The Network Structure of the Uniform Commerical Code: It's A Small World After All

' Seth J. Chandler 2005

The Uniform Commercial Code is the body of law governing many forms of transactions in the United States, including sales of goods, leases, most forms of payments, and so-called "secured transactions." Consisting of eleven "articles," which in turn contain a nested structure of sections and subsections, the UCC is a cross-referenced body of law developed in committees by two private organizations and then generally adopted with few modifications by the fifty American states. It is studied extensively in American law schools and constitutes the primary legal authority used by many commercial lawyers in the United States. This article uses *Mathematica* to study the formal and textual network structure of the UCC. It first uses the Regular Expression constructs of *Mathematica* to parse the UCC and reveal the structure of nodes and connections it implicitly contains. It then develops tools in *Mathematica* to analyze and visualize this network. Standard embedding algorithms, coupled with numeric analysis , highlight that the UCC is a classic "Small World Network" in which highly clustered portions of the law have weak ties to distant clusters in a way that reduces the diameter of the network. The article is also able to show the most important and intricate provisions of the UCC as well as to reveal useful methods of studying this significant body of law.

Creating the Textual, Formal and Full Networks

Introduction

What I propose to do in this work is to determine what features of the law are lost, revealed, or clarified when its textual representation is stripped of its treasured linguistic content and all that is preserved are the connections among different chunks of the text. This radical compression cannot be done without human judgment. Because legal texts and not explicitly designed as mathematical objects, decisions must be made about what counts as a "node" and what counts as an "edge." Such an effort must be exogenously informed by expertise in the law; the ghost of the text thus persists in the formation of the network.

The Cornell-LII Text

The solution chosen here draws on the expertise of Cornell University's Legal Information Institute ("Cornell-LII"), which has created and which maintains an electronic version (HTML format) of the entire Uniform Commercial Code (http://www.law.cornell.edu/ucc/index.htm). The text is stored in 11 files, one for each article of the code, that collectively consume about 1.5 megabytes of text. This version is particularly useful in that Cornell-LII has carefully inserted HTML anchors for each section and subsection of the each of the articles as well as for each obviously defined term (such as "goods" or "instrument"). The Cornell-LII has likewise carefully inserted HTML hyperlinks each time the text cross references either another section of the UCC or one of the defined terms. The By way of example, here is how the Cornell-LII text denotes the connection between various provisions of law cited in section 2-207(2) of the UCC. I have highlighted the hyperlinks. What I would like is to extract links, therefore, from "s2-2072" to "contract" and from "s2-2072" to "BetweenMerchants". The Cornell-LII version is thus well suited as the foundation for a mathematical network in which hyperlink anchors form the nodes of the network and the hyperlinks themselves form the links. I will refer to this network as the "textual UCC network" because it imposes no structure on the text other than that perceived by Cornell-LII, one which seems quite logical to me along with other professors of contract law. It is important to note that terms that are used often, such as "person" or "reasonable" that are not themselves defined in the UCC are not considered nodes (even if they are used quite frequently) and thus have no links coming to them or emerging from them.

```
<a name="s2-2072"></a>(2) The additional terms are to
      be construed as proposals for addition to the <a href="#contract">contract</a>.  <a href="#BetweenMerchants">Between
      merchants</a> such terms become part of the contract unless:</p>
<p class="Text-Level2">(a) the offer expressly limits acceptance to the terms
      of the offer;</p>
<p class="Text-Level2">(b) they materially alter it;  or</p>
<p class="Text-Level2">(c) notification of objection to them has already been
      given or is given within a reasonable time after notice of them is received.</p>
```

There is another latent network I believe within the Uniform Commercial Code, however, that Cornell-LII has not captured, perhaps because to conventional legal scholars it is so obviously present. The UCC, like many modern legal codes, is formally structured as a tree. Articles contain chapters that contain sections that contain subsections. And while Cornell-LII has created hyperlinks noting the connectivity between items of text and these various sections or subsections, no hyperlinks are inserted to denote the internal tree structure. To illustrate, subsection (b) of section 2-207 of the UCC, which has its own anchor, does not contain a hyperlink to section 2-207 of the UCC, even though the latter full section has an anchor and even though subsection (b) is obviously part of section 2-207. The internal structure of the Cornell-LII version is essentially "flat," with no attempt being made to use XML or other methods to replicate the nested structure of the original legal code. (XML structuring has been used on other some of the other texts maintained by Cornell-LII). One of my tasks here, therefore, will be to breathe life into the latent formal tree structure and to integrate the resulting "formal network" with the textual UCC network.

Importation of the Cornell-LII Text from the Web

Mathematica is the technology I use to accomplish these tasks. It can be used swiftly to import and then combine the HTML files into one large textual file, which can be locally stored for simplicity. The **StringReplace** command is used to address some typos and mild irregularities in the Cornell-LII markup of the text. Those who are more interested in the results of this analysis than the methodology may safetly skip this and the next subsection.

I begin by loading two packages that will be used extensively in this notebook.

```
Needs["DiscreteMath‘GraphPlot‘"]; Needs["DiscreteMath‘Combinatorica‘"];
```

```
Export["H:\\CDR\\LAW&ECON\\ARTICLE\\Graph
    Theoretic Structure of Common Law\\UCC\\ucctext.txt",
  ucctext = With[{fixes = {RegularExpression["(?<!s)(\\d(?:A|a)-\\d.+)"] :>
        "s" <> "$1", RegularExpression["^UCC(\\d.*)"] :> "s" <> "$1",
       RegularExpression["(?<!s)2(?:a|A)-(\\d\\w+)"] :> "s2A-" <> "$1",
       RegularExpression["s2(?:a|A)-(\\d\\w+)"] :> "s2A-" <> "$1",
       RegularExpression["(?<!s)4(?:a|A)-(\\d\\w+)"] :> "s4A-" <> "$1",
       RegularExpression["s4(?:a|A)-(\\d\\w+)"] :> "s4A-" <> "$1"}},
    Fold[StringReplace[#1, #2] &, StringJoin @@
      Map[Import["http://www.law.cornell.edu/ucc/" <>
          # <> "/article" <> # <> ".htm", "Text"] &,
        {"1", "2", "2A", "3", "4", "4A", "5", "6", "7", "8", "9"}], fixes]], "Text"];
```

Once this is done,we can, of course, simply import the local version of the file.

```
getucc[location_String:
    "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph Theoretic Structure of Common
      Law\\UCC\\ucctext.txt"] := Import[location, "Text"];

ucctext = getucc[];
```

The Textual Structure of the UCC

I now disinter two networks from the text of the UCC: the "textual network" and the "formal network." Ultimately, I combine the two networks into the "Full UCC Networks" by taking the union of the nodes and the union of the edges.

Finding the Textual Structure

To recover the textual structure of the UCC, I start by splitting the entire text into pairs of anchors and the text that follows up until the next anchor. I insert a "dummy anchor"at the very beginning of the text. Regular Expressions coupled with the **StringCases** and **StringSplit** functions perform this task swiftly.

```
anchorNameRegExp =
  RegularExpression["<(?:a|A)\\s+[^>]*name\\s*=\\s*\"([^\">]*)\"[^>]*>"];

AbsoluteTiming[anchortextpairs =
   Thread[List[Prepend[StringCases[ucctext, anchorNameRegExp :> "$1"], "ZeroStart"],
     StringSplit[ucctext, anchorNameRegExp]]];]
```

```
{0.1718904 Second, Null}
```

I now need to take the textual component of these anchor-text pairs, extract the hyperlinks contained within the text, and then thread the anchors over the hyperlinks to show that the anchored segment of text contains a reference to the anchor contained in the hyperlink. This process produces a list of edges in the textual UCC network. Because some of the hyperlinks in the Cornell-LII database reference anchors within files that are stored separately on the Cornell web site, and because some of the hyperlinks reference things that simply are not relevant to this project, each of the links must be properly resolved.

```
resolvetarget[s_String] := With[{
hashString = "#([^%]+)", cornellString = "http://www.law.cornell.edu/ucc(.+)"},
  Which[
   StringMatchQ[s, RegularExpression[hashString]],
   First[StringCases[s, RegularExpression[hashString] :> "$1"]],
   StringMatchQ[s, RegularExpression[cornellString <> hashString]],
   First[StringCases[s, RegularExpression[cornellString <> hashString] :> "$2"]],
   True, "UNRESOLVABLE"]]
```

```
Short[textualuccedges = With[{anchorHrefRegExp =
     RegularExpression["<(?:a|A)\\s+[^>]*href\\s*=\\s*\"([^!\">]*)\"[^>]*>"]},
   Flatten[DeleteCases[Map[Thread[
        MapAt[DeleteCases[resolvetarget /@ StringCases[#, anchorHrefRegExp :> "$1"],
           "UNRESOLVABLE"] &, #, 2]] &, anchortextpairs], {}], 1]]]
```

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

```
{{s1-103a, Agreement}, <<7512>>, {s9-709b, dfinancingstatement}}
```

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

- **Preparing to Visualize the Textual Structure**

I can now use the same techniques as above to visualize the textual structure of the UCC. I first get a list of the labels for all nodes and then I extract a list of numbered edges.

```
Short[textualUCClabels = Union[Flatten[textualuccedges]]]
```

```
{2-104, 2-106, 2-401, 2-403, <<2192>>, warehouse, Warehousereceipt, Writing}
```

```
AbsoluteTiming[Short[textualuccgraph = Rule @@@
    (textualuccedges /. Dispatch[MapIndexed[Rule[#1, #2[[1]]] &, textualUCClabels]])]]
```

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

```
{0.0312528 Second, {309 → 52, 310 → 105, 311 → 281, 314 → 145,
  <<7506>>, 2161 → 194, 2163 → 1814, 2163 → 1682, 2163 → 155}}
```

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

I let Combinatorica construct a graph out of the edges (Combinatorica and GraphPlot currently use different network formats).

```
textualuccgraphC = SetVertexLabels[
  FromOrderedPairs[List @@@ textualuccgraph, Type → Directed], textualUCClabels]
```

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

```
-Graph:<7514, 2199, Directed>-
```

