分散グラフ処理におけるグラフ分割の技術評価

藤森　俊匡† 　塩川　浩昭†† 　鬼塚　真†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1 丁目 5 番
†† 筑波大学 計算科学研究センター　〒 305–8573 茨城県つくば市天王台 1 丁目 1 番 1 号
E-mail: †{fujimori.toshimasa,onizuka}@ist.osaka-u.ac.jp, ††shiokawa@cs.tsukuba.ac.jp

あらまし　現実世界で見られるグラフデータの大規模化に伴い，グラフデータに対する分析処理を高速に行う分散グラフ処理フレームワークに対する需要が高まっている．分散グラフ処理フレームワークでは，入力となるグラフを分割し各計算機に割り当ててから分析処理を行うが，最初のグラフ分割の品質によってその後の分析処理に要する時間が大きく左右される．我々は先行研究において，グラフを高速に分割するとともにその後の分析処理を高速化するグラフ分割手法を提案した．先行研究では，本手法の効果が入力となるグラフデータの性質に依存することが明らかになっており，本手法において効果的なグラフデータの性質はまだ検証されていない．したがって本稿では，先行研究に加えたさらなる実験を行うことで，本手法が効果的であるグラフの特徴を調査し，本手法の特性を明らかにした．

キーワード　グラフ分割, グラフマイニング, 分散処理

1. はじめに

グラフデータに対する分析処理は，データ間の関係性を考慮した分析ができることから様々な分野で利用されている．近年，多くの分野において扱うデータの大規模化が進み，それに伴って大規模なグラフデータに対する分析処理の重要性が高まっている．例えば，Facebook の月間アクティブユーザ数は，2015 年 9 月時点で 15 億 5000 万人であり，前年度比で 14 %増加したと報告されている (注1)．ここで，ユーザを頂点，ユーザ間のつながりをエッジとすれば，Facebook におけるユーザとそのつながりをきわめて巨大なグラフデータとして表現できる．このような大規模なグラフデータを高速に処理するための技術として，分散グラフ処理フレームワークへの注目が高まっている．

著名な分散グラフ処理フレームワークとして，Pregel [1] や GraphLab [2]，PowerGraph [3]，GraphX [4] 等が挙げられる．これらのフレームワークは共通して，分析処理の前に，入力となるグラフを計算機台数と同じ数の部分グラフに分割し，それらを各計算機に割り当てるという処理を行う．分析処理中はグラフデータの再割り当ては行われず，各頂点に紐付けられたデータを，頂点間でやりとりしながら繰り返し処理によって更新していく．グラフ分割が分析処理の性能に影響を与える要素は，計算機間の通信コストとタスク量の偏りの二つがある．例えば，グラフ分割によって切断された頂点またはエッジは計算機間をまたがって存在することになるが，それらを通じてデータのやりとりが行われる場合，計算機間で通信が発生する．したがって，グラフ分割によって切断される頂点またはエッジの数が少ないほど分析処理中の通信コストは小さくなる．また，グラフデータに対する分析処理の多くは，処理するエッジ数が多いほどタスク量が増大する [11] ことから，他の計算機に比べ多くのエッジを割り当てられた計算機は，分析処理に要する時間が長くなる．分散グラフ処理フレームワークではグラフデータの再割り当てを行わないことから，処理時間が長くなった計算機以外の計算機は，処理を終えたあと待機状態になり非効率的である．したがって，各計算機にエッジ数が同程度（等粒度）となるような部分グラフを割り当てることによって，処理時間の偏りを小さくすることができる．

精度よくグラフ分割を行うための方法として，METIS [10] を用いる方法が挙げられる．METIS は優れたグラフ分割手法を複数実装したライブラリであり，等粒度性を高めながら切断エッジ数が少なくなるようにグラフを分割する．METIS は精度よくグラフ分割を行うことができるものの，入力とするグラフデータのサイズが大きくなると，グラフ分割処理に膨大な時間を要するという欠点がある．近年の分散グラフ処理フレームワークでは，グラフ分割を高速に行うために，頂点やエッジの割り当て先をそれらのデータを読み込んだ時点で逐次決定するグラフ分割手法を採用することも多い．これらの手法はグラフを高速に分割することができるものの，METIS などのグラフデータ全体の構造を利用する方法に比べグラフ分割の精度が低いという欠点がある．

