LA-UR-05-6469

Approved for public release;
distribution is unlimited.

Title: SIMULATING NETWORK INFLUENCE ALGORITHMS USING PARTICLE-SWARMS: PAGERANK AND PAGERANK-PRIORS

Author(s): Marko A. Rodriguez
Johan Bollen

Submitted to: Journal of Complexity

Simulating Network Influence Algorithms Using Particle-Swarms: PageRank and PageRank-Priors

Marko A. Rodriguez [a,b,c,*] Johan Bollen [a]

[a] *Research Library, Los Alamos National Laboratory*
Los Alamos, New Mexico 87545

[b] *Computer Science Department, University of California at Santa Cruz*
Santa Cruz, California 95064

[c] *Center for Evolution, Complexity, and Cognition, Vrije Universiteit Brussel*
Brussel, Belgium 1050

Abstract

A particle-swarm is a set of indivisible processing elements that traverse a network in order to perform a distributed function. This paper will describe a particular implementation of a particle-swarm that can simulate the behavior of the popular PageRank algorithm in both its *global-rank* and *relative-rank* incarnations. PageRank is compared against the particle-swarm method on artificially generated scale-free networks of 1,000 nodes constructed using a common gamma value, $\gamma = 2.5$. The running time of the particle-swarm algorithm is $O(|P| + |P|t)$ where $|P|$ is the size of the particle population and t is the number of particle propagation iterations. The particle-swarm method is shown to be beneficial in its ease of extension and running time.

Key words: Particle-Swarm, PageRank, PageRank-Priors, In-Degree, Impact Metrics

1 Introduction

Influence, prestige, impact, and authority are all terms that refer to a class of network metrics that utilize the structure of a graph, $G = \{N, E\}$, to derive an influence ranking, $\vec{I} \in \mathbb{R}^{|N|}$, over all its constituent nodes. Generally these metrics determine a node's importance in a recursive fashion. A node's influence, $\vec{I}_k$, is a function of the influence of the nodes that project to it. This idea is represented in

* Corresponing author.
Email addresses: marko@lanl.gov (Marko A. Rodriguez), jbollen@lanl.gov (Johan Bollen).

Eq. (1), where $e_{j,k}$ is a directed edge from n_j to n_k, $\mathrm{out}(n_j)$ is the set of outgoing edges from node n_j, and t is the current iteration represented in discrete time. The collection of influences across all nodes in the network is represented by the vector $\vec{I}$ which is the principle eigenvector of the adjacency matrix formed by the graph (1).

$$\vec{I}_{k,t+1} = \sum_{\forall e_{j,k} \in E} \frac{\vec{I}_{j,t}}{|\mathrm{out}(n_j)|} \tag{1}$$

Since the inception of these algorithms there has been a strong focus on *global-rank*, $I(n|N)$ or simply $I(n)$, and only recently has there been research interest in *relative-rank* $I(n|R)$, where $R \subseteq N$ (2). Global-rank determines the relative influence of each node with respect to the entire node population, N, while on the other hand, relative-rank determines the relative influence of each node with respect to a particular subset of the network, $R \subseteq N$. Global-rank algorithms have found themselves at the forefront of web search techniques: PageRank (1), HITS (3), and their respective extensions. While on the other hand, biased, or relative ranking has found application in domain-specific authority using web-page networks (4), company-specific idea influence using collaboration networks (2), and manuscript-specific peer-review influence using co-authorship networks (5). It is important to note that global-rank can be interpreted as a special case of relative-rank where each node's influence is calculated relative to a root node set that is the entire node population, $R = N$.

The contribution set forth by this paper is two fold. First, this paper demonstrates the application of particle-swarms to the calculation of these two popular influence metrics: PageRank (global-rank) (1) and PageRank-Priors (relative-rank) (2). The particle-swarm algorithm is beneficial in its running time and flexibility. Unlike, the matrix models of most popular metrics, a particle-swarm has a more tangible appeal that lends itself towards various functional modifications. This paper will only provide the rudimentary data structures and functions necessary to simulate PageRank and PageRank-Priors, but will provide room in the framework for possible extensions. The second contribution of this paper is that it provides an introduction to the use of particle-swarms in the broader context of graph analysis and manipulation. Currently there is little research in this area. Of those manuscripts found, most of them analyze graphs from the perspective of a single random-walker and do no include more advanced functions and properties such as particle energy, decay, and teleportation (6) (7) (8) (9).

The outline of the paper is as follows. *Section 2* will discuss both PageRank and PageRank-Priors from the standpoint of an object-oriented random-walker model.

Section 3 will then describe the graph theoretic model of the particle-swarm method with emphasis on the various parameters and functions of the particles as they apply to simulating PageRank and PageRank-Priors. *Section 4* compares both PageRank algorithms and the particle-swarm algorithm on artificially generated scale-free networks. Finally, *Section 5* discusses the running-time of the particle-swarm method and two optimizations. The paper concludes, *Section 6*, with a short discussion of related PageRank algorithm implementations.