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

For visualization purposes, I eliminate from the graph a few very small unconnected components.

```
textualUCCgraphCC = With[{cc = ConnectedComponents[textualuccgraphC]},
  InduceSubgraph[textualuccgraphC, cc[[First@Ordering[Length /@ cc, -1]]]]]
```

```
A very large output was generated. Here is a sample of it:
-Graph:<7500, 2175, Directed>-
ShowLess  ShowMore  ShowFullOutput  SetSizeLimit...
```

I can also create an undirected variant of this graph.

```
textualUCCgraphCCU = MakeUndirected[textualUCCgraphCC]
```

```
A very large output was generated. Here is a sample of it:
-Graph:<6269, 2175, Undirected>-
ShowLess  ShowMore  ShowFullOutput  SetSizeLimit...
```

■ A Picture of the Textual Structure of the Uniform Commercial Code

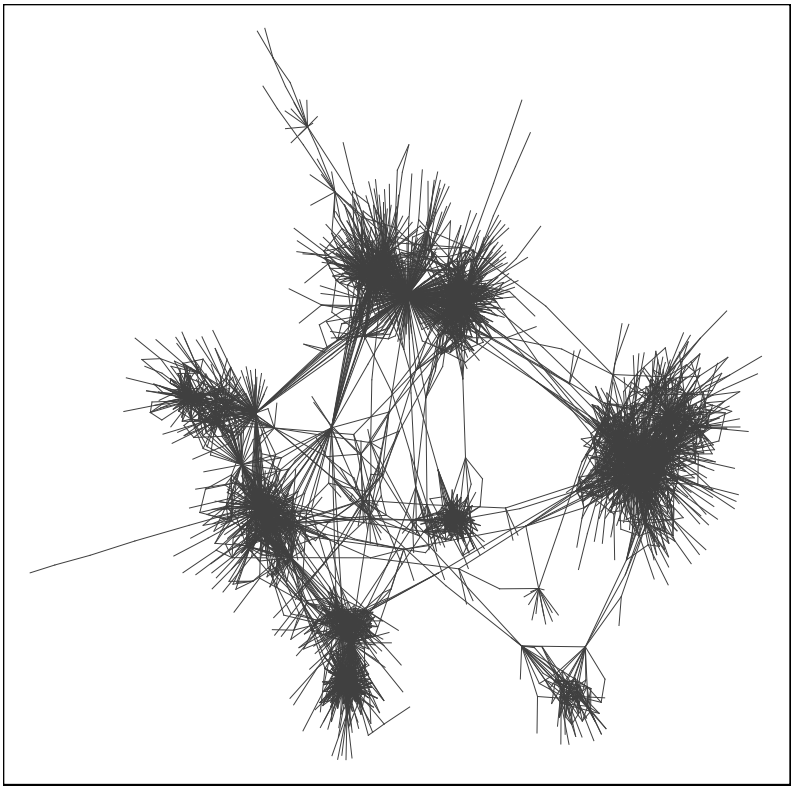
I now use GraphPlot to visualize the textual structure of the UCC. Here, I use *Mathematica*'s implementation of the Fruchterman-Reingold algorithm, termed the "SpringElectricalModel," to embed the nodes. This algorithm assigns the nodes of the network some initial position in space. The space is usually two dimensional but can have a higher dimensionality. All nodes then repel each other but do so in a fashion that is inversely proportional to the Euclidean physical distance between them, much in the way two particles with like electrical charge would repel each other. Thus, two nodes that lie at a distance of two from each other repel each other more strongly than do two nodes that lie at a distance of five from each other. Nodes that are connected to each other, however, attract in a counterbalancing fashion that is proportional to the physical distance between them. The general idea, quite complex in its implementation and subject to numerous variations, is to position the nodes in a fashion that minimizes the total attractive and repulsive force exerted. This method tends to produce a layout that is both attractive and that often both recapitulates and extends intuitions about the structure of the network under consideration. I use the Tooltip construct of *Mathematica* 6 to permit those using this notebook in an interactive way to determine the content of each of the embedded nodes.

```
Short[textualUCCcoords = GraphCoordinates[textualUCCgraphCC,
   Method → {"SpringElectricalModel", "RepulsiveForcePower" → -1.5, RecursionMethod →
      {"Multilevel", CoarseningScheme → "MaximalIndependentVertexSetRugeStuben"}}]]
```

```
{{3.40983, -4.77647}, <<2173>>, {-4.24539, -<<19>>}}
```

```
gptextual =
 If[$VersionNumber ≥ 6., Panel[GraphPlot[textualUCCgraphCC, VertexStyleFunction :>
     ({Hue[0], Tooltip[Disk[#, 0.002], textualUCClabels[[#]]]} &),
    EdgeStyleFunction → ({GrayLevel[0.25], Line[{#1, #2}]} &),
    VertexCoordinates → textualUCCcoords, AspectRatio → 1],
   "The Textual Structure\nof the Uniform Commercial Code"],
  GraphPlot[textualUCCgraphCC, VertexStyleFunction :> ({Hue[0], Disk[#, 0.002]} &),
   EdgeStyleFunction → ({GrayLevel[0.25], Line[{#1, #2}]} &),
   VertexCoordinates → textualUCCcoords, AspectRatio → 1,
   PlotLabel → "The Textual Structure\nof the Uniform Commercial Code"]]
```

The Textual Structure
of the Uniform Commercial Code



Notice that the graph embedding routines contained in Mathematica have constructed a plausible structure for the Uniform Commercial Code simply from the connectivity information contained in the text, without any knowledge as to the tree structure of the Code, and without any understanding of what the Code actually says. I say "plausible" because the picture may be interpreted as clusters of highly interrelated provisions with weak ties to other clusters of highly interrelated provisions, which corresponds with many scholars intuition as to the structure of the UCC.

The Formal Structure of the UCC

In addition to this textual structure, the UCC also has an implicit formal structure created by the hierarchical labelling of articles, chapters, sections and subsections. As noted above, however, the Cornell-LII text has not preserved this nested structure. One way to recover it, however, is to find all the "leaves" of the tree using Regular Expressions and then to find the parent "branches" by combining Regular Expression constructs with independent knowledge of naming conventions adopted by the UCC. Again, those interested primarily in the legal aspects of this article may skip the remainder of this subsection.

Finding the Formal Structre

The following code captures all "leaves" contained in the Uniform Commercial Code.

```
AbsoluteTiming[Short[uccleaves =
   With[{likelyuccreference = RegularExpression["(s\\d(?:A|a)?-\\d{3}\\w*)"]},
    Union[StringCases[ucctext, likelyuccreference :> "$1"]]]]]
```

```
{0.0781320 Second,
 {s1-101a, s1-101b, s1-102, s1-103, <<2422>>, s9-708, s9-709, s9-709a, s9-709b}}
```

I now show how to recover the "geneaology"of these statutory leaves. I know, for example, that the parent of UCC subsection "s2-207b" is "s2-207" because I know that the framers of the UCC denote a section by a chapter number (a one or two character string), followed by a hyphen, followed by three digits. I know that the parent of UCC section "s9-102a80" is "s9-102a" because a shift from numeric characters to letter characters after the three digit section number is used in the Cornell-LII text to mark a change from subsection to subsubsection. The *Mathematica* code set forth below shows how all this may be done. Basically, it tries to match a list of Regular Expressions and then capture that part of the appropriate regular expression that represents the parent.

```
parent[s_] := With[
  {articlestring = "^s\\d(?:A|a)?-", chapterstring = "\\d", sectionstring = "\\d\\d",
   terminalsubsectionstring = "(?:(?<=\\D)\\d+|(?<=\\d)\\D+)$"},
  StringReplace[s, {RegularExpression["(" <> articlestring <> chapterstring <>
       sectionstring <> ".+" <> ")" <> terminalsubsectionstring] :> "$1",
    RegularExpression["(" <> articlestring <> chapterstring <> sectionstring <>
       ")" <> "(\\d+|\\D+)$"] :> "$1", RegularExpression[
      "(" <> articlestring <> chapterstring <> ")" <> sectionstring <> "$"] :> "$1",
    RegularExpression["(" <> articlestring <> ")" <> chapterstring <> "$"] :>
     StringReplace["$1", "-" -> ""]}
  ]]
```

I can do this recursively using *Mathematica*'s **FixedPoint** construct to find the entire geneaology of a UCC leaf.

```
parentlist[s_] := Most[FixedPointList[parent, s, 20]]
```

I can now find all the "edges" of the formal UCC graph.

```
AbsoluteTiming[Short[formaluccedges =
   Union[Flatten[Map[Partition[parentlist[#], 2, 1] &, uccleaves], 1]]]]
```

```
{0.3594072 Second,
 {{s1-1, s1}, {s1-101, s1-1}, <<2481>>, {s9-709a, s9-709}, {s9-709b, s9-709}}}
```

■ Preparing to Visualize the Formal Structure of the Uniform Commercial Code

I can use the same techniques as before to go from a listing of these edges to creation of network objects suitable for manipulation and analysis using various Mathematica packages.

```
AbsoluteTiming[Short[formaluccgraph = Rule @@@(formaluccedges /.
      Dispatch[MapIndexed[Rule[#1, #2[[1]]] &, Union[Flatten[formaluccedges]]]])]]
```

```
{0.0156264 Second, {2 → 1, 3 → 2, 4 → 3, 5 → 3, 6 → 2, 7 → 2, <<2474>>,
  2492 → 2487, 2493 → 2466, 2494 → 2466, 2495 → 2494, 2496 → 2494}}
```

```
Short[formalUCClabels = Union[Flatten[formaluccedges]]]
```

```
{s1, s1-1, s1-101, s1-101a, s1-101b, <<2487>>, s9-708, s9-709, s9-709a, s9-709b}
```

```
edges2label = Dispatch[MapIndexed[Rule[#2[[1]], #1] &, formalUCClabels]];
```

```
Short[formalUCCcoords = GraphCoordinates[formaluccgraph, Method -> "RadialDrawing"]]
```

```
{{0.0821706, 9.14582}, <<1>>, <<2492>>, {<<1>>}, {3.45919, <<19>>}}
```

I can also create a Combinatorica version of the network.

