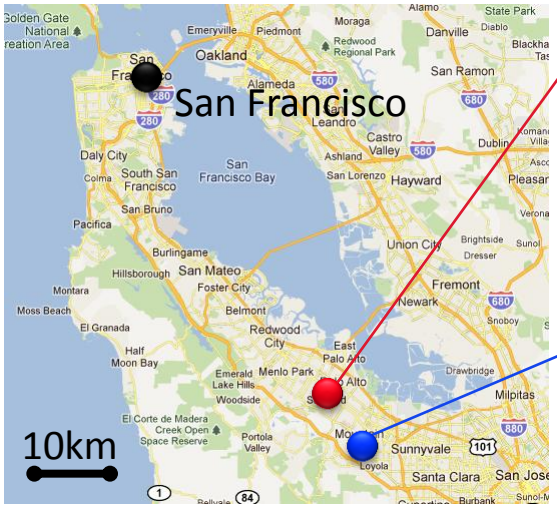
PageRankの効率的計算

Haveliwala, T.H., Efficient Computation of PageRank, Stanford Univ. Technical Report, 1999.

(大規模なハイパーテキスト的な Web 検索エンジンに関する解剖
Brin, S., Page, L., The anatomy of a large-scale hypertextual web search engine, In Proceedings of the Seventh International World Wide Web Conference, 1998.)

(California, US)



Stanford

Mountain View

Google

2011.10.17論文集中ゼミ
B4 伊藤 創太

目次

Efficient Computation of PageRank

1. 序論 : PageRankアルゴリズムの背景と概要
2. PageRankのアルゴリズム : 具体的な計算方法
3. 効率的なメモリ使用 : 大規模ネットワークでの効率的計算
4. 正確さ : 単精度vs倍精度
5. 収束の分析 : 検索結果順位安定のための反復回数
6. 今後の課題 : 展望と課題
7. 結論 : まとめ

The anatomy of a large-scale hypertextual web search engine

1. 導入 : サーチエンジンの歴史と重要性
2. システムの特徴 : PageRankの概念と計算法
3. 関連研究
4. システムの解剖 : 実装方法とweb巡回・検索結果向上について
5. 結果と成果 : PageRank導入による成果と検索に必要となる容量・時間について
6. 結論 : 今後の大規模拡張・テキスト利用の可能性

1. 序論

▼背景

・Webの急速成長により、webページ数は**8億ページ**(1999年当時)**以上**に

・webサーチエンジンの検索結果の**質の向上への期待**

▼アルゴリズムの概要

・GoogleのPageRankアルゴリズムは、親ページの重要度(**importance**)に基づいて、子のページの重要度を決めるアルゴリズム

・IBM HITSのシステムも、ハブとその重み(**authority score**)を決め、親のハブスコア(**hub score**)によって子の重みも決定される

▼アルゴリズムへの要請

・webの成長のために、**個別の検索**に関する技術はより重要に

・アルゴリズムの**実行時間**と必要な**メモリ**を減らすことも重要となる

2. PageRankアルゴリズム

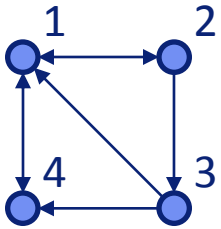
▼定義

N	全体のページ数
N_u	ページuからのリンク数
$Rank(p)$	ページpの重要度、初期値は$1/N$
$Rank$	全ページの重要度を表すベクトル
$link(u,v)$	リンクの重要度、$Rank(u)/N_u$とする
B_v	vへのリンクがあるページの集合
M	リンク構造を表す行列、各リンクに対応して$1/N_u$が入る

▼例

$$M=\begin{pmatrix} 0.000 & 1.000 & 0.000 & 0.500 \\ 0.333 & 0.000 & 1.000 & 0.000 \\ 0.333 & 0.000 & 0.000 & 0.500 \\ 0.333 & 0.000 & 0.000 & 0.000 \end{pmatrix}$$

