Vol.2015-DBS-162 No.18
2015/11/26

グラフ再構築による大規模ウェブグラフ解析の性能評価

武井　良太[1,a)]　新美　礼彦[1,b)]

概要：近年, ウェブグラフは大規模になっており, 解析にかかる時間が増大している. そのため, 大規模ウェブグラフに対して解析時間を短縮する手法が必要とされている. 我々は, ウェブグラフを並列分散処理可能なグラフへ再構築を行い, ウェブグラフ解析を行う際に並列分散処理を用いることで, 解析に必要な時間を短縮する手法を提案した. ウェブグラフをグラフクラスタリングにより, クラスタをノードと見立てたクラスタ間の関係を表現したグラフとクラスタに属するノードで構成されるグラフにウェブグラフを再構築する. 再構築によって作成された各グラフは互いに関係性が無いため, 並列分散処理による解析が可能になる. これまでに, 小規模ウェブグラフの解析時間短縮に成功しているが, 大規模ウェブグラフで解析時間短縮が可能であるか検証していなかった. 本稿では, 提案手法を大規模ウェブグラフで実験を行い, 性能に関する考察を行う. 実験より解析時間を 6 割以上短縮することに成功したが, 性能に関しては改善の必要があることが判明した.

キーワード：グラフ再構築，並列分散処理，大規模データ，グラフマイニング

1. はじめに

ウェブグラフとはウェブをグラフ構造化したものであり, ノードをウェブページ, エッジをリンクで表現したものである. ウェブグラフを解析することで知見を導出することが盛んに行われている. 近年では, ノード数とエッジ数が数億個になるグラフが存在している. 例えば Internet Live Stats は 2014 年 9 月の時点で, 全世界で少なくとも 10 億ページ以上存在すると報告している [1]. ウェブグラフは大規模になっており, 解析にかかる時間は膨大になる. そのため, 大規模ウェブグラフに対して解析時間を短縮する手法が必要とされている. 解析時間を短縮する手法 [2],[3],[4],[5],[6] は複数あるが, その内の 1 つとしてウェブグラフを再構築し解析時間を短縮する手法が挙げられる. 我々は, 大規模ウェブグラフの解析時間の短縮を目標とし, ウェブグラフの解析を並列分散が可能な形状にウェブグラフを再構築, 解析に必要な時間を短縮する手法を提案した [7]. 提案手法は, ウェブグラフをグラフの密構造部分を切り出したクラスタに属するノードで構成されるグラフと, クラスタをノードと見立てたクラスタ間の関係性を表現したグラフに再構築することで, 並列分散処理が可能となる. 再構築によって作成された各グラフは互いに関係性が無いため, 並列分散処理による解析が可能になる. これまでに, 小規模ウェブグラフの解析時間短縮に成功しているが, 大規模ウェブグラフで解析時間短縮が可能であるか検証していなかった. 本稿では, 提案手法を大規模ウェブグラフで実験を行い, 性能に関する考察を行う.

[1] 公立はこだて未来大学
Future University Hakodate
[a)] g2114020@fun.ac.jp
[b)] niimi@fun.ac.jp

2. 提案手法

我々は, 並列分散処理によるウェブグラフ解析が可能なウェブグラフ再構築手法を提案している [7]. 提案手法によるウェブグラフ再構築は,2 つのステップで行われる.

Step 1
グラフクラスタリングを用いてウェブグラフをクラスタリングする.

Step 2
並列分散処理によるウェブグラフ解析を可能にするため, クラスタをノードに見立てたグラフである Compression graph とクラスタに属するノードを用いたグラフである Cluster graph の 2 種類のグラフを作成し, ウェブグラフを再構築する.

再構築したウェブグラフの並列分散処理による解析方法は次章で述べる.

