Data Import

In[ ]:=
```
SetDirectory[
  "C:/Users/serha/OneDrive/Masaüstü/MyRepo/master_thesis_MMT003/210324_disc_time_windows
    _and_OR_model"];
```

In[ ]:=
```
Get["../algoritm_packages/SingleNetworks-algorithm-package.wl"]
(* ?SingleNetworks`* *)
```

In[ ]:=
```
datafull = Import["../data/ccm_manipulated_396096.csv", HeaderLines → 1];
```

In[ ]:=
```
data = TakeList[datafull,
   {39871, 39696, 38791, 40063, 39620, 39311, 39795, 39264, 39974, 39711}];
```

In[ ]:=
```
<< IGraphM`
```

Out[ ]=
```
IGraph/M 0.5.1 (October 12, 2020)
Evaluate IGDocumentation[] to get started.
```

Investigation of Constraints Impact in Time Windows

Defining Fixed Step Size and Fixed Bucket Size

Girvan & Newman Modularity

M.E.J.Newman, Modularity and community structure in networks (2006).

$$Q = \frac{1}{4\,m}\sum_{i\,j}\left(A_{i\,j} - \frac{k_i\,k_j}{2\,m}\right)s_i\,s_j \quad , \quad B_{i\,j} = A_{i\,j} - \frac{k_i\,k_j}{2\,m}$$

$$Q = \frac{1}{4\,m}\,s^T\,B\,s = \frac{1}{4\,m}\sum_{i=1}^{n}\left(u_i^T\,.\,s\right)^2\,\beta_i$$

$$\Delta\,Q = \frac{1}{4\,m}\,s^T\,B^{(g)}\,s \quad , \quad B_{i\,j}^{(g)} = B_{i\,j} - \delta_{i\,j}\sum_{k\in g}B_{i\,k}$$

```
GirvanNewmanmodularity[xx_] :=
 Module[{network, m, Aij, B, u, β, s, Q, div1, div2, moditer, r1,
   r11, r12, r2, r21, r22, subfinal1Q, subfinal2Q, finalQ},
  network = xx;
  m = EdgeCount@network;
  Aij = AdjacencyMatrix@network;
  B = N@(Aij - Outer[Times, VertexDegree@network, VertexDegree@network] / (2 * m));
  u = Eigenvectors@B;
  β = Eigenvalues@B;
  s = (u /. x_ /; x < 0 → -1) /. x_ /; x ≥ 0 → 1;
  Q = (Total@(Table[(u[[i]].Flatten@s[[Flatten@Position[β, Max@β][[1]]]])^2,
        {i, VertexList@network}] * β)) / (4 * m);
  div1 = Flatten@Position[Flatten@s[[Flatten@Position[β, Max@β][[1]]]], _?Negative];
  div2 = Flatten@Position[Flatten@s[[Flatten@Position[β, Max@β][[1]]]], _?Positive];
  moditer[pos_, B1_] := Module[{Bg, ug, βg, sg, deltaQ, divv1, divv2, Bg1, Bg2},
    Bg = (Table[B1[[i, j]] - KroneckerDelta[i, j] * Total@B1[[i, pos]],
        {i, Length@B1}, {j, Length@B1}])[[pos, pos]];
    ug = Eigenvectors@Bg;
    βg = Eigenvalues@Bg;
    sg = (ug /. x_ /; x < 0 → -1) /. x_ /; x ≥ 0 → 1;
    deltaQ =
     (Total@(Table[(ug[[i]].Flatten@sg[[Flatten@Position[βg, Max@βg][[1]]]])^2,
           {i, Range@Length@pos}] * βg)) / (4 * m);
    divv1 = Flatten@Position[Flatten@sg[[Flatten@Position[βg, Max@βg][[1]]]],
       _?Negative];
    divv2 = Flatten@Position[Flatten@sg[[Flatten@Position[βg, Max@βg][[1]]]],
       _?Positive];
    {deltaQ, divv1, divv2, Bg}];
  r1 = moditer[div1, B];
  r11 = If[r1[[1]] > 0.001, moditer[r1[[2]], r1[[4]]], {0, 0, 0, 0}];
  r12 = If[r1[[1]] > 0.001, moditer[r1[[3]], r1[[4]]], {0, 0, 0, 0}];
  r2 = moditer[div2, B];
  r21 = If[r2[[1]] > 0.001, moditer[r2[[2]], r2[[4]]], {0, 0, 0, 0}];
  r22 = If[r2[[1]] > 0.001, moditer[r2[[3]], r2[[4]]], {0, 0, 0, 0}];
  subfinal1Q = If[r1[[1]] > 0.001, If[r11[[1]] > 0.001,
     If[r12[[1]] > 0.001, r12[[1]] + r11[[1]] + r1[[1]], r11[[1]] + r1[[1]]],
     If[r12[[1]] > 0.001, r12[[1]] + r1[[1]], r1[[1]]]], 0];
  subfinal2Q = If[r2[[1]] > 0.001, If[r21[[1]] > 0.001,
     If[r22[[1]] > 0.001, r22[[1]] + r21[[1]] + r2[[1]], r21[[1]] + r2[[1]]],
     If[r22[[1]] > 0.001, r22[[1]] + r2[[1]], r2[[1]]]], 0];
  finalQ = Q + subfinal1Q + subfinal2Q]
```

Assortativity Modularity (Wolfram)

$$Q = \frac{1}{2\,m} \sum_{i\,j} \left(A_{i\,j} - \frac{k_i\,k_j}{2\,m} \right) \delta_{C_i\,C_j}$$

where m is the number of edges,
A is the adjacency matrix,
k_i is the degree of node i,
c_i is the community that node i belongs to,
$\delta_{i\,j}$ is the Kronecker δ symbol.

For weighted graphs,
A is the weighted adjacency matrix,
k_i are the sum of weights of edges incident on node i,
m is the sum of all weights.

```
GraphAssortativity[graph, communities, "Normalized" → False]
```

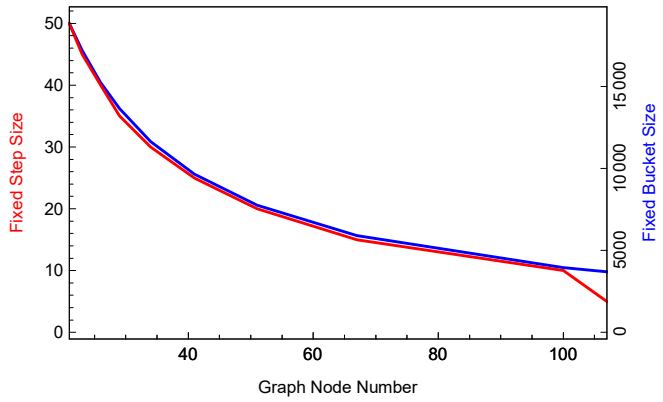
Modularity Change Across Varying Step & Bucket Sizes

```
In[ ]:= AbsoluteTiming[widthdatavaryingFixedstep =
          Table[snetworkdatabinned[9, i, "datafull"], {i, Range[5, 50, 5]}];]
        graphsandnodenumbers = Table[snetworkgraph[widthdatavaryingFixedstep[[i]][[1]],
           widthdatavaryingFixedstep[[i]][[2]], 2, 7, 400, Green], {i, Range@10}];
        GNmodularityvalues = Table[GirvanNewmanmodularity[graphsandnodenumbers[[i]][[1]]],
          {i, Length@graphsandnodenumbers}];
        modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
            FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
            "Normalized" → False], {i, Length@graphsandnodenumbers}];

Out[ ]= {302.106, Null}

In[ ]:= AbsoluteTiming[widthdatavaryingFixedbucket =
          Table[snetworkdatafxdbucket[9, i, "datafull"], {i, graphsandnodenumbers[[All, 2]]}];]
        graphsandnodenumbersFixedbucket = Table[snetworkgraph[widthdatavaryingFixedbucket[[i]][[
            1]], widthdatavaryingFixedbucket[[i]][[2]], 2, 7, 400, Green], {i, Range@10}];
        GNmodularityvaluesFixedbucket = Table[GirvanNewmanmodularity[
           graphsandnodenumbersFixedbucket[[i]][[1]]],
          {i, Length@graphsandnodenumbersFixedbucket}];
        modularityvaluesFixedbucket = Table[
          N@GraphAssortativity[graphsandnodenumbersFixedbucket[[i]][[1]],
            FindGraphCommunities[graphsandnodenumbersFixedbucket[[i]][[1]]],
            "Normalized" → False], {i, Length@graphsandnodenumbersFixedbucket}];

Out[ ]= {120.951, Null}

In[ ]:= stepamounts = Range[5, 50, 5]
        nodenumbers = graphsandnodenumbers[[All, 2]]
        bucketamounts = Table[(Values@Counts@widthdatavaryingFixedbucket[[i, 1]])[[1]], {i, 10}]

Out[ ]= {5, 10, 15, 20, 25, 30, 35, 40, 45, 50}

Out[ ]= {107, 100, 67, 51, 41, 34, 29, 26, 23, 21}
```

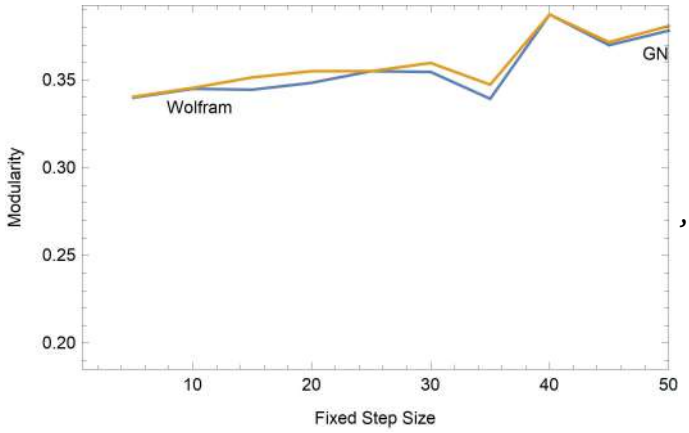
Out[•]= {3702, 3961, 5912, 7767, 9661, 11 650, 13 659, 15 235, 17 222, 18 862}

In[•]:=
```
tickvalues = {{0, 0}, {5000, 5000}, {10 000, 10 000}, {15 000, 15 000}};
ticks = MapAt[Rotate[#, Pi / 2] &, tickvalues, {All, -1}];
{Overlay[{ListLinePlot[Thread[{nodenumbers, bucketamounts}], Frame → True,
    ImagePadding → 35, FrameTicks → {{None, ticks}, {All, None}}, FrameLabel →
     {{None, Style["Fixed Bucket Size", Blue]}, {None, None}}, PlotStyle → Blue,
    ImageSize → 350, PlotRange → {{First@nodenumbers, Last@nodenumbers}, All}],
   ListLinePlot[Thread[{nodenumbers, stepamounts}], Frame → True,
    ImagePadding → 35, FrameTicks → {{All, None}, {All, None}},
    FrameLabel → {"Graph Node Number", Style["Fixed Step Size", Red]}, PlotStyle → Red,
    ImageSize → 350, PlotRange → {{First@nodenumbers, Last@nodenumbers}, All}]}],
 ListLinePlot[{Thread[{stepamounts, GNmodularityvalues}],
   Thread[{stepamounts, modularityvalues}]}, Frame → True,
  FrameLabel → {"Fixed Step Size", "Modularity"}, ImageSize → 350,
  PlotRange → {{First@stepamounts, Last@stepamounts},
    {Min[(MinMax@GNmodularityvalues)[[1]], (MinMax@GNmodularityvaluesFixedbucket)[[
        1]]] - 0.005, Max[(MinMax@GNmodularityvalues)[[2]],
       (MinMax@GNmodularityvaluesFixedbucket)[[2]]] + 0.005}},
  PlotLabels → Placed[{"GN", "Wolfram"}, {Scaled[1], Below}]](*,
 ListLinePlot[Thread[{Range[5,50,5],modularityvalues}],Frame→True,
  FrameLabel→{"Fixed Step Size","Modularity (Wolfram)"},
  ImageSize→350,PlotRange→Automatic]*),
 ListLinePlot[{Thread[{bucketamounts, GNmodularityvaluesFixedbucket}],
   Thread[{bucketamounts, modularityvaluesFixedbucket}]}, Frame → True,
  FrameLabel → {"Fixed Bucket Size", "Modularity"}, ImageSize → 350,
  PlotRange → {{First@bucketamounts, Last@bucketamounts},
    {Min[(MinMax@GNmodularityvalues)[[1]], (MinMax@GNmodularityvaluesFixedbucket)[[
        1]]] - 0.005, Max[(MinMax@GNmodularityvalues)[[2]],
       (MinMax@GNmodularityvaluesFixedbucket)[[2]]] + 0.005}},
  PlotLabels → Placed[{"GN", "Wolfram"}, {Scaled[1], Above}]](*,
 ListLinePlot[Thread[{graphsandnodenumbers[[All,2]],modularityvaluesFixedbucket}],
  Frame→True,FrameLabel→{"Fixed Bucket Size","Modularity (Wolfram)"},
  ImageSize→350,PlotRange→Automatic]*)}
```

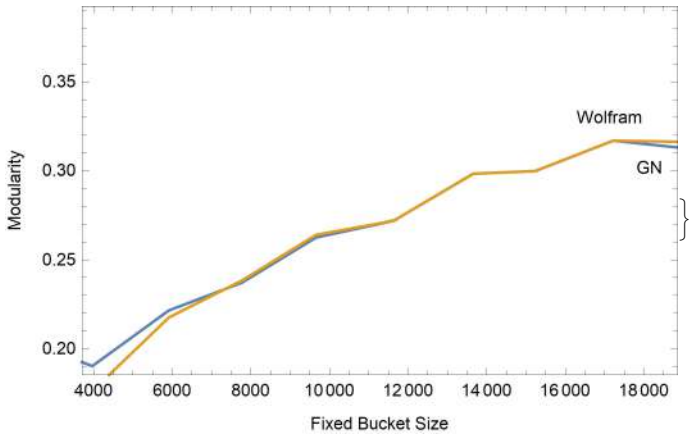
Out[•]=


Fixed Step Size
Fixed Bucket Size
Graph Node Number

,


Modularity
Wolfram
GN
Fixed Step Size

,


Modularity
Wolfram
GN
Fixed Bucket Size

In[•]:=
```
AbsoluteTiming[thicknessdatavaryingFixedstep =
   Table[snetworkdatabinned[10, i, "datafull"], {i, Range@10}];]
graphsandnodenumbers = Table[snetworkgraph[thicknessdatavaryingFixedstep[[i]][[1]],
    thicknessdatavaryingFixedstep[[i]][[2]], 1.5,
    7, 400, RGBColor[0.1, 0.5, 1.]], {i, Range@10}];
GNmodularityvalues = Table[GirvanNewmanmodularity[graphsandnodenumbers[[i]][[1]]],
   {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
     FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
     "Normalized" → False], {i, Length@graphsandnodenumbers}];
```

Out[•]= {93.5853, Null}

In[•]:=
```
AbsoluteTiming[thicknessdatavaryingFixedbucket =
   Table[snetworkdatafxdbucket[10, i, "datafull"], {i, graphsandnodenumbers[[All, 2]]}];]
graphsandnodenumbersFixedbucket = Table[snetworkgraph[
    thicknessdatavaryingFixedbucket[[i]][[1]], thicknessdatavaryingFixedbucket[[i]][[
     2]], 1.5, 7, 400, RGBColor[0.1, 0.5, 1.]], {i, Range@10}];
GNmodularityvaluesFixedbucket = Table[GirvanNewmanmodularity[
    graphsandnodenumbersFixedbucket[[i]][[1]]],
   {i, Length@graphsandnodenumbersFixedbucket}];
modularityvaluesFixedbucket = Table[
   N@GraphAssortativity[graphsandnodenumbersFixedbucket[[i]][[1]],
     FindGraphCommunities[graphsandnodenumbersFixedbucket[[i]][[1]]],
     "Normalized" → False], {i, Length@graphsandnodenumbersFixedbucket}];
```

Out[•]= {818.97, Null}

In[•]:=
```
stepamounts = Range@10
nodenumbers = graphsandnodenumbers[[All, 2]]
bucketamounts =
 Table[(Values@Counts@thicknessdatavaryingFixedbucket[[i, 1]])[[1]], {i, 10}]
```

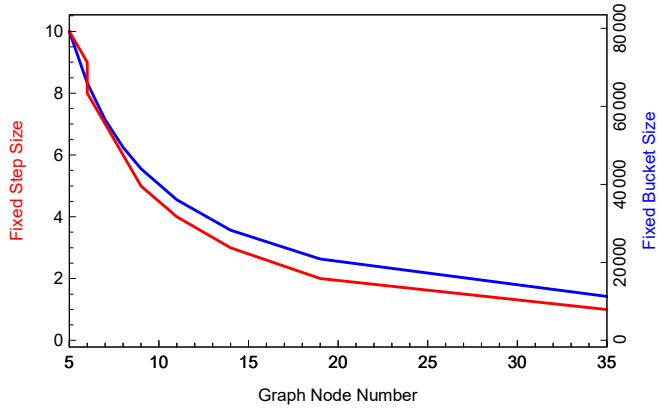
Out[•]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Out[•]= {35, 19, 14, 11, 9, 8, 7, 6, 6, 5}

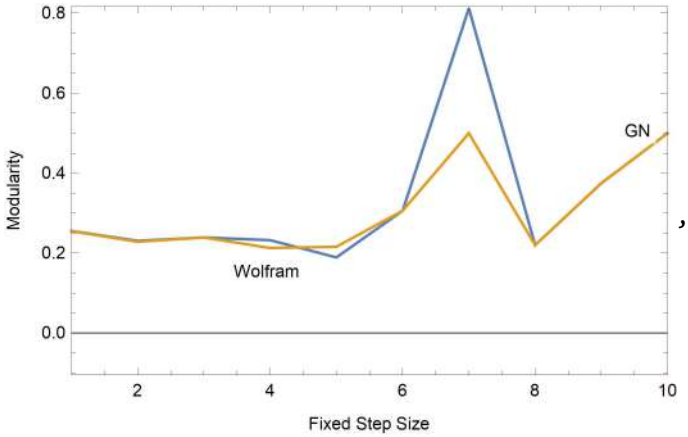
Out[•]= {11 284, 20 832, 28 287, 36 006, 44 008, 49 512, 56 580, 66 016, 66 016, 79 216}

In[ ]:=
```
tickvalues = {{0, 0}, {20 000, 20 000}, {40 000, 40 000}, {60 000, 60 000}, {80 000, 80 000}};
ticks = MapAt[Rotate[#, Pi / 2] &, tickvalues, {All, -1}];
{Overlay[{ListLinePlot[Thread[{nodenumbers, bucketamounts}], Frame → True,
    ImagePadding → 35, FrameTicks → {{None, ticks}, {All, None}}, FrameLabel →
     {{None, Style["Fixed Bucket Size", Blue]}, {None, None}}, PlotStyle → Blue,
    ImageSize → 350, PlotRange → {{First@nodenumbers, Last@nodenumbers}, All}],
   ListLinePlot[Thread[{nodenumbers, stepamounts}], Frame → True,
    ImagePadding → 35, FrameTicks → {{All, None}, {All, None}},
    FrameLabel → {"Graph Node Number", Style["Fixed Step Size", Red]}, PlotStyle → Red,
    ImageSize → 350, PlotRange → {{First@nodenumbers, Last@nodenumbers}, All}]}],
 ListLinePlot[{Thread[{stepamounts, GNmodularityvalues}],
   Thread[{stepamounts, modularityvalues}]}, Frame → True,
  FrameLabel → {"Fixed Step Size", "Modularity"}, ImageSize → 350,
  PlotRange → {{First@stepamounts, Last@stepamounts},
    {Min[(MinMax@GNmodularityvalues)[[1]], (MinMax@GNmodularityvaluesFixedbucket)[[
        1]]] - 0.005, Max[(MinMax@GNmodularityvalues)[[2]],
       (MinMax@GNmodularityvaluesFixedbucket)[[2]]] + 0.005}},
  PlotLabels → Placed[{"GN", "Wolfram"}, {Scaled[1], Below}]],
 ListLinePlot[{Thread[{bucketamounts, GNmodularityvaluesFixedbucket}],
   Thread[{bucketamounts, modularityvaluesFixedbucket}]}, Frame → True,
  FrameLabel → {"Fixed Bucket Size", "Modularity"}, ImageSize → 350,
  PlotRange → {{First@bucketamounts, Last@bucketamounts},
    {Min[(MinMax@GNmodularityvalues)[[1]], (MinMax@GNmodularityvaluesFixedbucket)[[
        1]]] - 0.005, Max[(MinMax@GNmodularityvalues)[[2]],
       (MinMax@GNmodularityvaluesFixedbucket)[[2]]] + 0.005}},
  PlotLabels → Placed[{"GN", "Wolfram"}, {Scaled[1], Above}]]}
```

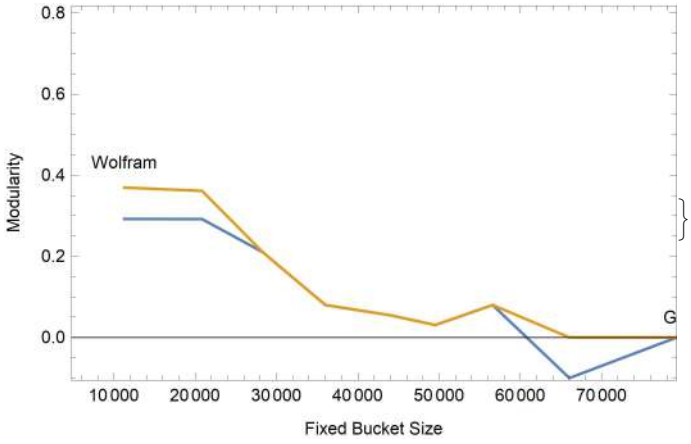
Out[•]=


Fixed Step Size
Fixed Bucket Size
Graph Node Number

,


Modularity
GN
Wolfram
Fixed Step Size

,


Modularity
Wolfram
Fixed Bucket Size

In[•]:=
```
AbsoluteTiming[lengthdatavaryingFixedstep =
   Table[snetworkdatabinned[12, i, "datafull"], {i, Range[400, 4000, 400]}];]
graphsandnodenumbers = Table[snetworkgraph[lengthdatavaryingFixedstep[[i]][[1]],
    lengthdatavaryingFixedstep[[i]][[2]], 2, 7,
    400, RGBColor[1., 0.53, 0.2]], {i, Range@10}];
GNmodularityvalues = Table[GirvanNewmanmodularity[graphsandnodenumbers[[i]][[1]]],
   {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
     FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
     "Normalized" → False], {i, Length@graphsandnodenumbers}];
```

Out[•]= {186.879, Null}

In[•]:=
```
AbsoluteTiming[lengthdatavaryingFixedbucket =
   Table[snetworkdatafxdbucket[12, i, "datafull"], {i, graphsandnodenumbers[[All, 2]]}];]
graphsandnodenumbersFixedbucket = Table[snetworkgraph[
    lengthdatavaryingFixedbucket[[i]][[1]], lengthdatavaryingFixedbucket[[i]][[2]],
    2, 7, 400, RGBColor[1., 0.53, 0.2]], {i, Range@10}];
GNmodularityvaluesFixedbucket = Table[GirvanNewmanmodularity[
    graphsandnodenumbersFixedbucket[[i]][[1]]],
   {i, Length@graphsandnodenumbersFixedbucket}];
modularityvaluesFixedbucket = Table[
   N@GraphAssortativity[graphsandnodenumbersFixedbucket[[i]][[1]],
     FindGraphCommunities[graphsandnodenumbersFixedbucket[[i]][[1]]],
     "Normalized" → False], {i, Length@graphsandnodenumbersFixedbucket}];
```

Out[•]= {89.5098, Null}

In[•]:=
```
stepamounts = Range[400, 4000, 400]
nodenumbers = graphsandnodenumbers[[All, 2]]
bucketamounts =
 Table[(Values@Counts@lengthdatavaryingFixedbucket[[i, 1]])[[1]], {i, 10}]
```

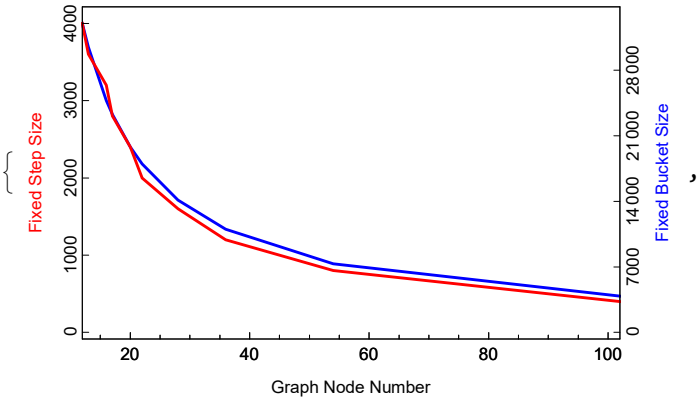
Out[•]= {400, 800, 1200, 1600, 2000, 2400, 2800, 3200, 3600, 4000}

Out[•]= {102, 54, 36, 28, 22, 20, 17, 16, 13, 12}

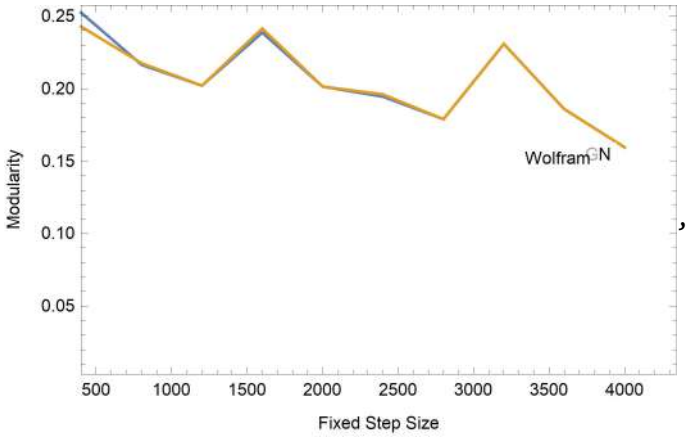
Out[•]= {3884, 7336, 11 003, 14 147, 18 005, 19 805, 23 300, 24 756, 30 469, 33 008}

In[ ]:=
```
tickvalues =
  {{0, 0}, {7000, 7000}, {14000, 14000}, {21000, 21000}, {28000, 28000}, {35000, 35000}};
ticks = MapAt[Rotate[#, Pi / 2] &, tickvalues, {All, -1}];
tickvaluesst = {{0, 0}, {1000, 1000}, {2000, 2000}, {3000, 3000}, {4000, 4000}};
ticksst = MapAt[Rotate[#, Pi / 2] &, tickvaluesst, {All, -1}];
{Overlay[{ListLinePlot[Thread[{nodenumbers, bucketamounts}], Frame → True,
    ImagePadding → 35, FrameTicks → {{None, ticks}, {All, None}}, FrameLabel →
     {{None, Style["Fixed Bucket Size", Blue]}, {None, None}}, PlotStyle → Blue,
    ImageSize → 350, PlotRange → {{First@nodenumbers, Last@nodenumbers}, All}],
   ListLinePlot[Thread[{nodenumbers, stepamounts}], Frame → True,
    ImagePadding → 35, FrameTicks → {{ticksst, None}, {All, None}},
    FrameLabel → {"Graph Node Number", Style["Fixed Step Size", Red]}, PlotStyle → Red,
    ImageSize → 350, PlotRange → {{First@nodenumbers, Last@nodenumbers}, All}]}],
 ListLinePlot[{Thread[{stepamounts, GNmodularityvalues}],
   Thread[{stepamounts, modularityvalues}]}, Frame → True,
  FrameLabel → {"Fixed Step Size", "Modularity"}, ImageSize → 350,
  PlotRange → {{First@stepamounts, Last@stepamounts},
    {Min[(MinMax@GNmodularityvalues)[[1]], (MinMax@GNmodularityvaluesFixedbucket)[[
        1]]] - 0.005, Max[(MinMax@GNmodularityvalues)[[2]],
       (MinMax@GNmodularityvaluesFixedbucket)[[2]]] + 0.005}},
  PlotLabels → Placed[{"GN", "Wolfram"}, {Scaled[1], Below}]],
 ListLinePlot[{Thread[{bucketamounts, GNmodularityvaluesFixedbucket}],
   Thread[{bucketamounts, modularityvaluesFixedbucket}]}, Frame → True,
  FrameLabel → {"Fixed Bucket Size", "Modularity"}, ImageSize → 350,
  PlotRange → {{First@bucketamounts, Last@bucketamounts},
    {Min[(MinMax@GNmodularityvalues)[[1]], (MinMax@GNmodularityvaluesFixedbucket)[[
        1]]] - 0.005, Max[(MinMax@GNmodularityvalues)[[2]],
       (MinMax@GNmodularityvaluesFixedbucket)[[2]]] + 0.005}},
  PlotLabels → Placed[{"GN", "Wolfram"}, {Scaled[1], Above}]]}
```

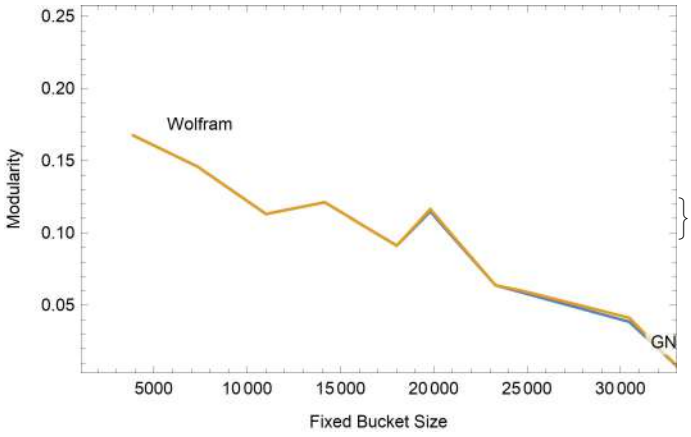
Out[•]=


Fixed Step Size
Fixed Bucket Size
Graph Node Number
0
1000
2000
3000
4000
7000
14000
21000
28000
20
40
60
80
100

,


Modularity
Fixed Step Size
Wolfram
GN
0.05
0.10
0.15
0.20
0.25
500
1000
1500
2000
2500
3000
3500
4000

,


Modularity
Fixed Bucket Size
Wolfram
GN
0.05
0.10
0.15
0.20
0.25
5000
10000
15000
20000
25000
30000

In[ ]:=
```
AbsoluteTiming[weightdatavaryingFixedstep =
   Table[snetworkdatabinned[11, i, "datafull"], {i, Range[200, 2000, 200]}];]
graphsandnodenumbers = Table[snetworkgraph[weightdatavaryingFixedstep[[i]][[1]],
    weightdatavaryingFixedstep[[i]][[2]], 2, 7, 400, Red], {i, Range@10}];
GNmodularityvalues = Table[GirvanNewmanmodularity[graphsandnodenumbers[[i]][[1]]],
   {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
     FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
     "Normalized" → False], {i, Length@graphsandnodenumbers}];
```

Out[ ]= {227.761, Null}

In[ ]:=
```
AbsoluteTiming[weightdatavaryingFixedbucket =
   Table[snetworkdatafxdbucket[12, i, "datafull"], {i, graphsandnodenumbers[[All, 2]]}];]
graphsandnodenumbersFixedbucket = Table[snetworkgraph[weightdatavaryingFixedbucket[[i]][[
     1]], weightdatavaryingFixedbucket[[i]][[2]], 2, 7, 400, Red], {i, Range@10}];
GNmodularityvaluesFixedbucket = Table[GirvanNewmanmodularity[
    graphsandnodenumbersFixedbucket[[i]][[1]]],
   {i, Length@graphsandnodenumbersFixedbucket}];
modularityvaluesFixedbucket = Table[
   N@GraphAssortativity[graphsandnodenumbersFixedbucket[[i]][[1]],
     FindGraphCommunities[graphsandnodenumbersFixedbucket[[i]][[1]]],
     "Normalized" → False], {i, Length@graphsandnodenumbersFixedbucket}];
```

Out[ ]= {91.914, Null}

In[ ]:=
```
stepamounts = Range[200, 2000, 200]
nodenumbers = graphsandnodenumbers[[All, 2]]
bucketamounts =
 Table[(Values@Counts@weightdatavaryingFixedbucket[[i, 1]])[[1]], {i, 10}]
```

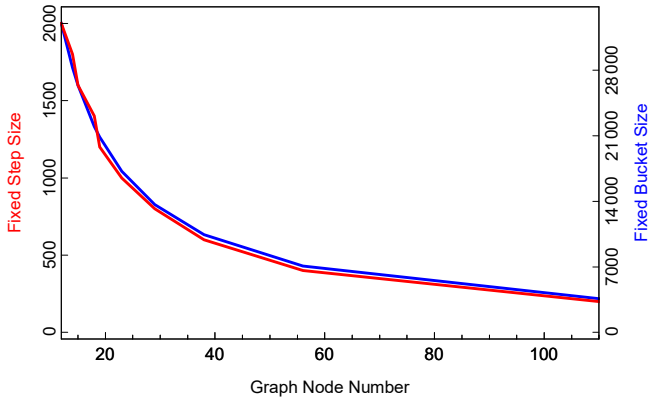
Out[ ]= {200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000}

Out[ ]= {110, 56, 38, 29, 23, 19, 18, 15, 14, 12}

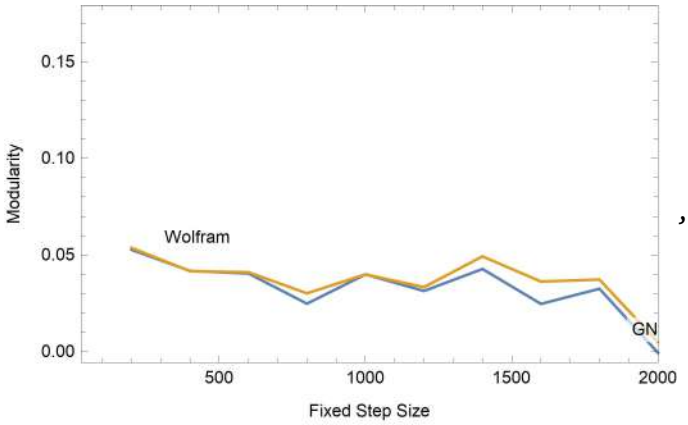
Out[ ]= {3601, 7074, 10 424, 13 659, 17 222, 20 848, 22 006, 26 407, 28 293, 33 008}

In[ ]:=
```
tickvalues =
  {{0, 0}, {7000, 7000}, {14000, 14000}, {21000, 21000}, {28000, 28000}, {35000, 35000}};
ticks = MapAt[Rotate[#, Pi / 2] &, tickvalues, {All, -1}];
tickvaluesst = {{0, 0}, {500, 500}, {1000, 1000}, {1500, 1500}, {2000, 2000}};
tickssst = MapAt[Rotate[#, Pi / 2] &, tickvaluesst, {All, -1}];
{Overlay[{ListLinePlot[Thread[{nodenumbers, bucketamounts}], Frame → True,
    ImagePadding → 35, FrameTicks → {{None, ticks}, {All, None}}, FrameLabel →
     {{None, Style["Fixed Bucket Size", Blue]}, {None, None}}, PlotStyle → Blue,
    ImageSize → 350, PlotRange → {{First@nodenumbers, Last@nodenumbers}, All}],
   ListLinePlot[Thread[{nodenumbers, stepamounts}], Frame → True,
    ImagePadding → 35, FrameTicks → {{tickssst, None}, {All, None}},
    FrameLabel → {"Graph Node Number", Style["Fixed Step Size", Red]}, PlotStyle → Red,
    ImageSize → 350, PlotRange → {{First@nodenumbers, Last@nodenumbers}, All}]}],
 ListLinePlot[{Thread[{stepamounts, GNmodularityvalues}],
   Thread[{stepamounts, modularityvalues}]}, Frame → True,
  FrameLabel → {"Fixed Step Size", "Modularity"}, ImageSize → 350,
  PlotRange → {{First@stepamounts, Last@stepamounts},
    {Min[(MinMax@GNmodularityvalues)[[1]], (MinMax@GNmodularityvaluesFixedbucket)[[
        1]]] - 0.005, Max[(MinMax@GNmodularityvalues)[[2]],
       (MinMax@GNmodularityvaluesFixedbucket)[[2]]] + 0.005}},
  PlotLabels → Placed[{"GN", "Wolfram"}, {Scaled[1], Above}]],
 ListLinePlot[{Thread[{bucketamounts, GNmodularityvaluesFixedbucket}],
   Thread[{bucketamounts, modularityvaluesFixedbucket}]}, Frame → True,
  FrameLabel → {"Fixed Bucket Size", "Modularity"}, ImageSize → 350,
  PlotRange → {{First@bucketamounts, Last@bucketamounts},
    {Min[(MinMax@GNmodularityvalues)[[1]], (MinMax@GNmodularityvaluesFixedbucket)[[
        1]]] - 0.005, Max[(MinMax@GNmodularityvalues)[[2]],
       (MinMax@GNmodularityvaluesFixedbucket)[[2]]] + 0.005}},
  PlotLabels → Placed[{"GN", "Wolfram"}, {Scaled[1], Top}]]}
```

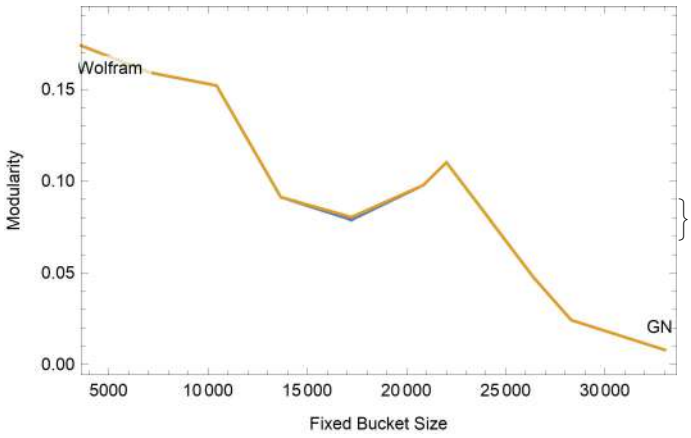
Out[ ]=



,



,

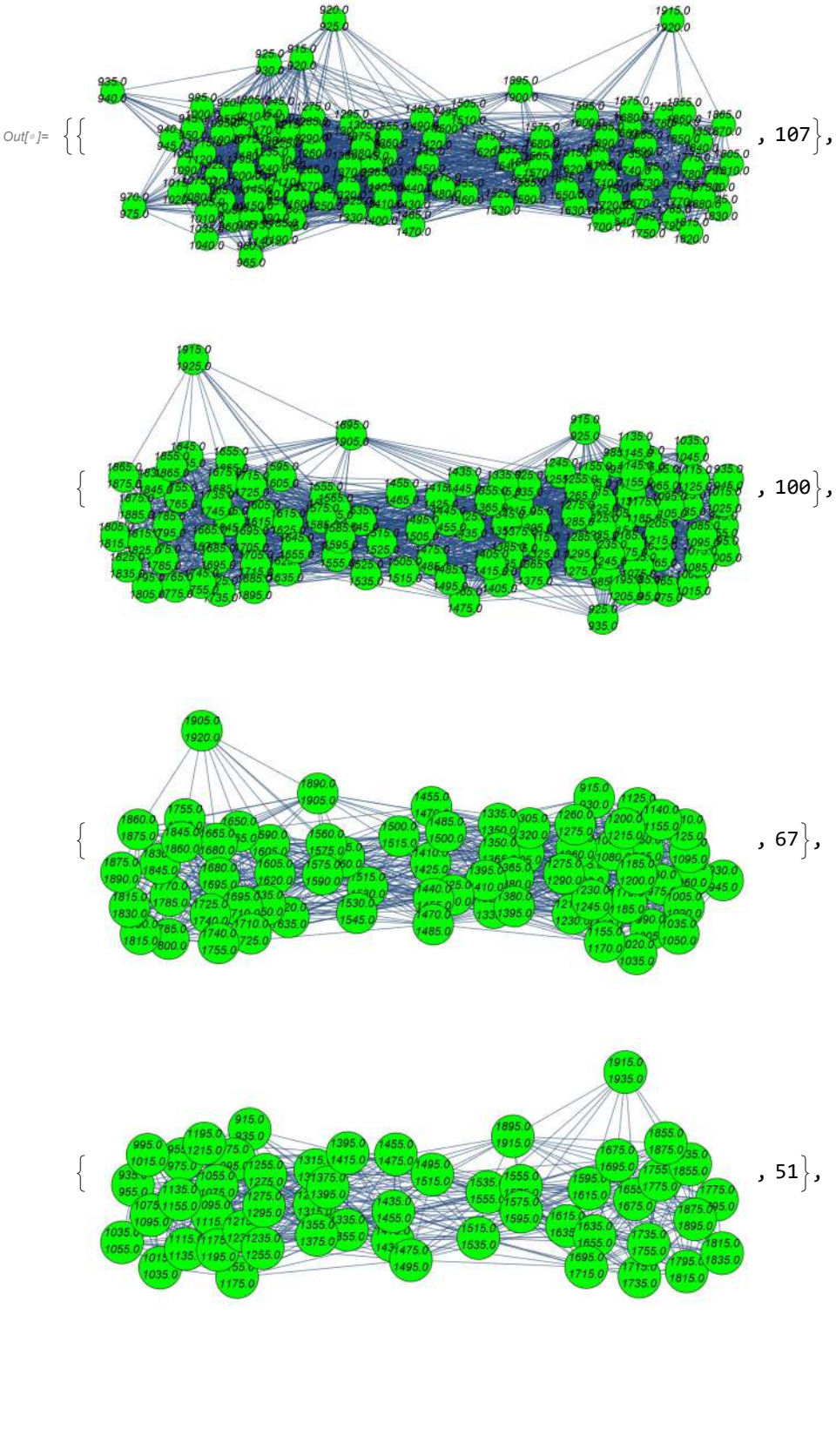


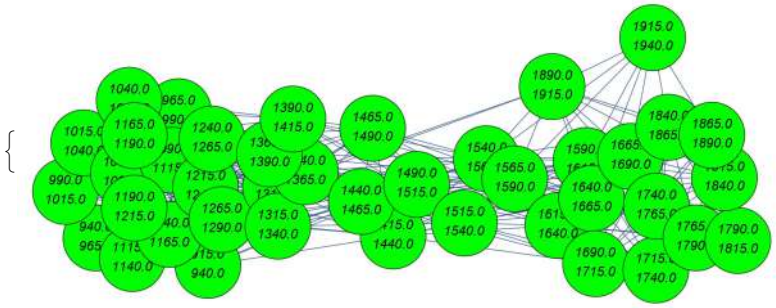
}

Width Graphs for Varying Fixed Step Sizes and Fixed Bucket Sizes

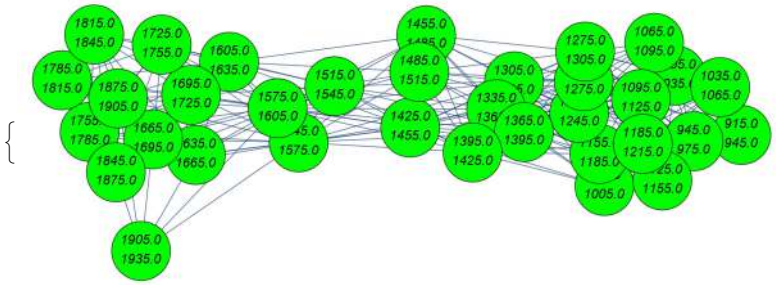
In[ ]:= **graphsandnodenumbers**

Out[•]=

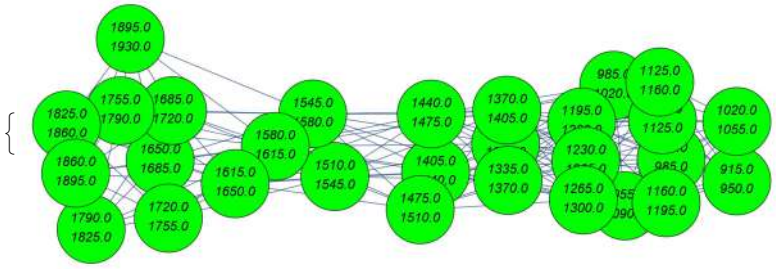



1915.0
1940.0
1890.0
1915.0
1040.0
1165.0
1190.0
1240.0
1265.0
1390.0
1415.0
1465.0
1490.0
1840.0
1865.0
1890.0
1015.0
990.0
1015.0
1190.0
1215.0
1265.0
1290.0
1315.0
1340.0
1440.0
1465.0
1490.0
1515.0
1540.0
1565.0
1590.0
1640.0
1665.0
1690.0
1715.0
1740.0
1765.0
1790.0
1815.0
940.0
1140.0

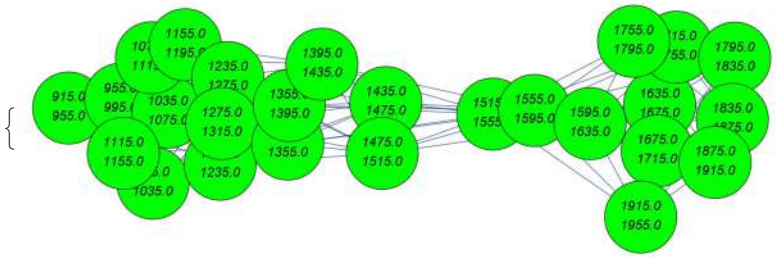
, 41},


1815.0
1845.0
1725.0
1755.0
1605.0
1635.0
1785.0
1815.0
1875.0
1905.0
1695.0
1725.0
1575.0
1605.0
1515.0
1545.0
1455.0
1485.0
1515.0
1425.0
1455.0
1305.0
1335.0
1365.0
1395.0
1425.0
1275.0
1305.0
1065.0
1095.0
1035.0
1065.0
1245.0
1125.0
1185.0
1215.0
945.0
975.0
915.0
945.0
1005.0
1155.0
1905.0
1935.0