3から4へのリンクに対応、N_3=2より、1/2が入る



$$Rank_1=\begin{pmatrix} 0.250 \\ 0.250 \\ 0.250 \\ 0.250 \end{pmatrix}$$

初期値としてそれぞれ1/4が入る

2. PageRankアルゴリズム

▼計算

$$Rank_{i+1}(v) = \sum_{u \in B_v} \frac{Rank_i(u)}{N_u}$$ を繰り返し演算

具体的には、 $Rank_{i+1} = M \times Rank_i$

を$Rank$がMの固有値$Rank^*$になるまで反復

$Residual_i = Rank_{i+1} - Rank_i$
と定義すると、$||Residual_i||$が$Rank^*$への近づき具合の指標となる

残差$||Residual_i||$が一定値以下になり収束すれば反復計算終了

▼例

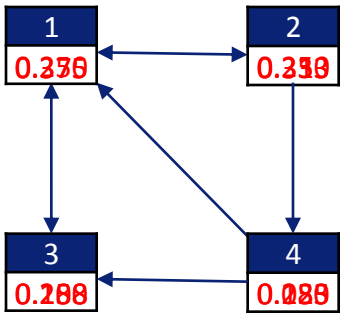
$$\begin{pmatrix} 0.375 \\ 0.333 \\ 0.208 \\ 0.083 \end{pmatrix} = \begin{pmatrix} 0.000 & 1.000 & 0.000 & 0.500 \\ 0.333 & 0.000 & 1.000 & 0.000 \\ 0.333 & 0.000 & 0.000 & 0.500 \\ 0.333 & 0.000 & 0.000 & 0.000 \end{pmatrix} \begin{pmatrix} 0.250 \\ 0.250 \\ 0.250 \\ 0.250 \end{pmatrix}$$

$Rank_2$ M $Rank_1$

$$Residual_1 = \begin{pmatrix} 0.125 \\ 0.083 \\ -0.042 \\ -0.167 \end{pmatrix} \quad ||Residual_1|| = 0.228$$

$||Residual_i||$の閾値を0.0001とすると、

$$Rank_{17} = Rank^* = \begin{pmatrix} 0.375 \\ 0.313 \\ 0.188 \\ 0.125 \end{pmatrix}$$



2. PageRankアルゴリズム

前ページの計算ではG(全体の集合)の全てのノードが外向きのedgeを持つという仮定が置かれていた

そこで、外向きedgeが0のノードにも概念的に外向きedgeを与える

$$M' = cM + (1-c) \times \left[\frac{1}{N}\right]_{N \times N}$$

リンク先の中からランダムに選択することに対応

web内の全ページの中からランダムに選択することに対応

$$M' \times Rank = cM \times Rank + (1-c) \times \left[\frac{1}{N}\right]_{N \times 1}$$

通常cとして0.85が用いられる

▼例

$$\begin{pmatrix} 0.000 & 1.000 & 0.500 & 0.000 & 0.000 \\ 0.333 & 0.000 & 0.000 & 1.000 & 1.000 \\ 0.333 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.333 & 0.000 & 0.500 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{pmatrix}$$

M

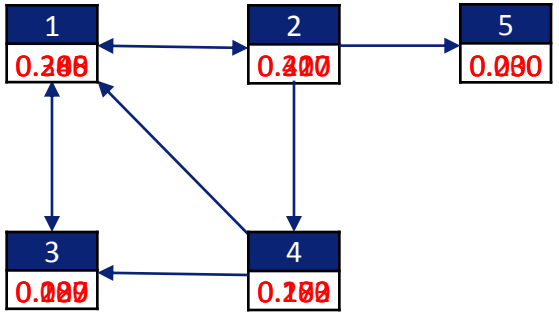
$1/N = 0.200$

$\times 0.85$ $\times 0.15$

$$\begin{pmatrix} 0.285 \\ 0.427 \\ 0.087 \\ 0.172 \\ 0.030 \end{pmatrix} = \begin{pmatrix} 0.030 & 0.880 & 0.455 & 0.030 & 0.030 \\ 0.313 & 0.030 & 0.030 & 0.880 & 0.880 \\ 0.313 & 0.030 & 0.030 & 0.030 & 0.030 \\ 0.313 & 0.030 & 0.455 & 0.030 & 0.030 \\ 0.030 & 0.030 & 0.030 & 0.030 & 0.030 \end{pmatrix} \begin{pmatrix} 0.200 \\ 0.200 \\ 0.200 \\ 0.200 \\ 0.200 \end{pmatrix}$$

$Rank_2$ M' $Rank_1$

$$Rank_{17} = Rank^* = \begin{pmatrix} 0.348 \\ 0.310 \\ 0.129 \\ 0.183 \\ 0.030 \end{pmatrix}$$



