ノードの逐次集約による大規模グラフクラスタリングの高速化

塩川　浩昭† 　藤原　靖宏† 　鬼塚　真†

† NTT サイバースペース研究所　〒 239–0847 神奈川県横須賀市光の丘 1–1
E-mail: †{shiokawa.hiroaki,fujiwara.yasuhiro,onizuka.makoto}@lab.ntt.co.jp

あらまし　ソーシャルグラフに代表される大規模なグラフデータの登場により，グラフデータに対する高速なクラスタリング手法が重要な技術要素となっている．しかしながら，近年研究されてきた Louvain 法や Newman 法などのクラスタリング手法では，大規模なグラフデータに対して膨大な処理時間が必要となるだけでなく，計算対象となるノードの選択順序に依存して処理時間が大幅に増加する問題点が我々の検証により明らかになった．そこで本稿では大規模なグラフデータを対象としたグラフクラスタリングの高速化について取り組む．本稿で提案するクラスタリング手法はクラスタに含まれるノードの逐次集約，クラスタリング処理結果が自明なノードの枝刈り，クラスタリング処理時に参照するエッジ数抑制の 3 手法を用いることで，グラフクラスタリング処理の高速化を実現する．本稿では提案手法のプロトタイプを作成し，Louvain 法に対して，クラスタリング結果の質を表す指標である Modularity を同程度に示しつつ，クラスタリング処理速度を 100 倍程度高速化可能であることを示した..

キーワード　グラフ，クラスタリング，高速化

1. ま え が き

グラフデータはデータをノードとエッジで表現した基本的なデータ構造であり，情報推薦や情報検索，科学データ分析などの様々な分野で利用されている．特に近年では，数億ノードから構成される大規模なグラフデータが登場し，このようなグラフデータに対する高速な解析処理技術への需要が高まっている．例えば，Facebook では 2011 年に 1 日当たりのアクティブユーザ数が 5 億人，総会員数が 8 億人を突破したことが報告されている(注1)(注2)．このように，大規模のグラフデータは現実に存在し，今後もその規模をさらに増大させていくことが考えられ，これらのグラフデータに対する高速な解析手法は将来的に必要不可欠な技術となってくると言える．

前述したグラフデータ解析において，グラフデータを構成する膨大な数のノードを一定の尺度に従って自動分類するクラスタリングは非常に重要な要素技術のひとつであり，これまで様々なクラスタリング手法が研究されてきた [1, 2, 3, 4]．グラフデータにおけるクラスタリングでは，クラスタ内に存在するエッジが密であり，クラスタ間に存在するエッジの密度が疎となるようなクラスタを抽出することを目的としており，クラスタリングにより生成されたクラスタの良さを示す指標が幾つか提案されている [5, 6, 7]．その中でも Modularity [8] を用いた手法は大規模なグラフデータを高速にクラスタリングする手法として近年最も注目を集めている手法の一つである [9]．Modularity は処理対象のグラフデータをランダムグラフとみなしてモデル化し構築した指標であり，グラフがランダムグラフから離れれば離れるほど良いスコアを示す．すなわち Modularity は，よりクラスタ内のエッジが密であり，かつ，クラスタ間のエッジが疎であるクラスタリング結果であるほど良い Modularity のスコアを示し，より適切にグラフデータ内のクラスタを抽出できていることを表す特徴を持つ．ゆえに，この Modularity を用いたクラスタリング手法の研究 [8, 10, 11, 12, 13] では，Modularity のスコアを高速に最大化することが課題となる．しかし，Modularity の最大化は NP 困難 [14] であることから，Modularity を用いたクラスタリング手法では高速にかつ高いスコアの Modularity をいかにして求めるかということが研究課題となっている．

その中でも，Blondel らによる Louvain 法 [13] は，数千万ノード規模のグラフデータに対して，高速かつ高い Modularity のスコアを示す手法として知られている．Louvain 法では，グラフデータを構成する各ノードがそれぞれ別のクラスタである状態から処理を開始する．その後，任意の順にノードを選択し，選択したノードに対して隣接するノードの中から最も Modularity を向上させる隣接ノードを 1 つ選択し，同一のクラスタとみなす．同様の処理を Modularity が向上しなくなるまで繰り返した後，同一のクラスタと判定された全てのノードとエッジを 1 ノードへ集約する．集約するまでの処理を 1 反復 (1 パス) とし，集約したグラフデータに対して Modularity が向上する限りパスを繰り返す．従来手法 [10, 11, 12] ではクラスタ同士の統合を行う際に，クラスタに含まれる全てのノードとエッジを参照し，Modularity の向上量を計算する必要があった．これに対し，Louvain 法では収束したクラスタリング結果を 1 ノードへ集約することにより，クラスタ同士の統合処理に必要となるノードとエッジの参照数を削減させることに成功している．また，任意の順に選択されたノードが，それに隣接す

(注1)：http://blog.facebook.com/blog.php?post=409753352130
(注2)：http://www.facebook.com/f8?sk=app_283743208319386

表 1　Louvain 法における処理時間と Modularity 向上量の評価

パス	1	2	3	4	合計
Time(sec)	856.854	0.287	0.034	0.026	857.201
Modularity	0.564	0.004	0.001	0	0.569

るノードに対してのみ統合の可能性を検証するため，結果として生成されるクラスタのサイズに大きな偏りが生じることを防ぎ，良い Modularity の値を示すことがわかっている．