2.1 グラフクラスタリング

グラフクラスタリングを用いてウェブグラフをクラスタ

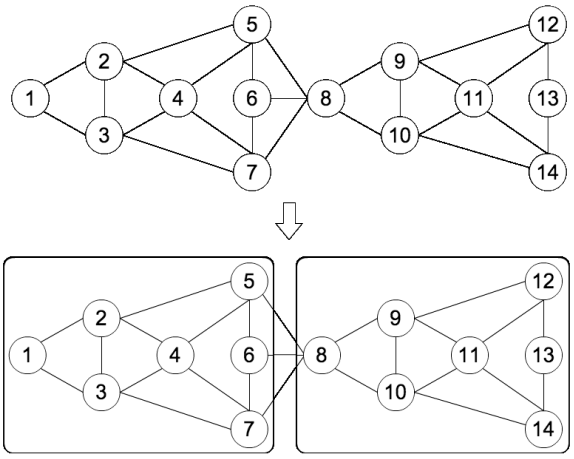

図 1 グラフクラスタリングの流れ

リングする. グラフクラスタリング処理の例である図 1 は上段のグラフに対してグラフクラスタリングを実行し, 下段は 2 つのクラスタを導出している.

2.2 再構築

グラフクラスタリングの結果を用いてウェブグラフを再構築する. クラスタをノードに見立てたグラフである Compression graph と,Cluster に属するノードで構成されたグラフである Cluster graph の 2 種類を作成する. 再構築する際, クラスタ間のエッジ数を計数し, エッジに総数の重みをつける. 再構築の例である図 2 は 2.1 でグラフクラスタリングを用いてノードをクラスタリングした上段のグラフより, クラスタをノードに見立てたグラフである Compression graph と, クラスタに属するノードで構成される Cluster graph に再構築したものを下段に示す.

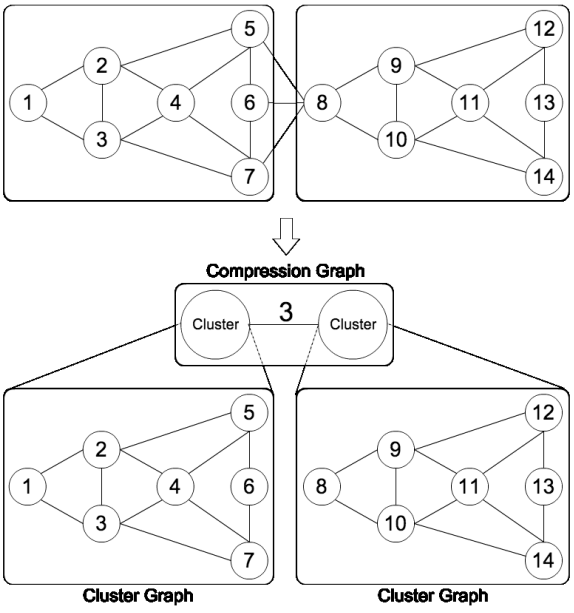

図 2 再構築の流れ

3. 並列分散処理による解析

再構築したウェブグラフの解析ステップは 3 つのステップに分かれている.1 つ目のステップは Compression graph と Cluster graph の解析,2 つ目のステップは各解析結果の乗算によるマージ,3 つ目は乗算結果のソートである.1 つ目のステップである Compression graph の解析と Cluster graph の解析は互いに関係性が無いため, 並列分散処理による解析を適応することが出来る. 今回は例として解析には PageRank を用いて説明する. 最初にクラスタをノードに見立てたグラフである Compression graph の PageRank 値とクラスタに属するノードのグラフである Cluster graph の PageRank 値を算出する.Compression graph の PageRank 値を算出する際, 異なる 2 つのクラスタ間のエッジのみを考慮し, クラスタに属するノード間のエッジは無視する. また Cluster graph の PageRank 値を算出する際, クラスタ外に接続されているエッジは無視する. 次に, それぞれの解析結果の乗算マージ処理であるが, それぞれ対応する Compression graph に属するクラスタの PageRank 値と,Cluster graph のクラスタに属するノードの PageRank 値を乗算する. 乗算する理由として, 高速なマージが可能なためである. 最後に, 乗算結果をソートすることで最終的なランキングを算出する. 解析のステップを図 3 に示す.