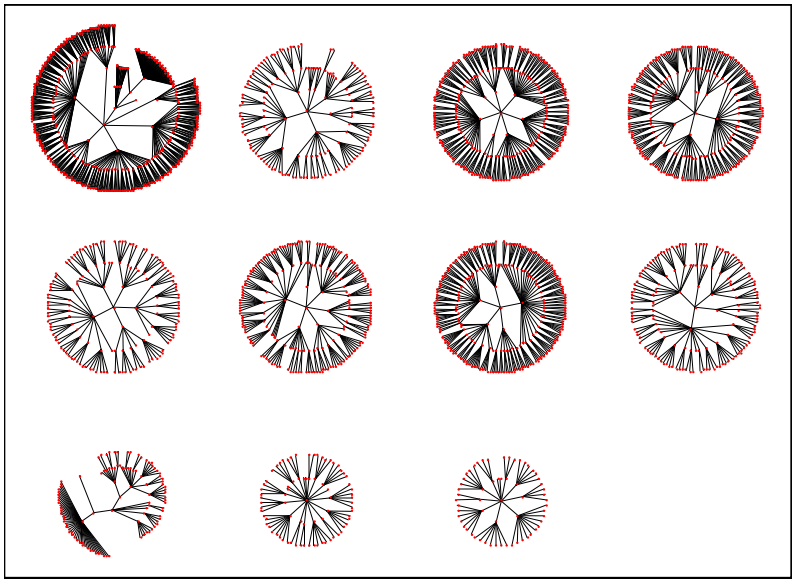
```
formalUCCgraphC = SetVertexLabels[
  FromOrderedPairs[List @@@formaluccgraph, Type → Directed], formalUCClabels]
```

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

-Graph:<2485, 2496, Directed>-

ShowLess | ShowMore | ShowFullOutput | SetSizeLimit...
```

■ A Picture of the Formal Structure of the Uniform Commercial Code

The GraphPlot package can now be used to show the formal structure of the Uniform Commercial Code. I use a "radial embedding" method because I know, in advance, that the structure will be tree like. The remaining code essentially mimics that used above to prepare the visualization of the textual structure of the UCC. The result is, indeed, a set of disconnected trees, each reflecting an article of the UCC. The trees are disconnected because there is no way for any provision in, say Article 2 of the UCC to be part of the hierarchy for, say, Article 7.

```
gpformal = If[$VersionNumber ≥ 6., Panel[GraphPlot[formaluccgraph,
    VertexStyleFunction → ({Hue[0], Tooltip[Disk[#, 0.05], formalUCClabels[[#]]} &),
    VertexCoordinates → formalUCCcoords],
   "The Formal Structure of the Uniform Commercial Code"],
  GraphPlot[formaluccgraph, VertexStyleFunction → ({Hue[0], Disk[#, 0.05]} &),
   VertexCoordinates → formalUCCcoords,
   PlotLabel -> "The Formal Structure\nof the Uniform Commercial Code"]]
```



The Full Structure of the UCC

Creating the Full Structure

With the formal and textual structures of the UCC now recovered, I can combine the two to obtain the full structure of this body of law. Basically, I do this by joining the edges of the textual graph with the edges of the formal graph.

```
Short[uccedges = Join[formaluccedges, textualuccedges]]
```

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

```
{{s1-1, s1}, {s1-101, s1-1}, <<9996>>, {s9-709b, dfinancingstatement}}
```

ShowLess | ShowMore | ShowFullOutput | SetSizeLimit...

I can use the same techniques shown above to produce data structures representing this full UCC network to both the GraphPlot package and the Combinatorica package.

```
Short[allucclabels = Union[Flatten[uccedges]]]
```

```
{2-104, 2-106, 2-401, 2-403, <<2833>>, warehouse, Warehousereceipt, Writing}
```

```
AbsoluteTiming[Short[uccgraph =
   Rule @@@ (uccedges /. Dispatch[MapIndexed[Rule[#1, #2[[1]]] &, allucclabels]])]]
```

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

{0.0312528 Second, {310 → 309, 311 → 310, 312 → 311, 313 → 311,
  <<9991>>, 2801 → 194, 2804 → 2404, 2804 → 2251, 2804 → 155}}

ShowLess  ShowMore  ShowFullOutput  SetSizeLimit...
```

```
alledges2label = Dispatch[MapIndexed[Rule[#2[[1]], #1] &, allucclabels]];
```

```
gd = SetVertexLabels[
  FromOrderedPairs[List @@@ uccgraph, Type → Directed], allucclabels]
```

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

-Graph:<9999, 2840, Directed>-

ShowLess  ShowMore  ShowFullOutput  SetSizeLimit...
```

I can also create an undirected version of the network, which is useful for portions of the analysis.

```
g = RemoveSelfLoops[MakeUndirected[gd]]
```

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

-Graph:<8758, 2840, Undirected>-

ShowLess  ShowMore  ShowFullOutput  SetSizeLimit...
```

And I can extract the largest connected component of this undirected network.

```
gu = With[{cc = ConnectedComponents[g]},
  InduceSubgraph[g, cc[[First@Ordering[Length /@ cc, -1]]]]]
```

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

-Graph:<8756, 2837, Undirected>-

ShowLess  ShowMore  ShowFullOutput  SetSizeLimit...
```

- **Preparing to Visualize the Full Structure of the Uniform Commercial Code**

I can now visualize the full Uniform Commercial Code using the same techniques as before.

```
Short[fullUCCcoords = GraphCoordinates[gu,
   Method → {"SpringElectricalModel", "RepulsiveForcePower" → -1.5, RecursionMethod →
      {"Multilevel", CoarseningScheme → "MaximalIndependentVertexSetRugeStuben"}}]]
```

```
{{2.59161, -2.54955}, <<2835>>, {-3.67459, <<19>>}}
```

```
Short[fullUCClabels = GetVertexLabels[gu]]
```

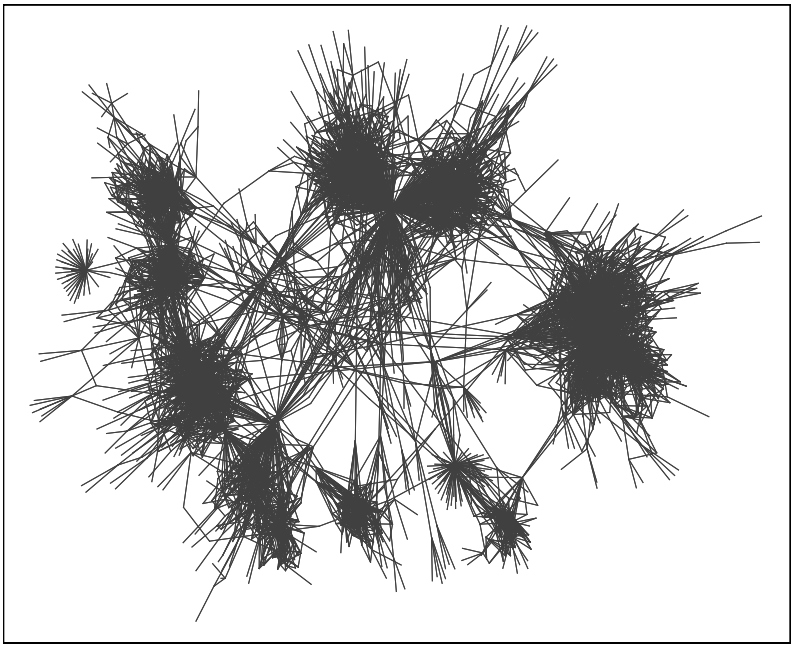
```
{2-104, 2-106, 2-401, 2-403, <<2830>>, warehouse, Warehousereceipt, Writing}
```

A Picture of the Full Structure of the Uniform Commercial Code

I use GraphPlot again to visualize the full structure of the Uniform Commercial Code. The result is a set of intricately connected clusters each of which contains a few ties to most of the other clusters and each of which contains a small number of satellite "clusterettes." As discussed below, the picture resembles that of "small world networks" in which the diameter of a highly clustered system is reduced through the existence of just a few weak ties between key nodes of each of the clusters.

```
gpfull = If[$VersionNumber ≥ 6., Panel[GraphPlot[gu,
    EdgeStyleFunction → ({Thickness[0.001], GrayLevel[0.25], Line[{#1, #2}]} &),
    VertexCoordinates → fullUCCcoords,
    VertexStyleFunction → ({Hue[0], Tooltip[Disk[#, 0.002], fullUCClabels[[#]]]} &)],
   "The Structure of the Full Uniform Commercial Code"], GraphPlot[gu,
   EdgeStyleFunction → ({Thickness[0.001], GrayLevel[0.25], Line[{#1, #2}]} &),
   VertexCoordinates → fullUCCcoords,
   VertexStyleFunction → ({Hue[0], Disk[#, 0.002]} &)]]
```

The Structure of the Full Uniform Commercial Code



Analyzing the Uniform Commercial Code Network

I can use Mathematica conveniently to do some basic exploration of the full UCC network and of the two components from which it is derived: the textual graph and the formal graph.

Basic Analysis

Graph Density

One basic measure of graphs is their density: the ratio between edges that exist and all possible edges that could potentially exist. All the variants of the UCC network the directed version, the undirected version, the formal version, the textual version are extremely sparse. Of the connections that might possibly exist among the different "nodes," only 2/10 of a percent exist at most.

```
graphdensity[g_Graph] := M[g] / (V[g] (V[g] - 1) / 2)
```

```
With[{t = Thread[{{"Full UCC Undirected Graph", "Full UCC Directed Graph",
      "Full UCC Undirected Graph (largest connected component)",
      "Formal UCC Graph", "Textual UCC Graph (largest connected component)"},
     N[graphdensity /@ {g, gd, gu, formalUCCgraphC, textualUCCgraphCC}]}]},
 If[$VersionNumber ≥ 6., Panel[Grid[t, RowLines → True, ColumnLines → True,
    ColumnAlignments → Left], "Graph Density"], TableForm[t]]]
```

Graph Density

Full UCC Undirected Graph	0.0021724
Full UCC Directed Graph	0.0024802
Full UCC Undirected Graph (largest connected component)	0.0021765
Formal UCC Graph	0.0007980
Textual UCC Graph (largest connected component)	0.0031722

Degree Distribution

Each node in a directed network has an "out degree" and an "indegree." The node's out degree is the number of other nodes it points to. The node's in degree is the number of nodes that point to it. The statistical distribution of in degrees among the nodes is often a useful measure of the structure of a graph.