本研究では，前述のとおり高速に処理可能とされている Louvain 法について実在のグラフデータを用いて予備実験を行った．予備実験において，ノード数 36,692，エッジ数 367,662 から構成されるグラフデータに対して Louvain 法を適用した際の，処理時間と Modularity の変化の内訳を表 1 に示す．表 1 のパスは Louvain 法を実行した際のクラスタリング処理の反復回数，Time は各パスに消費した処理時間，Modularity は各パスで到達した Modularity のスコア値を表している．表 1 からも分かる通り，Louvain 法では，第 1 パスにその処理時間の 99%以上を消費している．また Modularity については，第 2 パス以降では大きな向上は見られないことがわかる．この傾向は，予備実験で使用したデータセットのみならず，他のデータセットにおいても同様に見られるものであることから，我々は Louvain 法における第 1 パスはグラフクラスタリング処理の性能向上に大きな影響力をもつものであると考えた．予備実験では，高々数万ノード規模のグラフデータのクラスタリングを対象にしている．しかしながら，本研究ではさらに大規模なグラフデータの処理を対象としており，第 1 パスで処理されるべきノード数は予備実験よりも格段に増加することから，現状高速とされている Louvain 法を以てしても，極端な処理性能の悪化の原因となる可能性があると考えた．

次に予備実験における処理時間の分布を図 1 に示す．このグラフは，表 1 に示したグラフデータに対して Louvain 法を用いて 50 回試行した際に，クラスタリング処理時間がどの時間幅にどれだけ存在したかを示したものである．このグラフからもわかるように，処理時間は広く分布しており，最悪処理時間は最良処理時間の約 2 倍程度の処理時間を有している．このように Louvain 法は，その処理の特性上クラスタリング処理に要する処理時間が大きく変動する可能性があるという特性を持っている．ゆえに，処理時間の平均値を見ると高速にみえる Louvain 法においても，処理の実行毎に極端な性能低下を起こす可能性を有していると考えられる．

そこで本稿では，より大規模なグラフデータを対象に，Modularity を用いたグラフクラスタリング処理のさらなる高速化について取り組む．提案手法では，グラフクラスタリング処理時において，ノードの逐次集約を行い，所属するべきクラスタが自明となるようなノードを枝刈りして不要なエッジへの参照を避けることにより，不要な計算を削減し，処理の高速化を実現する．具体的な処理の手順は次のとおりである．まず (1) 計算が不要となるノードの事前枝刈りによるノード参照数の削減し，(2) 次数の小さいノードを選択することによるエッジ参照数の削減する．最後に，(3) クラスタに含まれるノードを逐次

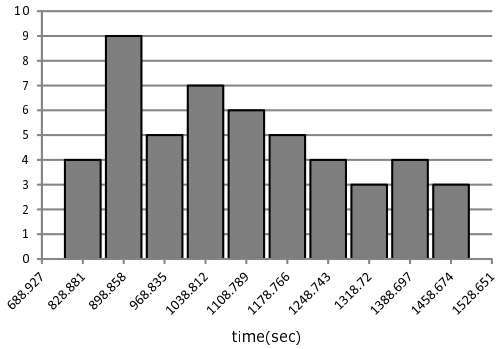

図 1　クラスタリング処理時間のぶれ幅

集約することによるノード参照数の削減を行うことにより，グラフクラスタリング処理の高速化を図る．これらの処理では，計算不要なノードの枝刈りとノードの逐次集約を繰り返し行うことで，既に所属すべきクラスタが自明であるノードを適宜計算対象から除外し，処理が参照すべきノード数を効率的に削減可能とする．また，次数の小さいノードの優先的な選択と逐次集約を併用することにより，ノードを 1 回参照する毎に計算が必要となるエッジ数の増加を抑制する．本稿では，提案手法のプロトタイプを実装し実データを用いた評価実験を行った．評価実験により，Louvain 法に対して 100 倍程度高速にクラスタリング処理可能であることを示した．

本稿の構成は以下の通りである．2 節で関連研究について述べ，3 節で本研究の前提となる概念と手法について説明する．4 節で提案手法の詳細を説明し，5 節で提案手法の評価を行う．最後に 6 節で本稿のまとめと今後の課題について述べる．

2. 関 連 研 究

Modularity を用いたグラフクラスタリング手法について様々な研究が行われている．代表的な手法として Girvan–Newman 法 [8]，Newman 法 [10]，CNM 法 [11]，WT 法 [12]，Louvain 法 [13] などが挙げられるが，いずれの手法においても本研究で対象とする大規模なグラフデータを高速に処理することは難しい．

Girvan らは，トップダウンにグラフデータを分割する Girvan–Newman 法 [8] を提案した．彼らの手法ではグラフデータ全体を 1 つのクラスタとみなし，クラスタ間を横断する可能性の高いエッジから順に切り離すことにより小さなクラスタへと分割していく．エッジを切り離す度に各エッジに付与されたスコアを再計算することから，エッジが疎なグラフデータに対してノード数を n とすると $O(n^3)$ の計算量を必要とする．計算量が膨大であるため，2004 年当時の計算機では 1 万ノード規模のグラフデータのクラスタリングには適用できないことが報告されている．

Newman は，貪欲法によりボトムアップにクラスタリングする Newman 法 [10] を提案した．この手法では，各ノードがそれぞれ別のクラスタである状態から処理を開始し，Modularity が最も向上するノード対を貪欲法により同一クラスタへと統合していく．統合させるべきノード対を全探索することから，エッ

