The Network Structure of Supreme Court Jurisprudence

Seth J. Chandler

This article begins a program of research examining the network structure of precedent-based judicial decision making, using the United States Supreme Court as an initial example. It develops a set of *Mathematica* tools that facilitate studies of large networks, including vehicles for *Mathematica* to communicate with external network analysis software.

■ Introduction

In common law jurisdictions such as the United States, courts frequently resolve disputes by citation and analysis of prior legal cases. The law may thus be thought of as a giant network containing textual information embedded in cases (nodes) and relationship information called citations (arcs) going from node to node. In recent years, the science of studying networks has developed [1] but, while there have been some primitive attempts to look at subsets of the vast legal network, until recently there has been little done to take advantage of modern technology and network theory. This article borrows techniques developed largely in sociology [2, 3] and physics and uses modern technology to learn about the law simply by studying its network structure. The article makes extensive use of the Java Universal Network/Graph Framework (JUNG) [4] via *J/Link* technology and facilitates communication of *Mathematica* graph structures to other network analysis programs such as Pajek by developing methods of import and export using GraphML. Although this article focuses on tool building, it is my hope that these efforts, along with pending publications on legal networks by Professor Thomas A. Smith of the University of San Diego Law School [5] and Professors James H. Fowler and Sangick Jeon of the University of California, Davis [6], will catalyze a set of studies in this field that will expand to cover other judicial systems and yet more sophisticated analysis of network information.

■ The Database

Construction of the data used for this project was a significant undertaking. And because similar challenges are likely to confront others working in this field,

some researchers involved in either legal networks or the XML and regular expressions technology used in their creation may find the following account useful. Those with a predominant focus on legal issues may wish to skip, however, to the Characteristics of the Supreme Court Network section.

□ Description of the Database

The raw data employed for this article is a set of files created in preparation of a commercial product known as USSC+ [7] and used for academic purposes here under a license generously granted by its owner. Each of the 27,000 or so files in the database contains marked-up versions of a full text of every United States Supreme Court decision rendered from 1831 to early 2005, as well as important decisions from the inception of the court in 1790 until 1831. (Basic information on the court may be found at en.wikipedia.org/wiki/Special:Search/Supreme_Court_of_the_United_States.) There are approximately 26,000 cases in the database at present, spanning approximately a gigabyte of information. A completed database extending the entire lifetime of the court is projected to be available in a year. (See Additional Material for a sample case file.)

This database was selected for several reasons. It contains the works of a court of importance so that the conclusions reached in this article may have importance for an audience of legal academics as well as a more multidisciplinary group. It is not so large as to be completely unmanageable and yet sufficiently large to pose significant challenges which, if surmounted, can be used as a guide to analysis of other textual databases. It is structured as a closed database in which links to cases other than Supreme Court cases are not followed. While in some sense this artificially confines the database, such constriction is essential to render issues of network analysis tractable. And the owner of the database was willing to donate its use for this academic purpose without complex negotiations over intellectual property rights.

□ Converting the Database into a Useful *Mathematica* Expression

Creation of the XML Files

Mathematica was used extensively in the elaborate and somewhat ugly process of converting this textual database into useful network information. To speed future input/output operations, individual case files were combined into approximately 500 large but simple XML files, each of which conforms to the following XML schema. (The schema was created by feeding a program known as XMLSpy [8] a copy of the XML file, having it deduce the schema that must have produced it, and then hand-tweaking that schema [9].) During this process, problematic characters such as ampersands were converted into entities such as "&amp."

```
XMLObject[Document][
 {XMLObject[Declaration][Version → 1.0, Encoding → UTF-8,
   Standalone → yes], XMLObject[Comment][W3C Schema generated
    by XMLSpy v2005 rel. 3 U (http://www.altova.com)]},
 XMLElement[{http://www.w3.org/2001/XMLSchema, schema},
  {{http://www.w3.org/2000/xmlns/, xs} →
    http://www.w3.org/2001/XMLSchema,
   elementFormDefault → qualified},
  {XMLElement[{http://www.w3.org/2001/XMLSchema, element},
    {name → case, type → xs:string}, {}], XMLElement[
    {http://www.w3.org/2001/XMLSchema, element}, {name → usreport},
    {XMLElement[{http://www.w3.org/2001/XMLSchema, complexType}, {},
      {XMLElement[{http://www.w3.org/2001/XMLSchema, sequence}, {},
        {XMLElement[{http://www.w3.org/2001/XMLSchema, element},
          {ref → case, maxOccurs → unbounded}, {}]}]}]}]}], {}]
```

Regular expressions [10] were then defined to capture the various forms of citation contained in the simple XML files. This task was complicated by (a) the Supreme Court's use, during its earlier days, of different forms of citation and (b) the original database creator's occasional inconsistency in marking up citations with his own identifying flags. Two examples of these regular expressions follow. The group of all these regular expressions, along with the XML elements into which each was to be transformed, was set forth in a list called `report2fullxmlrules`.

Extraction of citation information from the database tests the capabilities of regular expressions. In general, a citation to a Supreme Court opinion takes the following form: a three digit volume number, U.S., an optional alternative volume designation for older versions, and a reference to the page of the volume on which the opinion begins (the base page). Thus, "326 U.S. 434" or "68 U.S. (5 How.) 116" would both be syntactically correct citations. (For a basic exposition of U.S. citation forms, see en.wikipedia.org/wiki/Special:Search/Court_citation.)

Simple extraction of such information from the database via `StringCases` will prove overinclusive and underinclusive, however. It will prove overinclusive because some citations of this form ("@cert. denied,@ 535 U.S. 1091") are in fact citations to something known as "denials of certiorari," which are generally terse explanations that the Supreme Court will not hear a particular case. These denials of certiorari are themselves understandably not included in the underlying database, hence permitting a link to them would cause serious problems of closure. Additionally page break information within the opinions has been denoted by encapsulating something that looks like a Supreme Court citation with the tag Á. The compiler of the database generally (though not always) added some markup near such citations so that we see things like "@ªAnderson v. Celebrezze,@ 460 U.S. 780 <L=|460 U.S. 794|>794-95." This makes the process overinclusive (and creates the possibility of double counting). The regular expression that follows captures this sort of notation. The process is underinclusive because in older Supreme Court opinions the citations encapsulate the name of the private contractor publishing the works (sometimes abbreviated) between the volume and page number.

The following principles were therefore implemented in the extraction of citation information. If a marked-up link was found, it was extracted. If a Supreme Court citation was found that was not preceded by a character suggesting a denial of certiorari and not followed by a marked-up link string, it was extracted. And, finally, legacy citation strings were extracted.

```
supremecourtcitation = "(?:(\\d{1,3})\\s*(?:[Uu][[:punct:]][
    Ss][[:punct:]])\\s*(?:\\(.{0,20}\\))?\\s*(\\d{1,4}))";

legacycitation =
  "&lt;\\|(\\d{1,2})\\s+((?:Wall\\.)|(?:Black)|(?:How\\.)|(?:Pet
    \\.)|(?:Wheat\\.)|(?:Cranch)|(?:Dall\\.))\\s+(\\d{1,4})\\|&gt;";
```

The command `report2fullxml` then used `report2fullxmrule` to translate each of the simple XML files into a more complex XML file matching a schema set forth in allcases290.xml (see Additional Material).

```
report2fullxml[s_String] :=
 ImportString[StringReplace[s, report2fullxmlrules], "SymbolicXML"]
   XMLElement["segment", attribs_List, body_List] :>
  XMLElement["segment", attribs, Cases[body, _XMLElement]]
```

