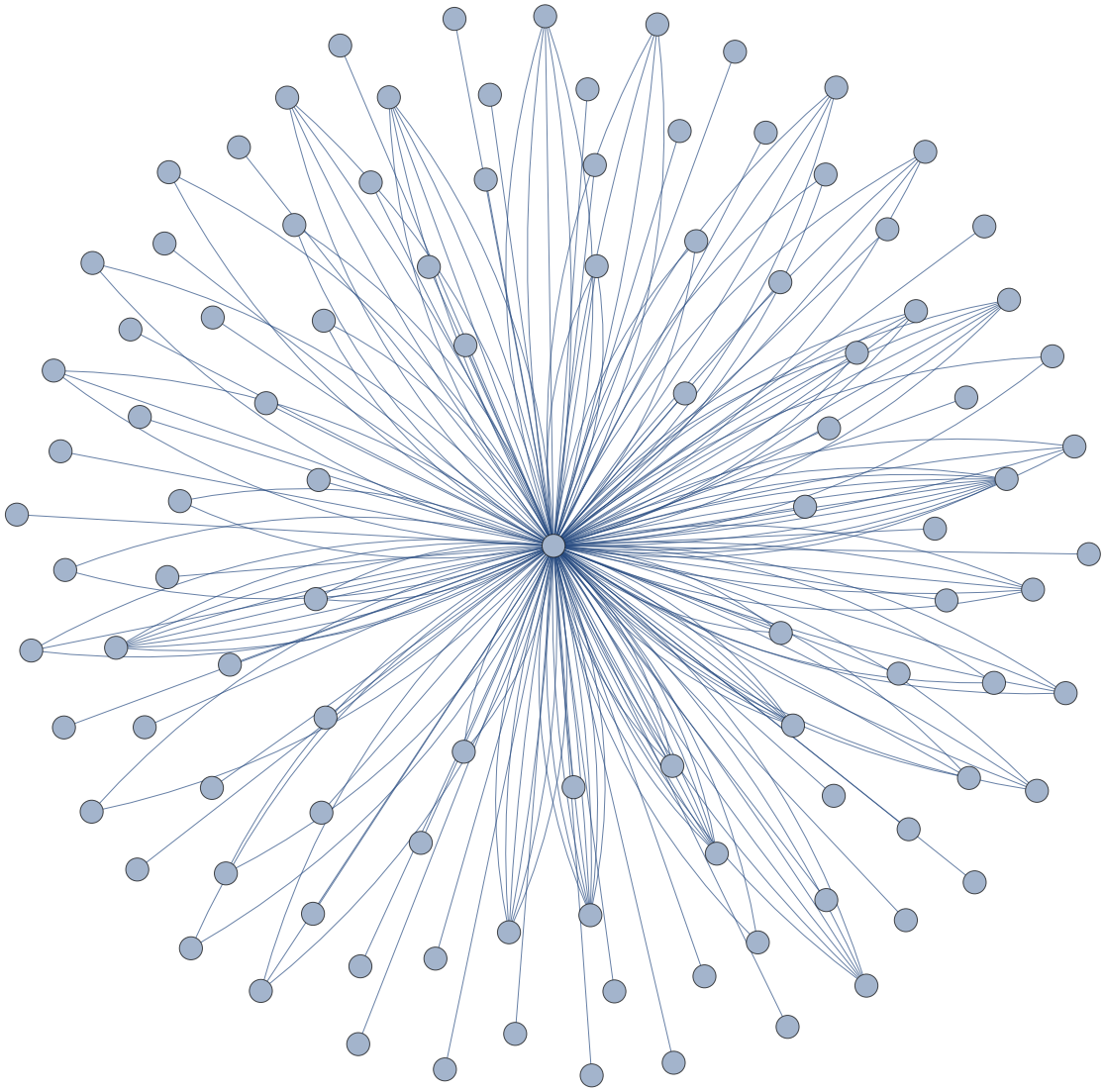
,



