Internet Page Ranking Algorithms

Michael Monagan
Department of Mathematics
Simon Fraser University
mmonagan@sfu.ca

In the Fall of 2014 Cedric Chauve, a colleague of mine in mathematics at SFU showed me how google ranks the pages it presents to you when you use google to search for a page on the internet. Cedric is using this example in teaching our Linear Algebra class at SFU. He, and two others who have taught the course think that this is such a wonderful example that we have decided to include this in all our Linear Algebra offerings. I have also used it in teaching MACM 204 Computing with Calculus, a second year course for teaching students how to use Maple for doing Calculus. MACM 204 is a 2 credit course with one hour of lecture and one hour of lab each week. In this course, unlike the linear algebra course, I get the students to simulate the "web surfer" described below. This requires some programming on the part of the student. The problem is nice because it introduces some graph theory, stochastic matrices, and some ideas from computation, namely using the power method to compute a steady state vector.

The Page Ranking Problem.

Suppose you type "Gaussian Elimination" or "How do I factor a polynomial" into google. Google will first identify which pages on the internet are of interest. Let's label them pages $\{P_1, P_2, P_3, ..., P_N\}$ and denote this set of pages by P. Usually there will be a lot of possible pages, thousands of them, maybe hundreds of thousands of them, so N is large. How does google order them from most likely useful to least likely useful? And if you click on the

I'm Feeling Lucky

search button, how does google choose the best page to show you? The answer (ignoring embellishments) is that it uses the hyperlinks between the pages $P_1, P_2, ..., P_N$ to rank them. Let's consider an example.

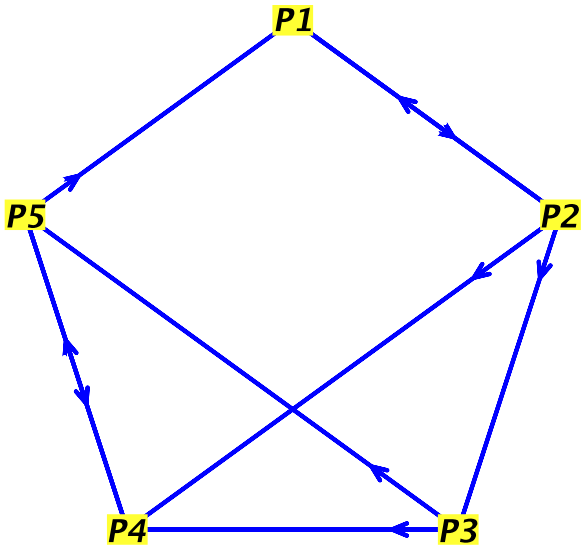
Suppose we have five pages P_1, P_2, P_3, P_4, P_5 and suppose there are hyperlinks from page
P_1 to pages P_2, P_4 and
P_2 to pages P_1, P_3 and
P_3 to pages P_4, P_5 and
P_4 to page P_5 and from
P_5 to pages P_1, P_4 .

The best way to picture this is to look at it as a network, that is, a directed acyclic graph. I'll use Maple's *GraphTheory* package to draw this

```
> restart; with(GraphTheory):
> G := Graph(directed,[P1,P2,P3,P4,P5],
   {[P1,P2],[P2,P4],[P2,P1],[P2,P3],[P3,P4],
     [P3,P5],[P4,P5],[P5,P1],[P5,P4]});
```

G := Graph 1: a directed unweighted graph with 5 vertices and 9 arc(s)

```
> DrawGraph(G);
```



Looking at the graph we can see that page P4 has hyperlinks to it from pages P2, P3 and P5. It's the only page with three hyperlinks pointing to it. We say page P4 is an ***authority*** meaning that because many other web pages reference it, it must be useful. So one possible way to rank the pages would be simply to order them by the number of hyperlinks pointing to them. Let us define for each page $Authority(P_i)$ is the number of hyperlinks to page P_i. In Maple the **Arrivals** command gives, for each vertex $v \in G$ a list of vertices with directed edges to v.

```
> Arrivals(G);
```

$$[[P2, P5], [P1], [P2], [P2, P3, P5], [P3, P4]]$$

```
> Authority := map(numelems,Arrivals(G));
```

$$Authority := [2, 1, 1, 3, 2]$$

So we would rank them $P_4 > P_1$, $P_5 > P_2, P_3$. There are two problems with this crude ranking algorithm. First, there will probably be a lot of pages with the same number of hyperlinks pointing to them so this does not rank them sufficiently. Second, although page P_5 has authority 2, since the only hyperlink on page P_4 is to P_5 we might argue that P_5 also has high authority, implicitly, because of P_4. Somehow, we need to transfer the authority of P_4 to P_5.