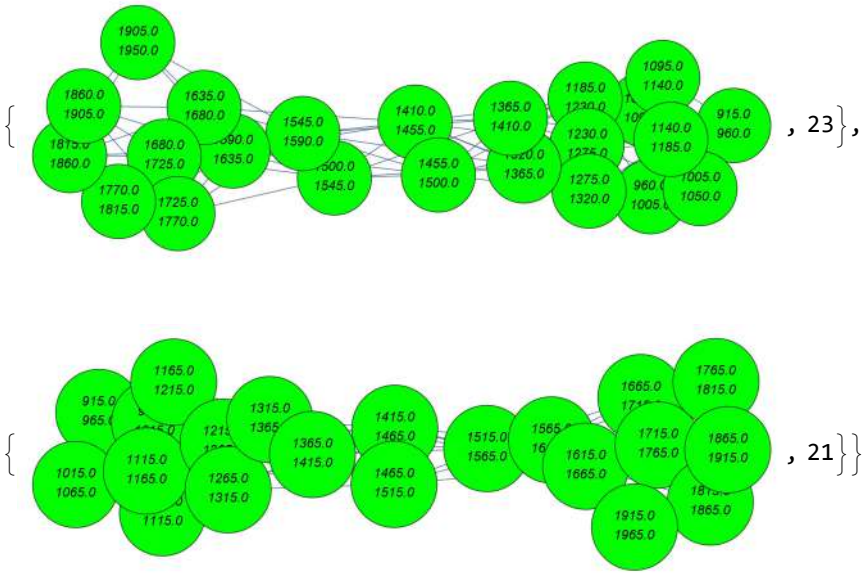
, 34},


1895.0
1930.0
1825.0
1860.0
1755.0
1790.0
1685.0
1720.0
1650.0
1685.0
1860.0
1895.0
1790.0
1825.0
1720.0
1755.0
1580.0
1615.0
1545.0
1580.0
1615.0
1650.0
1510.0
1545.0
1440.0
1475.0
1370.0
1405.0
1405.0
1440.0
1475.0
1510.0
1335.0
1370.0
1195.0
1230.0
1265.0
1300.0
985.0
1020.0
1125.0
1160.0
1125.0
1160.0
1195.0
1020.0
1055.0
915.0
950.0

, 29},

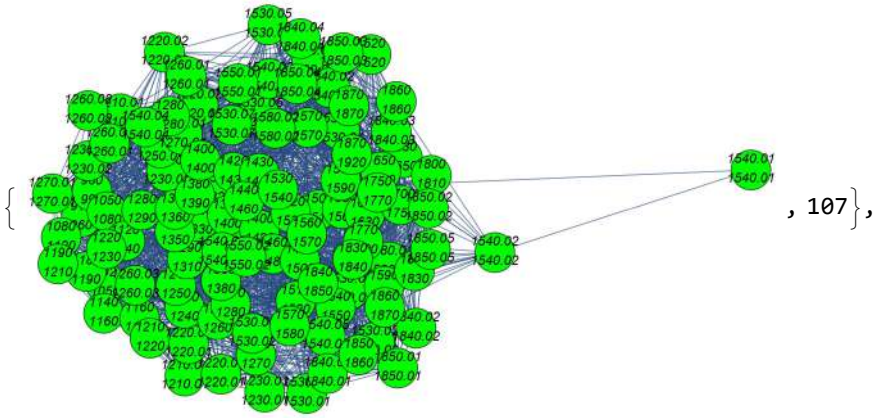

1155.0
1195.0
1235.0
1275.0
1315.0
1355.0
1395.0
1435.0
1035.0
1075.0
1115.0
1155.0
1235.0
1275.0
1355.0
1395.0
1435.0
1475.0
1475.0
1515.0
1515.0
1555.0
1555.0
1595.0
1595.0
1635.0
1675.0
1715.0
1755.0
1795.0
1795.0
1835.0
1635.0
1675.0
1835.0
1875.0
1915.0
1915.0
1955.0

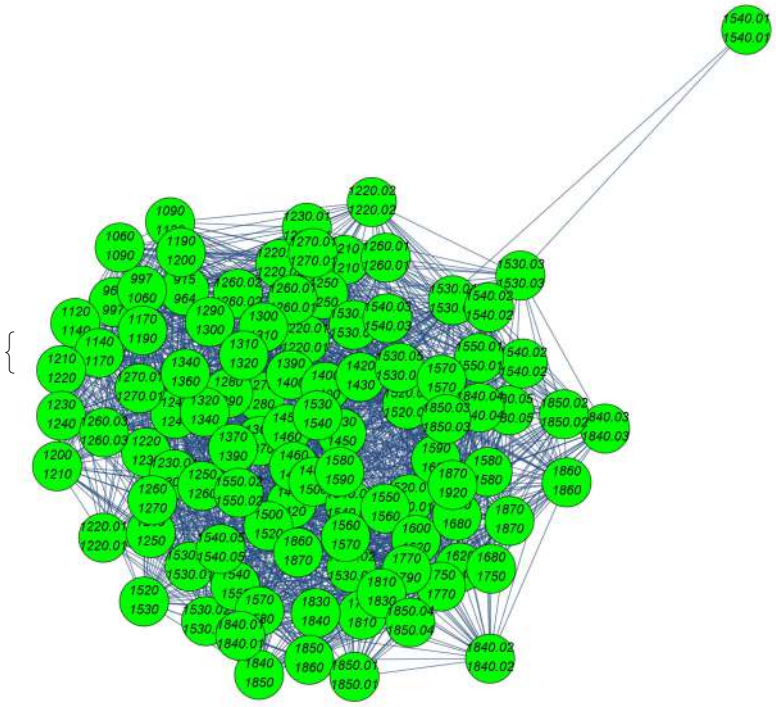
, 26},



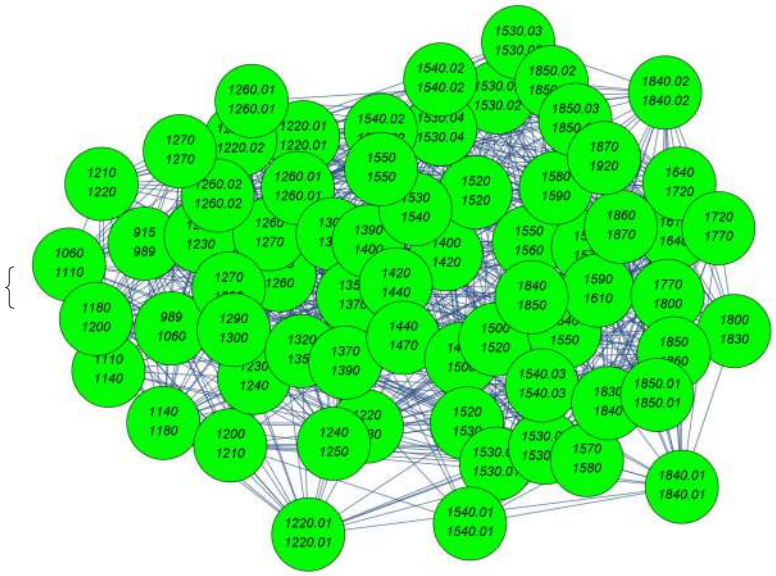
In[ ]:= **graphsandnodenumbersFixedbucket**

Out[ ]=

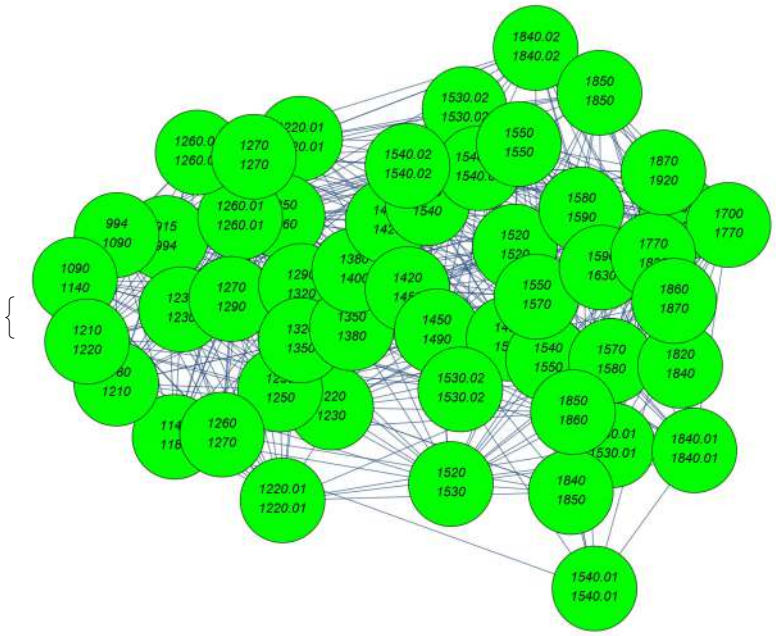


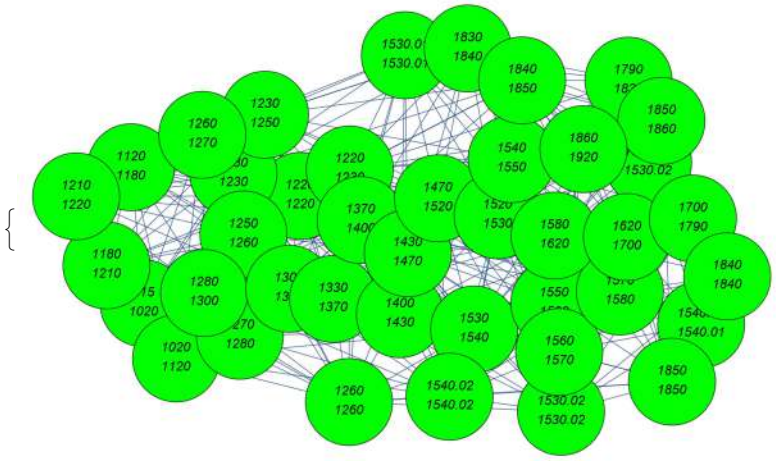


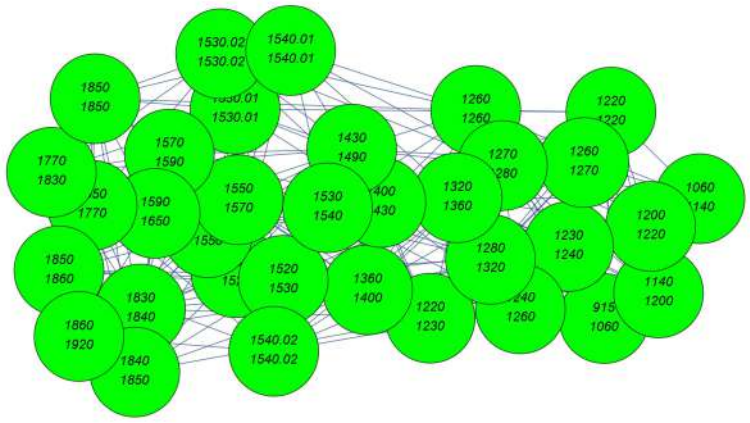
, 100},

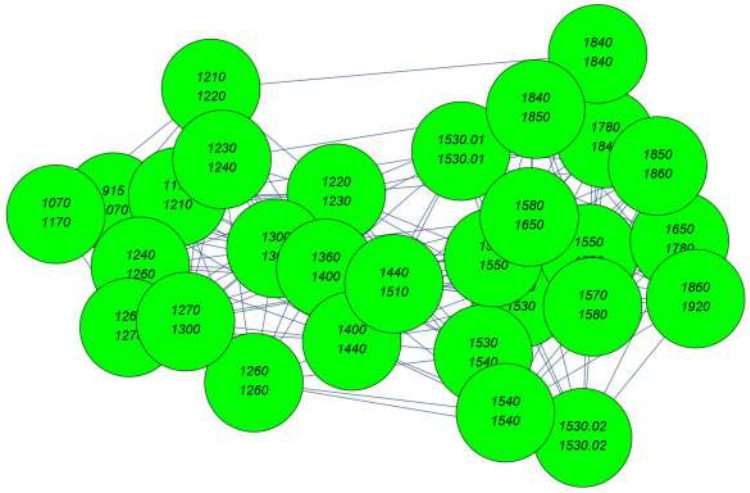


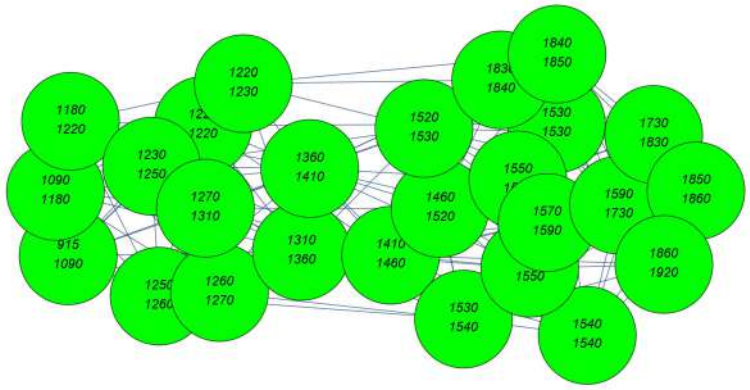
, 67},

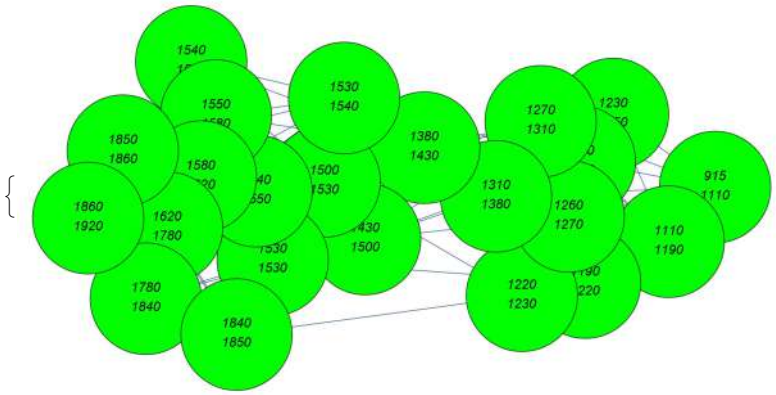
{  , 51},

{  , 41},

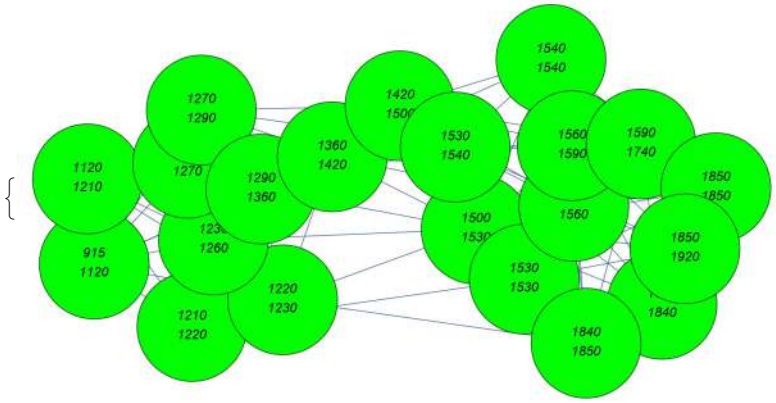
{  , 34},

{  , 29},

{  , 26},



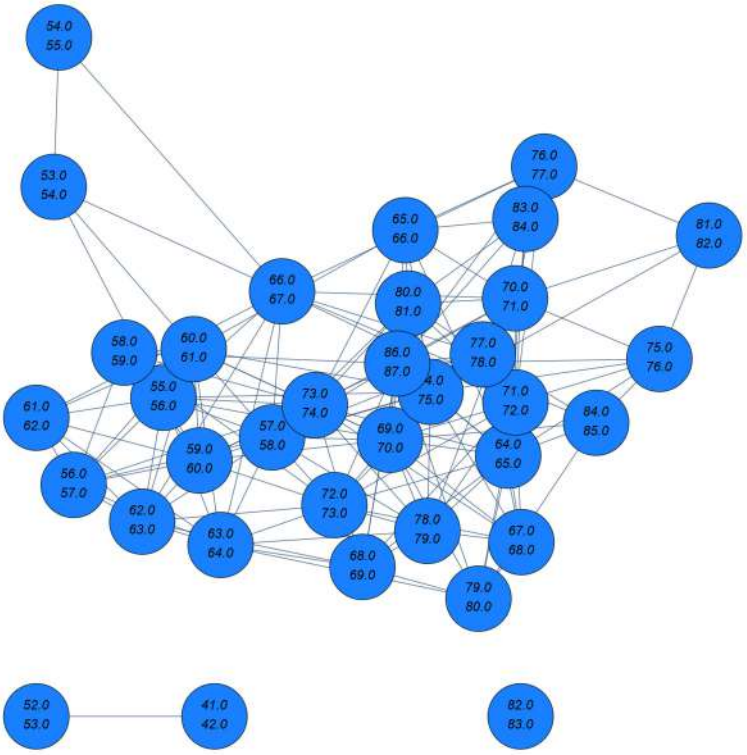
, 23},

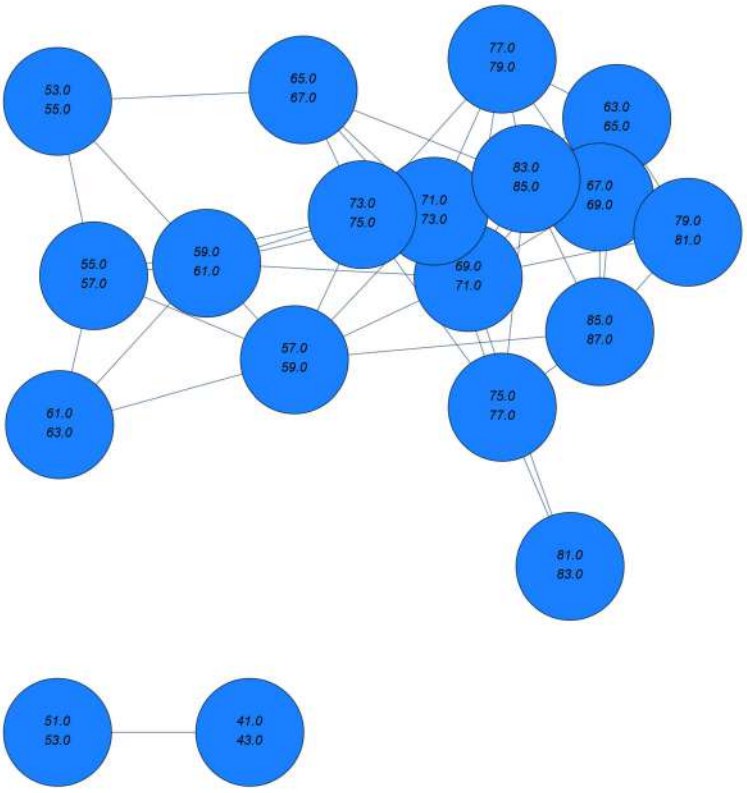


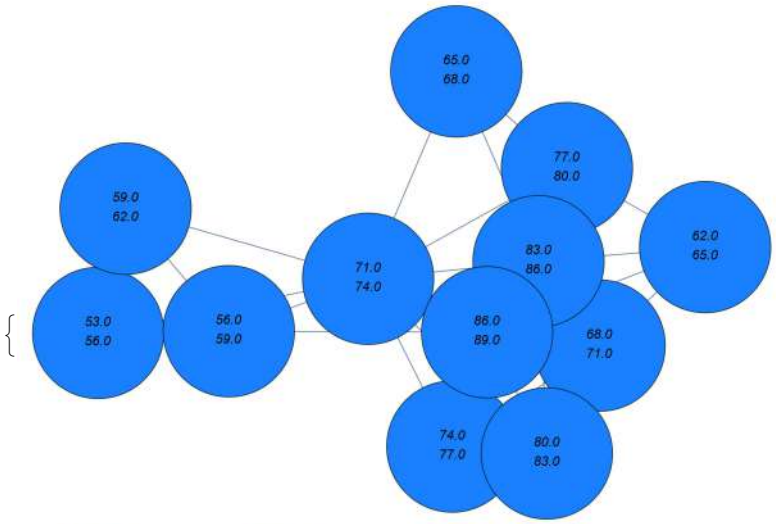
, 21}}

Thickness Graphs for Varying Fixed Step Sizes and Fixed Bucket Sizes

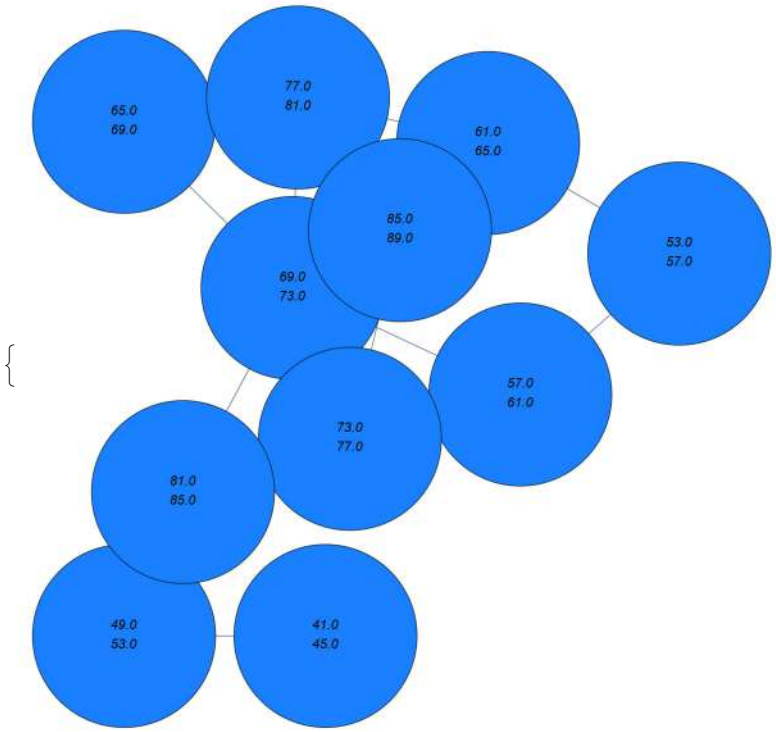
In[ ]:= **graphsandnodenumbers**

Out[•]= {{ 
54.0
55.0
53.0
54.0
76.0
77.0
83.0
84.0
65.0
66.0
81.0
82.0
66.0
67.0
80.0
81.0
70.0
71.0
58.0
59.0
60.0
61.0
86.0
87.0
77.0
78.0
75.0
76.0
55.0
56.0
73.0
74.0
4.0
75.0
71.0
72.0
84.0
85.0
61.0
62.0
57.0
58.0
69.0
70.0
64.0
65.0
59.0
60.0
56.0
57.0
72.0
73.0
62.0
63.0
78.0
79.0
67.0
68.0
63.0
64.0
68.0
69.0
79.0
80.0
52.0
53.0
41.0
42.0
82.0
83.0
, 35},

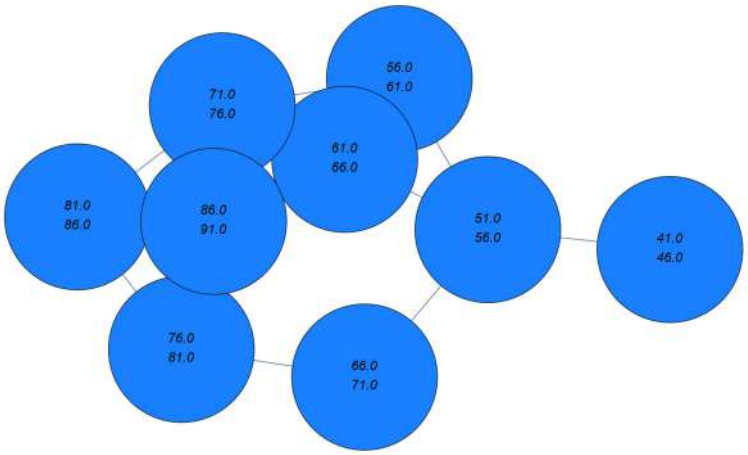
{ 
77.0
79.0
53.0
55.0
65.0
67.0
63.0
65.0
83.0
85.0
73.0
75.0
71.0
73.0
67.0
69.0
79.0
81.0
55.0
57.0
59.0
61.0
69.0
71.0
85.0
87.0
57.0
59.0
61.0
63.0
75.0
77.0
81.0
83.0
51.0
53.0
41.0
43.0
, 19},

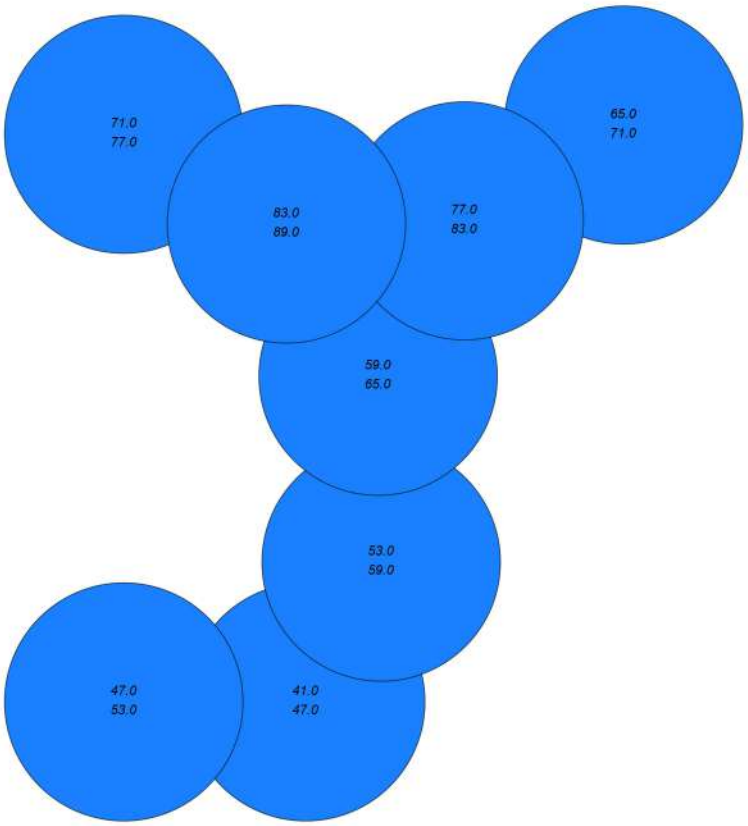

65.0
68.0
77.0
80.0
59.0
62.0
62.0
65.0
71.0
74.0
83.0
86.0
53.0
56.0
56.0
59.0
86.0
89.0
68.0
71.0
74.0
77.0
80.0
83.0
50.0
53.0
41.0
44.0

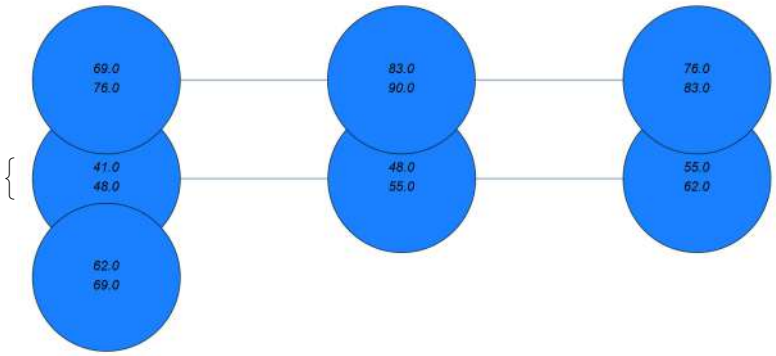
{ , 14},


65.0
69.0
77.0
81.0
61.0
65.0
85.0
89.0
53.0
57.0
69.0
73.0
57.0
61.0
73.0
77.0
81.0
85.0
49.0
53.0
41.0
45.0

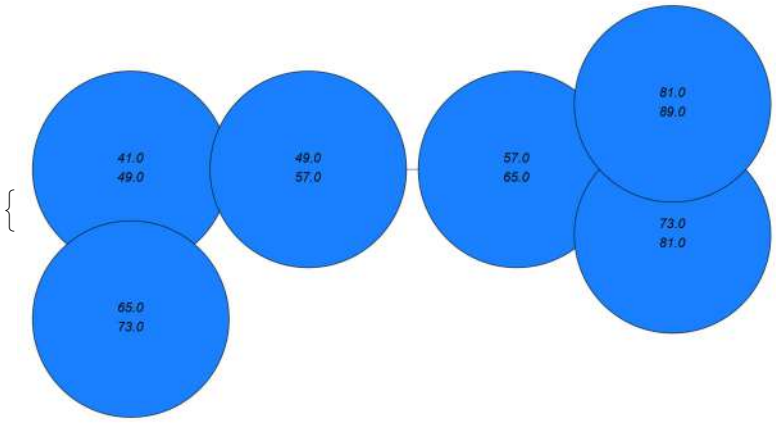
{ , 11},

{ 
71.0
76.0
56.0
61.0
61.0
66.0
81.0
86.0
86.0
91.0
51.0
56.0
41.0
46.0
76.0
81.0
66.0
71.0
, 9},

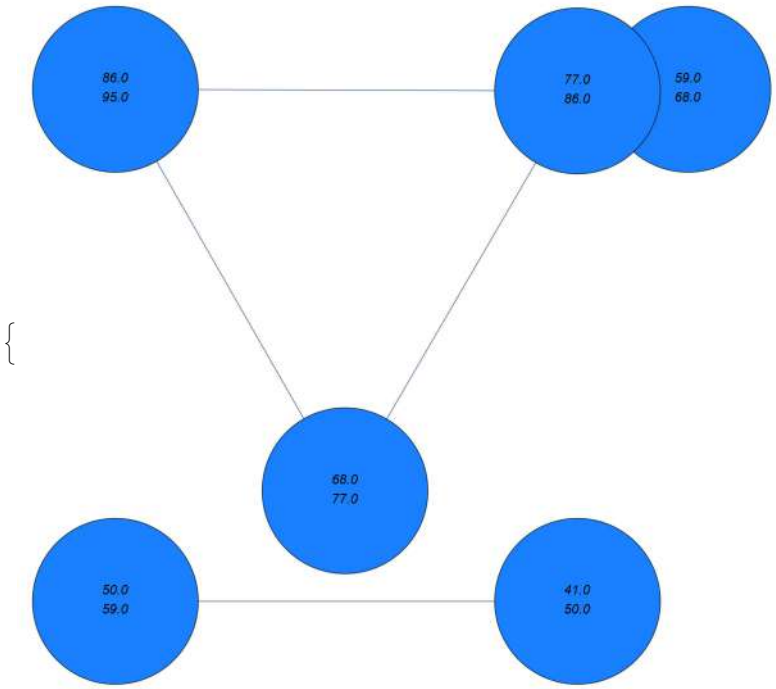
{ 
71.0
77.0
65.0
71.0
83.0
89.0
77.0
83.0
59.0
65.0
53.0
59.0
47.0
53.0
41.0
47.0
, 8},


69.0
76.0
83.0
90.0
76.0
83.0
41.0
48.0
48.0
55.0
55.0
62.0
62.0
69.0

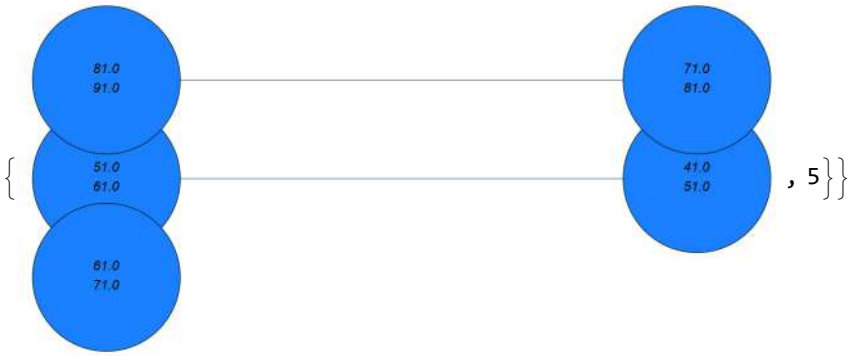
{ , 7},


41.0
49.0
49.0
57.0
57.0
65.0
81.0
89.0
73.0
81.0
65.0
73.0

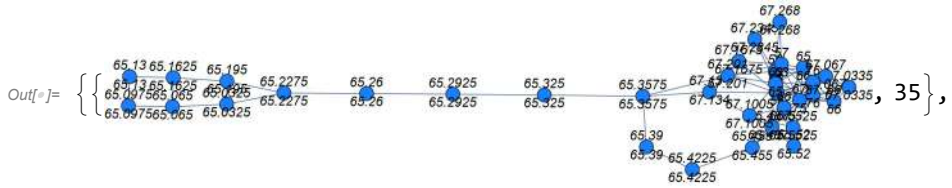
{ , 6},

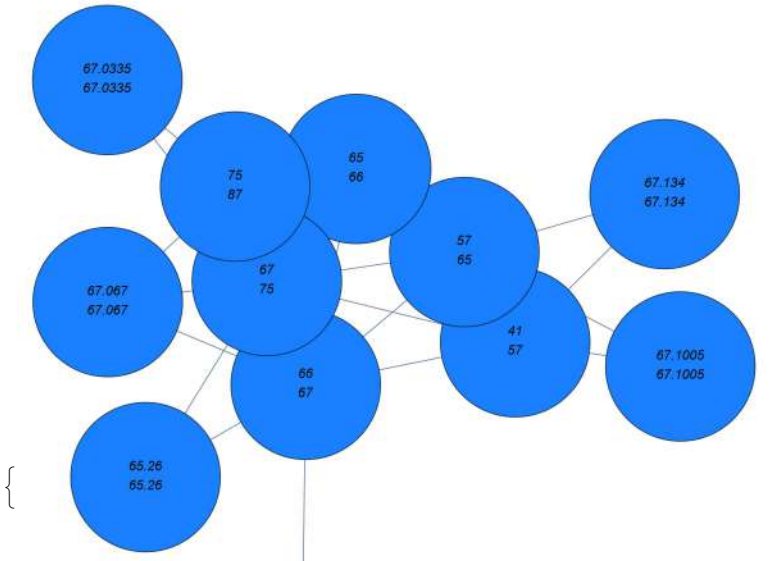

86.0
95.0
77.0
86.0
59.0
68.0
68.0
77.0
50.0
59.0
41.0
50.0

{ , 6},

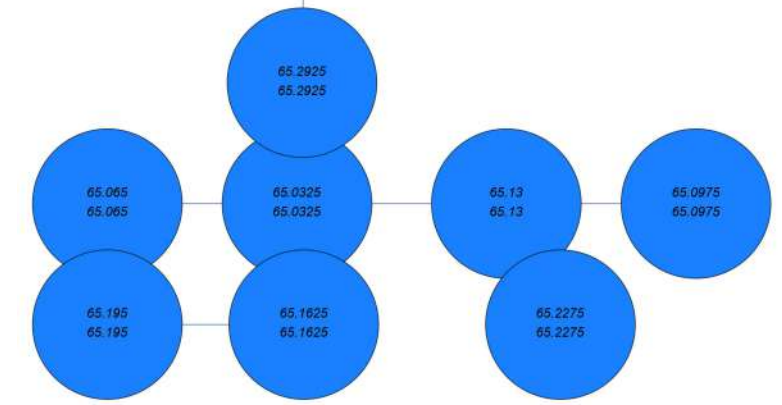


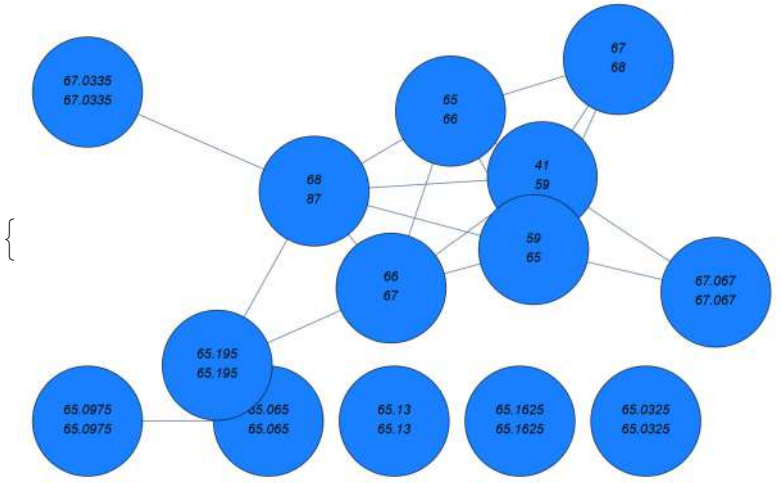
In[ ]:= **graphsandnodenumbersFixedbucket**

Out[ ]= 

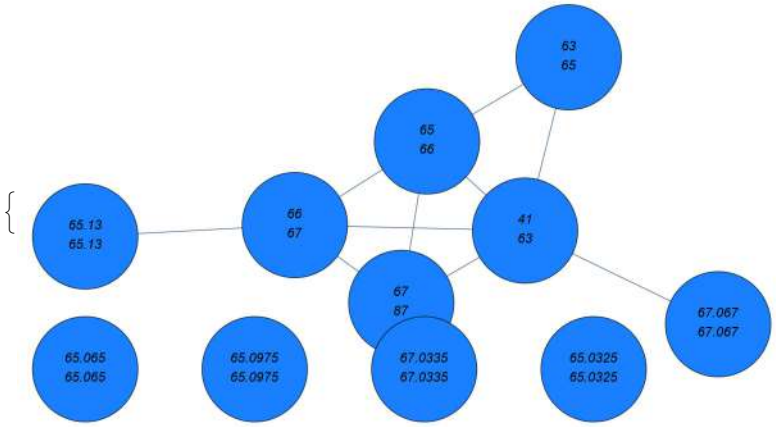

67.0335
67.0335
75
87
65
66
57
65
67.134
67.134
67.067
67.067
67
75
41
57
67.1005
67.1005
66
67
65.26
65.26

{ , 19},

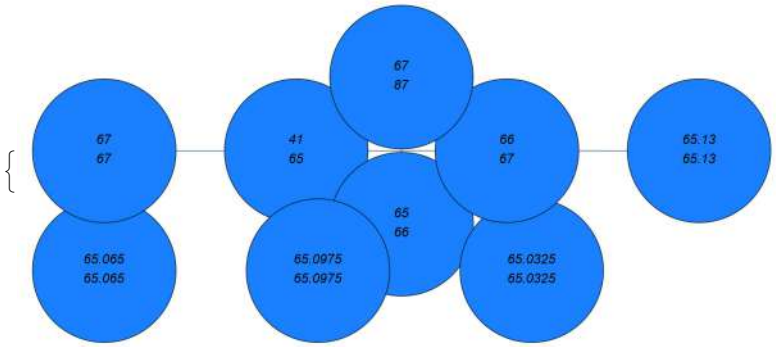

65.2925
65.2925
65.065
65.065
65.0325
65.0325
65.13
65.13
65.0975
65.0975
65.195
65.195
65.1625
65.1625
65.2275
65.2275


67
68
67.0335
67.0335
65
66
41
59
68
87
59
65
66
67
67.067
67.067
65.195
65.195
65.0975
65.0975
65.065
65.065
65.13
65.13
65.1625
65.1625
65.0325
65.0325

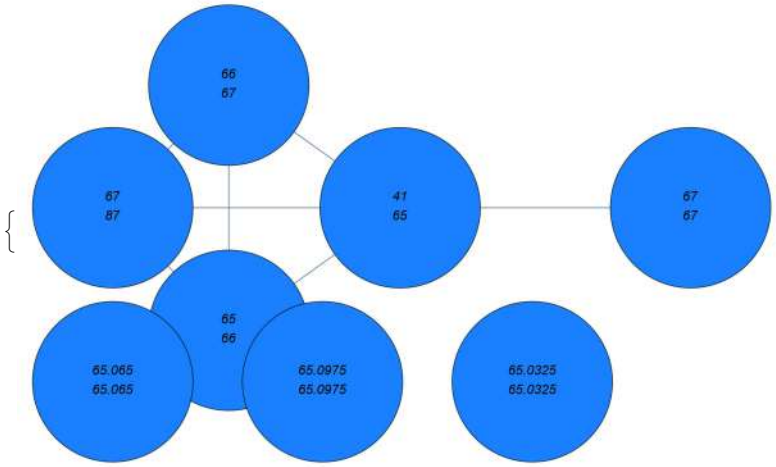
{ , 14},


63
65
65
66
65.13
65.13
66
67
41
63
67
87
67.067
67.067
65.065
65.065
65.0975
65.0975
67.0335
67.0335
65.0325
65.0325

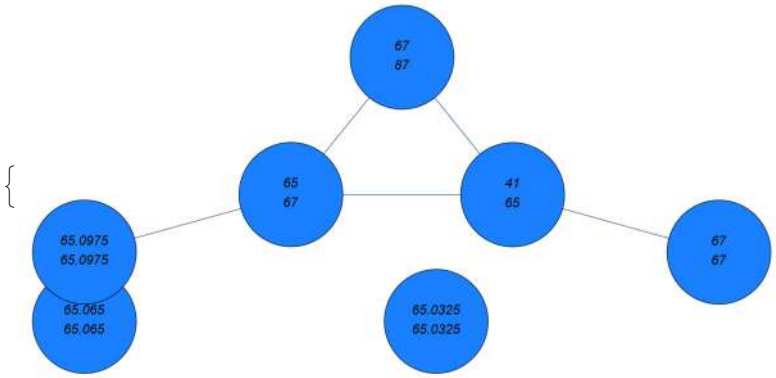
, 11},


67
87
67
67
41
65
66
67
65.13
65.13
65
66
65.065
65.065
65.0975
65.0975
65.0325
65.0325

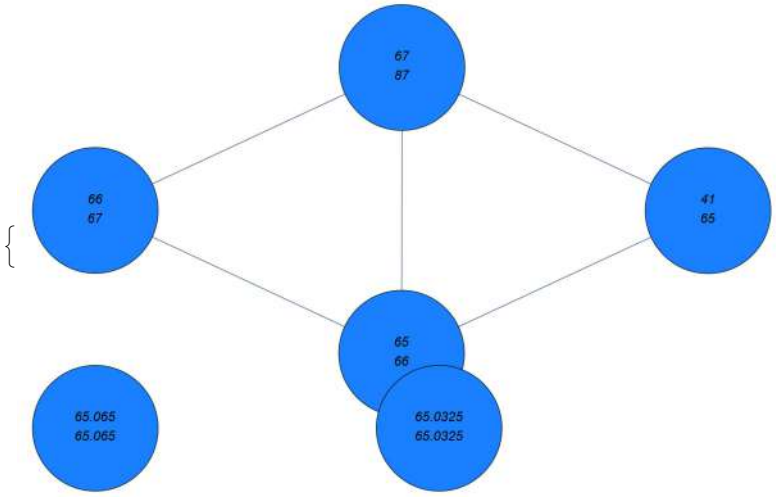
, 9},


66
67
67
87
41
65
67
67
65
66
65.065
65.065
65.0975
65.0975
65.0325
65.0325

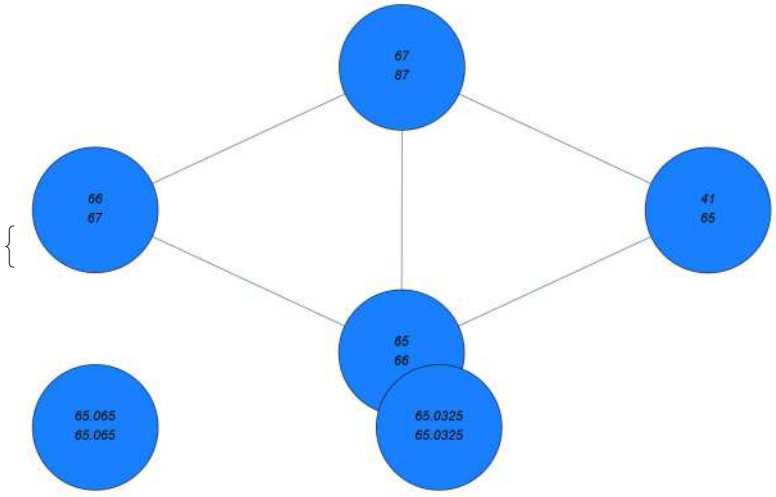
, 8},


67
87
65
67
41
65
65.0975
65.0975
67
67
65.065
65.065
65.0325
65.0325

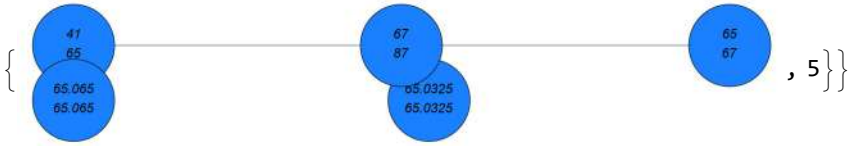
{ , 7},


67
87
66
67
41
65
65
66
65.065
65.065
65.0325
65.0325

{ , 6},


67
87
66
67
41
65
65
66
65.065
65.065
65.0325
65.0325

{ , 6},

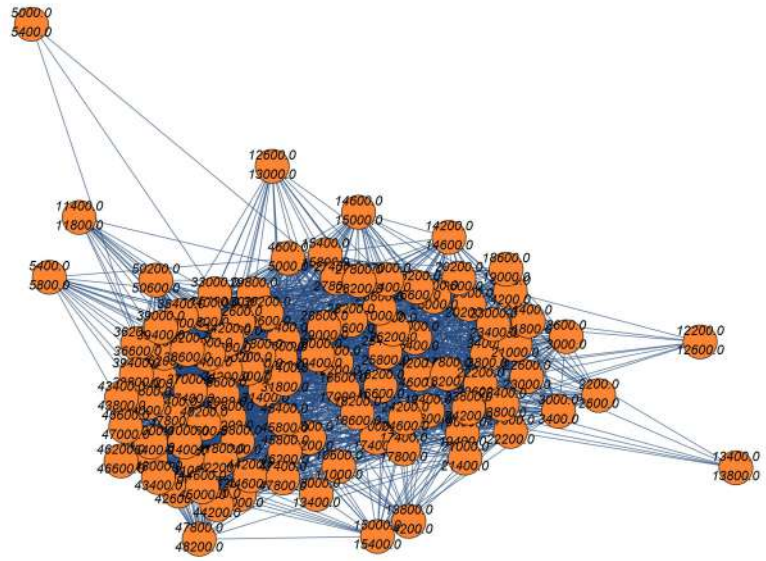


, 5}}

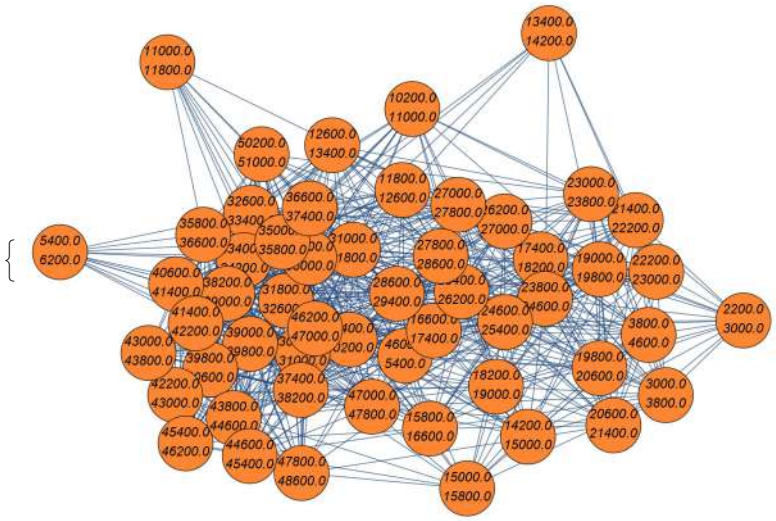
Length Graphs for Varying Fixed Step Sizes and Fixed Bucket Sizes