4. 実験

提案手法の評価実験として, グラフクラスタリングを用いて大規模ウェブグラフを再構築し, グラフ解析にかかる時間や再構築によって解析時にあらわれる影響について検証する. 実験で使用したグラフクラスタリングは構造的類似度に基づいたグラフクラスタリングである SCAN[8], グラフ解析手法は PageRank を用いた.SCAN の閾値は ε=0.7, μ=2 を用いた.SCAN を用いた理由として, ノードを Cluster と Hub,Outlier の 3 種類に分別することができ, 外れ値である Outlier を解析対象から外すことで, 高速な解析が可能になるためである.SCAN の詳細は 6.2 章で述べる. 実験環境は MacBook Air を用いた. 詳細なスペックは表 2 に示したとおりである. 実装環境は再構築を Hadoop,PageRank を Spark で実装した. 詳細なバージョンは表 4 に示したとおりである. 実験データは Laboratory for Web Algorithmics[9] で公開している表 3 に示すデータを利用した.

表 1 実験環境

CPU	Intel Core i5 1.4GHz
Memory	8.0GB
Disk	SSD 128GB

4.1 解析時間

大規模ウェブグラフを提案手法によって再構築したグラ

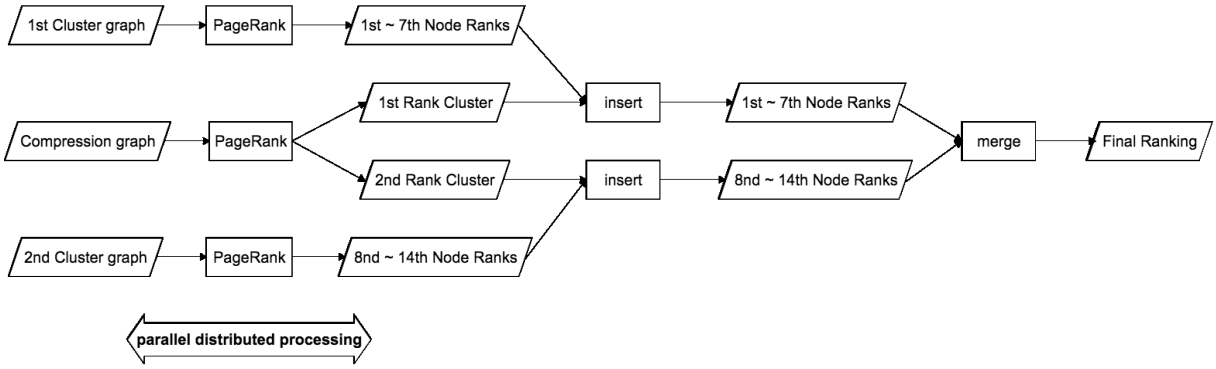

図 3 並列分散処理による解析の流れ

表 2 実装環境

Hadoop	1.0.3
Spark	1.4.0

表 3 データセット

Data Set	Nodes	Edges
cnr-2000	325,557	3,216,152

フの PageRank 値算出時間と, 未再構築グラフの PageRank 値算出時間を比較する. 比較結果は表 4 に示す. 比較より解析時間を 6 割以上短縮することに成功し, 大規模ウェブグラフを提案手法によって再構築したグラフでも解析時間短縮が可能であることを示した.

表 4 処理時間比較

	処理時間
PageRank	476 秒
提案手法	173 秒

4.2 ランキング

提案手法で算出されたランキングの上位 1000 件を検証する. まず, ランキングの適合率と再現率を算出し, 図 4 に示すような適合率再現率曲線をプロットした. 図より性能改善の必要があると判明した.

次に, ランキングの差を算出し, 図 5 に示すような *Gap* をプロットした. ランキングの差 *Gap* は以下で示される.

$$Gap = OriginalRank - ProposalRank$$

OriginalRank は未再構築のウェブグラフに対して PageRank を実行し得られたランキングであり,*ProposalRank* は提案手法によって再構築されたウェブグラフに対して PageRank を実行し得られたランキングである. 図 5 より, 上位 100 件のノードは *ProposalRank* が本来のランキングである *OriginalRank* より低く算出され,100 件以降は逆転している傾向が判明した.

傾向の分析と改善方法は考察で述べる.

4.3 前処理時間

提案手法のクラスタリングと再構築に要した時間を述べる. 要した時間は表 5 に示す. 実験で使用した SCAN は並列分散処理がおこなえないため, 処理に時間を要している. よって, 並列分差処理が可能なグラフクラスタリング手法の検討が必要であることが判明した. また, 再構築に関しても同じく処理に時間がかかっているため, 効率のよい実装方法の検討が必要である.