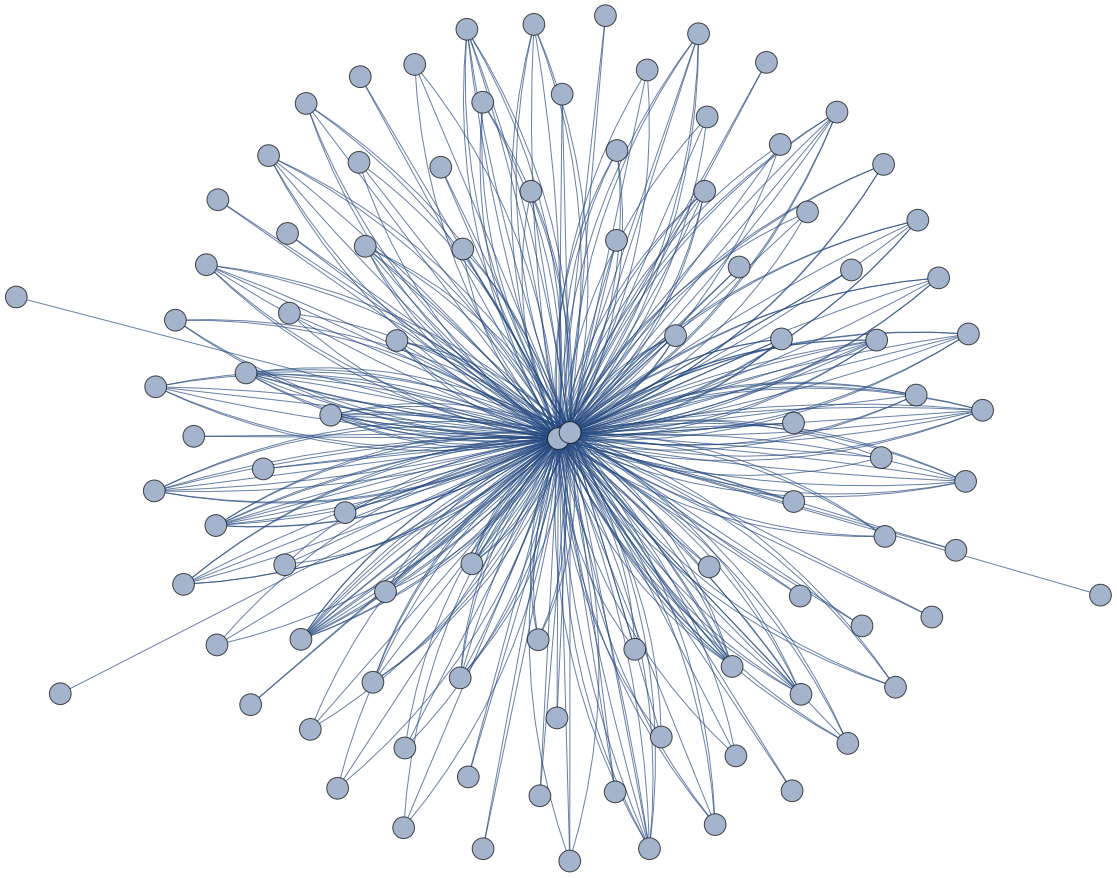
,