```
Shallow[networkdegreedistributions =
  Outer[#2[#1] &, {gd, textualUCCgraphCC}, {OutDegree, InDegree}, 1, 1], 4]
```

```
{{{<<2840>>}, {<<2840>>}}, {{<<2175>>}, {<<2175>>}}}
```

```
Short[exceedances =
  Map[With[{n = Last[#]}, Table[{i, Count[n, _?(# ≥ i &)]}, {i, 1, Max[#[[2]]]}]] &,
   networkdegreedistributions]]
```

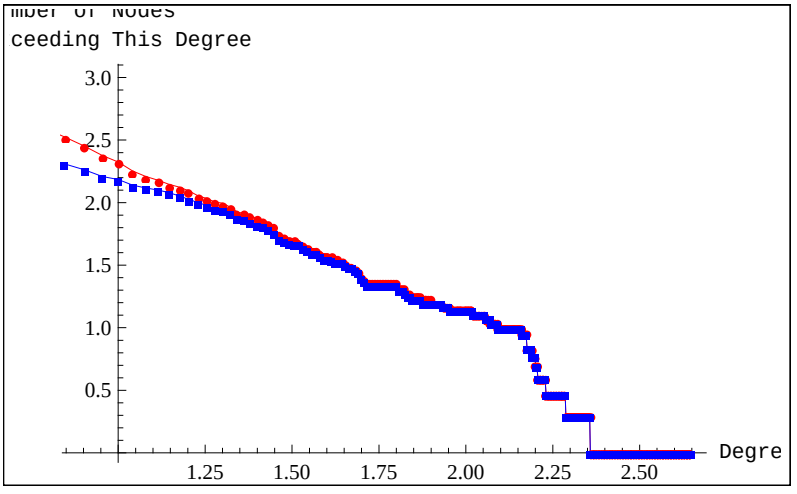
```
{{{1, 1070}, {2, 876}, <<440>>, {443, 1}, {444, 1}}, {<<1>>}}
```

```
Needs["Graphics`Graphics`"]
```

```
logexceedances = exceedances /. {a_Integer, b_Integer} -> {Log[10, a], Log[10, b]};
```

```
Needs["Graphics`MultipleListPlot`"]
```

```
If[$VersionNumber ≥ 6., Panel[ListLinePlot[logexceedances, Axes → True,
    Ticks → Automatic, PlotMarkers → Automatic, PlotStyle → {{Red}, {Blue}},
    AxesLabel → {"Degree", "Number of Nodes\nExceeding This Degree"}],
  "Log-Log Plot of Degree Distribution\nfor Full and Textual UCC networks"],
 MultipleListPlot[Sequence @@ logexceedances, Axes → False, Frame → True,
  Ticks → Automatic, PlotMarkers → Automatic, PlotStyle → {{Red}, {Blue}},
  AxesLabel → {"Degree", "Number of Nodes\nExceeding This Degree"}, PlotLabel ->
   "Log-Log Plot of Degree Distribution\nfor Full and Textual UCC networks"]]
```



Most of the resulting plot looks fairly linear, suggesting that the UCC has similarities to a truncated scale free network. (http://en.wikipedia.org/wiki/Scale-free_network; http://mathworld.wolfram.com/Scale-FreeNetwork.html). Several points follow from this resemblance. First, the existence of some nodes of very high degree means that there are a couple of key provisions of the UCC as to which any imprecision or confusion in their meaning can have cascading effects throughout the legal system. Problems with other less well connected provisions of the UCC may not ruin operation of the legal system, however. Second, it means that studies of other roughly scale free systems, such as the Internet, may have relevance to an understanding of the Uniform Commercial Code.

■ Centrality Measures

One can also use network methods to try and determine the most "important" or central provisions or concepts in the UCC. Three methods are often used: (1) citation; (2) Markov Centrality; (3) Closeness or "Kevin Bacon" centrality; and (4) Betweenness Centrality.

■ Most Cited

The chart below shows the most cited "nodes" of the full UCC network and the textual UCC network. As one can see, they are quite similar. (The prefixing of the letter "d" in front of certain terms denotes that the term in question is defined specifically for Article 9 of the Uniform Commercial Code, which has a special provenance).

```
With[{t = MapThread[Part[
      Thread[{GetVertexLabels[#1], Last[#2]}], Reverse@Ordering[Last@#2, -20]] &,
    {{gd, textualUCCgraphCC}, networkdegreedistributions}]},
 If[$VersionNumber ≥ 6., Row[MapThread[Panel[Grid[#, ColumnAlignments → Left], #2] &,
    {t, {"Citations to Full UCC", "Citations to Textual UCC"}}]],
  ColumnForm[Insert[Map[TableForm, t], " ", 2]]]]
```

Citations to Full UCC

Goods	444
ddebtor	227
Instrument	193
dsecuredparty	169
Lessee	160
Buyer	158
Seller	154
Lessor	149
Bank	149
Leasecontract	144
dfinancingstatement2	123
Item	117
contract	113
Paymentorder	104
Issuer	89
dgoods	85
s9-102a	79
Receivingbank	74
drecord	69
Lease	67

Citations to Textual UCC

Goods	444
ddebtor	227
Instrument	193
dsecuredparty	169
Lessee	160
Buyer	158
Seller	154
Lessor	149
Bank	149
Leasecontract	144
dfinancingstatement2	123
Item	117
contract	113
Paymentorder	104
Issuer	89
dgoods	85
Receivingbank	74
drecord	69
Lease	67
Indorsement	66

■ Markov Centrality

Another frequent measure of the centrality of a particular node on a graph, and a cousin of the "most cited" measure, is its "Markov Centrality." The idea here is to start at some random position on the graph and then take an infinite-length random walk on the graph. By random walk, I mean that the walker starts at some node and then randomly chooses an outgoing edge to follow to the next node. The process then repeats itself. The "Markov Centrality" of a node is the probability that, at the end of such an infinite process, the walker will find itself on a particular node. It is generally simpler to use this process on an undirected graph, and that is the course of action that will be followed here.

The Mathematica code below determines the Markov Centrality of each node in the graph. The idea is to create a Markov transition matrix in which the probability of going from node i to node j is zero if the two nodes are unconnected and one divided by the number of outgoing nodes otherwise. In theory, the first eigenvector of the transpose of the matrix created by this process is the sought-for result. In practice, for large graphs, computation of the eigenvector may require some methodological tweaking and a willingness to accept some modest approximation.

```
Options[eigenGraph] = Options[Eigenvectors];

eigenGraph[g_, opts___] :=
 With[{ev = First[Eigenvectors[Map[# / Max[1, Total[#]] &, N[ToAdjacencyMatrix[g]]]^T,
      1, opts]]}, ev / Total[ev]]
```

```
Options[markovcentrality] = Options[eigenGraph];

markovcentrality[g_, opts___] := With[{ϵ = eigenGraph[g, opts]},
  Part[Thread[{GetVertexLabels[g], ϵ}], Reverse[Ordering[ϵ]]]];
```

Here, I use this code to determine the Markov Centrality of the nodes in the full UCC graph and the textual UCC graph. I show the most central nodes in the table below.

```
AbsoluteTiming[gmarkovc =
  markovcentrality[g, Method → {"Arnoldi", "StartingVector" → Table[1./V[g], {V[g]}],
    "MaxIterations" → 1000, "Tolerance" → 0.0000001}];]
```

```
{4.3753920 Second, Null}
```

```
AbsoluteTiming[
 textualUCCgraphCCUmarkovc = markovcentrality[MakeUndirected[textualUCCgraphCC],
   Method → {"Arnoldi", "StartingVector" → Table[1./V[textualUCCgraphCC],
     {V[textualUCCgraphCC]}], "MaxIterations" → 1000, "Tolerance" → 0.0000001}];]
```

```
{2.7658728 Second, Null}
```

```
If[$VersionNumber ≥ 6.,
 Row[MapThread[Panel, {{Grid[Take[gmarkovc, 20], ColumnLines → True, RowLines → True,
      ColumnAlignments → Left], Grid[Take[textualUCCgraphCCUmarkovc, 20],
      ColumnLines → True, RowLines → True, ColumnAlignments → Left]},
    {"Markov Centrality, Full UCC", "Markov Centrality, Textual UCC"}}]],
 ColumnForm[Insert[Map[TableForm, {Take[gmarkovc, 20],
     Take[textualUCCgraphCCUmarkovc, 20]}], " ", 2]]]
```

Markov Centrality, Full UCC	
Goods	0.0185391
dsecuredparty	0.0102108
ddebtor	0.0088417
Instrument	0.0081571
Seller	0.0073585
Lessee	0.0072445
Buyer	0.0072445
Lessor	0.0070163
Bank	0.0069592
Leasecontract	0.0066170
dfinancingstatement	0.0064459
Paymentorder	0.0053050
contract	0.0052479
Issuer	0.0052479
dgoods	0.0050768
Item	0.0047345
s9–102a	0.0045634
drecord	0.0034226
Receivingbank	0.0034226
Indorsement	0.0034226

Markov Centrality, Textual UCC	
Goods	0.0259419
dsecuredparty	0.014288
ddebtor	0.0123723
Instrument	0.0114144
Seller	0.0102969
Lessee	0.0101373
Buyer	0.0101373
Lessor	0.0098180
Bank	0.0098179
Leasecontract	0.0092592
dfinancingstatement	0.0090198
Paymentorder	0.0074233
contract	0.0073435
Issuer	0.0073435
dgoods	0.0071040
Item	0.0066251
drecord	0.0047892
Receivingbank	0.0047892
Indorsement	0.0047892
Lease	0.0047094

The results are again quite similar regardless of whether the full or the textual network is used. The results are also quite similar to those for simple "citation." Goods is the most central part of the UCC, with the terms "secured party" (as used in Article 9), "debtor" (as used in Article 9), Instrument, Seller, Lessee, Buyer and Bank being the next most central.

- **Kevin Bacon Centrality**

A third way of measuring the importance of nodes in a network is average closeness. This measurement depends on the concept of a "geodesic" on a graph. A "geodesic" between two nodes on a graph is the smallest set of edges that go from one node to the other. Notice that for a connected graph with n nodes, there are n^2 geodesics (n(n-1)/2 if the graph is undirected, each of which may contain up to n values. Thus a 2,000 node connected graph will have 4 million geodesics, each of which may contain a number of integers bounded only by the diameter of the graph. Nodes at the "center" of a network will tend to be closer on average to other nodes via geodesics than nodes on the periphery. The "Kevin Bacon" name for this measure derives from the belief (since proven not quite correct) that actor Kevin Bacon lay at the center of the Hollywood movie database using this measure of centrality. As it turns out, actor Rod Steiger lies at the center.

One can use Combinatorica to find the nodes on the shortest path between any two nodes. Unfortunately, the algorithm depends on *Combinatorica*'s **AllPairsShortestPath** function, which does not scale particularly well to large graphs. The code listed below takes up to eight hours to run on a relatively swift computer. Some of the code below is thus devoted to file storage and retrieval of this information to avoid recomputation costs.

