

Predicting and Efficiently Modeling Social Networks in Work Environments

There are two goals here.

- 1) Predict the real interaction frequencies using any available indicators including pre-existing proximity. The edge interaction frequencies are expected values. Since all possible edges have (possibly very small) non-zero expected value, this network is a complete graph.
- 2) Produce a computationally efficient network that can provide comparable cost estimates to 1) for any allocation.

Need to include various modules.

```
In[1]:= << Combinatorica`
```

```
In[2]:= << Histograms`
```

```
In[3]:= << HierarchicalClustering`
```

Predicting Interaction Frequencies and Cost

Number of nodes in the graph

```
In[211]:= nodes = 100
```

```
Out[211]= 100
```

Number of organizations

```
In[212]:= orgs = 5
```

```
Out[212]= 5
```

Generate a set of points randomly in the unit square. These represent the pre-existing allocation.

```
In[213]:= points = RandomReal[{0, 1}, {nodes, 2}]
```

```
Out[213]= {{0.0227528, 0.921143}, {0.697553, 0.578926}, {0.756413, 0.263904}, {0.158695, 0.879409},
{0.234351, 0.367106}, {0.805198, 0.668976}, {0.776511, 0.525406}, {0.757345, 0.0700208},
{0.925107, 0.548014}, {0.160594, 0.195309}, {0.540873, 0.748473}, {0.62142, 0.33447},
{0.323116, 0.104255}, {0.533331, 0.0153083}, {0.537144, 0.108067}, {0.859579, 0.95176},
{0.806581, 0.0251349}, {0.74047, 0.588511}, {0.93408, 0.798679}, {0.901725, 0.256776},
{0.984232, 0.339768}, {0.560359, 0.953807}, {0.255034, 0.782547}, {0.872511, 0.910297},
{0.575288, 0.708571}, {0.62884, 0.449881}, {0.84967, 0.307338}, {0.408387, 0.442263},
{0.27196, 0.408409}, {0.294967, 0.839944}, {0.234151, 0.208085}, {0.863248, 0.357641},
{0.999655, 0.015023}, {0.368424, 0.768063}, {0.653948, 0.2052}, {0.865673, 0.716182},
{0.865014, 0.189436}, {0.123679, 0.611979}, {0.0557611, 0.56615}, {0.576108, 0.154402},
{0.606924, 0.990442}, {0.931908, 0.187068}, {0.202017, 0.262864}, {0.859249, 0.863681},
{0.604146, 0.778798}, {0.822259, 0.971978}, {0.0289493, 0.566913}, {0.32616, 0.0195857},
{0.555516, 0.991426}, {0.244224, 0.820216}, {0.0880962, 0.257836}, {0.912521, 0.0285247},
{0.48082, 0.316335}, {0.970213, 0.0445751}, {0.647453, 0.633048}, {0.298062, 0.896592},
{0.915568, 0.650645}, {0.655096, 0.0335565}, {0.610232, 0.515612}, {0.634052, 0.602577},
{0.932056, 0.834487}, {0.548868, 0.115068}, {0.240963, 0.315278}, {0.772589, 0.334549},
{0.40799, 0.692979}, {0.718554, 0.25195}, {0.220628, 0.423291}, {0.199902, 0.502708},
{0.483567, 0.865436}, {0.485266, 0.368363}, {0.0468272, 0.441996}, {0.380164, 0.461219},
{0.331862, 0.908292}, {0.511541, 0.071544}, {0.980868, 0.383993}, {0.379072, 0.306559},
{0.339537, 0.32771}, {0.641397, 0.070617}, {0.1283, 0.990855}, {0.675502, 0.51316},
{0.412441, 0.974361}, {0.90403, 0.998995}, {0.567777, 0.363596}, {0.830615, 0.579211},
{0.171311, 0.166433}, {0.0353376, 0.0983618}, {0.0273757, 0.0642769}, {0.898557, 0.708878},
{0.0612559, 0.472583}, {0.33976, 0.38275}, {0.132011, 0.131786}, {0.122053, 0.510312},
{0.512445, 0.479185}, {0.272499, 0.998603}, {0.645964, 0.18648}, {0.499137, 0.29984},
{0.91593, 0.0931559}, {0.609232, 0.0627998}, {0.728319, 0.742585}, {0.810176, 0.520445}}
```

Generate a set of random points representing organizational mass centers

```
In[214]:= orgPoints = RandomReal[{0, 1}, {orgs, 2}]
```

```
Out[214]= {{0.578864, 0.83516}, {0.412174, 0.217201},
{0.587647, 0.402024}, {0.410466, 0.920864}, {0.167807, 0.375698}}
```

Randomly assign organizations to nodes using the inverse distances to each orgPoint as the probability weights

```
In[215]:= nodeOrgs = Map[RandomChoice[# -> Range[orgs]] &, Partition[
Map[1 / EuclideanDistance[#[[1]], #[[2]]]^6 &, Tuples[{points, orgPoints}]], orgs]]
```

```
Out[215]= {4, 4, 3, 4, 5, 3, 3, 2, 3, 5, 1, 3, 2, 2, 2, 1, 1, 1, 1, 2, 3, 4, 4, 1, 1, 3, 3, 2, 5, 4, 5, 3, 2, 4,
3, 1, 3, 5, 5, 2, 1, 3, 5, 1, 1, 1, 5, 2, 4, 4, 5, 2, 3, 2, 1, 4, 3, 2, 3, 3, 4, 2, 5, 3, 1, 3, 5,
5, 1, 3, 5, 5, 4, 2, 3, 2, 2, 2, 4, 3, 4, 1, 3, 1, 2, 5, 5, 1, 5, 2, 5, 5, 3, 4, 3, 2, 3, 2, 1, 1}
```

```

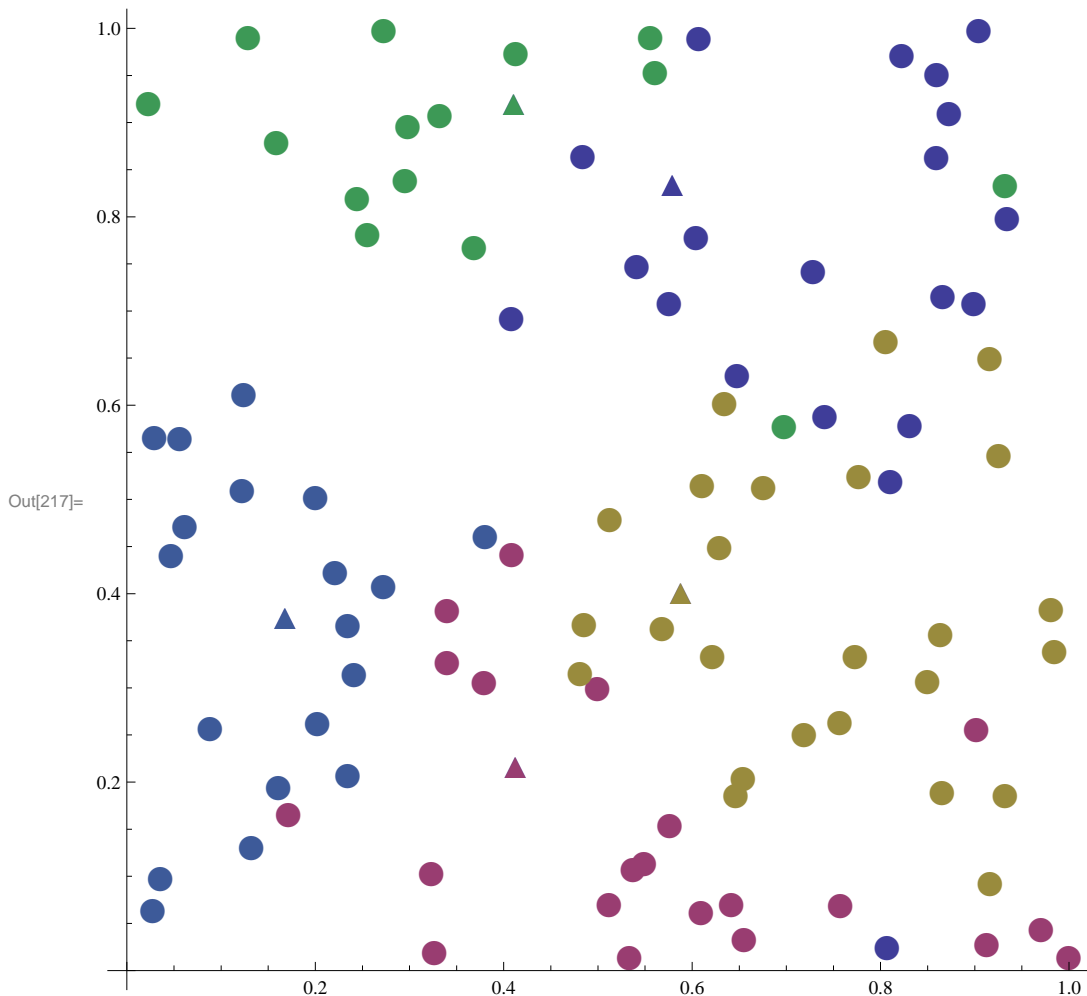
In[216]:= pointsByOrg = Table[Pick[points, Map[If[o == #, True, False] &, nodeOrgs]], {o, orgs}]

Out[216]= {{{0.540873, 0.748473}, {0.859579, 0.95176}, {0.806581, 0.0251349}, {0.74047, 0.588511},
{0.93408, 0.798679}, {0.872511, 0.910297}, {0.575288, 0.708571}, {0.865673, 0.716182},
{0.606924, 0.990442}, {0.859249, 0.863681}, {0.604146, 0.778798}, {0.822259, 0.971978},
{0.647453, 0.633048}, {0.40799, 0.692979}, {0.483567, 0.865436}, {0.90403, 0.998995},
{0.830615, 0.579211}, {0.898557, 0.708878}, {0.728319, 0.742585}, {0.810176, 0.520445}},
{{0.757345, 0.0700208}, {0.323116, 0.104255}, {0.533331, 0.0153083},
{0.537144, 0.108067}, {0.901725, 0.256776}, {0.408387, 0.442263}, {0.999655, 0.015023},
{0.576108, 0.154402}, {0.32616, 0.0195857}, {0.912521, 0.0285247},
{0.970213, 0.0445751}, {0.655096, 0.0335565}, {0.548868, 0.115068},
{0.511541, 0.071544}, {0.379072, 0.306559}, {0.339537, 0.32771}, {0.641397, 0.070617},
{0.171311, 0.166433}, {0.33976, 0.38275}, {0.499137, 0.29984}, {0.609232, 0.0627998}},
{{0.756413, 0.263904}, {0.805198, 0.668976}, {0.776511, 0.525406},
{0.925107, 0.548014}, {0.62142, 0.33447}, {0.984232, 0.339768}, {0.62884, 0.449881},
{0.84967, 0.307338}, {0.863248, 0.357641}, {0.653948, 0.2052}, {0.865014, 0.189436},
{0.931908, 0.187068}, {0.48082, 0.316335}, {0.915568, 0.650645},
{0.610232, 0.515612}, {0.634052, 0.602577}, {0.772589, 0.334549}, {0.718554, 0.25195},
{0.485266, 0.368363}, {0.980868, 0.383993}, {0.675502, 0.51316}, {0.567777, 0.363596},
{0.512445, 0.479185}, {0.645964, 0.18648}, {0.91593, 0.0931559}},
{{0.0227528, 0.921143}, {0.697553, 0.578926}, {0.158695, 0.879409},
{0.560359, 0.953807}, {0.255034, 0.782547}, {0.294967, 0.839944}, {0.368424, 0.768063},
{0.555516, 0.991426}, {0.244224, 0.820216}, {0.298062, 0.896592}, {0.932056, 0.834487},
{0.331862, 0.908292}, {0.1283, 0.990855}, {0.412441, 0.974361}, {0.272499, 0.998603}},
{{0.234351, 0.367106}, {0.160594, 0.195309}, {0.27196, 0.408409}, {0.234151, 0.208085},
{0.123679, 0.611979}, {0.0557611, 0.56615}, {0.202017, 0.262864}, {0.0289493, 0.566913},
{0.0880962, 0.257836}, {0.240963, 0.315278}, {0.220628, 0.423291}, {0.199902, 0.502708},
{0.0468272, 0.441996}, {0.380164, 0.461219}, {0.0353376, 0.0983618},
{0.0273757, 0.0642769}, {0.0612559, 0.472583}, {0.132011, 0.131786}, {0.122053, 0.510312}}}