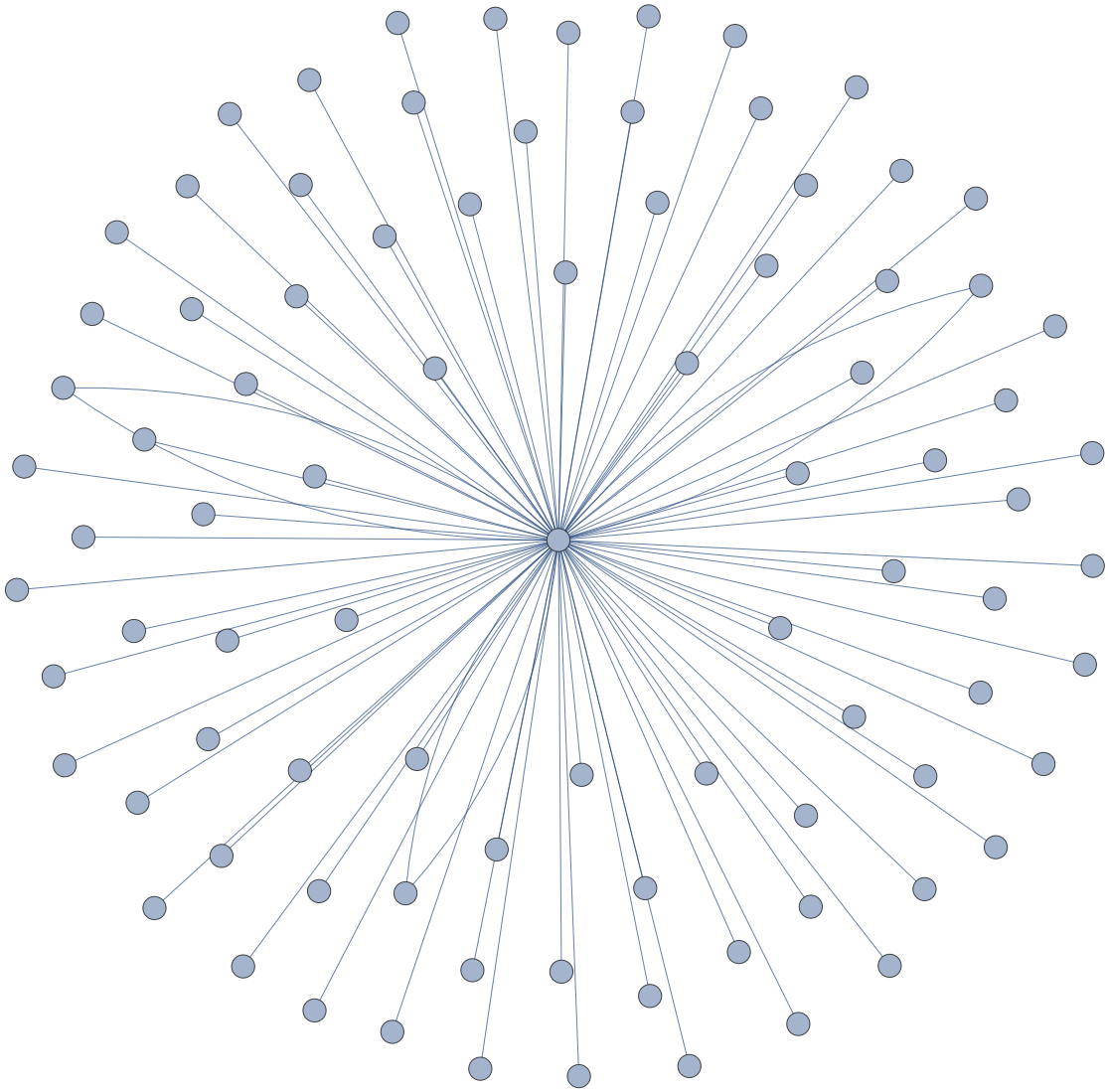
,



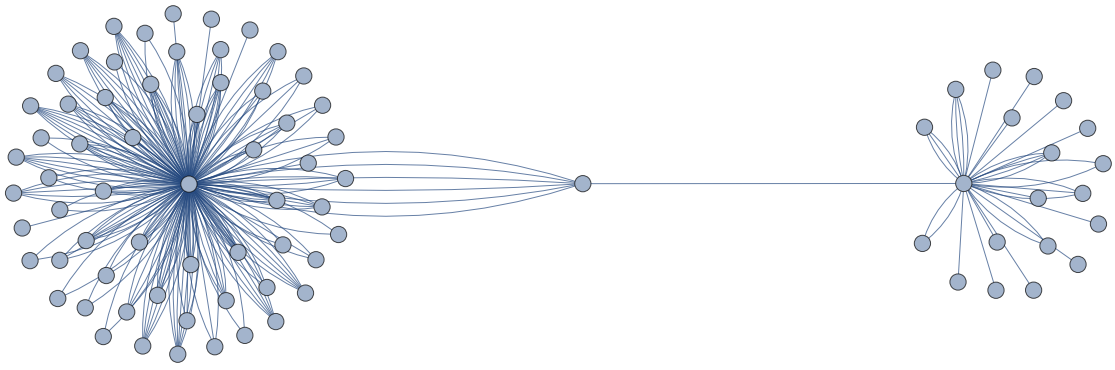
