4C-2

Personalized PageRank の高速計算手法

藤原靖宏†

NTT ソフトウェアイノベーションセンタ†

1. はじめに

Personalized PageRank(PPR) [1]はグラフのノードの関連度として最も注目を集めているもののひとつで，様々なアプリケーションに応用されている[2]．しかし PPR には計算量が高いという問題がある．そのため本論文では PPR による関連度を高速に計算する問題を取り組む．

2. Personalized PageRank

PPR では問い合わせ分布から各ノードの存在確率を決定し，そのノードからランダムウォークを行う．そしてノードに到達するたびに一定の確率cで問い合わせ分布に基づき各ノードに戻る．グラフのノード数をnとする．sをn×1行列とし，s[u]要素をノードuにランダムウォークする確率を表すとする．またdを問い合わせ分布で決定されるn×1行列とする．またWを列が正規化されたグラフの隣接行列とする（すなわち W[u,v]要素はノードvからノードuへランダムウォークする確率を表す）．定常状態における各ノードにおける存在確率は以下の式を再帰的に収束するまで繰り返すことで計算できる．

$$s = (1-c)Ws + cd$$

PPR は定常状態における確率を関連度とする方法である．すなわちs行列のs[u]要素はノードuの分布dに対する関連度となる．

Algorithm 1 に関連度を計算するアルゴリズムを示す．このアルゴリズムにおいてVとC_uをそれぞれグラフにおける全てのノードの集合及びノードuから出ているエッジがつながっている隣接ノードの集合とする．また s′を繰り返しにおいて計算される更新後の関連度とし，pをノード間で計算される関連度の伝搬値とする．

Algorithm 1 ははじめに関連度の分布 sを問い合わせ分布dに初期化する（1 行目）．繰り返し計算のなかでまず更新後の関連度を 0 に初期化する（3～5 行目）．そしてすべてのノードに対してそれらのノードから出ているエッジが他のノードに伝搬する関連値を計算する（6～11 行目）．そして伝搬された関連度から更新後の関連度を求める（12～13 行目）．これらの処理は所定の繰り返し回数まで再帰的に実行される．

この繰り返し計算による処理はグラフのサイズが大きくなると関連度の計算を高速に行えない．これは このアルゴリズムがグラフ全体を用いて関連度の計算を行っているからである．

Algorithm 1 オリジナルの類似度計算アルゴリズム

```
1:  s = d;
2:  for t = 1 から T に対して do
3:    for それぞれのノード u ∈ V に対して do
4:      s′[u] = 0;
5:    end for
6:    for それぞれのノード u ∈ V に対して do
7:      for それぞれのノード v ∈ C_u に対して do
8:        p = W[v,u]s[u];
9:        s′[v] = s′[v] + p;
10:     end for
11:   end for
12:   s′ = (1 − c)s′ + cd;
13:   s = s′;
14: end for
15: return s;
```

3. 提案手法

本論文では関連度の計算を高速に行うために，繰り返し計算の中で関連度の小さいノードを枝刈りする方法と，関連度の小さいエッジを枝刈りする方法の２つの手法を述べる．

3.1 ノードを枝刈りする手法

この手法では繰り返し計算のなかで関連度がθ以上の関連度の高いノードからのみ関連度の伝搬値を計算し，結果高速に関連度の計算を行う．

Algorithm 2 にこの手法のアルゴリズムを示す．ここでV′を枝刈り後のノードの集合とし，θを枝刈りにおける閾値とする．まず関連度の分布sとノードの集合V′をそれぞれ問い合わせ分布とノード全体の集合に初期化する（1～2 行目）．繰り返し計算の中で各ノードからの伝搬値は全てのノードからでなく，枝刈り後のノードの集合から計算する（7～12 行目）．枝刈り後のノードの集合は各繰り返し計算の中で更新された関連度から求める（15～20 行目）．

3.2 エッジを枝刈りする手法

この手法は関連度の伝搬値の小さなエッジを枝刈りする．すなわち各ノードから伝搬値を計算する過程でその後に計算する伝搬値が小さくなることがわかった段階で計算を枝刈りし，高速な関連度の計算を行う．

Algorithm 3 にこの手法におけるアルゴリズムを示す．まず関連度の分布sを問い合わせ分布に初期化する（１行目）．そして繰り返し計算のなかで効率的な計算の枝刈りを行うために，各ノードに対してそれらのノードから出ているノードをエッジの重みが大きい順で並び替える（2～

Efficient computation for Personalized PageRank
†Yasuhiro Fujiwara, NTT Software Innovation Center