我々は先行研究において，少ない切断エッジ数と高い等粒度性をバランスよく両立することで，分散グラフ処理フレームワークにおける分析処理を高速にするグラフ分割手法を提案した [17]．本手法は，グラフクラスタリング指標のひとつである Modularity の値が大きくなるようにグラフ分割を行うと同時に，高い等粒度性を実現する．本手法に採用されている手法 [5] は，従来のグラフ分割手法に比べ高速に，かつ高い精度でグラフ分割を行うことが可能である．一般的に，Modularity の値が大きくなることで，部分グラフが内包するエッジ数は多くなり，部分グラフ間に存在するエッジ数は少なくなることが多い．そのため，多くの場合において Modularity の値を大きくすることでグラフの切断エッジ数を少なくすることができる．先行研究では，本手法を既存の分散グラフ処理フレームワークであ

(注1)：http://investor.fb.com/releasedetail.cfm?ReleaseID=940609

る PowerGraph に組み込み，複数のデータを用いて性能評価を行った．そして，本手法を用いることにより，従来手法を用いた場合に比べ分析処理を高速化できることを確認した．しかし，本手法による分析処理の高速化の効果は適用するグラフデータによって異なることが分かったものの，その違いがグラフデータのどのような特徴に起因するかの分析が不十分であった．また，本手法が着目している Modularity と等粒度性が，通信コストとタスク量の偏りにどの程度影響しているかの分析も不十分であった．

そこで本稿では，通信コストとタスク量の偏りの 2 つの観点から，本手法のさらなる検証を行った．まず，グラフ分割における通信コストを評価するための指標であるレプリケーションファクタ [3] と，タスク量の偏りを評価するための指標であるロードバランスファクタの 2 つの指標による評価を行った．そして，本手法の従来手法に対する通信コスト削減の効果と，グラフデータの持つ Modularity の値との関係性を実験により明らかにした．また，本手法を PowerGraph に組み込み，実際に分析処理を行った場合の通信バイト量および各計算機の処理時間の偏りを計測し，従来手法との評価比較を行った．そして，計算機台数が変化した場合の分析処理時間，グラフ分割時間および合計実行時間を計測することで，本手法のスケーラビリティに関する評価を行った．

本稿の構成は以下の通りである．2. 節でグラフ分割の精度を評価するための指標と本手法の詳細について概説する．3. 節では本手法の評価と分析を行う．4. 節で関連研究について述べ，5. 節で本稿をまとめ，今後の課題について述べる．

2. 前提知識

2.1 レプリケーションファクタとロードバランスファクタ

先行研究では，提案した手法を既存の分散グラフ処理フレームワークである PowerGraph [3] に組み込み性能評価を行った．PowerGraph ではグラフ分割手法として，エッジを各部分グラフに一意に割り当てて頂点を切断する vertex-cut 方式を採用している．vertex-cut 方式を採用しているフレームワークでは，頂点が切断されると，その頂点と連結するエッジが割り当てられた計算機上に頂点の複製が格納される．これらの複製間では，分析処理中にデータの整合性を保つための通信が行われるため，グラフ分割によって生成される頂点の複製が少ないほど通信コストを抑えることができる．vertex-cut 方式のグラフ分割における通信コストを評価するための指標としてレプリケーションファクタがある [3]．レプリケーションファクタは各頂点の平均複製数であり，グラフデータの頂点集合を V，頂点 $v \in V$ の複製の集合を $R(v)$ とした時，以下の式で定義される．

$$\frac{1}{|V|}\sum_{v\in V}|R(v)| \tag{1}$$

vertex-cut 方式のグラフ分割手法には，上記のレプリケーションファクタの値を小さくすることが求められる．

また，分散グラフ処理フレームワークにおけるグラフ分割では，タスク量の偏りを小さくするために各計算機に割り当てるエッジ数を同程度（等粒度）にすることが求められる．vertex-cut 方式における負荷の偏りを評価する場合には，以下のような式 [3] が用いられる．