,



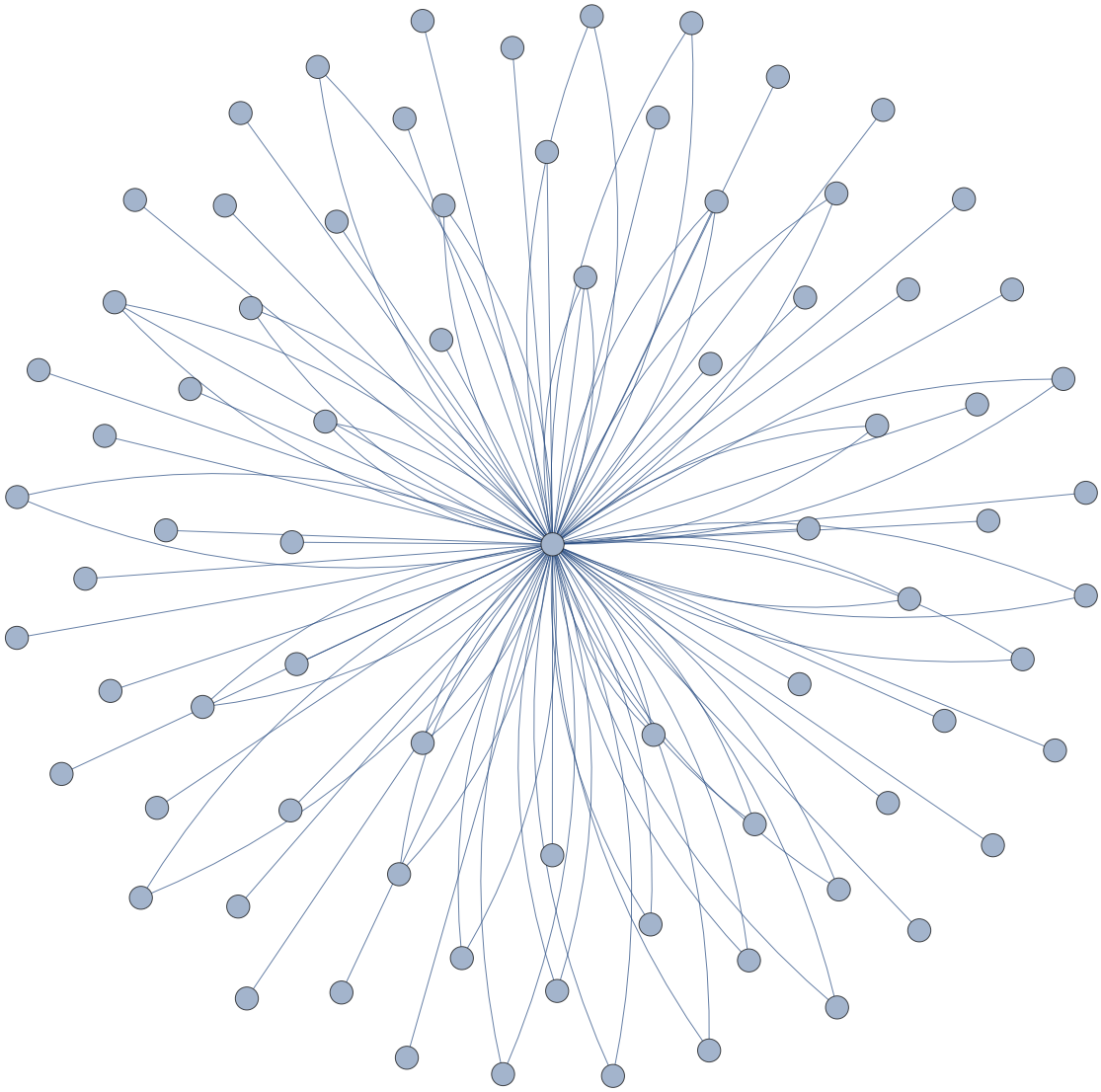