2 Random-Walker Model

Both PageRank (1) and PageRank-Priors (2) can be described in a random-walk fashion where a stochastic token, or particle, moves throughtout a network, G. The rank influence of any node $n_k \in N$ is the probability that that particle-token, p, will be seen at that node, $\vec{I}_k = P(p|n_k)$. This conceptual analogy is explicitly represented within the object-oriented framework of this paper as a swarm of particle-tokens, P, that traverse the network landscape depositing their energy footprint on each node they traverse. In doing so, the particles generate an influence ranking of the nodes in terms of the node population's normalized energy distribution, $\vec{I}$. This particle-swarm model can reach near perfect correlations with both PageRank and PageRank-Priors with a more efficient running-time.

2.1 PageRank Walker

The PageRank algorithm, as described in (1), was the driving force which has carried the Google search engine to the forefront of web search-engine technology. Simply speaking, the algorithm is calculated in a recursive fashion where a particular page in a network of web-pages is influential if it is referenced by, or linked from, other influential pages. Imagine a random-walker, p, traversing a network of web-pages such as the World Wide Web, $G = \{N, E\}$. If that random-walker continously finds itself at a particular page n, then that random-walker is said to have a high probability of being at that web page. This probability is interpreted as the page's, or node's, influence. The random-walker is consitently located at that web-page because the incoming edges to n_k, $\text{in}(n_k) \subseteq E$, are either numerous, nearing the limit $|\text{in}(n_k)| \approx |E|$, or the nodes that point to n_k have a numerous set of incoming edges which allow the random-walker to consistently reappear at n_k. Taken to its recurssive limit, a node's influence is a measure of all the aggregate influence it receives from pages pointing to it whether direct or indirect. A modification to this algorithm incorporates a dampening-factor, $\lambda \in [0, 1]$, to reduce the spread of influence over time (10). The further the random-walker travels, the less influence the random-walker should have, such that at full dampening, $\lambda = 1.0$, the random-walker can not take a step and all nodes are ranked equivalent, $\vec{I}_{k,t=0} = \frac{1}{|N|}$. The

combination of random-walker propagation and dampening is expressed in Eq. (2). The equation represents the proportion of influence distributed to n_u by n_v. This can also be interpreted as the probability of the random walker taking the edge $e_{v,u}$ given the the condition that its current location is n_v.

$$p(u|v) = \left(\frac{1-\lambda}{\text{out}(n_v)}\right) + \left(\frac{\lambda}{|N|}\right) \tag{2}$$

2.2 PageRank-Priors Walker

The priors idea was first proposed by (2) in their formalization of a relative-rank extension to both PageRank and HITS. Suppose the network data structure, $G = \{N, E\}$, is supplied with a root node set, $R \subseteq N$. This root set is the set of nodes used to rank all other nodes relative too. Suppose that at each time step, the random-walker has a probability, β, of 'teleporting' to particular node $r \in R$ as defined by the probability distribution, $P(r) = \frac{1}{|R|}$. A variation to the algorithm can bias the probability distribution over R. As β approaches 1.0 the probability of seeing the random-walker at any node in R becomes greater and therefore the influence of the nodes in R, as well as those nodes that R projects to, increases. At the limit when $\beta = 1.0$, the influence distribution of all $n \notin R = 0.0$ and the influence of all $n \in R = \frac{1}{|R|}$. In this way, the random-walker is biasing the ranking of the network nodes, N, towards the subset R. When $\beta = 0.0$ there is still a bias towards the initial root node set since the random-walker will initiate its walk from that set, but the probability of the random-walker's location diffuses over the network the more time steps alloted.

The next section will now extend the random-walker model to a particle-swarm model where a collection of random-walkers, P, traverse the network depositing an energy footprint at each step of the way. These energy footprints, as stored in the node's 'memory', $\vec{I_k}$, represent the probability of having a particle at that particluar node. It is important to note that the random-walker model can be easily extended to account for weighted graphs, $G = \{N, E, W\}$, where the outgoing edges of a node are normalized to create a probability distribution. This probability distribution biases the random-walkers decision when taking an outgoing edge and in such cases is called a biased random-walker. In this way, weighted PageRank and weighted PageRank-Priors can be calculated. The next section will discuss the full weighted model of the particle-swarm framework though the simulations are only for the PageRank and PageRank-Priors non-weighted counterparts.

3 Particle-Swarm Model

A particle-swarm, P, is a collection of unique processing entities that, by traversing a network in a stochastic manner, collectively perform a distributed function. In relation to the random-walker model, a particle-swarm is simply a collection of many random-walkers. The unification of the network particles, nodes, roots, edges, and weights form the data structure $G = \{P, N, R, E, W\}$ where each edge is assigned a weight, $|E| = |W|$, and $R \subseteq N$. A single particle, $p_i \in P$, can contain any number of properties and behaviors, but for the purposes of this paper only those properties and behaviors that apply to PageRank and PageRank-Priors are described, $P = \{\epsilon, \delta, r, \beta, c\}$. A particle is an indivisible entity, but its local energy content, $\epsilon_i \in [0, 1]$, is not. Each time a particle traverses an edge, its local energy content is affected by a decay-scalar, $\delta_i \in [0, 1]$, which is related to the dampening factor, λ, described previous. To simulate PageRank-Priors a particle must have a reference to its originating, or root node, $h_i \in R$, so that it can 'teleport' home as detemined by a back-probability, $\beta_i \in [0, 1]$ and a back selection function $B(\beta_i) \in \{0, 1\}$. Finally a particle traverses an outgoing edge from its current node location, $c_i \in N$, according to an edge selection function, $\theta(\mathrm{out}(c_i)) \in \mathrm{out}(c_i)$. These properties and behaviors are enumerated below for ease of reference. Note that δ and β are the same for every particle in the following simulations, $\delta_i = \delta_l$ and $\beta_i = \beta_l$. Extensions to this framework can assign different values to different particles.

