Appendix

Proof. Let S be the PageRank transition matrix and let $\vec{x}_{(j)}$ be a vector in the Markov chain, with starting probability vector $\vec{x}_{(0)}$, for some iteration j. If N is very large, the direct multiplication $\vec{x}_{(j+1)} = S\vec{x}_{(j)}$ is computationally intensive. However, the computation can be simplified dramatically if we take advantage of the structured components of S given in the equation above. Because M is sparse, the multiplication $\vec{w}_{(j)} = M\vec{x}_{(j)}$ is computationally much simpler. We will then show that if we set

$$\vec{b} = \frac{1-d}{N}\vec{e}$$

then $E\vec{x}_{(j)} = \vec{e}$ and $\vec{x}_{(j+1)} = d\vec{w}_{(j)} + \vec{b}$ where $\vec{e}$ is an N-vector of all ones.

It is trivial to show that $E\vec{x}_{(j)} = \vec{e}$. Recall that the probabilities in $\vec{x}_{(j)}$ add up to 1.

Then

$$\vec{x}_{(j+1)} = S\vec{x}_{(j)} = \left(dM + \frac{1-d}{N}E\right)\vec{x}_{(j)} = \left(dM + \frac{1-d}{N}\vec{e}\cdot\vec{e}^{T}\right)\vec{x}_{(j)}$$

$$= dM\vec{x}_{(j)} + \frac{1-d}{N}\vec{e}\cdot\vec{e}^{T}\vec{x}_{(j)} = dM\vec{x}_{(j)} + b\vec{e}^{T}\vec{x}_{(j)}$$

It is given that $\vec{w}_{(j)} = M\vec{x}_{(j)}$, and since $\vec{e}^{T}\vec{x}_{(j)} = 1$, It follows that $\vec{x}_{(j+1)} = d\vec{w}_{(j)} + \vec{b}$. □

Algorithm 1 Calculate PageRank.

Input:
A := adjacency matrix where the the columns of A correspond to the outgoing links and the rows correspond to the incoming links.
d := the damping factor.
$iter$:= the number of iterations.

Initialize:
$p \leftarrow$ number of pages
$sink \leftarrow$ pages with no outgoing links
for Any page $i \in sink$ **do**
  $A[, i] \leftarrow (1, 1, \cdots, 1)$ // length p vector of ones
end for
$diag(A) \leftarrow 0$ // no self loops
$M \leftarrow$ transition matrix associated with A
$e \leftarrow$ p length vector of all ones
$prev.ranks \leftarrow$ create list of initial ranks // initial ranks are equal to 1/(p)
$new.ranks \leftarrow$ p length vector of all zero
$b \leftarrow ((1-d)/p) * e$
$j \leftarrow 0$ //counter for interations
while $j < iter$ **do**
  $w_j \leftarrow M \cdot prev.ranks$
  $new.ranks \leftarrow (d * w_j) + b$
  $prev.ranks \leftarrow new.ranks$ // update step
  $new.ranks \leftarrow$ reinitialize to p length vector of all zero
  $j \leftarrow j + 1$
end while
return $prev.ranks$