A typical physically bound volume of the United States Reports, which will cover all or a portion of a Supreme Court term, might thus now be reduced to something like this (in which the ellipsis represents text deleted to permit a compact representation).

```
<usreport vol='134'> <case> <segment segmentname='datasegment'> <
  ft>Hans v. Louisiana, 134 U.S. 1 (1890)</ft> <cg> <cgi>134 U.
  S. 1</cgi> <cgi>contract impairment</cgi> <cgi>contracts</
  cgi> <cgi>eleventh amendment</cgi> <cgi>federal question
  jurisdiction</cgi> <cgi>immunity</cgi> <cgi>jurisdiction</
  cgi> <cgi>sovereign immunity</cgi> <cgi>states</cgi> </cg> <
  ct>Hans v. Louisiana, 134 U.S. 1 (1890)</ct> <yendata>Hans v.
  Louisiana</yendata> <yendata>No. 4</yendata> <yendata>Argued
  and submitted January 22, 1890</yendata> <yendata>Decided March
  3, 1890</yendata> <yendata>134 U.S. 1</yendata> </segment> <
  segment segmentname='syll* '> <cite> <vol>2</vol> <page>419</
  page> </cite> </segment> <segment segmentname\n...\n<segment
  segmentname='leadop*blatchford* '> <cite> <vol>45</vol> <
  page>503</page> </cite> <cite> <vol>67</vol> <page>715</
  page> </cite> <cite> <vol>74</vol> <page>299</page> </cite> <
  cite> <vol>88</vol> <page>178</page> </cite> <cite> <vol>
  88</vol> <page>183</page> </cite> <cite> <vol>88</vol> <
  page>616</page> </cite> <cite> <vol>91</vol> <page>587</
  page> </cite> <cite> <vol>103</vol> <page>651</page> </
  cite> <cite> <vol>109</vol> <page>522</page> </cite> <
  cite> <vol>127</vol> <page>589</page> </cite> <cite> <
  vol>98</vol> <page>61</page> </cite> <cite> <vol>111</
  vol> <page>505</page> </cite> </segment> </case></usreport>
```

From XML to Edges

The final major step in preparing the network was to convert the citation information contained in the complex XML files into the directed "edges" (arcs) of a directed graph.

This step likewise had its own subtleties. First, to varying degrees over the history of the Supreme Court, opinions have been adorned with a syllabus prepared by a reporter that attempts to summarize the case and may contain citation information but that is not considered part of the canonical case report. Citations contained in the syllabus segment of the opinion (along with a "datasegment" we created to hold "metadata" on the opinion) thus had to be discounted. The `XML2cites` function illustrates how SymbolicXML was manipulated by the `Cases` command and pattern recognition to permit this extraction of only pertinent citation information.

```
XML2cites[x_] :=
 Map[Cases[DeleteCases[#, XMLElement["segment", {Alternatives[
        "segmentname" → "datasegment", "segmentname" → "syll*",
        "segmentname" → "syll* "], ___}, ___], {2}],
    XMLElement["cite", {}, {XMLElement["vol", {}, {vol_}],
       XMLElement["page", {}, {page_}]}] :>
     FromDigits[ToExpression /@ {vol, page}, 10000], {4}] &,
  Cases[x, XMLElement["case", ___], {3}]]
```

Second, an authoritative definition of each case had to be developed because these cases would serve as the nodes of the network. The `XML2caseid` function again shows *Mathematica*'s use of SymbolicXML and pattern matching to extract the citation string. The `caseid2int` function shows *Mathematica*'s use of regular expressions to convert this string into a (hopefully unique) seven-digit number.

```
XML2caseid[x_] := Map[
  First[First[Cases[#, XMLElement["ft", {}, cn : {___}] :> cn, {2}]]] &,
  Cases[x, XMLElement["segment",
    {"segmentname" → "datasegment", ___}, ___], {5}]]

caseid2int[s_String] :=
 First[StringCases[s, RegularExpression[supremecourtcitation] :>
    FromDigits[ToExpression /@ {"$1", "$2"}, 10000]]]
```

We can now write a function `XML2Arcs` that, with the help the auxiliary `XML2int` function, takes a SymbolicXML representation of a complex XML file and creates a list of citations. The citations take the form of a unique representation of each case to a page of some other case.

```
XML2int[x_] := caseid2int /@ XML2caseid[x]

XML2Arcs[x_] :=
 MapThread[Thread[{##}] &, Through[{XML2int, XML2cites}[x]]]
```

The following code illustrates how rapidly `XML2Arcs` works, as well as its output.

The variable `xmldatabasedirectory` is used throughout the notebook. Users of this notebook will have to reset this variable in order to accommodate their own systems. It should also be noted that key expressions generated in this notebook

(such as arcs, fullcasenames, and so on) that may take time to regenerate may be written to a file using the `Put` command.

```
xmldatabasedirectory =
  ToFileName[{$HomeDirectory, "Databases", "USSC-ISI", "XML"}];

allxmlfiles = FileNames["*.xml", {xmldatabasedirectory}];

Timing[Short[XML2Arcs[Import[
    xmldatabasedirectory <> "\\allcases381.xml", "SymbolicXML"]], 3]]

{0.047 Second,
 {{{3810001, 3790809}, {3810001, 3120248}, {3810001, 3570125},
   {3810001, 3780505}, {3810001, 3120246}, {3810001, 3100354},
   {3810001, 3810017}, {3810001, 3070171},
   <<72>>, {3810001, 3670130}, {3810001, 3340785},
   {3810001, 3570128}, {3810001, 3810011}, {3810001, 2880294},
   {3810001, 3810014}, {3810001, 3760410}}, <<31>>}}

Timing[Short[startingpages = XML2int[Import[#]] & /@ allxmlfiles, 3]]

{32.656 Second, {{}, {20401, 20402, 20409, 20415, 20419},
  <<539>>, {5420001, 5420055, 5420074, 5420088, 5420129,
   5420155, 5420177, 5420200, 5420225, 5420241}}}

Timing[Short[arcs =
   Flatten[DeleteCases[(XML2Arcs[Import[#]] & /@ allxmlfiles), {}], 2]]]

{37.813 Second,
 {{140304, 100286}, {170518, 100087}, {170518, 130043}, <<493141>>,
  {5420241, 4740146}, {5420241, 4140146}, {5420241, 5300437}}}

Timing[Short[fullcasenames =
   Flatten[Map[XML2caseid[Import[#, "XML"]] &, allxmlfiles]], 3]]

{32.657 Second, {Oswald v. New York, 2 U.S. 401,
  Georgia v. Brailsford, 2 U.S. 402, Hayburn's Case, 2 U.S. 409,
  <<27031>>, Aetna Health Inc. v. Davila, 542 U.S. 200 (2004),
  Pliler v. Ford, No. 542 U.S. 225 (2004),
  Intel Corp. v. Advanced Micro Devices, Inc., 542 U.S. 241 (2004)}}

Timing[Short[casenames =
   Map[StringReplace[#, RegularExpression["(.+),[^,]+"] → "$1"] &,
    fullcasenames], 3]]

{0.282 Second, {Oswald v. New York, Georgia v. Brailsford,
  Hayburn's Case, Georgia v. Brailsford, Chisholm v. Georgia,
  <<27028>>, Hiibel v. Sixth Judicial District Court of Nevada,
  Aetna Health Inc. v. Davila, Pliler v. Ford,
  Intel Corp. v. Advanced Micro Devices, Inc.}}
```

The `XML2Arcs` function was mapped over all complex XML files in the database to create a list of approximately 500,000 citations, which was stored in a file.