ジが疎なグラフデータに対してノード数を n とすると $O(n^2)$ の計算量を必要とする．これに対し，Clauset らは Newman 法に対してヒープ構造を用いることにより Newman 法の高速化する CNM 法 [11] を提案し，$O(nlog^2n)$ の計算量まで高速化を実現した．また，脇田らはクラスタ統合時のクラスタサイズの不均衡による処理速度の低下を指摘し，クラスタ統合時に利用する Modularity の値をクラスタサイズで正規化するヒューリスティクスを用いることで CNM 法のさらなる高速化を行う WT 法 [12] を提案している．これらの手法では最大で 2007 年当時に計算機において数百万ノード規模までのグラフデータのクラスタリングが処理可能と報告されている．

Blondel らは，Newman 法の高速化手法として Louvain 法 [13] を提案した．Louvain 法は Modularity の局所最適化とノードの一括集約により，クラスタリング処理におけるノードとエッジの参照数の削減に成功し，既存手法の中で最も高速にクラスタリング処理が可能であるとされている．また，結果として生成されるクラスタのサイズに大きな偏りが生じることを防ぎ，既存手法の中で最も良い Modularity の値を示すことがわかっている．近年 Louvain 法について様々な拡張手法 [15, 16, 17] や応用 [18] が提案されているが，いずれの手法においても Louvain 法のさらなる高速化については取り組まれていない．これに対して本稿は大規模なグラフデータに向けた高速なクラスタリング処理を実現するために，Louvain 法のさらなる高速化に取り組む．Louvain 法の詳細について 3. 節にて述べる．

3. 事 前 準 備

3.1 クラスタリング指標 Modularity

本稿で提案するクラスタリング手法では，クラスタリング結果の良さを示す指標 Modularity を準最適化するようにクラスタリングを行う．そこで，クラスタリング指標 Modularity について概説する．

Modularity はクラスタ内に含まれたノード間のエッジが密であり，クラスタ間に存在するエッジが疎となる程良い値を示す指標である．グラフクラスタリング手法により抽出したクラスタの集合を C，クラスタ i からクラスタ j へ接続されているエッジ数を e_{ij}，グラフ全体に含まれる総エッジ数を m とするとき，Modularity Q は定義 1 のように定義される．Modularity が負の値を取る場合は $Q = 0$ とし，常に Modularity Q は $0 \leqq Q \leqq 1$ の値を示す．本研究では，この Modularity を用いたクラスタリング手法を対象とする．

［定義 1］ Modularity Q

$$Q = \sum_{i \in C} \left\{ \frac{e_{ii}}{2m} - \left(\frac{\sum_{j \in C} e_{ij}}{2m} \right)^2 \right\} \qquad (1)$$

3.2 既存手法：Louvain 法

本稿では現状最速とされる Louvain 法に対して，クラスタリング結果が自明なノードの枝刈り，次数順ノード参照によるエッジ参照数の削減，ノードの逐次集約によるノードとエッジの参照数削減を用いることにより，クラスタリング処理のさらなる高速化を行う．そこで本節では，既存のクラスタリング手法 Louvain 法について概説する．

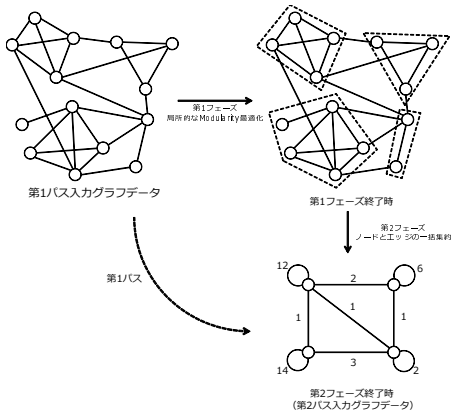

図 2 Louvain 法の処理の流れ

Louvain 法は 2 つのフェーズから構成されるグラフクラスタリング手法である．第 1 フェーズにて，各ノードと隣接するノード間のみにおいて Modularity の値を準最適化し，第 2 フェーズにて，第 1 フェーズで得られた結果を 1 ノードに一括集約する．この 2 つのフェーズの組みをパスと呼び，Modularity の値が上昇しなくなるまで，ノードが集約されたグラフデータに対してパスを繰り返し実行する．各フェーズの詳細について以下に説明する．

3.2.1 第 1 フェーズ：Modularity の局所最適化

第 1 フェーズではまず，入力されたグラフデータに対して，任意の順（ランダム順）にノードを選択する．その後，選択されたノードの全隣接ノードに対して，クラスタを統合した際に上昇する Modularity の値 $\triangle Q$ を計算する．この際に，定義 1 を計算するためには，統合予定のクラスタ対に含まれる全てのノードとエッジを参照する必要があり効率が悪い．しかしながら，Louvain 法において計算する必要がある値は，選択されたノードの属するクラスタに対して，その隣接ノードを統合した際の Modularity の変化量のみである．そこで Blondel らは，定義 1 から 2 つのクラスタ i，j を統合した際の Modularity 変化量 $\triangle Q$ を導出し，定義 2 のように定義している [13].

［定義 2］ Modularity 変化量 $\triangle Q$

$$\triangle Q = 2 \left\{ \frac{e_{ij}}{2m} - \left(\frac{\sum_{k \in C} e_{ik}}{2m} \right) \left(\frac{\sum_{k \in C} e_{jk}}{2m} \right) \right\} \qquad (2)$$

第 1 フェーズでは，上記の定義に従い $\triangle Q$ を計算し，最も Modularity 変化量 $\triangle Q$ が大きくなるクラスタ対を選出する．その後，選出したクラスタ対を同一のクラスタとして統合する．この処理は Modularity の値が向上しなくなるまで継続する．

3.2.2 第 2 フェーズ：ノードとエッジの一括集約

第 2 フェーズでは，第 1 フェーズで得られたクラスタリング処理結果に対してノードとエッジを一括集約し，クラスタリング処理で扱うグラフデータのサイズを縮小する．図 2 に示したように，同一のクラスタに含まれるノードは，1 つのノードにまとめられる．また，クラスタ内に存在するエッジについて