The way google does this and ranks the pages is as follows. Imagine you are surfing the web and you restrict your surfing to the pages in P. At each step you are on some page P_i for $1 \le i \le N$ and you randomly choose a hyperlink to one of the other pages in P and go to that page. Let's assume we pick the hyperlinks with equal probability. If you surf in this way indefinitely you will land on certain pages more often than others. We can associate with each page P_i the probability q_i that we are on that page. Now order the pages by decreasing probability. That's the ranking.

One way we could compute the probabilities q_i would be to start on page P_1 and simulate

the random web surfer going through the pages in P, and calculate the number of times f_i we are on page P_i. If we do this for sufficiently large number of steps M then $\frac{f_i}{M} \sim q_i$. This simulatation is a wonderful exercise for the students to undertake. I will show how to do this later. For now I want to develop an approach to compute the q_i using linear algebra.

A Linear Algebra Approach

Let G be the graph representing the hyperlinks between the pages. Let $A_{i,j}$ be the probability of going from page P_i to P_j assuming we take each hyperlink on page P_i with equal probability. For example, in our graph $A_{2,4} = \frac{1}{3}$ because we can go from page P_2 to pages P_1, P_3 or P_4 each with probability $\frac{1}{3}$. I will construct the matrix A from our graph G starting with the adjacency matrix for G.

```
> A := AdjacencyMatrix(G);
```

$$A := \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

```
> DP := Departures(G);
```

$$DP := [[P2], [P1, P3, P4], [P4, P5], [P5], [P1, P4]]$$

```
> n := 5:
  for i to n do
     s := numelems(DP[i]);
     for j to n do A[i,j] := A[i,j]/s od;
  od:
> A;
```

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 1 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

Now the matrix A^T is a stochastic matrix or Markov matrix because the values are probabilities and the columns of A^T sum to 1. The key property is that if we have a

probability vector Q where Q_i is the current probability of being on page P_i, then the vector W, given by $W = A^T \cdot Q$, is a probability vector where W_i is the probability of being on page P_i after taking one hyperlink at random. Since this is not obvious, the reader may wish to check this. We will check this for one value. We'll start with the vector $Q_0 = \left[\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N}\right]^T$, which assumes we are initially on each page with equal probability.

```
> with(LinearAlgebra):
> Q[0] := <1/5,1/5,1/5,1/5,1/5>:
> Q[1] := Transpose(A).Q[0];
```

$$Q_1 := \begin{bmatrix} \frac{1}{6} \\ \frac{1}{5} \\ \frac{1}{15} \\ \frac{4}{15} \\ \frac{3}{10} \end{bmatrix}$$

The probability of being on page P_1 after one time step is 1/6. This came from the dot product of the two vectors

```
> A[1..5,1], Q[0];
```

$$\begin{bmatrix} 0 \\ \frac{1}{3} \\ 0 \\ 0 \\ \frac{1}{2} \end{bmatrix}, \begin{bmatrix} \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \\ \frac{1}{5} \end{bmatrix}$$

So the dot product of these two vectors represents the probability of being on page P_1 after one hyperlink from each page is taken at random. Now we can only get to page 1 from page 2 and 5 with probability 1/3 and 1/2 respectively. Hence after taking one hyperlink we are on page P_1 with probability $\frac{1}{3} \cdot \frac{1}{5} + \frac{1}{2} \cdot \frac{1}{5} = \frac{1}{6}$. Letting $B = A^T$ for convenience we can now determine the probabilities of being on each page after taking k random hyperlinks by multiplying Q_0 by B^k. Here is what happens for $k = 5$.

```
> B := Transpose(A);
```

$$B := \begin{bmatrix} 0 & \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 1 & 0 \end{bmatrix}$$

```
> for k to 5 do Q[k] := B.Q[k-1] od:
  evalf(Q[5]);
```

$$\begin{bmatrix} 0.2018518519 \\ 0.2138888889 \\ 0.06851851852 \\ 0.2379629630 \\ 0.2777777778 \end{bmatrix}$$

Now because our particular graph is strongly connected, that is, you can get from every page P_i to every other page P_j, this means the matrix A^T is a regular matrix. It follows from the Peron-Frobenius theorem that the sequence Q_0, Q_1, Q_2, ... will converge to a steady state vector $V = \lim_{k \to \infty} B^k Q_0$ that satisfies $A^T \cdot V = V$. From the latter identity we have $(A^T - I) \cdot V = 0$ which gives us a direct way to solve for the steady state vector V. The vector V is in the nullspace of $A^T - I$. The equation $A^T \cdot V = V$ also means that V is an eigenvector of A^T with eigenvalue 1. Let us try to solve for V using $(A^T - I) \cdot V = 0$.