(1) ϵ: a local energy value $\epsilon \in [0, 1]$
(2) δ: a energy decay-scalar $\delta \in [0, 1]$
(3) h: a reference to its home, or root, node $h \in R$
(4) β: a back-probability $\beta \in [0, 1]$
(5) c: a reference to the current node location $c \in N$
(6) a probabilistic back selection function $B(\beta) \in \{0, 1\}$
(7) a probabilistic outgoing edge selection function $\theta(\mathrm{out}(c)) \in \mathrm{out}(c)$

A network node, n_k, is represented by the triplet $\{P(n_k), \mathrm{out}(n), \vartheta_k\}$ where $P(n_k)$ is a unique set of particles located at n_k, $\mathrm{out}(n_k)$ is a unique set of outgoing edges from n_k, and $\vartheta \in \mathbb{R}$ is n_k's local energy value. Any edge in the network, $e_{k,j}$, is a directed edge, from n_k to n_j, with an associated weight, $w_{k,j} \in [0, 1]$. The weights of the set of all outgoing edges from any node, $\mathrm{out}(n_k)$, must be normalized to create a probability distribution for each particle's propagation function (Eq. 3).

$$w_{k,j_{(t+1)}} = \frac{w_{k,j_{(t)}}}{\sum_{i=0}^{|\mathrm{out}(n_k)|} w_{k,i_{(t)}}} \tag{3}$$

Initially a set of nodes in the network are seeded with a collection of particles, P. This distribution can be an equal distribution or a biased distribution depending on the desired functional output. For global-rank metrics, each node in the network is provided with an equal initial distribution, $|P(n_k)| = \frac{|P|}{|N|}$, while for relative-rank methods, only an initial root set, $R \subseteq N$, will be provided with particles, $|P(r_k)| = \frac{|P|}{|R|}$ where $r_k \in R$. At each time step of the algorithm a particle performs three behaviors. First, the particle increments its current node's energy content, ϑ_k, with its current energy content, ϵ_i, by way of $\vartheta_{k(t+1)} = \vartheta_{k(t)} + \epsilon_{i(t)}$ where $n_k = c_i$ (Alg. 1-16). Next, the particle decays its energy content by the parameterized decay-scalar, δ_i (Eq. 4, Alg. 1-17).

$$\epsilon_{i(t+1)} = \epsilon_{i(t)} - (\delta_i \epsilon_{i(t)}) \tag{4}$$

Lastly, the particle calculates $B(\beta)$ (Alg. 1-18). If the function returns 1, then the node will return home, $c_{i(t+1)} = h_i$. If the function returns 0, then the particle chooses an outgoing edge of its local node depending on the probability of that node's outgoing edge weights, $c_{i(t+1)} = \theta(\mathrm{out}(c_i))$ (Alg. 1-26). The outgoing edge chosen, $e_{i,j}$, determines the particles new nodal reference, $c_{i(t+1)} = n_j$. A particle's death occurs when $\epsilon_i = 0.0$. Since the decay function of the particle is based on the percentage of its current energy content, formally the particle energy will approach, but never reach 0.0. Therefore, a threshold for particle death is given when $\epsilon_i \leq 10^{-8}$. Unlike the 'random teleport' functionality of most PageRank implementations, if node c_i does not have an outgoing edge, then the particle is destroyed, $\epsilon_i = 10^{-8}$ (Alg. 1-22). Once all the particles in the network have died or a desired t has been reached the particle propagation algorithm is complete. The energy content, ϑ_k, of all nodes can be normalized to yield the proportion of energy every node has with respect to one another. This proportion can be interpreted as the probability of seeing a random-walker at that particular node. The aggregated values of all energy in the network forms the influence vector $\vec{I}$.

The pseudocode for the particle-swarm implementation of PageRank is provided in Algorithm 1. The first functional block expresses a particle-distribution algorithm and the second block expresses the particle-propagation algorithm. To implement PageRank-Priors the loop on line 3 should run through R not N and a desired β should be set at line 6. An overview of the different Big-O running times of the two functions are presented in their respective comments and will be examined more closely in the *Section 5*.