```

Plot of the nodes, colored by organization. The triangles are the organization mass centers. Note that there are two different shades of blue.

```
In[217]:= Show[ListPlot[Map[{-#} &, orgPoints], PlotMarkers → Magnify[▲, 2]],
ListPlot[pointsByOrg, PlotMarkers → Magnify[●, 2]],
{AspectRatio → Automatic, AxesOrigin → {0, 0}, PlotRange → {{0, 1}, {0, 1}}}]
```



Define all possible edges as pairs of indices into the points array

```
In[218]:= edges = Subsets[Range[nodes], {2}]
```

A very large output was generated. Here is a sample of it:

```
Out[218]:= {{1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7}, {1, 8}, {1, 9}, {1, 10}, {1, 11},
{1, 12}, {1, 13}, {1, 14}, {1, 15}, {1, 16}, {1, 17}, {1, 18}, {1, 19}, {1, 20},
{1, 21}, {1, 22}, {1, 23}, {1, 24}, {1, 25}, {1, 26}, <<4900>>, {93, 97}, {93, 98},
{93, 99}, {93, 100}, {94, 95}, {94, 96}, {94, 97}, {94, 98}, {94, 99}, {94, 100},
{95, 96}, {95, 97}, {95, 98}, {95, 99}, {95, 100}, {96, 97}, {96, 98}, {96, 99},
{96, 100}, {97, 98}, {97, 99}, {97, 100}, {98, 99}, {98, 100}, {99, 100}}
```

Show Less

Show More

Show Full Output

Set Size Limit...

A convenience function to define the distance between nodes using a points array for the locations

```
In[219]:= distance[points_, {a_, b_}] := EuclideanDistance[points[[a]], points[[b]]]
```

The indicators for interaction frequency are assumed to be independent, so we will define each while holding the other indicators constant. We will only define two indicators here, one based on existing spatial proximity and the other based on organization. Other indicators might be role (manager, administrative staff, etc) or more direct indicators like telephone and email traffic logs. If each indicator is independent, we should be able to multiply them together to get the composite indicator. The composite indicators will be scaled by a constant factor so they represent a fraction of the total expected number of interactions across all the edges.

For the spatial proximity indicator, we'll assume that interactions are proportional to the inverse square of the distance. So, an edge twice as long will have 1/4 the number of interactions. (Assumes common value for other indicators, i.e. both edges have nodes with the same (or different) organizations)

```
In[220]:= DistanceIndicator[d_] := 1 / d^2
```

Another indicator is organization. If two edges have equal length and the nodes of the first edge are in the same organization, while the nodes of the second edge are in different organizations, we'll assume that interactions across the first edge occur with 10 times the frequency.

```
In[221]:= OrgIndicator[orgs_, edge_] := If[orgs[[edge[[1]]]] == orgs[[edge[[2]]]], 10, 1]
```

The relevant normalization constant for this network is

```
In[222]:= K = 1 /
Sum[DistanceIndicator[distance[points, edge]] OrgIndicator[nodeOrgs, edge], {edge, edges}]
```

```
Out[222]= 1.85954 × 10-6
```

And the resulting scaled values for interaction frequency expressed as fractions of the total expected number of interactions

```
In[223]:= i = Map[K DistanceIndicator[distance[points, #]] OrgIndicator[nodeOrgs, #] &, edges]
```

A very large output was generated. Here is a sample of it:

```
Out[223]= {0.0000324828, 1.91662 × 10-6, 0.000919564, 5.28682 × 10-6, 2.75158 × 10-6, 2.56574 × 10-6,
1.47111 × 10-6, 1.95029 × 10-6, <<4934>>, 7.48164 × 10-6, 0.0000127881, 0.0000195772,
4.06941 × 10-6, 9.59713 × 10-6, 3.90422 × 10-6, 7.44358 × 10-6, 0.000331785}
```

Show Less

Show More

Show Full Output

Set Size Limit...

```
In[224]:= Sum[x, {x, i}]
```

```
Out[224]= 1.
```

This will bind the edge node indices to the values of interaction frequency for that edge

```
In[225]:= iedges = Transpose[{edges, i}]
```

A very large output was generated. Here is a sample of it:

```
Out[225]= {{1, 2}, 0.0000324828}, {{1, 3}, 1.91662 × 10-6}, {{1, 4}, 0.000919564},
{{1, 5}, 5.28682 × 10-6}, {{1, 6}, 2.75158 × 10-6}, {{1, 7}, 2.56574 × 10-6},
<<4939>>, {{97, 99}, 4.06941 × 10-6}, {{97, 100}, 9.59713 × 10-6},
{{98, 99}, 3.90422 × 10-6}, {{98, 100}, 7.44358 × 10-6}, {{99, 100}, 0.000331785}}
```

Show Less

Show More

Show Full Output

Set Size Limit...

The cost is defined as the sum over all edges of the distance times the interaction frequency. Note that the cost for a real network should be scaled by the total expected number of interactions and by the cost per linear unit.

```
In[226]:= cost[points_, iedges_] := Sum[iedge[[2]] distance[points, iedge[[1]]], {iedge, iedges}]
```

So the cost for the pre-existing allocation is

```
In[227]:= initialCost = cost[points, iedges]
```

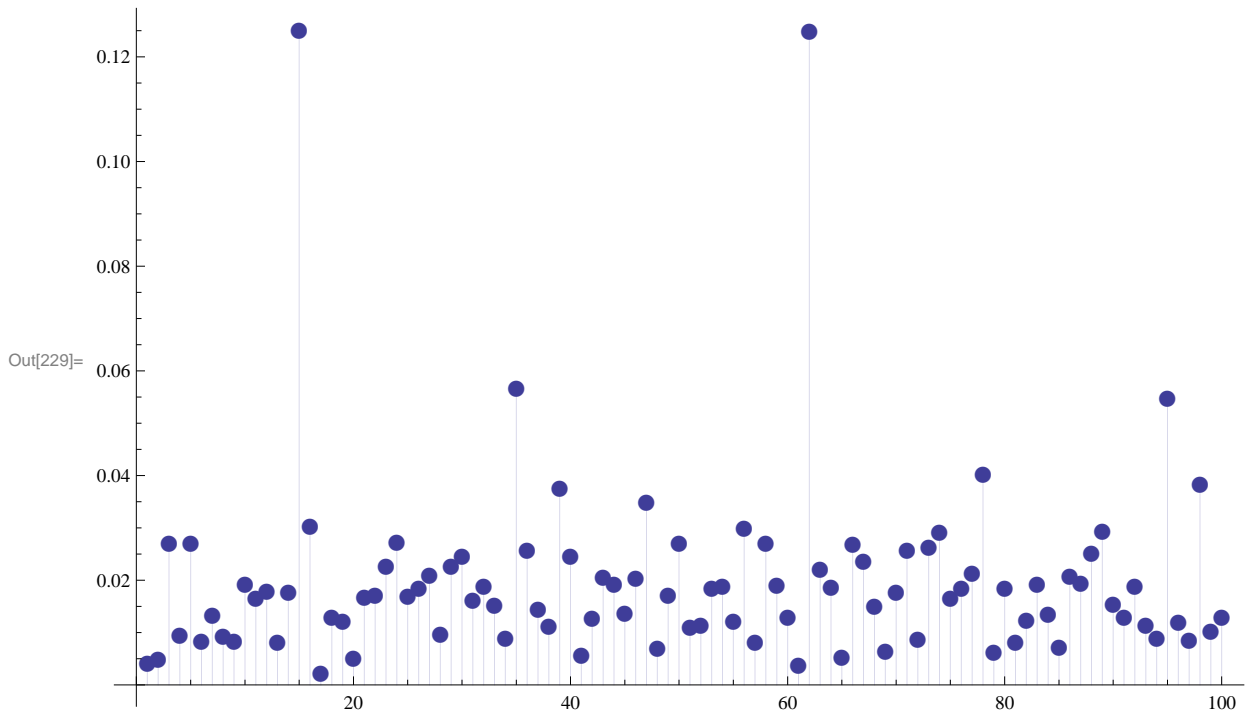
```
Out[227]= 0.108957
```

It may be of interest to see the distribution of interactions associated with each node (summing across all it's edges).