は，自己ループのエッジ 1 本へ集約し元のエッジ本数の 2 倍の重みをつける，クラスタ間に存在するエッジについては，同一のクラスタ対につきエッジ 1 本に集約し，エッジ本数分の重みをつける．第 2 フェーズで生成された集約された重み付きグラフデータは，次回以降のパスで入力データとして与えられる．

3.2.3 Louvain 法の課題

Louvain 法はクラスタリング処理時間の短縮について 2 つの課題を有している．各課題の詳細について以下に述べる．

a） 第 1 パスにおける処理時間の増加

Louvain 法は，第 1 パスの第 1 フェーズにおいて入力された全てのノードをランダム順に参照し，Modularity 変化量 $\triangle Q$ が増加する可能性を検証することで，クラスタの統合を行う．この処理は，Modularity 変化量 $\triangle Q$ が増加しなくなるまで継続されるため，入力されるグラフデータのノード数増加に伴い，第 1 パスにおける処理時間は増加することとなる．これにより，より大規模なグラフデータであるほど，第 1 パスにおけるクラスタリング処理時間がボトルネックとなり，クラスタリング処理時間の短縮が困難となる．

b） ノードの選択順序による処理時間の増加

Louvain 法は，各パスにおいてランダム順に計算対象とするノードを選出する．ゆえに，クラスタリング処理が参照する総エッジ数が実行毎に変化することになり，ノードの選択順序に応じてクラスタリング処理時間に差が生じることになる．したがって，ソーシャルグラフに代表されるような，大規模かつ各ノードの持つ次数の分布に偏りが大きなグラフデータであるほど，クラスタリング処理が参照する総エッジ数の差が顕著となると考えられる．ゆえに，クラスタリング処理時のノード選択順序の決定が処理時間短縮の課題となる．

本稿ではこれらの課題に着目しノードとエッジの参照数削減によるクラスタリング手法の高速化を手法を提案する．

4. 提 案 手 法

4.1 基本的なアイデア

本節では，提案手法の基本となるアイデアについて述べる．前節で述べた Louvain 法の課題に対して，提案手法ではクラスタリング処理において計算対象となるノード数とエッジ数を削減することにより，クラスタリング処理の高速化を実現する．具体的には，(1) 計算対象ノードの逐次集約による参照ノード数とエッジ数の削減，(2) クラスタリング結果が自明なノードの枝刈りによる参照ノード数とエッジ数の削減，(3) 次数順のノード選択による参照エッジ数の削減の 3 つの手法を用いる．以降 4.1.1 節，4.1.2，節 4.1.3 節にて各手法の詳細について述べる．

4.1.1 計算対象ノードの逐次集約

Louvain 法は，第 1 フェーズにおいて入力されたグラフデータに含まれる全てのノードを参照し，クラスタリング処理を実行する必要がある．これは，Louvain 法において，クラスタリング処理を行う第 1 フェーズとノードの集約処理を行う第 2 フェーズが分離していることに起因している．これにより，第 1 フェーズ中において同一のクラスタ内に複数のノードが存在

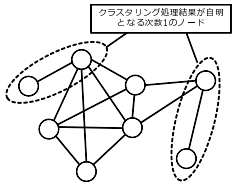

図 3 ノードの次数が 1 の場合

した状態で維持されることとなる．ゆえに，クラスタが変化し得ないノードや同一のクラスタに張られた複数本のエッジ，クラスタ内に存在するエッジへの不必要な参照が発生し，処理速度の低下をもたらす結果となっている．

そこで提案手法では計算対象ノードの逐次集約手法を導入する．逐次集約手法では，Louvain 法のようにフェーズを分離し一括して集約処理を行うのではなく，ノードが所属するべきクラスタが判明した時点でクラスタに含まれるノードを 1 ノードへ集約する．この際，クラスタに含まれるノードが持つエッジについては Louvain 法と同様に集約を行う．逐次的にノードを集約することで，各ノードが所属するクラスタの決定に伴いグラフデータのノード数とエッジ数を縮小することが可能となり，ノードやエッジに対する不必要な参照を削減する．

また，逐次集約手法は Louvain 法の第 1 フェーズにおけるクラスタリング処理の過程を各クラスタにつき 1 ノードで表現した手法である．そのため，本手法は Louvain 法の示すクラスタの隣接関係を維持したままクラスタリング処理することになる．ゆえに，**同一なノード選択順序が与えられれば Louvain 法と等価にクラスタリング処理可能**であり，Louvain 法の持つ高い Modularity 値を本手法も同様に示すことができる．

以上により，提案手法は Louvain 法と同等の高い Modularity を維持しつつ，Louvain 方よりもさらに高速にクラスタリング処理することを可能とする．

4.1.2 クラスタリング結果が自明なノードの枝刈り

従来の Louvain 法ではグラフデータに含まれる全てのノードに対して均等に参照される機会を与え，クラスタリング処理を実行している．しかしながら，グラフデータの中にはグラフクラスタリング処理を実行せずとも，ノードが所属するべきクラスタが自明となる場合が存在する．ゆえに，グラフクラスタリング処理の結果が自明となるノードを計算対象から除外してグラフデータのノード数を減少させることにより，グラフクラスタリング処理を高速化することができると考えられる．

そこでまず，本稿では 1 節に示した Modularity の定義を基に，グラフクラスタリング処理の結果が自明となるノードを定義する．クラスタリング処理結果が自明となるノードの定義をそれぞれ定義 3，定義 4 に示す．