The next section will provide simulation results of the aforementioned particle-swarm algorithm, with varying parameters. The results of these simulations are compared to the results given by PageRank, PageRank-Priors, and In-Degree.

```
#distribute particles: O(|N|particlesPerNode) = O(|P|);
int i = 0;
foreach (n_k ∈ N) do
    int particlesPerNode = 10;
    for (l = 0, l < particlesPerNode, l++) do
        ε_i = 1.0; δ_i = 0.15; r_i = n_k; β_i = 0.0; c_i = n_k;
        i++;
    end
end
#disseminate particles: O(|P|t);
int t = 0;
while (t < pageIterations) do
    t++;
    for (i = 0, i < |P|, i++) do
        if (ε_i > 10^−8) then
            ϑ_{c_i} = ϑ_{c_i} + ε_i;
            ε_i = ε_i − (δ_i ∗ ε_i);
            if (B(β_i) == 1) then
                c_i = r_i;
            end
            else
                if (|θ(out(c_i))| == 0) then
                    ε_i = 10^−8
                end
                else
                    c_i = θ(out(c_i));
                end
            end
        end
    end
end
```

Algorithm 1: Particle-Swarm implementation of PageRank

4 Simulation Correlations

This algorithm test suite was originally run on random networks and scale-free networks of a varying $\gamma \in [2.0, 3.0]$ and size $|N| \in [100, 10000]$ with insignificant variation on the particle-swarm's simulation peformance. Since the network size and topology are not dimensions for analysis, only a collection of scale-free networks of $\gamma = 2.5$ and $|N| = 1000$ are used for the remainder of the paper. For scale-free construction, each node is given a predetermined set size for their incoming connections as defined by Eq. (5), where the random number $\psi \in [0, 1]$, $|\text{in}(n_k)| \leq |N| - 1$, and $\text{in}(n_k)$ is the set of incomming edges to n_k (11).

$$|\text{in}(n_k)| = \lfloor \psi^{-[1.0/(\gamma-1.0)]} \rfloor \tag{5}$$

From here nodes randomly connect to each other until their maximum incoming connectivity is reached, at which point the network construction algorithm is complete. By predetermining the maximum incoming connectivity of a node, the topology of the network maintains a small portion of node hubs and a relatively large portion of sparsely connected nodes which is characteristic of many naturally occuring networks (12).

4.1 In-Degree as a Trivial Case of PageRank and Particle-Swarm

The trivial case of the random-walker model is when the random-walker is only allowed to take one step. This is a method for calculating the influence of a node with respects to In-Degree and is an extreme case of PageRank as $\lambda \rightarrow 1.0$ and $\delta \rightarrow 1.0$ or the algorithm is halted at $t = 1$. To simulate In-Degree, each edge in the network must be traversed at $t = 1$. To accomplish this, every node is supplied with a collection of random-walkers proportional to its outgoing edge size, $|P(n_j)| = \alpha|\text{out}(n_j)|$ where $\alpha \in \mathbb{N}^+$. Now if each random-walker has an equal probability of taking any outgoing edge, then at $t = 1$ the distribution of random-walkers across the set of nodes N is the In-Degree influence of that node (Eq. 6).

$$\vec{I_k} = \sum_{\forall e_{j,k} \in E} \frac{|P(n_j)|}{|out(n_j)|} \tag{6}$$

Since the set of all $e_{j,k} \equiv \text{in}(n_k)$, then when substituting $|P(n_j)|$ for $\alpha|\text{out}(n_j)|$ yields $\vec{I_k} = \alpha|\text{in}(n_k)|$ and therefore produces an influence calculation perfectly correlated to In-Degree. Given that this is a random-walker, stochastic noise will disrupt the probability that each outgoing edge of every node is taken once and only once. Therefore as the size of the intital distribution of particles increases (as α increases), but at the same time remaining proportionally equal for every node, the noise is reduced and the appropriate In-Degree influence vector is returned. If the distribution of random-walkers is equal, $|P(n_k)| = \frac{|P|}{|N|}$, then only an approximation of In-Degree can occur. In such cases, the more uniform the distribution of outgoing edges of all the nodes, the more accurate the approximation.

To simulate In-Degree influence using PagePank, λ was scaled between 0.005 and 0.995 to produce the following correlation plot (Fig. 1a). The reason for limiting the experiment to $\lambda = 0.995$ is because when $\lambda = 1.0$ there is no deviation in

the rank vector because $\vec{I}_k = \frac{1}{|N|}$. It is shown that PageRank best approximates In-Degree influence at the limit as $\lambda \to 1.0$. For example, at $\lambda = 0.995$, $C = 0.998$. Next, the particle-swarm method for simulating In-Degree was determined using various initial particle distribution sizes of $|P(n_k)| \in [1, 20]$, $|P| \in [1000, 20000]$, and $\beta = 0.0$. The importance of $|P|$ will become apparent in the *Section 5* when the particle-swarm's running time is discussed. The δ of each particle was scaled from 0.005 to 0.995 and as $\delta \to 1.0$, $\delta = 0.995$, In-Degree influence is approximated most closely, $C = 0.997$ (Fig. 1b). Figure 1b has 20 superimposed particle distribution size plots. The following influence vector relationship exists between these three algorithms: $\vec{I}_{\text{IN}} \approx \vec{I}_{\lambda \to 1.0} \approx \vec{I}_{\delta \to 1.0}$. Notice that PageRank and the particle-swarm method are nearly equivalent in their behavior for the respective $\delta = \lambda$ values, $\vec{I}_\lambda \approx \vec{I}_\delta$ when $|P(n_k)| > 1$. Also note that the divergent plot in Figure 1b occurs when $|P(n_k)| = 1$, $|P| = 1000$.