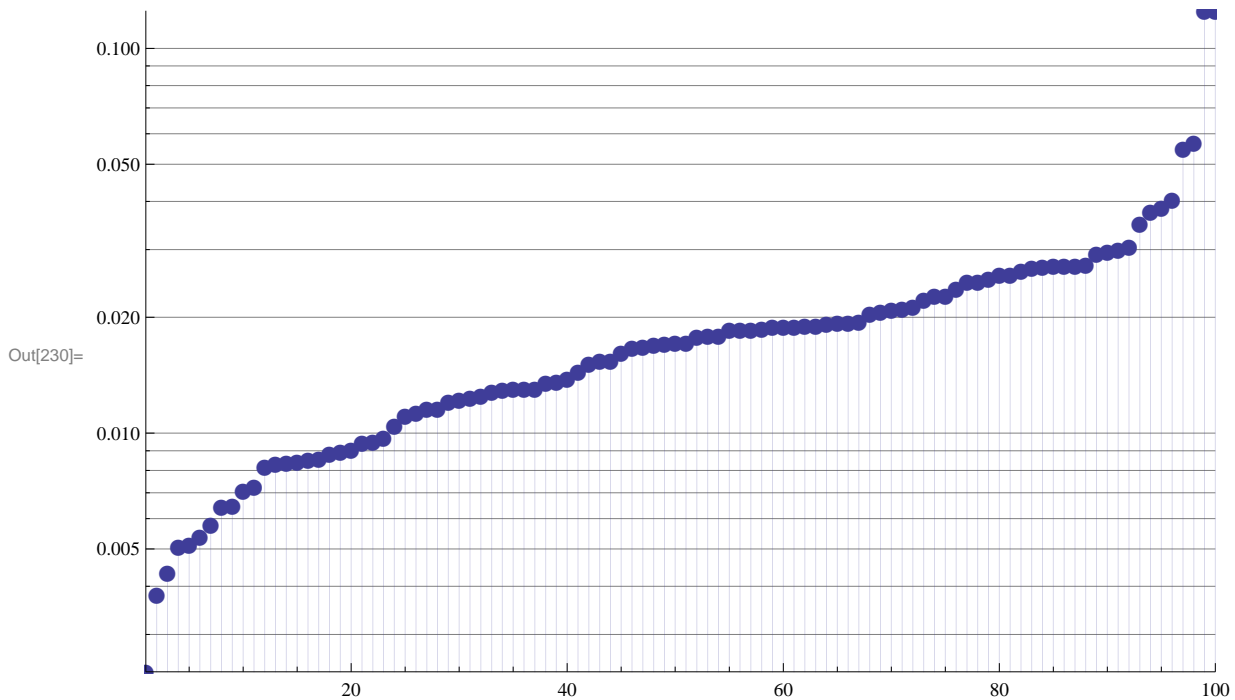
```
In[228]:= nodeTotals = Table[Total[Select[iedges, MemberQ[#[[1]], x] &][[All, 2]]], {x, 1, nodes}]
```

```
Out[228]= {0.00434446, 0.00506114, 0.0272704, 0.00951328, 0.027151, 0.00841237, 0.0135141, 0.00945758,
0.00851899, 0.019394, 0.0167373, 0.0179754, 0.0083197, 0.0177643, 0.125191, 0.0304965,
0.00239803, 0.0129883, 0.0122237, 0.00512669, 0.0169383, 0.0171898, 0.0228156,
0.0274131, 0.0170416, 0.0186693, 0.0211005, 0.00970824, 0.0227196, 0.0247307, 0.0162076,
0.0190574, 0.0154082, 0.00907865, 0.0567474, 0.0257564, 0.0144769, 0.0113144,
0.0377416, 0.0247841, 0.00579041, 0.0128359, 0.020664, 0.0193344, 0.0138373, 0.0204976,
0.0350325, 0.00710107, 0.0171598, 0.027165, 0.0110955, 0.0115734, 0.0186318, 0.0189248,
0.0123689, 0.0300718, 0.00818556, 0.027269, 0.0190743, 0.0130276, 0.00379912, 0.124948,
0.0222936, 0.0188599, 0.00536799, 0.0269179, 0.0237376, 0.0151144, 0.00646574,
0.0178823, 0.0257999, 0.00886788, 0.0264418, 0.0293055, 0.0166539, 0.0185429, 0.0213587,
0.0403791, 0.006431, 0.018528, 0.00836032, 0.0124822, 0.0192732, 0.0136344, 0.0072826,
0.0208937, 0.019498, 0.0252123, 0.0295553, 0.0154707, 0.0130517, 0.0189496, 0.0115948,
0.00897187, 0.0549644, 0.0120561, 0.00860968, 0.0385454, 0.0104362, 0.0130639}
```

```
In[229]:= ListPlot[nodeTotals,
  {Filling -> Axis, AxesOrigin -> {0, 0}, PlotRange -> All, PlotMarkers -> {Automatic, Medium}}]
```



```
In[230]:= ListLogPlot[Sort[nodeTotals], {Filling -> Axis, PlotRange -> All,
  GridLines -> {None, Automatic}, PlotMarkers -> {Automatic, Medium}}]
```



The previous graph, plotted on a logarithmic scale, would seem to indicate that the network is dominated by a small number of nodes with very high interaction frequencies.

Can we simplify the graph?

Suppose we wish to limit the computational complexity by choosing a reasonable number of edges per node on average

```
In[231]:= edgesPerNode = 10
```

```
Out[231]= 10
```

The total number of edges in the subgraph is therefore

```
In[232]:= edgesInSubGraph = nodes edgesPerNode / 2
```

```
Out[232]= 500
```

■ Random Sampling Approach

An early approach focused on using the edges most likely to affect the cost, those with the largest interaction frequency. The results showed that the subgraph produced similar cost differences compared to the full graph, but the cost curves gradually diverged from each other. A better approach seems to be to use a roulette wheel selection process by building a map on the interval 0,1 based on the expected value of every possible edge and then building the graph and edge weights (interaction frequencies) by randomly sampling. Random sampling stops when we have the desired number of edges.

```
In[233]:= genRandomSample[objects_, weights_, num_] :=
  Module[{w = Map[#/Total[weights] &, Accumulate[weights]], e = {}, t = {}, i, j, c = 0},
    While[Length[e] < num, c++; i = BinarySearch[w, RandomReal[] + 1/2; j = Position[e, i];
      If[Length[j] == 0, e = Append[e, i]; t = Append[t, 1], t[[j[[1, 1]]]++]];
      Print[c]; Map[{objects[[#[[1]]]], N[#[[2]]/c]} &, Transpose[{e, t}]]]
```

```
In[234]:= ranedges = genRandomSample[iedges[[All, 1]], iedges[[All, 2]], edgesInSubGraph]
```

```
1704
```

A very large output was generated. Here is a sample of it:

```
Out[234]= {{{{4, 30}, 0.000586854}, {{15, 62}, 0.111502}, {{55, 82}, 0.00117371},
  {{59, 80}, 0.00469484}, {{23, 30}, 0.00410798}, {{3, 66}, 0.0129108},
  <<489>>, {{49, 73}, 0.000586854}, {{77, 78}, 0.000586854},
  {{32, 83}, 0.000586854}, {{41, 56}, 0.000586854}, {{50, 73}, 0.000586854}}
```

[Show Less](#)
[Show More](#)
[Show Full Output](#)
[Set Size Limit...](#)

```
In[235]:= edgeHasNode[n3_, {{n1_, n2_}, _}] := n3 == n1 || n3 == n2
```

```
In[236]:= edgeToNode[n3_, {{n1_, n2_}, i_}] := {If[n3 == n1, n2, n1], i}
```

```

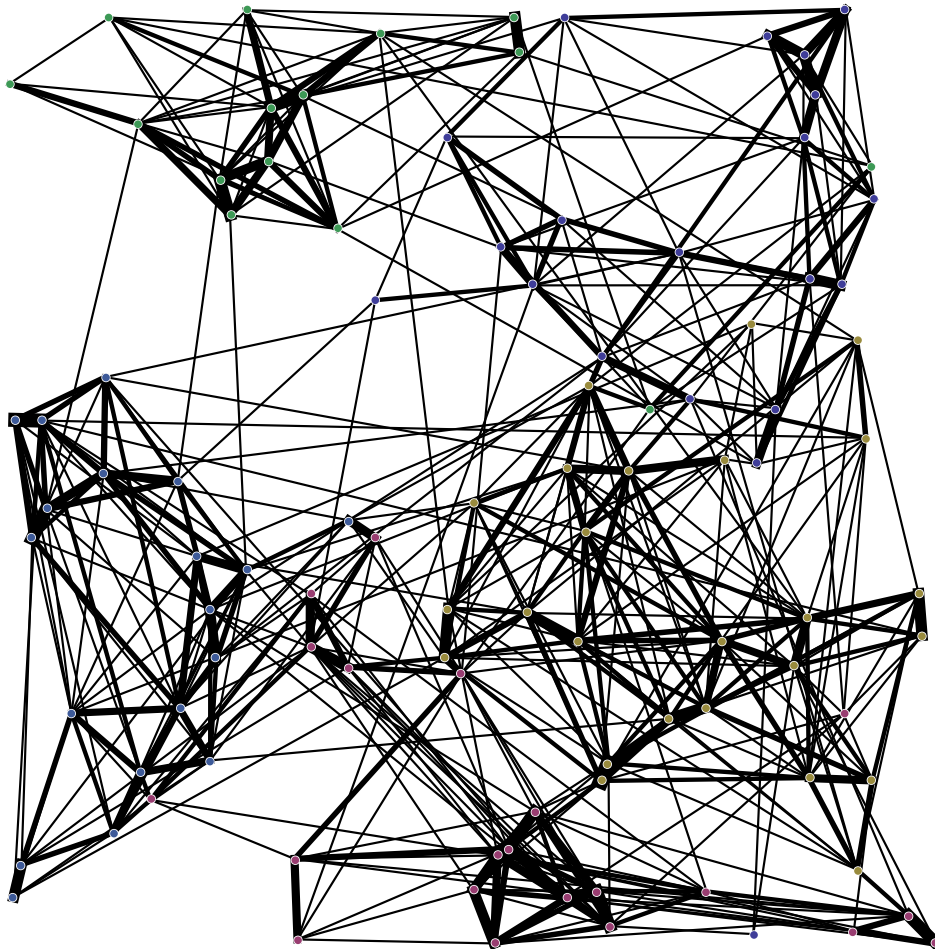
In[237]:= ShowWeightedSubGraph[points_, iedges_] :=
DynamicModule[{offset, nodeMarkers, nodeEdges, thick, range, zoom = 0.7, cx = 0.5, cy = 0.5},
  offset = 1 - Log[2, Min[iedges[[All, 2]]]];
  thick[x_] := AbsoluteThickness[Log[2, x] + offset];
  range[x_, y_, z_] := {{Dynamic[x] - Dynamic[z], Dynamic[x] + Dynamic[z]},
    {Dynamic[y] - Dynamic[z], Dynamic[y] + Dynamic[z]}};
  nodeMarkers = MapIndexed[{White, Disk[#1, 0.005],
    ColorData[1, nodeOrgs[[First[#2]]]], Disk[#1, 0.004]} &, points];
  nodeEdges = Table[Join[Transpose[
    Map[{{White, thick[#[[2]] + 1], Line[#[[1]]]}, {Blue, thick[#[[2]]], Line[#[[1]]}}] &,
    Map[{points[[{x, edgeToNode[x, #][[1]]}]], #[[2]]] &,
    Select[iedges, edgeHasNode[x, #] &]]], nodeMarkers], {x, nodes}];
  Column[{Slider[Dynamic[zoom], {0.1, 0.7}],
    EventHandler[Graphics[{Map[{thick[#[[2]]], Line[points[[#[[1, 1 ;; 2]]]]]} &, iedges],
      Orange, MapIndexed[Mouseover[nodeMarkers[[#2]], nodeEdges[[#2]]] &, points]},
    PlotRange -> range[Dynamic[cx], Dynamic[cy], Dynamic[zoom]]],
    {"MouseClicked" -> ({cx, cy} = MousePosition["Graphics"]), PassEventsDown -> True}]]]

```

```
In[238]:= ShowWeightedSubGraph[points, ranedges]
```



```
Out[238]=
```



In the above graph, we show the nodes colored by organization, along with the edges, where the thickness is related to the interaction frequency.