In[ ]:= **graphsandnodenumbers**

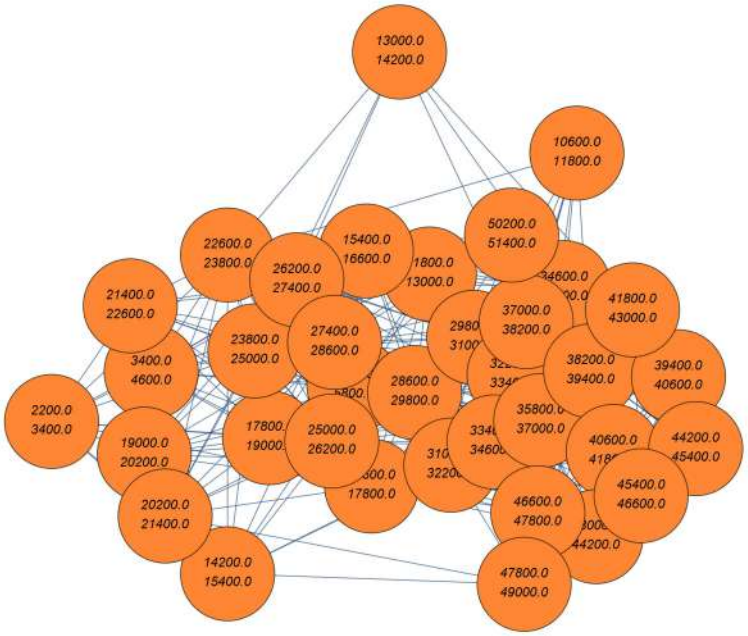
Out[ ]= {{

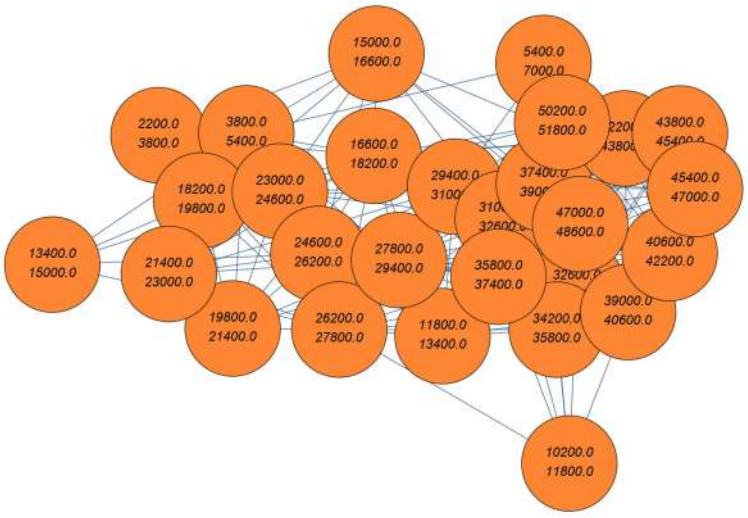


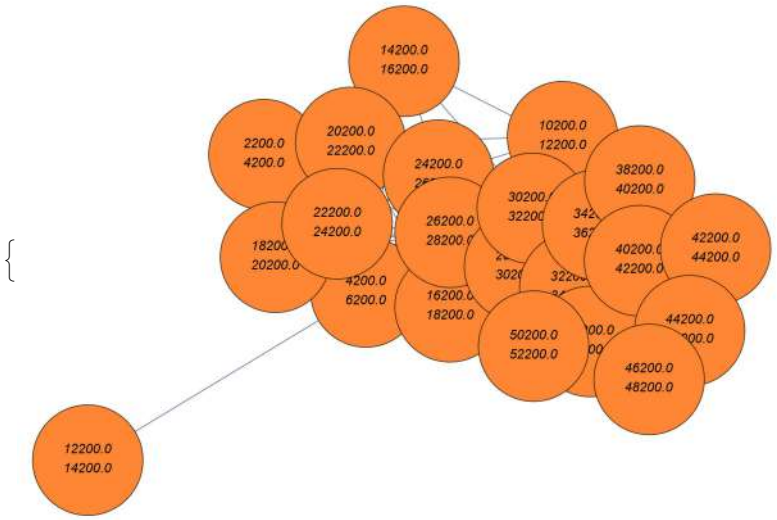
, 102},



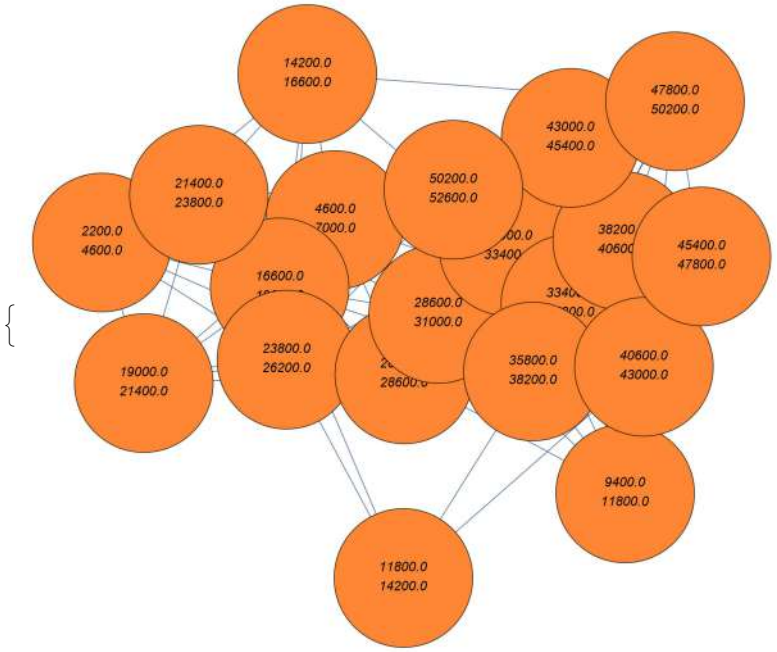
, 54},

{ 
13000.0
14200.0
10600.0
11800.0
50200.0
51400.0
22600.0
23800.0
15400.0
16600.0
26200.0
27400.0
1800.0
13000.0
21400.0
22600.0
41800.0
43000.0
37000.0
38200.0
27400.0
28600.0
23800.0
25000.0
3400.0
4600.0
38200.0
39400.0
39400.0
40600.0
28600.0
29800.0
2200.0
3400.0
35800.0
37000.0
17800.0
19000.0
25000.0
26200.0
40600.0
44200.0
45400.0
19000.0
20200.0
45400.0
46600.0
20200.0
21400.0
46600.0
47800.0
14200.0
15400.0
47800.0
49000.0
, 36},

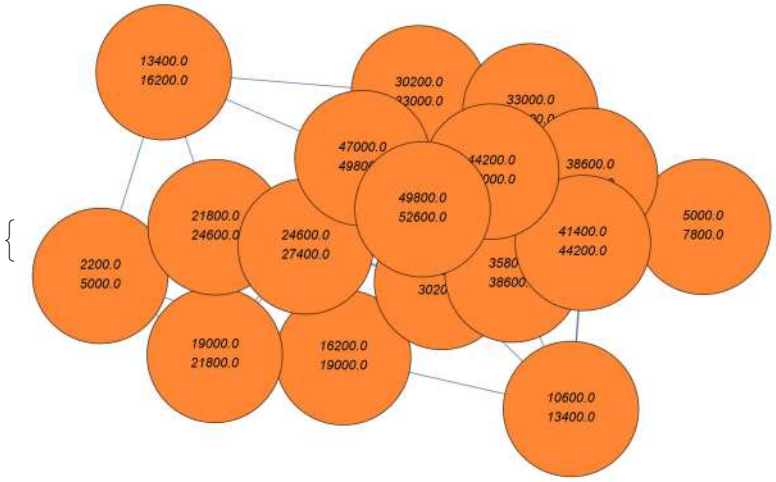
{ 
15000.0
16600.0
5400.0
7000.0
50200.0
51800.0
2200.0
3800.0
3800.0
5400.0
43800.0
45400.0
16600.0
18200.0
18200.0
19800.0
23000.0
24600.0
45400.0
47000.0
47000.0
48600.0
13400.0
15000.0
21400.0
23000.0
24600.0
26200.0
27800.0
29400.0
40600.0
42200.0
35800.0
37400.0
39000.0
40600.0
19800.0
21400.0
26200.0
27800.0
11800.0
13400.0
34200.0
35800.0
10200.0
11800.0
, 28},

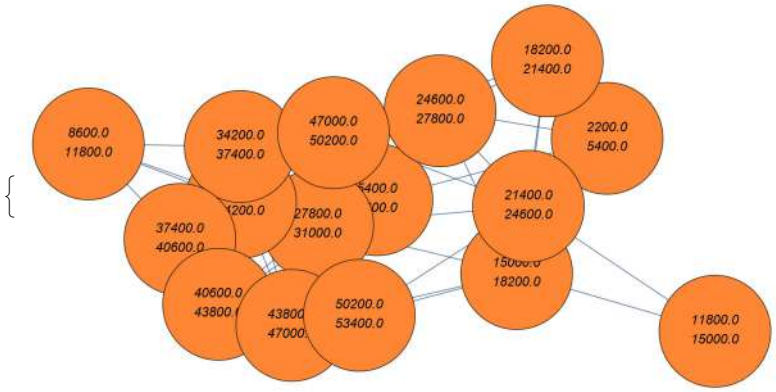

14200.0
16200.0
2200.0
4200.0
20200.0
22200.0
10200.0
12200.0
24200.0
38200.0
40200.0
22200.0
24200.0
26200.0
28200.0
42200.0
44200.0
40200.0
42200.0
4200.0
6200.0
16200.0
18200.0
50200.0
52200.0
44200.0
46200.0
48200.0
12200.0
14200.0

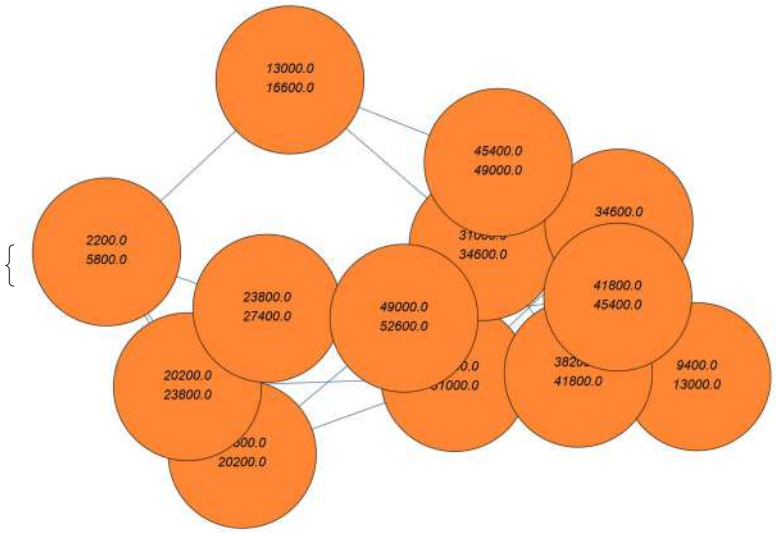
, 22},

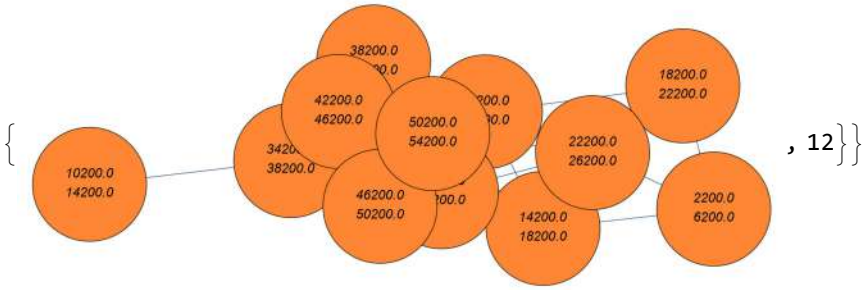

14200.0
16600.0
47800.0
50200.0
43000.0
45400.0
21400.0
23800.0
50200.0
52600.0
2200.0
4600.0
4600.0
45400.0
47800.0
16600.0
28600.0
31000.0
19000.0
21400.0
23800.0
26200.0
35800.0
38200.0
40600.0
43000.0
9400.0
11800.0
11800.0
14200.0

, 20},

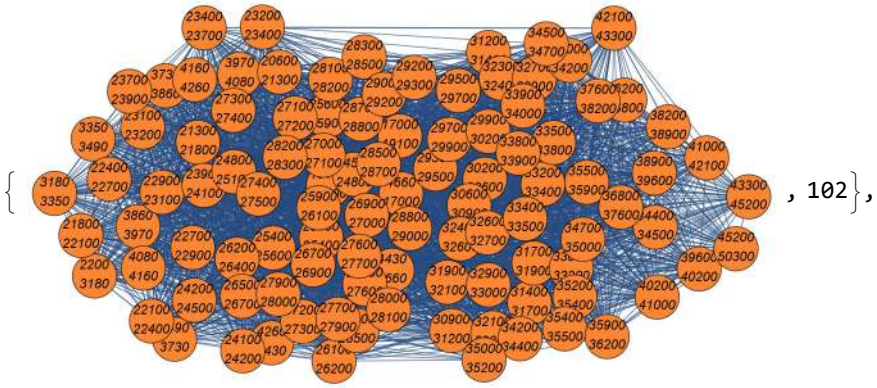
{ 
13400.0
16200.0
30200.0
33000.0
47000.0
49800.0
52600.0
44200.0
38600.0
21800.0
24600.0
24600.0
27400.0
41400.0
44200.0
5000.0
7800.0
2200.0
5000.0
19000.0
21800.0
16200.0
19000.0
10600.0
13400.0
, 17},

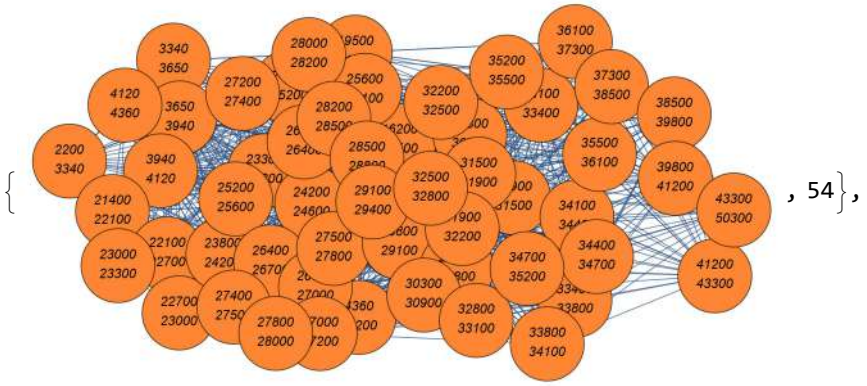
{ 
18200.0
21400.0
8600.0
11800.0
34200.0
37400.0
47000.0
50200.0
24600.0
27800.0
2200.0
5400.0
21400.0
24600.0
37400.0
40600.0
27800.0
31000.0
40600.0
43800.0
50200.0
53400.0
18200.0
11800.0
15000.0
, 16},

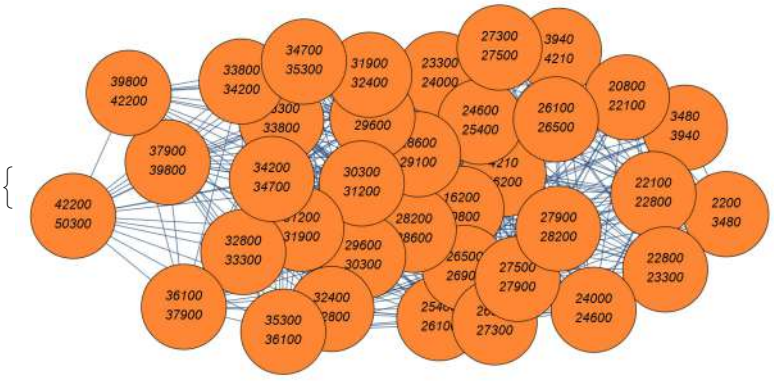
{ 
13000.0
16600.0
45400.0
49000.0
34600.0
34600.0
2200.0
5800.0
23800.0
27400.0
49000.0
52600.0
41800.0
45400.0
20200.0
23800.0
41800.0
9400.0
13000.0
20200.0
, 13},



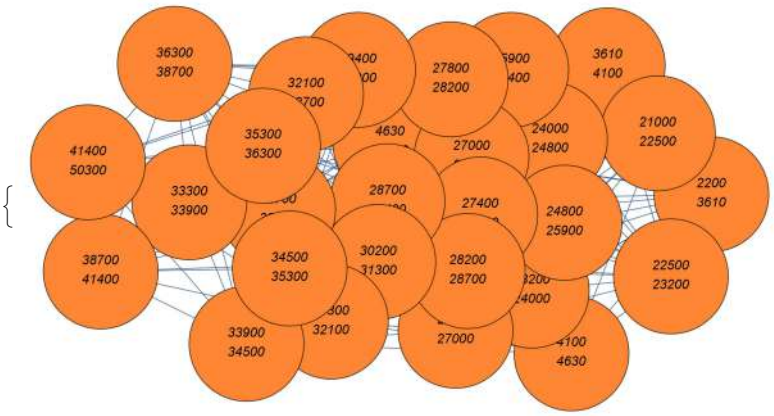
In[ ]:= **graphsandnodenumbersFixedbucket**

Out[ ]= 

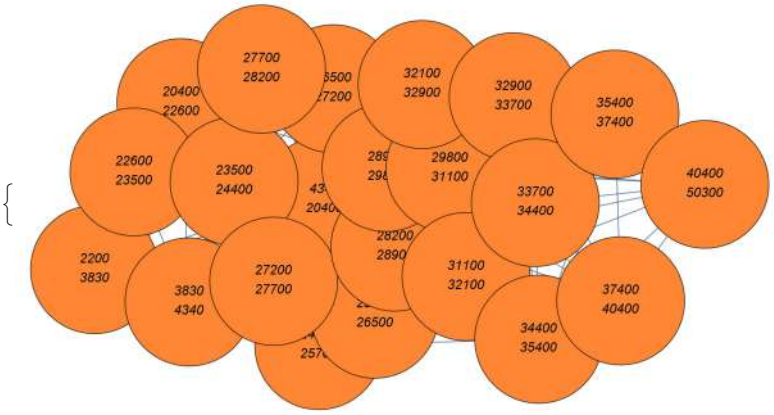



39800
42200
33800
34200
34700
35300
31900
32400
23300
24000
27300
27500
3940
4210
20800
22100
3480
3940
24600
25400
26100
26500
29600
37900
39800
34200
34700
30300
31200
22100
22800
2200
3480
42200
50300
32800
33300
27900
28200
22800
23300
36100
37900
35300
36100
32400
24000
24600
27300

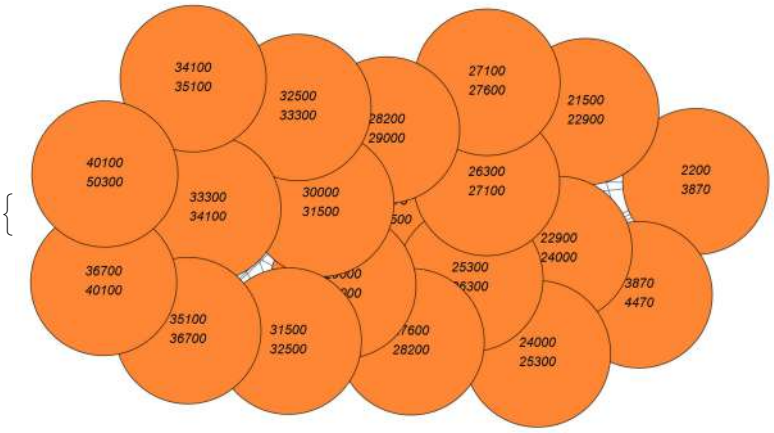
{ , 36},


36300
38700
32100
27800
28200
3610
4100
21000
22500
41400
50300
35300
36300
24000
24800
27000
33300
33900
28700
27400
24800
25900
2200
3610
38700
41400
34500
35300
30200
31300
28200
28700
22500
23200
33900
34500
32100
27000
4100
4630

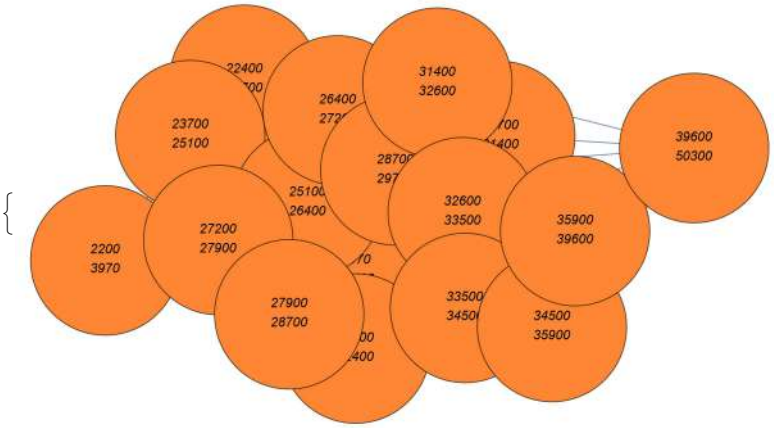
{ , 28},


27700
28200
20400
22600
32100
32900
32900
33700
35400
37400
22600
23500
23500
24400
29800
31100
33700
34400
40400
50300
28200
2890
2200
3830
27200
27700
31100
32100
3830
4340
37400
40400
26500
34400
35400

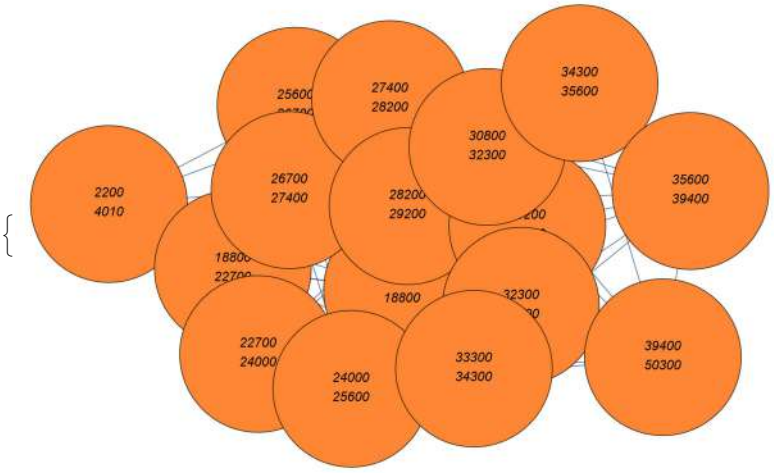
{ , 22},


34100
35100
32500
33300
28200
29000
27100
27600
21500
22900
2200
3870
40100
50300
33300
34100
30000
31500
26300
27100
22900
24000
25300
36700
40100
35100
36700
31500
32500
24000
25300
3870
4470

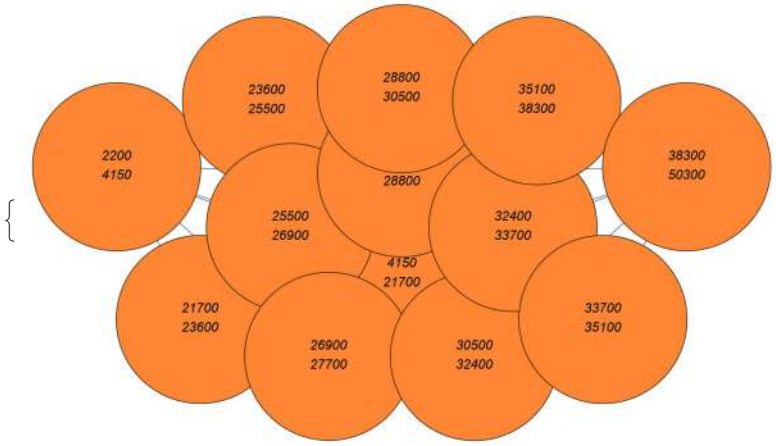
, 20},


31400
32600
26400
23700
25100
39600
50300
32600
33500
35900
39600
27200
27900
2200
3970
27900
28700
34500
35900

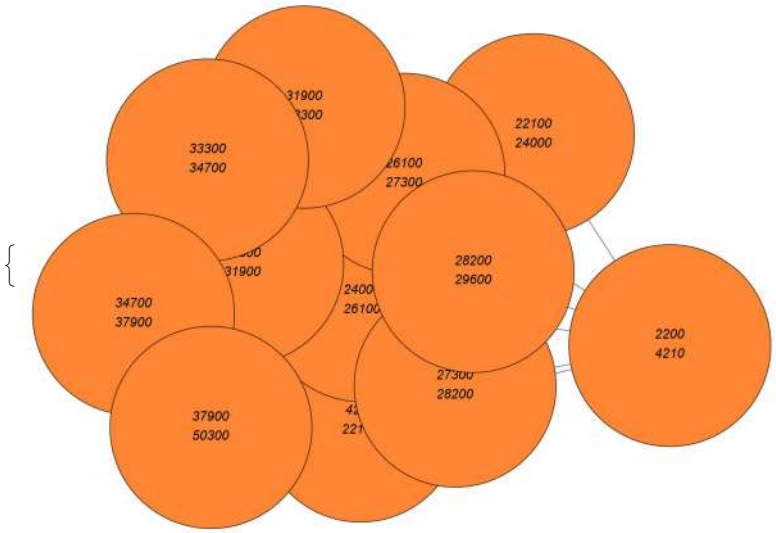
, 17},


25600
27400
28200
34300
35600
30800
32300
2200
4010
26700
27400
28200
29200
35600
39400
18800
32300
22700
24000
24000
25600
33300
34300
39400
50300

, 16},



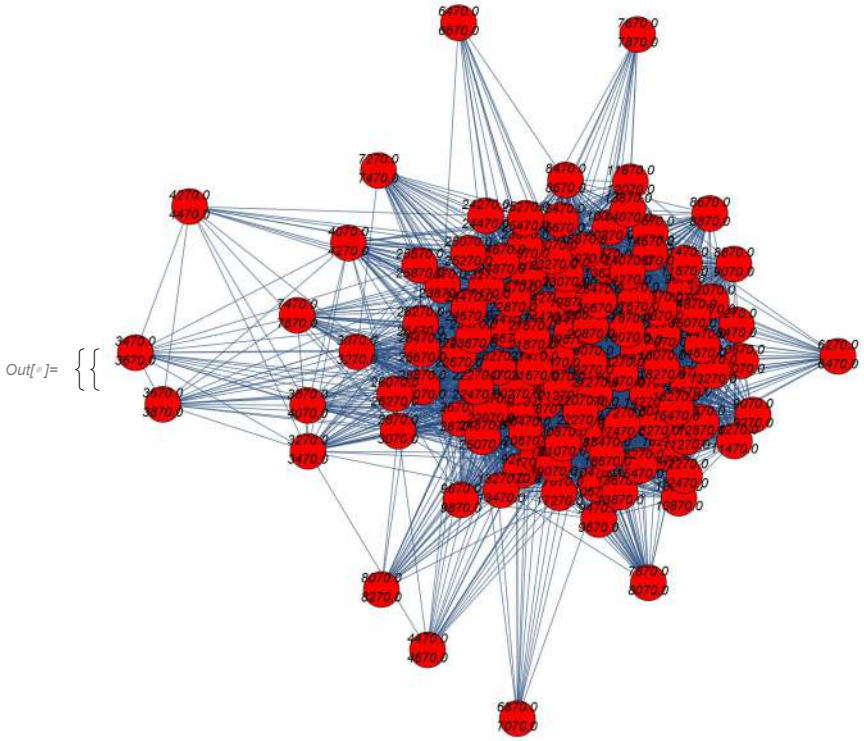
, 13},

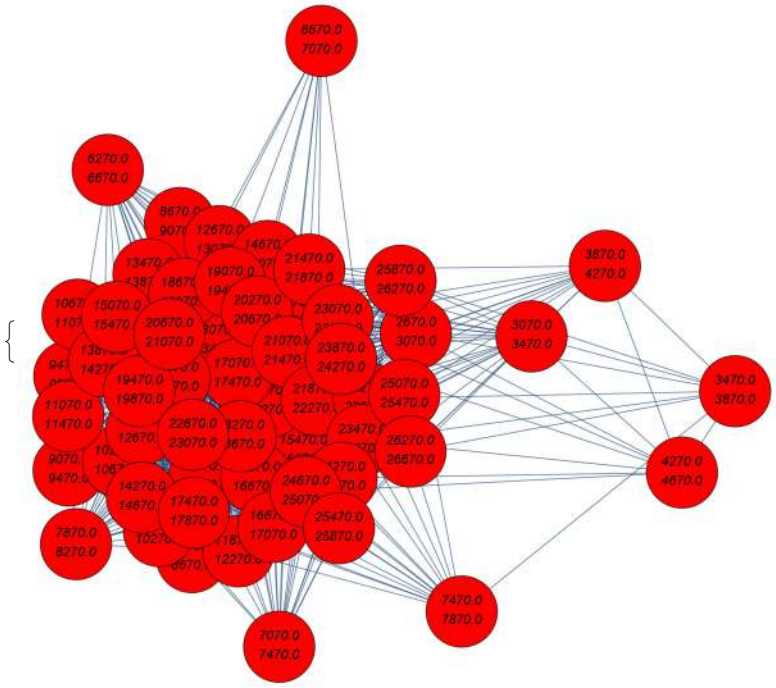


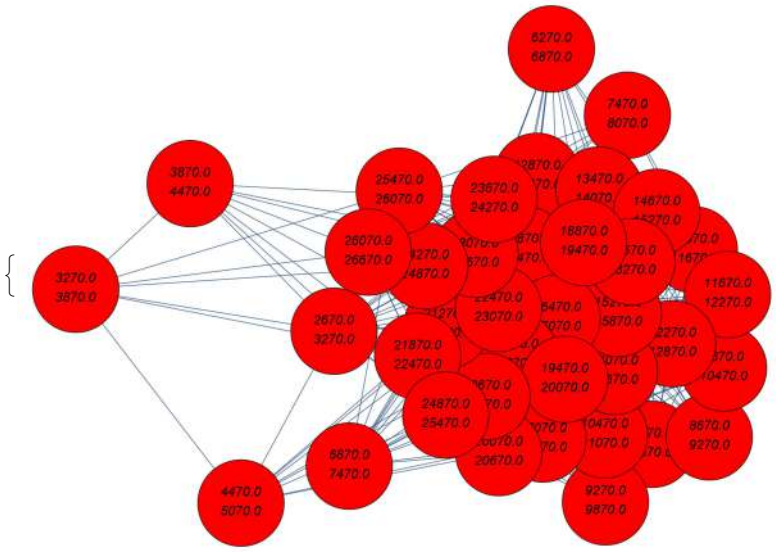
, 12}}

Weight Graphs for Varying Fixed Step Sizes and Fixed Bucket Sizes

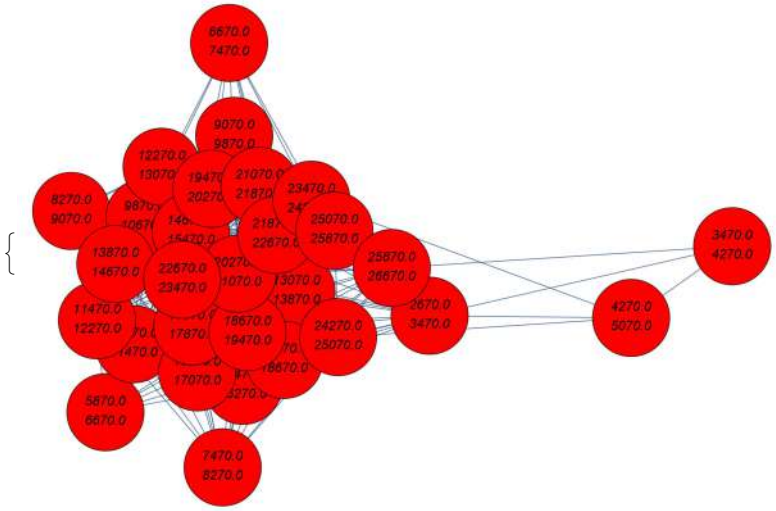
In[ ]:= **graphsandnodenumbers**

Out[•]=  , 110},

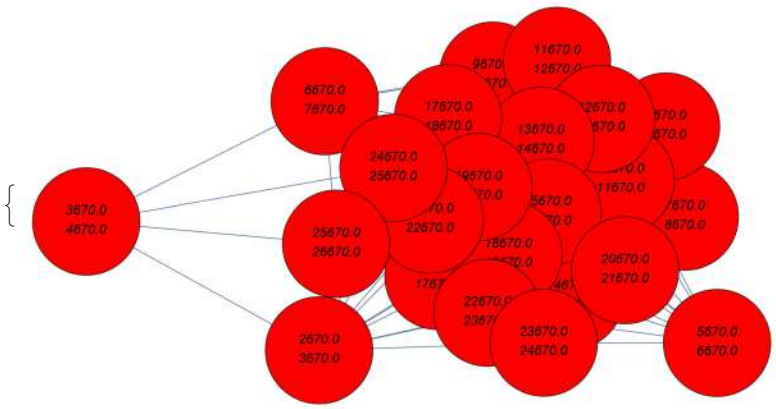
 , 56},


6270.0
6870.0
7470.0
8070.0
3870.0
4470.0
25470.0
26070.0
23670.0
24270.0
13470.0
14670.0
26070.0
26670.0
18870.0
19470.0
3270.0
3870.0
11670.0
12270.0
2670.0
3270.0
21870.0
22470.0
19470.0
20070.0
24870.0
25470.0
8670.0
9270.0
6870.0
7470.0
4470.0
5070.0
9270.0
9870.0

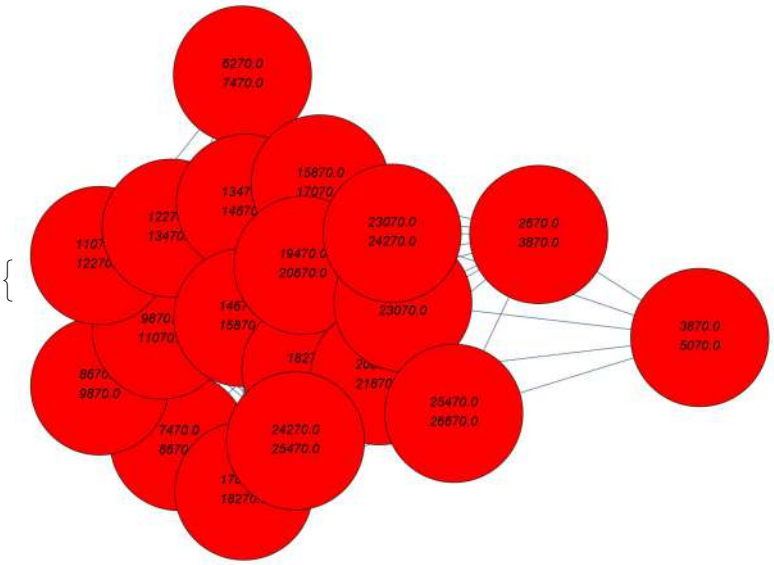
, 38},


6670.0
7470.0
9070.0
9870.0
12270.0
13070.0
21070.0
21870.0
23470.0
8270.0
9070.0
25070.0
25870.0
22670.0
3470.0
4270.0
13870.0
14670.0
25870.0
26670.0
22670.0
23470.0
13070.0
13870.0
2670.0
3470.0
11470.0
12270.0
18670.0
19470.0
24270.0
25070.0
4270.0
5070.0
17070.0
5870.0
6670.0
7470.0
8270.0

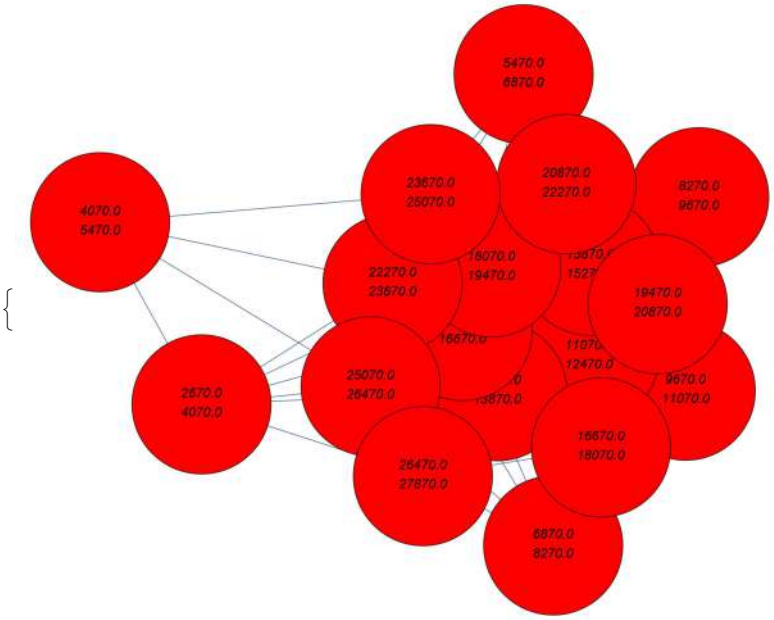
, 29},


11670.0
12670.0
6670.0
7670.0
17670.0
18670.0
12670.0
13670.0
14670.0
24670.0
25670.0
11670.0
3670.0
4670.0
25670.0
26670.0
22670.0
18670.0
20670.0
21670.0
22670.0
23670.0
24670.0
2670.0
3670.0
5670.0
6670.0

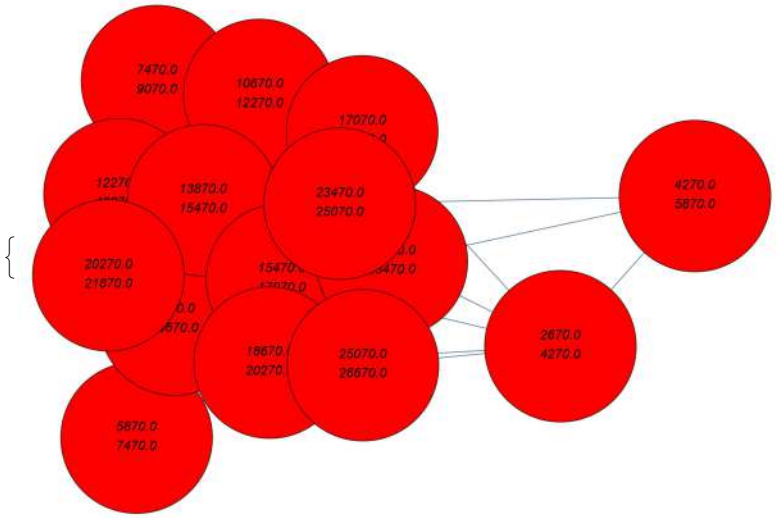
, 23},

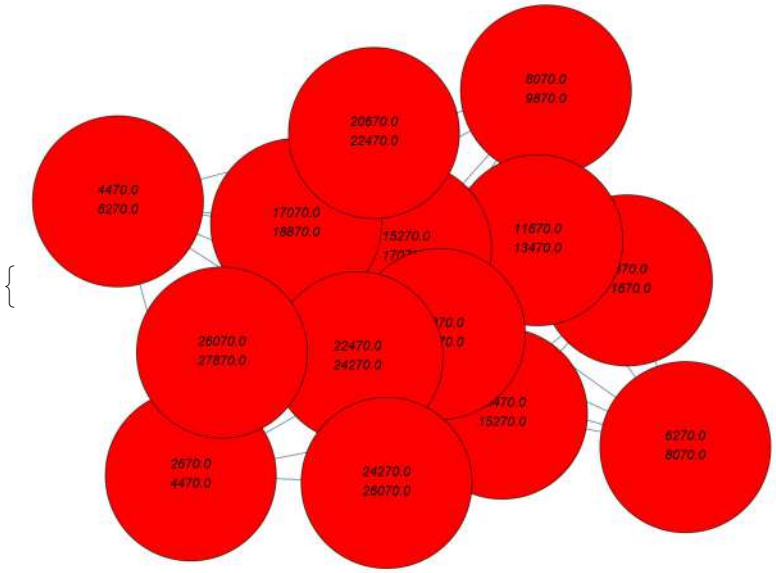

6270.0
7470.0
15870.0
17070.
23070.0
24270.0
2670.0
3870.0
19470.0
20670.0
23070.0
3870.0
5070.0
25470.0
26670.0
24270.0
25470.0

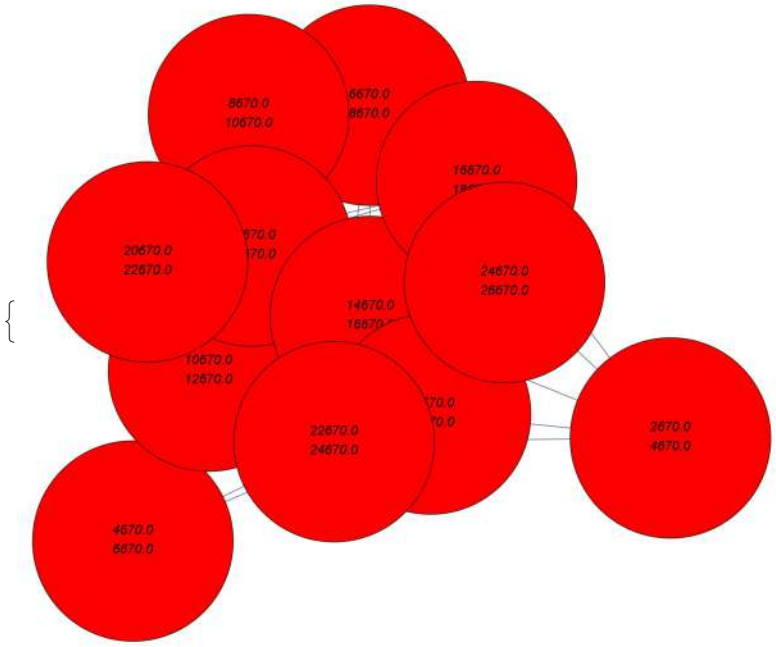
, 19},


5470.0
6870.0
23670.0
25070.0
20870.0
22270.0
8270.0
9670.0
4070.0
5470.0
22270.0
23670.0
18070.0
19470.0
19470.0
20870.0
16670.0
11070.
12470.0
25070.0
26470.0
9670.0
11070.0
2670.0
4070.0
16670.0
18070.0
26470.0
27870.0
6870.0
8270.0