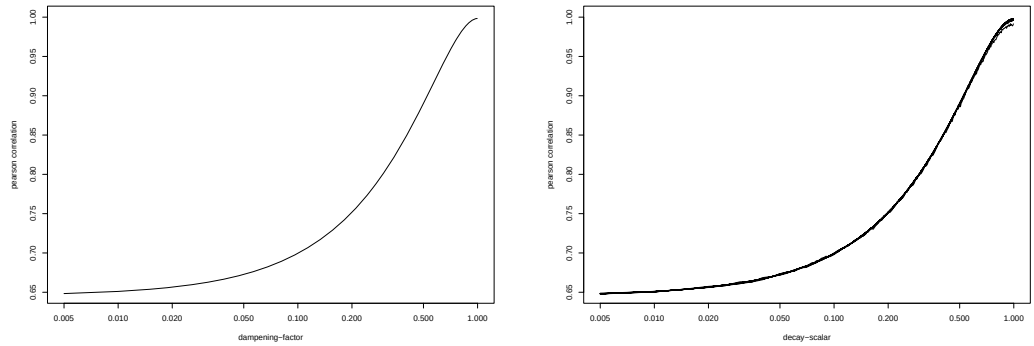

Fig. 1. **a.** PR vs. IN over $\lambda \in [0.005, 0.995]$ **b.** PS vs. IN over $\delta \in [0.005, 0.995]$

4.2 Correlating Particle-Swarm to PageRank and PageRank-Priors

To simulate the results of PageRank (global-rank), the decay-scalar δ was varied between 0.005 and 0.995 for every potential dampening factor λ between 0.005 and 0.995. The iterations of the particle-swarm method were constrained to $t_{PS} = t_{PR}$, where t_{PS} and t_{PR} are the amount of iterations for the particle-swarm method and PageRank, respectively. Note that when δ is high, particle death occurs before the amount of iterations is complete. For this experiment $|P(n_k)| = 10$, $|P| = 10000$. The resulting figure, (Fig. 2a), demonstrates that an equal distribution of particles across all of N with $\beta = 0.0$ simulates the respective PageRank calculation with a near 1.0 Pearson correlation when $\delta = \lambda$.

PageRank-Priors (relative-rank), on the other hand, is a function of two variables, the size of the root node set, R, and the back-probability, β. The root node set was determined by randomly assigning a portion of the node population to R, $R =$

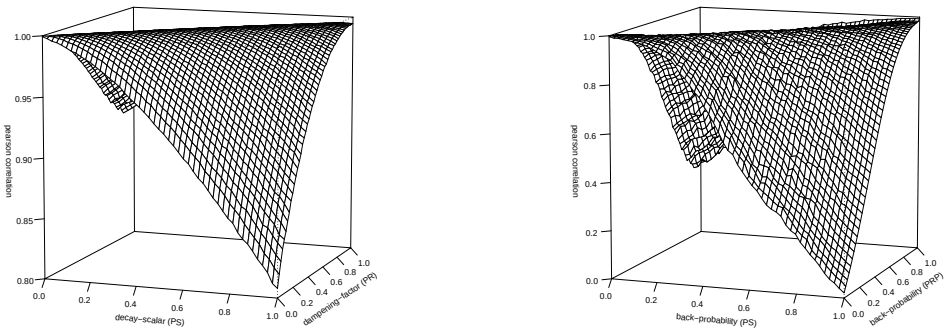

Fig. 2. **a.** PR vs. PS over δ and λ **b.** PRP vs. PS over β

$f(N, \varphi)$ where the percentage $\varphi \in [0.01, 1.0]$ and $|R| = \varphi|N|$. The selection of the root node set had no effect on the correlation between PageRank-Priors and the particle-swarm method. Therefore, to represent the correlations in a 3-D plot, the φ factor was omitted. The iterations of the particle-swarm method were constrained to $t_{PS} = t_{PRP}$ where t_{PRP} is the amount of iterations required for PageRank-Priors to converge. Furthermore, $\delta = 0.0$ since PageRank-Priors has no dampening-factor parameter. Figure 2b provides the correlations values when the particle-swarms $\beta_{\mathrm{PS}} \in [0.1, 1.0]$ for all $\beta_{\mathrm{PRP}} \in [0.1, 1.0]$ of PageRank-Priors. The root node set was generated from 10% of the node population, $\varphi = 0.10$, therefore when $|P(r_k)| = 10$, $|P| = 1000$. PageRank-Priors and the particle-swarm method have near perfect correlation when $R_{\mathrm{PRP}} = R_{\mathrm{PS}}$ and $\beta_{\mathrm{PRP}} = \beta_{\mathrm{PS}}$.

5 Optimizations and Running Time

This section will extend the current particle-swarm model to express two particular optimizations: *iteration constraining* and *random seeding*. Currently, the running time of the particle-swarm method is $O(|P| + |P|t)$ where $|P|$ is the number of particles used in the simulation, and t is the number of particle propagation iterations. In comparison, the running time of both PageRank and PageRank-Priors is $O(|E|t)$ where E is the set of edges in the network and t is the number of iterations required till convergence (13) (14). It is important to note that $|P|$ is a function of $|N|$, $|P| = \alpha|N|$, not $|E|$, and for most real-world networks $|N| << |E|$. An accurate particle-swarm simulation of PageRank is possible when $|P(n_k)| = 1$ and therefore $|P| = 1000$. While for a $\gamma = 2.5$ scale-free network of 1000 nodes $|E| \approx 2575$. Therefore, the Big-O speed up, given 20 iterations for each algorithm, is a factor of approximately $2.45 = \frac{(2575)(20)}{(1000)+(1000)(20)}$, $\frac{|E|t}{|P|+|P|t}$.