3. 効率的なメモリ使用

Stanford WebBaseには、約2500万ページがあり、リンクグラフの中に8100万以上のURLが含まれている。そのうち、外向きのグラフが無いものを除くと18,922,290のノードになる

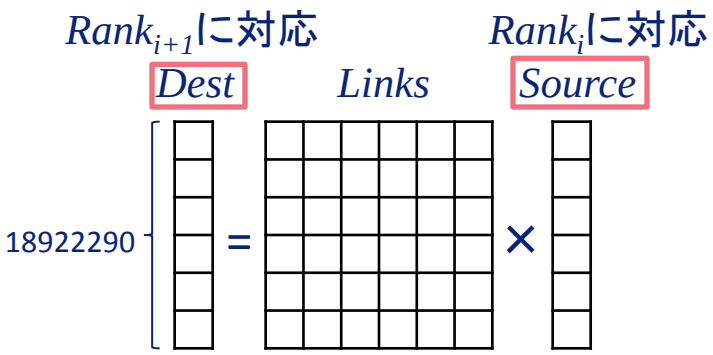
このグラフ構造を以下のように格納する

16bit：65536まで格納可能
32bit：4294967295まで格納可能

起点ノード	外向きリンク数	終点ノード
16bit	16bit	32bitのセット
0	4	12,26,58,94
1	3	15,56,81
2	5	9,10,38,45,78
:	:	:
18922290		

→1.01GB

以下の要領で計算（*Dest*と*Source*の合計サイズ150MB）



$\forall_s Source[s] = 1/N$
while$(residual > \tau)$ {
  $\forall_d Dest[d] = 0$
  while(not $Links$.eof()) {
    $Links$.read($source$, n, $dest_1, dest_2, ..., dest_n$)
    for $j = 1 \ldots n$
      $Dest[dest_j] = Dest[dest_j] + Source[source]/n$
  }
  $\forall_d Dest[d] = c \times Dest[d] + \frac{1-c}{N}$ /* dampening or personalization*/
  $residual = ||Source - Dest||$ /* recompute only every few iterations */
  $Source = Dest$
}

3. 効率的なメモリ使用

▼入出力コスト

メモリが*Dest*と*Source*を両方格納できるとき

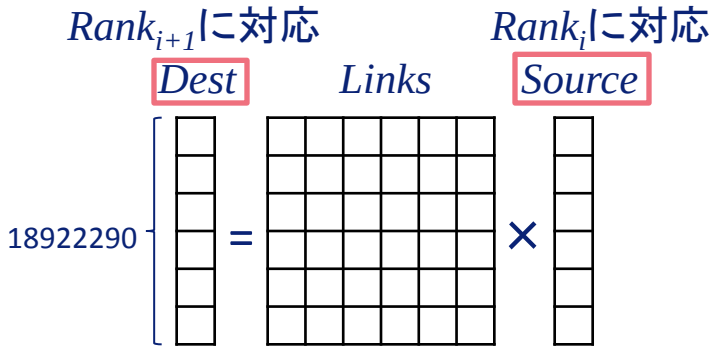
・・・$C=|Links|$

メモリが*Dest*のみ格納できるとき

・・・$C=|Dest|+|Links|+|Source|$

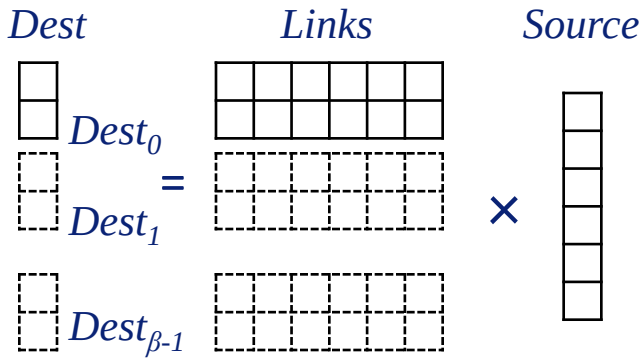
メモリがDestを格納できないとき

・・・莫大な時間がかかる→現実の8億ページに対応するのは不可能



3. 効率的なメモリ使用

▼ブロック分割法



Dページごとにβ個に分割
ただし、メモリがPページ分格納できる
とすると、D≦P-2とする。
（LinksとSourceを読み込むため）

Linkファイルの分割（例）

起点ノード	リンク数	分割内リンク数	終点ノード
16bit	16bit	16bit	32bitのセット
0	4	2	12,26
1	3	1	15
2	5	2	9,10
:	:		:
18922290			