［定義 3］ ノードの次数が 1 の場合

図 3 のように，ノードの次数が 1 である場合，ノードは明らかに隣接するノードと同一のクラスタに分類される．

［定義 4］ 隣接ノードが全て同一のクラスタである場合

図 4 のように，隣接するノードが全て同一のクラスタに含まれる場合，ノードは明らかに隣接するノードと同一のクラスタ

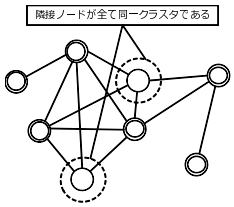

図 4　隣接ノードが全て同一のクラスタである場合

に分類される．

提案手法は，定義 3 と定義 4 に示されたノードをグラフデータから削除することで，計算対象ノード数の削減を実現する．ここで，定義 3 に示した，ノードの次数が 1 の場合についてはグラフデータが入力された時点において，容易に枝刈りすることが可能である．しかしながら，定義 4 に示した，隣接ノードが全て同一のクラスタである場合については，全ての隣接ノードが所属するクラスタが判明した後でなければ枝刈りすることができない．したがって，提案方式では次数順にソートされたヒープ木を構築し，ヒープ木に対して以下の処理を繰り返し，逐次的にグラフデータから計算不要となるノードを枝刈りをする．

（1） ヒープ木の先頭から次数 1 のノードを選択
（2） 選択したノードを隣接ノードに逐次集約
（3） ヒープ木を更新

提案手法は 4.1.1 節にて述べたとおり，Louvain 法において第 1 フェーズ終了後に一括して実行されたノード集約を，ノードが所属するべきクラスタが判明した時点で逐次的に実行するノードの逐次集約手法を採用している．ゆえに提案手法は，同一のクラスタに分類されたノードを逐次的に集約することにより，定義 4 に示したパターンの抽出が容易となる．定義 4 で定義されたパターンは，ノードの逐次集約を利用することで，最終的に次数 1 のノードとして表現されることになる．したがって，次数 1 のノードに対する枝刈りとノードの逐次集約を交互に実行することで，定義 3，定義 4 に示した両パターンを計算対象から外していくことが可能となり，クラスタリング処理の高速化を可能とする．また，4.1.1 節と同様に，提案手法はクラスタリング処理結果が自明なノードを枝刈りするものの，Louvain 法と同一の処理を行うことになるため，ノードの選択順序が同一である場合に Louvain 法と同等の Modularity を示すことが考えられる．

4.1.3　次数順のノード選択による参照エッジ数の抑制

従来の Louvain 法では，任意の順（ランダム順）にノードを選択し，グラフクラスタリング処理を実行していく．ゆえに，ノードの選択順序に依存してグラフクラスタリング処理時に参照するエッジ数が増減することから，グラフクラスタリングの処理時間が増減する可能性がある．Louvain 法において，次数の多い順にノードを選択した場合と次数が少ない順にノードを選択した場合では，後者の場合の方がエッジを参照する回数が少なくなると考えられる．

そこで提案手法では，4.1.2 節で述べたヒープ木を利用して次数の少ない順にノードを選択し，クラスタリング処理を行うことにより，エッジ参照数の増加を抑制する．具体的には以下の方針によりグラフクラスタリング処理中におけるエッジの参照数の抑制を行う．

（1） ヒープ木の先頭から次数の少ないノードを選択
（2） $\triangle Q$ が最大となる隣接ノードを選択
（3） 同一のクラスタとなるノードを逐次集約
（4） ヒープ木を更新

提案手法は，(1) により次数の少ないノードを優先的に選択し，かつ，(2)(3) により同一のクラスタに分類されたノードとエッジを逐次的に集約することで，エッジ参照数増加の抑制とグラフデータ中に含まれるエッジ数の削減を行う．以上により，クラスタリング処理の参照エッジ数増加を抑制し，処理の高速化を行う．

Algorithm 1 クラスタリング処理の流れ

Input: S
Output: Map
1: V_c に S に含まれる全ノードを加える
2: $Q = 0$
3: $Q_{max} = -1$
4: **while** $Q > Q_{max}$ **do**
5:　V_c から次数 1 のノードを削除
6:　次数が最も少ないノード $u \in V_c$ を取得
7:　ノード u の隣接ノード集合 $V_c(u) \subset V_c$ を取得
8:　$\triangle Q(u) = 0$
9:　**while** $V_c(u) \neq \emptyset$ **do**
10:　隣接ノード $u_a \in V_c(u)$ を取得
11:　$\triangle Q(u, u_a) = calc\_mod(u, u_a)$
12:　**if** $\triangle Q(u, u_a) > \triangle Q(u)$ **then**
13:　$\triangle Q(u) = \triangle Q(u, u_a)$
14:　**end if**
15:　$V_c(u)$ から $\{u_a\}$ を削除
16:　**end while**
17:　**if** $\triangle Q(u) > 0$ **then**
18:　ノード u とノード u_a を集約してノード (u, u_a) を生成
19:　ノード u とノード u_a の持つエッジを集約，更新
20:　Map 中のノード u，ノード u_a のクラスタを (u, u_a) に変更
21:　V_c から u と u_a を削除し，(u, u_a) を加える
22:　**else**
23:　$V_c = V_c - \{u\}$
24:　**end if**
25:　$Q_{max} = Q$
26:　$Q = calc\_mod(Map, S)$
27: **end while**
28: **return** Map