The third subtlety is that the citation information contained in the underlying case reports, the complex XML files derived from them, and the edges derived from `XML2Arcs` point to *pages* of various volumes of the United States Reports.

But pages are not the desired nodes of the relevant graph. Rather, the desired nodes are the cases themselves. Thus, a mapping had to be developed between a page citation and a citation to one of these cases. The mapping needs to be fast due to the huge volume of citations that may need to be processed.

The fastest method we discovered (four orders of magnitude swifter than any alternative, including use of the `RangeLists` command) was inspired by a note in the advanced documentation for regular expressions, which states that there are cases where it is advantageous to translate a normal expression-matching problem to a string-matching problem. The concept, illustrated here with a very simple example, was to add a delimiter "a" to the pages of all the starting pages of cases and a delimiter "b" to all citations. We then sort the joinder of the thus-delimited starting and citation pages, convert each item in the resulting two-dimensional list to a string, flatten the list, and perform a `StringJoin` operation.

```
samplestartingpages =
 Thread[{Table[Random[Integer, {0, 5500000}], {4}], a}]
```

```
{{2867461, a}, {4078658, a}, {18459, a}, {2152337, a}}
```

```
samplecitationpages = Thread[{Table[Random[Integer,
     {Min[First /@ samplestartingpages], 5500000}], {8}], b}]
```

```
{{2470135, b}, {3980369, b}, {5060642, b}, {3793785, b},
 {4687066, b}, {5472645, b}, {1919039, b}, {756349, b}}
```

```
InputForm[sj = StringJoin @@ Flatten[Map[ToString,
     Sort[Join[samplestartingpages, samplecitationpages]], {2}]]]
```

```
"18459a756349b1919039b2152337a2470135b2867461a3793785b3980369b4078658a46
87066b5060642b5472645b"
```

We then find all instances that match the regular expression consisting of numbers followed by a, followed by anything except a, followed by numbers, and followed by b. These represent, in a fashion, all the citations that cite to the starting page of a case.

```
sc = StringCases[sj, RegularExpression["\\d+a[^a]*\\d+b"]]
```

```
{18459a756349b1919039b, 2152337a2470135b,
 2867461a3793785b3980369b, 4078658a4687066b5060642b5472645b}
```

With the `StringSplit` command, it then becomes a relatively simple matter to map each citation page to its base page.

```
Map[ToExpression, Flatten[Map[Thread[Rule[Rest[#], First[#]]] &,
   Map[StringSplit[#, "a" | "b"] &, sc]], {2}]
```

```
{756349 → 18459, 1919039 → 18459,
 2470135 → 2152337, 3793785 → 2867461, 3980369 → 2867461,
 4687066 → 4078658, 5060642 → 4078658, 5472645 → 4078658}
```

This can all be combined into a single `page2startingpage` function.

```
page2startingpage[startingpages_, citationpages_] :=
 Map[ToExpression, Flatten[Map[
    Thread[Rule[Rest[#], First[#]]] &,
    Map[StringSplit[#, "a" | "b"] &, StringCases[StringJoin @@
       Flatten[Map[ToString, Sort[Join[Thread[{startingpages, a}],
           Thread[{citationpages, b}]]], {2}]],
      RegularExpression["\\d+a[^a]*\\d+b"]]]], {2}]
```

The stunning speed of this approach is demonstrated on our problem, which involves 27,037 cases and 493,147 arcs.

```
Timing[realarcs = Module[{p2sprules, sp2rules, flatstartingpages},
    flatstartingpages = Flatten[startingpages];
    sp2rules =
     Dispatch[MapIndexed[#1 → #2[[1]] &, flatstartingpages]];
    p2sprules = page2startingpage[flatstartingpages,
      Union[Flatten[arcs]]];
    arcs /. Dispatch[p2sprules] /. Dispatch[
      MapIndexed[#1 → #2[[1]] &, flatstartingpages]];]
```

```
{8.032 Second, Null}
```

Alternatives of the Full Database

The bulk of the work in preparing the network for analysis is now complete. There remain, however, a few minor tasks. A graph produced from this list of citations would be multiply connected. However, it is often useful and faster to work with a simple network. We thus eliminate the multiple edges. The result is a list containing a little over a quarter million citations.

```
DeleteRepetitions[X_] :=
 Block[{t}, t[n_] := (t[n] = Sequence[]; n); t /@ X]
```

```
Length[uniquerealarcs = DeleteRepetitions[realarcs]]
```

```
258819
```

A second set of issues involves mistakes in the database. Neither the Supreme Court nor the individuals creating the original database are perfect. They occasionally transpose digits or simply miscite other opinions. While no algorithm is likely to be capable of spotting (let alone fixing) all these errors, we can at least spot instances in which a case purports to cite a case written well in the future. Given the inordinate amount of labor it would take to correct these erroneous citations, and given my judgment that the retention of unreformed erroneous citations would be worse than their deletion, a small program was written to delete them from the database.

```
Timing[Length[cleanedarcs =
   Select[uniquerealarcs, Min[1.1 * #[[1]], #[[1]] + 30] > #[[2]] &]]]
```

```
{1.219 Second, 258602}
```

A third set of issues involves bidirectional links. On occasion, judicial opinions issued closely in time cite each other. While these bidirectional links are permissible in a directed graph, they are impermissible in a simple undirected graph.

Since, on occasion, it is the undirected graph that needs to be studied, these bidirectional links had to be flagged and, as appropriate, removed.

```
Timing[Short[
  finalarcs = With[{strange = Select[cleanedarcs, #[[2]] > #[[1]] &],
     normal = Select[cleanedarcs, #[[2]] <= #[[1]] &]},
    Complement[cleanedarcs, Intersection[
      normal, Reverse /@ strange]]]]]
```

```
{1.203 Second, {{476, 266}, {597, 245}, {597, 410},
  {597, 436}, {765, 203}, <<258038>>, {27037, 26152},
  {27037, 26685}, {27037, 26719}, {27037, 26811}}}
```

■ Using *Mathematica* as a Communications Hub with JUNG, Pajek, and GraphML

Networks based on legal source material tend to be large. A court may issue thousands of opinions with many more thousands of links. Indeed, Smith reports in his work that a leading database of federal and state cases contains more than four million cases. A statute or legal code may contain thousands of nodes such as its sections, paragraphs, and subparagraphs. Often such codes contain thousands more cross references. It thus becomes important to have a toolkit available capable of handling large networks. This section of the article describes the building of that toolkit. Again, those with a predominant focus on the implications of all this for an understanding of the law may wish to forge swiftly ahead to the next section.

□ The Problem with Exclusive Use of *Mathematica*

Much as we might like to use *Mathematica* as the primary vehicle for analysis on such large network problems, that desire is challenged by the difficulty of *Combinatorica* [11], its leading network analysis package, in ubiquitously scaling to networks of this size. While some of the routines contained in the *Combinatorica* package designed for analysis of networks perform adequately, others do not scale well to large graphs; still others (including simple functions such as `InDegrees`) simply crash the kernel. Moreover, although *Combinatorica* indeed contains a rich library of network analysis functions, it lacks some of the more recently developed algorithms in the field, including those for computing various importance measures.