Now, one thing I would like to do is use the letter I for the identity matrix. In Maple, by default, I is the complex unit, that is $I = \sqrt{-1}$.

```
> I^2;
```

$$-1$$

To disable this we can do the following which makes $\_i$ the complex unit and frees up the letter I for general usage.

```
> interface(imaginaryunit=_i);
```

$$I$$

```
> with(LinearAlgebra):
> I := IdentityMatrix(5);
```

$$I := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

```
> V := NullSpace(B-I);
```

$$V := \left\{ \begin{bmatrix} \frac{3}{4} \\ \frac{3}{4} \\ \frac{1}{4} \\ \frac{7}{8} \\ 1 \end{bmatrix} \right\}$$

This is a basis for the nullspace (a set of vectors) so we need to pick the probability vector in $Span(V)$ and impose that the entries are probabilities which sum to 1.

```
> V := V[1]:
  s := add(V[i],i=1..5);
  V := V/s;
```

$$s := \frac{29}{8}$$

$$V := \begin{bmatrix} \frac{6}{29} \\ \frac{6}{29} \\ \frac{2}{29} \\ \frac{7}{29} \\ \frac{8}{29} \end{bmatrix}$$

So these are the probabilities the q_i. We can see how close Q_5 is to V.

```
> evalf( V-Q[5] );
```

$$\begin{bmatrix} 0.005044699872 \\ -0.006992337165 \\ 0.0004469987229 \\ 0.003416347382 \\ -0.001915708812 \end{bmatrix}$$

Now the matrix A^T is very large but very sparse. Most of the entries in it are zero. This is simply because each web page typically has only a few hyperlinks to pages in P. This direct method of solving for V using Gaussian elimination on the matrix $A^T - I$ leads to fill-in which makes the direct approach very slow for large N. For this reason, google does not solve for V directly but instead generates the sequence $Q_1, Q_2, \ldots$ using the matrix vector multiplication $Q_{k+1} = A^T \cdot Q_k$ which, because the matrix is sparse, this matrix-vector multiplication is efficient. Typically the sequence will have converged in 100 steps. Let's try it

```
> for k from 6 to 100 do Q[k] := B.Q[k-1] od:
  evalf(Q[100]),evalf(V);
```

$$\begin{bmatrix} 0.2068969464 \\ 0.2068961167 \\ 0.06896567710 \\ 0.2413796169 \\ 0.2758616429 \end{bmatrix}, \begin{bmatrix} 0.2068965517 \\ 0.2068965517 \\ 0.06896551724 \\ 0.2413793103 \\ 0.2758620690 \end{bmatrix}$$

We can see that the sequence has converged to 6 decimal places. Thus we rank the pages $P_5 > P_4 > P_1, P_2 > P_3$. So page P_5 is the first page.

Non Regular Markov Chains

We have assumed that the Markov matrix A^T is regular, that is the graph G is strongly connected. There's no reason why this would be the case in practice. In fact, we can readily see that there will be many web pages with no hyperlinks to them - perhaps the majority. The existence of the steady state vector depends on this property of A^T. One simple solution is to connect every vertex in G with every other vertex and take those hyperlinks with low probability p. Let C be the matrix with $C_{i,j} = p$ for some small value of $p \ll \frac{1}{N}$. Now adjust the columns of the matrix A^T so that $A^T + C$ is a Markov (stochastic) matrix. This is a practical solution.

Simulating the web surfer.

I ask my students to do the following. Starting on page P_1 simulate a web surfer who takes $N = 10,000$ hyperlinks at random. Calculate the frequencies f_i of being on page P_i and use

these to estimate q_i. Now to see the convergence, periodically, after every 100 hyperlinks taken, calculate the estimates for the probabilities and graph them so we can see how fast this converges. The graph below will clarify what I want the students to display. Here is my solution.

```
> DP := Departures(G);
```

$$DP := [[P2], [P1, P3, P4], [P4, P5], [P5], [P1, P4]]$$

To take random hyperlinks, for page $P2$ here we need to generate one of $P1$, $P3$, $P4$ at random. I will use a random number generator R_3 to generate 1, 2, or 3 at random (uniformly) and then select one of those pages. I will also need random number generators for 1 page and 2 pages. Hence

```
> R[1] := rand(1..1):
  R[2] := rand(1..2):
  R[3] := rand(1..3):
> DP[2][R[3]()];
```

$$P1$$

```
> DP[2][R[3]()];
```

$$P4$$

Now I have labelled the pages $P1$, $P2$, $P3$, $P4$, $P5$ and in my Maple code I need to determine from P_i, the index i.

```
> Page := table():
  for i to 5 do Page[P||i] := i od:
  Page[P3];
```

$$3$$

```
> x := P1;  # x is the current page
  N := 10000; # number of steps
  F := Array(1..5); # frequencies
  V := Array(1..iquo(N,100)): # snapshots of probabilities
  for i to N do
     Nx := DP[Page[x]];
     r := R[nops(Nx)]();
     x := Nx[r]; # go to the next page
     F[Page[x]] := F[Page[x]] + 1;
     if i mod 100 = 0 then
        V[i/100] := evalf( [seq( F[j]/i, j=1..5 )] );
      fi;
  od:
```