```
Short[textualapsp = AllPairsShortestPath[textualUCCgraphCCU, Parent]]

Short[gapsp = AllPairsShortestPath[gu, Parent]]

gapsp >>
 "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph Theoretic Structure of Common Law\\UCC\\gapsp"

textualapsp >>"H:\\CDR\\LAW&ECON\\ARTICLE\\Graph
   Theoretic Structure of Common Law\\UCC\\textualapsp"

gapsp = <<"H:\\CDR\\LAW&ECON\\ARTICLE\\Graph
     Theoretic Structure of Common Law\\UCC\\gapsp";

textualapsp = <<"H:\\CDR\\LAW&ECON\\ARTICLE\\Graph
     Theoretic Structure of Common Law\\UCC\\textualapsp";
```

The "kevinbacon" function below converts the output of the **AllPairsShortestPath** function (when used Parent as its second argument) to the Kevin Bacon measure of average closeness.

```
kevinbacon[apsp_] := Mean[Cases[#, _Integer?(# > 0 &)]] & /@ (First@apsp)
```

I now determine Kevin Bacon closeness on the full UCC network and the textual network. The results are somewhat surprising. Although "goods" retains its place as the central theme of the UCC, other less well known provisions assume higher prominence. Section 1-301(g), a sort of meta-rule for determining which rule of the UCC to apply becomes quite prominent. So does the concept of a "buyer in the ordinary course of business," a concept addressing those who purchase goods in a fashion that violates the rights of third parties. Various provisions of UCC Article 7 (Documents of Title) also appear as central to the textual database.

```
If[$VersionNumber ≥ 6.,
 Row[MapThread[Panel, {MapThread[With[{kb = kevinbacon[#]}, Grid[
        Take[Part[Thread[{GetVertexLabels[#2], N[kb]}], Ordering[kb]], 10],
        RowLines → True, ColumnLines → True, ColumnAlignments → Left]] &,
     {{gapsp, textualapsp}, {gu, textualUCCgraphCCU}}],
    {"Kevin Bacon Closeness, Full UCC", "Kevin Bacon Closeness, Textual UCC"}}]],
 ColumnForm[Insert[MapThread[With[{kb = kevinbacon[#]}, TableForm[
       Take[Part[Thread[{GetVertexLabels[#2], N[kb]}], Ordering[kb]], 10]]] &,
    {{gapsp, textualapsp}, {gu, textualUCCgraphCCU}}], " ", 2]]]
```

Kevin Bacon Closeness, Full UCC	
Goods	3.7165
s1–301g	4.04231
Buyerinordinarycourseofbusi	4.17348
s2A–3091	4.23484
s7–402	4.25212
s7–305b	4.25353
s2–4022	4.25599
s2A–105	4.26904
s7–203	4.27292
s7–502a	4.27891

Kevin Bacon Closeness, Textual UCC	
Goods	3.55796
s2A–3091	3.95998
Buyerinordinarycourseofbusi	4.00578
s7–502a	4.10488
s7–301a	4.11132
s7–305b	4.11132
s7–402	4.11132
s7–203	4.11224
s7–301b	4.11224
s7–301d	4.11224

Betweenness Centrality

Another way of measuring the importance of nodes in a network is there so-called "Betweenness Centrality." This measurement likewise depends on the concept of a "geodesic" on a graph. The "betweenness" of a particular node is the fraction of geodesics between all possible nodes on the graph that traverse the particular node. In theory, one can also compute the "betweenness" of a particular edge as the fraction of geodesics between all possible nodes on the graph that traverse the particular edge.

To implement this in Mathematica, one first takes the list of path predecessors created as the second part of the output from **AllPairsShortestPath** and then creates the actual shortest paths. The UpperDiagonalOnly option prevents double counting of paths on undirected graphs and the ProcessingFunction option permits different forms of computation as well as side effects to occur, thus potentially saving memory. The long set of somewhat ugly code below outlines the rather complex process of determining betweenness centrality for the fairly large UCC network without crashing the *Mathematica* kernel due to memory limitation issues. Basically, it involves using a somewhat elaborate "ProcessingFunction" to compute "betweenness" as shortest paths are being determined rather than storing information and computing it afterwards.

```
Needs["LinearAlgebra`MatrixManipulation`"]
```

```
Options[MyParentsToPaths] = {UpperDiagonalOnly → True, ProcessingFunction → (# &)};
```

```
MyParentsToPaths[m_?SquareMatrixQ, opts___] :=
 With[{f = ProcessingFunction /. {opts} /. Options[MyParentsToPaths]},
  MapIndexed[Function[{z, i}, If[And[i[[1]] > i[[2]],
      UpperDiagonalOnly /. {opts} /. Options[MyParentsToPaths]], {Indeterminate},
     f[Rest[Reverse[FixedPointList[m[[i[[1]], #1]] &, i[[2]]]]]]]], m, {2}]]
```

```
Short[textualnodecounts = Array[0 &, Length[Last@textualapsp]]]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, <<2151>>, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
MyParentsToPaths[Last@textualapsp, ProcessingFunction → (Function[q,
     (textualnodecounts = MapAt[# + 1 &, textualnodecounts, Partition[q, 1]]; "x")])];
```

```
textualnodecounts >> "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph
    Theoretic Structure of Common Law\\UCC\\textualnodecounts";
```

```
Short[fullnodecounts = Array[0 &, Length[Last@gapsp]]]
```

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, <<2813>>, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

```
MyParentsToPaths[Last@gapsp, ProcessingFunction → (Function[q,
     (fullnodecounts = MapAt[# + 1 &, fullnodecounts, Partition[q, 1]]; 0)])];
```

```
fullnodecounts >> "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph
    Theoretic Structure of Common Law\\UCC\\fullnodecounts";
```

Once these complex computations are done, one can reload the data.

```
textualnodecounts =
  << "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph Theoretic Structure of Common
     Law\\UCC\\textualnodecounts";
```

```
fullnodecounts =
  << "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph Theoretic Structure of Common
     Law\\UCC\\fullnodecounts";
```

```
Needs["Statistics`DescriptiveStatistics`"]
```

```
betweenness[parentmatrix_, nodes] :=
 Sort[Frequencies[DeleteCases[Flatten[parentmatrix], Indeterminate]]]
```

```
betweenness[parentmatrix_, edges] :=
 Sort[Frequencies[Flatten[Map[Partition[#, 2, 1] &, parentmatrix, {2}], 2]]]
```

```
AbsoluteTiming[textualmp2p = MyParentsToPaths[Last@textualapsp];]
```

```
{94.6305360 Second, Null}
```

```
textualmp2p >> "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph
   Theoretic Structure of Common Law\\UCC\\textualmp2p"
```

```
AbsoluteTiming[gmp2p = MyParentsToPaths[Last@gapsp];]
```

```
{165.1987344 Second, Null}
```

```
gmp2p >>
 "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph Theoretic Structure of Common Law\\UCC\\gmp2p"
```

```
gmp2p = << "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph
     Theoretic Structure of Common Law\\UCC\\gmp2p";
```

```
textualmp2p = << "H:\\CDR\\LAW&ECON\\ARTICLE\\Graph
     Theoretic Structure of Common Law\\UCC\\textualmp2p";
```

I am now able to present results. As one can see, "goods" triumphs again as the center of the UCC with "isuser" and "secured party" (as used in Article 9) also playing important roles. This analysis reveals important roles for several somewhat obscure provisions, however. Section 2-309(1), which governs the situations in which goods (subject to the UCC) become fixtures (mostly governed by non-UCC real estate law) proves important as does section 9-502(a) governing the sufficiency of "financing statements," basically a notice of a security interest in property. Also highly "between" is section 7-502(a), which governs (roughly speaking) the circumstances under which "negotiation" of documents of title confers property interests in the underlying goods.

```
With[{fullbetweennesstable =
   Take[Reverse[Part[Thread[{GetVertexLabels[gu], fullnodecounts}],
      Ordering[fullnodecounts]]], 20], textualbetweennesstable = Take[
    Reverse[Part[Thread[{GetVertexLabels[textualUCCgraphCCU], textualnodecounts}],
      Ordering[textualnodecounts]]], 20]
 }, If[$VersionNumber ≥ 6., Row[MapThread[Panel[Grid[#, ColumnAlignments → Left],
      #2] &, {{fullbetweennesstable, textualbetweennesstable},
     {"Betweenness Full UCC", "Betweenness Textual UCC"}}]],
  {fullbetweennesstable, textualbetweennesstable}]]
```

Betweenness Full UCC	
Goods	1334082
Issuer	599586
dsecuredparty	560972
s7–502a	418204
s2A–3091	417056
s9–502a	415497
s1–301g	391457
s9–203b	344935
Bank	330844
Item	263832
s4–210c	256287
Instrument	246833
Goodfaith	244135
s9–102a	218556
Person	215542
ddebtor	212034
Seller	201549
s7–106	194859
dfinancingstatement	1642 69
s9–3	157784

Betweenness Textual UCC	
Goods	930520
dsecuredparty	482199
s2A–3091	457400
s9–502a	456893
Issuer	436724
s7–502a	338641
s9–203b	289691
s4–210c	236517
Item	233220
Bank	213625
ddebtor	189970
Instrument	171599
Person	167526
Goodfaith	158046
dfinancingstatement	149220
Buyerinordinarycourseofbusiness	1e+05
Draft	113560
Seller	111695
Securityinterest	105479
Creditor	103055

■ Deconstructing the Network: Minimum Cuts

A difficulty in studying highly connected networks is that everything is ultimately connected to everything else and, yet, studying everything at once is far too complicated. It is thus sometimes useful to break the small world network up into pieces and study each of the parts, accepting that this reductionist approach is flawed precisely because it omits what may be crucial connections to other components of the network. The "MinCut" algorithm provided in *Mathematica*'s GraphPlot package facilitates this approach. Basically, it tries to use a minimum number of edge cuts to create a graph with a user-specified number of disconnected components. The function implements this process by returning a list of lists, each of which contains the vertices in each of the resulting components of the network. These components can then be studied by inducing a subgraph based on the vertex list.

Traditionally, in American law schools, the Uniform Commercial Code is broken down into courses such as sales, payment systems and secured transactions, often with some sort of miscellaneous course added on to deal with matters such as uncertificated securities or bulk sales. I can use *Mathematica* to see if an algorithmically optimal partitioning of the textual version of the UCC matches the traditional compartmentalization of the subject matter employed by American law schools. I thus cut the full and textual UCC into four parts and then show the contents and structure of these four parts.