,



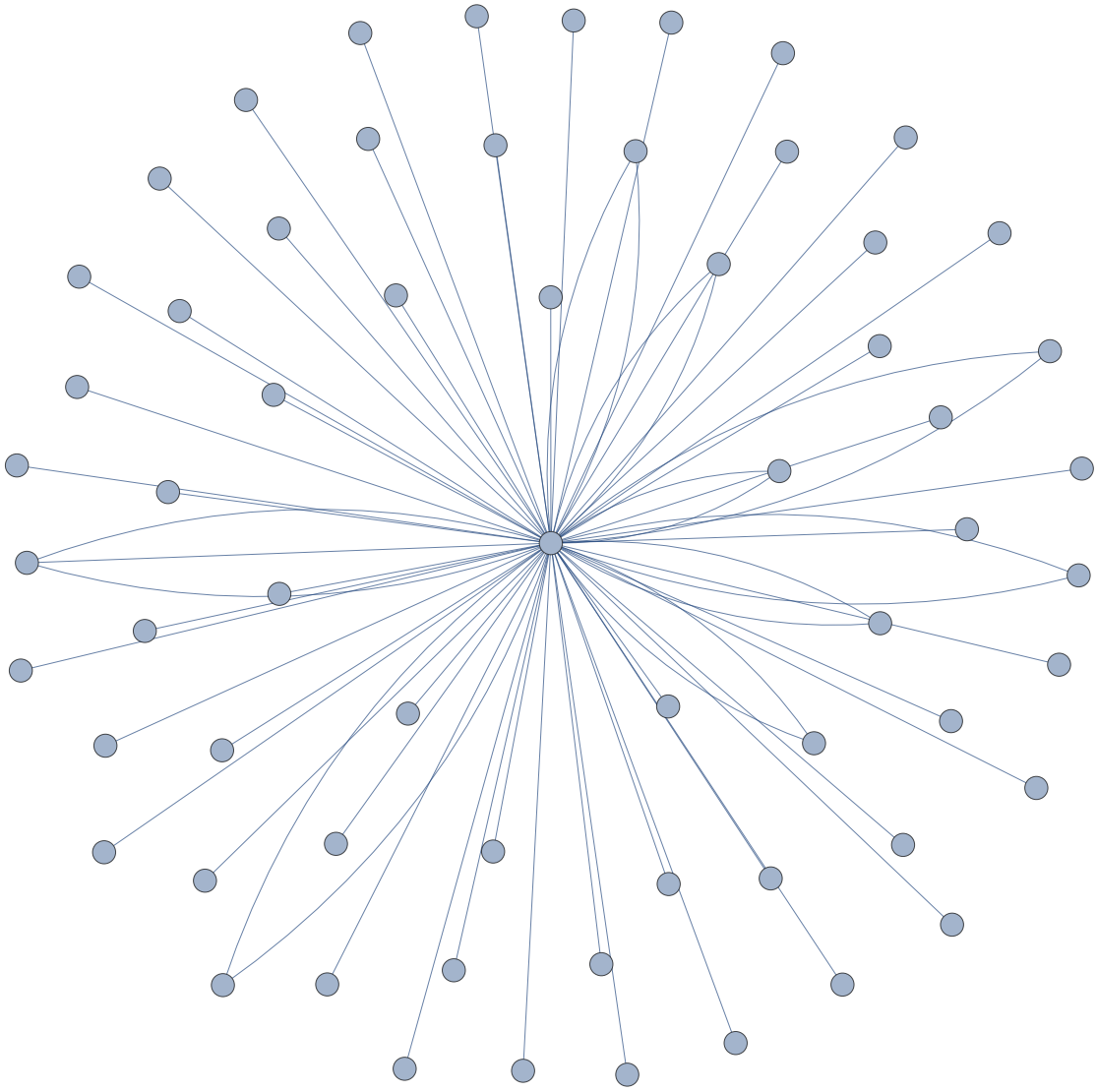
