***Algoritmos de Page Rank e casos de uso***

Daniel Santos Peixoto

Universidade Federal da Bahia

Salvador, BA, Brazil

Maria Luisa Santos Rodrigues

Universidade Federal da Bahia

Salvador, BA, Brazil

*Abstract*—O Page Rank original foi desenvolvida pela Google, desde então ele tem sofrido modificações e sido usado em diferentes tipos de situações. Um dos problemas atuais é lidar com alto volume de dados, nesse relatório iremos discutir como manipular tamanho volume de dados e manter um nível agradável de perfomance.

Keywords—large graphs; page rank; comparison

# Introdução

O algoritmo original do Page Rank não é tão bom quando os algoritmos dos buscadores atuais e não lida com os problemas atuais.

Atualmente, volumes enormes de dados sao gerados toddo dia, e a maior parte é composta de dados desestruturados. Em alguns casos, o algoritmo do Page Rank pode ser usado para agregar valor à esses dados, pois outros problemas comum podem ser adequados para se resolver com esse algoritmo, como, por exemplo, logs de aplicação, recomendação de produtos, ranqueamento de usuários, etc.

Como diferentes problemas lidam com diferentes tipos de recursos e possuem diferentes propósitos, necessitamos escolher um algoritmo que se encaixe ao problema. Alguns metódos e implementações podem se mostrar lentas, incapazes de garantir certas restrições, e, até mesmo, impossível de ser resolver, dependendo do caso.

Nesse relatório, nós iremos prover comparações entre diversos algoritmos que irão servir como uma guia para o desenvolvimento de novas aplicações, de acordo com seus recursos e problemática.

# Literatura

Dentre os primeiros algoritmos de ranqueamento de páginas e pontuação de sites, se destacaram o PageRank, desenvolvido por Larry Page e Sergey Brin e utilizado até hoje pelo Google, e o RankDex, desenvolvido por Robin Li e utilizado pelo Baidu.

O algoritmo PageRank, em sua forma original, determina o *rank* de uma página web baseado na probabilidade que um usuário chegue até ela, calculada a partir dos hiperlinks em outras páginas web que direcionam até ela. Para uso do algoritmo, as páginas pertencentes a um dado conjunto (no caso do Google, as páginas indexadas) são representadas como vértices de um grafo direcionado cujas arestas representam hiperlinks de uma página (vértice de origem) a outra (vértice de destino). O algoritmo funciona iterativamente, partindo de um PageRank inicial atribuído a todos os vértices igualmente e, a cada iteração, o PageRank de cada vértice é atualizado de acordo com seus adjacentes.

No algoritmo original, o PageRank inicial era igual a 1. Em versões seguintes, para facilitar a análise dos relatórios, o algoritmo utiliza uma distribuição de probabilidade de 0 a 1, logo o PageRank inicial de cada vértice é dado pelo quociente 1/N, sendo N o número de vértices. O objetivo é modelar o comportamento de um usuário, apelidado por Page e Brin como *random surfer*: partindo de uma página aleatória, esse usuário clica em um link aleatório presente nessa página, sendo levado a outra página. Por isso, a cada iteração, cada vértice transfere aos seus adjacentes o seu próprio PageRank dividido igualmente. As iterações continuam até que se detecte convergência, ou seja, que a diferença no PageRank de cada vértice entre uma iteração e outra tenda a um dado parâmetro suficientemente pequeno. Além disso, para evitar que páginas tenham PageRank zero e modelar mais fielmente o comportamento de um usuário, o algoritmo utiliza um parâmetro chamado *damping factor*, que representa a probabilidade do usuário parar de clicar em links aleatórios e requisitar uma nova página aleatória, reiniciando o processo. Em geral, o *damping factor* é fixado em 85%, ou seja, há 15% de chance que o usuário modelado requisite uma nova página ao invés de clicar em um link.

Na prática, ao longo dos anos, o Google desenvolveu diversos modelos alternativos e algoritmos auxiliares para refinar o cálculo do PageRank. Dentre eles, podemos citar versões mais inteligentes do *random surfer* que navegam entre páginas de acordo com uma dada consulta feita pelo usuário ao invés de aleatoriamente, assim como algoritmos de segurança que modificam detalhes do trânsito das probabilidades entre um vértice e outro para evitar fraudes no cálculo por usuários mal-intencionados.

Algoritmos similares são utilizados em diversos sites e aplicações para determinar sugestões aos usuários. Por exemplo, no Twitter, uma versão alterada do PageRank sugere perfis que possam ser interessantes aos usuários e filtram os resultados das buscas performadas dentro do site. Porém, para que a utilidade dos resultados a cada usuário seja máxima, é necessário que o site utilize o algoritmo de forma personalizada a cada busca, levando em consideração os interesses do usuário (por exemplo, no Twitter, a busca deveria considerar os perfis que o usuário já segue e construir o grafo a partir disso). Apesar do algoritmo PageRank em si ter em geral complexidade linear de acordo com os criadores, a modelagem de uma rede de interesses personalizada gera um grafo extremamente grande a ser percorrido, o que tornaria as buscas bastante demoradas.

Por isso, sites como o Twitter utilizam o PageRank de forma não-personalizada nas suas buscas, comprometendo a qualidade dos resultados. Buscando resolver esse problema, em 2014 foi proposto um algoritmo chamado FAST-PPR (sendo PPR a sigla em inglês para PageRank personalizado), que é capaz de estimar o PageRank personalizado em relação a um usuário específico de 20 a 160 vezes mais rapidamente do que os algoritmos anteriores .

O objetivo do FAST-PPR é encontrar múltiplos caminhos de um vértice *s* (o usuário em relação ao qual o PageRank é personalizado) a um vértice *t* (a página cujo PageRank personalizado se deseja estimar), calcular o PageRank de *t* seguindo cada caminho e usar os resultados para estimar o PageRank de *t* relacionado a *s*. O algoritmo é baseado em busca bidirecional: primeiro ele parte de *t* e segue as arestas para trás, estimando o PageRank dos predecessores de *t* até uma certa fronteira e, em seguida, busca caminhos de *s* até essa fronteira, calculando o PageRank dos vértices ao longo do caminho e utilizando-os juntamente com as estimativas dos PageRanks depois da fronteira para estimar o PageRank de *t*. De acordo com os criadores, a exatidão da estimativa é alta.

# Metodologia

A metodologia utilizada tem o propósito de verificr como os diferentes algoritmos lidam com as diferentes situações. Vamos discutir a lógica de cada algoritmo e verifica sua complexidade em tempo e espaço.

Vamos analizar como cada algoritmo lida com dados distribuídos, grafos esparsos, grafos largos, introdução e remoção de nós, pesquisa, habilidade de customização e limitações de tempo e de espaço.

Diferentes cenários comuns serão establecidos para este propósito. Baseando-se como um algoritmo se enaixa nesse cenário, vamos ranquear as diferentes versões do Page Rank e explicar como estas podem ser aprimoradas ou combinadas a fim de prover melhores resultados.

##### Referências

1. Page, Lawrence; Brin, Sergey; Motwani, Rajeev; Winograd, Terry “The PageRank citation ranking: Bringing order to the Web” Available: http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf
2. Peter Lofgren, Siddhartha Banerjee, Ashish Goel,C. Seshadhri “FAST-PPR: scaling personalized PageRank estimation for large graphs” (KDD 2014 Presentation) Available: https://arxiv.org/abs/1404.3181
3. Page, Lawrence; Brin, Sergey; “The Anatomy of a Large-Scale Hypertextual Web Search Engine” Available: http://infolab.stanford.edu/pub/papers/google.pdf