Algorithm 2 ノードの枝刈りによる類似度計算アルゴリズム

```
1: s = d;
2: V′ = V;
3: for t = 1 から T に対して do
4:   for それぞれのノード u ∈ V に対して do
5:     s′[u] = 0;
6:   end for
7:   for それぞれのノード u ∈ V′ に対して do
8:     for それぞれのノード v ∈ C_u に対して do
9:       p = W[v, u]s[u];
10:      s′[v] = s′[v] + p;
11:    end for
12:  end for
13:  s′ = (1 − c)s′ + cd;
14:  s = s′;
15:  V′ = ∅;
16:  for それぞれのノード u ∈ V に対して do
17:    if s′[u] ≥ θ then
18:      ノード u を V′ に加える;
19:    end if
20:  end for
21: end for
22: return s;
```

Algorithm 3 エッジの枝刈りによる類似度計算アルゴリズム

```
1: s = d;
2: for それぞれのノード u ∈ V に対して do
3:   ノードの集合 C_u をエッジの重みの降順で並び替え;
4: end for
5: for t = 1 から T に対して do
6:   for それぞれのノード u ∈ V に対して do
7:     s′[u] = 0;
8:   end for
9:   for それぞれのノード u ∈ V に対して do
10:    for それぞれのノード v ∈ C_u に対して do
11:      p = W[v, u]s[u];
12:      s′[v] = s′[v] + p;
13:      if p < θ then
14:        break;
15:      end if
16:    end for
17:  end for
18:  s′ = (1 − c)s′ + cd;
19:  s = s′;
20: end for
21: return s;
```

4 行目）．繰り返し計算のなかでは各ノードから他のノードへの伝搬値を計算するが（11～12 行目），もし計算された伝搬値がθより小さければそのノードからの伝搬値の計算を枝刈りする（13～15 行目）．これはエッジが重みの大きい他のノードへの伝搬値を計算するが（11～12 行目），もし計算された伝搬値がθより小さければそのノードからの伝搬値の計算を枝刈りする（13～15 行目）．これはエッジが重みの大きい順に並び替えられているため，その後に計算される伝搬量がθより大きくならないからである．
この手法は伝搬量が小さいエッジを効率的に枝刈りできるため，高速に関連度を計算できる．

4. 実験結果

提案手法の有効性を確認するために実験を行った．結果を表 1 に示す．実験では各手法の計算時間と計算により得られた関連度の平均誤差を調べた．実験データとしてはヨーロッパにおける研究機関から得られたメールのデータを用いた．このデータにおいてノード数は265,214でエッジ数420,045である．実験においては$c = 0.15$, $T = 100$とし，ランダムウォークを開始するノードはランダムに 1 つ設定した．またノードの枝刈りによる手法とエッジの枝刈りによる手法においては枝刈りの閾値θを10^{-3}及び10^{-7}にして実験を行った．
実験結果から提案手法はオリジナルの手法と比較して高速に関連度を計算できることがわかる．また同時に枝刈りの閾値が大きいほうがより高速な関連度の計算が可能であることがわかる．これは閾値が大きい方がより多くのノードまたはエッジを枝刈りできるからである．

表 1 実験結果

	ノード枝刈り		エッジ枝刈り		オリジナル
	θ =10^{-3}	θ =10^{-7}	θ =10^{-3}	θ =10^{-7}	
計算時間[s]	0.05	0.46	0.93	0.96	1.01
平均誤差	$3.15\cdot10^{-8}$	$4.81\cdot10^{-11}$	$3.54\cdot10^{-8}$	$3.17\cdot10^{-10}$	–

また結果から提案手法における関連度の誤差は非常に小さい値になることがわかる．提案手法はノードまたはエッジを枝刈りするが，関連度の値に影響がないようにしているからである．
また結果から提案手法は閾値が小さい方がより高い精度で関連度を計算できることがわかる．これは閾値が小さい方がより多くのノードまたはエッジが計算対象となるため，関連度の計算精度が向上するためである．
これらの結果から本発明において高速な計算を行いたい場合は閾値を大きくし，高精度な計算を行いたい場合は閾値を小さくすればいいことがわかる．

5. まとめ

本論文では PPR に対する高速な関連度の計算方法を提案した．実験から提案手法は関連度の精度はそれほど犠牲にせずに計算時間を削減できることを示した．

6. 参考文献

[1] Jeh et al., “Scaling Personalized Web Search”, WWW 2003
[2] Fujiwara et al., “Efficient Personalized PageRank with Accuracy Assurance”, KDD 2012