, 18},

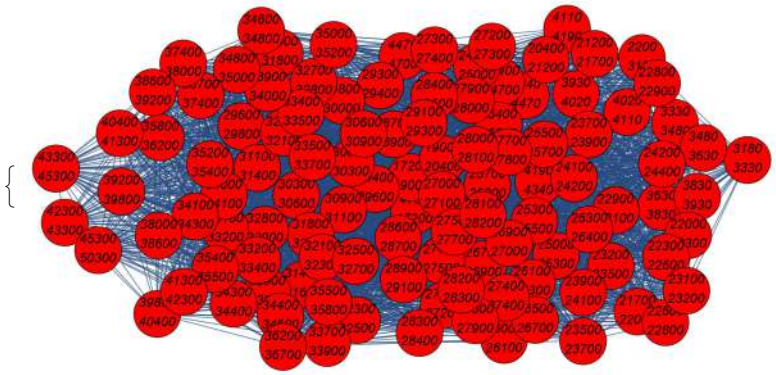
{ 
7470.0
9070.0
10670.0
12270.0
17070.0
13870.0
15470.0
23470.0
25070.0
4270.0
5870.0
20270.0
21870.0
2670.0
4270.0
18670.
20270.
25070.0
26670.0
5870.0
7470.0
, 15},

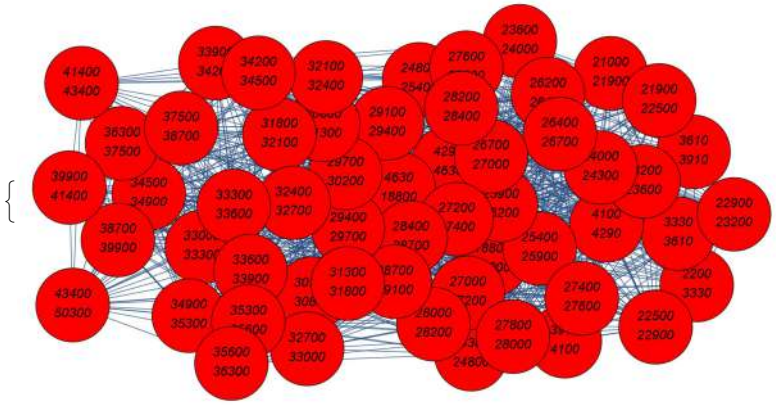
{ 
8070.0
9870.0
20670.0
22470.0
4470.0
6270.0
17070.0
18870.0
11670.0
13470.0
26070.0
27870.0
22470.0
24270.0
6270.0
8070.0
2670.0
4470.0
24270.0
26070.0
, 14},



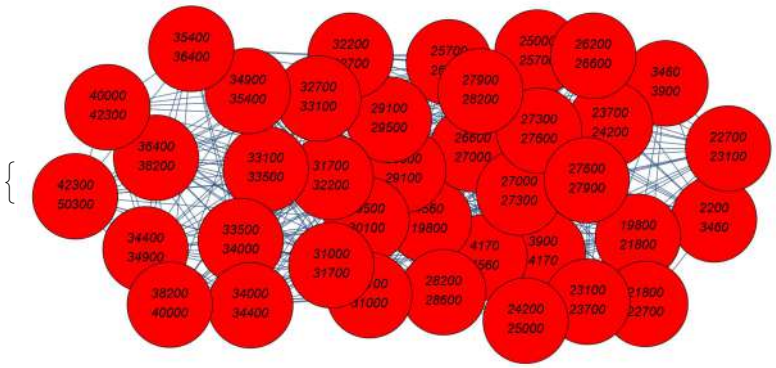
, 12}}

In[ ]:= **graphsandnodenumbersFixedbucket**

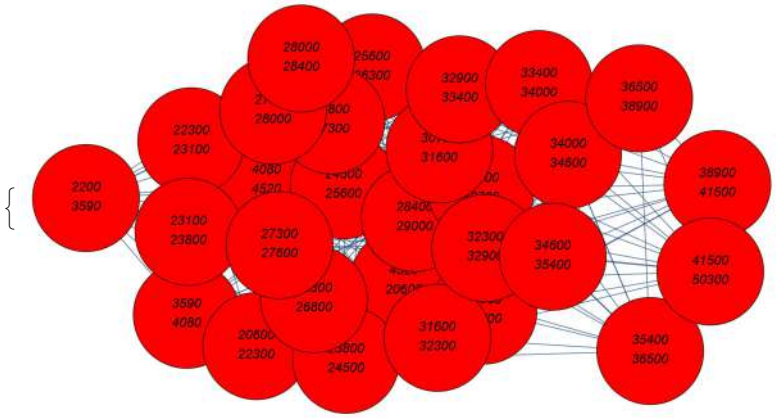
Out[ ]= {{  , 110},



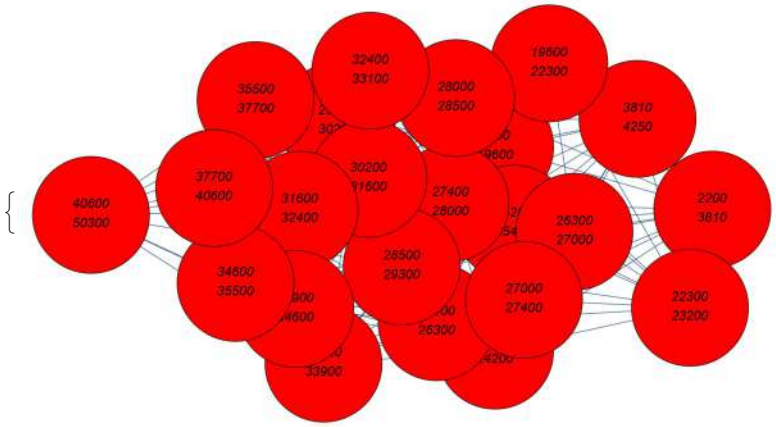
{, 56},


35400
36400
32200
34900
35400
32700
33100
40000
42300
29100
29500
27900
28200
25000
25700
26200
26600
3460
3900
27300
27600
23700
24200
22700
23100
36400
38200
33100
33500
31700
32200
29100
27000
27300
27800
27900
42300
50300
2200
3460
34400
34900
33500
34000
31000
31700
19800
21800
3900
4170
38200
40000
34000
34400
28200
28600
24200
25000
23100
23700
21800
22700

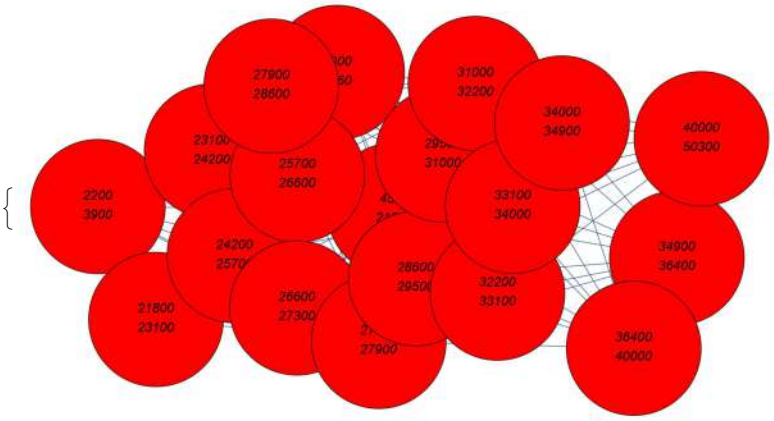
{ , 38},


28000
28400
25600
32900
33400
33400
34000
36500
36900
22300
23100
34000
34600
31600
38900
41500
2200
3590
4080
4520
25600
23100
23800
27300
27600
28400
29000
32300
32900
34600
35400
41500
50300
3590
4080
26800
31600
32300
20600
22300
24500
35400
36500

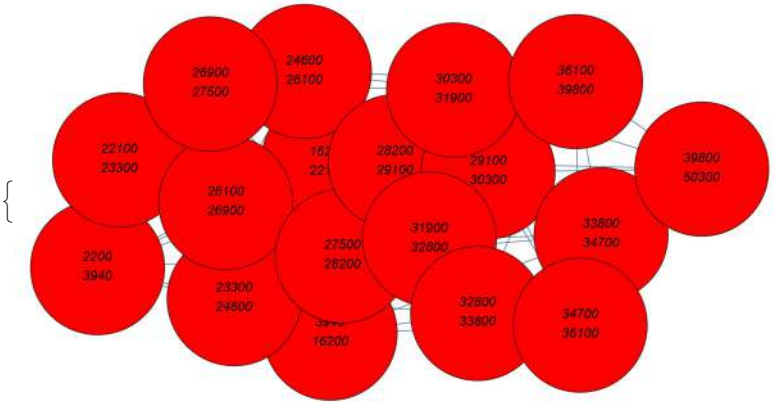
{ , 29},


32400
33100
19600
22300
35500
37700
28000
28500
3810
4250
37700
40600
30200
31600
27400
28000
40600
50300
31600
32400
26300
27000
2200
3810
34600
35500
26500
29300
27000
27400
22300
23200
33900
26300

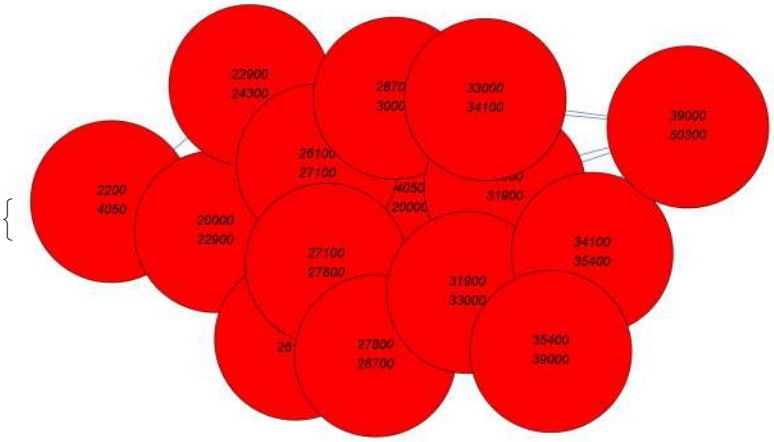
{ , 23},


27900
28600
31000
32200
34000
34900
40000
50300
23100
24200
25700
26600
2200
3900
33100
34000
24200
34900
36400
28600
32200
33100
21800
23100
26600
36400
40000
27900

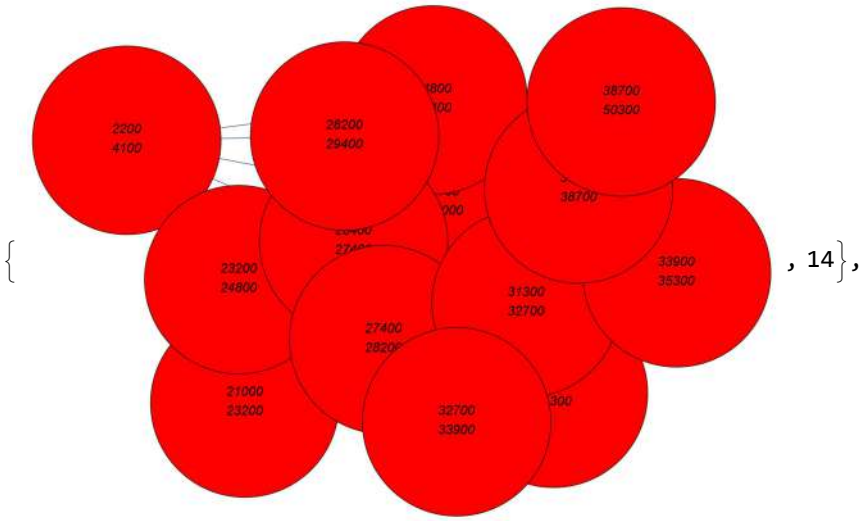
, 19},


26900
27600
24600
26100
30300
31900
36100
39800
22100
23300
28200
29100
29100
30300
39800
50300
26100
26900
31900
32800
33800
34700
2200
3940
27500
28200
23300
24600
32800
33800
34700
36100
16200

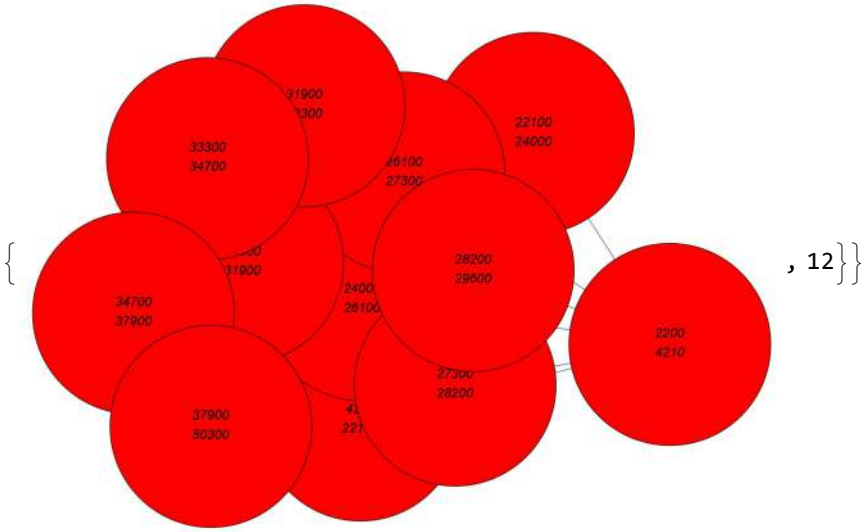
, 18},


22900
24300
33000
34100
39000
50300
26100
27100
2200
4050
20000
22900
31900
27100
27800
34100
35400
31900
33000
27800
28700
35400
39000

, 15},



, 14},



, 12}}

Plots Together

Fixed Step Size Networks

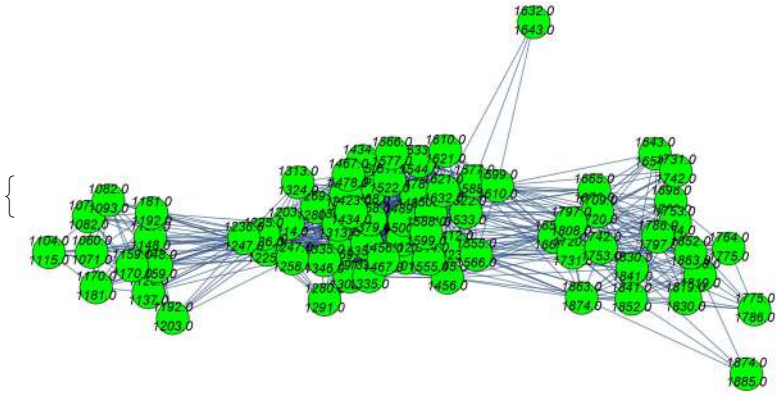
Width Feature

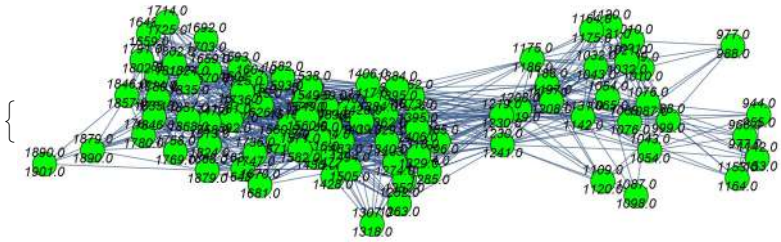
In[ ]:=
```
AbsoluteTiming[
 widthdataintimewindowsFixedstep = snetworkdatabinnedintimewindows[9, 11, 10];]
```

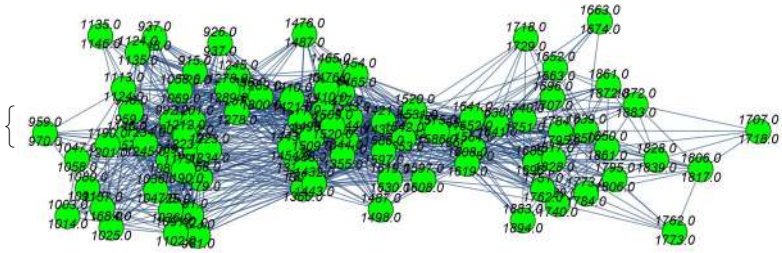
Out[ ]= {48.4787, Null}

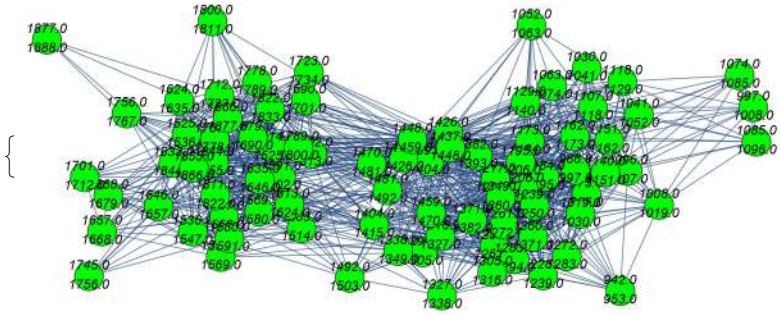
In[ ]:=
```
graphsandnodenumbers = Table[snetworkgraph[widthdataintimewindowsFixedstep[[1]][[i]],
    widthdataintimewindowsFixedstep[[2]][[i]], 2, 7, 400, Green], {i, Range@10}];
```

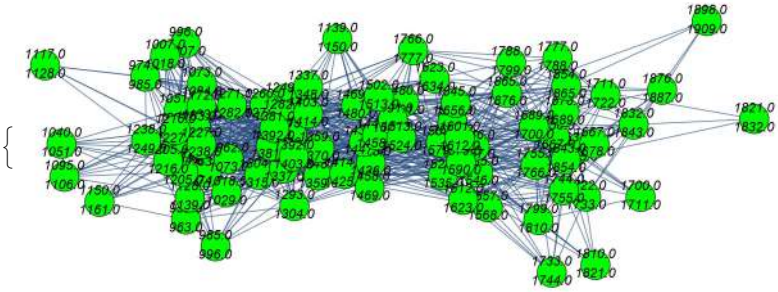
In[ ]:=
```
graphsandnodenumbers
```

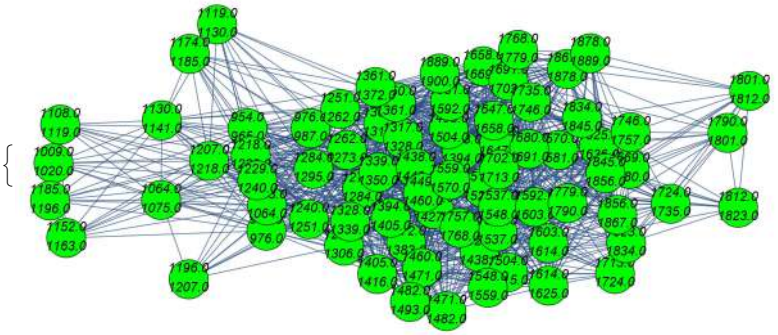
Out[•]= {{  , 71},

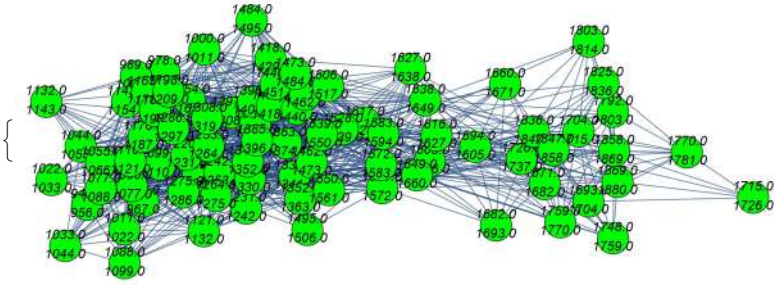
{  , 85},

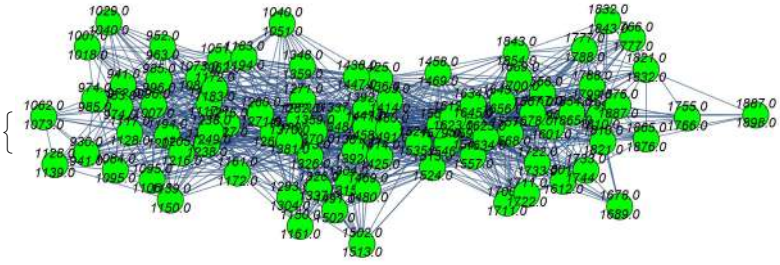
{  , 88},

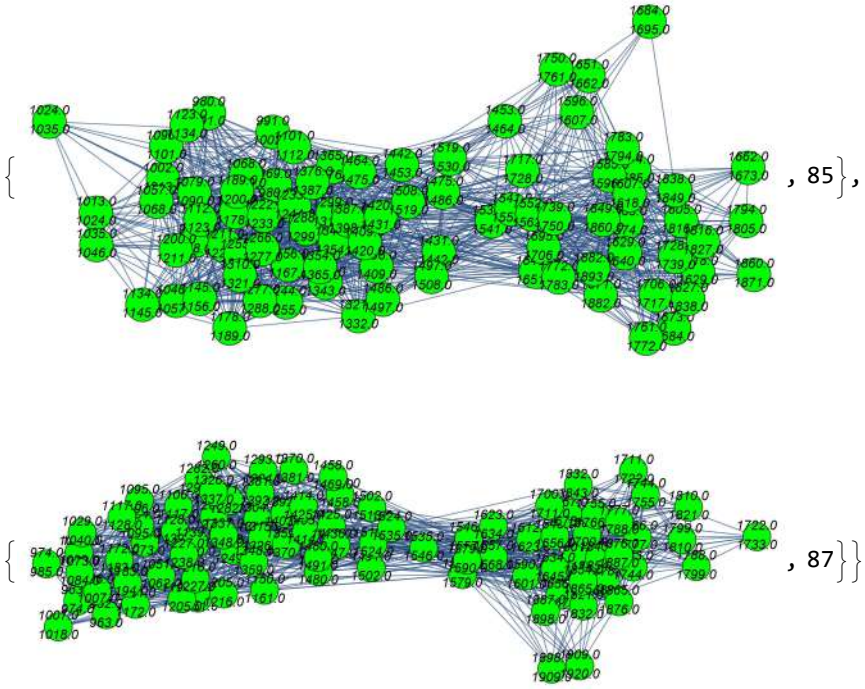
{  , 83},


, 83},


, 76},


, 81},


, 86},



In[ ]:=
```
ABCvalues = Table[Mean@BetweennessCentrality[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
      FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
      "Normalized" → False], {i, Length@graphsandnodenumbers}];
degreevalues = Table[N@Mean@VertexDegree[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
```

In[ ]:=
```
singlerandomgraphserdren = Table[
    RandomGraph[{VertexCount[i], EdgeCount[i]}], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomerdrenmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphserdren[[i]],
      FindGraphCommunities[singlerandomgraphserdren[[i]]], "Normalized" -> False],
    {i, Length@singlerandomgraphserdren}];
singlerandomgraphsdegreesfxd = Table[IGDegreeSequenceGame[Total[AdjacencyMatrix@i],
Method -> "VigerLatapy"], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomdegreesfxdmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphsdegreesfxd[[i]],
      FindGraphCommunities[singlerandomgraphsdegreesfxd[[i]]], "Normalized" -> False],
    {i, Length@singlerandomgraphsdegreesfxd}];
singlerandomgraphscomm = Table[randomizedgraphamongcommunities[i],
    {i, graphsandnodenumbers[[All, 1]]}];
singlerandomcommmodularityvalues = Table[N@GraphAssortativity[
      singlerandomgraphscomm[[i]], FindGraphCommunities[singlerandomgraphscomm[[i]]],
      "Normalized" -> False], {i, Length@singlerandomgraphscomm}];
```

```
In[ ]:= AbsoluteTiming[ZscoresmodularityWolf =
          Table[randomnessfunctionformodularity[i, "Wolf"], {i, graphsandnodenumbers[[All, 1]]}]]
```

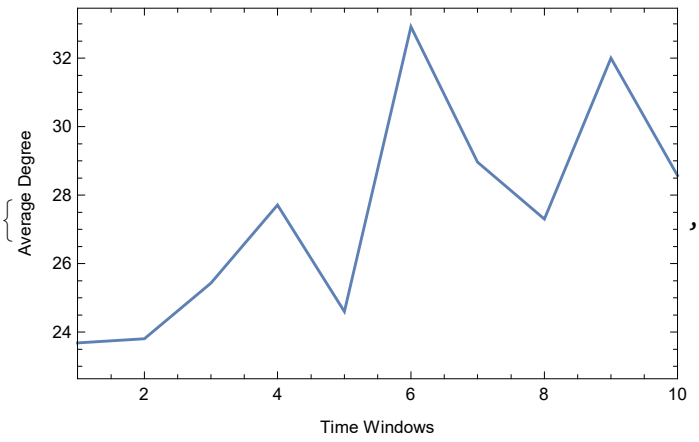
```
Out[ ]= {244.235, {{33.57, 40.9554, -19.8872}, {40.4218, 47.9585, -50.5403},
          {39.0145, 45.3088, -56.4996}, {43.6673, 49.1992, -28.4451},
          {41.7119, 47.7857, -37.8848}, {22.2202, 28.5683, -63.0805}, {35.7747, 41.8342, -58.0281},
          {39.7642, 47.8633, -39.7476}, {52.156, 60.1734, -34.2295}, {55.885, 59.8185, -23.5193}}}
```

```
In[ ]:= Table[Length@FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
         {i, Length@graphsandnodenumbers}]
```

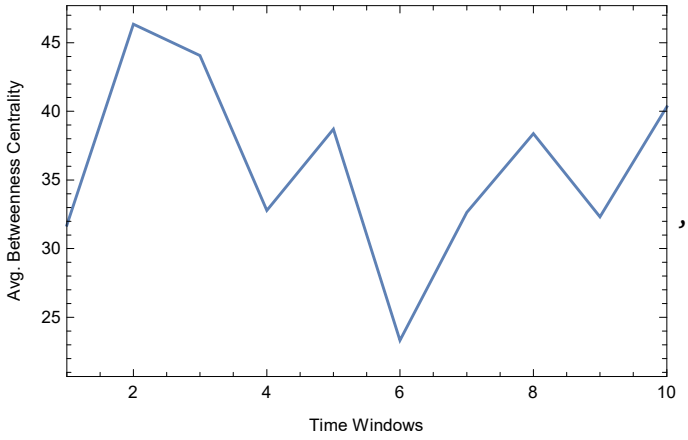
```
Out[ ]= {3, 3, 3, 3, 2, 2, 3, 2, 2, 2}
```

```
In[ ]:= {ListLinePlot[Thread[{Range@10, degreevalues}], Frame → True,
          FrameLabel → {"Time Windows", "Average Degree"}, ImageSize → 350,
          PlotRange → {{1, 10}, All}], ListLinePlot[Thread[{Range@10, ABCvalues}],
          Frame → True, FrameLabel → {"Time Windows", "Avg. Betweenness Centrality"},
          ImageSize → 350, PlotRange → {{1, 10}, All}],
         ListLinePlot[Thread[{Range@10, modularityvalues}], Frame → True,
          FrameLabel → {"Time Windows", "Modularity"}, ImageSize → 350, PlotRange → All],
         ListLinePlot[{Thread[{Range@10, singlerandomerdrenmodularityvalues}],
           Thread[{Range@10, singlerandomdegreesfxdmodularityvalues}],
           Thread[{Range@10, singlerandomcommmodularityvalues}]}, Frame → True,
          FrameLabel → {"Time Windows", "Modularity (Single Random Graphs)"},
          ImageSize → 350, PlotRange → {{1, 10}, All}, PlotLabels →
           Placed[{"Erdös-Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Below}]],
         ListLinePlot[{Thread[{Range@10, ZscoresmodularityWolf[[All, 1]]}],
           Thread[{Range@10, ZscoresmodularityWolf[[All, 2]]}],
           Thread[{Range@10, ZscoresmodularityWolf[[All, 3]]}]},
          Frame → True, FrameLabel → {"Time Windows", "Z-Score for Modularity"},
          ImageSize → 350, PlotRange → All, PlotLabels →
           Placed[{"Erdös Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]]}
```

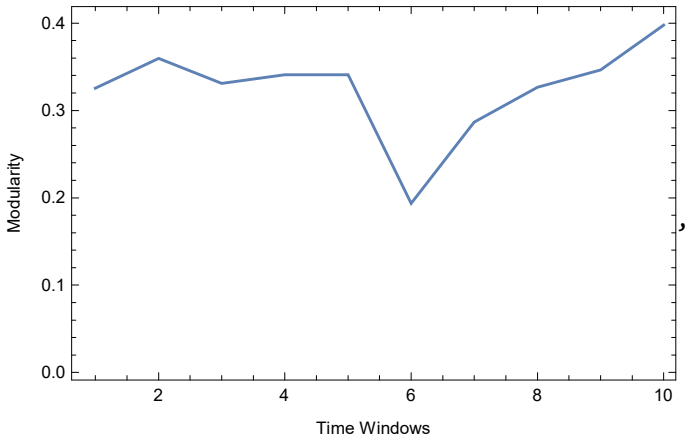
```
Out[ ]=
```



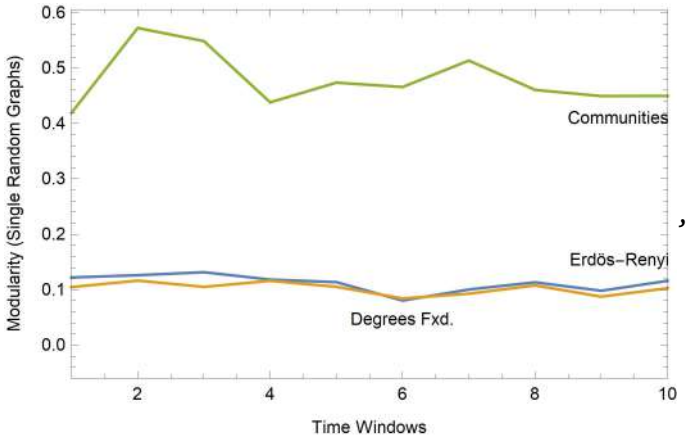
,


Avg. Betweenness Centrality
45
40
35
30
25
2
4
6
8
10
Time Windows

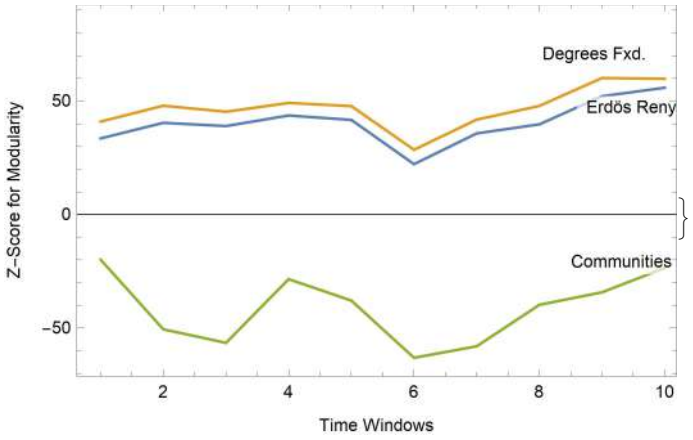
,


Modularity
0.4
0.3
0.2
0.1
0.0
2
4
6
8
10
Time Windows

,


Modularity (Single Random Graphs)
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Communities
Erdös–Renyi
Degrees Fxd.
2
4
6
8
10
Time Windows

,



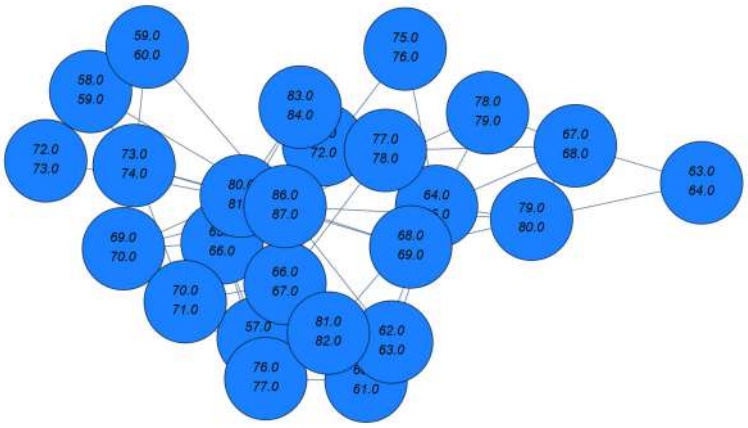
Thickness Feature

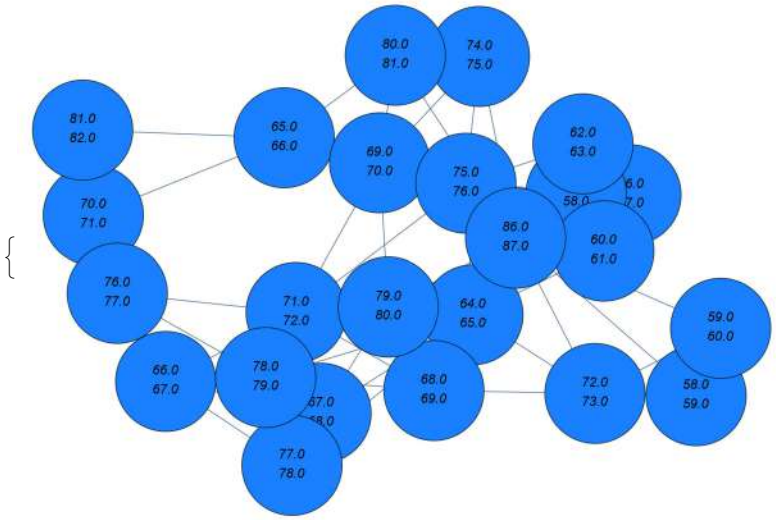
In[ ]:=
```
AbsoluteTiming[
 thicknessdataintimewindowsFixedstep = snetworkdatabinnedintimewindows[10, 1, 10];]
```

Out[ ]= {13.7807, Null}

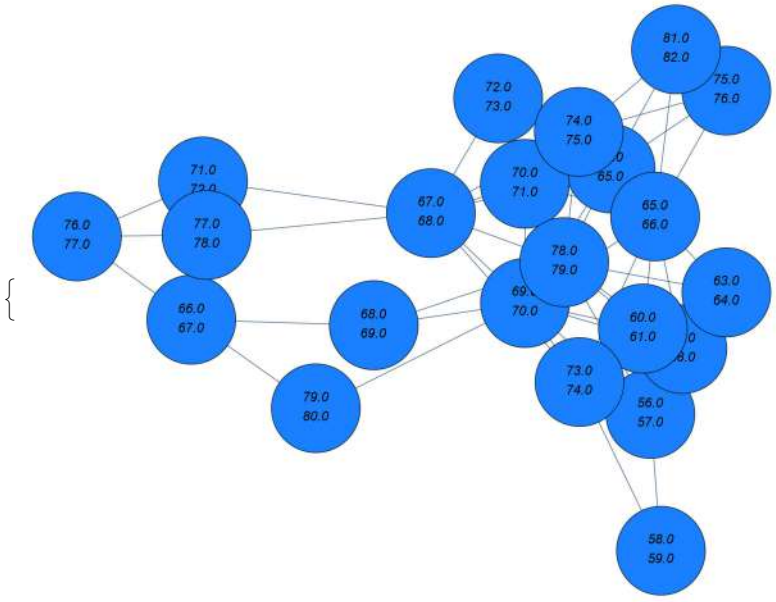
In[ ]:=
```
graphsandnodenumbers = Table[snetworkgraph[thicknessdataintimewindowsFixedstep[[1]][[i]],
    thicknessdataintimewindowsFixedstep[[2]][[i]],
    2, 7, 400, RGBColor[0.1, 0.5, 1.]], {i, Range@10}];
```

In[ ]:=
```
graphsandnodenumbers
```

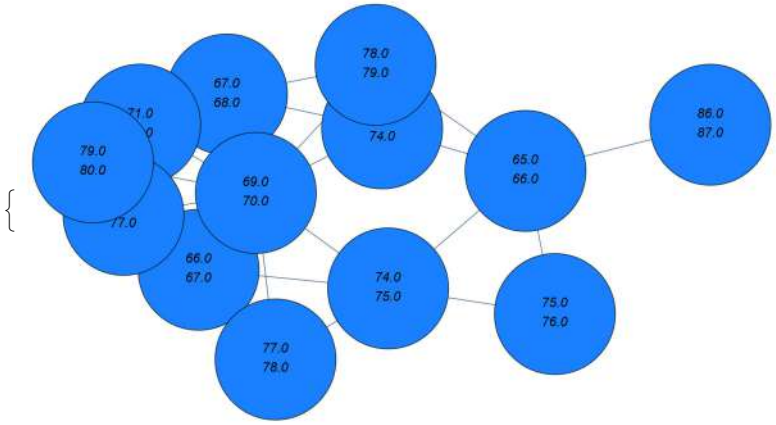
Out[ ]= {{  , 25},


80.0
81.0
74.0
75.0
81.0
82.0
65.0
66.0
69.0
70.0
62.0
63.0
75.0
76.0
58.0
70.0
71.0
86.0
87.0
60.0
61.0
76.0
77.0
71.0
72.0
79.0
80.0
64.0
65.0
59.0
60.0
66.0
67.0
78.0
79.0
68.0
69.0
72.0
73.0
58.0
59.0
77.0
78.0

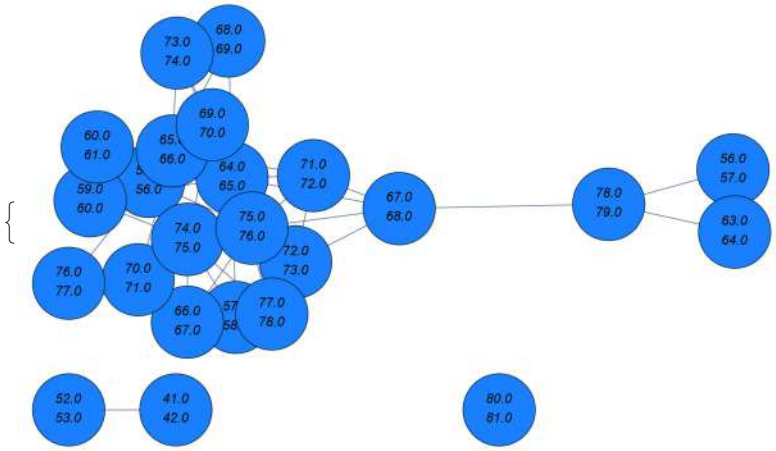
, 24},


81.0
82.0
72.0
73.0
75.0
76.0
74.0
75.0
71.0
70.0
71.0
65.0
67.0
68.0
65.0
66.0
76.0
77.0
77.0
78.0
78.0
79.0
63.0
64.0
66.0
67.0
68.0
69.0
60.0
61.0
73.0
74.0
79.0
80.0
56.0
57.0
58.0
59.0

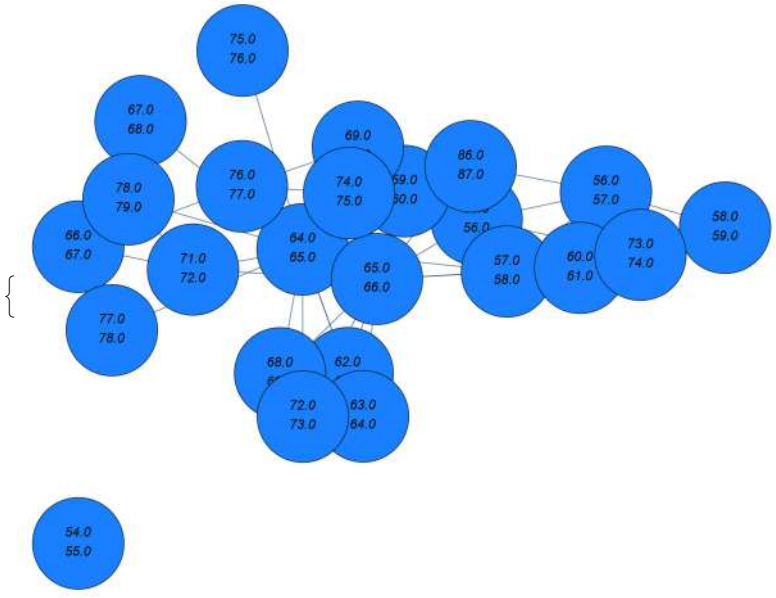
, 22},


78.0
79.0
67.0
68.0
74.0
86.0
87.0
65.0
66.0
79.0
80.0
69.0
70.0
66.0
67.0
74.0
75.0
75.0
76.0
77.0
78.0

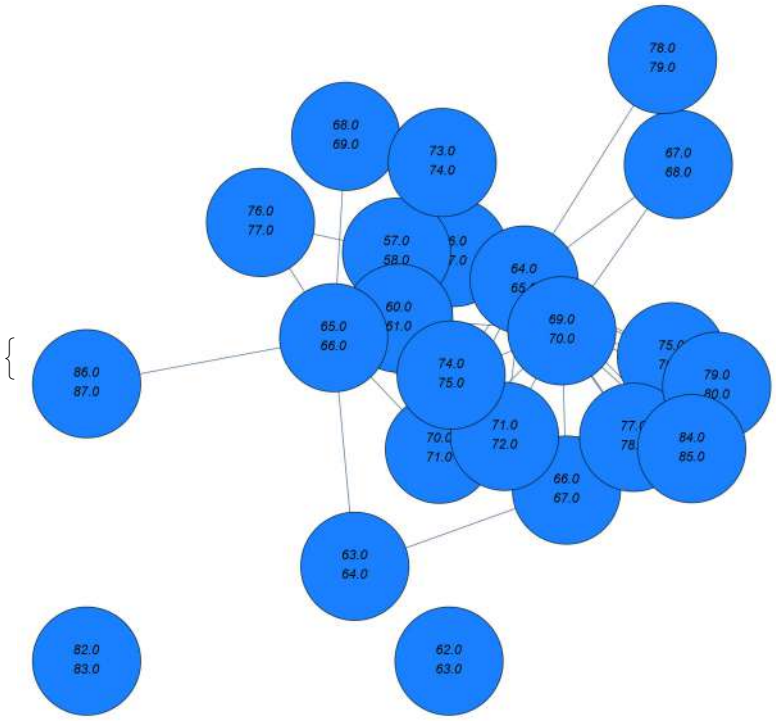
{ , 13},


73.0
74.0
68.0
69.0
69.0
70.0
60.0
61.0
64.0
65.0
71.0
72.0
67.0
68.0
78.0
79.0
56.0
57.0
59.0
60.0
56.0
74.0
75.0
75.0
76.0
63.0
64.0
72.0
73.0
76.0
77.0
70.0
71.0
66.0
67.0
77.0
78.0
52.0
53.0
41.0
42.0
80.0
81.0

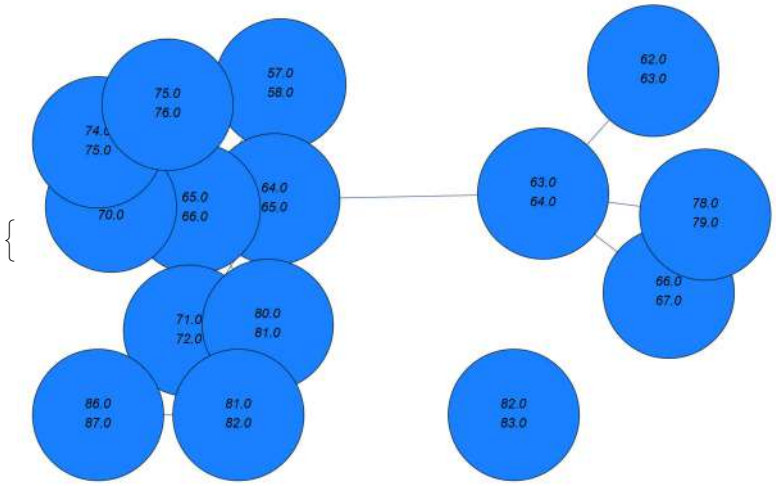
{ , 24},


75.0
76.0
67.0
68.0
69.0
86.0
87.0
78.0
79.0
76.0
77.0
74.0
75.0
56.0
57.0
58.0
59.0
66.0
67.0
64.0
65.0
56.0
73.0
74.0
71.0
72.0
65.0
66.0
57.0
58.0
60.0
61.0
77.0
78.0
68.0
62.0
72.0
73.0
63.0
64.0
54.0
55.0

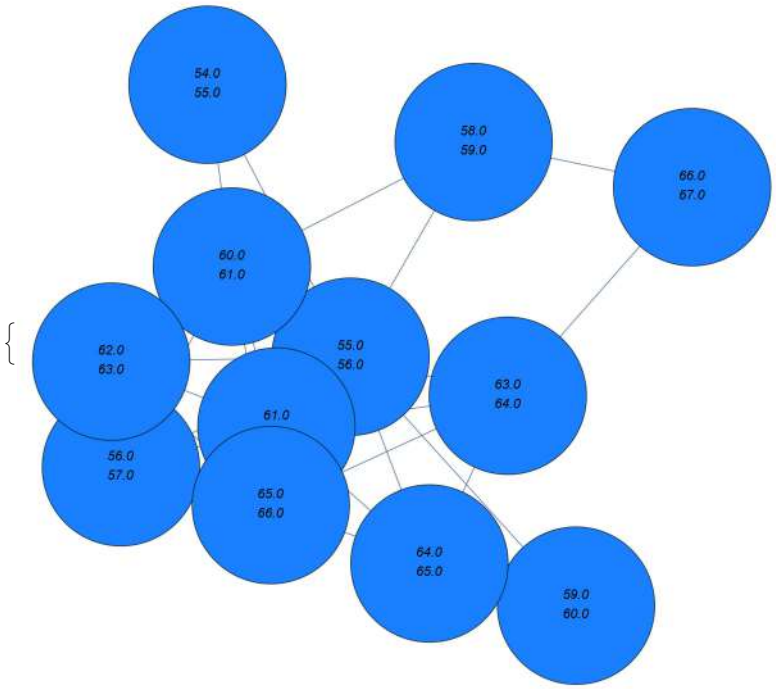
{ , 24},


78.0
79.0
68.0
69.0
73.0
74.0
67.0
68.0
76.0
77.0
57.0
58.0
64.0
65.0
60.0
61.0
65.0
66.0
69.0
70.0
74.0
75.0
79.0
80.0
70.0
71.0
71.0
72.0
84.0
85.0
66.0
67.0
63.0
64.0
86.0
87.0
82.0
83.0
62.0
63.0

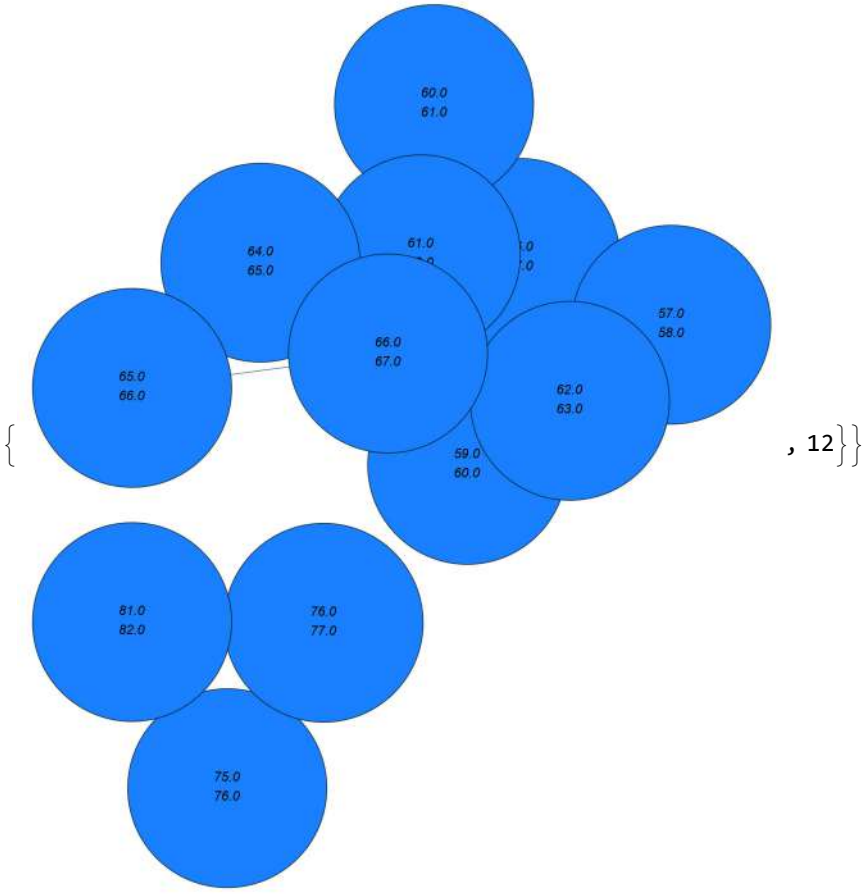
, 23},


57.0
58.0
75.0
76.0
74.0
75.0
64.0
65.0
65.0
66.0
70.0
62.0
63.0
63.0
64.0
78.0
79.0
66.0
67.0
71.0
72.0
80.0
81.0
86.0
87.0
81.0
82.0
82.0
83.0

, 15},


54.0
55.0
58.0
59.0
66.0
67.0
60.0
61.0
62.0
63.0
55.0
56.0
63.0
64.0
61.0
56.0
57.0
65.0
66.0
64.0
65.0
59.0
60.0

{ , 12},



{ , 12}}

In[ ]:=
```
ABCvalues = Table[Mean@BetweennessCentrality[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
      FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
      "Normalized" → False], {i, Length@graphsandnodenumbers}];
degreevalues = Table[N@Mean@VertexDegree[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
```

In[ ]:=
```
singlerandomgraphserdren = Table[
    RandomGraph[{VertexCount[i], EdgeCount[i]}], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomerdrenmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphserdren[[i]],
     FindGraphCommunities[singlerandomgraphserdren[[i]]], "Normalized" -> False],
   {i, Length@singlerandomgraphserdren}];
singlerandomgraphsdegreesfxd = Table[IGDegreeSequenceGame[Total[AdjacencyMatrix@i],
Method -> "VigerLatapy"], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomdegreesfxdmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphsdegreesfxd[[i]],
     FindGraphCommunities[singlerandomgraphsdegreesfxd[[i]]], "Normalized" -> False],
   {i, Length@singlerandomgraphsdegreesfxd}];
singlerandomgraphscomm = Table[randomizedgraphamongcommunities[i],
   {i, graphsandnodenumbers[[All, 1]]}];
singlerandomcommmodularityvalues = Table[N@GraphAssortativity[
     singlerandomgraphscomm[[i]], FindGraphCommunities[singlerandomgraphscomm[[i]]],
     "Normalized" -> False], {i, Length@singlerandomgraphscomm}];
```

In[ ]:=
```
AbsoluteTiming[ZscoresmodularityWolf =
  Table[randomnessfunctionformodularity[i, "Wolf"], {i, graphsandnodenumbers[[All, 1]]}]]
```

Out[ ]=
```
{93.2888,
 {{2.50502, 4.54607, -6.06845}, {4.11302, 5.20171, -6.38714}, {4.3782, 6.11277, -4.85},
  {0.716177, 1.36241, -3.21859}, {0.134648, ComplexInfinity, -4.1646},
  {<<1>>}, {-0.474878, ComplexInfinity, -6.80042},
  {0.585053, ComplexInfinity, 30 Sqrt[1110] (0.34814 + <<1>>/1000)/Sqrt[<<1>>]},
  {-0.897705, 0.446371, -3.28261}, {1.67402, 1.84314, -1.75217}}}
```

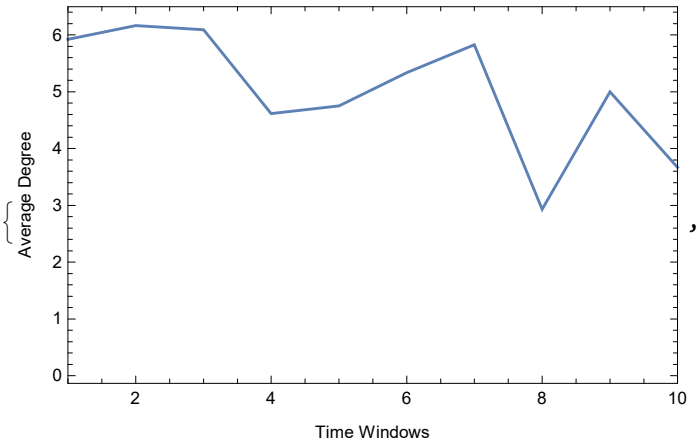
large output | show less | show more | show all | set size limit...