```
Shallow[fullmincut = MinCut[g, 4]]
```

```
{{51, 71, 78, 80, 81, 92, 95, 96, 102, 104, <<702>>},
 {3, 4, 5, 6, 10, 11, 12, 24, 27, 28, <<699>>},
 {1, 2, 7, 8, 9, 13, 14, 23, 25, 26, <<699>>},
 {15, 16, 17, 18, 19, 20, 21, 22, 40, 41, <<700>>}}
```

```
Shallow[textualmincut = MinCut[textualUCCgraphCCU, 4]]
```

```
{{1, 2, 7, 8, 9, 19, 21, 27, 31, 34, <<533>>},
 {3, 4, 5, 6, 10, 20, 22, 23, 24, 25, <<534>>},
 {40, 44, 46, 48, 57, 58, 59, 60, 64, 65, <<535>>},
 {11, 12, 13, 14, 15, 16, 17, 18, 36, 37, <<533>>}}
```

```
fulluccpartitions = Map[InduceSubgraph[g, #] &, fullmincut];
```

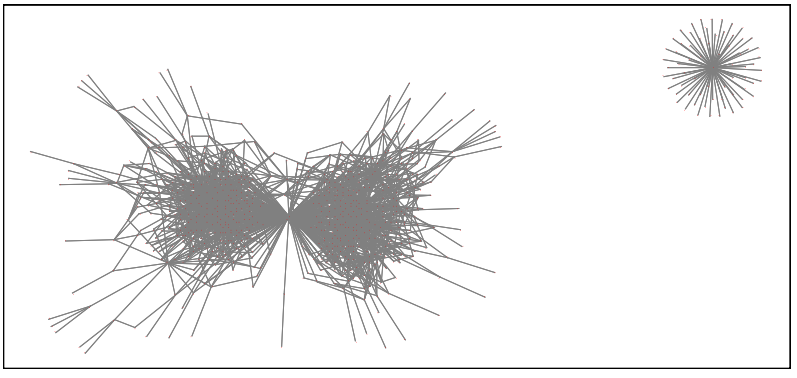
```
textualuccpartitions = Map[InduceSubgraph[textualUCCgraphCCU, #] &, textualmincut];
```

■ Deconstructing the Full Uniform Commercial Code

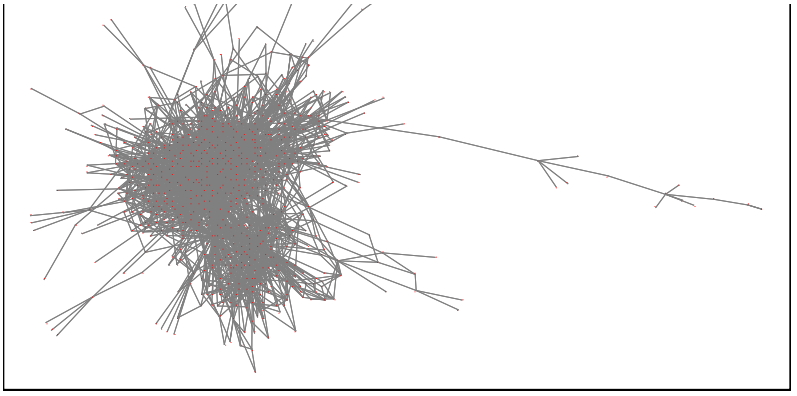
I now show the partitions for both the full UCC and the textual UCC. The first set of graphs shows that the full UCC indeed breaks down fairly well into the compartments traditionally used in teaching. The first partition is pretty much a traditional "Sales and Leasing" course with "goods" and "seller," "lease contract" and "lessor" constituting the central concepts. The second partition is predominantly a "Secured Transactions" course with "secured party", "debtor", "financing statement" and "goods" at the center. The third partition is the lesser-taught potpourri of more obscure UCC provisions governing bulk sales, uncertificated securities, warehouse receipts and similar arcane manners. The fourth partition clearly relates to payment systems and banking. It has "bank" and "instrument" at its center. But this algorithmic replication of the tradtiional learning may be caused partly by the confluence of the academy paying considerable attention to the formal structure of the UCC and the lack of connectivity in the formal graph between articles resulting in a preference for cutting the graph between articles.

```
If[$VersionNumber ≥ 6.,
 MapIndexed[With[{labels = GetVertexLabels[#]}, Panel[GraphPlot[#,
       Method → {"SpringElectricalModel", "RepulsiveForcePower" → -1.2,
         RecursionMethod → {"Multilevel", "CoarseningScheme" →
            "MaximalIndependentVertexSetRugeStuben"}}, Background → GrayLevel[1],
       EdgeStyleFunction → ({GrayLevel[0.5], Thickness[0.0001], Line[{#1, #2}]} &),
       VertexStyleFunction → ({Hue[0], Tooltip[Disk[#, 0.01], labels[[#]]]} &)],
      "Full Partition " <> ToString[#2[[1]]]]] &, fulluccpartitions] // Column]
```

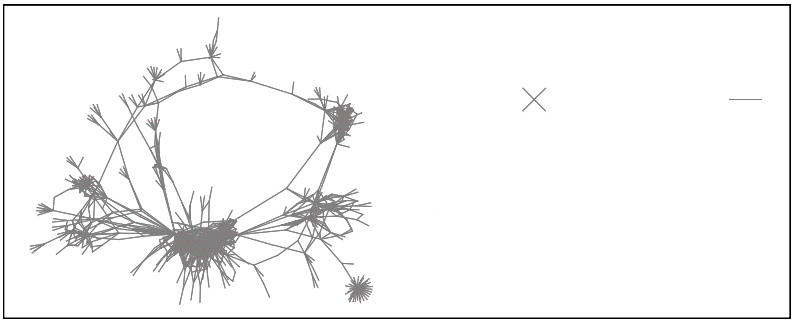
FullPartition1



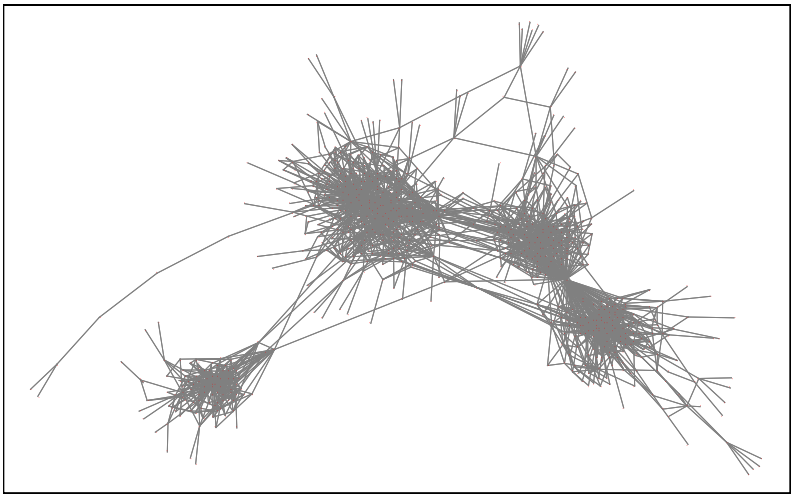
FullPartition2



FullPartition[3]



FullPartition[4]



The code and panels below show the most between nodes of the various partitions. I can use the simpler "betweenness" function here because the partitions are of a more manageable size than the entire UCC graph.

```
mostbetweenconcepts[g_, n_: 10] := Module[{t1apsp, t1mp2p},
  t1apsp = AllPairsShortestPath[g, Parent];
  t1mp2p = MyParentsToPaths[Last@t1apsp];
  Take[With[{b = betweenness[t1mp2p, nodes]},
    Reverse[Thread[{Part[GetVertexLabels[g], Last /@ b], First /@ b}]]], n]]
```