□ The Packages

We therefore created a set of three packages (see the GraphTheory folder in Additional Material) that permit *Mathematica* to serve as the hub of a network connecting its own graph structures to three alternative methodologies: (1) *JUNG*`, a lengthy package that interacts with JUNG, a software library that provides a common and extensible language for the modeling, analysis, and visualization of data that can be represented as a graph or network; (2) *Pajek*`

("spider"), a package that exports to and imports from the simple CSV-like file format used by Pajek, a Windows-based public domain program (vlado.fmf.uni-lj.si/pub/networks/pajek) that appears to have gained some traction among network analysts; and (3) *GraphML`*, a package that exports to and imports from files written in a species of XML known as GraphML, which is used by a variety of programs.

Figure 1 illustrates the functionality provided by these three packages. *JUNG`* is labeled in blue, *GraphML`* is labeled in green, and *Pajek`* is labeled in red.

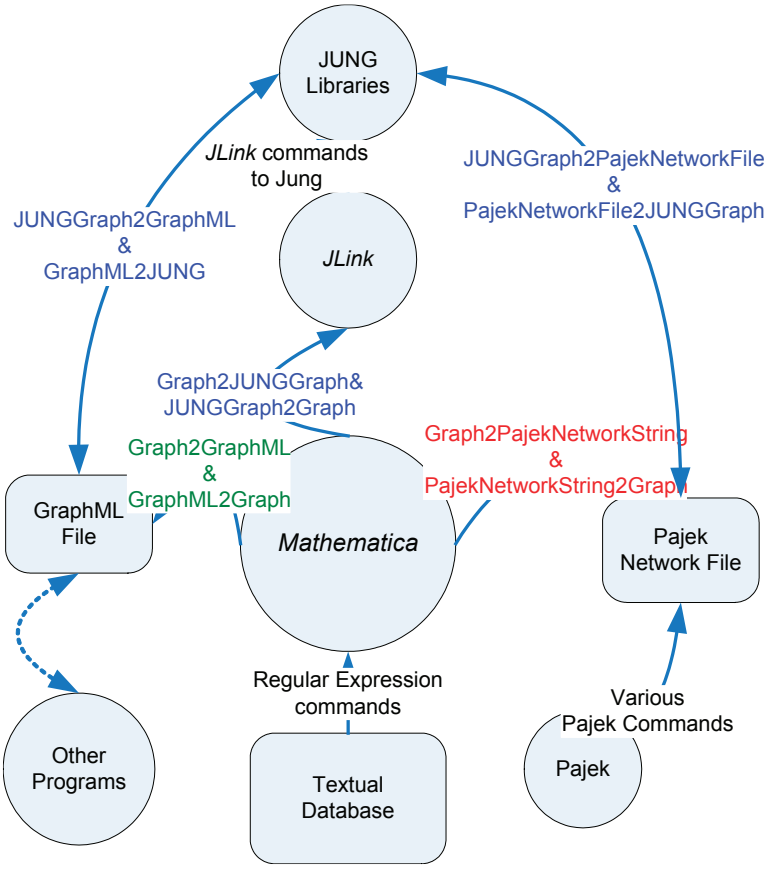


Figure 1. The functionality of the *JUNG`*, *GraphML`*, and *Pajek`* packages.

JUNG

The *JUNG`* package provides access to the JUNG libraries. This package requires the user to have previously downloaded the relevant Java archives (JAR files) from jung.sourceforge.net and placed them in a local directory. It also requires the user to download several needed dependent Java archives, such as XML parsers, according to the directions provided at that site. With these libraries in place, the *JUNG`* package leverages *J/Link* to permit access to JUNG algorithms and data structures from within *Mathematica*. In particular, it permits

the *Mathematica* user to convert a *Combinatorica* graph object into a JUNG graph object suitable for analysis using various JUNG algorithms—all without leaving the *Mathematica* environment. The user can likewise take a JUNG graph object and convert it back into a *Combinatorica* graph object.

Here are some illustrations of the functionality provided by the package. First, we load *JUNG`*, which in turn loads various Java classes and various *Mathematica* packages such as *Combinatorica*. (Readers may need to alter `$Path` to accommodate their own placement of the relevant packages.)

```
SeedRandom[102257]

With[{jungPath = ToFileName[{$InstallationDirectory,
      "AddOns", "Applications", "GraphTheory"}]},
  If[Not[MemberQ[$Path, jungPath]], AppendTo[$Path, jungPath]]];

Needs["JUNG`"]
```

Second, we create an edge-weighted and vertex-labeled random graph using basic *Combinatorica* commands.

```
rg = SetEdgeWeights[SetVertexLabels[
    RandomGraph[12, 0.25, Type → Directed], CharacterRange["A", "L"]],
   WeightingFunction → RandomInteger, WeightRange → {1, 5}];
```

Then we convert the *Combinatorica* graph into a JUNG graph object.

```
Timing[jg = Graph2JUNGGraph[rg]]

{0.312487 Second,
 «JavaObject[edu.uci.ics.jung.graph.impl.DirectedSparseGraph] »}
```

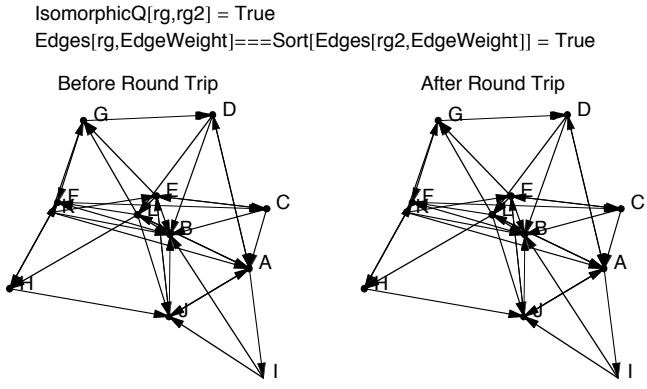
As the following code shows, the converted JUNG graph object has the same number of vertices and edges, the same edges and edge weights, and the same vertex labels as the *Combinatorica* graph from which it derives.

```
{V[rg] === JUNGV[jg], M[rg] === JUNGM[jg],
 Sort[JUNGEdges[jg, EdgeWeight]] === Edges[rg, EdgeWeight],
 JUNGGetVertexLabels[jg] === GetVertexLabels[rg]}

{True, True, True, True}
```

We can convert the JUNG graph back into a *Combinatorica* graph and confirm that the result of this round trip is visually the same and isomorphic to the original graph.

```
Show[GraphicsArray[{ShowGraph[SpringEmbedding[rg],
     TextStyle → {FontFamily → "Helvetica", FontSize → 10},
     PlotLabel → "Before Round Trip", DisplayFunction → Identity],
    ShowGraph[SpringEmbedding[rg2 = JUNGGraph2Graph[jg]],
     PlotLabel → "After Round Trip", DisplayFunction → Identity,
     TextStyle → {FontFamily → "Helvetica", FontSize → 10}]},
   PlotLabel → "IsomorphicQ[rg,rg2] = " <>
     ToString[IsomorphicQ[rg, rg2]] <>
     "\nEdges[rg,EdgeWeight]===Sort[Edges[rg2,EdgeWeight]] = " <>
     ToString[Edges[rg, EdgeWeight] === Sort[Edges[rg2, EdgeWeight]]],
   TextStyle → {FontFamily → "Helvetica", FontSize → 10}]];
```