In[ ]:=
```
Table[Length@FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
 {i, Length@graphsandnodenumbers}]
```

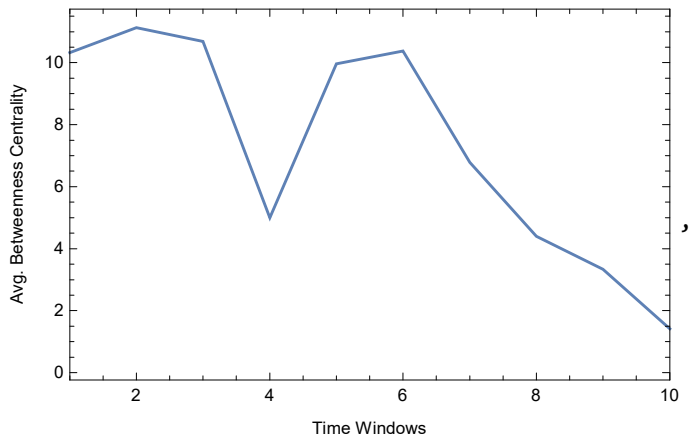
Out[ ]=
```
{4, 4, 3, 3, 5, 5, 5, 4, 2, 3}
```

In[ ]:=
```
{ListLinePlot[Thread[{Range@10, degreevalues}], Frame → True,
  FrameLabel → {"Time Windows", "Average Degree"}, ImageSize → 350,
  PlotRange → {{1, 10}, All}], ListLinePlot[Thread[{Range@10, ABCvalues}],
  Frame → True, FrameLabel → {"Time Windows", "Avg. Betweenness Centrality"},
  ImageSize → 350, PlotRange → {{1, 10}, All}],
 ListLinePlot[Thread[{Range@10, modularityvalues}], Frame → True,
  FrameLabel → {"Time Windows", "Modularity"}, ImageSize → 350, PlotRange → All],
 ListLinePlot[{Thread[{Range@10, singlerandomerdrenmodularityvalues}],
   Thread[{Range@10, singlerandomdegreesfxdmodularityvalues}],
   Thread[{Range@10, singlerandomcommmodularityvalues}]}, Frame → True,
  FrameLabel → {"Time Windows", "Modularity (Single Random Graphs)"},
  ImageSize → 350, PlotRange → {{1, 10}, All}, PlotLabels →
   Placed[{"Erdös-Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Below}]],
 ListLinePlot[{Thread[{Range@10, ZscoresmodularityWolf[[All, 1]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 2]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 3]]}]},
  Frame → True, FrameLabel → {"Time Windows", "Z-Score for Modularity"},
  ImageSize → 350, PlotRange → All, PlotLabels →
   Placed[{"Erdös Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]]}
```

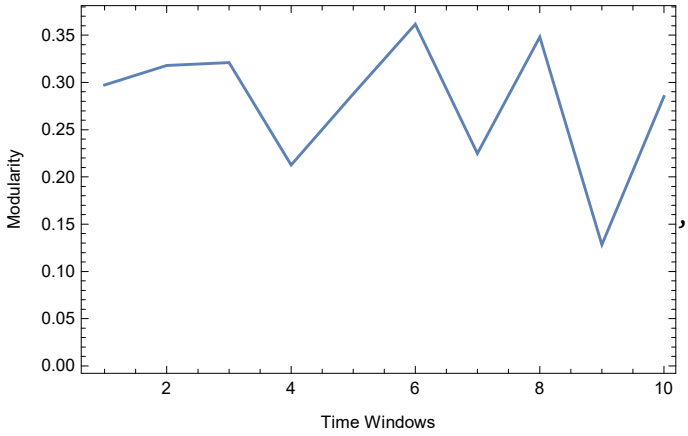
Out[ ]=

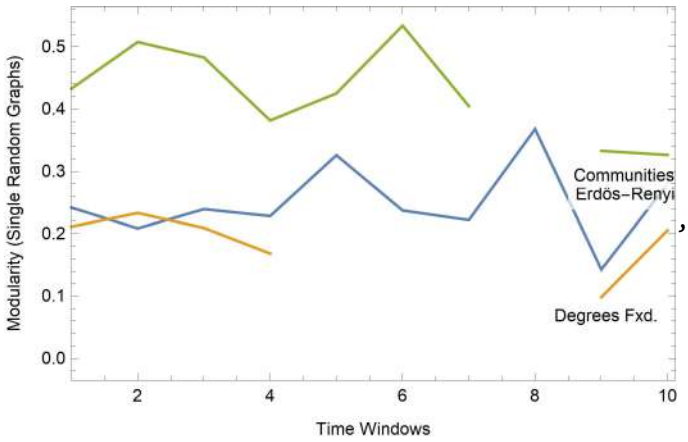


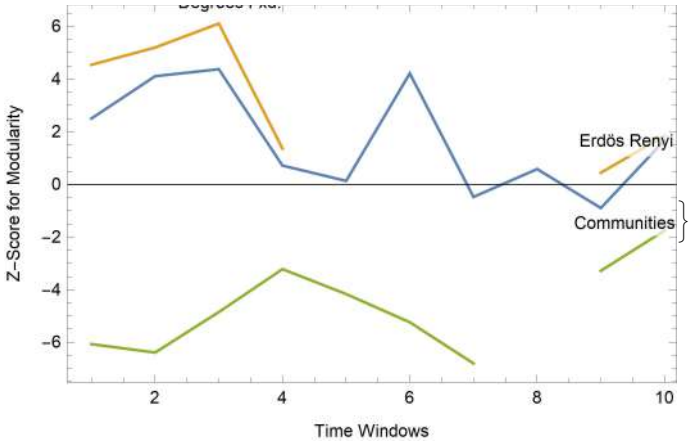
,



,







Length Feature

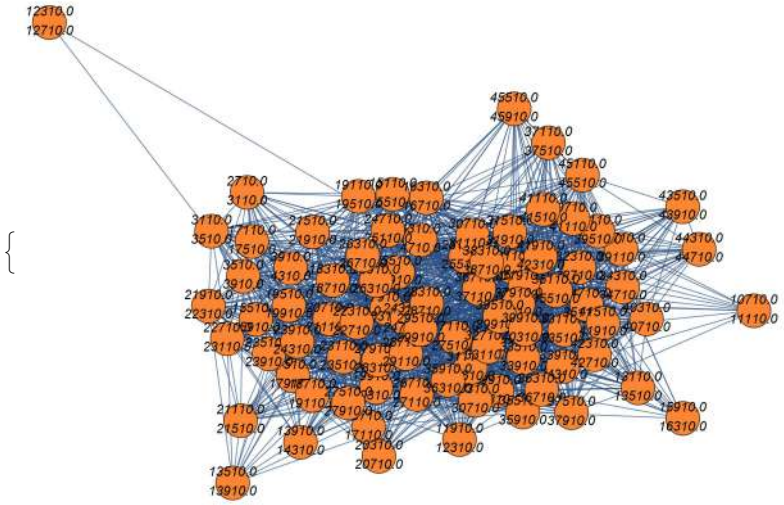
In[ ]:=
```
AbsoluteTiming[
 lengthdataintimewindowsFixedstep = snetworkdatabinnedintimewindows[12, 400, 10];]
```

Out[ ]= {61.2188, Null}

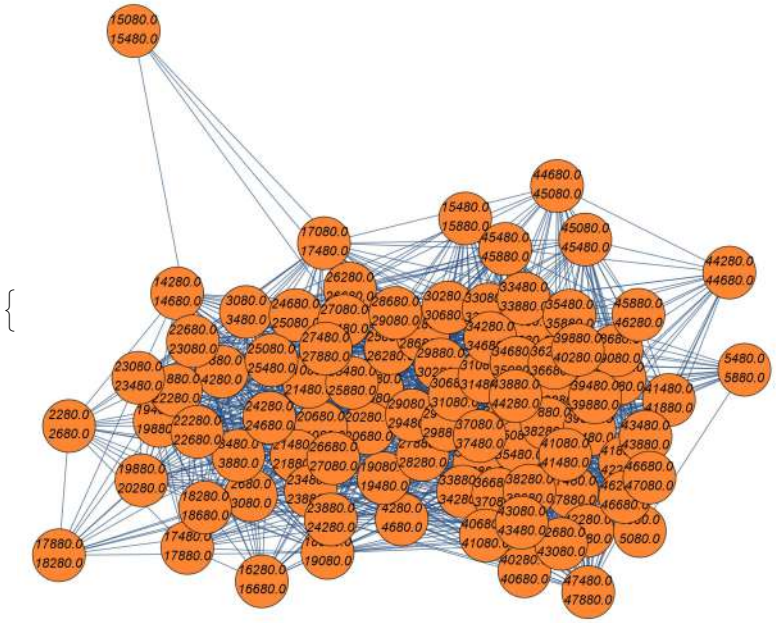
In[ ]:=
```
graphsandnodenumbers = Table[snetworkgraph[lengthdataintimewindowsFixedstep[[1]][[i]],
    lengthdataintimewindowsFixedstep[[2]][[i]], 2,
    7, 400, RGBColor[1., 0.53, 0.2]], {i, Range@10}];
```

In[ ]:=
```
graphsandnodenumbers
```

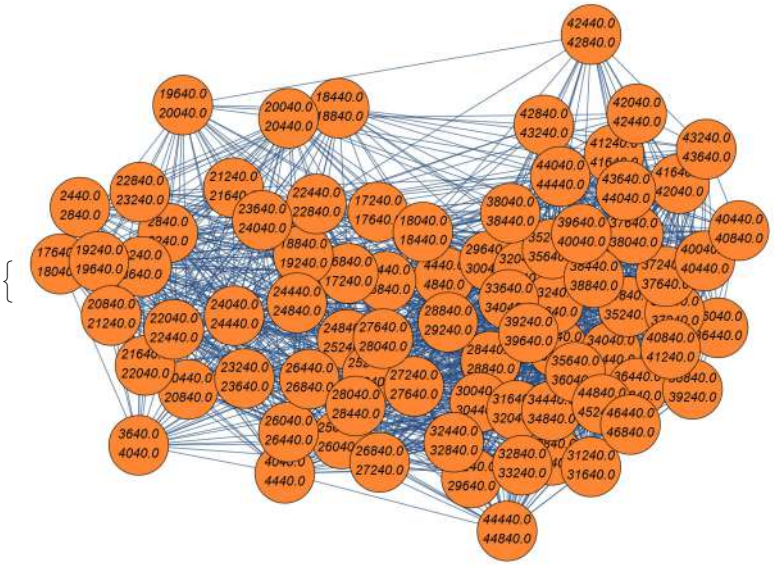
Out[ ]=



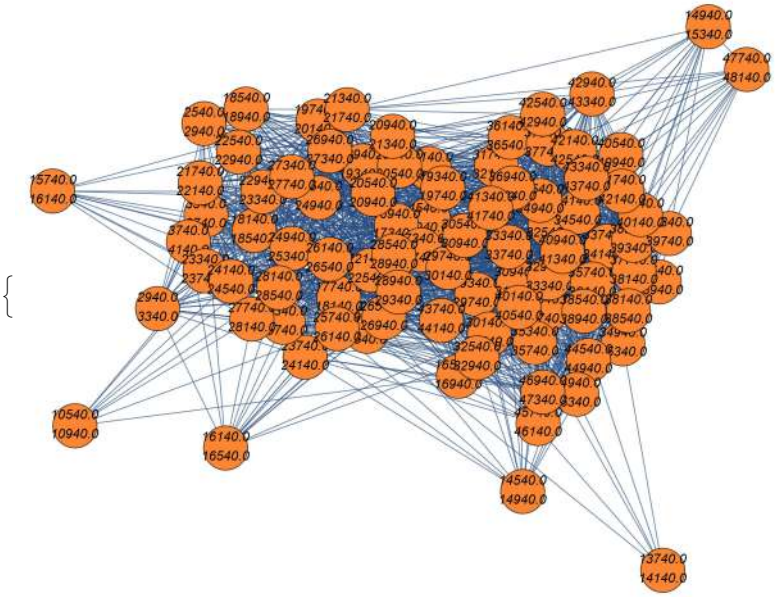
, 84},



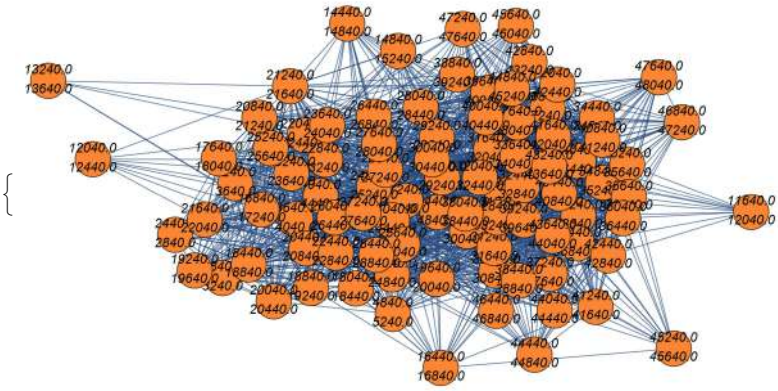
, 89},



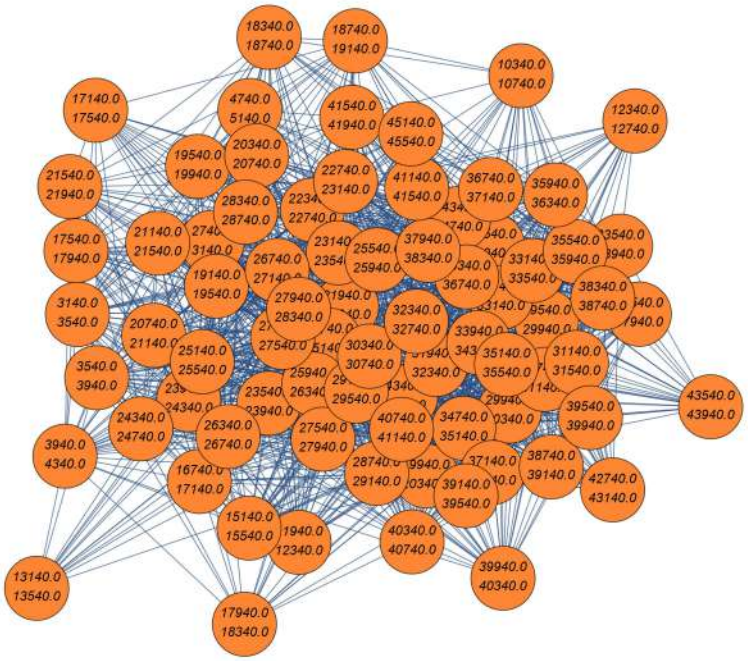
, 79},



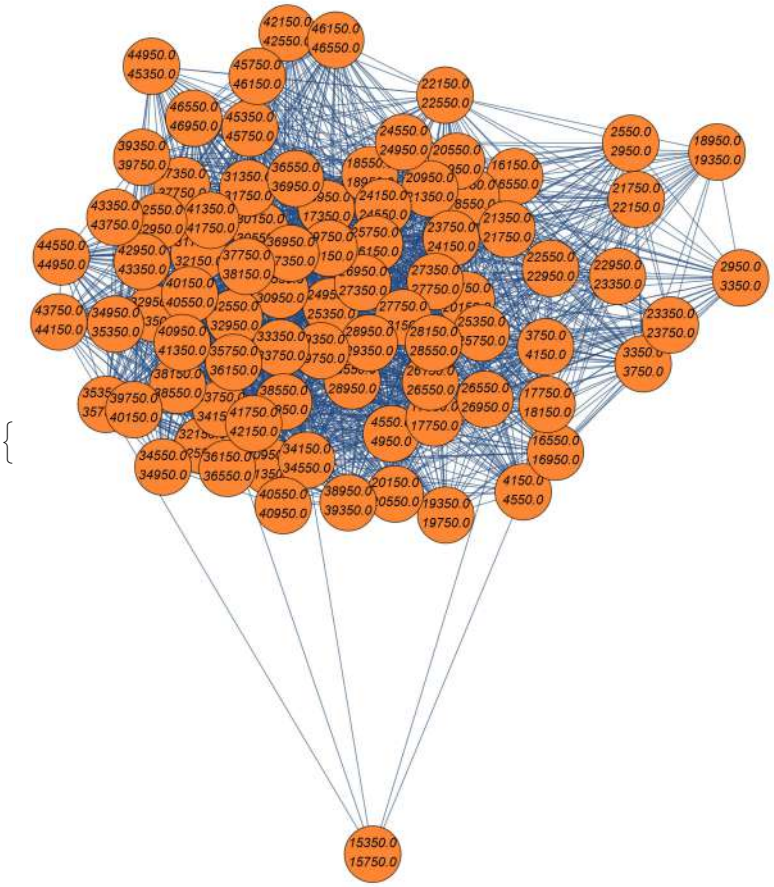
, 87},



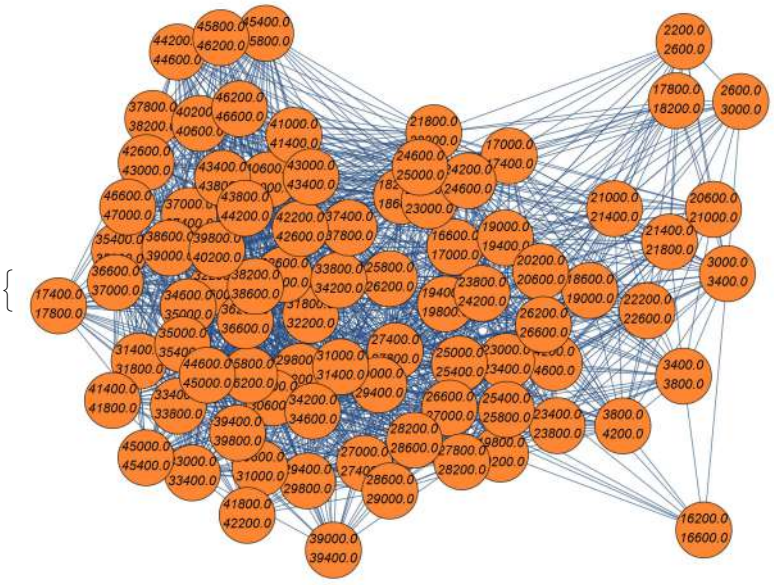
{ , 90},



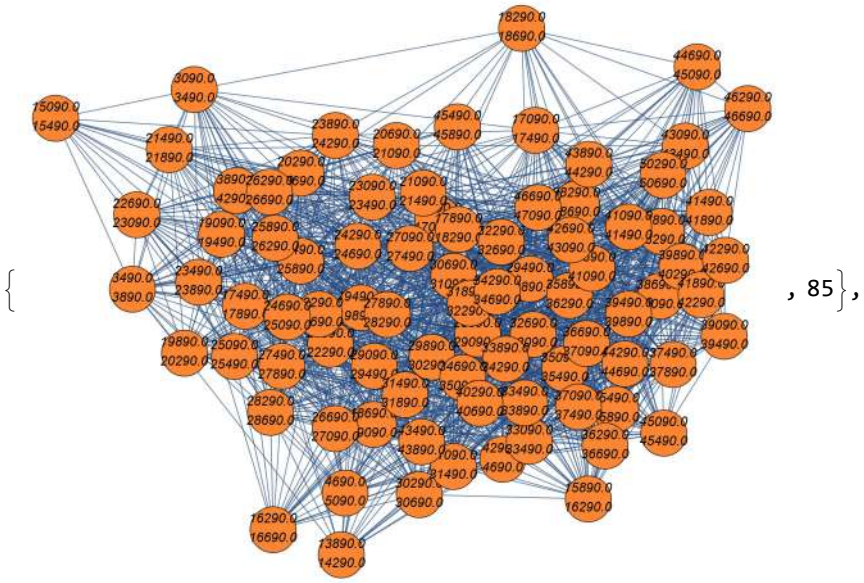
{ , 77},

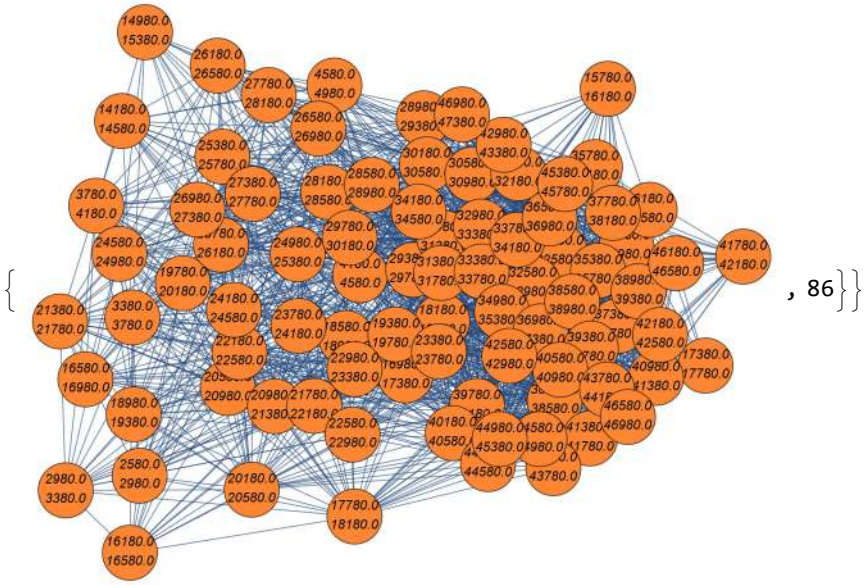


, 83},



, 83},



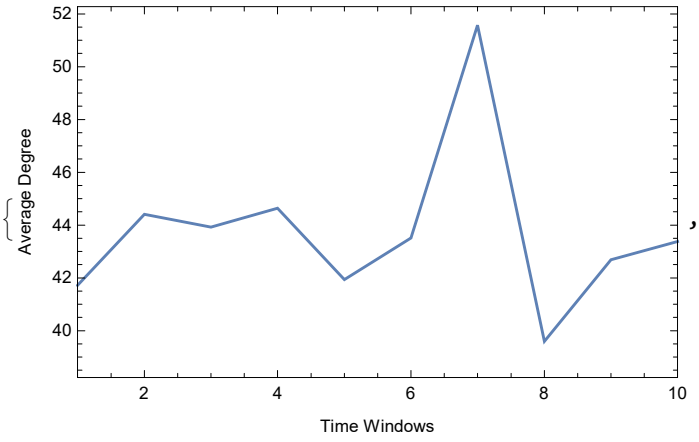


In[ ]:=
```
ABCvalues = Table[Mean@BetweennessCentrality[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
      FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
      "Normalized" → False], {i, Length@graphsandnodenumbers}];
degreevalues = Table[N@Mean@VertexDegree[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
```

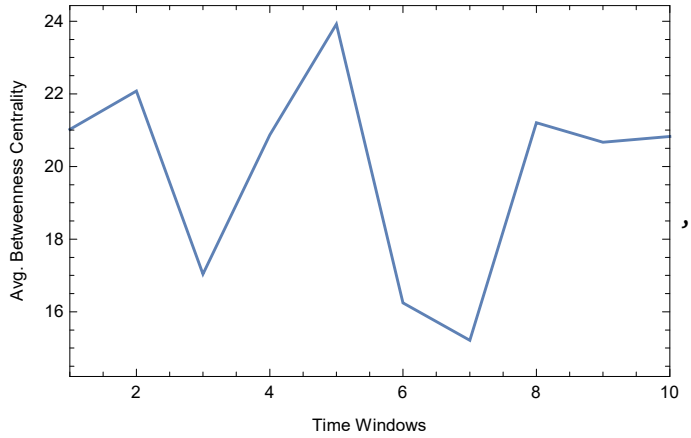
```
In[ ]:= singlerandomgraphserdren = Table[
          RandomGraph[{VertexCount[i], EdgeCount[i]}], {i, graphsandnodenumbers[[All, 1]]}];
        singlerandomerdrenmodularityvalues =
          Table[N@GraphAssortativity[singlerandomgraphserdren[[i]],
              FindGraphCommunities[singlerandomgraphserdren[[i]]], "Normalized" -> False],
            {i, Length@singlerandomgraphserdren}];
        singlerandomgraphsdegreesfxd = Table[IGDegreeSequenceGame[Total[AdjacencyMatrix@i],
        Method -> "VigerLatapy"], {i, graphsandnodenumbers[[All, 1]]}];
        singlerandomdegreesfxdmodularityvalues =
          Table[N@GraphAssortativity[singlerandomgraphsdegreesfxd[[i]],
              FindGraphCommunities[singlerandomgraphsdegreesfxd[[i]]], "Normalized" -> False],
            {i, Length@singlerandomgraphsdegreesfxd}];
        singlerandomgraphscomm = Table[randomizedgraphamongcommunities[i],
            {i, graphsandnodenumbers[[All, 1]]}];
        singlerandomcommmodularityvalues = Table[N@GraphAssortativity[
              singlerandomgraphscomm[[i]], FindGraphCommunities[singlerandomgraphscomm[[i]]],
              "Normalized" -> False], {i, Length@singlerandomgraphscomm}];

In[ ]:= AbsoluteTiming[ZscoresmodularityWolf =
          Table[randomnessfunctionformodularity[i, "Wolf"], {i, graphsandnodenumbers[[All, 1]]}]]

Out[ ]= {530.131, {{35.7081, 45.2461, -78.1516}, {41.6224, 48.2656, -78.9717},
          {33.8486, 37.5899, -75.6403}, {48.8782, 55.501, -79.2671}, {39.6166, 47.0115, -76.2427},
          {24.1346, 30.4739, -84.9721}, {24.3581, 29.614, -107.266}, {30.2289, 35.8713, -71.9001},
          {33.6259, 36.8646, -79.0163}, {35.8483, 40.8516, -73.0956}}}

In[ ]:= Table[Length@FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
          {i, Length@graphsandnodenumbers}]

Out[ ]= {2, 2, 2, 2, 2, 2, 2, 2, 3, 2}
```

In[ ]:=
```
{ListLinePlot[Thread[{Range@10, degreevalues}], Frame → True,
  FrameLabel → {"Time Windows", "Average Degree"}, ImageSize → 350,
  PlotRange → {{1, 10}, All}], ListLinePlot[Thread[{Range@10, ABCvalues}],
  Frame → True, FrameLabel → {"Time Windows", "Avg. Betweenness Centrality"},
  ImageSize → 350, PlotRange → {{1, 10}, All}],
 ListLinePlot[Thread[{Range@10, modularityvalues}], Frame → True,
  FrameLabel → {"Time Windows", "Modularity"}, ImageSize → 350, PlotRange → All],
 ListLinePlot[{Thread[{Range@10, singlerandomerdrenmodularityvalues}],
   Thread[{Range@10, singlerandomdegreesfxdmodularityvalues}],
   Thread[{Range@10, singlerandomcommmodularityvalues}]}, Frame → True,
  FrameLabel → {"Time Windows", "Modularity (Single Random Graphs)"},
  ImageSize → 350, PlotRange → {{1, 10}, All}, PlotLabels →
   Placed[{"Erdös-Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Below}]],
 ListLinePlot[{Thread[{Range@10, ZscoresmodularityWolf[[All, 1]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 2]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 3]]}]},
  Frame → True, FrameLabel → {"Time Windows", "Z-Score for Modularity"},
  ImageSize → 350, PlotRange → All, PlotLabels →
   Placed[{"Erdös Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]]}
```

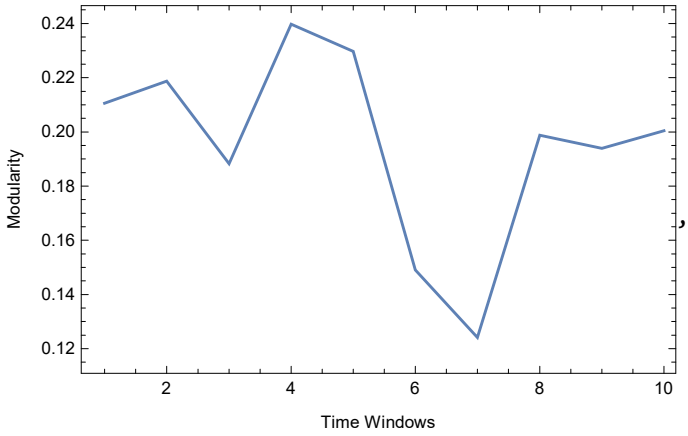
Out[ ]=



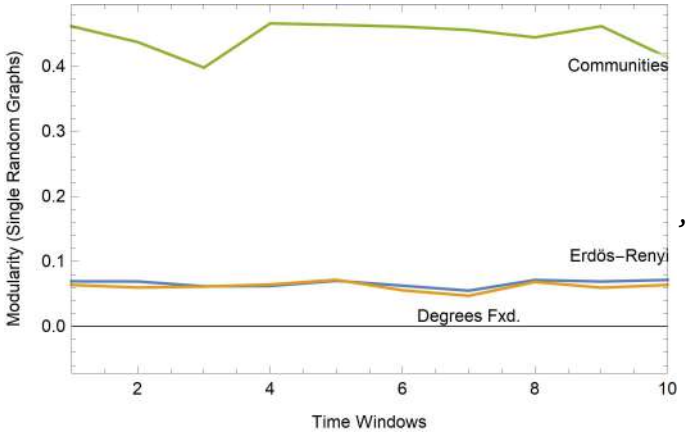
,



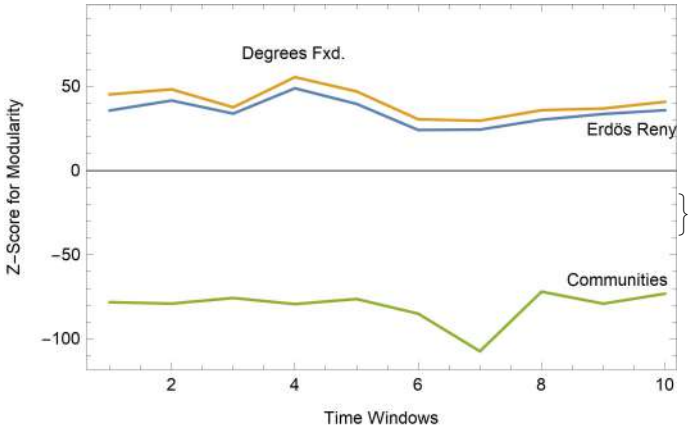
,



,



,



}

Weight Feature

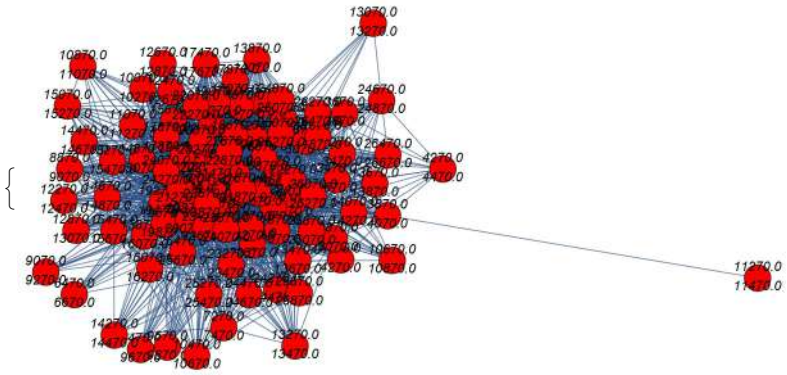
In[ ]:=
```
AbsoluteTiming[
 weightdataintimewindowsFixedstep = snetworkdatabinnedintimewindows[11, 200, 10];]
```

Out[ ]= {77.0267, Null}

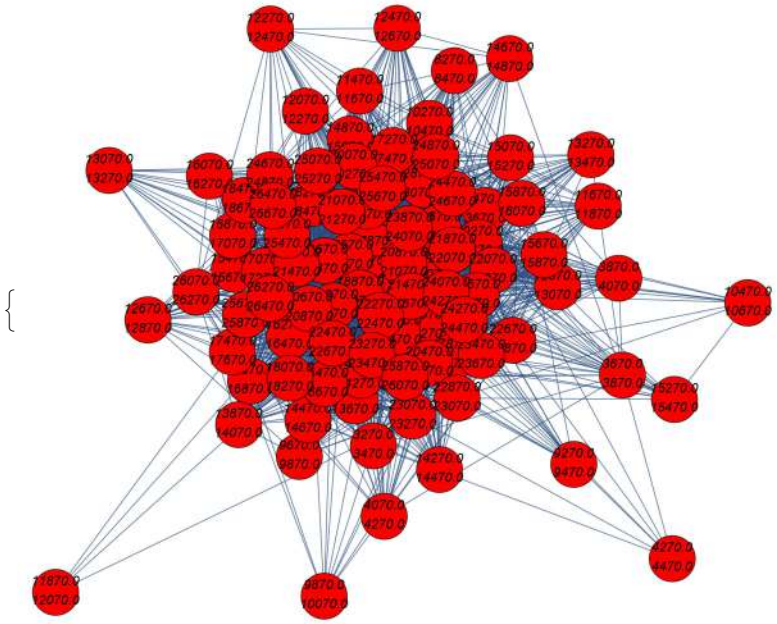
In[ ]:=
```
graphsandnodenumbers = Table[snetworkgraph[weightdataintimewindowsFixedstep[[1]][[i]],
    weightdataintimewindowsFixedstep[[2]][[i]], 2, 7, 400, Red], {i, Range@10}];
```