```
Column[MapThread[Panel[Grid[mostbetweenconcepts[#, 10], ColumnAlignments → Left],
    "Most Between Nodes of Full Partition " <> #2] &,
  {fulluccpartitions, {"1", "2", "3", "4"}}]]
```

MostBetweenNodesofFullPartition1

Goods	112015
Seller	39502
Leasecontract	25393
Lessor	19404
contract	15848
Termination	10234
Lessee	8890
s2A–2	7890
s2A–221	7099
Supplier	6875

MostBetweenNodesofFullPartition2

dsecuredparty	763788
ddebtor	440977
dfinancingstatement	351166
dgoods	227977
s9–3	190677
drecord	143633
s9–406	120155
s2–2102	104255
s9–708	97966
s2–210	97777

MostBetweenNodesofFullPartition3

Issuer	114921
Delivery	79761
Securitycertificate	64706
s7–502a	37033
Securityinterest	36522
s1–2	34928
s1–204	32998
s1–310	31661
s1–201	30177
s1–201b	29814

MostBetweenNodesofFullPartition4

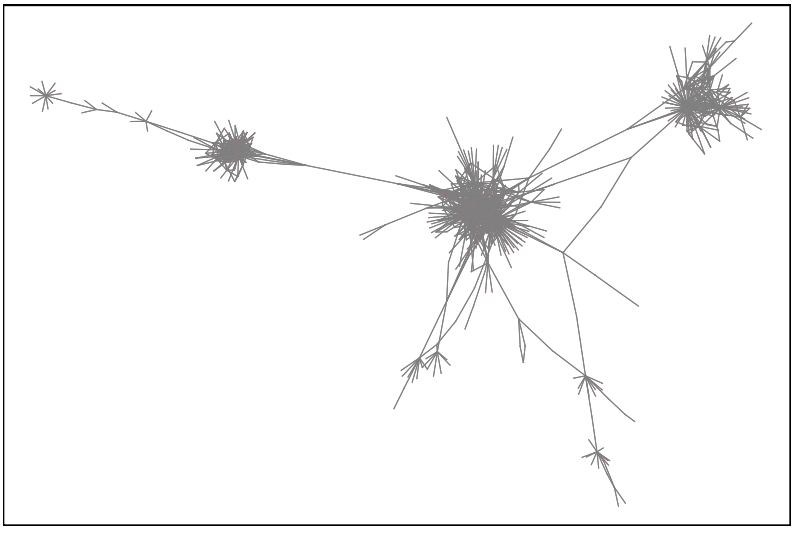
Bank	690899
value	517199
Instrument	437577
s4–211	399633
Item	356133
Draft	316755
Goodfaith	175144
s3–303	161433
s3–103b	150500
s4A–302b	147144

■ Deconstructing the Textual Uniform Commercial Code

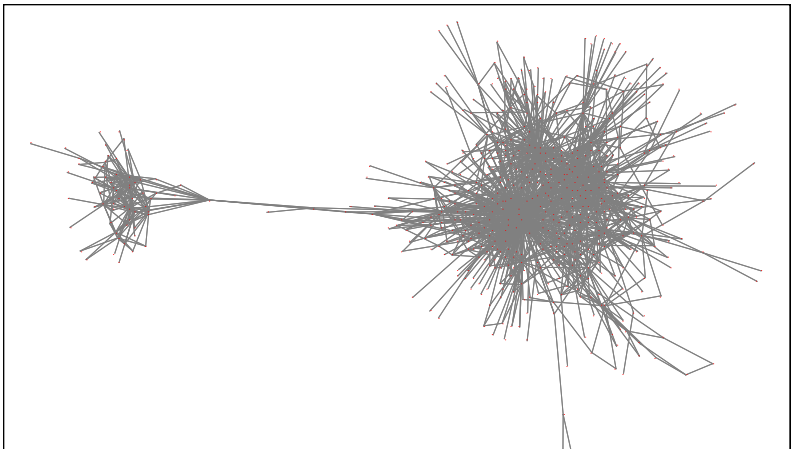
If one looks at the textual UCC, the breakdown into four partitions results in a far less traditional decomposition of the code. The first partition appears to have leasing (Article 2A) at is center but contains significant satellites involving letters of credit (Article 5) and filing of securities interests under Article 9. The second partition involves mostly secured transactions (Article 9) but also a significant satellite involving bulk transfers (Article 6). The term "secured party" is the most between in the second partition. The third partition involves two distinct segments (suggesting that, really, one should break the textual UCC into five parts), the first involving sales and the second involving aspects of bank deposits (Article 4) and fund transfers (Article 4A), with goods, seller and bank being the central concepts. The final partition involves payment systems (Article 3) and uncertificated securities (Article 8) with issuer, instrument, party and indorsement providing the central concepts.

```
If[$VersionNumber ≥ 6.,
 MapIndexed[With[{labels = GetVertexLabels[#]}, Panel[GraphPlot[#,
       Method → {"SpringElectricalModel", "RepulsiveForcePower" → -1.2,
         RecursionMethod → {"Multilevel", "CoarseningScheme" →
            "MaximalIndependentVertexSetRugeStuben"}}, Background → GrayLevel[1],
       EdgeStyleFunction → ({GrayLevel[0.5], Thickness[0.0001], Line[{#1, #2}]} &),
       VertexStyleFunction → ({Hue[0], Tooltip[Disk[#, 0.01], labels[[#]]]} &)],
      "Textual Partition " <> ToString[#2[[1]]]] &, textualuccpartitions] // Column]
```

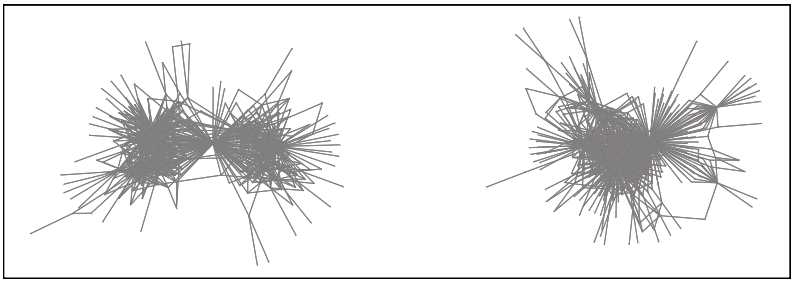
Textual Partition 1



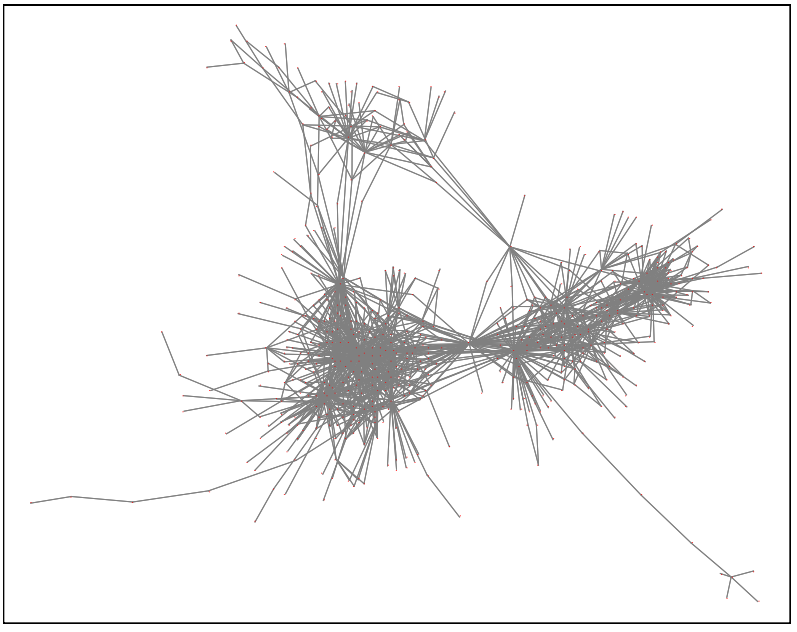
Textual Partition 2



Textual Partition 3



Textual Partition 4



I can again run a "betweenness centrality" computation on the four partitions of the textual network.

```
Column[MapThread[Panel[Grid[mostbetweenconcepts[#, 10], ColumnAlignments → Left],
    "Most Between Nodes of Textual Partition " <> #2] &,
  {textualuccpartitions, {"1", "2", "3", "4"}}]]
```

Most Between Nodes of Textual Partition 1

Lessee	48280
s2A–5281	46119
s2A–3091	46117
s1–302	45084
s9–502a	44991
s5–103c	44793
s5–117d	40902
Lessor	40341
dfinancingstatement	36454
value	19457

Most Between Nodes of Textual Partition 2

dsecuredparty	63015
ddebtor	30396
s6–1033	30066
s9–610	19195
s9–623c	17547
dgoods	15983
s9–602	12127
s9–620	11580
dproceeds	7454
s9–109d	6964

Most Between Nodes of Textual Partition 3

Goods	27220
Seller	19936
Bank	10733
contract	6769
Item	5270
Paymentorder	5254
Receivingbank	3519
warehouse	2923
sale	2753
Sender	2559

Most Between Nodes of Textual Partition 4

Issuer	56170
Instrument	49122
Party	20299
Indorsement	17698
Uncertificatedsecurity	16359
Securitycertificate	14756
Delivery	13402
Financialasset	13397
s3–103b	11210
Draft	9417

The Core of the Uniform Commercial Code

One often interesting characteristic of a graph or network is its so-called "main core." The concept here is to find the maximal subset of nodes in a (undirected) graph that are connected to at least *x* other members of the subset, where *x* is an integer such that, for any number greater than *x*, the subset becomes empty. This subset can be taking a low value of *x* and then iteratively eliminating nodes that are connected to fewer than *x* other nodes. The process continues until a fixed point is reached. One then increases *x* by one. This outer loop continues until the subset disappears. So one remembers the subset generated by the last value of *x* for which the subset didn't disappear. While there is no exact meaning that can be ascribed to the main core of a graph composed of legal "stuff," I like to think of it as areas particularly prone to complexity, in which the level of interconnection is greatest. To be sure, complexity may occur at lower levels of connectivity, but in the main core, understanding of any one of the concepts constituting a node requires an understanding of relationships to a large variety of other concepts and provisions.

```
core[g_Graph, v_Integer, maxiters_Integer] := FixedPoint[
  InduceSubgraph[#, Flatten[Position[Degrees[#], _?(# ≥ v &)]]] &, g, maxiters]

maincore[g_Graph, vinit_Integer, maxiters_Integer, maxv_Integer] :=
 NestWhile[{#[[1]] + 1, core[#[[2]], #[[1]], maxiters]} &, {vinit, g},
   Function[V[#[[2]]] ≠ 0], 1, maxv, -1] /. {a_, gr_Graph} :⇒ {a - 1, gr}
```

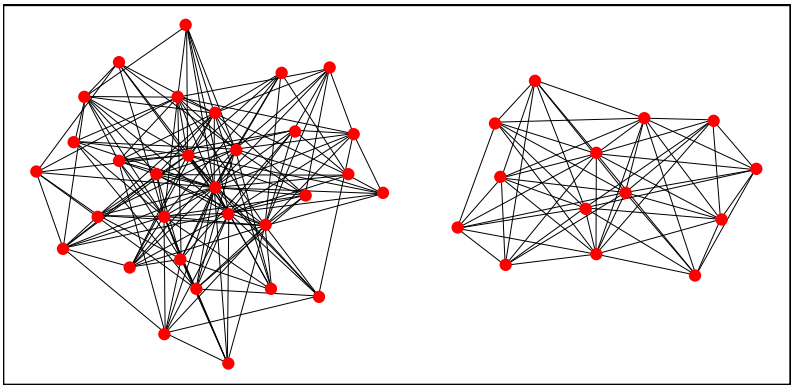
I now find the main core of the full and textual UCC networks.