The subgraph produces very similar cost to the full graph.

```
In[239]:= initialCostRan = cost[points, ranedges]
```

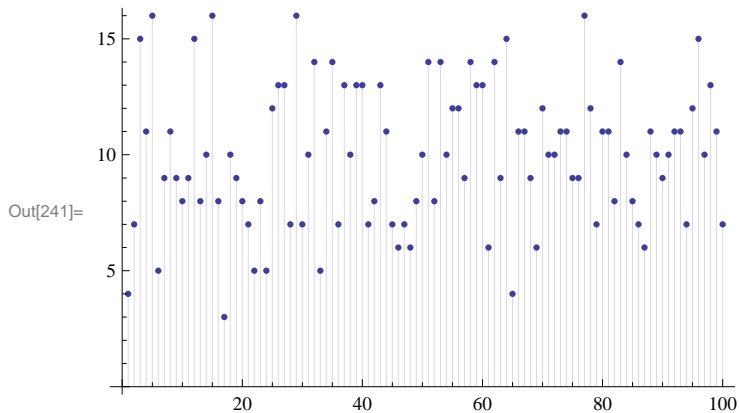
```
Out[239]= 0.107584
```

```
In[240]:= initialCost
```

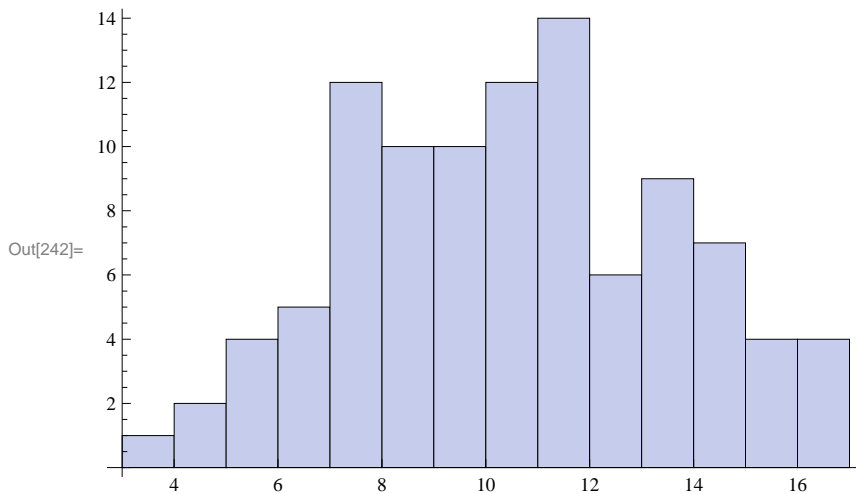
```
Out[240]= 0.108957
```

The following plots show that the number of edges per node appears to be normally distributed around the desired average value.

```
In[241]:= ListPlot[Tally[Flatten[rangedges[[All, 1]]]], {Filling -> Axis, AxesOrigin -> {0, 0}}
```



```
In[242]:= Histogram[Tally[Flatten[rangedges[[All, 1]]]][[All, 2]]]
```

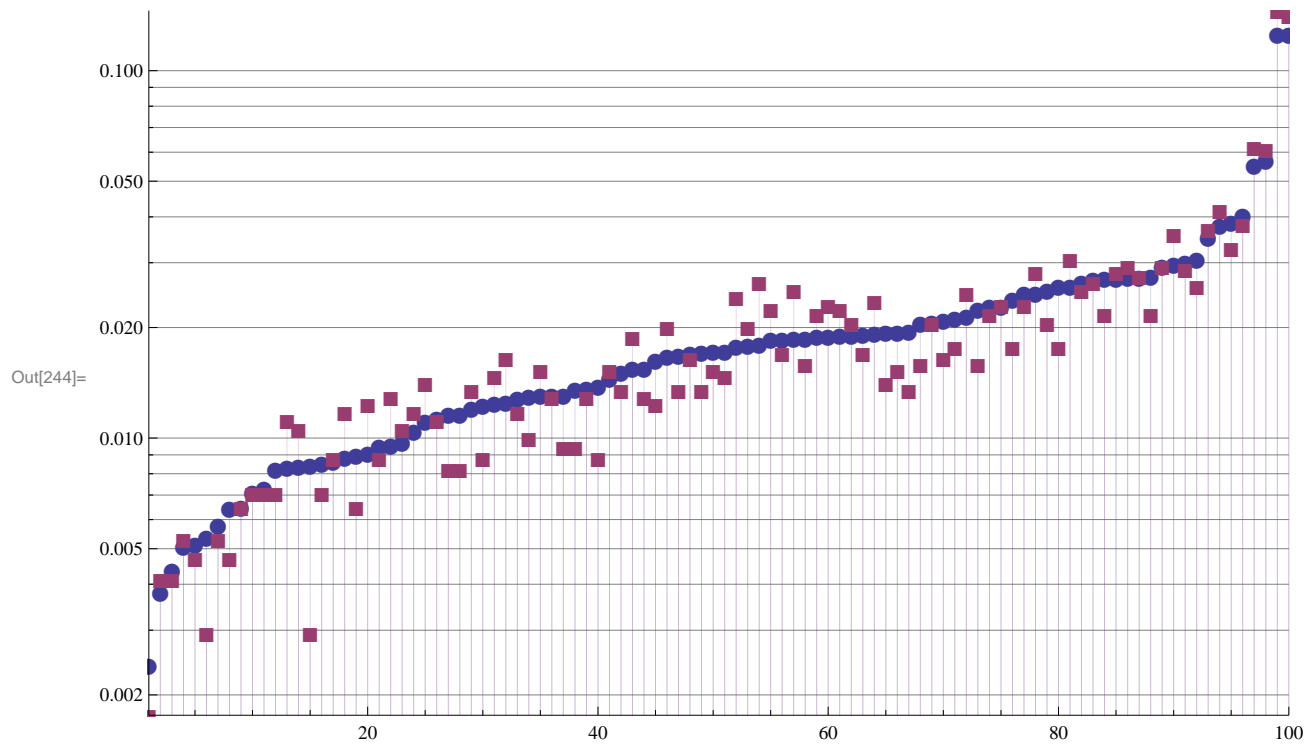


Let's see how the distribution of interactions across the nodes compares with the full graph.

```
In[243]:= nodeTotalsRan = Table[Total[Select[rangedges, MemberQ[#][1], x] &][[All, 2]]], {x, 1, nodes}]
```

```
Out[243]= {0.00410798, 0.00528169, 0.0275822, 0.0129108, 0.0217136, 0.00293427, 0.00938967,
0.00880282, 0.00704225, 0.0152582, 0.0134977, 0.0264085, 0.0111502, 0.024061,
0.140845, 0.0258216, 0.00176056, 0.00997653, 0.00880282, 0.00469484, 0.0164319,
0.0146714, 0.0228873, 0.0217136, 0.0134977, 0.0158451, 0.0176056, 0.0105634, 0.0217136,
0.0228873, 0.0123239, 0.0205399, 0.0187793, 0.0123239, 0.0610329, 0.0176056, 0.0152582,
0.0111502, 0.0416667, 0.028169, 0.00528169, 0.0117371, 0.0205399, 0.0140845,
0.00880282, 0.0158451, 0.0369718, 0.00704225, 0.0152582, 0.028169, 0.0140845,
0.00821596, 0.0252347, 0.0228873, 0.0146714, 0.0287559, 0.00704225, 0.0293427,
0.0170188, 0.0152582, 0.00410798, 0.14554, 0.0158451, 0.0217136, 0.00293427,
0.0264085, 0.0176056, 0.0134977, 0.0064554, 0.0199531, 0.0305164, 0.0117371,
0.0252347, 0.0293427, 0.0199531, 0.0170188, 0.0246479, 0.0381455, 0.00469484,
0.0223005, 0.0105634, 0.0164319, 0.0234742, 0.0129108, 0.00704225, 0.0164319,
0.0134977, 0.0205399, 0.0357981, 0.0129108, 0.0129108, 0.0223005, 0.00821596,
0.0064554, 0.0616197, 0.0134977, 0.00880282, 0.0328638, 0.0117371, 0.00938967}
```

```
In[244]:= ListLogPlot[Transpose[Sort[Transpose[{nodeTotals, nodeTotalsRan}], #1[[1]] < #2[[1]] &]],
  {Filling -> Axis, PlotRange -> All,
  GridLines -> {None, Automatic}, PlotMarkers -> {Automatic, Medium}}]
```



- **How does the subgraph compare to the full graph in predicting cost under different allocations?**

The following defines a new random allocation

```
In[245]:= newPoints = RandomReal[{0, 1}, {nodes, 2}]
```

```
Out[245]= {{0.754019, 0.0803345}, {0.870407, 0.929021}, {0.630178, 0.515911}, {0.774674, 0.240616},
{0.187317, 0.686154}, {0.735555, 0.665499}, {0.726101, 0.217648}, {0.845144, 0.930699},
{0.618852, 0.324274}, {0.094238, 0.843284}, {0.03447, 0.593122}, {0.916987, 0.652832},
{0.940754, 0.92041}, {0.199617, 0.438534}, {0.340695, 0.922838}, {0.281152, 0.21914},
{0.779687, 0.055578}, {0.392363, 0.0135526}, {0.461895, 0.331411}, {0.326107, 0.0888008},
{0.262427, 0.666727}, {0.52416, 0.173364}, {0.966029, 0.0694266}, {0.722873, 0.0323551},
{0.846787, 0.58781}, {0.568086, 0.363711}, {0.922727, 0.213063}, {0.571463, 0.504812},
{0.438106, 0.342155}, {0.408929, 0.937845}, {0.225431, 0.0426959}, {0.637224, 0.0617268},
{0.0929888, 0.312252}, {0.00939137, 0.540705}, {0.877204, 0.691697}, {0.221922, 0.341925},
{0.963696, 0.458087}, {0.737903, 0.563091}, {0.93831, 0.398875}, {0.409406, 0.970852},
{0.0205243, 0.713494}, {0.906377, 0.773337}, {0.204697, 0.324977}, {0.25182, 0.0869001},
{0.166373, 0.723517}, {0.349964, 0.72697}, {0.364403, 0.59381}, {0.272953, 0.181504},
{0.639195, 0.00342352}, {0.00965012, 0.49209}, {0.704417, 0.553032}, {0.628612, 0.125726},
{0.242015, 0.11517}, {0.112986, 0.999646}, {0.0501466, 0.0989114}, {0.665913, 0.144127},
{0.634521, 0.707912}, {0.58899, 0.0363039}, {0.205111, 0.7225}, {0.937526, 0.651046},
{0.371428, 0.854334}, {0.336674, 0.284506}, {0.147829, 0.99388}, {0.243695, 0.862492},
{0.711687, 0.715209}, {0.176478, 0.189498}, {0.481107, 0.811641}, {0.546774, 0.771804},
{0.338725, 0.430993}, {0.191376, 0.231963}, {0.635145, 0.420761}, {0.495906, 0.440561},
{0.84536, 0.319459}, {0.448004, 0.843871}, {0.353294, 0.498437}, {0.58053, 0.303096},
{0.243133, 0.797141}, {0.742002, 0.647105}, {0.671797, 0.820473}, {0.913016, 0.651277},
{0.937945, 0.440625}, {0.467778, 0.34404}, {0.722393, 0.0391241}, {0.256547, 0.620734},
{0.934945, 0.892537}, {0.999054, 0.663126}, {0.989802, 0.351013}, {0.0769366, 0.591097},
{0.926812, 0.681829}, {0.827659, 0.00228332}, {0.1444, 0.126957}, {0.163463, 0.527736},
{0.129693, 0.296886}, {0.433839, 0.656868}, {0.269196, 0.796457}, {0.0340652, 0.374623},
{0.435358, 0.658092}, {0.184599, 0.892019}, {0.198449, 0.882066}, {0.786615, 0.522325}}
```