,



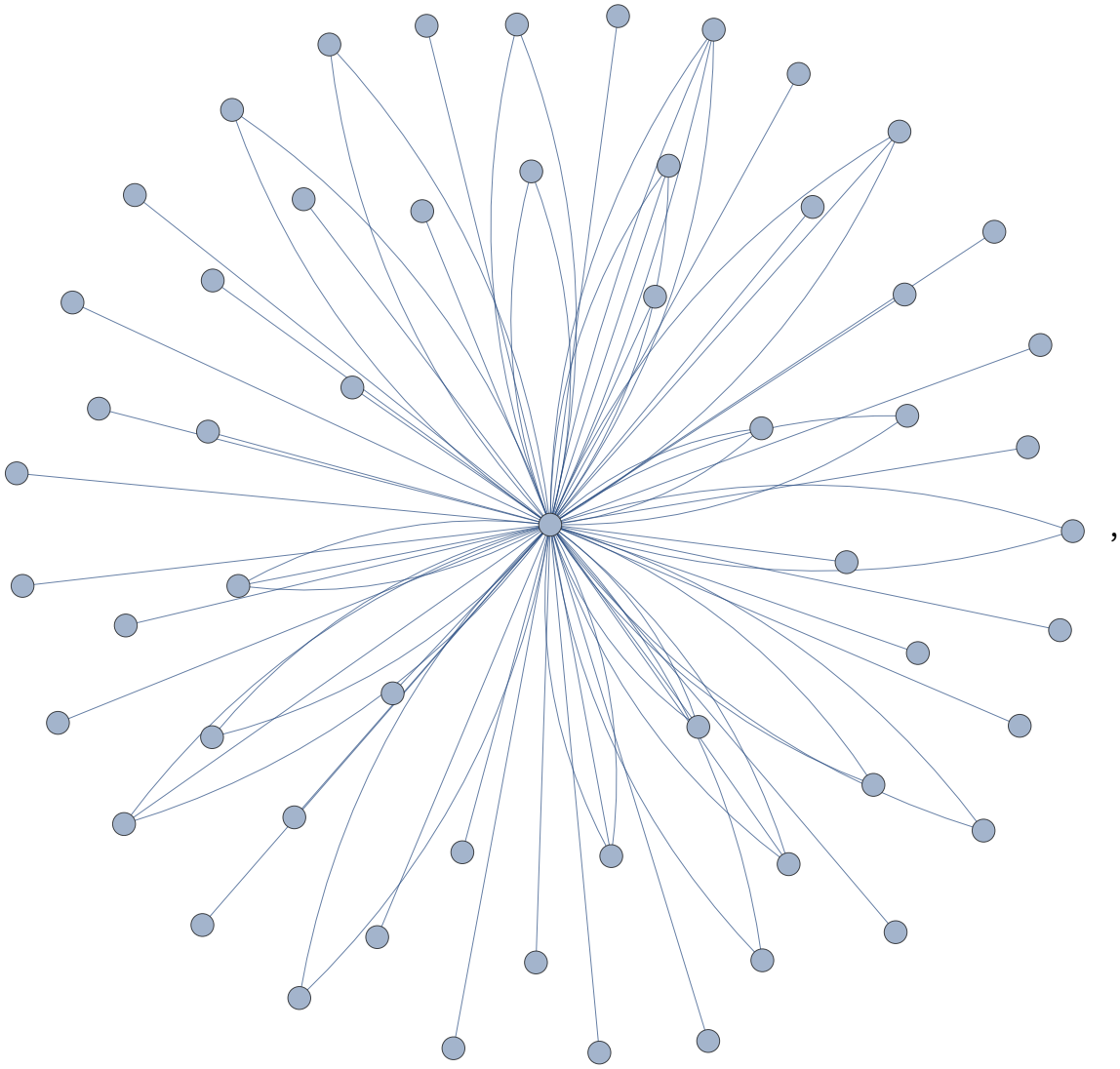