4.2　アルゴリズム

本節では，提案手法のアルゴリズムについて説明する．提案手法の処理の流れをアルゴリズム 1 に示す．また主な記号の定義を表 2 に示す．

このアルゴリズムは，グラフデータをノードと，それに隣接するノードのリストで表現したスパースマトリクス S を入力として受け取る．入力されたスパースマトリクス S からグラフを構成する全ノードを計算対象ノードとして加える（1 行）．その後，グラフ全体の Modularity の値 Q が向上し続ける限り，クラスタリング処理（4 行〜25 行）を継続する．クラスタリング処理は大きく分けて 4 つの処理の繰り返しから構成され，それぞれ計算不要ノードの枝刈り（5 行），処理対象ノードの選択（6〜7 行），統合するクラスタ対の選出（9〜16 行），ノードとエッジの逐次集約（17〜21 行）となっている．

5.　評 価 実 験

提案手法の性能を評価するために Blondel らによる Louvain

表 2　主な記号の定義

記号	定義
S	グラフデータのスパースマトリクス
Map	ノードとクラスタの対応を表すハッシュマップ
V_c	クラスタリングの計算対象ノード集合
$V_c(u)$	ノード u の隣接ノード集合
Q	Modularity の値
Q_{max}	最大 Modularity の値
$\triangle Q(a,b)$	ノード a とノード b 間の Modularity 変化量
$\triangle Q(a)$	ノード a の持つ最大の Modularity 変化量
$calc\_mod(Map,S)$	Map と S から全体の Modularity を計算

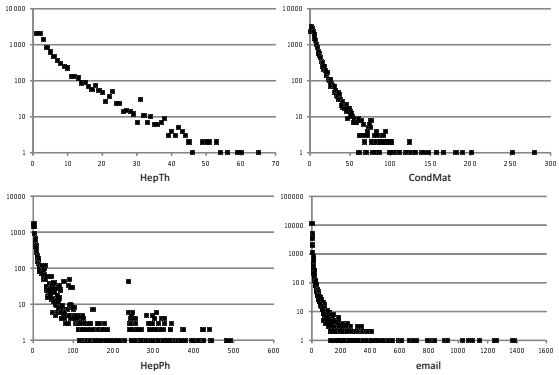

図 5　各データセットの次数分布

法 [13] と比較実験を行った．Louvain 法は Blondel らが報告しているとおり，従来のクラスタリング手法の中で最も速く，最も高い Modularity の値を示すことが知られている．

本実験で用いたデータセットは以下のとおりである．また，各データセットに対する次数分布を図 5 に示す．図 5 の各グラフは横軸はノードの次数，縦軸は各次数に対するノードの出現回数を表している．

- HepTh(注3)：論理物理学系の論文の共著者関係から取得したグラフデータである．ノード数は 9,877 であり，エッジ数は 51,971 である．
- CondMat(注4)：論文の共著者関係から得られたグラフデータである．ノード数は 23,133 であり，エッジ数は 186,936 である．
- HepPh(注5)：エネルギー工学系の論文の共著者関係から取得したグラフデータである．ノード数は 12,008 であり，エッジ数は 237,010 である．
- email(注6)：このデータは電子メールの送受信の関係から得られるグラフデータである．ノード数は 36,692 であり，エッジ数は 367,662 である．

本実験では，これらのデータセットに対して，クラスタリング処理が終了するまで処理を行った際の処理時間とクラスタリング結果の Modularity の値を示し，比較を行う．本実験には CPU が Intel Xeon Quad–Core L5640，メモリが 144GB の Linux サーバを利用した．また，全てのアルゴリズムは Java1.7 を用いて実装した．

(注3)：http://snap.stanford.edu/data/ca-HepTh.html
(注4)：http://snap.stanford.edu/data/ca-CondMat.html
(注5)：http://snap.stanford.edu/data/ca-HepPh.html
(注6)：http://snap.stanford.edu/data/email-Enron.html

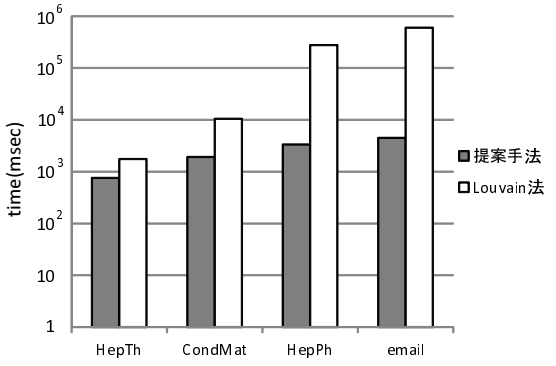

図 6　クラスタリング処理時間の比較

5.1　クラスタリングの高速性

図 6 の実験結果により，提案手法と Louvain 法のクラスタリング処理速度について比較する．なお，本実験における提案手法では，ノードの逐次集約処理，クラスタリング結果が自明なノードの枝刈り処理，次数順によるノード選択手法を利用している．

図 6 からわかるように，提案手法は全てのグラフデータにおいて，Louvain 法よりも高速にクラスタリング処理できていることがわかる．具体的には Louvain 法に対して，HepTh で 2.31 倍，CondMat で 5.47 倍，HepPh で 82.39 倍，email で 133,25 倍の高速化に成功している．さらに，グラフデータに含まれるエッジ数の増加に伴い，Louvain 法よりも大幅に速く処理ができていることがわかる．