(0≦Dest＜32)

0	4	1	58
1	3	1	56
2	5	2	38,45
:	:		:
18922290			

(32≦Dest＜64)

0	4	1	94
1	3	1	81
2	5	1	78
:	:		:
18922290			

(64≦Dest＜96) ・・・

入出力コスト

・・・$C = \beta \times |Source| + |Dest| + |Links| \times (1 + \varepsilon)$

分割による増加率

3. 効率的なメモリ使用

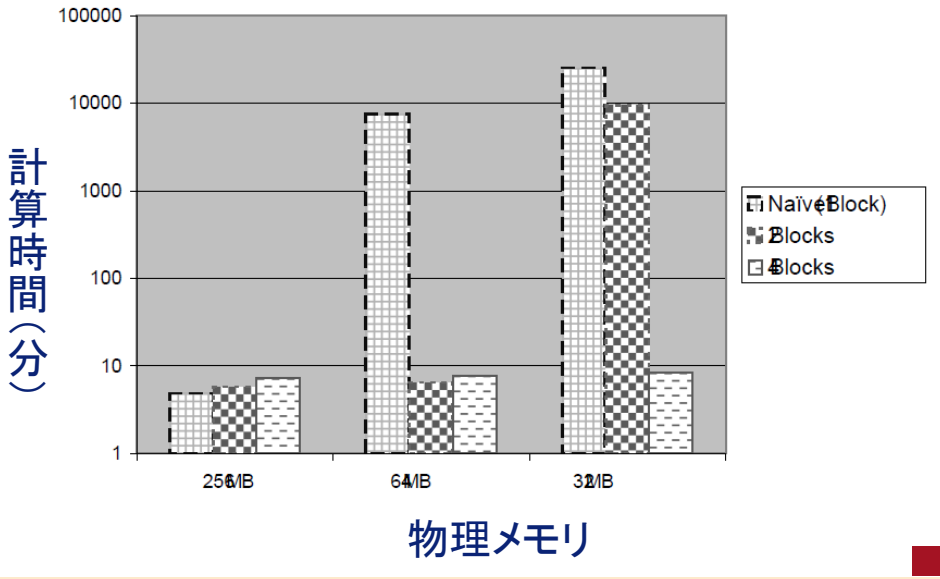
▼分割による変化

分割数	予想されるメモリ使用	*Links*の大きさ	ε
1	72MB	1.01GB	0.00
2	36MB	1.14GB	0.13
4	18MB	1.29GB	0.28

計算実験結果

450MHz Pentium-III、7200-RPM 18.0GB Western Digital Caviar AC418000使用
物理メモリは256MB・64MB・32MBの3種類を用意
（因みに当機は1.2GHz Celeron SU2300、250GB HD、物理メモリ2GB搭載）

→普通のコンピューターで
計算可能！

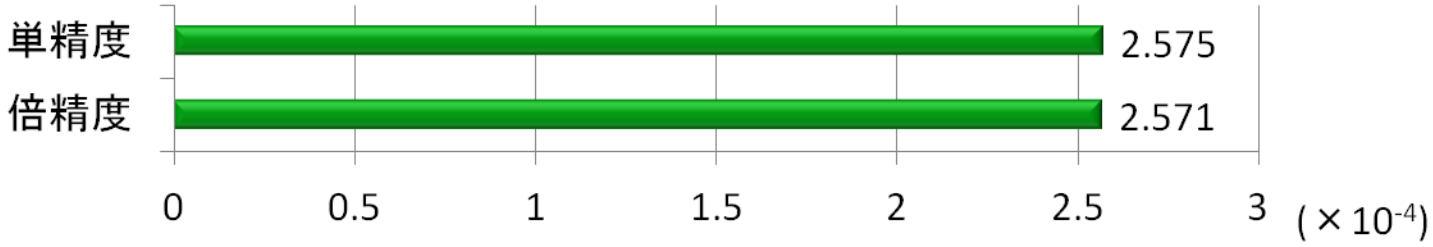


4. 正確さ

▼単精度 vs 倍精度

※
単精度浮動小数点型：32ビット（符号1ビット+指数部8ビット+仮数部23ビット）
倍精度浮動小数点型：64ビット（符号1ビット+指数部11ビット+仮数部52ビット）