表 5 前処理時間

	要した時間
クラスタリング	231 分 32 秒
再構築	40 分 12 秒

5. 考察

5.1 実験結果からの考察

1 つ目の考察は Compression graph のランキング結果と Cluster graph のランキング結果のマージ方法を考察する. 図 4 より性能改善の必要があると判明した. そこで, 新たなマージ方法を検討することで性能向上を図る. 現在はそれぞれ対応する Compression graph に属するクラスタの PageRank 値と,Cluster graph のクラスタに属するノードの PageRank 値を乗算することで最終的なランキングを算出している. 実験で使用したグラフクラスタリングである SCAN は, クラスタだけでなく Hub も算出する.Hub は Cluster graph に含まれないため, 乗算する際 PageRank 値を 1.0 とみなし計算している. そのため,Hub のランキングは相対的に低く算出されていると考えられる. そこで, 乗算によるマージではなく,Compression graph に属するクラスタの対応する部分に Cluster graph に属するノードのランキングを挿入するマージ方法を検討する. これにより,Hub のランキングが低く算出されることを防止することが可能になり, より良いランキングを算出することが可能になると考えられる.

2 つ目の考察は実験結果の分析に関する考察である. 図 5 より, 上位 100 件のノードは *ProposalRank* が本来のラ

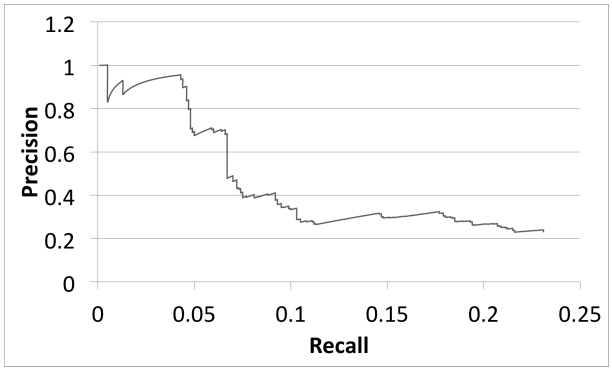

図 4 適合率再現率曲線

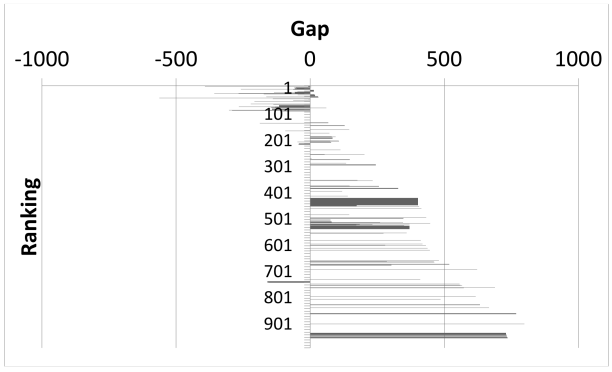

図 5 ランキングの差

ンキングである *OriginalRank* より低く算出され,100 件以降は逆転している傾向が判明した. 何故このような傾向が出ているのか原因が不明であるので, より詳しい解析を行う必要があると考える.

5.2 実験設定からの考察

1 つ目はグラフクラスタリングの選択に関して考察する. 今回の実験ではグラフクラスタリングに構造的類似度に基づいたクラスタリング手法である SCAN を用いた. これは外れ値である Outlier を作成することで, 解析対象となるノード数とエッジ数を削減することが出来るためであった. しかし, クラスタリング時に複数の小規模クラスタが作成された. 並列分散処理の効果を最大限までに高めるためには, 小規模クラスタは大規模クラスタに併合されるか外れ値として算出されることが望ましい. よって,SCAN の閾値を変更し, 小規模クラスタが作成されにくくすることで提案手法の効力を最大限高めることが出来ると考えられる. 更に, 各ウェブグラフの特性に合わせたグラフクラスタリングを適応することが可能になれば, より高速かつより良いランキングを算出することが可能になると考えられる. また, 現状グラフクラスタリングに時間がかかっているため, 高速なグラフクラスタリングを選択することで前処理の時間を短縮することが出来る.