In[ ]:=
```
graphsandnodenumbers
```

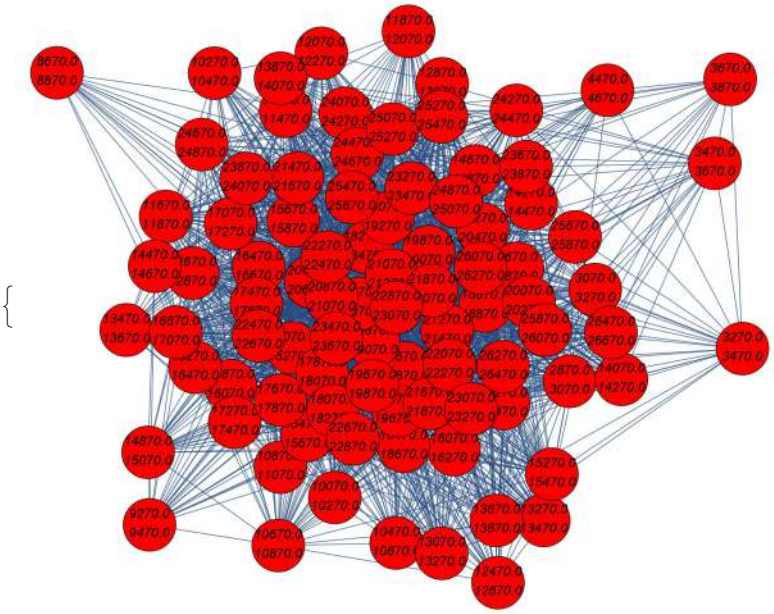
Out[ ]=



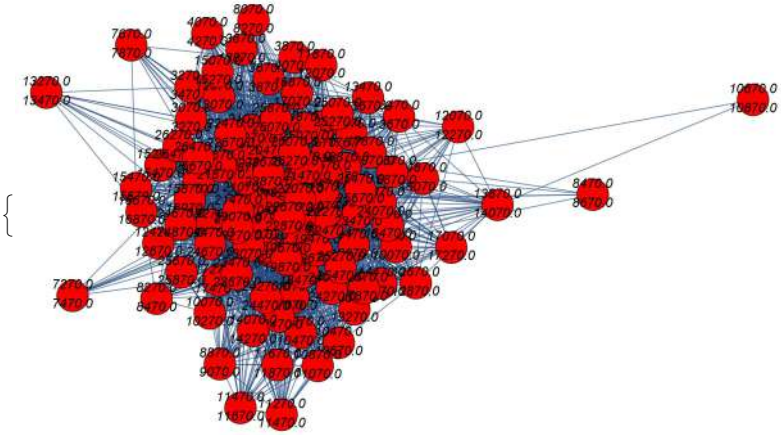
, 94},



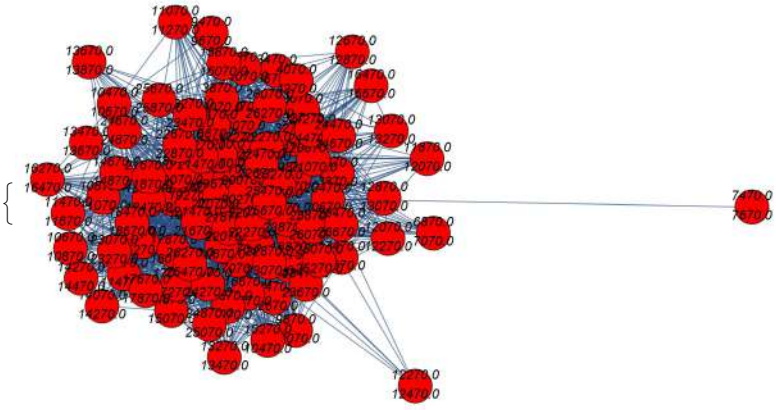
, 90},



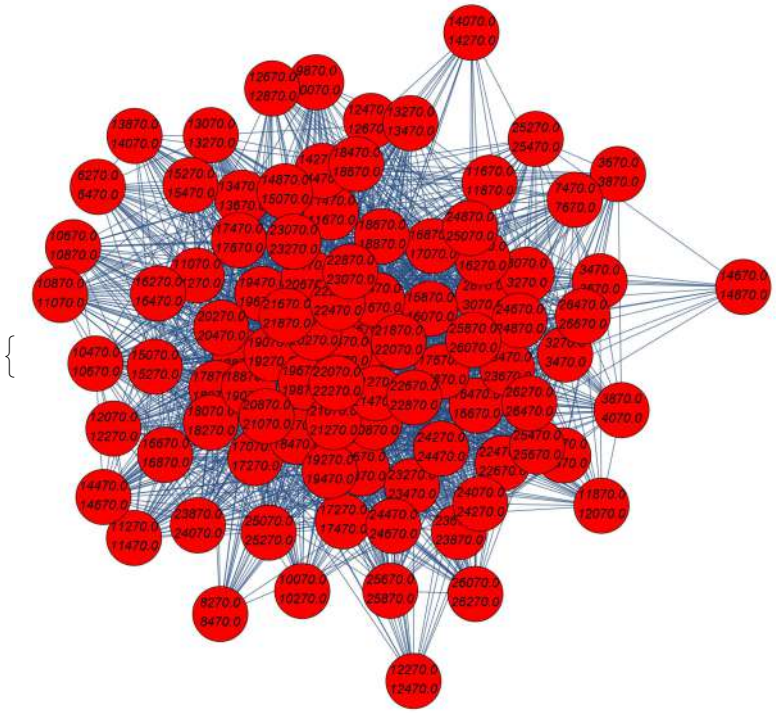
{ , 89},



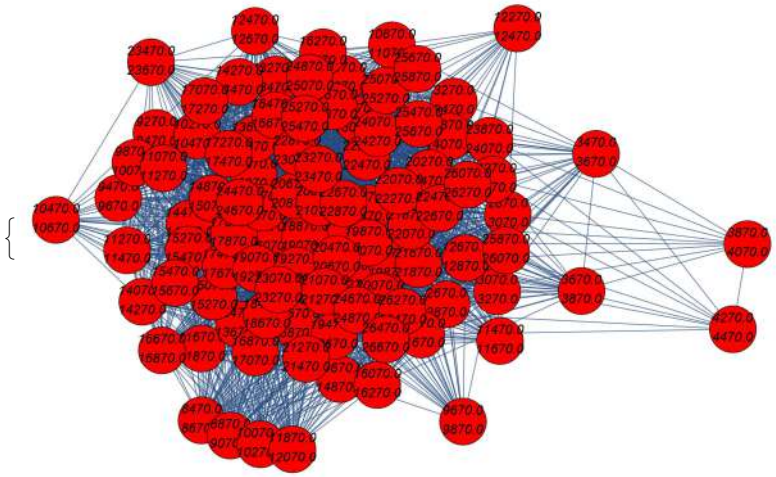
{ , 97},



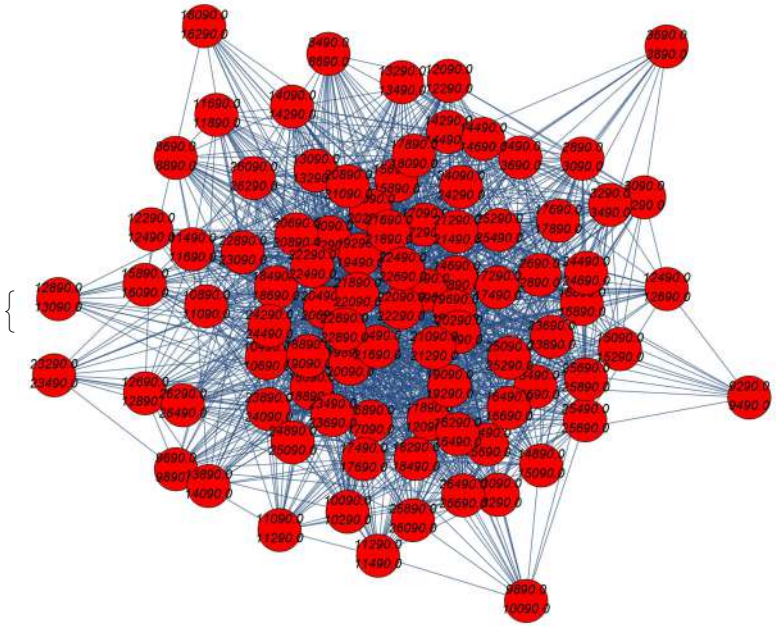
{ , 91},



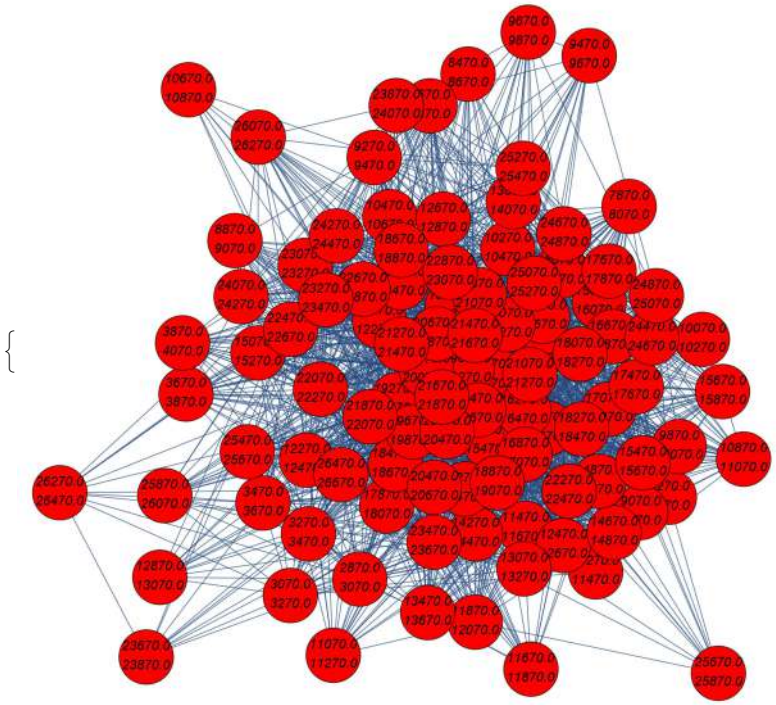
, 92},



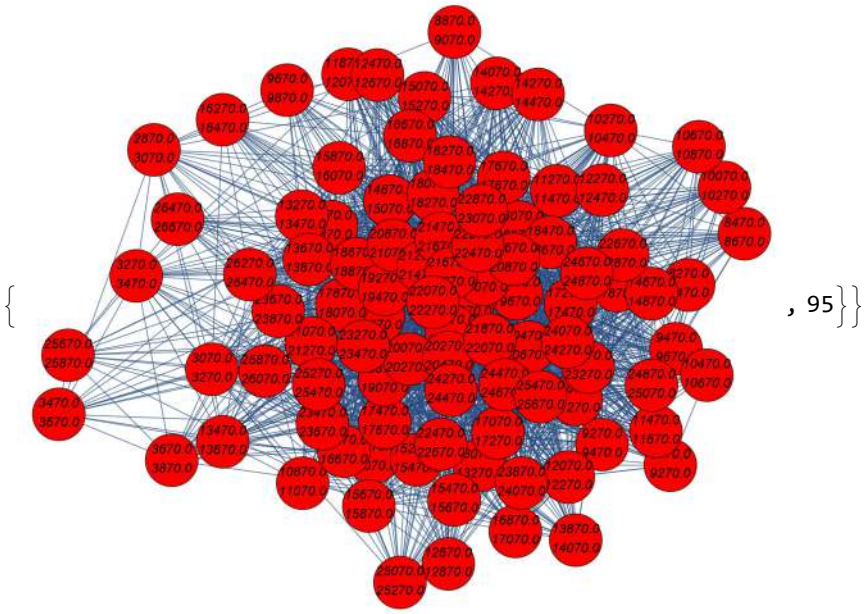
, 96},



, 89},



, 100},



In[ ]:=
```
ABCvalues = Table[Mean@BetweennessCentrality[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
      FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
      "Normalized" → False], {i, Length@graphsandnodenumbers}];
degreevalues = Table[N@Mean@VertexDegree[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
```

In[ ]:=
```
singlerandomgraphserdren = Table[
    RandomGraph[{VertexCount[i], EdgeCount[i]}], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomerdrenmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphserdren[[i]],
      FindGraphCommunities[singlerandomgraphserdren[[i]]], "Normalized" -> False],
    {i, Length@singlerandomgraphserdren}];
singlerandomgraphsdegreesfxd = Table[IGDegreeSequenceGame[Total[AdjacencyMatrix@i],
Method -> "VigerLatapy"], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomdegreesfxdmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphsdegreesfxd[[i]],
      FindGraphCommunities[singlerandomgraphsdegreesfxd[[i]]], "Normalized" -> False],
    {i, Length@singlerandomgraphsdegreesfxd}];
singlerandomgraphscomm = Table[randomizedgraphamongcommunities[i],
    {i, graphsandnodenumbers[[All, 1]]}];
singlerandomcommmodularityvalues = Table[N@GraphAssortativity[
      singlerandomgraphscomm[[i]], FindGraphCommunities[singlerandomgraphscomm[[i]]],
      "Normalized" -> False], {i, Length@singlerandomgraphscomm}];
```

In[ ]:=
```
AbsoluteTiming[ZscoresmodularityWolf =
  Table[randomnessfunctionformodularity[i, "Wolf"], {i, graphsandnodenumbers[[All, 1]]}]]
```

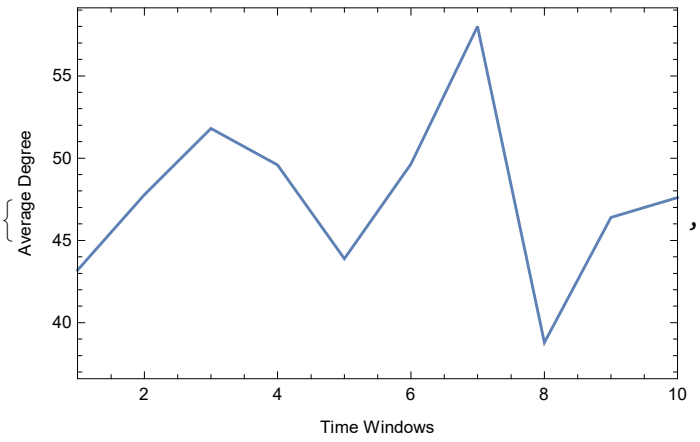
Out[•]= {674.48, {{7.62639, 14.3356, -82.984}, {1.14625, 5.60766, -109.218},
{3.50924, 7.04036, -114.502}, {8.94567, 14.6903, -120.16}, {7.66123, 14.1205, -99.8434},
{12.782, 18.6146, -111.619}, {7.75521, 10.747, -138.651}, {3.42677, 6.94266, -80.2143},
{3.2996, 7.69011, -94.6969}, {8.27245, 12.244, -117.493}}}

In[•]:=
```
Table[Length@FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
 {i, Length@graphsandnodenumbers}]
```

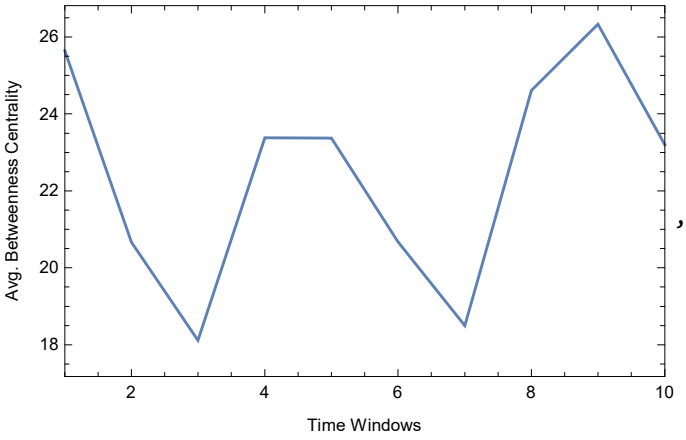
Out[•]= {3, 3, 3, 3, 3, 2, 3, 3, 3, 3}

In[•]:=
```
{ListLinePlot[Thread[{Range@10, degreevalues}], Frame → True,
  FrameLabel → {"Time Windows", "Average Degree"}, ImageSize → 350,
  PlotRange → {{1, 10}, All}], ListLinePlot[Thread[{Range@10, ABCvalues}],
  Frame → True, FrameLabel → {"Time Windows", "Avg. Betweenness Centrality"},
  ImageSize → 350, PlotRange → {{1, 10}, All}],
 ListLinePlot[Thread[{Range@10, modularityvalues}], Frame → True,
  FrameLabel → {"Time Windows", "Modularity"}, ImageSize → 350, PlotRange → All],
 ListLinePlot[{Thread[{Range@10, singlerandomerdrenmodularityvalues}],
   Thread[{Range@10, singlerandomdegreesfxdmodularityvalues}],
   Thread[{Range@10, singlerandomcommmodularityvalues}]}, Frame → True,
  FrameLabel → {"Time Windows", "Modularity (Single Random Graphs)"},
  ImageSize → 350, PlotRange → {{1, 10}, All}, PlotLabels →
   Placed[{"Erdös-Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]],
 ListLinePlot[{Thread[{Range@10, ZscoresmodularityWolf[[All, 1]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 2]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 3]]}]},
  Frame → True, FrameLabel → {"Time Windows", "Z-Score for Modularity"},
  ImageSize → 350, PlotRange → {All, {0, 20}}, PlotLabels →
   Placed[{"Erdös Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Below}]],
 ListLinePlot[{Thread[{Range@10, ZscoresmodularityWolf[[All, 1]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 2]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 3]]}]},
  Frame → True, FrameLabel → {"Time Windows", "Z-Score for Modularity"},
  ImageSize → 350, PlotRange → {All, {-150, 0}}, PlotLabels →
   Placed[{"Erdös Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Below}]]}
```

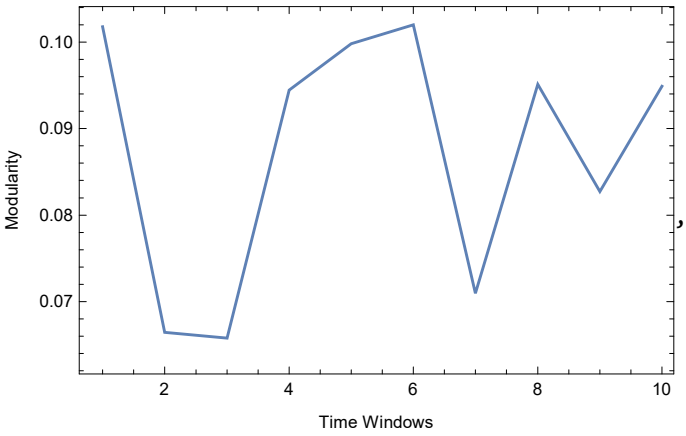
Out[•]=



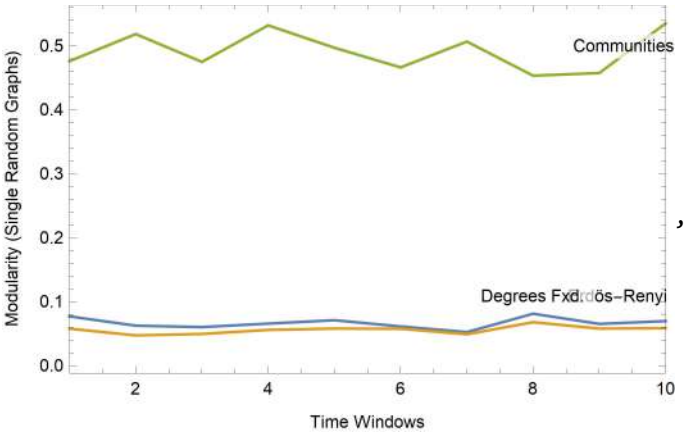
,


Avg. Betweenness Centrality
26
24
22
20
18
2
4
6
8
10
Time Windows

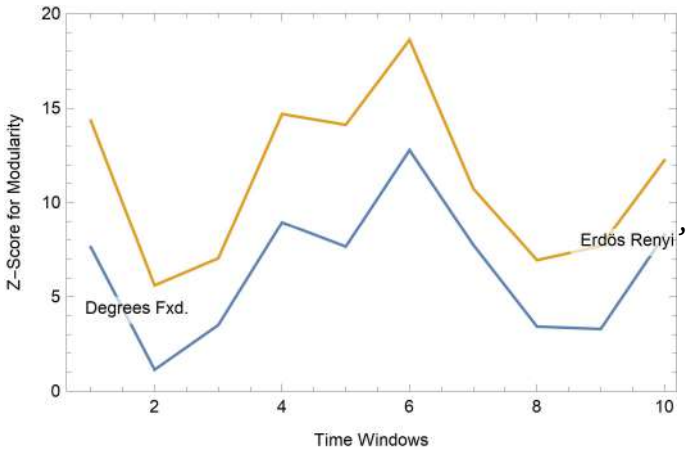
,

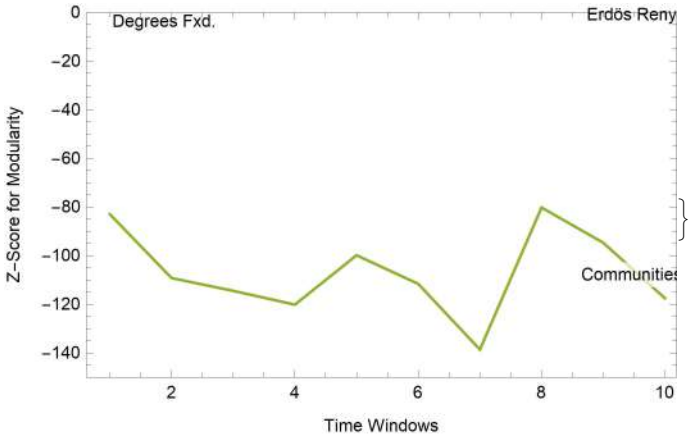

Modularity
0.10
0.09
0.08
0.07
2
4
6
8
10
Time Windows

,


Modularity (Single Random Graphs)
0.5
0.4
0.3
0.2
0.1
0.0
Communities
Degrees Fxd. Erdös–Renyi
2
4
6
8
10
Time Windows

,

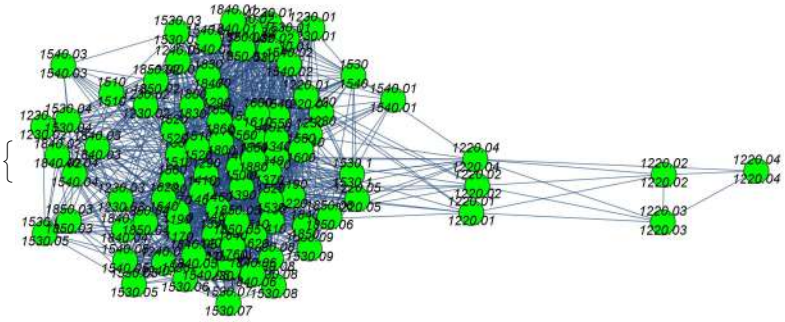


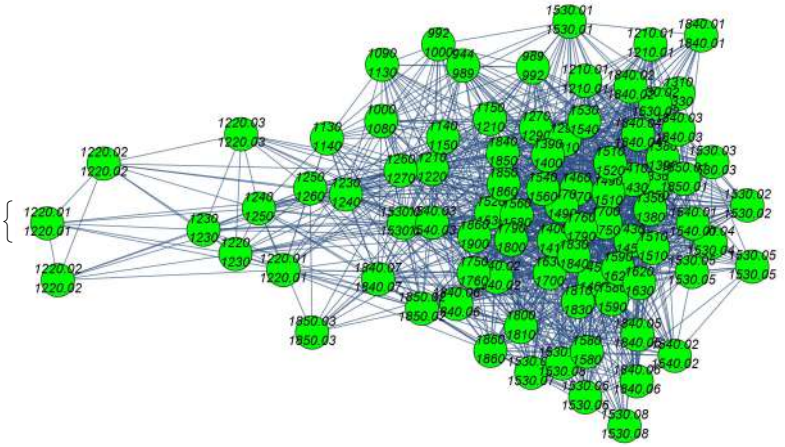


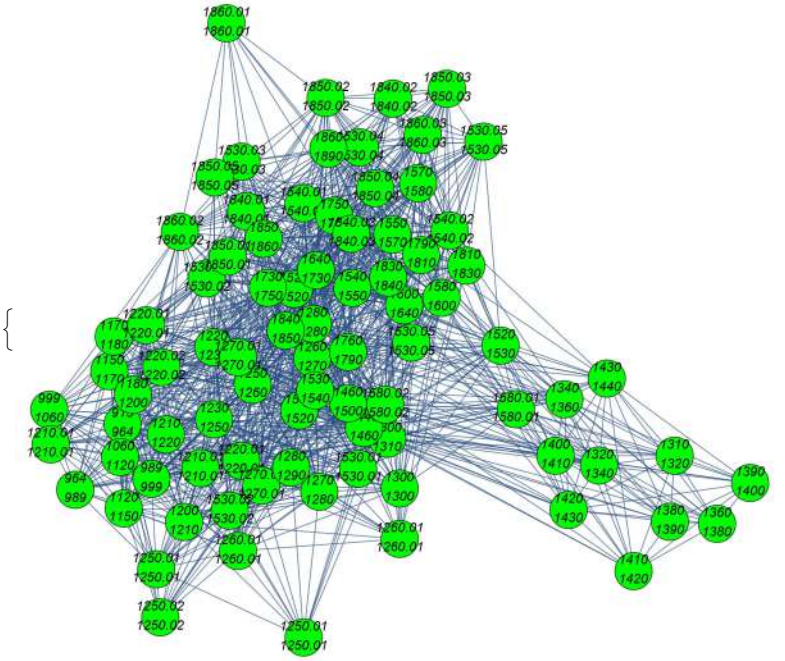
Fixed Bucket Size Networks

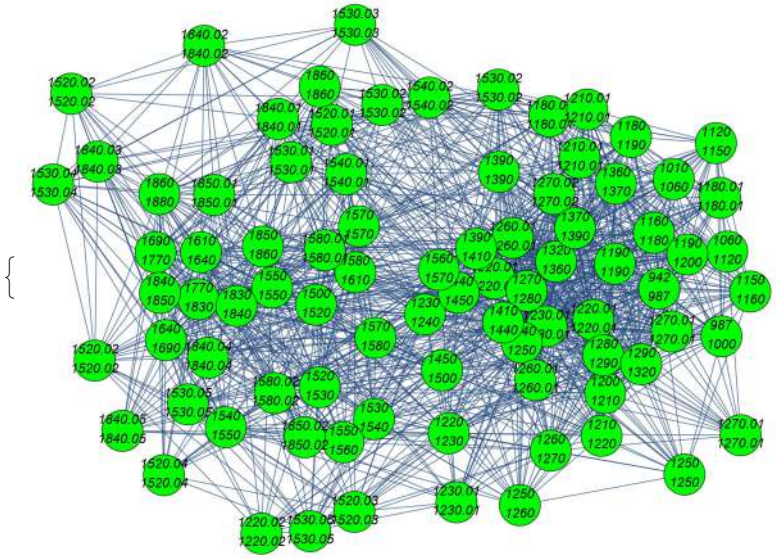
Width Feature

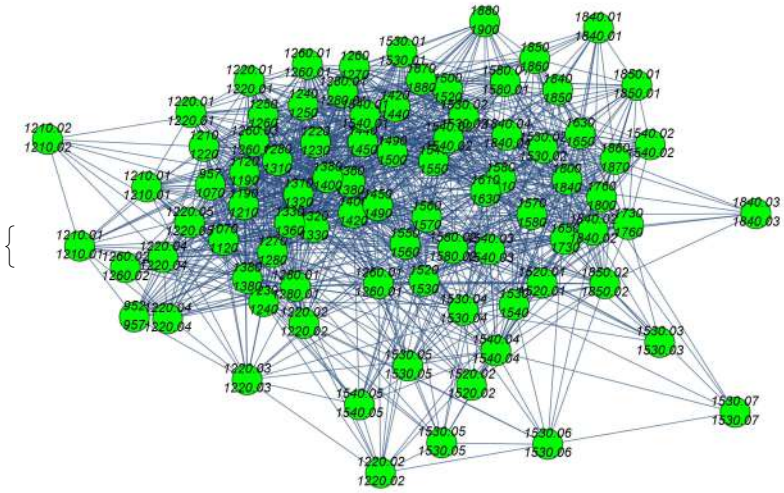
```
In[ ]:= AbsoluteTiming[
         widthdataintimewindowsFixedbucket = snetworkdatafxdbucketintimewindows[9, 85, 10];]
Out[ ]= {5.36296, Null}

In[ ]:= graphsandnodenumbers = Table[snetworkgraph[widthdataintimewindowsFixedbucket[[1]][[i]],
          widthdataintimewindowsFixedbucket[[2]][[i]], 1.5, 7, 400, Green], {i, Range@10}];

In[ ]:= graphsandnodenumbers
```

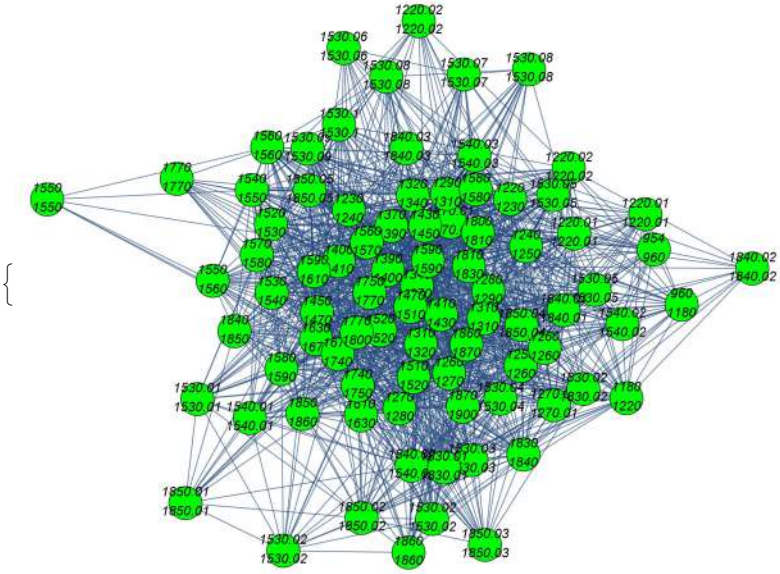
Out[•]=  , 85},

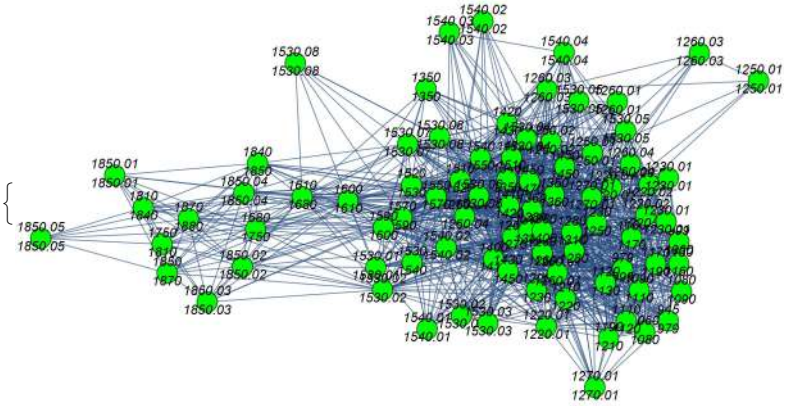
 , 85},

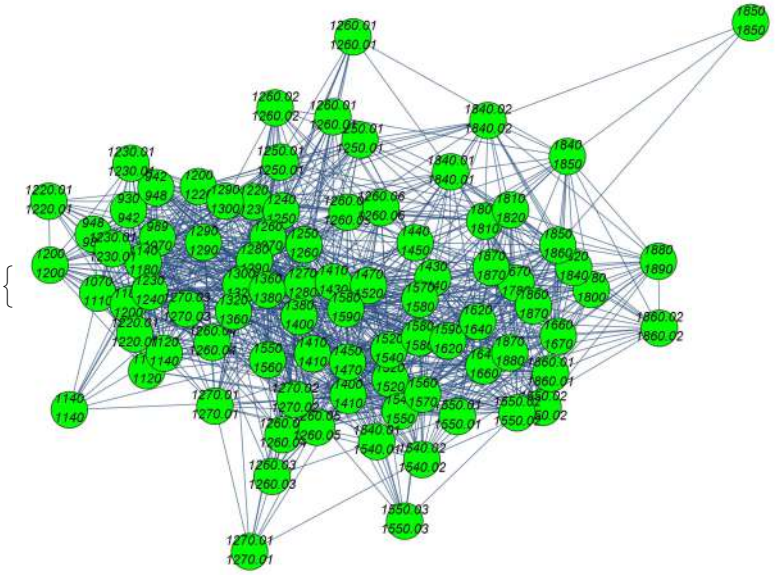
 , 85},

{  , 85},

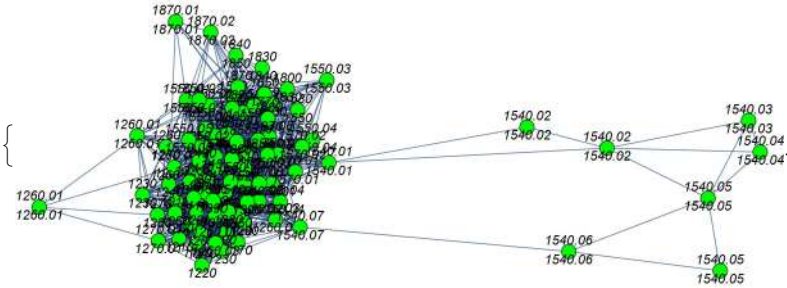
{  , 85},

{  , 85},

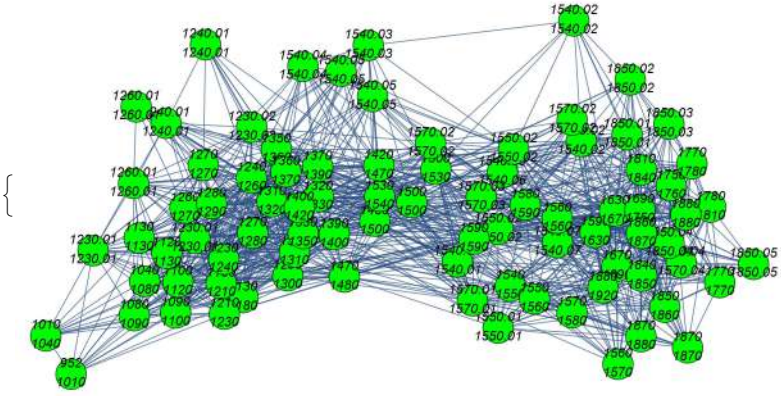
{  , 85},



, 85},



, 85},



, 85}}

In[•]:=
```
ABCvalues = Table[Mean@BetweennessCentrality[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
      FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
      "Normalized" → False], {i, Length@graphsandnodenumbers}];
degreevalues = Table[N@Mean@VertexDegree[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
```

In[•]:=
```
singlerandomgraphserdren = Table[
    RandomGraph[{VertexCount[i], EdgeCount[i]}], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomerdrenmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphserdren[[i]],
      FindGraphCommunities[singlerandomgraphserdren[[i]]], "Normalized" -> False],
    {i, Length@singlerandomgraphserdren}];
singlerandomgraphsdegreesfxd = Table[IGDegreeSequenceGame[Total[AdjacencyMatrix@i],
Method -> "VigerLatapy"], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomdegreesfxdmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphsdegreesfxd[[i]],
      FindGraphCommunities[singlerandomgraphsdegreesfxd[[i]]], "Normalized" -> False],
    {i, Length@singlerandomgraphsdegreesfxd}];
singlerandomgraphscomm = Table[randomizedgraphamongcommunities[i],
    {i, graphsandnodenumbers[[All, 1]]}];
singlerandomcommmodularityvalues = Table[N@GraphAssortativity[
      singlerandomgraphscomm[[i]], FindGraphCommunities[singlerandomgraphscomm[[i]]],
      "Normalized" -> False], {i, Length@singlerandomgraphscomm}];
```

In[•]:=
```
AbsoluteTiming[ZscoresmodularityWolf =
  Table[randomnessfunctionformodularity[i, "Wolf"], {i, graphsandnodenumbers[[All, 1]]}]]
```

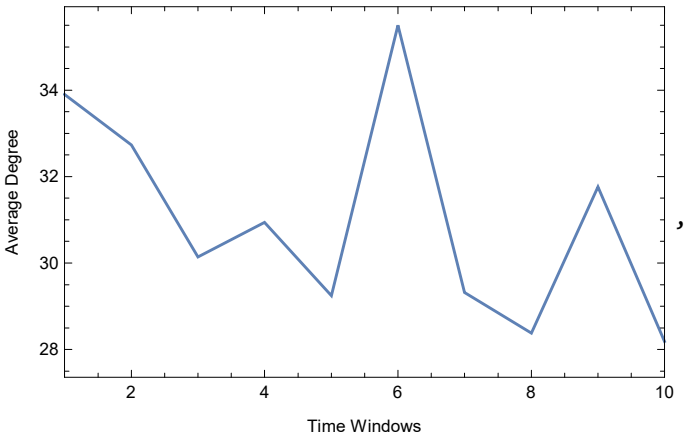
Out[•]=
```
{281.521, {{18.1208, 22.7519, -65.7198}, {18.4316, 23.9744, -50.6182},
  {21.7818, 24.2696, -53.3967}, {36.0531, 40.7257, -44.7595}, {26.9589, 33.176, -48.7689},
  {12.3565, 16.4717, -82.235}, {19.0099, 25.0379, -63.0921}, {32.3239, 37.1309, -45.9318},
  {22.5486, 32.0465, -62.4136}, {50.035, 55.7312, -34.7445}}}
```

In[•]:=
```
Table[Length@FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
 {i, Length@graphsandnodenumbers}]
```

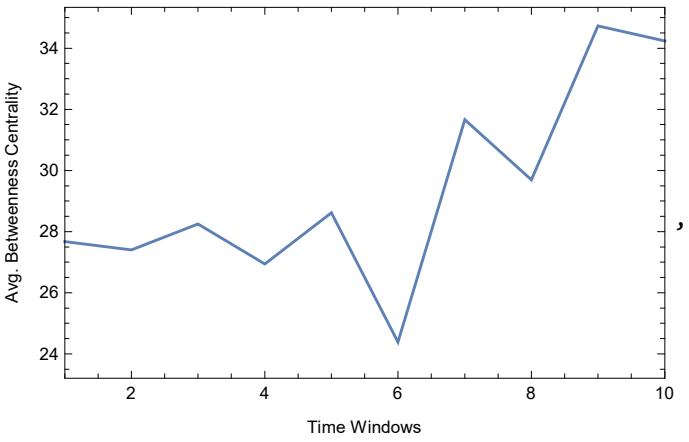
Out[•]=
```
{2, 3, 2, 2, 2, 3, 3, 2, 3, 3}
```

In[ ]:=
```
{ListLinePlot[Thread[{Range@10, degreevalues}], Frame → True,
  FrameLabel → {"Time Windows", "Average Degree"}, ImageSize → 350,
  PlotRange → {{1, 10}, All}], ListLinePlot[Thread[{Range@10, ABCvalues}],
  Frame → True, FrameLabel → {"Time Windows", "Avg. Betweenness Centrality"},
  ImageSize → 350, PlotRange → {{1, 10}, All}],
 ListLinePlot[Thread[{Range@10, modularityvalues}], Frame → True,
  FrameLabel → {"Time Windows", "Modularity"}, ImageSize → 350, PlotRange → All],
 ListLinePlot[{Thread[{Range@10, singlerandomerdrenmodularityvalues}],
   Thread[{Range@10, singlerandomdegreesfxdmodularityvalues}],
   Thread[{Range@10, singlerandomcommmodularityvalues}]}, Frame → True,
  FrameLabel → {"Time Windows", "Modularity (Single Random Graphs)"},
  ImageSize → 350, PlotRange → {{1, 10}, All}, PlotLabels →
   Placed[{"Erdös-Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]],
 ListLinePlot[{Thread[{Range@10, ZscoresmodularityWolf[[All, 1]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 2]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 3]]}]},
  Frame → True, FrameLabel → {"Time Windows", "Z-Score for Modularity"},
  ImageSize → 350, PlotRange → All, PlotLabels →
   Placed[{"Erdös Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]]}
```

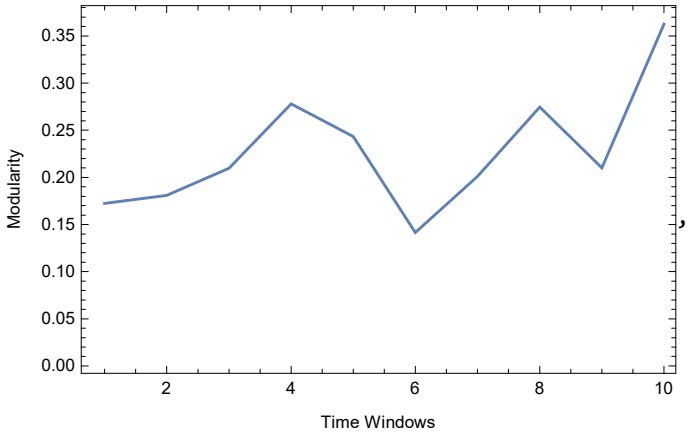
Out[ ]=

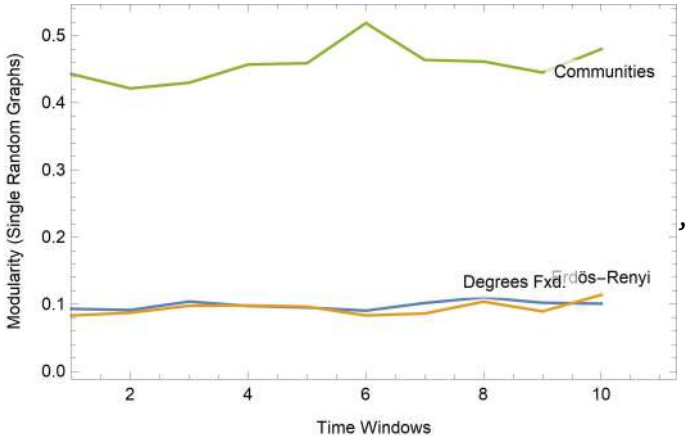


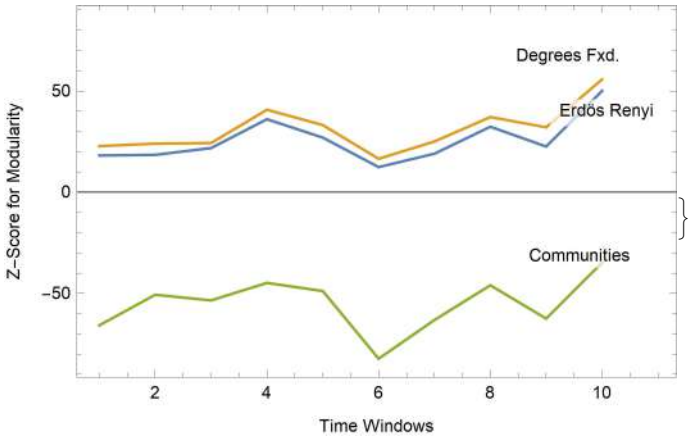
,



,







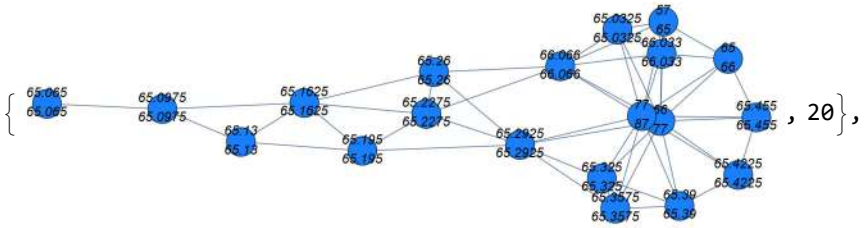
Thickness Feature

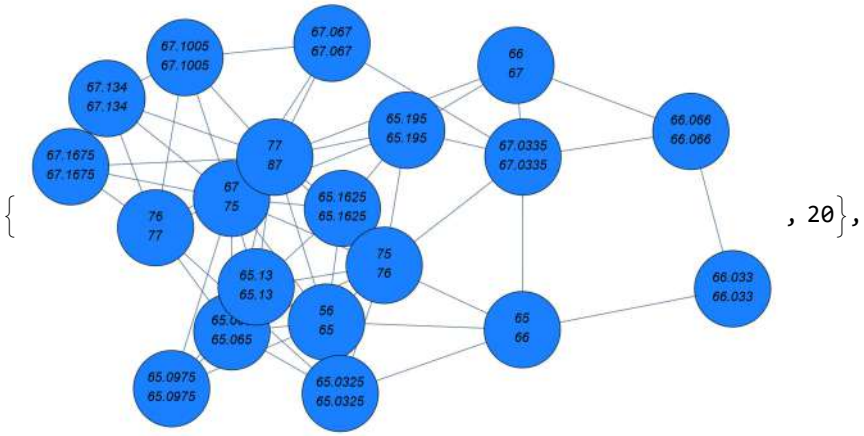
```
In[ ]:= AbsoluteTiming[
          thicknessdataintimewindowsFixedbucket = snetworkdatafxdbucketintimewindows[10, 20, 10];]
Out[ ]= {5.16073, Null}
```

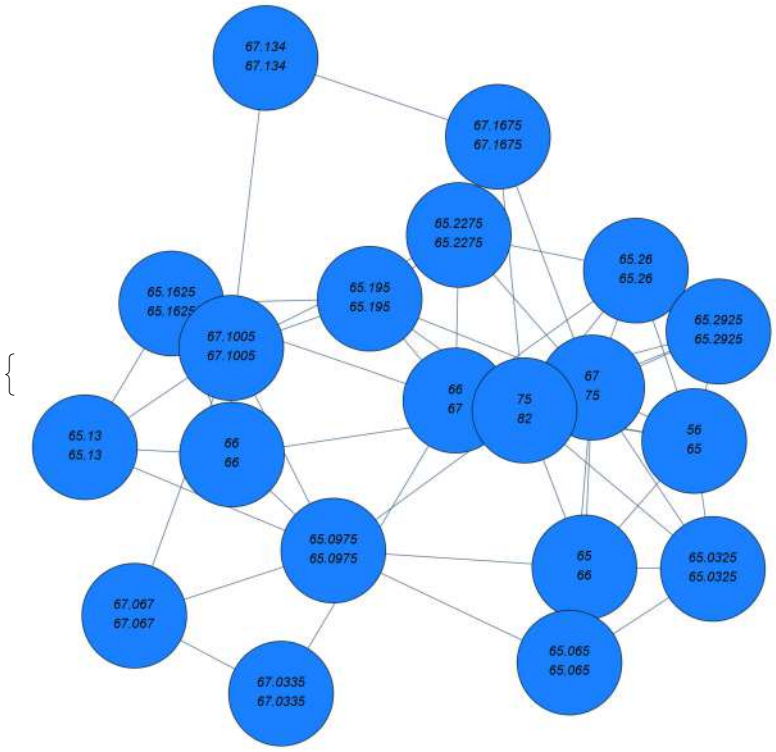
In[ ]:=
```
graphsandnodenumbers =
  Table[snetworkgraph[thicknessdataintimewindowsFixedbucket[[1]][[i]],
    thicknessdataintimewindowsFixedbucket[[2]][[i]],
    1.5, 7, 400, RGBColor[0.1, 0.5, 1.]], {i, Range@10}];
```

In[ ]:=
```
graphsandnodenumbers
```

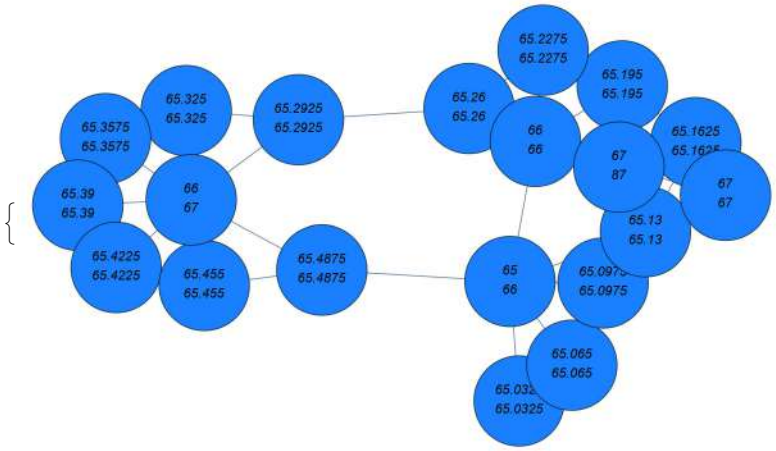
Out[ ]=



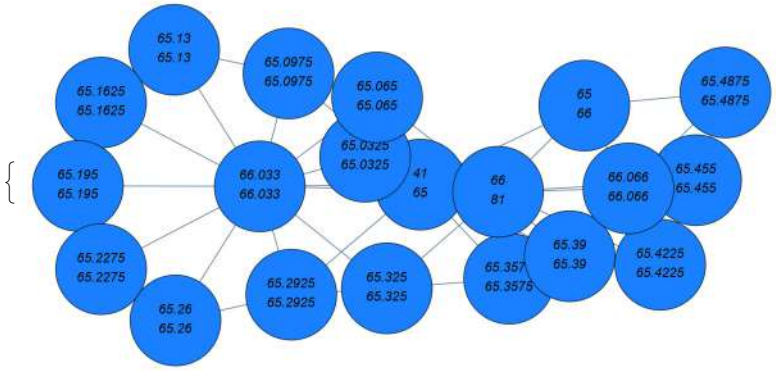



67.134
67.134
67.1675
67.1675
65.2275
65.2275
65.26
65.26
65.1625
65.1625
65.195
65.195
67.1005
67.1005
65.2925
65.2925
66
67
75
82
67
75
65.13
65.13
66
66
56
65
65.0975
65.0975
65
66
65.0325
65.0325
67.067
67.067
65.065
65.065
67.0335
67.0335

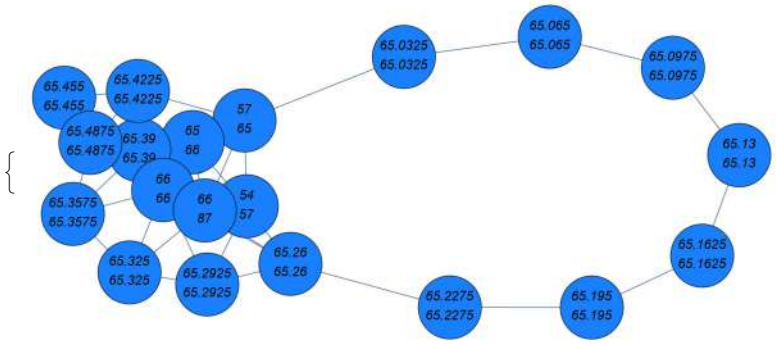
, 20},


65.2275
65.2275
65.325
65.325
65.2925
65.2925
65.26
65.26
65.195
65.195
65.3575
65.3575
66
66
65.1625
65.1625
66
67
67
87
67
67
65.39
65.39
65.13
65.13
65.4225
65.4225
65.455
65.455
65.4875
65.4875
65
66
65.0975
65.0975
65.065
65.065
65.0325
65.0325