Source・*Dest*に対して単精度浮動小数点型と倍精度浮動小数点型で精度がどれくらい変わるのか。100回計算を繰り返した時点での*Source*・*Dest*ベクトルの*residual*を比べる

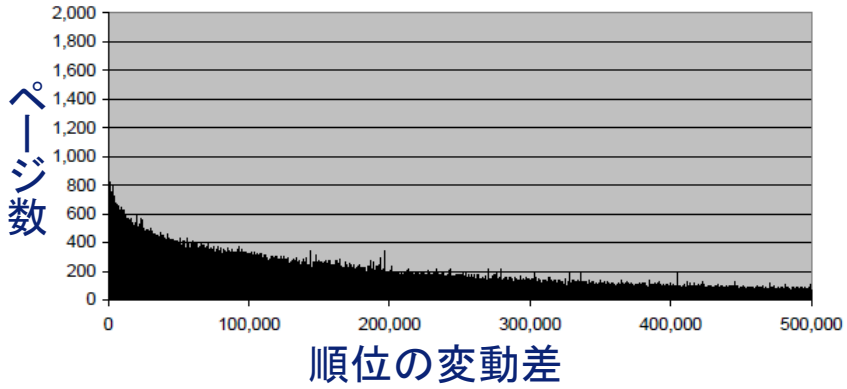


→顕著な差はなし

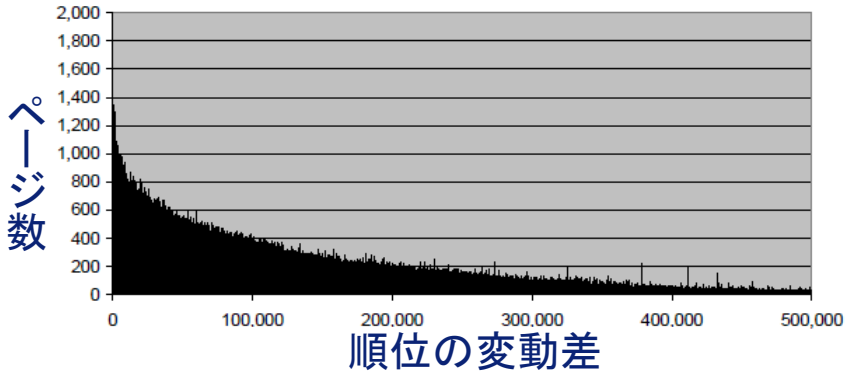
5. 収束の分析

▼計算の反復による順位変動ヒストグラム

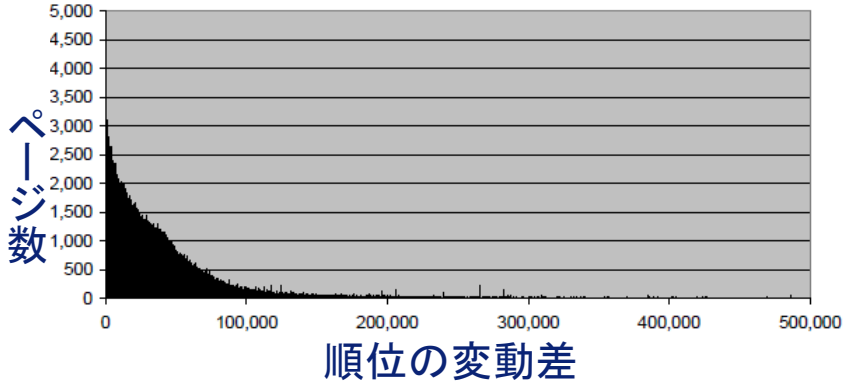
5回 vs 100回



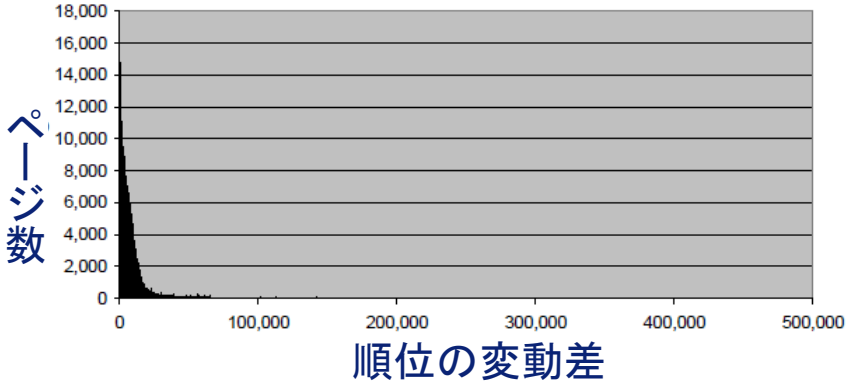
10回 vs 100回



25回 vs 100回



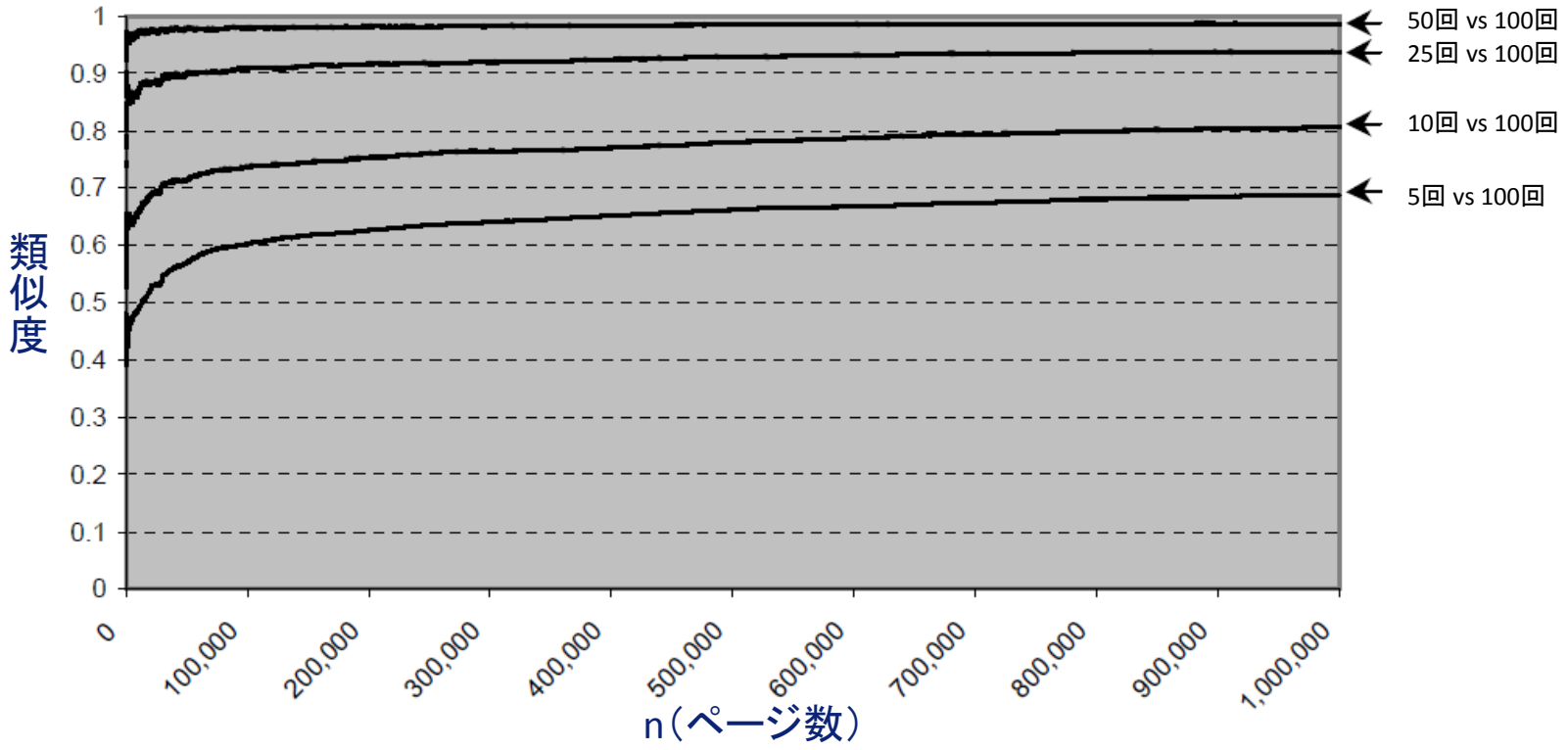
50回 vs 100回