2 つ目は解析方法の選択に関して考察する. 今回の実験ではウェブグラフ解析手法にウェブページの重要度を算出するアルゴリズムである PageRank を用いた. これは PageRank がベンチマークとして多くの研究者が使用しており, 実験結果の比較などがし易いと考えたため使用した. 今後は他の解析手法でも適用出来るかどうか確認し, 多くの解析手法の高速化実現に向けて実験を進める.

6. 関連研究

関連研究として, ウェブグラフ再構築手法の先行研究と実験で用いた SCAN の紹介を行う.

6.1 先行研究

大規模なウェブグラフを再構築することは, ウェブグラフを用いる解析手法において重要な技術である. ウェブグラフの再構築は大きく分けると, 符号化による手法と構造改良による手法の 2 種類に分かれる.

6.1.1 符号化

符号化による手法は既存のデータ圧縮を使用するか, ウェブグラフの特徴を用いる手法である.P. Boldi らはウェブグラフの特徴と WebGraph[9] という名の手法を提案した. ウェブグラフの特徴として, 局所性と類似性, 連続性を挙げている.

・局所性

多くのウェブページは同じホスト内のページに誘導するためのリンクを含んでいる. 例えば,“home”,“next”,“previous”,“up” などのリンクが挙げられる. 例で上げたリンクは同じホスト内でリンクされることが多い. つまり, リンク元とリンク先のウェブページの URL を比較すると先頭文字列 (ホスト名) が一致しやすく, 局所性が現れる.

・類似性

同じホストのウェブページは似たようなウェブページにリンクを持つ傾向がある. 例えば, 共通のテンプレートで作成されるページなどが挙げられる. 例で上げたページは同じ構造のページがたくさんあるため同じウェブページにリンクを貼ることが多い. つまり, 同一ホスト名のウェブページのリンクの URL を比較すると URL が一致しやすく, 類似性が現れる.

・連続性

ウェブページに含まれるリンクの URL は一定の固まりで現れる傾向がある. 例えば, 大量の写真を掲載しているページが挙げられる. 例で上げたページはある一定の連続したリンクの固まりが存在する. つまり,URL を辞書順に整列し順に ID を割り当てると数字が連続し, 連続性が現れる.

P. Boldi らは上記の特徴を用いて, 連続性を利用した差分符号化と, 局所性と類似性を利用したランレングス圧縮

を用いた手法を提案している. これにより,1 リンクあたり 3.08bit に抑えることに成功している.S. Tei らは WebGraph の差分符号化の後算術符号を用いることで, ウェブグラフの情報量を WebGraph よりも 2 割ほど多く削減することに成功している [10]. しかし, 符号化による手法では情報量を削減することは出来るが, ノード数とエッジ数を削減出来ない. また, 解析する際復号化する必要がありかえって PageRank の解析時間が増加してしまう. よって本研究では, 符号化による手法は用いない.

6.1.2 構造改良

構造改良はウェブグラフの構造を一部組み替えることでノード数やエッジ数を削減する手法である.G. Buehrer らは Virtual Node Miner[2] と呼ばれる, 架空のノード (以下,VN) をウェブグラフに挿入することでエッジ数を削減する手法を提案した.VN は図 6 のような完全 2 部グラフの中間に図 7 のように挿入する. これによって, リンクを集約しノードを削減している.Virtual Node Miner は完全 2 部グラフの探査に最小値独立置換族を利用している. これにより, ノード数約 7800 万, エッジ数約 3 億のウェブグラフに対して, 約 1700 万の VN を挿入しエッジ数を 1/7 に削減出来たとしている.

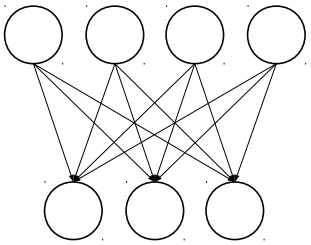

図 6 圧縮前

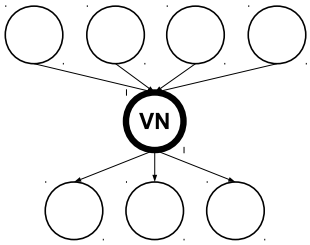

図 7 圧縮後

片瀬らは LittleWeb[3] と呼ばれる, 類似ノード集約によるウェブグラフ圧縮手法を提案した.LittleWeb は集約対象となるノードを探査するためのクラスタリング処理と, クラスタリング結果を用いた構造改良の 2 ステップでウェブグラフを再構築している. これにより, ノードを 67.3 %, エッジを 56.3 %削減している. 更に,PageRank の解析時間を 67 %短縮することに成功している. 構造改良による手法は, ノード数とエッジ数を削減することが出来かつ,PageRank の解析時間を短縮することが可能である. よって本研究では,PageRank の解析時間の短縮を目標としているため構造改良による手法を用いる.