Greater gains are seen in the particle-swarms simulation of PageRank-Priors when

$|R| < |N|$. Since the particle population of a node is a proportion of the total population, $|P(n_k)| = \frac{|P|}{|R|}$, then this ratio allows for a smaller particle population when simulating PageRank-Priors without degrading the accuracy of the calculation. Therefore, $|P(r_k)| = \frac{|P_{\mathrm{PRP}}|}{|R|} = \frac{|P_{\mathrm{PR}}|}{|N|}$, where P_{PRP} and P_{PR} are the particle sets for PageRank-Priors and PageRank, respectively. For $|P| = |R|$, the particle-swarm algorithms has a running time of $O(|R| + |R|t)$ when simulating PageRank-Priors. The PageRank-Priors particle-swarm simulation is more efficient in terms of running time than its originally, and only, published analysis of $O(|E|t)$ (2). The benefits of the particle-swarm simulation of PageRank-Priors are best realized when $|R| << |N| << |E|$.

These calculations assume that the particle-swarm method and PageRank/PageRank-Priors both share the same amount of interations, $t_{PS} = t_{PR}$, and that the particle-swarm method has a homogenous initial particle seeding of at least 1 particle per node. Both of these parameters can be reduced to lower the particle-swarms running time with varying effects on the correlation. The following list of variables will be discussed in the following subsections and are presented here for ease of reference.

(1) t_{PS}: number of iterations to propagate particles $t \in \mathbb{N}^+$
(2) ϕ: proportion of nodes to receive an initial seeding of particles $\phi \in [0, 1]$
(3) α: number of particles per node in the initial seeding $\alpha \in \mathbb{N}^+$
(4) S: the set of nodes receiving particle from the intitial seeding $S \subseteq N$ and $|S| = \phi|N|$

5.1 Constraining Particle Iterations and Random Particle Seeding

Algorithm 1-12 assumes that a particle propagates for the same amount of iterations as PageRank, $t_{PS} = t_{PR}$. This isn't the best method for setting t_{PS} since it requires PageRank to be executed in order to determine the amount of iterations required. Another way of determining the amount of interations for the particle-swarm method is to wait until all particles have died, which occurs when the particle's energy content has decayed to $\epsilon_i = 10^{-8}$ or when c_i no longer has outgoing edges. For a $\delta = 0.15$ and when c_i always has at least one outgoing edge, particle death occurs after 113 iterations, while the average PageRank converges after 22.7 interations on a $\gamma = 2.5$ scale-free network. This obviously is not the fastest method. Therefore, Figure 3a plots the correlation between the particle-swarm method and PageRank as the particle-swarm method's iteration value is constrained, $t_{PS} \in [1, 25]$. The range from $25 < t_{PS} \leq 113$ is omitted due to insignificant variation in the algorithm's behavior. The result demonstrates that the particle-swarm method is strongly correlated with PageRank, $C = 0.953$, after only

4 iterations, $t_{PS} = 4$.

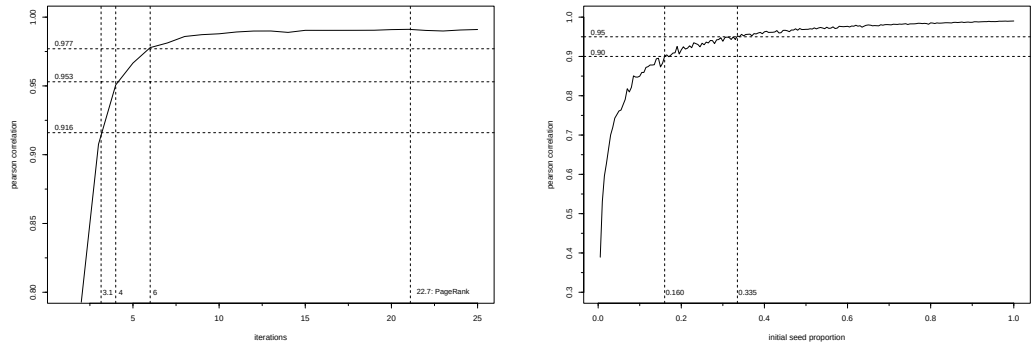

Fig. 3. **a.** PS vs. PR over $t \in [1, 25]$ **b.** PS vs PR over $\phi \in [0, 1]$

The particle-swarm method can also be optimized by randomly choosing a subset of the network to initially seed with particles, $S \subseteq N$. This random subset can be expressed as a proportion of the whole network, $\phi|N|$ where $\phi \in [0, 1]$ and $|S| = \phi|N|$. Figure 3b plots the correlation between PageRank and the particle-swarm method for different initial particle seed proportions. It is shown that at $\phi = 0.335$, when only 33.5% of the nodes in the network are seeded with a single particle, the Pearson correlation is approximately 0.95. Therefore an accurate PageRank calculation does not require all nodes to begin with an equal set of particles. Thus, $|P| << |N|$.