Since this is a new random allocation, we would expect the cost to increase. First lets see what the original graph predicts as the cost.

```
In[246]:= initialCost
```

```
Out[246]= 0.108957
```

```
In[247]:= newcost = cost[newPoints, iedges]
```

```
Out[247]= 0.53325
```

Now, let's see what the subgraph predicts.

```
In[248]:= newCostRan = cost[newPoints, ranedges]
```

```
Out[248]= 0.53342
```

■ One node move at a time

Moving every node was a fairly drastic change. How is the cost predicted if we move one node at a time?

First a more efficient definition of the cost delta for each move. We begin by creating a map of the adjacent nodes with the associated interaction frequency for every node. We'll do this for the full graph and the subgraph.

```
In[249]:= nodeEdges = Table[Map[edgeToNode[x, #] & , Select[iedges, edgeHasNode[x, #] &]], {x, nodes}]
```

A very large output was generated. Here is a sample of it:

```
Out[249]= {{ {2, 0.0000324828}, {3, 1.91662 × 10-6}, {4, 0.000919564},
           {5, 5.28682 × 10-6}, {6, 2.75158 × 10-6}, {7, 2.56574 × 10-6}, {8, 1.47111 × 10-6},
           <<86>>, {95, 2.00355 × 10-6}, {96, 3.03371 × 10-6}, {97, 1.25363 × 10-6},
           {98, 1.72066 × 10-6}, {99, 3.51051 × 10-6}, {100, 2.38221 × 10-6}, <<99>> }
```

Show Less Show More Show Full Output Set Size Limit...

```
In[250]:= nodeEdgesRan =
Table[Map[edgeToNode[x, #] & , Select[ranedges, edgeHasNode[x, #] &]], {x, nodes}]
```

A very large output was generated. Here is a sample of it:

```
Out[250]= {{ {4, 0.00234742}, {56, 0.000586854}, {79, 0.000586854}, {34, 0.000586854}},
           <<98>>, { {7, 0.000586854}, {84, 0.00528169}, {18, 0.000586854},
           {88, 0.00117371}, {9, 0.000586854}, {6, 0.000586854}, {34, 0.000586854} }
```

Show Less Show More Show Full Output Set Size Limit...

Then we can define functions for the change in cost based on the change in edge distances as we move each node.

```
In[251]:= deltadistance[points_, newpoints_, step_, node_] :=
EuclideanDistance[newpoints[[step]], If[node < step, newpoints[[node]], points[[node]]] -
EuclideanDistance[points[[step]], If[node < step, newpoints[[node]], points[[node]]]
```

```
In[252]:= deltacost[points_, newpoints_, nodeEdges_, step_] :=
Sum[nodeEdge[[2]] deltadistance[points, newpoints, step, nodeEdge[[1]]],
{nodeEdge, nodeEdges[[step]]}]
```

So here is the cost at each move using the full graph and using the subgraph.

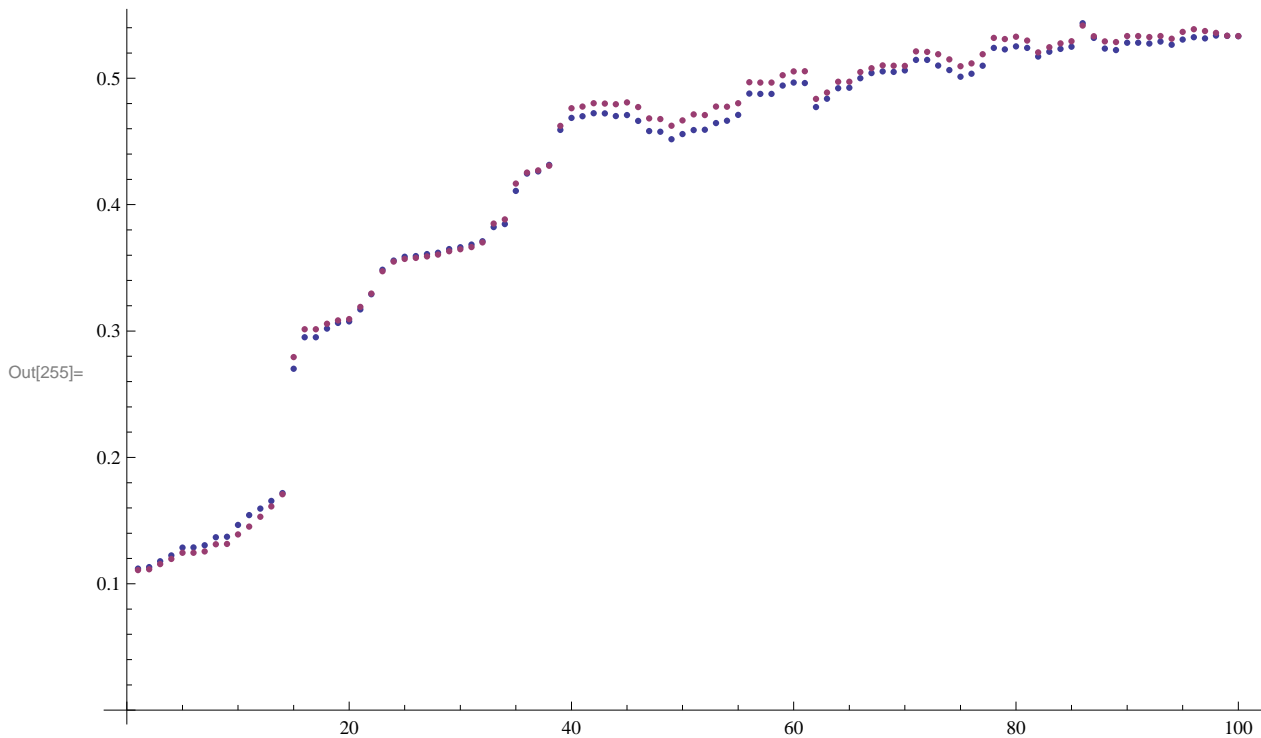
```
In[253]:= costMoves = Map[# + initialCost &,
Accumulate[Table[deltacost[points, newPoints, nodeEdges, x], {x, nodes}]]]
```

```
Out[253]= {0.112004, 0.113162, 0.117773, 0.122444, 0.128648, 0.128706, 0.130533, 0.136808, 0.137192,
0.146623, 0.1544, 0.159505, 0.165518, 0.171776, 0.270149, 0.295132, 0.29508, 0.301978,
0.306448, 0.307572, 0.317153, 0.329063, 0.348515, 0.355752, 0.358713, 0.35936, 0.360908,
0.361936, 0.36483, 0.366248, 0.368374, 0.371057, 0.382285, 0.384669, 0.410814,
0.424521, 0.426247, 0.431438, 0.459191, 0.468654, 0.469901, 0.472318, 0.472168,
0.470087, 0.470888, 0.466245, 0.458264, 0.457741, 0.451712, 0.455806, 0.458991,
0.459267, 0.464486, 0.46641, 0.470978, 0.487925, 0.487682, 0.487595, 0.494246,
0.49658, 0.496167, 0.477202, 0.483806, 0.492118, 0.49257, 0.499978, 0.50408, 0.505481,
0.505037, 0.506191, 0.514571, 0.514534, 0.510063, 0.506476, 0.501267, 0.503581,
0.509956, 0.524043, 0.522873, 0.525191, 0.523973, 0.517181, 0.520994, 0.523257,
0.524948, 0.543527, 0.531965, 0.523443, 0.522307, 0.528165, 0.528129, 0.527403,
0.528978, 0.526586, 0.530645, 0.532487, 0.531516, 0.533727, 0.533465, 0.53325}
```

```
In[254]:= costMovesRan = Map[initialCostRan + # &,
  Accumulate[Table[deltacost[points, newPoints, nodeEdgesRan, x], {x, nodes}]]]
Out[254]= {0.110791, 0.111422, 0.115631, 0.119725, 0.124576, 0.124557, 0.125641, 0.131209, 0.131567,
  0.139105, 0.145273, 0.152991, 0.161169, 0.170802, 0.279385, 0.301401, 0.301355,
  0.30576, 0.308386, 0.309435, 0.319114, 0.329642, 0.347266, 0.35489, 0.357091,
  0.357793, 0.358983, 0.360536, 0.36308, 0.364755, 0.366433, 0.370133, 0.385005,
  0.388331, 0.416711, 0.425438, 0.427029, 0.430753, 0.46236, 0.476255, 0.47762,
  0.480228, 0.480023, 0.479468, 0.480919, 0.47717, 0.468263, 0.467757, 0.462371,
  0.466622, 0.471401, 0.470899, 0.47765, 0.477433, 0.480176, 0.496847, 0.496542,
  0.496577, 0.502444, 0.505488, 0.505598, 0.483674, 0.488638, 0.497254, 0.497249,
  0.504897, 0.507976, 0.510226, 0.509862, 0.509802, 0.521217, 0.520803, 0.519064,
  0.51499, 0.509538, 0.51177, 0.518954, 0.531995, 0.530961, 0.532887, 0.5298, 0.520421,
  0.524576, 0.52754, 0.529342, 0.541734, 0.533306, 0.529201, 0.528682, 0.53343, 0.533408,
  0.53254, 0.533407, 0.531343, 0.536705, 0.538801, 0.537326, 0.535847, 0.53361, 0.53342}
```

Here is a plot of both predicted costs, showing good agreement. Note that the discontinuities coincide with moving the nodes associated with high interaction frequencies.