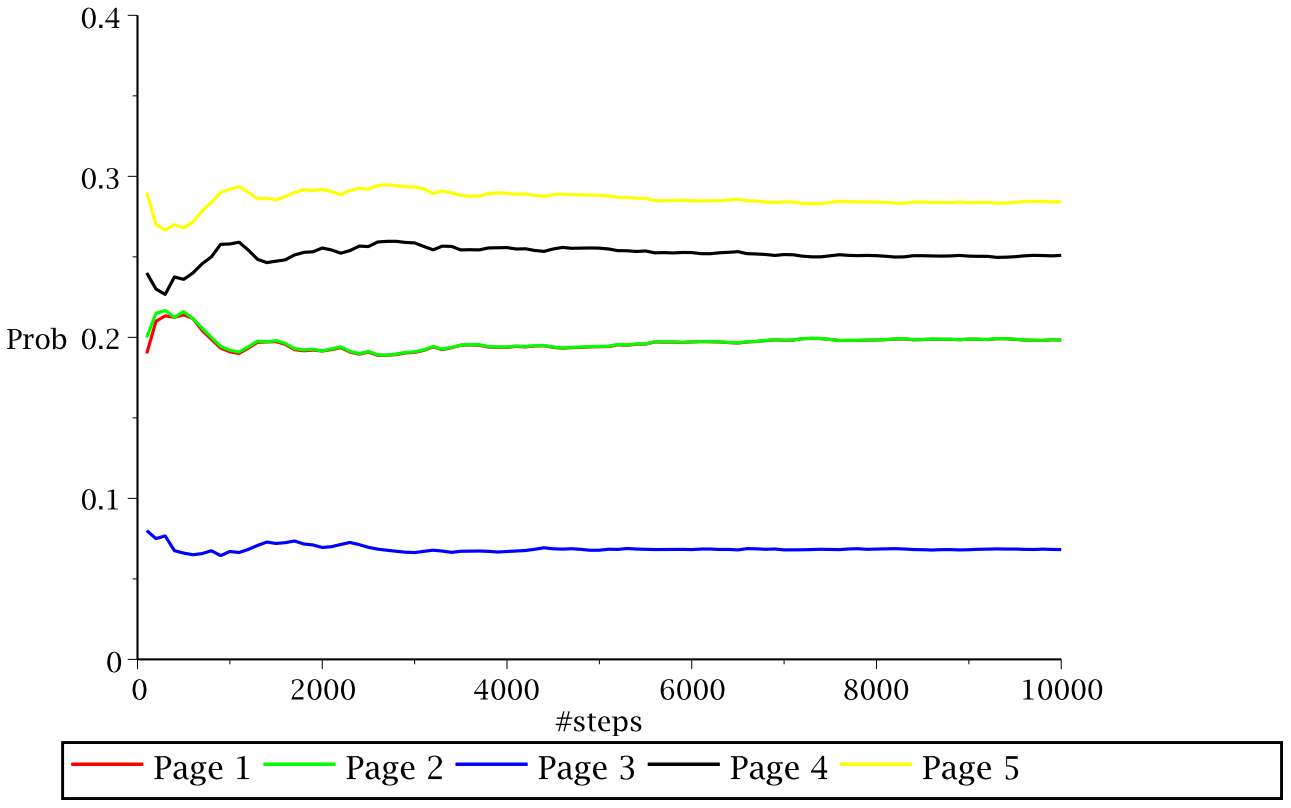
$$x := P1$$

$$N := 10000$$

$$F := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> data := [seq( [seq([100*i,V[i][j]],i=1..i/100)],j=1..5 )]:
> cols := [red,green,blue,black,yellow];
  plot( data, style=line, color=cols, view=[0..N,0..0.4], labels=
  ["#steps","Prob"],
      legend=["Page 1", "Page 2", "Page 3", "Page 4", "Page 5"] );
```

$cols := [red, green, blue, black, yellow]$



So the vertical axis is the probability of being on one of the pages. You can see that it converges quite slowly, slower than the sequence Q_0, Q_1, Q_2, ... The value of generating this plot is that the student learns how to do a simulation, assemble data and display it in a meaningful way. Incidentally, if you right click on the plot you will get a menu of options. One of them is the probe option, which if selected, causes Maple to display (when you move the cursor over the plot) the (x, y) co-ordinates of the cursor position. This is a very useful feature. Here, I can see what probability each of the above curves is converging to.