提案手法では，同一のクラスタにラベル付けされたノードを逐次的に集約し，これによりクラスタリング処理が参照するノード数とエッジ数を削減する．ノード数やエッジ数が大きいグラフであるほど，集約による参照の削減がより効果的になり，Louvain 法と比べて処理時間が短縮されやすくなる考えられる．また，これに対して，規模の小さいデータセットであるほど，逐次集約による参照削減のメリットよりも，集約処理時に必要となる処理時間が占める割合が大きくなり，Louvain 法に対して大きな処理速度の差異が現れにくくなっていると考えられる．加えて，email のように Power–Law 特性が明確に現れたグラフデータにおいては，クラスタリング処理結果が自明なノードの枝刈りや次数順ノード選択が有効に作用し，処理速度の高速化に貢献していると考えられる．

5.2　クラスタリングの精度

表 3 に示した実験結果により，提案手法と Louvain 法のクラスタリング精度について比較する．

表 3 からわかるように，提案手法は各データセットにおいて，Louvain 法とほぼ同等，もしくは少し高い値を示している．前節でも述べたように，ノードの選択順序が同一である場合，提案手法は Louvain 法と等価なクラスタリング処理を実行することが可能であり，本実験もそれに準じた結果を示していることがわかる．一部，Louvain 法よりも高い値を示している結果では，脇田 [12] らによる研究でも指摘されているように，次数の少ない順にエッジを参照したことにより，グラフ中に含まれる

表 3　Modularity の比較

	HepTh	CondMat	HepPh	email
提案手法	0.744	0.703	0.615	0.562
Louvain 法	0.685	0.644	0.621	0.57

各ノードに対して均等に処理機会が与えられたためであると考えられる．

5.3　手法の有効性

本実験では提案手法で用いた，クラスタリング結果が自明なノードの枝刈りと樹数順ノード選択による参照エッジ数の抑制について，どの程度性能向上に寄与しているのか検証を行う．

5.3.1　クラスタリング結果が自明なノードの枝刈り

提案手法では，ノードの逐次集約手法に加えて，クラスタリング結果が自明なノードを計算対象から除外するための枝刈り手法を提案している．本実験では枝刈り手法が提案手法に対してどの程度処理時間削減に効果的であるかの検証を行う．本実験では，枝刈り手法の有無それぞれを実行した際のクラスタリング処理時間を比較する．なお，検証の際には枝刈り手法の有無にかかわらず，次数の少ない順にノードの選択を行うものとした．図 7 に枝刈り手法の有無によるクラスタリング処理時間の比較を結果を示す．

図 7 からもわかるように，全てのデータセットに対して枝刈り有の方が高速に処理できていることがわかる．具体的には，枝刈り無に対して HepTh で 1.09 倍，CondMat で 1.34 倍，HepPh で 1.27 倍，email で 1.30 倍の高速化に成功している．

提案手法では，次数が 1 となるノードの枝刈りと，逐次集約を組み合わせることにより，計算不要ノードの候補の選出を逐次行う．これにより，図 5 における CondMat や email の様な，ノード規模が比較的大きく，少ない次数に多くのノードが分布しているグラフデータに対する枝刈り処理が効率的に実行される．

これに対して，ノード規模が小さく次数に対するノードの分布の偏りが比較的小さな HepTh の様なグラフデータでは，処理の進行に伴い各ノードとも均一な次数を持ちやすくなる傾向があるため，枝刈りの候補となるノードが選出され難くなる．これにより，クラスタリング結果が自明なノードの枝刈りが処理速度向上に貢献しにくくなっていると考えられる．

以上のことからも，本枝刈り手法は，より Power–Law 特性が顕著に出たグラフデータに対して有効なのであると考えられる．

5.3.2　次数順ノード選択による参照エッジ数の抑制

提案手法では，ノードの逐次集約手法に加えて，クラスタリング処理時に次数の少ない順にノードを選択することで参照エッジ数の抑制を行なっている．本実験では，次数順ノード選択により，提案手法に対してどの程度処理時間削減に効果的であるのか検証を行う．本実験では，次数順ノード選択とランダム順ノード選択のそれぞれを実行した際のクラスタリング処理時間を比較する．なお，検証の際には次数順ノード選択の有無にかかわらず，クラスタリング結果が自明なノードの枝刈りは行わないものとした．図 8 に次数順ノード選択の有無によるクラスタリング処理時間の比較を結果を示す．

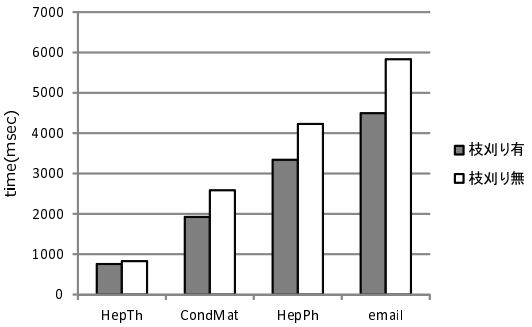

図 7　枝刈りの有無によるクラスタリング処理時間の比較

図 8 の結果からもわかるように，次数順ノード選択を用いることによりクラスタリング処理に要する時間を大幅に短縮できていることがわかる．具体的にはランダム順ノード選択に対して，HepTh で 1.36 倍，CondMat で 1.67 倍，HepPh で 1.02 倍，email で 1.80 倍となっている．

提案手法では，次数の少ない順にノードを選択することにより，参照すべきエッジ数の増加を抑制する．特に図 5 に示した email や CondMat の様な，Power–Low 特性の顕著なグラフデータに対して効率的に処理時間を削減できている．このような Power–Low 特性を持つグラフデータでは次数が少ないノードが参照されやすく，また逐次集約により次数が少ないノードが出現しやすい．そのため，次数順ノード選択による処理時間削減効率が良くなると考えられる．