Once the graph has been converted to a JUNG graph object, we can use *J/Link* to access the considerable JUNG library of graph algorithms on a more custom basis. Here, for example, we show how we might compute the "betweenness centrality" of the various nodes of the graph—a topic discussed further in The Most Between Cases section.

```
betweenness = JavaNew[
  "edu.uci.ics.jung.algorithms.importance.BetweennessCentrality",
  jg, True, False]; betweenness@setMaximumIterations[50];
betweenness@setNormalizeRankings[True];
betweenness@evaluate[]

Outer[#1@#2 &, betweenness@getRankings[]@toArray[],
  {originalPos, rankScore}] /.
 {vertexno_, score_} :→ {JUNGGetVertexLabels[jg][[vertexno]], score}
```

```
{{A, 26.3333}, {C, 18.6667}, {J, 17.9167},
 {E, 16.3333}, {L, 16.25}, {F, 15.5}, {D, 6.25}, {B, 4.83333},
 {G, 2.33333}, {H, 0.333333}, {I, 0.25}, {K, 0.}}
```

Communicating with Pajek and GraphML

The tools developed in preparation for this article also permit us to get either JUNG or *Combinatorica* graphs into and out of *Mathematica* in ways that facilitate their analysis by other specialized programs. *Pajek*` uses basic *Mathematica* string manipulation commands to translate *Combinatorica* graphs into Pajek network

strings, which can then be exported in the conventional fashion. Pajek network files can likewise be imported back into *Mathematica* as *Combinatorica* graph objects.

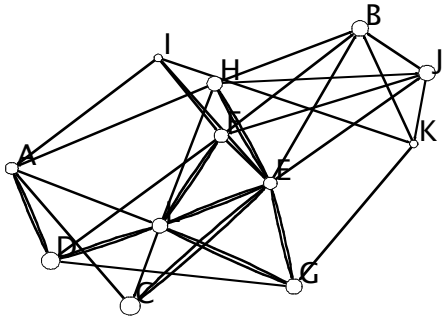
```
Needs["Pajek`"]

Export[ToFileName[{$HomeDirectory, "Temp"}, "pfn1.net"],
  Graph2PajekNetworkString[rg], "Text"];

JUNGGraph2PajekNetworkFile[
 ToFileName[{$HomeDirectory, "Temp"}, "pfn2.net"], jg]
```

The following graphic illustrates some of the possibilities for analysis facilitated by this link to Pajek. Here, for example, we show a diagram produced by Pajek (and exported in encapsulated PostScript). Pajek has taken a graph exported by *Mathematica*, embedded it using the Fruchterman–Reingold algorithm and labeled the vertices.

```
Show[Import[ToFileName[{$HomeDirectory, "Temp"}, "pnf2.eps"]],
 PlotRange → {{0, 1200}, {0, 800}}]
```



The *GraphML`* package uses *Mathematica*'s SymbolicXML capabilities to permit users to export *Combinatorica* graphs into a species of XML known as GraphML, which is used by some network analysis and visualization programs as a storage medium. GraphML files can likewise be imported back into *Mathematica* as *Combinatorica* graphs.

```
Needs["GraphML`"]

Shallow[Graph2SymbolicXML[rg], 6]

XMLObject[Document][
 {XMLObject[Declaration][Version → 1.0, Encoding → UTF-8],
  XMLObject[Comment][ This file was written by Mathematica 5.1]},
 XMLElement[graphml,
  {{<<2>>} → http://graphml.graphdrawing.org/xmlns,
   {<<2>>} → http://www.w3.org/2001/XMLSchema-instance,
   {<<2>>} → http://graphml.graphdrawing.org/xmlns http://
      graphml.graphdrawing.org/xmlns/1.0/graphml.xsd},
  {XMLElement[graph, {<<1>>}, {<<53>>}]}], {}]
```

The following code shows that basic information is not lost in a round trip from a *Mathematica* graph to a SymbolicXML representation.

```
IsomorphicQ[SymbolicXML2Graph[Graph2SymbolicXML[rg]], rg]
```

```
True
```

■ Characteristics of the Supreme Court Network

The variety of tools now in place permits a multi-pronged rudimentary probe of the structure of the Supreme Court network. The efforts shown here are not, it should be emphasized, anything approaching an exhaustive analysis, but simply a proof of concept along with an exposition of several initial findings.

We can create a Pajek network file that represents our database.

```
pajekdatadirectory =
  ToFileName[{$HomeDirectory, "Databases", "Pajek"}];

Export[ToFileName[pajekdatadirectory, "sctg.net"],
 StringJoin @@ {"*Vertices 27037\n",
   MapIndexed[ToString[#2[[1]]] <> " \"" <> #1 <> "\"\n" &, casenames],
   "*Arcs\n", Map[ToString[#[[1]]] <> " " <> ToString[#[[2]]] <> " 1\n" &,
    finalarcs]}, "Text"]
```

Assuming the particular machine on which this process is implemented can accommodate a large Java heap, we can also read it back in as a JUNG graph object. The process takes less than a minute.

```
j = PajekNetworkFile2JUNGGraph[
  ToFileName[pajekdatadirectory, "sctg.net"]]
```

```
«JavaObject[edu.uci.ics.jung.graph.impl.SparseGraph] »
```

We can also swiftly read it in as a *Mathematica* graph object.

```
Timing[g = With[{fop = FromOrderedPairs[finalarcs, Type → Undirected]},
   Graph[fop[[1]], MapThread[Append[#1, VertexLabel → #2] &,
     {fop[[2]], fullcasenames}], EdgeDirection → False]]]
```

```
{0.5 Second, -Graph:<258047, 27037, Undirected>-}
```

□ Density

The density of a network is the number of edges it contains divided by the number of edges a completely connected network could contain. We calculate density using both JUNG and *Mathematica*. The Supreme Court network has a very low density of about 0.0007, reflecting the fact that only 258,047 of the theoretically possible 365 million possible citations exist.

$$\left\{N\left[\frac{\text{JUNGM[j]}}{\text{JUNGV[j] (JUNGV[j] - 1) / 2}}\right], N\left[\frac{\text{M[g]}}{\text{V[g] (V[g] - 1) / 2}}\right]\right\}$$

```
{0.000706038, 0.000706038}
```

□ Degree Distribution

The degree of a node in a network is the number of connections it contains to other nodes in the network. The distribution of degrees among the nodes in a network is often indicative of how information passes throughout the network. The following code shows how *Mathematica* and JUNG via *J/Link* can combine to generate this information.

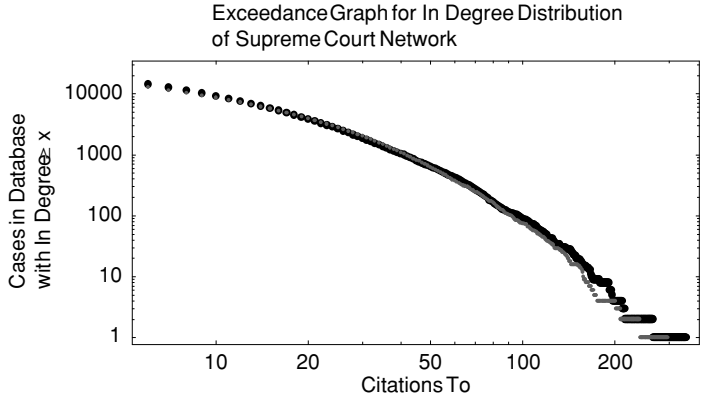
```
networkdegreedistributions =
  With[{index = Indexer`newIndexer[j, 1]}, Table[
    Through[{#@outDegree[] &, #@inDegree[] &}[index@getVertex[i]]],
    {i, 1, JUNGV[j]}]];
```

The following graphic shows the distribution of degrees in our Supreme Court database using in degree exceedance: the x-axis shows the number of citations to a case and the y-axis shows the number of cases for which the number of citations to it equals or exceeds the value on the x-axis. (This method of analysis avoids binning problems that potentially plague alternative methods.) The larger black dots show the results for our Supreme Court database.