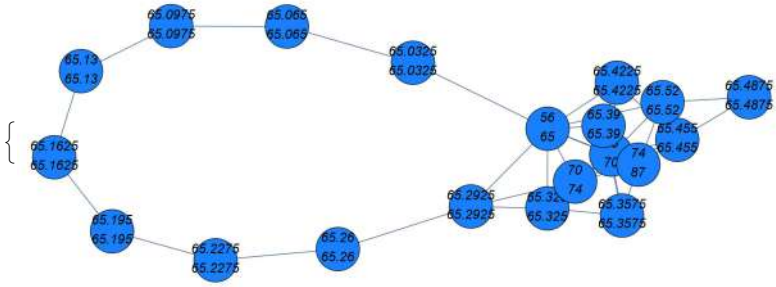
, 20},


65.13
65.13
65.0975
65.0975
65.065
65.065
65.1625
65.1625
65
66
65.4875
65.4875
65.0325
65.0325
65.195
65.195
66.033
66.033
41
65
66
81
66.066
66.066
65.455
65.455
65.39
65.39
65.2275
65.2275
65.4225
65.4225
65.357
65.3575
65.2925
65.2925
65.325
65.325
65.26
65.26

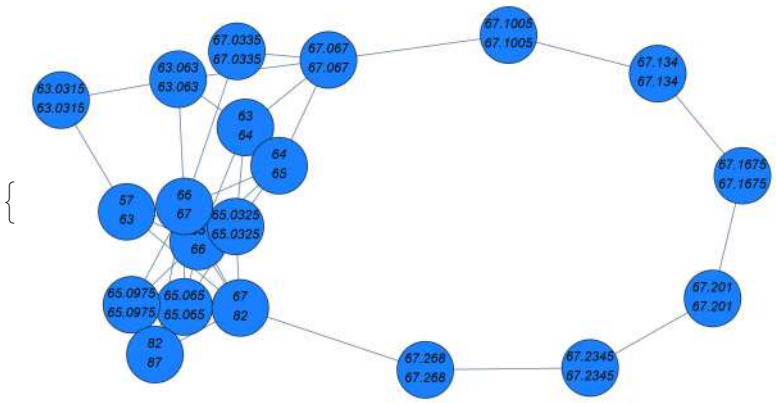
, 20},


65.065
65.065
65.0325
65.0325
65.0975
65.0975
65.455
65.455
65.4225
65.4225
57
65
65.13
65.13
65.4875
65.4875
65.39
65.39
65
66
66
66
66
87
54
57
65.3575
65.3575
65.26
65.26
65.325
65.325
65.2925
65.2925
65.1625
65.1625
65.2275
65.2275
65.195
65.195

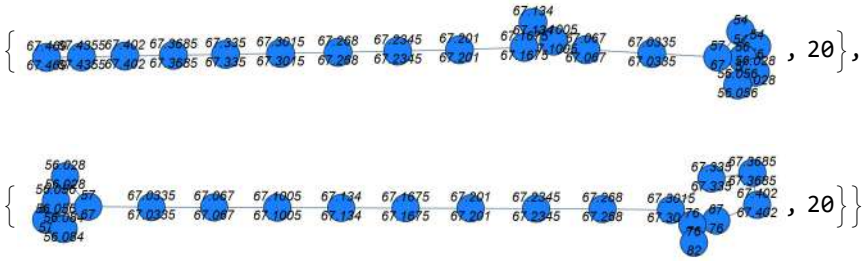
, 20},


65.0975
65.0975
65.065
65.065
65.13
65.13
65.0325
65.0325
65.4225
65.4225
65.52
65.52
65.4875
65.4875
56
65
65.39
65.39
65.455
65.455
74
87
70
70
74
65.1625
65.1625
65.2925
65.2925
65.32
65.325
65.3575
65.3575
65.195
65.195
65.2275
65.2275
65.26
65.26

, 20},


67.1009
67.1005
67.0335
67.0335
67.067
67.067
67.134
67.134
63.063
63.063
63.0315
63.0315
63
64
64
65
67.1675
67.1675
57
63
66
67
65.0325
65.0325
66
67.201
67.201
65.0975
65.0975
65.065
65.065
67
82
82
87
67.268
67.268
67.2345
67.2345

, 20},



In[ ]:=
```
ABCvalues = Table[Mean@BetweennessCentrality[graphsandnodenumbers[[i]][[1]]],
   {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
     FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
     "Normalized" → False], {i, Length@graphsandnodenumbers}];
degreevalues = Table[N@Mean@VertexDegree[graphsandnodenumbers[[i]][[1]]],
   {i, Length@graphsandnodenumbers}];
```

In[ ]:=
```
singlerandomgraphserdren = Table[
   RandomGraph[{VertexCount[i], EdgeCount[i]}], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomerdrenmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphserdren[[i]],
     FindGraphCommunities[singlerandomgraphserdren[[i]]], "Normalized" -> False],
   {i, Length@singlerandomgraphserdren}];
singlerandomgraphsdegreesfxd = Table[IGDegreeSequenceGame[Total[AdjacencyMatrix@i],
Method -> "VigerLatapy"], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomdegreesfxdmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphsdegreesfxd[[i]],
     FindGraphCommunities[singlerandomgraphsdegreesfxd[[i]]], "Normalized" -> False],
   {i, Length@singlerandomgraphsdegreesfxd}];
singlerandomgraphscomm = Table[randomizedgraphamongcommunities[i],
   {i, graphsandnodenumbers[[All, 1]]}];
singlerandomcommmodularityvalues = Table[N@GraphAssortativity[
     singlerandomgraphscomm[[i]], FindGraphCommunities[singlerandomgraphscomm[[i]]],
     "Normalized" -> False], {i, Length@singlerandomgraphscomm}];
```

In[ ]:=
```
AbsoluteTiming[ZscoresmodularityWolf =
  Table[randomnessfunctionformodularity[i, "Wolf"], {i, graphsandnodenumbers[[All, 1]]}]]
```

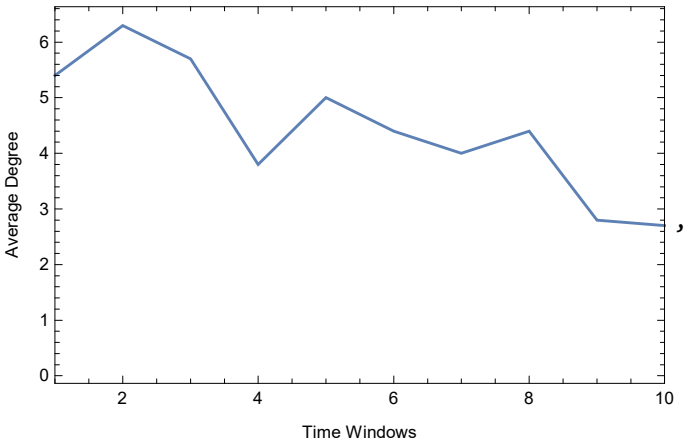
Out[ ]=
```
{65.1567, {{4.07445, 5.50428, -4.0838}, {2.68955, 4.14274, -5.86804},
  {1.83862, 2.54654, -3.07004}, {5.26397, 4.6218, -0.125558}, {3.0181, 4.33807, -3.21577},
  {0.924623, 1.94833, -3.23749}, {0.213684, 0.962007, -2.55683},
  {1.64527, 2.95783, -1.56844}, {4.68317, 5.01019, 1.85761}, {4.64162, 4.59406, 1.34961}}}
```

In[ ]:=
```
Table[Length@FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
 {i, Length@graphsandnodenumbers}]
```

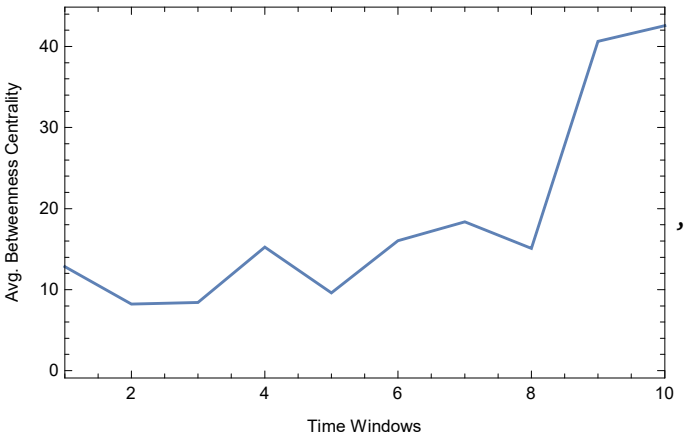
Out[ ]=
```
{3, 3, 3, 3, 3, 3, 3, 3, 4, 4}
```

In[ ]:=
```
{ListLinePlot[Thread[{Range@10, degreevalues}], Frame → True,
  FrameLabel → {"Time Windows", "Average Degree"}, ImageSize → 350,
  PlotRange → {{1, 10}, All}], ListLinePlot[Thread[{Range@10, ABCvalues}],
  Frame → True, FrameLabel → {"Time Windows", "Avg. Betweenness Centrality"},
  ImageSize → 350, PlotRange → {{1, 10}, All}],
 ListLinePlot[Thread[{Range@10, modularityvalues}], Frame → True,
  FrameLabel → {"Time Windows", "Modularity"}, ImageSize → 350, PlotRange → All],
 ListLinePlot[{Thread[{Range@10, singlerandomerdrenmodularityvalues}],
   Thread[{Range@10, singlerandomdegreesfxdmodularityvalues}],
   Thread[{Range@10, singlerandomcommmodularityvalues}]}, Frame → True,
  FrameLabel → {"Time Windows", "Modularity (Single Random Graphs)"},
  ImageSize → 350, PlotRange → {{1, 10}, All}, PlotLabels →
   Placed[{"Erdös-Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Below}]],
 ListLinePlot[{Thread[{Range@10, ZscoresmodularityWolf[[All, 1]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 2]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 3]]}]},
  Frame → True, FrameLabel → {"Time Windows", "Z-Score for Modularity"},
  ImageSize → 350, PlotRange → All, PlotLabels →
   Placed[{"Erdös Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]]}
```

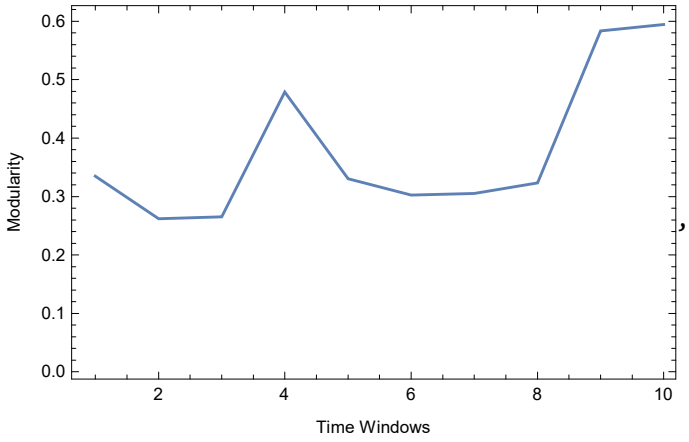
Out[ ]=



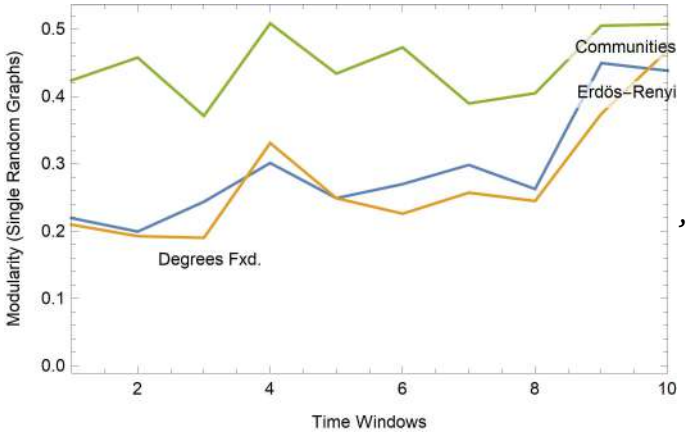
,



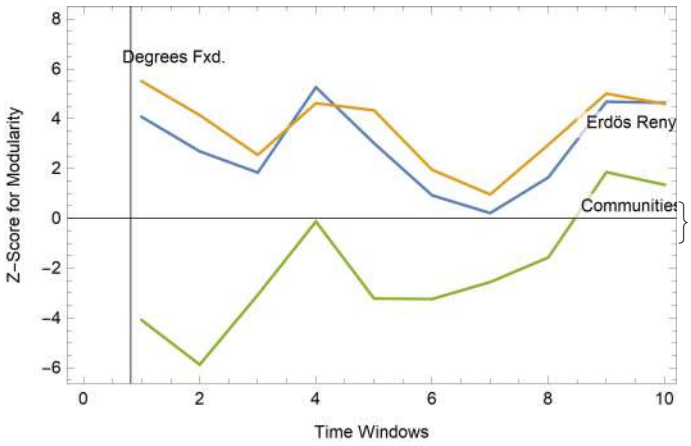
,



,



,



Length Feature

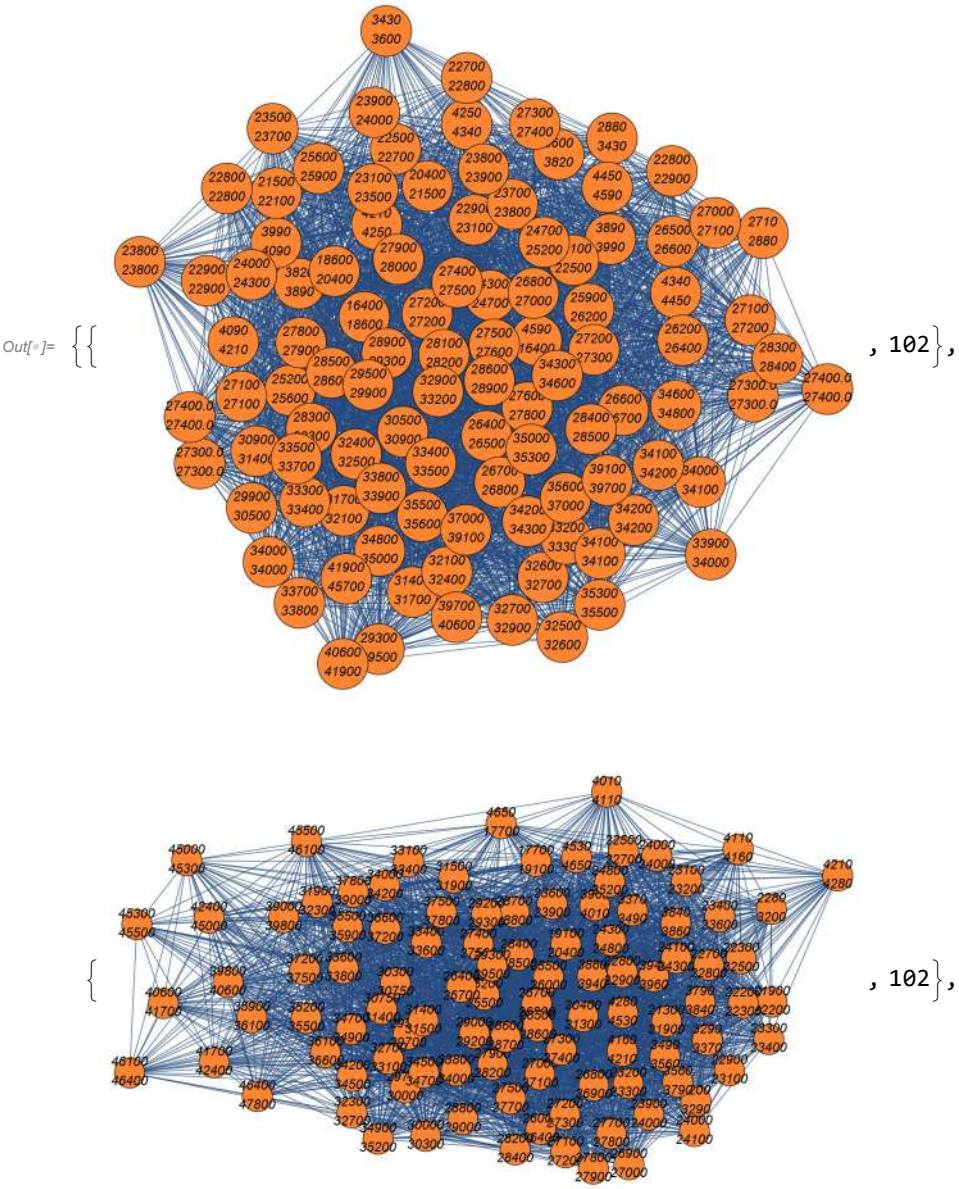
In[ ]:=
```
AbsoluteTiming[
 lengthdataintimewindowsFixedbucket = snetworkdatafxdbucketintimewindows[12, 102, 10];]
```

Out[ ]= {6.6219, Null}

In[ ]:=
```
graphsandnodenumbers = Table[snetworkgraph[lengthdataintimewindowsFixedbucket[[1]][[i]],
    lengthdataintimewindowsFixedbucket[[2]][[i]], 1.5,
    7, 400, RGBColor[1., 0.53, 0.2]], {i, Range@10}];
```

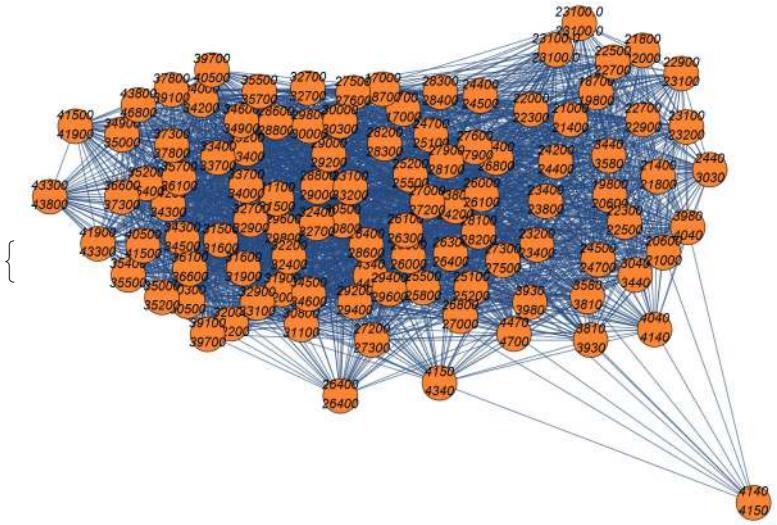
In[ ]:=
```
graphsandnodenumbers
```

Out[ ]=

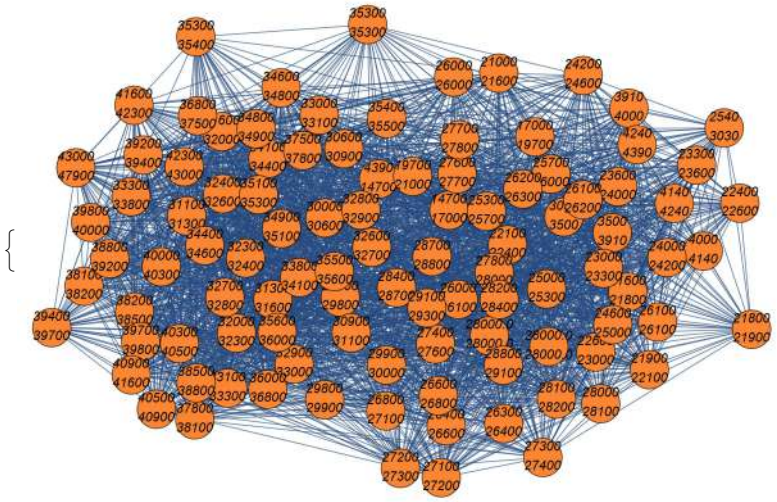


, 102},

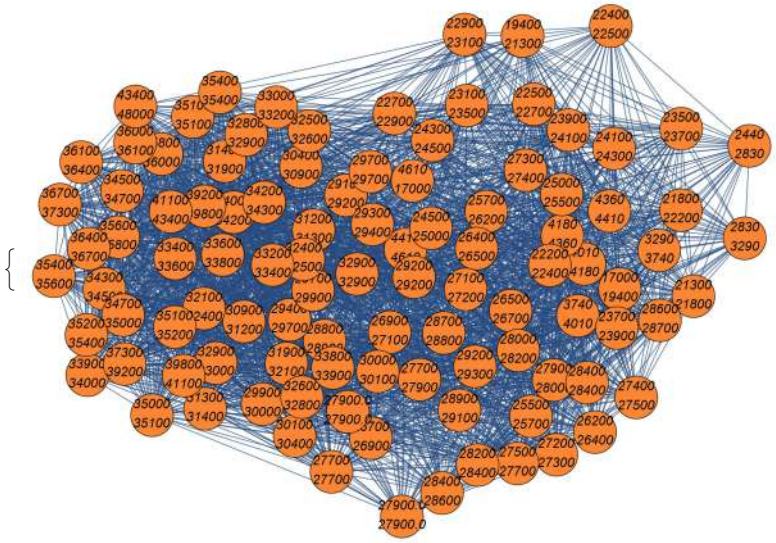
, 102},



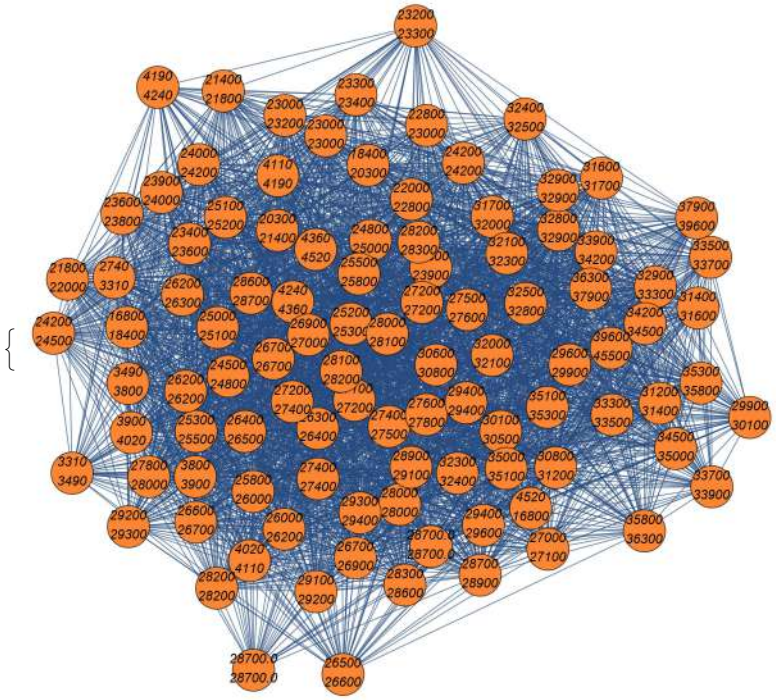
, 102},



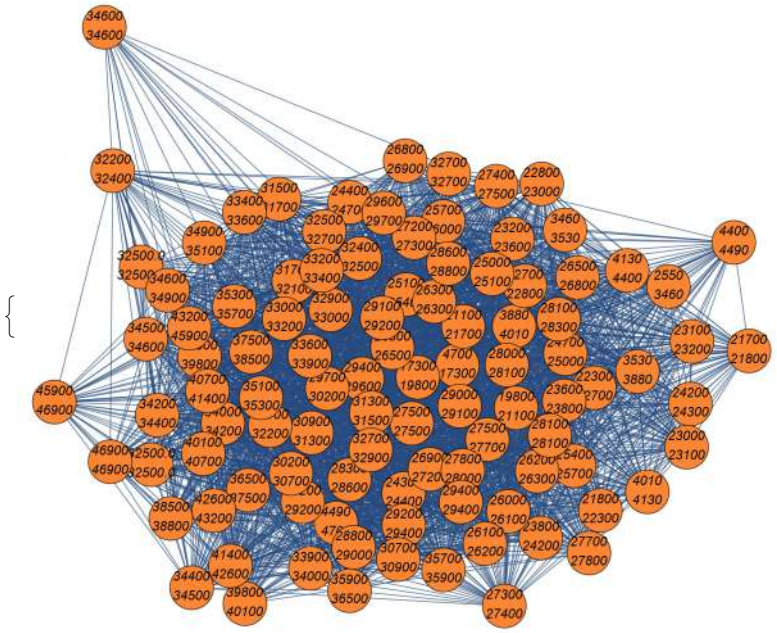
, 102},



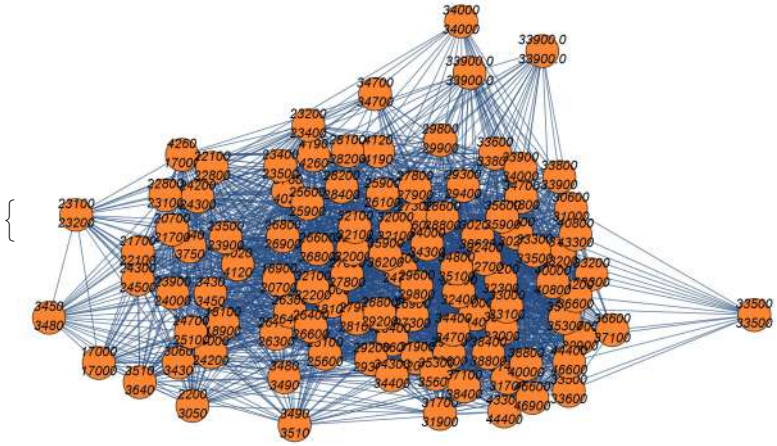
, 102},



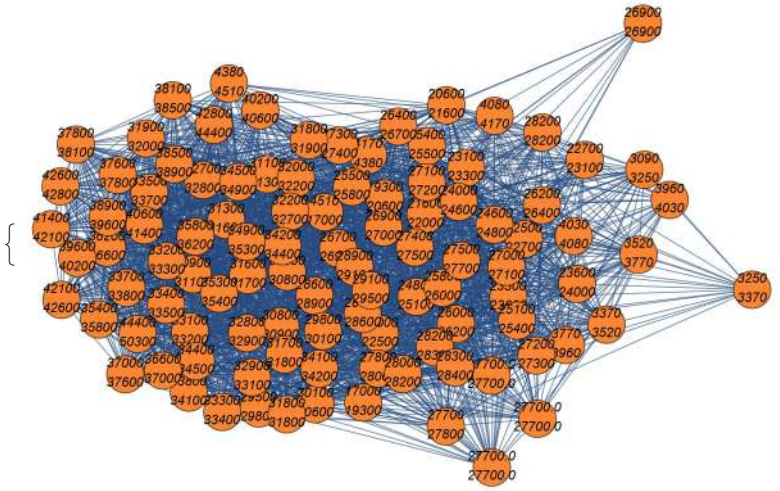
, 102},

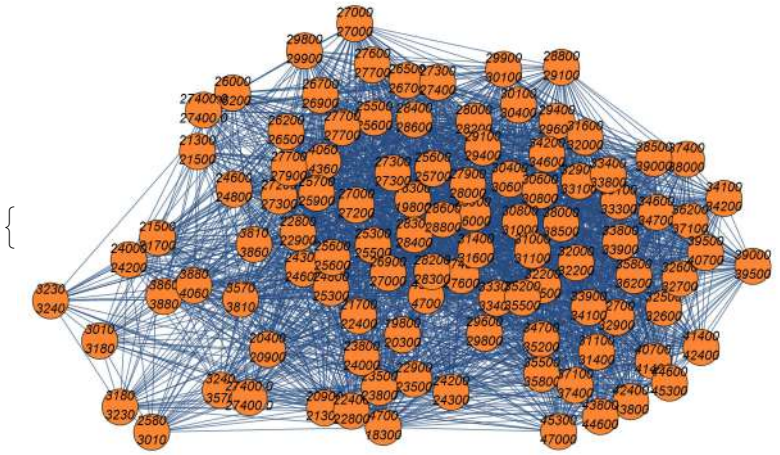


, 102},



, 102},



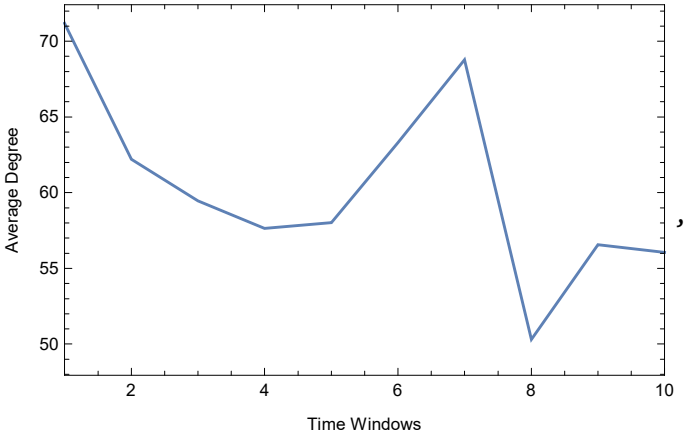


In[ ]:=
```
ABCvalues = Table[Mean@BetweennessCentrality[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
      FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
      "Normalized" → False], {i, Length@graphsandnodenumbers}];
degreevalues = Table[N@Mean@VertexDegree[graphsandnodenumbers[[i]][[1]]],
    {i, Length@graphsandnodenumbers}];
```

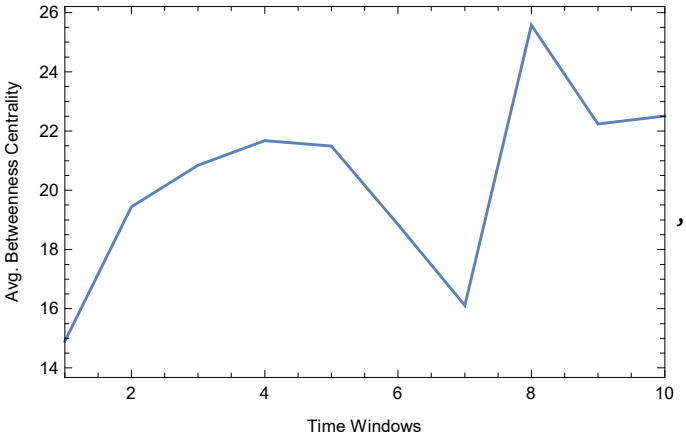
```
In[•]:= singlerandomgraphserdren = Table[
          RandomGraph[{VertexCount[i], EdgeCount[i]}], {i, graphsandnodenumbers[[All, 1]]}];
        singlerandomerdrenmodularityvalues =
          Table[N@GraphAssortativity[singlerandomgraphserdren[[i]],
              FindGraphCommunities[singlerandomgraphserdren[[i]]], "Normalized" -> False],
            {i, Length@singlerandomgraphserdren}];
        singlerandomgraphsdegreesfxd = Table[IGDegreeSequenceGame[Total[AdjacencyMatrix@i],
        Method -> "VigerLatapy"], {i, graphsandnodenumbers[[All, 1]]}];
        singlerandomdegreesfxdmodularityvalues =
          Table[N@GraphAssortativity[singlerandomgraphsdegreesfxd[[i]],
              FindGraphCommunities[singlerandomgraphsdegreesfxd[[i]]], "Normalized" -> False],
            {i, Length@singlerandomgraphsdegreesfxd}];
        singlerandomgraphscomm = Table[randomizedgraphamongcommunities[i],
            {i, graphsandnodenumbers[[All, 1]]}];
        singlerandomcommmodularityvalues = Table[N@GraphAssortativity[
              singlerandomgraphscomm[[i]], FindGraphCommunities[singlerandomgraphscomm[[i]]],
              "Normalized" -> False], {i, Length@singlerandomgraphscomm}];

In[•]:= AbsoluteTiming[ZscoresmodularityWolf =
          Table[randomnessfunctionformodularity[i, "Wolf"], {i, graphsandnodenumbers[[All, 1]]}]]

Out[•]= {1229.7, {{18.9852, 19.9745, -163.711}, {46.3861, 53.5962, -133.914},
          {60.6419, 63.9224, -116.304}, {59.2833, 63.7099, -108.884}, {53.4703, 56.04, -114.006},
          {30.6836, 31.0555, -151.224}, {27.8457, 32.9705, -158.79}, {41.9923, 47.0041, -103.988},
          {47.0245, 51.2727, -117.141}, {43.1295, 49.4626, -123.508}}}

In[•]:= Table[Length@FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
          {i, Length@graphsandnodenumbers}]

Out[•]= {2, 2, 2, 2, 3, 2, 3, 2, 2, 2}
```

In[ ]:=
```
{ListLinePlot[Thread[{Range@10, degreevalues}], Frame → True,
  FrameLabel → {"Time Windows", "Average Degree"}, ImageSize → 350,
  PlotRange → {{1, 10}, All}], ListLinePlot[Thread[{Range@10, ABCvalues}],
  Frame → True, FrameLabel → {"Time Windows", "Avg. Betweenness Centrality"},
  ImageSize → 350, PlotRange → {{1, 10}, All}],
 ListLinePlot[Thread[{Range@10, modularityvalues}], Frame → True,
  FrameLabel → {"Time Windows", "Modularity"}, ImageSize → 350, PlotRange → All],
 ListLinePlot[{Thread[{Range@10, singlerandomerdrenmodularityvalues}],
   Thread[{Range@10, singlerandomdegreesfxdmodularityvalues}],
   Thread[{Range@10, singlerandomcommmodularityvalues}]}, Frame → True,
  FrameLabel → {"Time Windows", "Modularity (Single Random Graphs)"},
  ImageSize → 350, PlotRange → {{1, 10}, All}, PlotLabels →
   Placed[{"Erdös-Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]],
 ListLinePlot[{Thread[{Range@10, ZscoresmodularityWolf[[All, 1]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 2]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 3]]}]},
  Frame → True, FrameLabel → {"Time Windows", "Z-Score for Modularity"},
  ImageSize → 350, PlotRange → All, PlotLabels →
   Placed[{"Erdös Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]]}
```

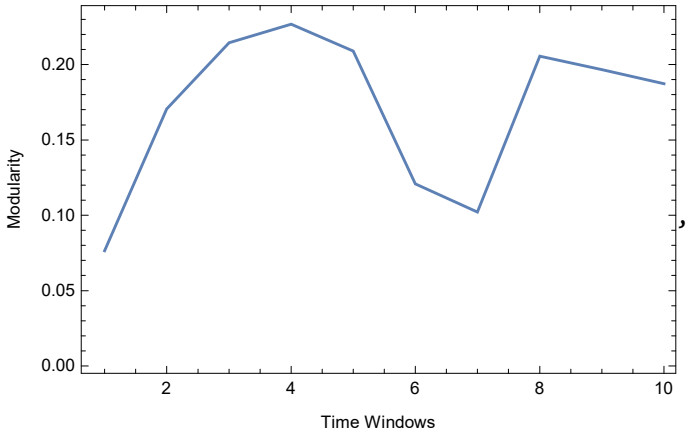
Out[ ]=



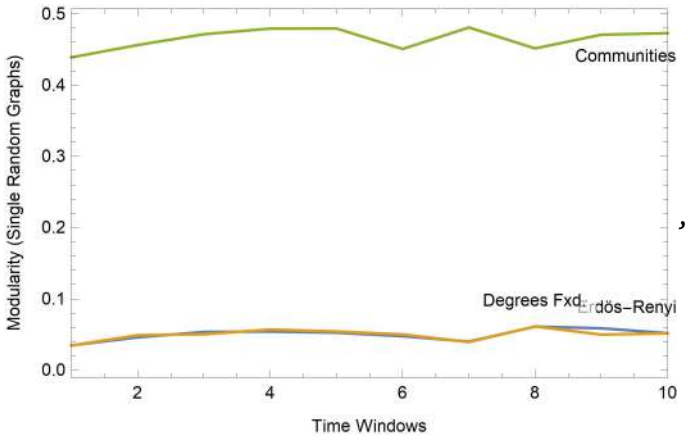
,



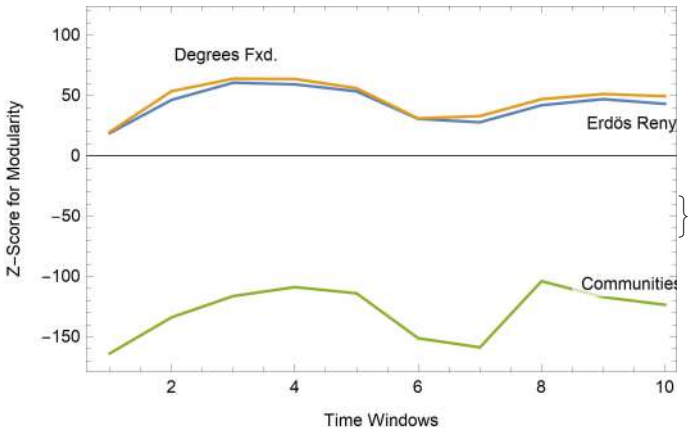
,



,



,



}

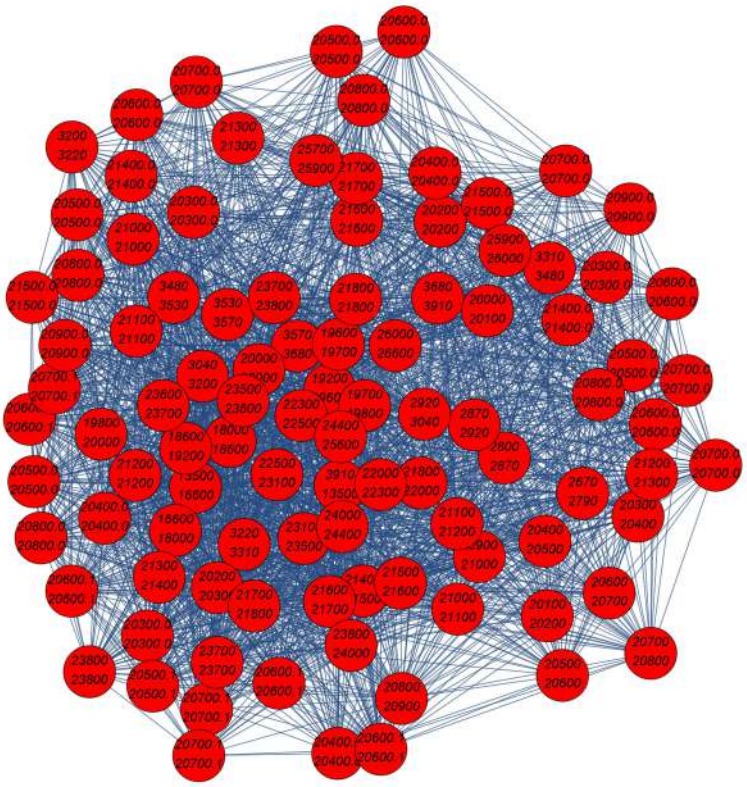
Weight Feature

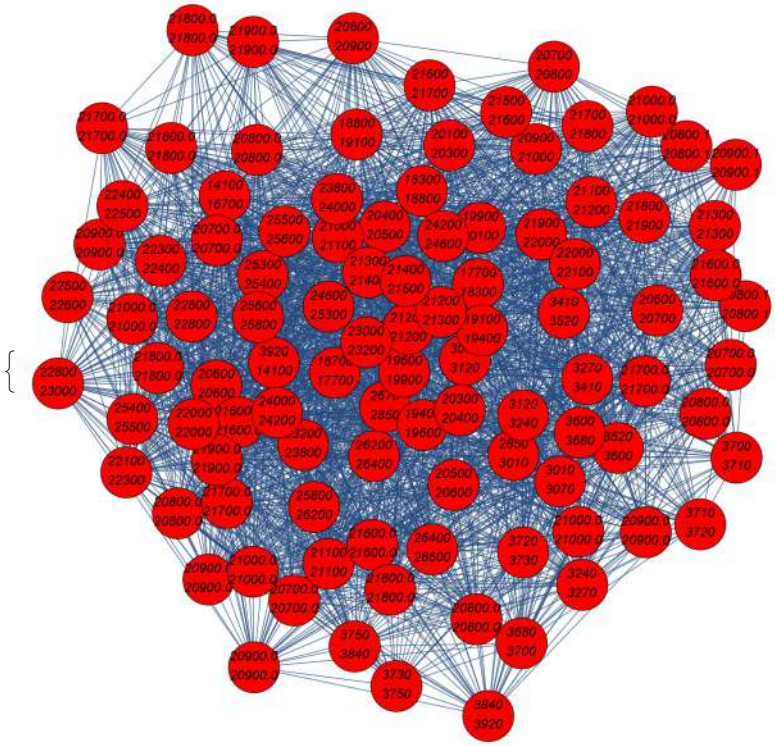
In[ ]:=
```
AbsoluteTiming[
 weightdataintimewindowsFixedbucket = snetworkdatafxdbucketintimewindows[11, 100, 10];]
```

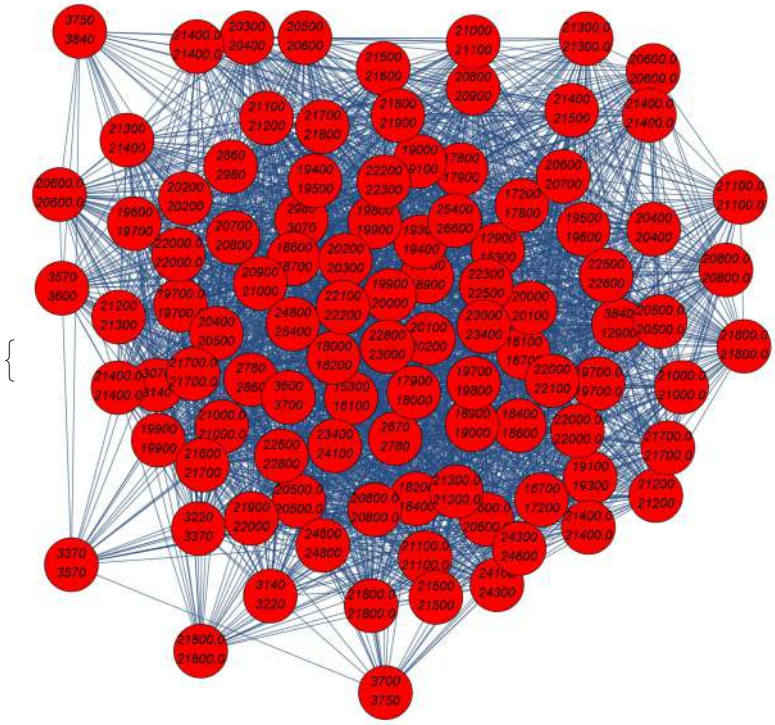
Out[ ]= {6.95992, Null}

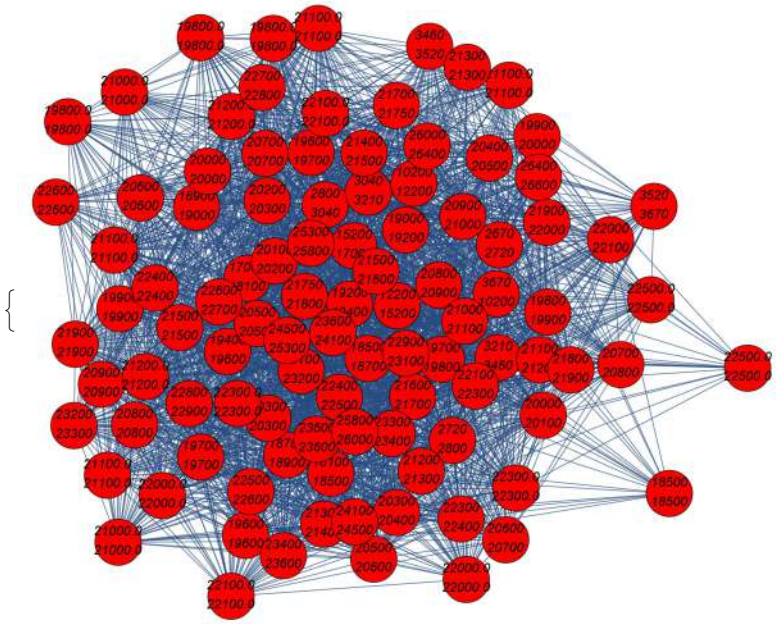
In[ ]:=
```
graphsandnodenumbers = Table[snetworkgraph[weightdataintimewindowsFixedbucket[[1]][[i]],
    weightdataintimewindowsFixedbucket[[2]][[i]], 1.5, 7, 400, Red], {i, Range@10}];
```