これに対して HepTh や HepPh では，あまり良い高速化効率が得られていない．これは図 5 に示したように，これらのグラフデータにおける次数分布が原因であると考えられる．HepTh や HehPh などの次数の多いノードの出現頻度も比較的高いグラフデータでは，処理の進行に伴い，グラフデータの最低次数が Power–Law 特性を有するグラフデータと比較して大きくなる．したがって，次数順ノード選択を以てしても，参照エッジ数抑制効果が低くなり，高速化に寄与できなくなると考えられる．

以上の実験結果から，大規模なグラフデータであり，かつ，Power–Law 特性が顕著なグラフデータに対してより効率的にエッジ数の削減が可能になり，性能改善に貢献することができると考えられる．

5.4　考　　察

本節における実験データから，提案手法がより効率的に動作する環境を考察する．

図 6 の結果からもわかるように，エッジ数が大規模なグラフデータで有るほど，Louvain 法に対して相対的に高速にクラスタリング処理出ることがわかる．また，図 7，図 8 の実験結果から，次数分布において Power–Law 特性が顕著なグラフデータであるほど，効率的に枝刈りや参照エッジ数の抑制が可能となり，高速にクラスタリング処理できる可能性を示唆している．したがって，本稿で提案するクラスタリング手法は，より大規模かつ，次数分布の偏りが顕著なソーシャルグラフや Web グ

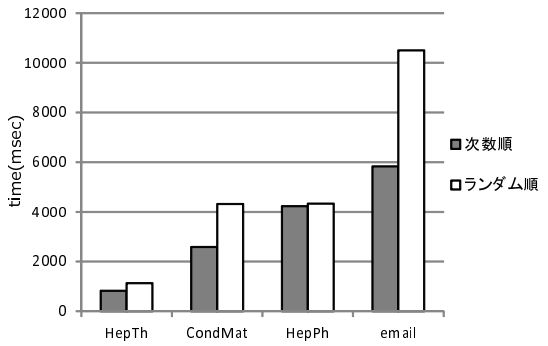

図 8　樹数順ノード選択の有無によるクラスタリング処理時間の比較

ラフのクラスタリング処理に向いていると考えられる．

6. おわりに

本稿では，大規模なグラフデータに対して，高速にクラスタリング処理を行う手法を提案した．提案手法はまず (1) 計算が不要となるノードの事前枝刈りによるノード参照数の削減し，(2) 次数の小さいノードを選択することによるエッジ参照数の削減する．最後に，(3) クラスタに含まれるノードを逐次集約することによるノード参照数の削減を行うことにより，グラフクラスタリング処理の高速化を実現した．本稿では提案手法のプロトタイプを作成し評価実験で比較することで，提案手法は Louvain 法に対して，同等の Modularity 値を保ちながらも 100 倍程度処理を高速化できることを示した．これにより，従来最速とされてきた Louvain 法と比べ，より大規模なグラフデータに対して有効な手法であることを明らかにした．

また，本稿で提案したクラスタリング手法では，クラスタリング結果が自明なノードの枝刈り，次数順ノード選択といった処理を有している．これにより，Power–Law 特性が顕著なグラフデータに対してより高速に処理できる可能性を評価実験により示した．ゆえに，本稿で提案したクラスタリング手法は，エッジ数が特に多く，Power–Law 特性が明確なソーシャルグラフデータや Web グラフデータに対してより有効な手法であると考えられる．

今後は，次の点について継続して研究を行う予定である．まず本提案手法に対して，より大規模なグラフデータ（数億ノード以上）を用いた評価を行う必要がある．これにより，提案手法の性能を評価すると共に，大規模なグラフデータのクラスタリング処理におけるさらなる課題を抽出する．続いて，本提案手法や既存の Louvain 法，CNM 法を中心に，グラフクラスタリング処理の分散並列化について検討する．これまで，クラスタリング処理の分散並列化については，BSP モデルに基づいた Yuzhou らによる研究 [19] がよく知られている．しかしながら，Yuzhou らによる手法では，BSP モデルの同期やメモリ上の制限により大きな処理待ち時間が発生し，本研究で対象とするような処理時間を実現することができないことが指摘されている．そこで，グラフクラスタリング処理のさらなる高速化に向けた分散並列化について検討する予定である．

文　献

[1] B W Kernighan and S Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical journal*, 49(2):291–307, 1970.

[2] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[3] Stijn Van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, 2000.

[4] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, December 2007.

[5] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[6] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51:497–515, May 2004.

[7] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[8] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004.

[9] Charu C. Aggarwal. *Social Network Data Analytics*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[10] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, Jun 2004.

[11] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, Dec 2004.

[12] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007*, pages 1275–1276, 2007.

[13] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008, October 2008.

[14] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On finding graph clusterings with maximum modularit. In *Graph-Theoretic Concepts in Computer Science*, pages 121–132, 2007.

[15] Xianchao Zhang, Liang Wang, Yueting Li, and Wenxin Liang. Extracting local community structure from local cores. In *DASFAA Workshops*, pages 287–298, 2011.

[16] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. Generalized louvain method for community detection in large networks. In *Intelligent Systems Design and Applications*, 2011.

[17] Qi Ye, Bin Wu, Zhixiong Zhao, and Bai Wang. Detecting link communities in massive networks. In *Advances in Social Network Analysis and Mining*, 2011.

[18] Nam P. Nguyen, Thang N. Dinh, Ying Xuan, and My T. Thai. Adaptive algorithms for detecting community structure in dynamic social networks. In *IEEE INFOCOM*, pages 2282–2290, 2011.

[19] Yuzhou Zhang, Jianyong Wang, Yi Wang, and Lizhu Zhou. Parallel community detection on large networks with propinquity dynamics. In *KDD*, pages 997–1006, 2009.