This plot, with a significant upward bowing, does not correspond with the downward sloping straight line that we would expect to see were the distribution taken from a power distribution, such as that generated by a scale-free network. Neither does it correspond well to what we would expect to see were the distribution taken from a normal distribution, which is roughly what we would see if the distribution were generated by a truly random graph.

However, one distribution that does match quite well the sort of exceedance found in the Supreme Court database is a Weibull Distribution ($e^{-\left(\frac{x}{\beta}\right)^{\alpha}} x^{\alpha-1} \alpha \beta^{-\alpha}$), which measures the lifetime of an object before it fails. The little gray points on the plot show what we would expect to see resulting from a Weibull distribution. As we can see, it is a pretty close match, suggesting that perhaps the number of citations to a case is simply a function of its intellectual longevity and that cases fail much in the same manner as mechanical parts.

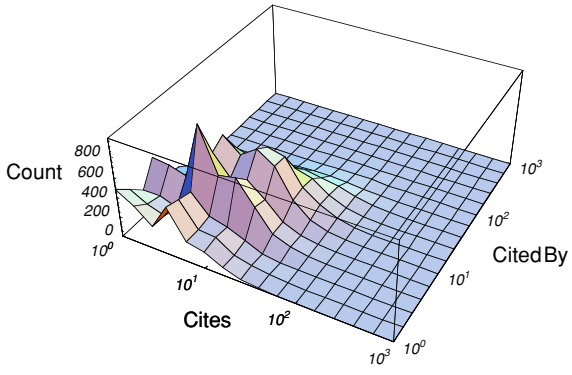
```
exceedance = With[{n = Last /@ networkdegreedistributions},
   Table[Count[n, _?(# ≥ i &)], {i, 0, 340}]];
```



```
Needs["Statistics`DataManipulation`"]
```

We can also look at a correlation between in degree and out degree in the Supreme Court network. We see modest correlation, suggesting that the number of cases an opinion cites may be a proxy for the importance of the case, as well as for the territory covered by the case, both of which would increase the probability that a subsequent case would cite it.

```
ListPlot3D[BinCounts[Map[Log[10, #] &,
     DeleteCases[networkdegreedistributions, {___, 0, ___}], {2}] // N,
   {-0.0001, 3, 0.2}, {-0.001, 3, 0.2}], PlotRange → {0, 970},
  TextStyle → {FontFamily → "Helvetica-Oblique", FontSize → 8},
  AxesLabel → {StyleForm["Cites",
     FontFamily → "Helvetica", FontSize → 10],
    StyleForm["Cited By", FontFamily → "Helvetica", FontSize → 10],
    StyleForm["Count  ", FontFamily → "Helvetica", FontSize → 10]},
  MeshRange → {{0, 3}, {0, 3}},
  Ticks → {{#, HoldForm[10^#]} & /@ Range[0, 3],
    {#, HoldForm[10^#]} & /@ Range[0, 3], Automatic}];
```



□ Clustering

We can use JUNG conveniently to figure out how clustered the Supreme Court network is. The following code, which takes upwards of a minute to execute, creates a set of clusters.

```
wcc = JavaNew["edu.uci.ics.jung.algorithms.
     cluster.WeakComponentClusterer"]@extract[j]
```

```
«JavaObject[edu.uci.ics.jung.algorithms.cluster.VertexClusterSet] »
```

We can sort the clusters by size from largest to smallest and then get the size of the largest cluster, which turns out to be 25,654. Thus, virtually all of the cases in the Supreme Court database are connected to each other. The remainder lie in tiny subnetworks.

```
wcc@sort[]; wcc@getCluster[0]@size[]
```

```
25654
```

We can also determine the *k*-radius clustering coefficient for the undirected version of the graph. We do so by taking a sample of vertices in the network and dividing, for each vertex, the number of edges in the subgraph induced on a *k*-radius neighborhood of that vertex with the number of edges ($v(v-1)/2$, where v is the number of vertices in that neighborhood) that would exist in a complete graph impressed on that *k*-radius neighborhood. We then take the average of these quotients to approximate the clustering coefficient for the entire graph. All of this can be done within *Mathematica*, albeit somewhat slowly.

When these experiments are done, they tend to yield clustering coefficients that have medians and means in the 0.3 to 0.4 range. This suggests that the scope of many cases is sufficiently narrow and the court's research ability is sufficiently high that most related cases are able to "communicate" with each other. And this fact may be partly responsible for (or at least reflective of) what is generally regarded as the complexity of much Supreme Court jurisprudence.

```
clusteringcoefficient[g_Graph, vertex_Integer, k_Integer] :=
 Module[{isg = InduceSubgraph[g, Neighborhood[g, vertex, k]]},
  v = V[isg];
  If[v > 1, 2 M[isg]/(V[isg] V[isg] - 1), {}]]
```

```
Short[coefficients =
  DeleteCases[With[{r = RandomKSubset[Range[V[g]], 500]}, Thread[
     {r, Map[clusteringcoefficient[g, #, 1] &, r]}]], {_Integer, {}}]]
```

$\{\{81, \frac{2}{5}\}, \{174, \frac{54}{143}\}, \{212, \frac{8}{15}\}, \{365, \frac{1}{2}\}, \{494, \frac{2}{3}\}, \ll 453 \gg,$
$\{26612, \frac{214}{783}\}, \{26646, \frac{86}{195}\}, \{26720, \frac{61}{220}\}, \{26907, \frac{2}{5}\}, \{27008, \frac{2}{3}\}\}$

```
N[Through[{Mean, Median}[Last /@ coefficients]]]
```

```
{0.36035, 0.325}
```

We can also test whether clustering has changed over the years. The linear regression shows that the sequence number of the case (a proxy for age) accounts for only about 1% of the variation in clustering coefficients. And while, technically, sequence number has a statistically significant negative coefficient, the effect is awfully small. It is fair to say, then, that clustering has not changed dramatically over the years.

```
Needs["Statistics`LinearRegression`"]
```

```
r = Regress[coefficients, {1, caseseqno}, caseseqno]
```

RSquared → 0.0124372

	Estimate	SE	TStat	PValue
1	$4.\times10^{-1}$	$2.\times10^{-2}$	$2.\times10^{1}$	0.
caseseqno	-3.1×10^{-6}	1.3×10^{-6}	-2.4	1.6×10^{-2}

□ Node Centrality Measures

We can use our set of tools to study the importance of various Supreme Court decisions, as determined simply from the connectivity information in a way unaffected (and unbiased) by the contents of the decisions.