5.2 Combining the Optimizations

The combination of both optimizations is represented in Figure 4 where each initial seed proportion, $\phi \in [0.01, 0.5]$, is calulated for every iteration amount, $t_{PS} \in [1, 25]$. Next, Figure 5 plots the iteration amount against the seeding proportion for the lowest value pair obtaining a $C \approx 0.95$. Each plot point's shade value is calculated as ϕt, which represents the cost of performing that parameter pair to obtain a $C \approx 0.95$. Therefore, to achieve a $C \approx 0.95$, the most computationally efficient way is to use a moderate amount of particles ($\phi \approx 0.45$) propagated over a moderate amount of time steps ($t_{PS} \approx 8$).

The speed-up of the particle-swarm method with respects to PageRank is represented in Eq. (7) as Φ. Since $\phi|N|\alpha$ represents the particle population, the full running time can still be expressed as $O(|P| + |P|t_{PS})$. The numerator in Eq. (7) is based on the standard PageRank implementation of $O(|E|t_{PR})$.

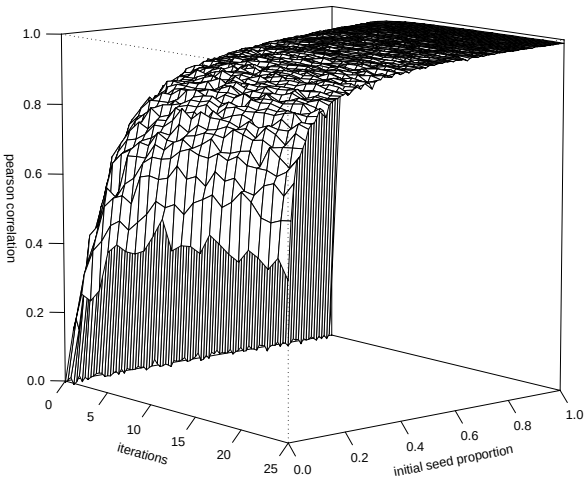

Fig. 4. PS vs. PR where PS combines iteration constraining and random seeding

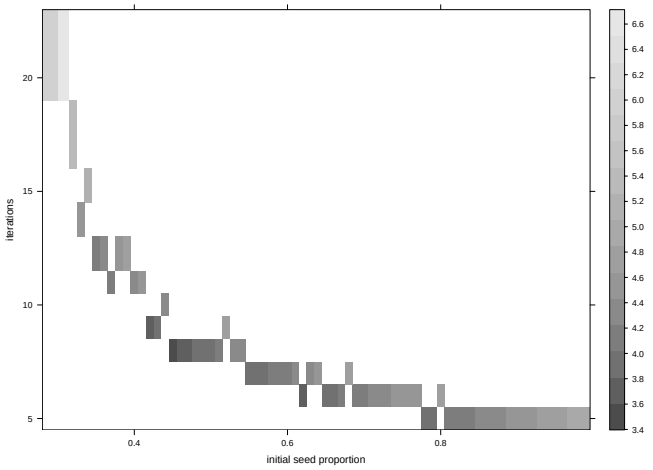

Fig. 5. ϕt for various iteration/seed proportion pairs at $C \approx 0.95$

$$\Phi = \frac{|E| t_{PR}}{(\phi |N| \alpha) + (\phi |N| \alpha)\, t_{PS}} \tag{7}$$

For a $\gamma = 2.5$ scale-free network of $|N| = 1000$, the theoretical speed-up of the fastest particle-swarm method yielding a $C \approx 0.95$ ($\alpha = 1$, $\phi = 0.45$, $t = 8$) is calculated to be $14.43 = \frac{(2575)(22.7)}{[(0.45)(1000)(1)]+[(0.45)(1000)(1)(8)]}$. To verify this hypothesis, PageRank, as implemented in (14), was compared against the most optimized particle-swarm method. The benchmark testing was done over 500 trials of 500

different $\gamma = 2.5$ scale-free networks of $|N| = 1000$ with the average speed-up factor determined to be 22.23. A potential explanation for the increased benchmark speed-up relative to the theoretical speed-up may be in part to the fact that over the course of the particle-swarm algorithm, particles die before all iterations are complete (Alg. 1-15,17,23). Therefore, the general rule is that as t increases, $|P|$ decreases.

Given different gamma values, the amount of iterations should vary. For example, a $\gamma = 2.0$ scale-free network only requires 12.52 iterations for PageRank to converge. Similarly, The particle-swarm method requires only 1.01 iterations to produce a $C \approx 0.95$. At the other extreme, a $\gamma = 3.0$ scale-free network requires approximately 28.88 iterations to converge while the particle-swarm method requires 6.23 iterations. The general trend, for producing a $C \approx 0.95$, is $t_{PS} \approx \frac{1}{5} t_{PR}$ or for each $\gamma \in [2.0, 3.0]$, $t_{PS} \approx 2\gamma$.