```
mc = Map[maincore[#, 5, 30, 10] &, {gu, textualUCCgraphCCU}]
```

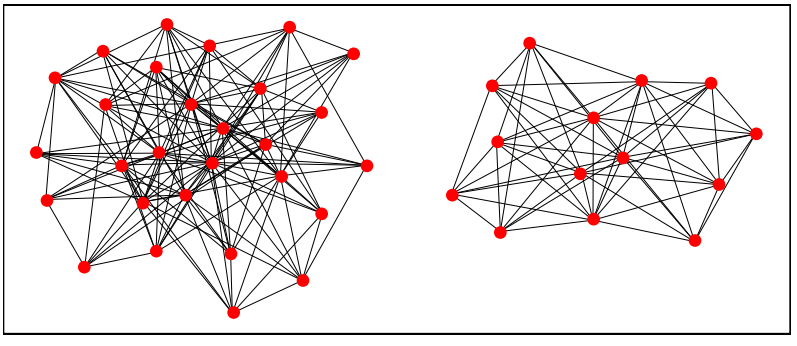
```
{{7, -Graph:<233, 45, Undirected>-}, {7, -Graph:<218, 42, Undirected>-}}
```

```
If[$VersionNumber ≥ 6.,
 Column[MapThread[Function[{z, ℓ}, Module[{g, gccoords, labels},
     g = Last@z;
     gccoords = GraphCoordinates[g];
     labels = GetVertexLabels[g]; Panel[GraphPlot[g,
       VertexStyleFunction → ({Hue[0], Tooltip[Disk[#, 0.05], labels[[#]]} &),
       VertexCoordinates → gccoords], ℓ]]],
   {mc, {"Main Core of Full Uniform Commercial Code",
     "Main Core of Textual Uniform Commercial Code"}}]],
 MapThread[Function[{z, ℓ}, Module[{g, gccoords, labels},
    g = Last@z;
    gccoords = GraphCoordinates[g];
    labels = GetVertexLabels[g];
    GraphPlot[g, VertexStyleFunction → ({Hue[0], Disk[#, 0.05]} &),
     VertexCoordinates → gccoords, PlotLabel → ℓ]]],
  {mc, {"Main Core of Full Uniform Commercial Code",
    "Main Core of Textual Uniform Commercial Code"}}]]
```

Main Core of Full Uniform Commercial Code



Main Core of Textual Uniform Commercial Code



Here I find two areas of maximal complexity, the first involving a cluster of issues relating to banks, payments and fund transfers, the second involving a cluster of issues relating mostly to defaults under a commercial lease.

■ Is the Uniform Commercial Code a "Small World Network"?

The embedding process visually indicates that,whether one considers the full structure of the Uniform Commercial Code or just the textual structure of the Uniform Commercial Code, the result is a classic "small world network," one which has a small diameter like random graphs but which is highly "clustered" like strongly regular graphs. (http://mathworld.wolfram.com/Scale-FreeNetwork.html; http://en.wikipedia.org/wiki/Small_world_phenomenon). *Mathematica* permits me to assess the hint provided by the visualizations.

I first assess the diameter of the full UCC network and the textual UCC network. I do so using the PseudoDiameter command of the GraphPlot package. I find that it they are 13 and 14 respectively.

```
(PseudoDiameter[#][[1, 1]]) & /@ {gu, textualUCCgraphCCU}
```

```
{13, 14}
```

Experiments with true random graphs of similar numbers of vertices and similar numbers of edges show that the diameter on the connected components is usually equal to 8 to 9, although sometimes the figure is higher. Thus, both the textual UCC graph and the full UCC are somewhat more "structured" than a random graph.

```
Module[{vfull = V[gu], vtextual = V[textualUCCgraphCCU],
  fullequivalent, textualequivalent},
 fullequivalent = RandomGraph[vfull,
   δ /. First[Solve[δ vfull (vfull - 1) / 2 == M[gu], δ]], Type → Undirected];
 textualequivalent = RandomGraph[vtextual,
   δ /. First[Solve[δ vtextual (vtextual - 1) / 2 == M[textualUCCgraphCCU], δ]],
   Type → Undirected];
 (PseudoDiameter[#][[1, 1]]) & /@ {fullequivalent, textualequivalent}]
```

```
{9, 10}
```

I now assess the clustering of the full and textual graphs. Clustering asks the following question: if node A is connected to node B and node A is also connected to node C, how likely is it that node B is connected to node C? The code below determines the approximate clustering coefficient of a node or set of nodes.

```
clustering[g_Graph?UndirectedQ, i : {__Integer}] := Module[{sg =
    Map[InduceSubgraph[g, #] &, Map[Complement[Neighborhood[g, #, 1], {#}] &, i]]},
  Map[{M[#], V[#] (V[#] - 1) / 2} &, sg]]
```

I can now determine and display for nodes of varying degrees, the mean clustering coefficient for nodes of the full UCC graph and the textual UCC graph. I use sampling to increase execution speed.

```
Short[uccclustertables = Map[Function[gr, With[{d = Degrees[gr]},
     Map[With[{z = Flatten[Position[d, #]]},
        {#, Length[z], clustering[gr, RandomChoice[z, Min[20, Length[z]]]]}] &,
      Select[Union[d], # ≥ 2 &]]]], {gu, textualUCCgraphCCU}]]
```

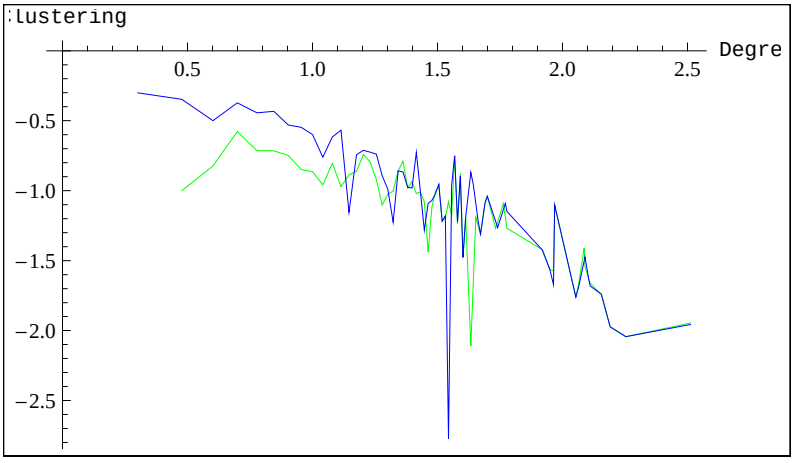
```
{{ <<1>> }, { <<1>> }}
```

```
Short[meanclusteringbydegree =
  Map[Map[{#[[1]], #[[2]], Mean[First /@ Part[#, 3]] / #[[3, 1, 2]]} &, #] &,
   uccclustertables]]
```

$$\left\{\left\{\{2, 333, 0\}, \ll 62 \gg, \left\{325, 1, \frac{298}{26\,325}\right\}\right\}, \{\ll 1 \gg, \ll 60 \gg, \{\ll 1 \gg\}\}\right\}$$

```
With[{means = Map[With[{t = Transpose[#]}, N[Dot @@ t / Total[First@t]]] &,
    Map[Part[#, {2, 3}] &, meanclusteringbydegree, {2}]]},
 Panel[ListLinePlot[DeleteCases[Map[Log[10, Part[#, {1, 3}]] &,
     meanclusteringbydegree, {2}], {_, -∞}, {2}],
   AxesLabel → {"Degree", "Clustering"}, PlotStyle → {{Green}, {Blue}}],
  "Clustering of UCC By Degree\nLog-Log Plot (Green is
    Full, Blue is Textual)\nMean Clustering is Full: " <>
   ToString[First@means] <> "; Textual: " <> ToString[Last@means]]]
```

Clustering of UCC By Degree
Log-Log Plot (Green is Full, Blue is Textual)
Mean Clustering is Full: 0.140713; Textual: 0.381221



The results for the full graph and textual graph are roughly consonant to what one sees in a small world graph. The full graph is less clustered because the subsections of a section of the UCC often do not reference each other.

Conclusion

Two types of conclusions that can one can draw from this study of the Uniform Commercial Code.

The first set of conclusions relates to the Uniform Commercial Code. This study shows the UCC to be a fairly sparse "small world network." Unlike most of the network of Supreme Court precedents earlier studied by this author, the UCC has definite "structure" and contains a set of loosely linked clusters. The UCC decomposes fairly well along the cleavages that traditionally mark its study, though part of this correspondence is an artifact of the confluence between the hierarchical structure of the UCC and the article-specific focus of most studies of the UCC. An alternative decomposition that permits study of portions of multiple articles simultaneously also appears to be sensible, however. This work shows the concepts of "goods" and of "secured parties" to have great prominence in the UCC, along with several other terms such as "debtor", "instrument" and "buyer in the ordinary course of business." The concept of "good faith," which many have heralded as somewhat of a UCC novelty, also has a promi-

nent place in the network. The most interrelated portions of the UCC, and those thus most likely to be complex, relate to fund transfers under article 4A and default under leases pursuant to article 2A.

The second set of conclusions relates to Mathematica. Mathematica's Regular Expression and Internet import capabilities make it feasible readily to reduce complex legal texts found on the Internet to be radically compressed into a mathematical network. This capability is facilitated when, as with the UCC, electronic versions of the text have already been expertly marked up. Once the network is created, Mathematica, with its GraphPlot package and Combinatorica augmenting its basic functionality, is able to handle many of the traditional problems in network analysis without great difficulty. For small networks, there is not a great need to resort to external functions via J/Link or to external programs using file transfer mechanisms. The system has considerable difficulty, however, scaling certain forms of analysis, notably those relying on distance measurements amongst the nodes, to larger networks. Some of the analyses for this paper took over eight hours to run on a relatively swift computers; others forced the Mathematica kernel to shut down with memory problems. Still others required uncomfortably baroque work arounds and a tolerance for approximation. Various extensions to the functionality of GraphPlot and Combinatorica would thus appear to be critical if Mathematica is to become a leading tool of network analysis. The UCC network, though large, is hardly among the larger networks that will be the subject of examination in the years ahead.