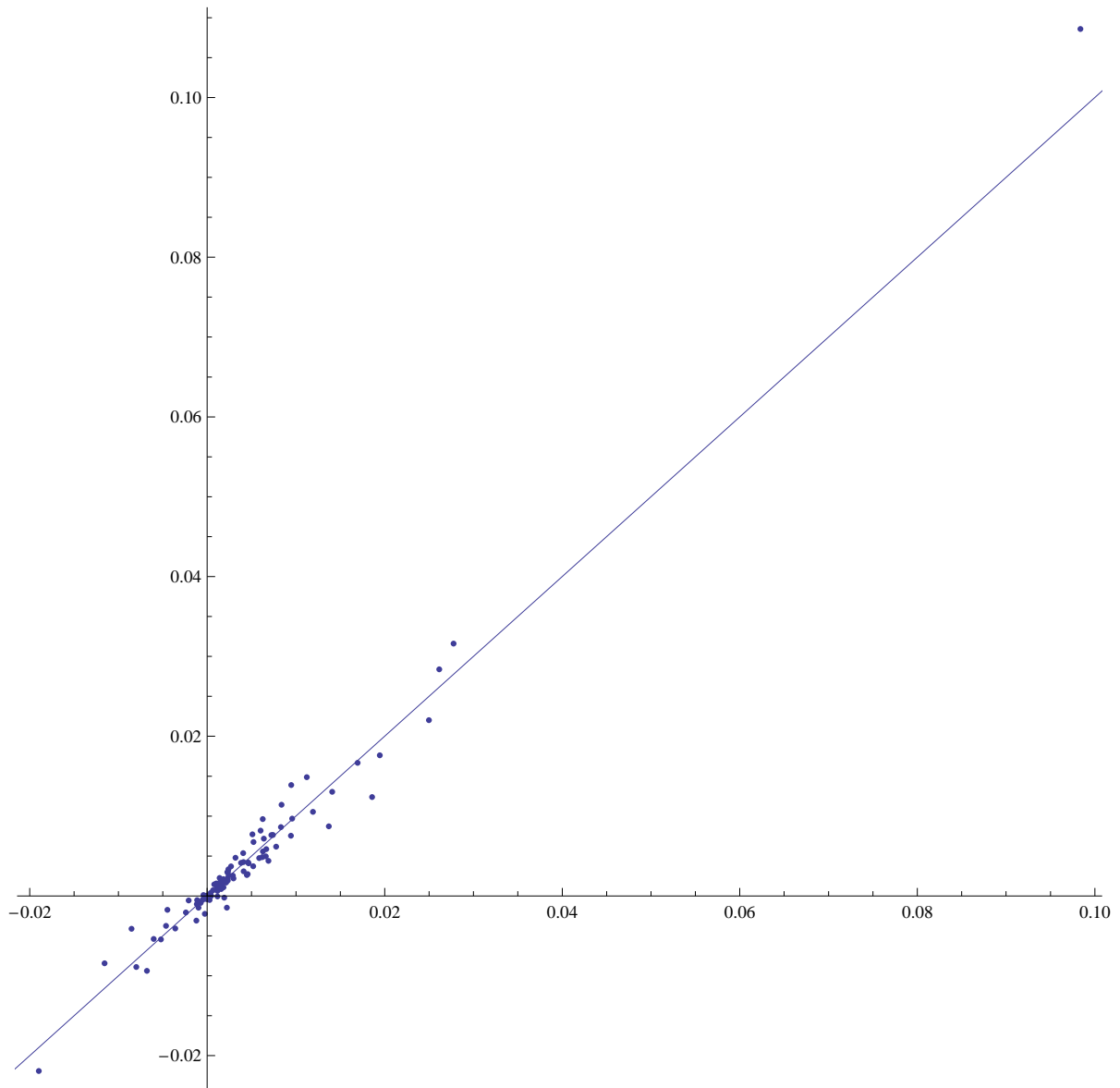
```
In[255]:= ListPlot[{costMoves, costMovesRan}, {PlotRange -> Full, AxesOrigin -> {0, 0}}]
```



Since we'll make decision based on differences, it may be useful to compare the delta predictions. Note that any points the upper left or lower right quadrant would indicate a bad decision.

```
In[256]:= Show[ListPlot[Transpose[{Differences[costMoves], Differences[costMovesRan]}],  
  {PlotRange -> Full, AxesOrigin -> {0, 0}, AspectRatio -> 1}], Plot[x, {x, -1, 1}]]
```

Out[256]=



■ Series of small moves applied to every node

Another approach to looking at changing conditions would be to move every node incrementally in a series of small steps.

```
In[257]:= moves = Transpose[Partition[
  Map[# + Accumulate[RandomReal[{-0.04, 0.04}], 40]] &, Flatten[points]], 2], {2, 3, 1}]
```

A very large output was generated. Here is a sample of it:

```
Out[257]= {{{{0.0553888, 0.929847}, {0.721209, 0.547383}, {0.764013, 0.288127}, {0.192771, 0.862647},
  <<92>>, {0.93996, 0.132395}, {0.610483, 0.038184}, {0.752904, 0.746831},
  {0.822502, 0.5273}}, {{<1>>}, <<36>>, {{<1>>, <<98>>, {{<1>>}}, {{<1>>}}
```

Show Less

Show More

Show Full Output

Set Size Limit...

```
In[258]:= costGradual = Table[cost[move, iedges], {move, moves}]
```

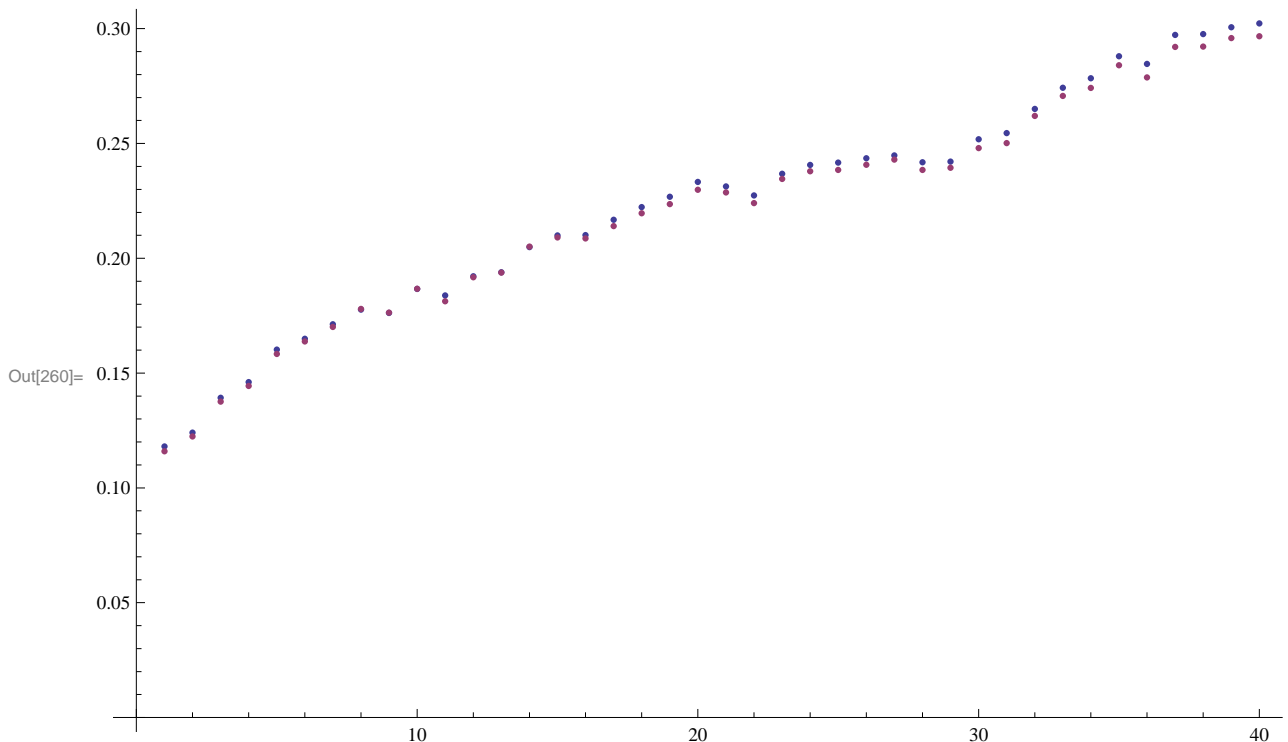
```
Out[258]= {0.118055, 0.124041, 0.139252, 0.146051, 0.160201, 0.164958, 0.171231, 0.177641,
  0.17621, 0.186622, 0.183807, 0.192158, 0.19392, 0.204836, 0.209916, 0.210102,
  0.216744, 0.222257, 0.226813, 0.233275, 0.231288, 0.227399, 0.236784, 0.24064,
  0.241638, 0.243561, 0.244795, 0.241787, 0.242093, 0.251864, 0.254542, 0.265048,
  0.274256, 0.278377, 0.287982, 0.284651, 0.297245, 0.29764, 0.300603, 0.30228}
```

```
In[259]:= costGradualRan = Table[cost[move, ranedges], {move, moves}]
```

```
Out[259]= {0.115942, 0.122423, 0.137551, 0.144437, 0.158348, 0.163786, 0.170089, 0.177965,
  0.176336, 0.186745, 0.18126, 0.191693, 0.193729, 0.205084, 0.209098, 0.208672,
  0.214002, 0.219567, 0.223615, 0.229829, 0.2287, 0.224034, 0.234524, 0.237918,
  0.238473, 0.240757, 0.243003, 0.238499, 0.239437, 0.248007, 0.25014, 0.262018,
  0.270677, 0.274174, 0.284079, 0.27878, 0.292048, 0.292163, 0.295914, 0.29672}
```

Again, very good agreement can be seen in the plot that follows.

```
In[260]:= ListPlot[{costGradual, costGradualRan}, {PlotRange -> Full, AxesOrigin -> {0, 0}}]
```



How does the random graph perform under Greedy Optimization?