6 Conclusion

Due to the popularity of the the global-rank implementation of PageRank there exists much literature on efficient implementations of the algorithm. One particulary example includes an algorithm that partitions the graph into related influence clusters (15). Unfortunately, this publication does not represent the algorithm's running times in terms of Big-O notation and only provides 'wall time' for a specific machine architecture. For comparison, the graph clustering method states a Spearman correlation of 0.95 and a 2 fold increase in calculation time relative to a 'highly optimized' implementation of PageRank. The graph clustering method groups nodes of a similar PageRank into a hyper-node and then calculates the full converging PageRank vector on the newly constructed hyper-network. In this way, the clustering method is able to reduce the total amount of edges, E, iterated over. The publication states that the typical edge reduction between the original network and the hyper-network is a factor of 20 for networks containing billions of edges. Edge reduction, by way of node grouping, also reduces the amount of nodes in the networks. Therefore, there is a strong incentive to combine the graph clustering method and the particle-swarm method. This has not been tested as of yet.

Finally the space contraints of the particle-swarm method are larger than traditional matrix methods since these methods do not represent particles, only the influence vector, $\vec{I}$, and the adjacency matrix of the graph. This representaiton lends itself towards efficient space modifications (16). The particle-swarm implementation discussed in this paper is calculated solely in main memory for small networks less than 10,000 nodes. This test-bed implementation is obviously not useful for calculations on web-sized networks. Future work will describe a system architecture for

performing particle-swarm algorithms on large-scale networks.

The particle-swarm method for graph analysis has an appeal in its potential for functional modification. From the object-oriented perspective, a particle can be seen as an 'agent' that can contain any number of properties and behaviors. The potential for modifying the particle-swarm framework presented in this paper can lead to a host of augmentations to the demonstrated influence metrics. One example includes the incorporation of 'negative' energy particles to reduce specific node influence as explained in (5). New particle-swarm metrics are currently being implemented and results will be presented in future publications. This paper's simulations were performed using the Confluence package (17). The Confluence API has been written such that new particles can be easily extended to the basic 'energy' particle framework.

7 Acknowledgments

The use of particle-swarms for network analysis was first introduced to the first author by Daniel Steinbock and then later applied in a joint paper on distributing voting influence within trust-based social-networks (18). Thanks to Carlos Gershenson and Francis Heylighen for many discussions on this topic. Finally, a special thanks to the producers of JGraph (www.jgraph.com) and Scott White's Java implementation of PageRank and PageRank-Priors. This work was partially funded by a GAANN Fellowship from the U.S. Department of Education and supported by Los Alamos National Labratory.

References

[1] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Computer Networks 30 (1998) 107–117.

[2] S. White, P. Smyth, Algorithms for estimating relative importance in networks, in: KDD '03: Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, NY, USA, 2003, pp. 266–275.

[3] J. M. Kleinberg, Authoritative sources in a hyperlinked environment, Journal of the ACM 46 (5) (1999) 604–632.

[4] T. H. Haveliwala, Topic-sensitive pagerank, in: Proceedings of the 11th International World Wide Web Conference, ACM Press, 2002, pp. 517–526.

[5] M. A. Rodriguez, H. V. de Sompel, J. Bollen, The convergence of digital-libraries and the peer-review process, Journal of Information Science In press.

[6] H. Zhou, Network landscape from a brownian particles perspective, Physical Review E 67 (4).

[7] M. E. Newman, A measure of betweenness centrality based on random walks, submitted to Social Networks.
URL http://arxiv.org/abs/cond-mat/0309045

[8] B. Tadic, Exploring complex graphs by random walks, in: Modeling Complex Systems AIP Conference Proceedings, 2003, p. 24.
URL http://arxiv.org/abs/cond-mat/0310014

[9] J. Noh, H. Reiger, Random walks on complex networks, arXiv preprint in condensed matter cond-mat/0307719.
URL http://arxiv.org/abs/cond-mat/0307719

[10] S. Brin, R. Motwani, L. Page, T. Winograd, What can you do with a web in your pocket?, Data Engineering Bulletin 21 (2) (1998) 37–47.
URL citeseer.ist.psu.edu/brin98what.html

[11] M. Aldana, Boolean dynamics of networks with scale-free topology, Physica D 185 (1) (2003) 45–66.
URL http://dx.doi.org/10.1016/S0167-2789(03)00174-X

[12] A. L. Barabasi, Linked: The New Science of Networks, Perseus Publishing, 2002.

[13] M. Bianchini, M. Gori, F. Scarselli, Inside pagerank, ACM Trans. Inter. Tech. 5 (1) (2005) 92–128.

[14] J. O'Madadhain, D. Fisher, T. Nelson, J. Krefeldt, Jung: Java universal network/graph framework (2005).
URL http://jung.sourceforge.net/

[15] A. Broder, R. Lempel, F. Maghoul, J. Pedersen, Efficient pagerank approximation via graph aggregation, in: Proceedings of the 13th International World Wide Web Conference, ACM Press, 2004, pp. 484–485.

[16] T. Haveliwala, Efficient computation of pagerank, Stanford Technical Report.
URL http://dbpubs.stanford.edu:8090/pub/1999-31

[17] M. A. Rodriguez, Confluence: A particle-swarm package for java (2005).
URL http://www.soe.ucsc.edu/~okram/confluence.html

[18] M. A. Rodriguez, D. Steinbock, A social network for societal-scale decision-making systems, in: NAACSOS '04: Proceedingss of the North American Association for Computational Social and Organizational Science Conference, Pittsburgh, PA, USA, 2004.
URL http://arxiv.org/abs/cs.CY/0412047