,



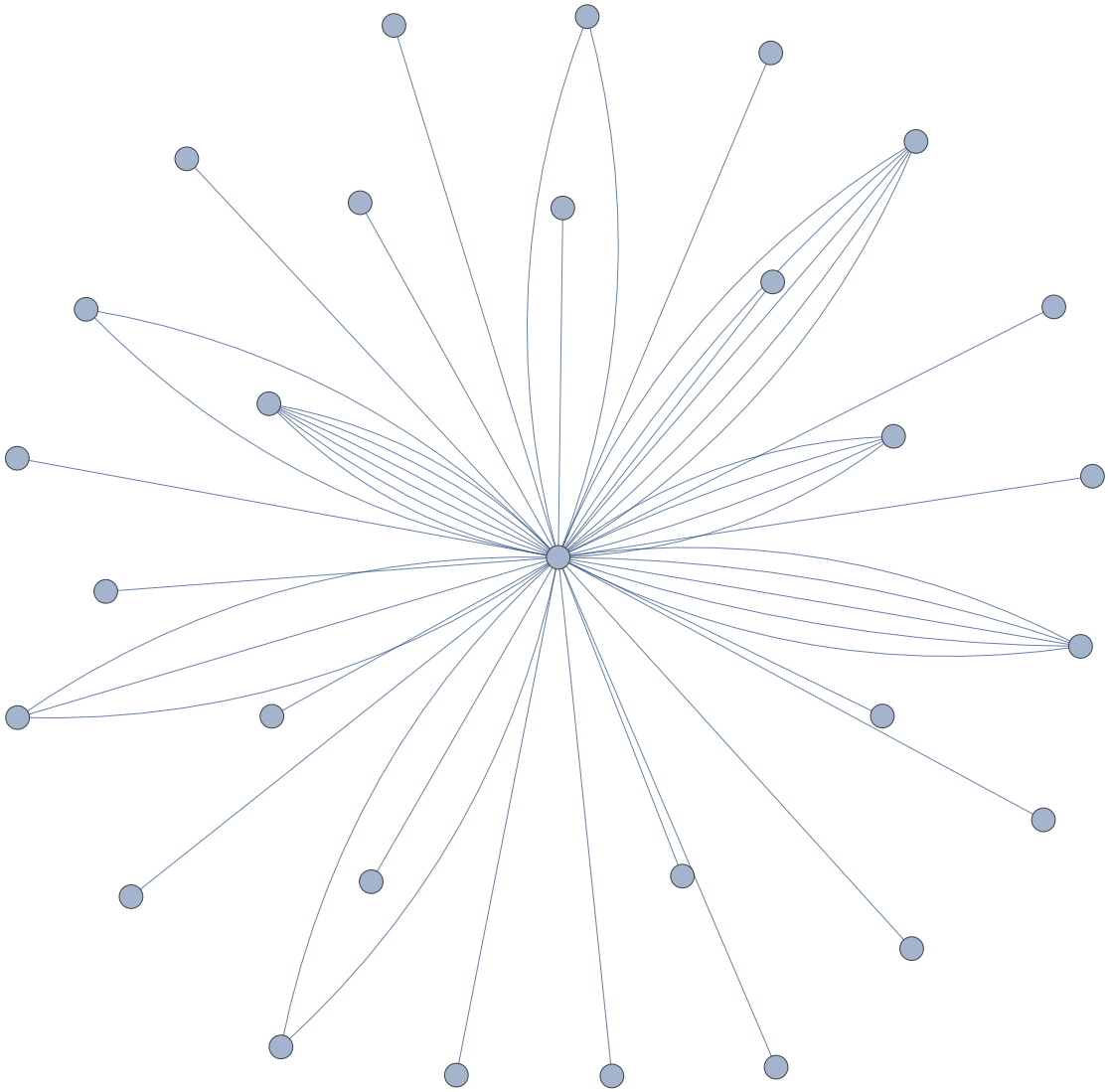
,



