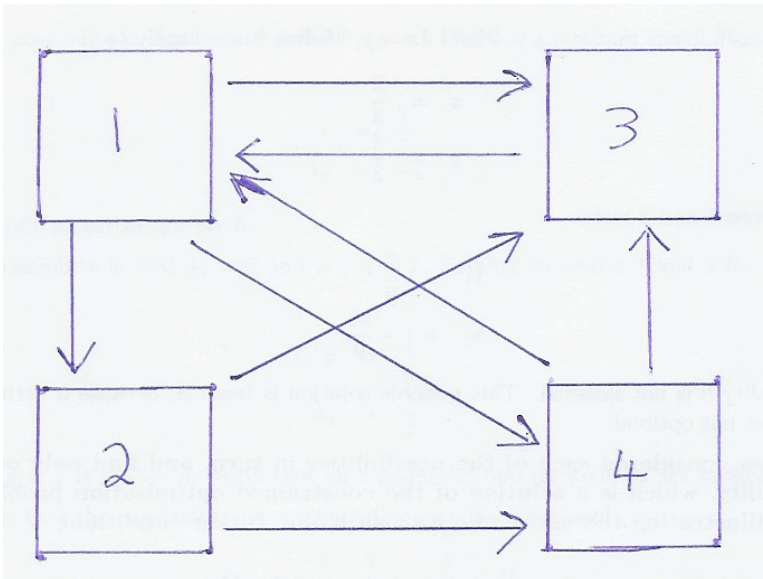


Google type Page Rank Algorithms (based on a SIAM Review article, but of course a google search will find many others on the web)

The success of google is predominantly due to its page rank algorithm. All web crawlers can index each page for the terms contained in each page. In response to a query such as "page rank algorithm" google presents the first 10 items out of a possible 2.5 million. No user can trawl through the whole 2.5 million pages, so we rely on the page rank algorithm to give us the pages we most want to see in the first 10 or 20 or 30 pages we are prepared to bring up on our screens. The function of the page rank algorithm is to satisfy that user requirement. Google's initial takeup in preference to existing search engines such as Alta Vista was due to their algorithm.

Page rank algorithms have to be automated, because of the huge number (billions) of pages which have been indexed, and they have to be re-run to reflect additions in the total web content. To illustrate the ideas behind google's page rank algorithm, consider a small web world consisting of just 4 pages with links between them as shown in the following diagram. An arrow pointing from page 2 to page 4 means that page 2 contains a reference to page 4.



We would like to compute a vector x_k reflecting the importance of page k . A simple minded approach would be just to count the number of links to each page.

k	1	2	3	4
Links to page k	2	1	3	2

This approach does not reflect the fact that some pages might be more significant than others, for some reason other than just the number of links to it, and also leaves open the possibility of artificially inflating the rank of a particular page by generating other trivial or advertising pages whose only function is to promote the importance of a particular page. Significant refinements are:

- Weight each in-link by the importance of the page which links to it.
- Give each page a total vote of 1.0, however many pages it links to, so that a page j with outgoing links m_j will have a weight assigned to it of $\frac{x_j}{m_j}$.

If L_k is the set of pages which link to page k , the importance x_k can be written

$$x_k = \sum_{j \in L_k, j \neq k} \frac{x_j}{m_j}$$

For our simple example of 4 web pages, we have:

$$\begin{aligned} x_1 &= x_3 + \frac{1}{2} x_4 \\ x_2 &= \frac{1}{3} x_1 \\ x_3 &= \frac{1}{3} x_1 + \frac{1}{2} x_2 + \frac{1}{2} x_4 \\ x_4 &= \frac{1}{3} x_1 + \frac{1}{2} x_2 \end{aligned}$$

or

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

which can be interpreted as an eigenvalue eigenvector problem

$$A \mathbf{x} = \lambda \mathbf{x}$$

where the eigenvalue is 1.

How do we know that the matrix A has a unit eigenvalue? We note that A is a column-stochastic matrix, which is to say that each column sums to 1.0. This is a consequence of the fact that we gave each page a total vote of 1, so provided each page has at least one outgoing link, each column will sum to 1. If we consider that

$$A^T \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

and that the eigenvalues of A and A^T are the same, we can see that A must have a unit eigenvalue whose corresponding eigenvector contains the page ranks that we seek.