The Most Cited Cases

One measure of importance is simply the number of times a case has been cited by other Supreme Court cases. We can use a mix of *Mathematica* and JUNG via *J/Link* to conduct this analysis. The resulting list of cases is largely familiar to most scholars of American constitutional law. The first two, *McCulloch v. Maryland* and *Gibbons v. Ogden*, by way of example, establish the federal government's broad power over commerce. The third and fourth involve the rights of accused criminals. And the fifth is a critical case involving allocation of power in the American government among its executive, legislative, and judicial branches. The only one of these most-cited cases decided in the last 40 years is *Chevron, U.S.A. v. NRDC*, a case allocating power between the executive and judicial branches over interpretation of statutes.

```
mostcited =
  Sort[Thread[{casenames, Last /@ networkdegreedistributions}],
   #1[[2]] > #2[[2]] &];
```

Case	Cited By
McCulloch v. Maryland	341
Gibbons v. Ogden	265
Boyd v. United States	214
Miranda v. Arizona	210
Marbury v. Madison	195
Erie Railroad Co. v. Tompkins	194
Ashwander v. Tennessee Valley Auth.	190
Osborn v. Bank of the United States	190
Cantwell v. Connecticut	177
Cohens v. Virginia	168
NAACP v. Button	166
Ex Parte Young	165
Yick Wo v. Hopkins	165
Gideon v. Wainwright	163
NAACP v. Patterson	160
New York Times Co. v. Sullivan	157
Chevron U.S.A., Inc. v. NRDC	155
Brown v. Board of Education of Topeka	154
Johnson v. Zerbst	153
Mapp v. Ohio	150

The Most Connected Cases (the Kevin Bacon Game Applied to Law)

We can use Pajek (with Pajek network files created by *Mathematica*) to play what is sometimes called the Kevin Bacon game on the Supreme Court network. (The original Kevin Bacon game is described more fully at en.wikipedia.org/wiki/

Special:Search/Six_Degrees_of_Kevin_Bacon). To do this, we compute the "geodesic" between each node (case) and each other node to which it is connected. By geodesic, network theorists mean the shortest path between two nodes in the same way that on earth a geodesic is the shortest path between two points on the surface. There are a couple of ways to measure closeness, but a typical one is to compute the total length of geodesics between each node and each other node to which it is connected and then to divide that sum by the number of vertices in the network. The following table, which is the result of Pajek churning through the data for close to an hour, shows the most central Supreme Court decisions among the large group of weakly connected cases. The network was converted into an undirected one prior to the analysis. Again, these tend to be cases that are well known. The only decision from the past 40 years on the list is *Seminole Tribe of Florida v. Florida*, a 1996 decision limiting the power to bring lawsuits against state governments.

Case or Citation	Average Distance
McCulloch v. Maryland	2.8660
Ashwander v. Tennessee Valley Auth.	2.9010
Erie Railroad Co. v. Tompkins	2.9170
Baker v. Carr	2.9270
Crowell v. Benson	2.9280
Cohens v. Virginia	2.9580
Marbury v. Madison	2.9690
Burnet v. Coronado Oil & Gas Co.	2.9750
Glidden Co. v. Zdanok	3.0020
Gibbons v. Ogden	3.0150
Monroe v. Pape	3.0190
Yakus v. United States	3.0210
Ex Parte Young	3.0260
Joint Anti - Fascist Refugee Committee v. McGrath	3.0280
Seminole Tribe of Florida v. Florida	3.0280
Osborn v. Bank of the United States	3.0380
Home Building & Loan Assn. v. Blaisdell	3.0440
National Mut. Ins. Co. v. Tidewater Tran	3.0450
Fay v. Noia	3.0520
Carter v. Carter Coal Co.	3.0570

We can also look at the closeness centralization of the main component of the Supreme Court network. This statistic basically measures the amount of variation in closeness scores of the vertices. Centralized networks such as star topologies have closeness centralization towards one. Decentralized networks have closeness centralization towards zero. The main component of the Supreme Court network has a closeness centralization of 0.19711. It is thus a fairly decentralized network.

The Most Between Cases

A related measure of importance frequently used in this field is betweenness. Just as various sites or routes lie on multiple earth geodesics with varying probability—Reykjavik, Iceland lies roughly on many geodesics flown by air traffic; Perth, Australia lies on much fewer—so too various nodes and edges on a network will lie on geodesics with varying probability. The intuition is that the nodes or edges that lie on the highest proportion of network geodesics are the most important purveyors of information due to the assumed proclivity of information to flow by the fastest route available. Cases with high betweenness serve as a communications hub that facilitates the transmission of ideas. Edges with high betweenness are vitally accelerating the transmission of ideas, too.

The following chart, generated using Pajek, shows the most between cases and citations in the Supreme Court database. Interestingly, it is a compendium largely of what we might call structural cases, involving issues such as the allocation of power between the federal government and the states, the separation of federal power among the executive, legislative, and judicial branches, or the appropriate role for precedent itself. Most of these cases would be well known to American constitutional experts and indeed comprise a significant part of the curriculum in courses in the area of federal jurisdiction.

Case or Citation	Betweenness
Crowell v. Benson	0.0133
Erie Railroad Co. v. Tompkins	0.0116
McCulloch v. Maryland	0.0112
Baker v. Carr	0.0097
Burnet v. Coronado Oil & Gas Co.	0.0087
Ashwander v. Tennessee Valley Auth.	0.0085
Cohens v. Virginia	0.0071
Glidden Co. v. Zdanok	0.0065
Fay v. Noia	0.0057
Home Building & Loan Assn. v. Blaisdell	0.0056
Marbury v. Madison	0.0056
Commissioner v. Estate of Church	0.0054
Osborn v. Bank of the United States	0.0052
Minnesota Rate Case	0.0052
Ferguson v. Moore - McCormack Lines, Inc.	0.0051
Gibbons v. Ogden	0.0050
Olmstead v. United States	0.0049
Joint Anti - Fascist Refugee Committee v. McGrath	0.0046
Yakus v. United States	0.0045
Murdock v. Memphis	0.0044

□ The Main Core of Supreme Court Jurisprudence

An experiment sometimes conducted on networks is to discover their main core. To do so, we start removing nodes that have fewer than some number (k) of connections to other nodes in the network. When we do this, we find, however, that we must iteratively repeat the procedure because, after the first pass, there

may have been some cases that had $k + m$ connections but, of those, more than m were removed in an earlier pass. By the time this procedure hits a fixed point, each case in the resulting network will be connected to at least k other cases in the resulting network. As we start increasing k, we get a set of cases that are increasingly well cited and increasingly interdependent. And we can see how high k can go before the network disappears altogether. This maximum value of k induces what is sometimes known as the main core of a network. The following code implements this procedure.

```
Options[kcore] = {Verbose → True, maxIters → 50, extraIters → 0};

kcore[g_Graph, k_Integer, opts___] := Module[{vb, mi, ei},
  {vb, mi, ei} =
   {Verbose, maxIters, extraIters} /. {opts} /. Options[kcore];
  NestWhile[Function[gr, (If[vb, Print[{V[gr], M[gr]}]];
     InduceSubgraph[gr, Flatten[Position[BinCounts[
         Flatten[gr[[1]]], {0.5, V[gr] + 0.5, 1}], _?(# ≥ k &)]]])],
   g, V[#1] > V[#2] &, {2, 2}, mi, ei]]
```

Some experimentation shows that the main core of Supreme Court jurisprudence consists of 122 cases each with 28 or more links to other cases in the main core.

```
Timing[maincoregraph = kcore[g, 28, Verbose → False]]

{218.5 Second, -Graph:<2485, 122, Undirected>-}
```

Figure 2 shows a visualization of the main core. Most of the cases involve rights of free speech and association under the American constitution. Perhaps not surprisingly, the density of the main core is about 0.34, more than 500 times greater than the density of the network as a whole.