6.1.3 提案手法と関連研究の違い

提案手法と関連研究の違いについて述べる. 違いとして, 関連研究で取り上げた従来手法はエッジ数やノード数を削減し,PageRank の解析時間を短縮することに成功している. これに対し提案手法では, エッジ数やノード数の削減だけでなく, ウェブグラフを解析が並列分散処理可能なグラフへ再構築を行い, ウェブグラフの解析を行う際に並列分散処理を用いることで, 解析に必要な時間を短縮する手法を提案した.

6.2 SCAN

実験で用いるクラスタリング手法の SCAN[8] について説明する.SCAN はクラスタリング対象とするグラフ $G = (V, E)$ を解析し, 構造類似度の下限値を示す閾値 ε と,Cluster の最小ノード数を示す閾値 μ に応じた structure-connected cluster を作成する.Cluster に属さなかったノードは Hub か Outlier の分別が行われる.

最初にすべてのノードに対して unclassfied のラベルを与える.SCAN は unclassfied のラベルが付いているノードに対して, そのノードが core であるかどうかの判定を行う.core ノードは Cluster の中心となるノードである.core であるか判断するためには, 対象となる unclassfied なノードの構造的隣接ノード集合を定め, 構造的類似度を計算, 閾値 ε を用いて ε-neighborhood を算出することで core かどうか判断することが出来る. 以下に構造的隣接ノード集合と構造的類似度,ε-neighborhood の定義を記述する.

・構造的隣接ノード集合

$v \in V$ であるとき, 構造的隣接ノード集合はノード v にエッジで接続するノードと, ノード v 自らで構成される集合 $\Gamma(v)$ で表される.

$$\Gamma(v) = \{v \in V \mid \{v, w\} \in E\} \cup \{v\}$$

・構造的類似度

$|\Gamma(v)|$ が隣接ノード集合に含まれるノード数とすると, ノード v, w 間の構造的類似度は $\sigma(v, w)$ で表される.

$$\sigma(v, w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$

・ε-neighborhood

構造類似度の下限値を示す閾値 ε を用いることで,ε-neighborhood と表される.

$$N_\varepsilon(v) = \{w \in \Gamma(v) | \sigma(v, w) \geq \varepsilon\}$$

・Core

$\varepsilon \in \mathbb{R}, \mu \in \mathbb{N}, v \in V, |N_\varepsilon(v)|$ をノード v の ε-neighborhood のノード数とすると,core は $CORE_{\varepsilon,\mu}(v)$ で表される.

$$CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geqq \mu$$

SCAN の判定によりノードが core であった時, この core ノードにユニーク ID である clusterID のラベルを与え,core ノードを中心に structure-connected cluster を作成する. ノードが core でなかった時, そのノードに non-menber のラベルを与える.

SCAN は core ノードと判定されたノードから,structure

reachability で到達可能なノードが所属する Cluster に clusterID のラベルを与える. 最初に, キュー Q へ core ノードの ε-neighborhood に含まれるすべてのノードを挿入する. 次に, キュー Q へ挿入されたノードに対して,direct structure reachability となるノードの集合である R を求める. 以下に direct structure reachability と structure reachability の定義を記述する.

・Direct structure reachability

ノード v とノード w における direct structure reachability は $DirREACH_{\varepsilon,\mu}(v,w)$ で表される.

$$DirREACH_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge N_{\varepsilon}(v)$$

・Structure reachability

$\varepsilon \in \mathbb{R}$,$\mu \in \mathbb{N}$,$v \in V$ とすると, ノード v とノード w における Structure reachability は $REACH_{\varepsilon,\mu}(v,w)$ で表される.

$$\begin{aligned} &REACH_{\varepsilon,\mu}(v,w) \Leftrightarrow \\ &\quad \exists_1, ...v_n \in V \ : \ v_1 = v \wedge v_n = w \wedge \\ &\forall i \in \{1, ..., n-1\} \ : \ DirREACH_{\varepsilon,\mu}(v_i, v_{i+1}) \end{aligned}$$