What is the situation with the real web?

1. the web consists of billions of pages. The consequence of this is that computing the eigenvector is going to be a massive computational undertaking. It does not have to be re-computed for each response to a query that google makes. All that is required, which is still substantial, is to serve up the pages containing the terms we searched for, in the order corresponding to decreasing eigenvector components. However, the eigenvector does have to be recomputed regularly to reflect the evolution of web content.
2. there are pages without any outgoing links. This means that we can no longer guarantee that the matrix A^T , and therefore the matrix A , has a unit eigenvalue.
3. there are groups of pages (cliques in graph theory terms) which have no links to pages outside their clique. This means that even if there were no pages with no outgoing links, there would be multiple eigenvectors corresponding to the unit eigenvalue. We call such a matrix A reducible.

The complications we have identified can be addressed by computing, instead of the matrix A , with the matrix

$$C = \mu A + (1 - \mu) B$$

where B is the matrix with every element $b_{i,j} = \frac{1}{N}$, N being the total number of pages indexed, and where μ is a small positive parameter chosen to avoid loss of significance in the computation.

Adding a multiple of the matrix B makes the matrix C irreducible, in that now every page is linked to every other page. At the same time it makes every element of the matrix C positive, and retains the original column stochastic property for any of the pages that already had outgoing links. We can apply the Perron-Frobenius theorem (1906) to the matrix C to assert that it has a real positive eigenvalue equal to its spectral radius, that this eigenvalue is actually bigger in modulus than any of the other eigenvalues of C , and that its corresponding eigenvector has all real positive components.

This dominant eigenvalue will not necessarily have eigenvalue 1 in the presence of nodes with no outgoing links, but its corresponding eigenvector will still be suitable for ranking all of the pages on the web. The fact that the eigenvalue is dominant means that it can be computed iteratively, using the power method, which we can summarise as:

$$\begin{aligned} \mathbf{x}_0 &= \text{a vector with all components } \frac{1}{N} \\ \textit{Repeat until convergence} \\ \mathbf{y} &= C \mathbf{x}_{k-1} \\ \eta &= \sum y_j \\ \mathbf{x}_k &= \frac{1}{\eta} \mathbf{y} \end{aligned}$$

In fact there are better methods than the simple power method, such as Chebyshev iteration, to converge to the dominant eigenvalue-eigenvector combination, which google undoubtedly use, but the power method is easy to explain. If \mathbf{z}_j are the eigenvectors of C , with corresponding eigenvalues λ_j , after k steps of the power method, we have

$$\begin{aligned} \mathbf{x}_0 &= \sum \alpha_j \mathbf{z}_j \\ \mathbf{x}_k &= \alpha_1 \mathbf{z}_1 + \sum_{j>1} \left(\frac{\lambda_j}{\lambda_1}\right)^k \alpha_j \mathbf{z}_j \end{aligned}$$

As the eigenvalue λ_1 is larger in modulus than any of the other λ_j it is easy to see that the power method will converge to the required eigenvector \mathbf{z}_1

One point worth observing is the use of the L_1 norm rather than the L_2 norm to re-normalise the eigenvector at each step of the iteration (dividing by η , the sum of the elements of \mathbf{x}). We have noted before that the original matrix A , and hence B and C are square matrices with billions of rows. Each web page is only likely to have a small number of outgoing links, so the matrix A will be very sparse, in contrast to both B and C which are both full matrices. If we had to multiply by the full matrix C this would be an extremely expensive computational step. However, with the L_1 renormalisation we note that

$$\begin{aligned} C \mathbf{x}_{i-1} &= (\mu A + (1 - \mu) B) \mathbf{x}_{i-1} \\ &= \mu A \mathbf{x}_{i-1} + (1 - \mu) \mathbf{x}_0 \end{aligned}$$

which means that multiplication by C is no more expensive than multiplying by A .

You can see the effect of using C rather than A on the eigenvector. Adding the same small constant to each of the components, which is already positive, should not change the rank ordering of those components.

Bryan,K and Leise,T (2006) The \$25,000,000,000 Eigenvector: The Linear Algebra Behind Google SIAM Review 48, 3, 569-581