We'll define the cost associated with a node (the cost of all the edges connected to that node).

```
In[261]:= costForNode[points_, nodeEdges_, i_] :=
  Sum[nedge[[2]] distance[points, {i, nedge[[1]]}], {nedge, nodeEdges[[i]]}]
```

Since every edge is counted twice, the total across all nodes should be twice the total cost.

```
In[262]:= Sum[costForNode[points, nodeEdgesRan, x], {x, nodes}]
```

```
Out[262]= 0.215169
```

```
In[263]:= initialCostRan * 2
```

```
Out[263]= 0.215169
```

■ Spatial subdivision

This is a simplified implementation of the spatial subdivision technique that produces our search list

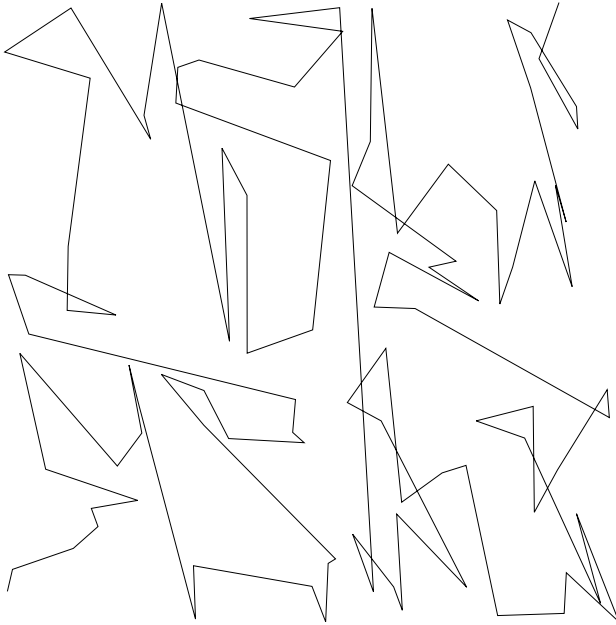
```
In[264]:= subdivision[points_] := Module[{splitOnAxis, nodes},
  splitOnAxis[indices_, xAxis_, addr_, bit_] := Module[{i, s},
    If[Length[indices] == 1, {{First[indices], addr}},
      If[xAxis, s = Sort[indices, points[[#1, 1]] < points[[#2, 1]] &],
        s = Sort[indices, points[[#1, 2]] < points[[#2, 2]] &];
      i = Floor[Length[s] / 2];
      Join[splitOnAxis[s[[ ; i]], !xAxis, addr, bit / 2],
        splitOnAxis[s[[i + 1 ;]], !xAxis, addr + bit, bit / 2]]];
  nodes = Length[points];
  splitOnAxis[Range[nodes], True, 0, 2^Ceiling[Log[2, nodes]]]
```

```
In[265]:= subdiv = subdivision[points]
```

```
Out[265]= {{87, 0}, {86, 4}, {91, 6}, {85, 8}, {10, 12}, {31, 14}, {51, 16}, {71, 20}, {43, 22},
  {63, 24}, {67, 28}, {5, 30}, {48, 32}, {13, 36}, {74, 38}, {14, 40}, {15, 44}, {62, 46},
  {77, 48}, {29, 52}, {90, 54}, {76, 56}, {96, 58}, {53, 60}, {70, 62}, {89, 64}, {47, 68},
  {39, 70}, {68, 72}, {92, 76}, {38, 78}, {4, 80}, {1, 84}, {79, 86}, {23, 88}, {50, 92},
  {94, 94}, {72, 96}, {34, 100}, {65, 102}, {28, 104}, {93, 108}, {11, 110}, {30, 112},
  {56, 116}, {73, 118}, {69, 120}, {22, 122}, {81, 124}, {49, 126}, {98, 128}, {40, 132},
  {78, 134}, {58, 136}, {95, 140}, {8, 142}, {12, 144}, {83, 148}, {26, 150}, {35, 152},
  {66, 156}, {3, 158}, {17, 160}, {52, 164}, {97, 166}, {33, 168}, {42, 172}, {54, 174},
  {27, 176}, {64, 180}, {32, 182}, {37, 184}, {20, 186}, {75, 188}, {21, 190}, {80, 192},
  {59, 196}, {60, 198}, {7, 200}, {2, 204}, {18, 206}, {25, 208}, {45, 212}, {41, 214},
  {55, 216}, {99, 220}, {6, 222}, {100, 224}, {84, 228}, {36, 230}, {9, 232}, {88, 236},
  {57, 238}, {44, 240}, {46, 244}, {16, 246}, {61, 248}, {19, 250}, {24, 252}, {82, 254}}
```

```
In[266]:= Graphics[Line[Map[points[[#]] &, subdiv[[All, 1]]]]]
```

```
Out[266]=
```



To traverse the list from a starting node, we look up and down the list, always preferring the node who's spatial address shows it to be "closer" to the starting node.

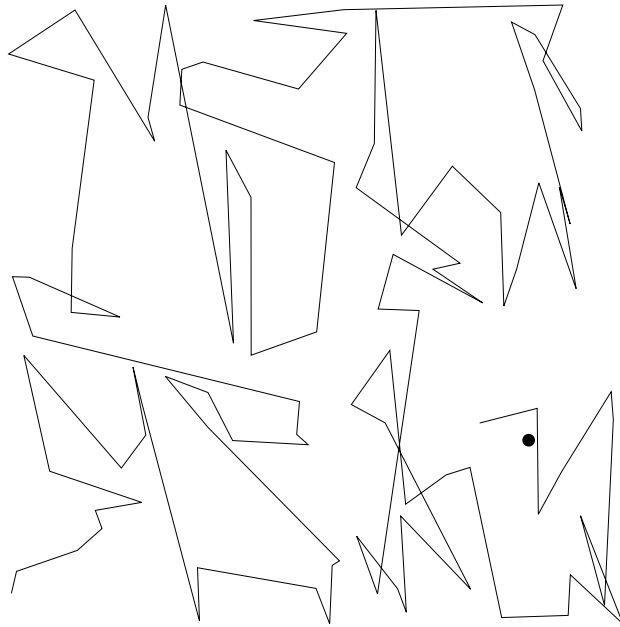
```
In[267]:= nextIndex[subdivision_, s_, u_, d_] := Module[{a}, a = subdivision[[s, 2]];
  If[d < 1,
    If[u > Length[subdivision], Null, {subdivision[[u, 1]], u + 1, d}],
    If[u > Length[subdivision], {subdivision[[d, 1]], u, d - 1},
      If[BitXor[subdivision[[u, 2]], a] > BitXor[subdivision[[d, 2]], a],
        {subdivision[[d, 1]], u, d - 1}, {subdivision[[u, 1]], u + 1, d}]]]
```

```
In[268]:= SubDivSequence[subdivision_, pointindex_] := Module[{s, u, d, l},
  s = First[First[Position[subdivision, {pointindex, _}]]]; u = s + 1; d = s - 1;
  Reap[While[d > 0 || u ≤ Length[subdivision], l = nextIndex[subdivision, s, u, d];
    u = l[[2]]; d = l[[3]]; Sow[l[[1]]]]][[2, 1]]]
```

An example of a traverse sequence. It produces an efficient kind-of-spiral search pattern.

```
In[269]:= Graphics[{Line[Map[points[[#]] &, SubDivSequence[subdiv, 27]]], Disk[points[[27]], 0.01]}]
```

Out[269]=



■ Greedy Optimization Heuristic

This is a simplified version of the greedy heuristic that implements a swapping operation.

```

In[270]:= Optimize[points_, nodeEdges_] := Module[
  {EvaluateSwap, nodes, p, contribs, subdiv, checked = 0, delta, s, u, d, l, count, time},
  EvaluateSwap[i_, j_] := Module[{c1, c2, t},
    c1 = costForNode[p, nodeEdges, i] + costForNode[p, nodeEdges, j];
    t = p[[j]];
    p[[j]] = p[[i]];
    p[[i]] = t;
    c2 = costForNode[p, nodeEdges, i] + costForNode[p, nodeEdges, j];
    If[c2 < c1, c2 - c1,
      p[[i]] = p[[j]];
      p[[j]] = t; 0];
  nodes = Length[points];
  p = points;
  contribs =
    Sort[Table[{i, costForNode[points, nodeEdges, i]}, {i, nodes}], #1[[2]] > #2[[2]] &];
  subdiv = subdivision[points];
  time = 0.;
  While[checked < nodes,
    s = First[First[Position[subdiv, {contribs[[1, 1]], _}]]];
    u = s + 1;
    d = s - 1;
    delta = 0;
    count = 0;
    While[delta == 0 && (d > 0 || u ≤ nodes),
      count++;
      l = nextIndex[subdiv, s, u, d];
      u = l[[2]]; d = l[[3]];
      time += Timing[delta = EvaluateSwap[contribs[[1, 1]], l[[1]] ]][[1]];
      contribs[[1, 2]] = -delta / count;
      contribs = Sort[contribs, #1[[2]] > #2[[2]] &];
      If[delta == 0, checked++, checked = 0; Print[delta]]; Print["Eval Time: ", time]; p]

```

Running the greedy heuristic using the subgraph produces the following node positions. Note that the printed numbers indicate the change in the total cost associated with each swap performed.

```

In[271]:= optPoints = Optimize[points, nodeEdgesRan]

```

-0.0000116012
-0.000104187
-0.0000694113
-0.0000239463
-0.0000459656
-0.0000758662
-0.000167941
-0.000113671
-0.0000548133
-0.0000400234
 -6.5989×10^{-6}
 -3.08808×10^{-7}
-0.0000213348
-0.0000236669
-0.000332135
-0.0000878734
-0.000052835
-0.0000758269

Eval Time: 6.473

Out[271]= {{0.0227528, 0.921143}, {0.74047, 0.588511}, {0.756413, 0.263904}, {0.158695, 0.879409},
{0.234351, 0.367106}, {0.925107, 0.548014}, {0.776511, 0.525406}, {0.757345, 0.0700208},
{0.830615, 0.579211}, {0.160594, 0.195309}, {0.575288, 0.708571}, {0.62142, 0.33447},
{0.323116, 0.104255}, {0.533331, 0.0153083}, {0.537144, 0.108067}, {0.859579, 0.95176},
{0.806581, 0.0251349}, {0.697553, 0.578926}, {0.932056, 0.834487}, {0.91593, 0.0931559},
{0.980868, 0.383993}, {0.555516, 0.991426}, {0.255034, 0.782547}, {0.872511, 0.910297},
{0.540873, 0.748473}, {0.62884, 0.449881}, {0.84967, 0.307338}, {0.408387, 0.442263},
{0.27196, 0.408409}, {0.294967, 0.839944}, {0.234151, 0.208085}, {0.863248, 0.357641},
{0.999655, 0.015023}, {0.368424, 0.768063}, {0.653948, 0.2052}, {0.898557, 0.708878},
{0.901725, 0.256776}, {0.123679, 0.611979}, {0.0557611, 0.56615}, {0.576108, 0.154402},
{0.606924, 0.990442}, {0.931908, 0.187068}, {0.202017, 0.262864}, {0.859249, 0.863681},
{0.604146, 0.778798}, {0.90403, 0.998995}, {0.0289493, 0.566913}, {0.32616, 0.0195857},
{0.560359, 0.953807}, {0.244224, 0.820216}, {0.0880962, 0.257836}, {0.912521, 0.0285247},
{0.48082, 0.316335}, {0.970213, 0.0445751}, {0.647453, 0.633048}, {0.298062, 0.896592},
{0.810176, 0.520445}, {0.655096, 0.0335565}, {0.610232, 0.515612}, {0.634052, 0.602577},
{0.93408, 0.798679}, {0.548868, 0.115068}, {0.240963, 0.315278}, {0.772589, 0.334549},
{0.40799, 0.692979}, {0.718554, 0.25195}, {0.220628, 0.423291}, {0.199902, 0.502708},
{0.483567, 0.865436}, {0.485266, 0.368363}, {0.0468272, 0.441996}, {0.380164, 0.461219},
{0.331862, 0.908292}, {0.511541, 0.071544}, {0.984232, 0.339768}, {0.379072, 0.306559},
{0.339537, 0.32771}, {0.641397, 0.070617}, {0.1283, 0.990855}, {0.675502, 0.51316},
{0.412441, 0.974361}, {0.822259, 0.971978}, {0.567777, 0.363596}, {0.805198, 0.668976},
{0.171311, 0.166433}, {0.0353376, 0.0983618}, {0.0273757, 0.0642769}, {0.865673, 0.716182},
{0.0612559, 0.472583}, {0.33976, 0.38275}, {0.132011, 0.131786}, {0.122053, 0.510312},
{0.512445, 0.479185}, {0.272499, 0.998603}, {0.645964, 0.18648}, {0.499137, 0.29984},
{0.865014, 0.189436}, {0.609232, 0.0627998}, {0.728319, 0.742585}, {0.915568, 0.650645}}