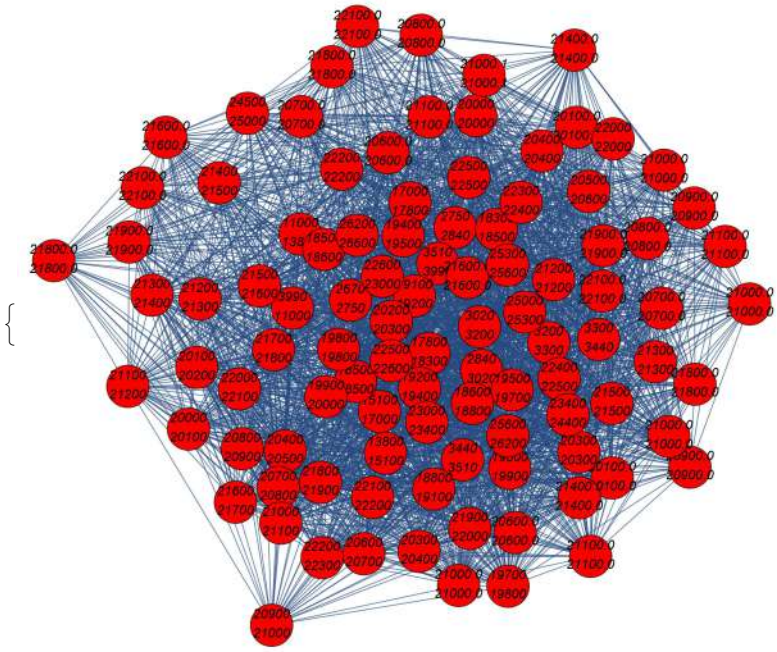
In[ ]:=
```
graphsandnodenumbers
```

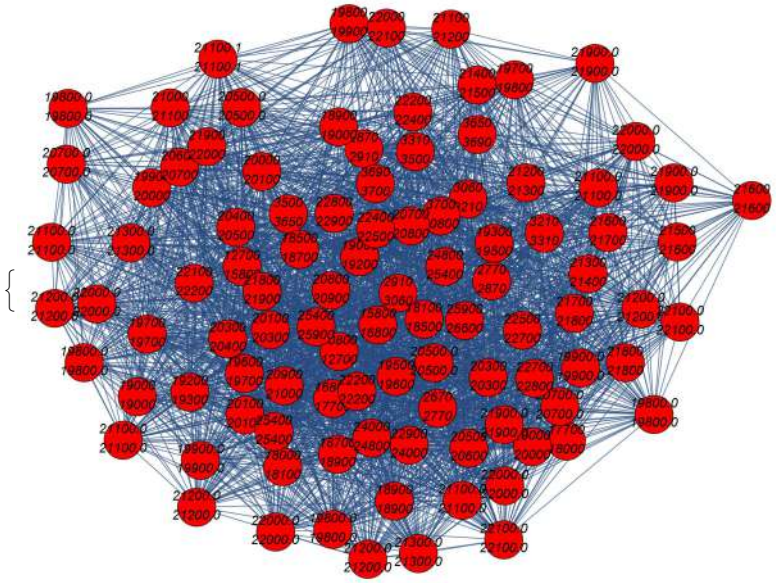
Out[ ]=



, 100},

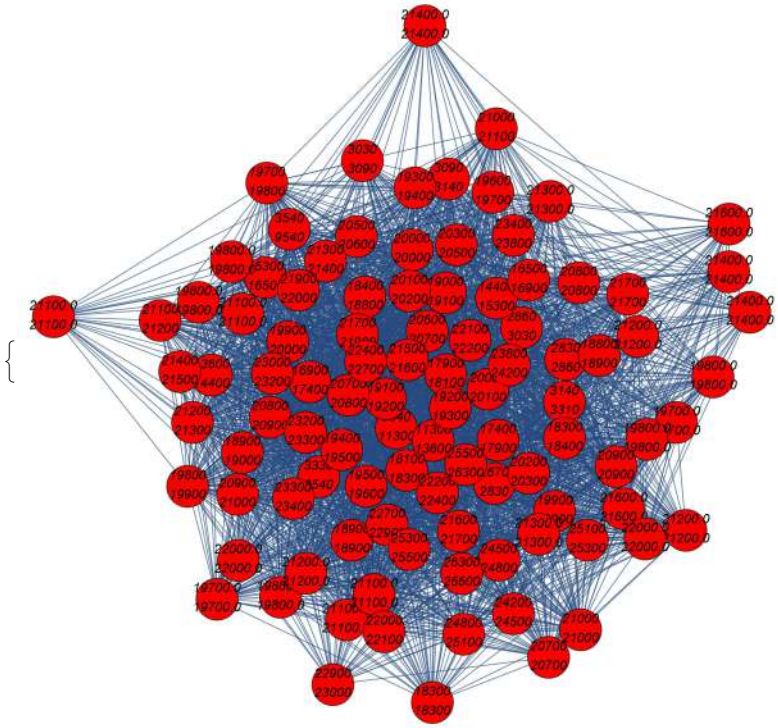

, 100},

{  , 100},

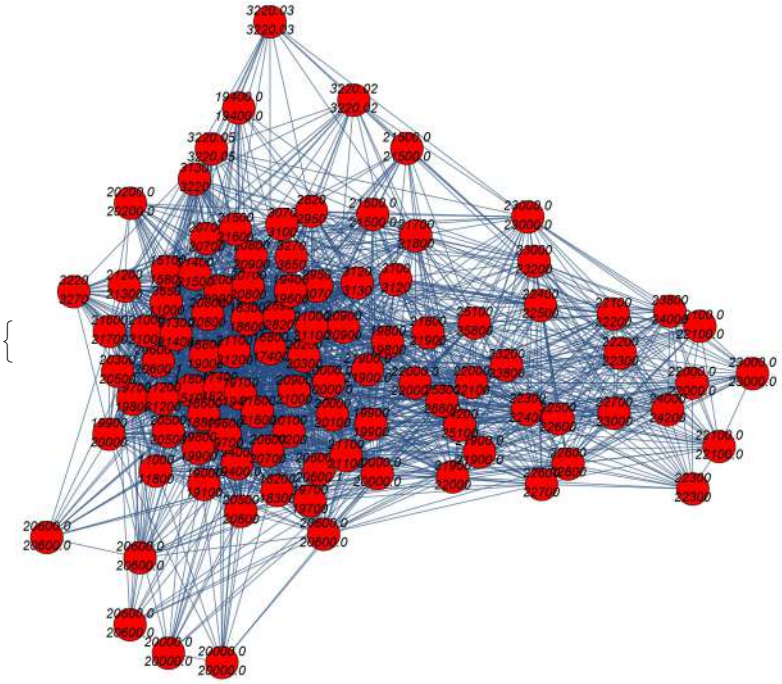
{  , 100},



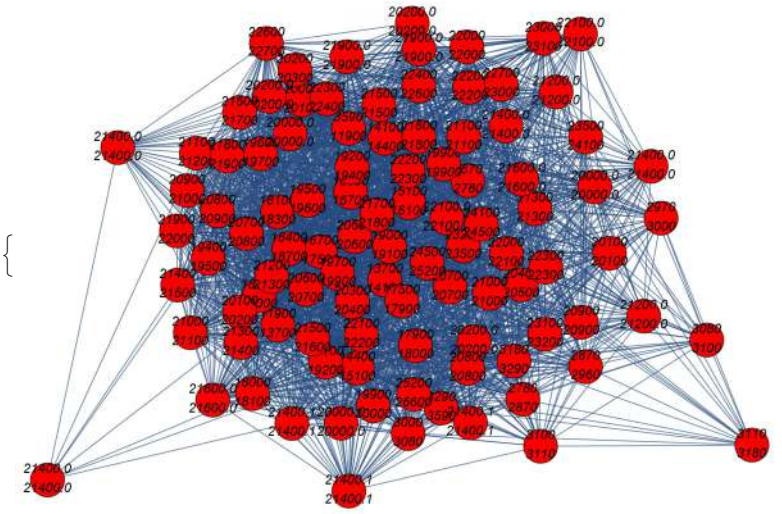
, 100},



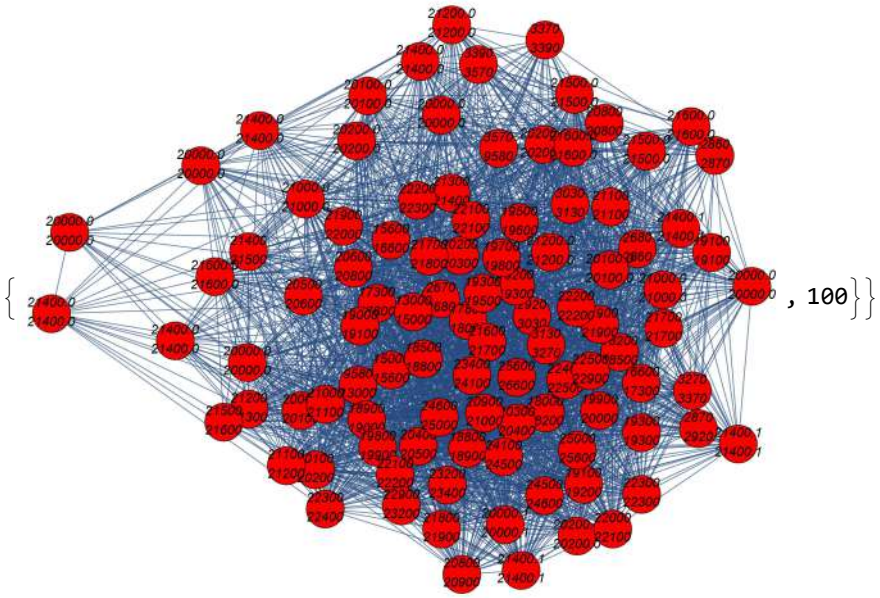
, 100},



{ , 100},



{ , 100},



In[ ]:=
```
ABCvalues = Table[Mean@BetweennessCentrality[graphsandnodenumbers[[i]][[1]]],
   {i, Length@graphsandnodenumbers}];
modularityvalues = Table[N@GraphAssortativity[graphsandnodenumbers[[i]][[1]],
     FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
     "Normalized" → False], {i, Length@graphsandnodenumbers}];
degreevalues = Table[N@Mean@VertexDegree[graphsandnodenumbers[[i]][[1]]],
   {i, Length@graphsandnodenumbers}];
```

In[ ]:=
```
singlerandomgraphserdren = Table[
   RandomGraph[{VertexCount[i], EdgeCount[i]}], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomerdrenmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphserdren[[i]],
     FindGraphCommunities[singlerandomgraphserdren[[i]]], "Normalized" -> False],
   {i, Length@singlerandomgraphserdren}];
singlerandomgraphsdegreesfxd = Table[IGDegreeSequenceGame[Total[AdjacencyMatrix@i],
Method -> "VigerLatapy"], {i, graphsandnodenumbers[[All, 1]]}];
singlerandomdegreesfxdmodularityvalues =
  Table[N@GraphAssortativity[singlerandomgraphsdegreesfxd[[i]],
     FindGraphCommunities[singlerandomgraphsdegreesfxd[[i]]], "Normalized" -> False],
   {i, Length@singlerandomgraphsdegreesfxd}];
singlerandomgraphscomm = Table[randomizedgraphamongcommunities[i],
   {i, graphsandnodenumbers[[All, 1]]}];
singlerandomcommmodularityvalues = Table[N@GraphAssortativity[
     singlerandomgraphscomm[[i]], FindGraphCommunities[singlerandomgraphscomm[[i]]],
     "Normalized" -> False], {i, Length@singlerandomgraphscomm}];
```

In[ ]:=
```
AbsoluteTiming[ZscoresmodularityWolf =
  Table[randomnessfunctionformodularity[i, "Wolf"], {i, graphsandnodenumbers[[All, 1]]}]]
```

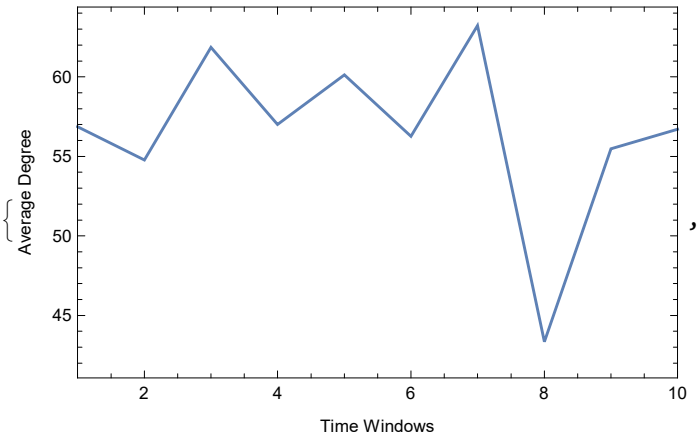
Out[•]= {855.782, {{34.8401, 39.9894, -123.48}, {28.4128, 32.2426, -120.603},
{22.9095, 26.2783, -152.214}, {17.1835, 18.729, -139.01}, {18.5607, 18.9559, -145.308},
{26.775, 28.4406, -127.48}, {11.8857, 13.5977, -158.882}, {31.1421, 37.6489, -76.5715},
{9.14574, 11.3237, -119.686}, {15.8447, 19.4654, -128.804}}}

In[•]:=
```
Table[Length@FindGraphCommunities[graphsandnodenumbers[[i]][[1]]],
 {i, Length@graphsandnodenumbers}]
```

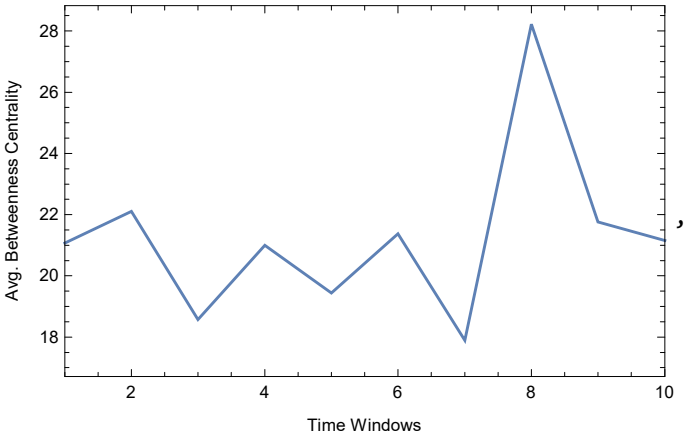
Out[•]= {2, 2, 2, 2, 3, 3, 3, 3, 3, 2}

In[•]:=
```
{ListLinePlot[Thread[{Range@10, degreevalues}], Frame → True,
  FrameLabel → {"Time Windows", "Average Degree"}, ImageSize → 350,
  PlotRange → {{1, 10}, All}], ListLinePlot[Thread[{Range@10, ABCvalues}],
  Frame → True, FrameLabel → {"Time Windows", "Avg. Betweenness Centrality"},
  ImageSize → 350, PlotRange → {{1, 10}, All}],
 ListLinePlot[Thread[{Range@10, modularityvalues}], Frame → True,
  FrameLabel → {"Time Windows", "Modularity"}, ImageSize → 350, PlotRange → All],
 ListLinePlot[{Thread[{Range@10, singlerandomerdrenmodularityvalues}],
   Thread[{Range@10, singlerandomdegreesfxdmodularityvalues}],
   Thread[{Range@10, singlerandomcommmodularityvalues}]}, Frame → True,
  FrameLabel → {"Time Windows", "Modularity (Single Random Graphs)"},
  ImageSize → 350, PlotRange → {{1, 10}, All}, PlotLabels →
   Placed[{"Erdös-Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Above}]],
 ListLinePlot[{Thread[{Range@10, ZscoresmodularityWolf[[All, 1]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 2]]}],
   Thread[{Range@10, ZscoresmodularityWolf[[All, 3]]}]},
  Frame → True, FrameLabel → {"Time Windows", "Z-Score for Modularity"},
  ImageSize → 350, PlotRange → All, PlotLabels →
   Placed[{"Erdös Renyi", "Degrees Fxd.", "Communities"}, {Scaled[1], Below}]]}
```

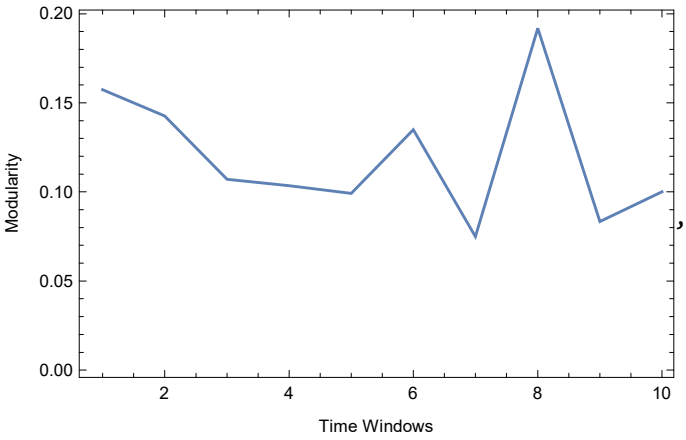
Out[•]=



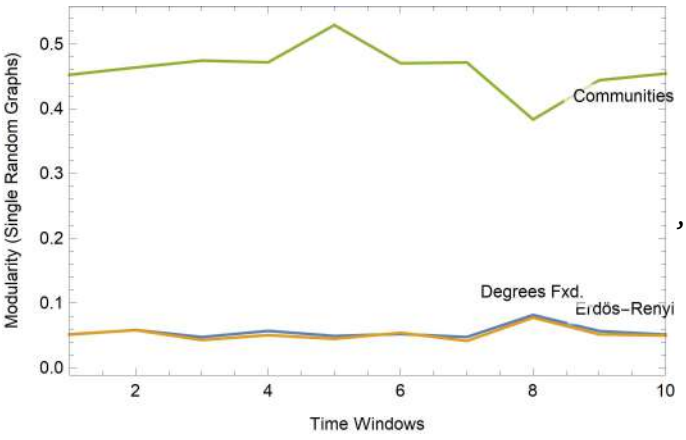
,


Avg. Betweenness Centrality
28
26
24
22
20
18
2
4
6
8
10
Time Windows

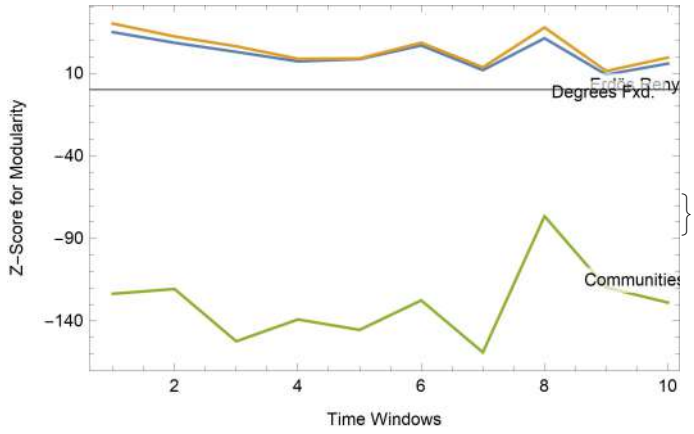
,


Modularity
0.20
0.15
0.10
0.05
0.00
2
4
6
8
10
Time Windows

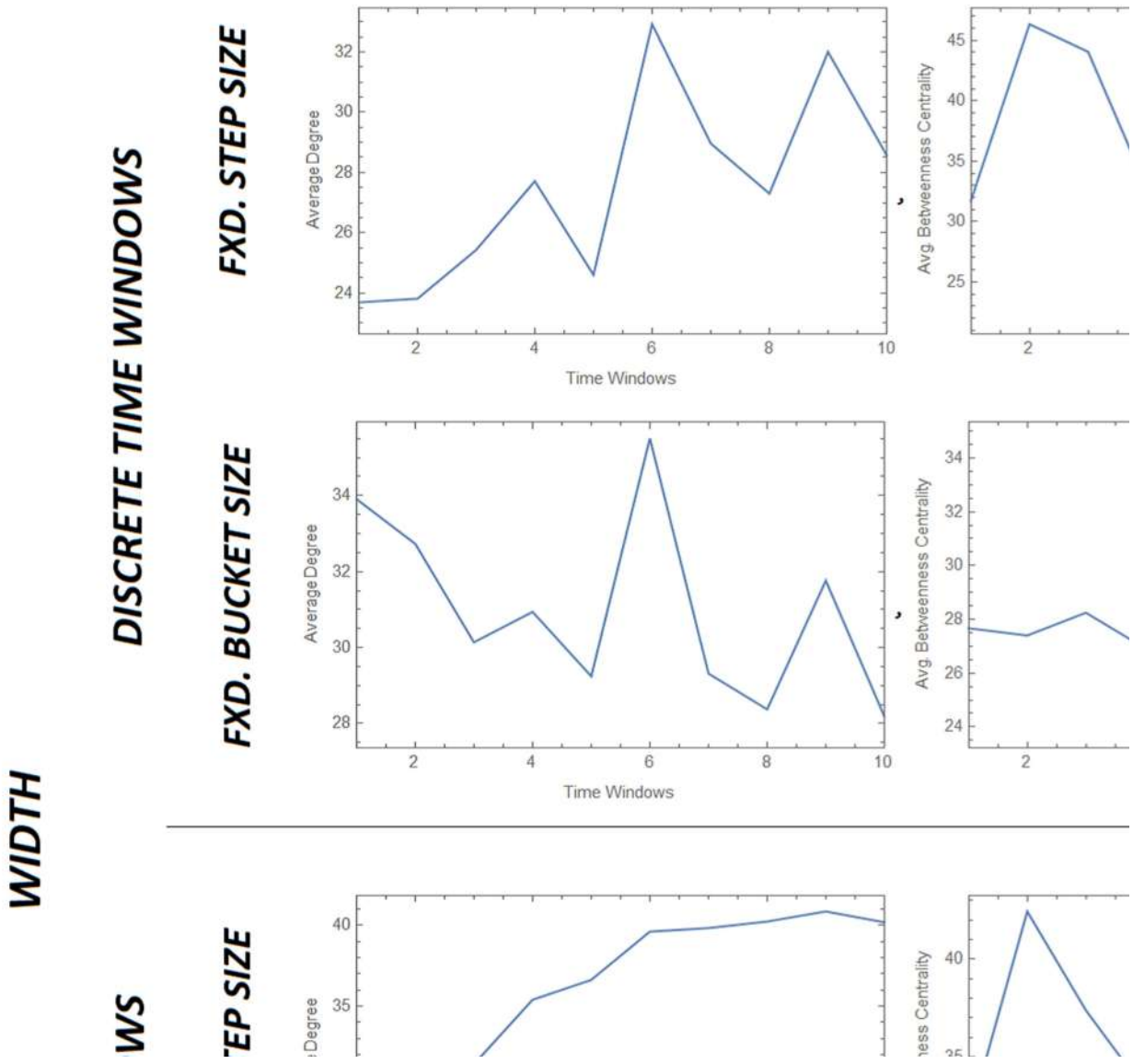
,


Modularity (Single Random Graphs)
0.5
0.4
0.3
0.2
0.1
0.0
Communities
Degrees Fxd.
Erdös–Renyi
2
4
6
8
10
Time Windows

,

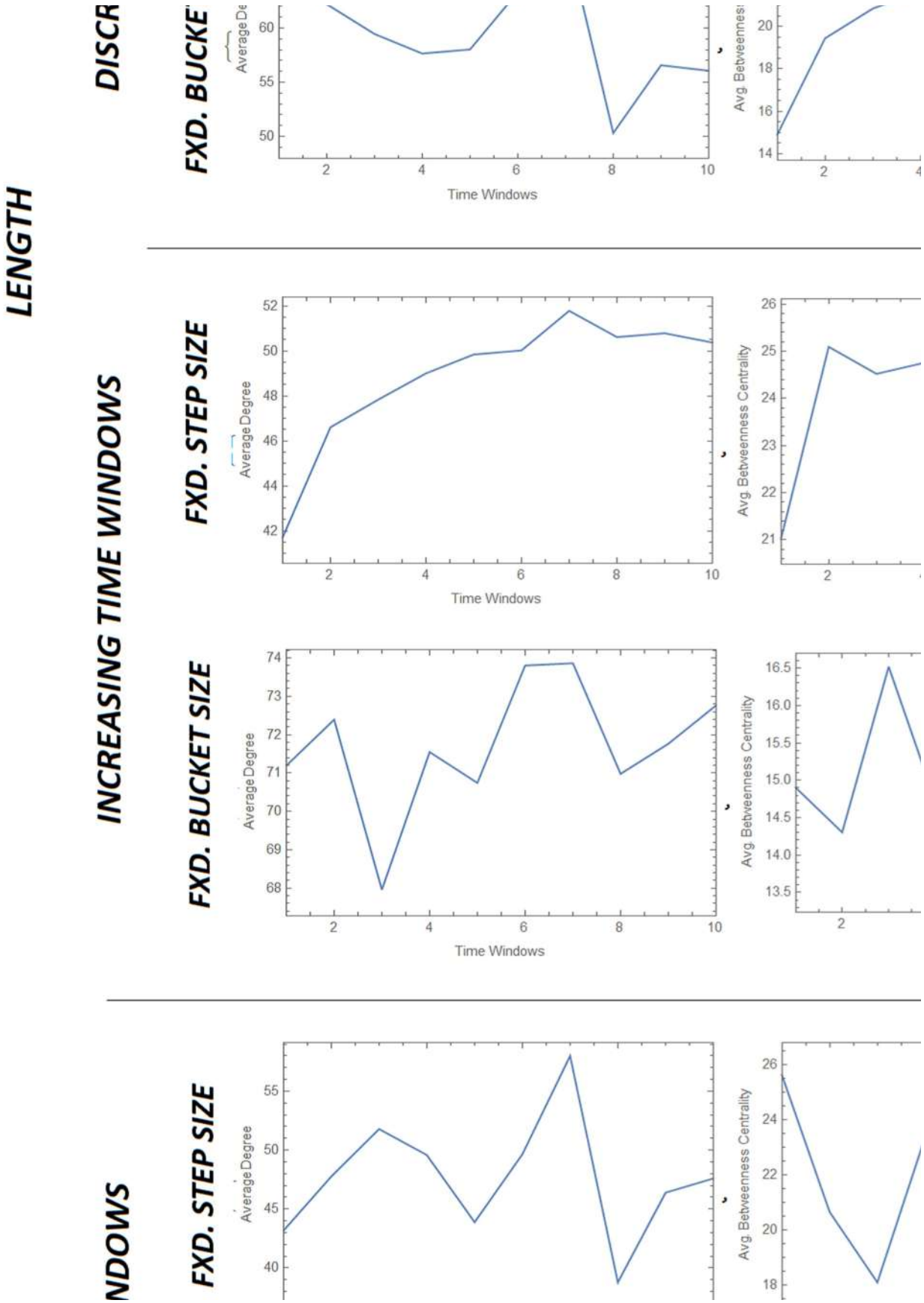


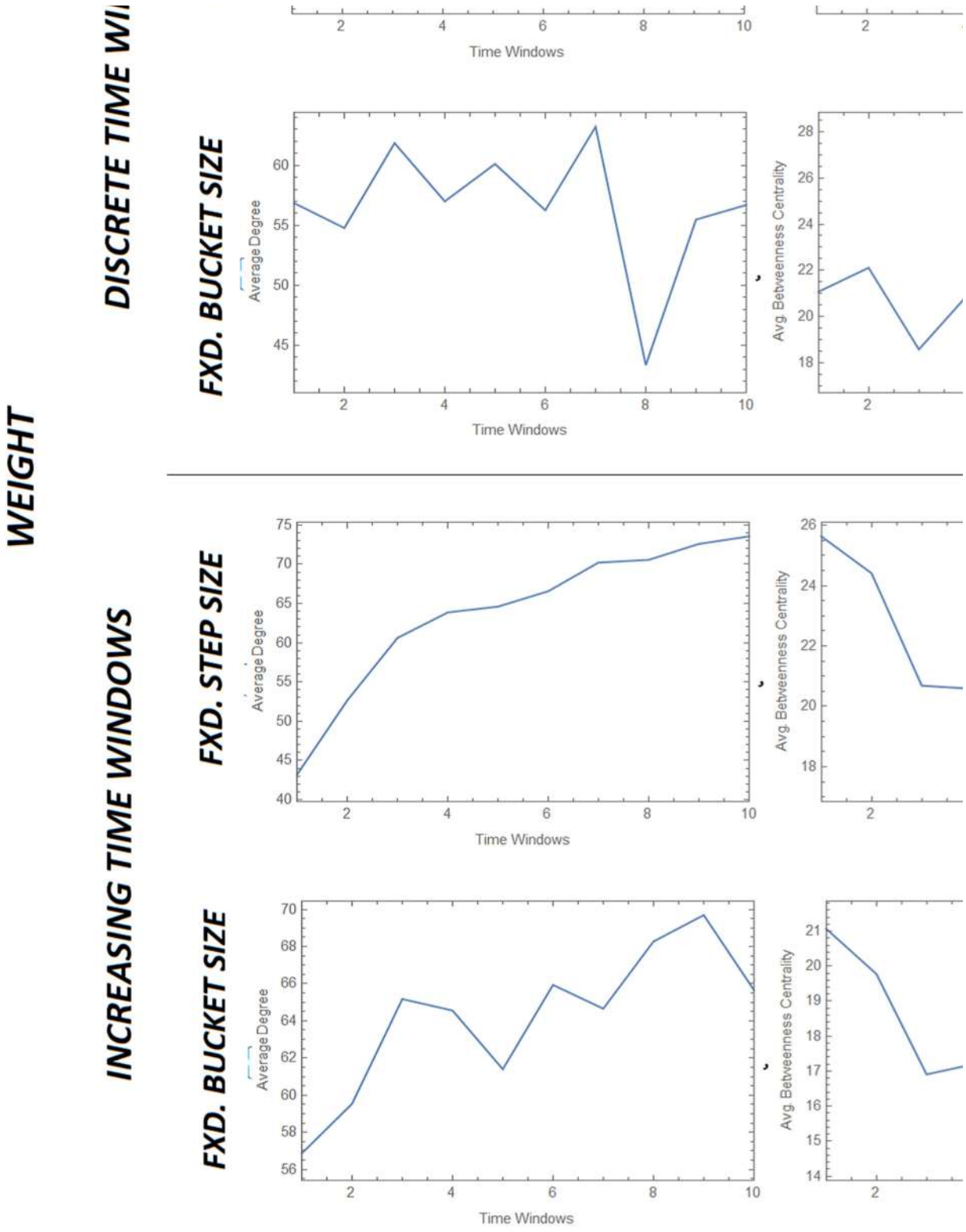
Plots for Network Metrics and Z-scores



INCREASING TIME WINDO
FXD. ST
Average
Time Windows
Avg. Betweenn
FXD. BUCKET SIZE
Average Degree
Time Windows
Avg. Betweenness Centrality
DISCRETE TIME WINDOWS
FXD. STEP SIZE
Average Degree
Time Windows
Avg. Betweenness Centrality
FXD. BUCKET SIZE
Average Degree
Avg. Betweenness Centrality

THICKN
INCREASING TIME WINDOWS
FXD. STEP SIZE
FXD. BUCKET SIZE
ETE TIME WINDOWS
T SIZE
Time Windows
Average Degree
Avg. Betweenness Centrality


LENGTH
DISCR
FXD. BUCKE
Average De
Time Windows
Avg Betweenness
INCREASING TIME WINDOWS
FXD. STEP SIZE
Average Degree
Time Windows
Avg Betweenness Centrality
FXD. BUCKET SIZE
Average Degree
Time Windows
Avg Betweenness Centrality
NDOWS
FXD. STEP SIZE
Average Degree
Avg Betweenness Centrality



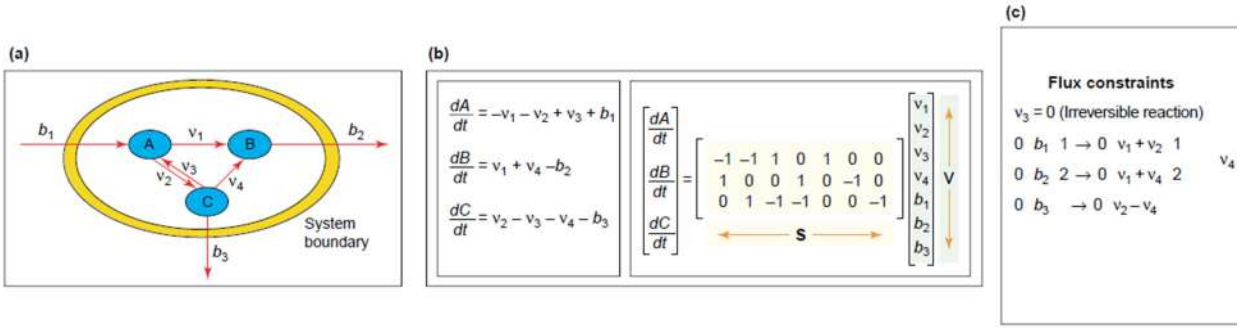
Modeling Works with OR Methods

FBA Model Reproducing from Kauffman et al. 2003

In[ ]:=
```
stochmatrix = {{-1, -1, 1, 0, 1, 0, 0}, {1, 0, 0, 1, 0, -1, 0}, {0, 1, -1, -1, 0, 0, -1}};
objectivefunc1 = {2, 0, 0, 1, 0, 0, 0};
objectivefunc2 = {1, 0, 0, 2, 0, 0, 0};
boundaries = {{0, Infinity}, {0, Infinity}, {0, 0}, {0, Infinity}, {0, 1}, {0, 2}, {0, 0}};
steadystatevector = {{0, 0}, {0, 0}, {0, 0}};
Vvector = {v1, v2, v3, v4, b1, b2, b3};
```

stochmatrix . Vvector = steadystatevector ， -objectivefunc ∈ solution space within boundaries

A, B, and C are metabolites. v_1, v_2, v_3, and v_4 are reactions. b_1, b_2, and b_3 are flux exchanges.

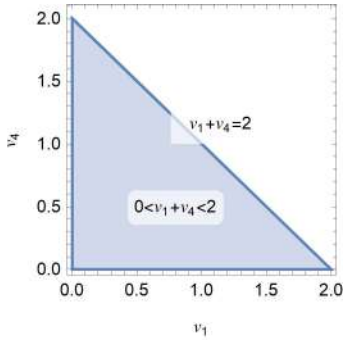



```
Print["1st objective function solution vector: ", Vvector1 =
  LinearProgramming[-objectivefunc1, stochmatrix, steadystatevector, boundaries]]
Print["2nd objective function solution vector: ", Vvector2 =
  LinearProgramming[-objectivefunc2, stochmatrix, steadystatevector, boundaries]]
```

```
1st objective function solution vector: {1, 0, 0, 0, 1, 1, 0}
2nd objective function solution vector: {0, 1, 0, 1, 1, 1, 0}
```

In[ ]:=
```
Show[RegionPlot[0 < v1 + v4 < 2, {v1, 0, 2}, {v4, 0, 2}, Frame -> True,
  PlotLabels → Placed["0<v1+v4<2", {0.8, 0.5}], FrameLabel → {"v1", "v4"}],
 Plot[-v1 + 2, {v1, 0, 1.2}, Frame -> True, PlotLabels → Placed["v1+v4=2", {1.1, 1}]],
 PlotRange → All, ImageSize -> Small]
```

Out[ ]=



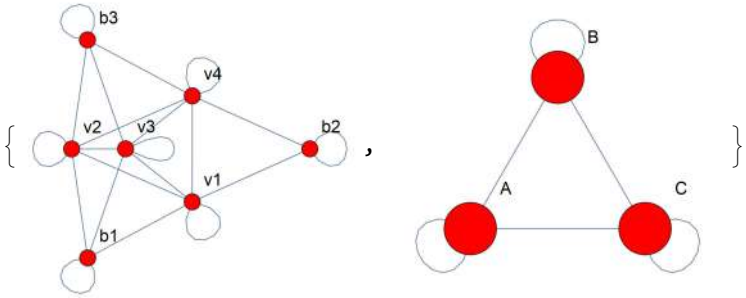
In[ ]:=
```
FindMaximum[{v4 + 2 v1, -v1 - v2 + v3 + b1 == 0 && v1 + v4 - b2 == 0 && v2 - v3 - v4 - b3 == 0 &&
    0 ≤ v1 + v2 ≤ 1 && 0 ≤ v1 + v4 ≤ 2 && v2 - v4 == 0 && 0 ≤ b1 ≤ 1 && 0 ≤ b2 ≤ 2 &&
    b3 == 0 && v1 ≥ 0 && v2 ≥ 0 && v3 == 0 && v4 ≥ 0}, {v1, v2, v3, v4, b1, b2, b3}]
```

Out[ ]=
```
{2., {v1 → 1., v2 → 0., v3 → 0., v4 → 0., b1 → 1., b2 → 1., b3 → 0.}}
```

In[ ]:=
```
AdjmatR = Transpose[stochmatrix].stochmatrix;
NormAdjmatR = AdjmatR /. x_ /; x ≠ 0 → 1;
AdjmatM = stochmatrix.Transpose[stochmatrix];
NormAdjmatM = AdjmatM /. x_ /; x ≠ 0 → 1;
```

In[ ]:=
```
{AdjacencyGraph[NormAdjmatR,
   {DirectedEdges → False, VertexSize → 0.3, VertexStyle → Red, VertexLabels →
     {1 -> "v1", 2 -> "v2", 3 -> "v3", 4 -> "v4", 5 -> "b1", 6 -> "b2", 7 -> "b3"}}],
 AdjacencyGraph[NormAdjmatM, {DirectedEdges → False, VertexSize → 0.3,
   VertexStyle → Red, VertexLabels → {1 -> "A", 2 -> "B", 3 -> "C"}}]}
```

Out[ ]=



Implementation Trials

```
stoichioforhomosapiens =
  Drop[Import["iAT_PLT_636_stoichiomat.csv", HeaderLines → 1], None, {1}];
SparseArray@stoichioforhomosapiens
```

SparseArray[ Specified elements: 4006
Dimensions: {738, 1008}
Default: 0.
Density: 0.00539
Elements:
{1, 12} → −1.
{1, 14} → 1.
{1, 345} → −1.
{1, 439} → −1.
⋮]

```
stoichiometricmatrix = stoichioforhomosapiens;
objectivefunctionamount = 200;
metabolites = 738;
fluxexchanges = 1008;
zerosforobj = Floor[0.7 * fluxexchanges, 1];
steadystatevector = ConstantArray[{0, 0}, metabolites];
SeedRandom[6];
boundaries = RandomChoice[{0.7, 0.3} → {{0, 500}, {0, 0}}, fluxexchanges];
SeedRandom[5];
objj = Table[RandomReal[{-2, 2}, fluxexchanges], objectivefunctionamount];
obj = Table[ReplacePart[i,
    Partition[RandomSample[Range@fluxexchanges, zerosforobj], 1] → 0.], {i, objj}];
obj // MatrixForm;
Solutionvectors = Table[LinearProgramming[-obj[[i]],
   stoichiometricmatrix, steadystatevector, boundaries], {i, Length@obj}]
```

In[ ]:=
```
Dimensions@Solutionvectors
```

Out[ ]=
```
{200, 1008}
```

In[ ]:=
```
nonzerocolumnsinsolutionvectorswithstoichioforhomosapiens =
  Position[Table[Count[Solutionvectors[[All, i]], x_ /; x ≠ 0.], {i, fluxexchanges}],
   objectivefunctionamount];
```

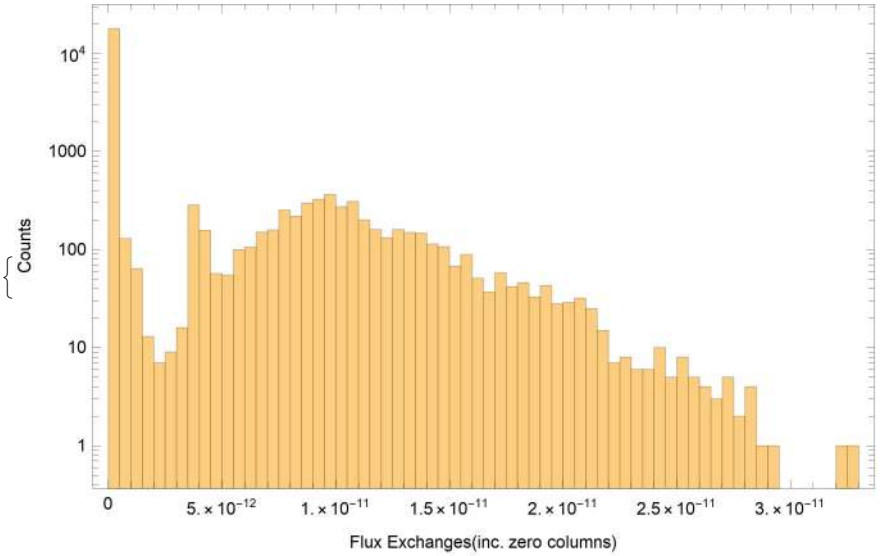
In[ ]:=
```
strial = stoichioforhomosapiens[[All,
   Flatten@nonzerocolumnsinsolutionvectorswithstoichioforhomosapiens]];
Dimensions@
 strial
```

Out[ ]=
```
{738, 396}
```

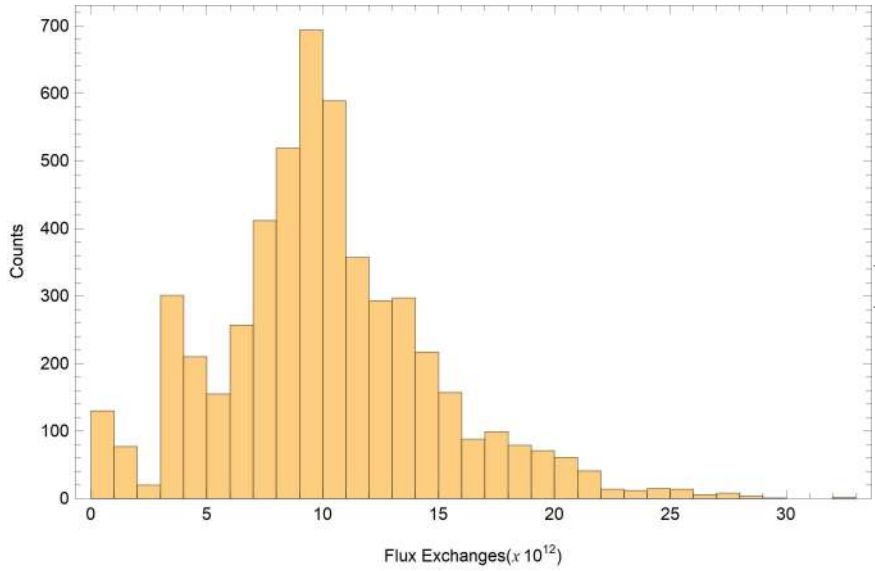
```
stoichiometricmatrix = strial;
objectivefunctionamount = 200;
metabolites = 738;
fluxexchanges = 396;
zerosforobj = Floor[0.7 * fluxexchanges, 1];
steadystatevector = ConstantArray[{0, 0}, metabolites];
SeedRandom[6];
boundaries = RandomChoice[{0.7, 0.3} → {{0, 500}, {0, 0}}, fluxexchanges];
SeedRandom[5];
objj = Table[RandomReal[{-2, 2}, fluxexchanges], objectivefunctionamount];
obj = Table[ReplacePart[i,
    Partition[RandomSample[Range@fluxexchanges, zerosforobj], 1] → 0.], {i, objj}];
obj // MatrixForm;
Solutionvectors = Table[LinearProgramming[-obj[[i]],
   stoichiometricmatrix, steadystatevector, boundaries], {i, Length@obj}]
```

```
In[ ]:= nonzerocolumnsinsolutionvectorswithstrial =
          Position[Table[Count[Solutionvectors[[All, i]], x_ /; x ≠ 0.], {i, fluxexchanges}],
           objectivefunctionamount];

In[ ]:= Dimensions@nonzerocolumnsinsolutionvectorswithstrial

Out[ ]= {116, 1}

In[ ]:= strial2 = strial[[All, Flatten@nonzerocolumnsinsolutionvectorswithstrial]];
        Dimensions@strial2

Out[ ]= {738, 116}

In[ ]:= stoichiometricmatrix = strial2;
        objectivefunctionamount = 200;
        metabolites = 738;
        fluxexchanges = 116;
        zerosforobj = Floor[0.7 * fluxexchanges, 1];
        steadystatevector = ConstantArray[{0, 0}, metabolites];
        SeedRandom[6];
        boundaries = RandomChoice[{0.7, 0.3} → {{0, 500}, {0, 0}}, fluxexchanges];
        SeedRandom[5];
        objj = Table[RandomReal[{-2, 2}, fluxexchanges], objectivefunctionamount];
        obj = Table[ReplacePart[i,
            Partition[RandomSample[Range@fluxexchanges, zerosforobj], 1] → 0.], {i, objj}];
        obj // MatrixForm;
        Solutionvectors = Table[LinearProgramming[-obj[[i]],
           stoichiometricmatrix, steadystatevector, boundaries], {i, Length@obj}]

        (* with 3rd trial, solution vectors resulted as zeros *)

In[ ]:= nonzerocolumnsinsolutionvectorswithstrial2 =
          Position[Table[Count[Solutionvectors[[All, i]], x_ /; x ≠ 0.], {i, fluxexchanges}],
           objectivefunctionamount];
        nonzerocolumnsinsolutionvectorswithstrial2

Out[ ]= {{22}, {23}, {25}, {26}, {39}, {40}, {51}, {53}, {54}, {55}, {56}, {62}, {63},
         {67}, {73}, {79}, {81}, {84}, {85}, {86}, {87}, {90}, {91}, {105}, {108}, {109}}

In[ ]:= networkfeaturesdata =
          Solutionvectors[[All, Flatten@nonzerocolumnsinsolutionvectorswithstrial2]] * 10^12;

In[ ]:= {Histogram[Flatten@Solutionvectors, Frame → True, ScalingFunctions → "Log",
          PlotRange → All, FrameLabel → {"Flux Exchanges(inc. zero columns)", "Counts"},
          ImageSize → 450], Histogram[Flatten@networkfeaturesdata, Frame → True,
          PlotRange → All, FrameLabel → {"Flux Exchanges (x 10^12)", "Counts"}, ImageSize → 450]}
```

Out[•]=



,



In[•]:=
```
Get["../algoritm_packages/SingleNetworks-algorithm-package.wl"]
(* ?SingleNetworks`* *)
```

In[•]:=
```
data = networkfeaturesdata;
Dimensions@data
```

Out[•]= {200, 26}

```
snetworkgraphsinglenodes[data[[]]]

(* snetworkgraphsinglenodes[aim_,campaign_,
 vertexsize_,vertexlabelsize_,imagesize_,vertexcolor_] *)
```