→たった50回の計算で100回計算とほぼ同じ結果が得られている

5. 収束の分析

▼Pagerank上位nページ同士の比較

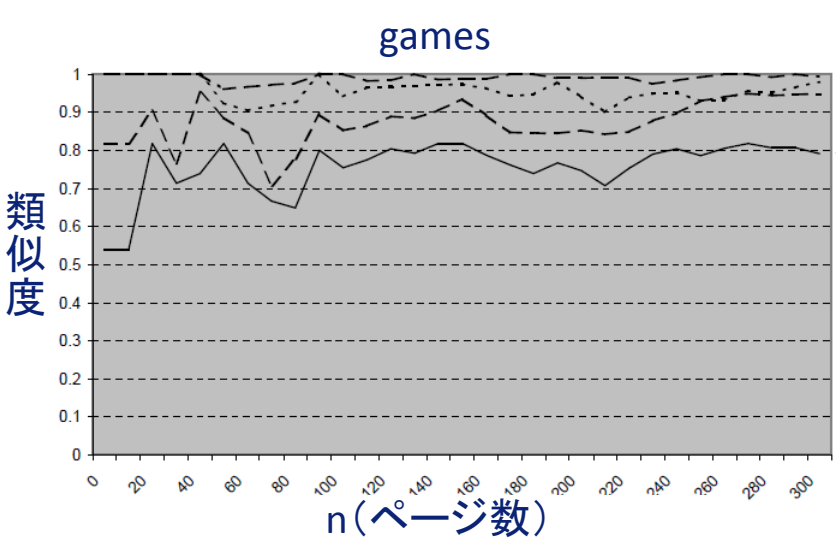


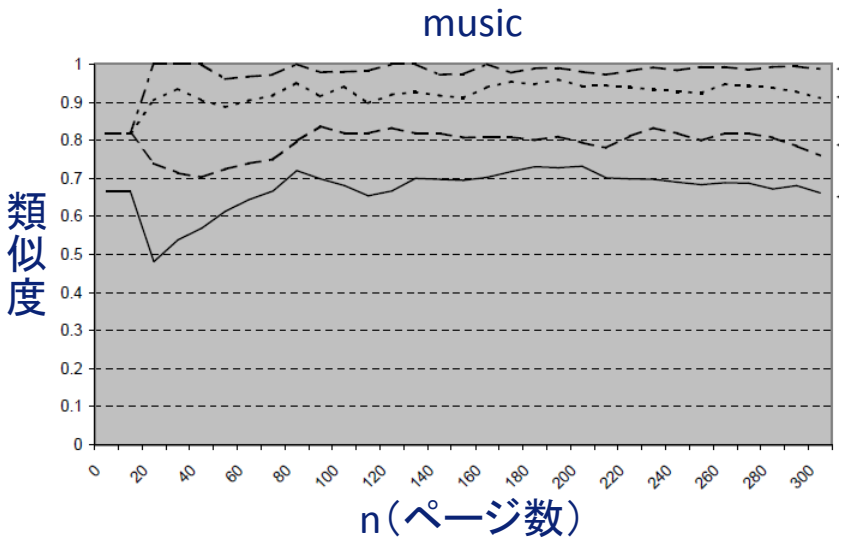
→上位ページに注目すると、25回計算すれば100回と非常に近い結果が得られているといえる

5. 収束の分析

▼個別キーワードのPageRankの上位nページ比較

gamesクエリ：action, adventure, game, role, playingを含む (1567のURLが該当)
musicクエリ：music, literatureを含む (19341のURLが該当)





（いずれも上より、上位nページ比較での、50vs100、25vs100、10vs100、5vs100の類似度を表す）

→わずか10回の計算で100回計算とほぼ同じ結果が得られているといえる

6. 今後の課題

・個人ユーザーが自分**個人にとってのPageRank**を計算できるようになる

・ユーザーの**閲覧履歴やブックマークを利用**して個別化のために使える

・安定したPageRank順位を決定するための**反復計算回数**を十分な実験で決めるべきだろう

7. 結論

・PageRankアルゴリズムは単純な概念だが、膨大なwebグラフの扱いには工夫が必要

・PageRankは**普通のコンピューター機械で**およそ1時間以内で計算可能であることを示した

・**単精度浮動小数点で十分**な精度で計算できる

・特定のクエリに結果を返すためには、**せいぜい10回の反復計算**でランク付け可能

・本稿で述べた方法論で**正確さを欠くことなくPageRank計算を早められる**