Extracting only the nodes who moved.

```
In[272]:= optMoves = Select[Transpose[{points, optPoints}], #[[1]] ≠ #[[2]] &]
```

```
Out[272]= {{{0.697553, 0.578926}, {0.74047, 0.588511}}, {{0.805198, 0.668976}, {0.925107, 0.548014}},
  {{0.925107, 0.548014}, {0.830615, 0.579211}}, {{0.540873, 0.748473}, {0.575288, 0.708571}},
  {{0.74047, 0.588511}, {0.697553, 0.578926}}, {{0.93408, 0.798679}, {0.932056, 0.834487}},
  {{0.901725, 0.256776}, {0.91593, 0.0931559}}, {{0.984232, 0.339768}, {0.980868, 0.383993}},
  {{0.560359, 0.953807}, {0.555516, 0.991426}}, {{0.575288, 0.708571}, {0.540873, 0.748473}},
  {{0.865673, 0.716182}, {0.898557, 0.708878}}, {{0.865014, 0.189436}, {0.901725, 0.256776}},
  {{0.822259, 0.971978}, {0.90403, 0.998995}}, {{0.555516, 0.991426}, {0.560359, 0.953807}},
  {{0.915568, 0.650645}, {0.810176, 0.520445}}, {{0.932056, 0.834487}, {0.93408, 0.798679}},
  {{0.980868, 0.383993}, {0.984232, 0.339768}}, {{0.90403, 0.998995}, {0.822259, 0.971978}},
  {{0.830615, 0.579211}, {0.805198, 0.668976}}, {{0.898557, 0.708878}, {0.865673, 0.716182}},
  {{0.91593, 0.0931559}, {0.865014, 0.189436}}, {{0.810176, 0.520445}, {0.915568, 0.650645}}}}
```

Comparing the initial cost and the optimized cost shows a small improvement.

```
In[273]:= initialCostRan
```

```
Out[273]= 0.107584
```

```
In[274]:= cost[optPoints, ranedges]
```

```
Out[274]= 0.106276
```

In this case the cost estimated by the full graph is actually slightly worse. Is it possible that the optimization is taking advantage of peculiarities in this particular random subgraph?

```
In[275]:= initialCost
```

```
Out[275]= 0.108957
```

```
In[276]:= cost[optPoints, iedges]
```

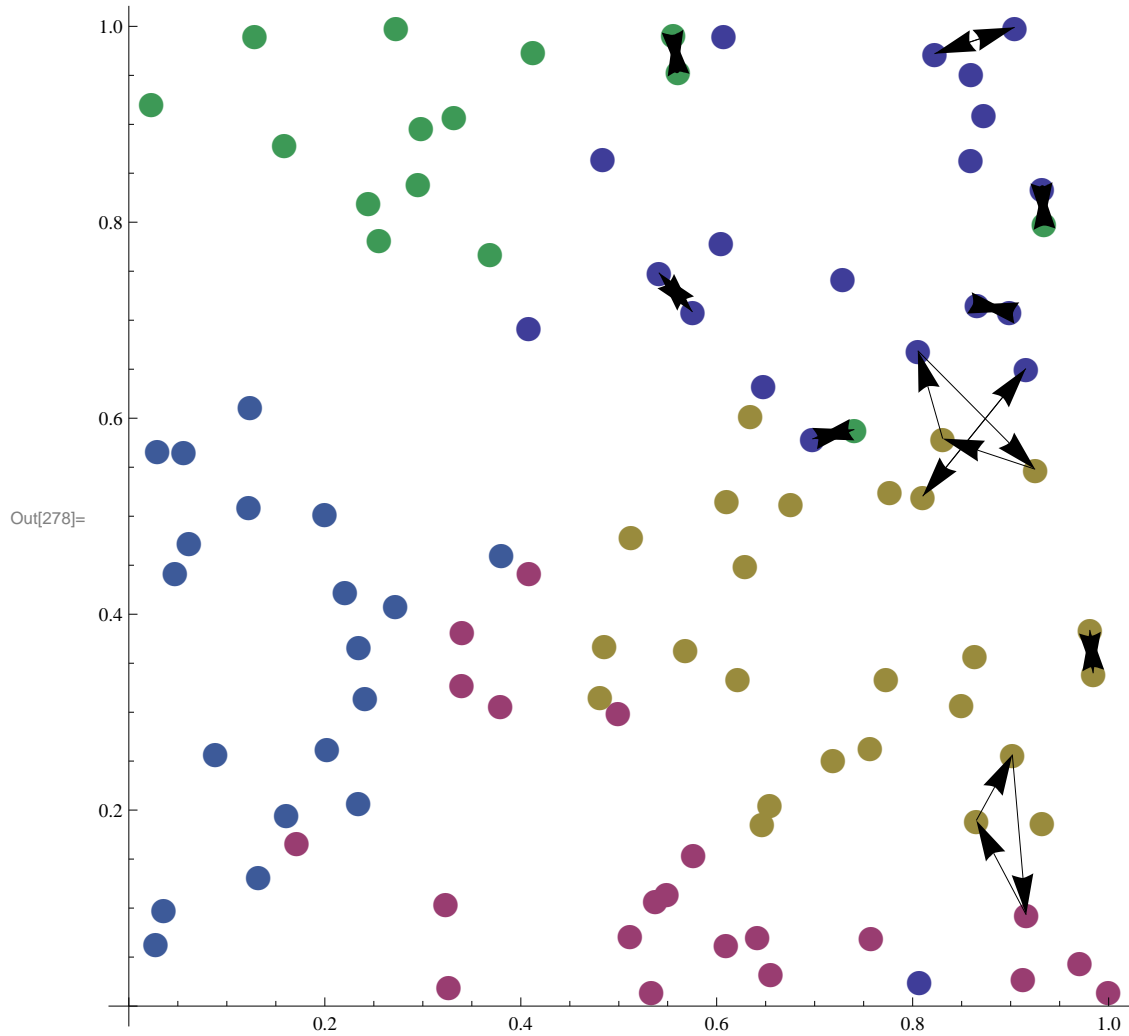
```
Out[276]= 0.109257
```

Pushing on, here is a plot of the actual swaps that were made.

```
In[277]:= optPointsByOrg = Table[Pick[optPoints, Map[If[o == #, True, False] &, nodeOrgs]], {o, orgs}]
```

```
Out[277]= {{0.575288, 0.708571}, {0.859579, 0.95176}, {0.806581, 0.0251349}, {0.697553, 0.578926},
{0.932056, 0.834487}, {0.872511, 0.910297}, {0.540873, 0.748473}, {0.898557, 0.708878},
{0.606924, 0.990442}, {0.859249, 0.863681}, {0.604146, 0.778798}, {0.90403, 0.998995},
{0.647453, 0.633048}, {0.40799, 0.692979}, {0.483567, 0.865436}, {0.822259, 0.971978},
{0.805198, 0.668976}, {0.865673, 0.716182}, {0.728319, 0.742585}, {0.915568, 0.650645}},
{{0.757345, 0.0700208}, {0.323116, 0.104255}, {0.533331, 0.0153083},
{0.537144, 0.108067}, {0.91593, 0.0931559}, {0.408387, 0.442263}, {0.999655, 0.015023},
{0.576108, 0.154402}, {0.32616, 0.0195857}, {0.912521, 0.0285247},
{0.970213, 0.0445751}, {0.655096, 0.0335565}, {0.548868, 0.115068},
{0.511541, 0.071544}, {0.379072, 0.306559}, {0.339537, 0.32771}, {0.641397, 0.070617},
{0.171311, 0.166433}, {0.33976, 0.38275}, {0.499137, 0.29984}, {0.609232, 0.0627998}},
{{0.756413, 0.263904}, {0.925107, 0.548014}, {0.776511, 0.525406},
{0.830615, 0.579211}, {0.62142, 0.33447}, {0.980868, 0.383993}, {0.62884, 0.449881},
{0.84967, 0.307338}, {0.863248, 0.357641}, {0.653948, 0.2052}, {0.901725, 0.256776},
{0.931908, 0.187068}, {0.48082, 0.316335}, {0.810176, 0.520445},
{0.610232, 0.515612}, {0.634052, 0.602577}, {0.772589, 0.334549}, {0.718554, 0.25195},
{0.485266, 0.368363}, {0.984232, 0.339768}, {0.675502, 0.51316}, {0.567777, 0.363596},
{0.512445, 0.479185}, {0.645964, 0.18648}, {0.865014, 0.189436}},
{{0.0227528, 0.921143}, {0.74047, 0.588511}, {0.158695, 0.879409}, {0.555516, 0.991426},
{0.255034, 0.782547}, {0.294967, 0.839944}, {0.368424, 0.768063},
{0.560359, 0.953807}, {0.244224, 0.820216}, {0.298062, 0.896592}, {0.93408, 0.798679},
{0.331862, 0.908292}, {0.1283, 0.990855}, {0.412441, 0.974361}, {0.272499, 0.998603}},
{{0.234351, 0.367106}, {0.160594, 0.195309}, {0.27196, 0.408409}, {0.234151, 0.208085},
{0.123679, 0.611979}, {0.0557611, 0.56615}, {0.202017, 0.262864}, {0.0289493, 0.566913},
{0.0880962, 0.257836}, {0.240963, 0.315278}, {0.220628, 0.423291}, {0.199902, 0.502708},
{0.0468272, 0.441996}, {0.380164, 0.461219}, {0.0353376, 0.0983618},
{0.0273757, 0.0642769}, {0.0612559, 0.472583}, {0.132011, 0.131786}, {0.122053, 0.510312}}}
```

```
In[278]:= Show[ListPlot[optPointsByOrg, PlotMarkers -> Magnify[●, 2]], Graphics[Map[Arrow, optMoves]],  
{AspectRatio -> Automatic, AxesOrigin -> {0, 0}, PlotRange -> {{0, 1}, {0, 1}}}]
```



```
In[279]:= ShowWeightedSubGraph[optPoints, ranedges]
```



Out[279]=