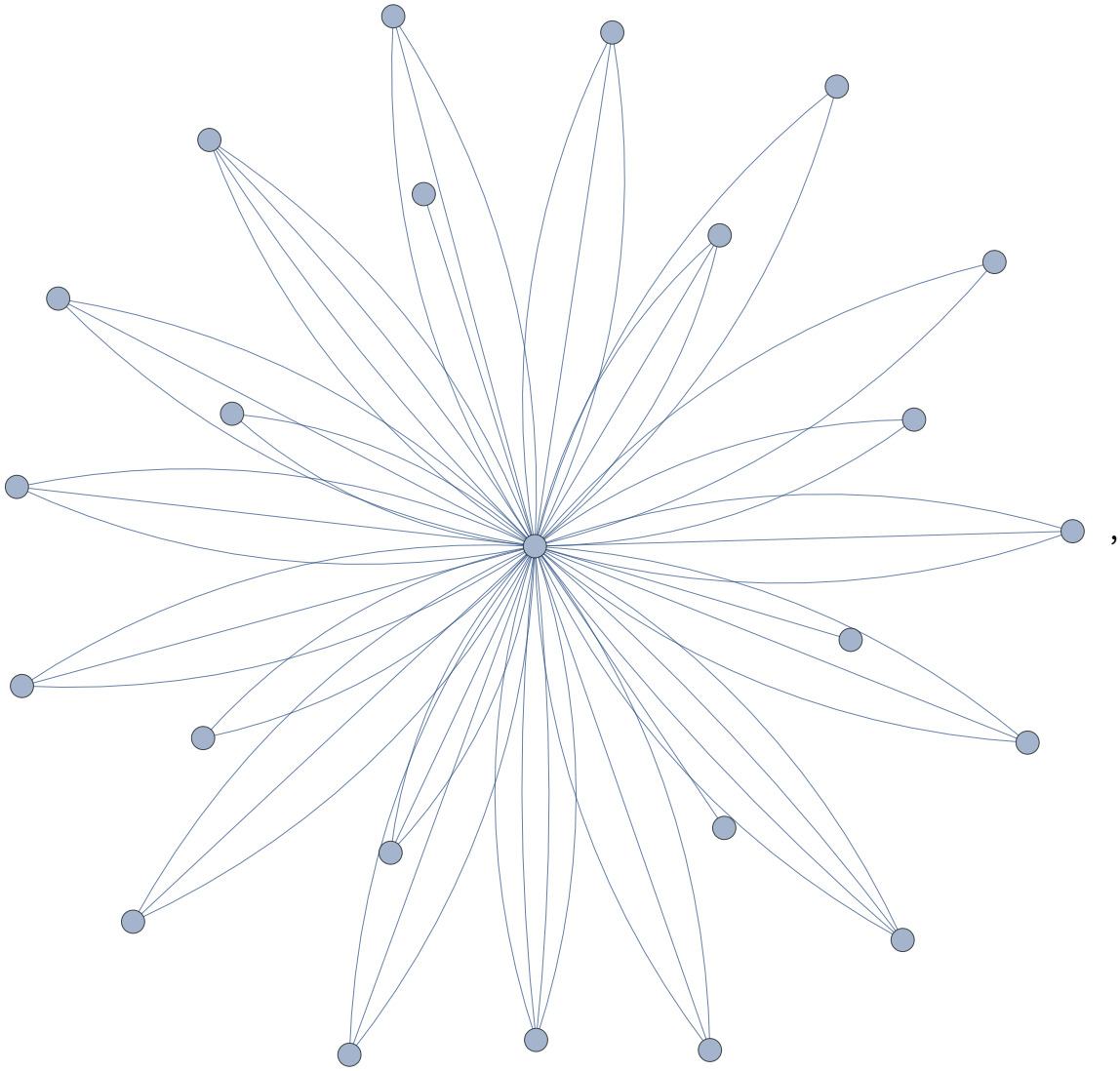
,



,



,



,