ノード集合 R を求めるには, キュー Q に含まれる全てのノードの構造的隣接ノード集合に対して, 構造的類似度を計算する. 最後にノード集合 R に含まれるノードの所属する Cluster が unclassfied の時, ノードをキュー Q へ挿入する. ノード集合 R に含まれるノードが unclassfied, または non-menber の時, 作成した clusterID のラベルを与える. 一連の処理をキュー Q がなくなるまで行い,structure-connected cluster を作成し続ける.

structure-connected cluster の作成が終了したら,non-menber のラベルが付いたノードに対して Hub と Outlier の分別を行う.non-menber のラベルが付いたノードが 2 つ以上の異なる Cluster と接続している場合, そのノードに対して Hub のラベルを与える.1 つのみの Cluster と接続している場合, そのノードに対して Outlier のラベルを与える.

以上より, 全てのノードに対してラベルが付け終わり SCAN によるクラスタリングは完了する.

7. おわりに

我々は, 大規模ウェブグラフの解析時間の短縮を目標とし, ウェブグラフの解析を並列分散が可能な形状にグラフを再構築, 解析に必要な時間を短縮する手法を提案した. 提案手法は, ウェブグラフをグラフの密構造部分を切り出したクラスタに属するノードで構成されるグラフと, クラスタをノードと見立てたクラスタ間の関係性を表現したグラフに再構築する. これにより, 再構築によって作成された各グラフは互いに関係性が無いため, 並列分散処理による解析が可能になる. 本稿では, 提案手法を大規模ウェブグラフで実験を行い, 性能に関する考察を行った. 実験の結果から, 解析時間は大規模ウェブグラフの PageRank 値算出にかかる時間を 6 割以上削減することに成功した. また, ランキングの性能は上位 1000 件の分析より, 性能改善の必要があることが判明した. 改善策として新たなランキングのマージ方法を考案した. 今後は考案した改善方法を組み込んだ手法の提案と実験を行い, より詳細な分析を行う予定である. また, 更に規模の大きいグラフでの実験や他の解析手法で適用できるかどうかの実験も合わせて行う.

参考文献

[1] AFP: 世界のウェブサイト数、10 億件を突破，AFP (オンライン)，入手先 ⟨http://www.afpbb.com/articles/-/3026121⟩ (参照 2015-08-27).

[2] Buehrer, G. and Chellapilla, K.: A scalable pattern mining approach to web graph compression with communities, *Proceedings of the 2008 International Conference on Web Search and Data Mining*, ACM, pp. 95–106 (2008).

[3] 片瀬弘晶，上田高徳，山名早人：LittleWeb 類似ノード集約による Web グラフ圧縮手法，*The 2nd Forum on Data Engineering and Information Management*, DBSJ, pp. E1–4 (2010).

[4] Kohlschütter, C., Chirita, P.-A. and Nejdl, W.: Efficient parallel computation of pagerank, *Advances in information retrieval*, Springer, pp. 241–252 (2006).

[5] Wicks, J. and Greenwald, A.: Parallelizing the computation of pagerank, *Algorithms and Models for the Web-Graph*, Springer, pp. 202–208 (2007).

[6] 片瀬弘晶， 松永拓，上田高徳， 田代崇，平手勇宇，山名早人：リンク構造解析アルゴリズム高速化のための縮小 Web の構築 (2008).

[7] 武井良太，新美礼彦：MapReduce による 2 段階 PageRank，*The 7th Forum on Data Engineering and Information Management*, DBSJ, pp. E5–4 (2015).

[8] Xu, X., Yuruk, N., Feng, Z. and Schweiger, T. A.: Scan: a structural clustering algorithm for networks, *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 824–833 (2007).

[9] Boldi, P. and Vigna, S.: The webgraph framework I: compression techniques, *Proceedings of the 13th international conference on World Wide Web*, ACM, pp. 595–602 (2004).

[10] シュウテイ， 片山薫：算術符号を用いたウェブグラフ表現のための圧縮方法，*The 6nd Forum on Data Engineering and Information Management*, DBSJ, pp. D6–1 (2014).