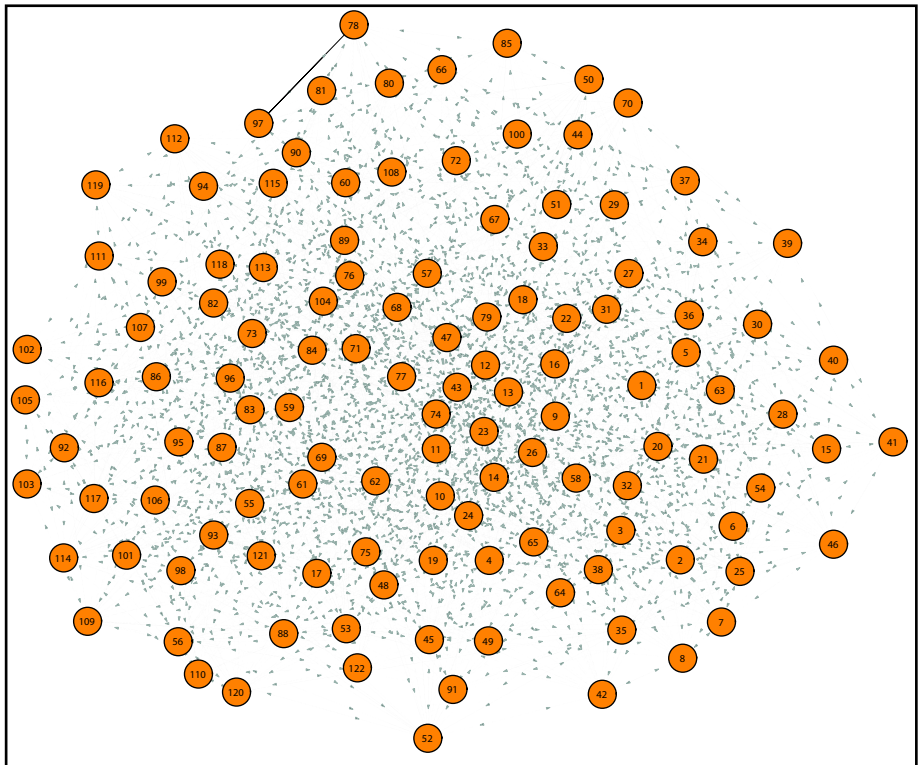


Figure 2. The main core of Supreme Court jurisprudence.

We can also examine the Markov Centrality of each of the cases in the main core. The Markov Centrality of a node in a network is simply the probability that we would end up at that node after taking a random walk on the network. It can

be computed using the eigenvectors of the transition matrix created by the graph. Here is a list of the 10 most central cases in the main core of Supreme Court jurisprudence, using the Markov Centrality measure.

```
Take[Part[GetVertexLabels[maincoregraph],
  Reverse[Ordering[With[{ev = First[
        Eigenvectors[Map[#/Max[1, Total[#]] &, N[ToAdjacencyMatrix[
            maincoregraph]]]^T , 1]]}, ev/Total[ev]]]], 10]
```

```
{Schneider v. State, NAACP v. Button,
 Cantwell v. Connecticut, Thornhill v. Alabama,
 Young v. American Mini Theatres, Inc., Kovacs v. Cooper,
 Chaplinsky v. New Hampshire, Grayned v. City of Rockford,
 Lovell v. City of Griffin, New York Times Co. v. Sullivan}
```

Our supposition is that, because of the high degree of connectivity in the free speech field, changes in doctrine produced by new cases are particularly influential and may create considerable disequilibrium in the jurisprudential system, much in the same way that a new disease is likely to produce a more prolonged and extensive disequilibrium state of public health in a highly connected community than in some isolated pocket of the world. This high degree of interdependence provides scientific support for the notion that the law of free speech is particularly complex.

■ Conclusion

□ What We Can Learn About Network Analysis of the Law

This exploration delivers several valuable findings. First, it confirms that analysis of the sizeable works of a court as a form of large networks is now possible using modern computers [12]. Second, it shows how network analysis can refine our understanding of the seamless web to which the law is often analogized. The Supreme Court network is a large and intricately tangled web, to be sure, but deep within there is structure that our mathematical probes can now discern. Measures such as betweenness, closeness, and Markov Centrality help us find the cases that lie at the core of a judicial system. Measures such as clustering enable us to understand the degree of interdependence of cases that comprise the database. And notions such as that of a main core help identify areas that are particularly complex.

We also return with several specific preliminary findings about American law and its leading court. Although, precisely because networks are intricate creatures, these findings must be re-examined as the database is perfected. It appears that structural cases involving the role of the federal courts appear to lie quite literally at the core of Supreme Court jurisprudence. These older structural cases lie closer to other cases than do those in competing bodies of jurisprudence, and they appear more important in the transmission of information across the network as to the rules of the law. Cases involving the power of the federal

government to regulate commerce also appear critical. The law governing rights of free speech and association is one in which the density of the seamless web is greatest, in which the reductionism occasioned by study of a single case is most likely to mislead as to the functioning of the legal organism. In this area, the level of interconnection among cases is extraordinarily high. Reinterpretations of a particular case are likely to reverberate significantly throughout a complex system, making the area particularly difficult for scholars who would like an understanding of those rights to reach some sort of equilibrium. Although the density of the Supreme Court network is in general quite low, clustering appears to be quite high. Related cases tend to find each other with a high regularity and only a small fraction of cases lack connection to the main body of Supreme Court jurisprudence.

□ What We Can Learn about *Mathematica*

This article also delivers mixed findings about *Mathematica*. On the one hand, particularly as extended via *J/Link*, the language and program show great strength and flexibility. An inexperienced Java programmer was able to use *J/Link* to access the latest in network analysis technology and use the speed of Java to conduct complex analyses on a large database. On the other hand, some of the analysis proved difficult using either *Mathematica* directly or as a tool to manipulate JUNG libraries via *J/Link*. For some of the most complex computations, resort had to be made to Pajek, an external program with very limited extensibility; *Mathematica* was relegated to the lesser (though nontrivial) role of data preparation. As network analysis extends its domain outwards from physics and into biology, sociology, and now law, the case for *Mathematica* to internalize at least some of the capabilities presented becomes stronger. A revamped *Combinatorica*, where much of the analysis could be conducted without learning Java programming style or coping with the complexities or development of JUNG Java libraries, would certainly help. On the other hand, the time needed for JUNG or Pajek to complete some tasks, such as betweenness, suggests that some of the difficulty experienced in preparing this article may possibly result more from incurable shortcomings with existing algorithms than failures of computer implementation.

□ Future Explorations

This article has just scratched the surface of a large research program. While we have learned a few things about Supreme Court jurisprudence and have elucidated some choices scholars may make in teaching and studying its decisions, we yet know little about (a) how the Supreme Court network compares to other precedent-based legal systems; (b) how its network features have evolved over time; (c) how information about the content of the cases can be used to refine an analysis of its network structure; (d) whether its connectivity structure can be approximated or generated by any simple program; and (e) how the Supreme Court network compares to other common law networks and various statutory networks, including the Uniform Commercial Code, the United States tax code, or other legal texts. Moreover, while we now have powerful tools at our disposal,

About the Author

Seth J. Chandler is a law professor at the University of Houston Law Center, where he also serves as Co-Director of its Health Law & Policy Institute. Chandler has published and presented numerous articles on the use of *Mathematica* in insurance law and the economic analysis of law. He has also presented works using *Mathematica* at two conferences on *A New Kind of Science*. He holds a J.D. degree from Harvard University and an A.B. from Princeton University.

Seth J. Chandler
University of Houston Law Center
100 Law Center
Houston, Texas 77025
SChandler@uh.edu