$$\max_{m\in M}|E(m)| < \lambda\frac{|E|}{|M|} \tag{2}$$

ここで，E はグラフデータのエッジ集合，M は全計算機からなる集合，$E(m)$ は計算機 m に割り当てられているエッジ集合を表す．また，λ は 1 以上の小さな実数であり，グラフ分割によるタスク量の偏りをどの程度許容するかを表すパラメータである．すなわち式 (2) は，各計算機に割り当てられたエッジ数の中でもっとも大きなものが，各計算機ごとの平均エッジ数の何倍までを許容するかということを表している．ここで，式 (2) から以下のような指標を導くことができる

$$\lambda = \frac{|M|}{|E|}\max_{m\in M}|E(m)| \tag{3}$$

本稿では，式 (3) の λ をロードバランスファクタと呼び，等粒度性を評価するための指標として用いる．

2.2 Modularity

先行研究により提案した手法では，切断エッジ数を削減するために，グラフデータの Modularity [6] が高くなるようにクラスタのマージを行う．そこで，本節ではグラフクラスタリング指標の一つである Modularity について概説する．

Modularity は，実際にクラスタが内包するエッジ数が，ランダムにエッジを配置した場合にクラスタが内包するであろうエッジ数の期待値とかけ離れているほど大きい数値を示す．一般的に，Modularity の値が大きいときは，クラスタに内包されるエッジは密であり, クラスタ間に存在するエッジ (切断エッジ) は疎となっていることが多い．そのため，Modularity の値を大きくすることで，切断エッジ数を削減できると考えられる．グラフ分割により得られるクラスタの集合を C，クラスタ i からクラスタ j へ接続されているエッジの集合を E_{ij} とした時，Modularity の値 Q は以下の式で定義される．

$$Q = \sum_{i\in C}\left\{\frac{|E_{ii}|}{2|E|} - \left(\frac{\sum_{j\in C}|E_{ij}|}{2|E|}\right)^2\right\} \tag{4}$$

先行研究で提案した手法は，グラフデータの Modularity が高くなるようにクラスタをマージしていくが，マージのたびに上式の Modularity の値の計算を行うのは非効率的である．したがって本手法では，マージするクラスタの組を決定する際に，そのクラスタの組をマージした場合の Modularity の変化量のみを計算することで効率化を図っている．Clauset らは，上記の Modularity の定義式から，隣接する 2 つのクラスタ i, j を統合した際の Modularity の変化量 ΔQ_{ij} を導出し, 以下のように定義している [12].

$$\Delta Q_{ij} = 2\left\{\frac{|E_{ij}|}{2|E|} - \left(\frac{\sum_{k\in C}|E_{ik}|}{2|E|}\right)\left(\frac{\sum_{k\in C}|E_{jk}|}{2|E|}\right)\right\} \tag{5}$$

本手法では，上記の ΔQ_{ij} の値とそれぞれのクラスタが内包するエッジ数の比率を合成した指標を用いる．詳しくは 2.3.1 節で述べる．

2.3 先行研究で提案したグラフ分割手法

我々は先行研究において，分散グラフ処理フレームワークにおける分析処理を高速化するグラフ分割手法を提案した [17]. 本手法は，等粒度性を保ちつつ Modularity の値が大きくなるようにグラフを分割することで，分析処理におけるタスク量の偏りと通信コストを抑制し分析処理の高速化を実現する．また，先行研究では，エッジを各部分グラフに一意に割り当てて頂点を切断する edge-cut 方式である本手法を，vertex-cut 方式のグラフ分割を採用している PowerGraph に組み込み性能評価を行っている．以下，2.3.1 節で本手法の詳細について述べ，2.3.2 節で本手法の vertex-cut 方式への変換手順について述べる．

2.3.1 本手法の詳細

本手法は，グラフデータの全ての頂点が異なるクラスタに属した状態からスタートし，クラスタの組をマージしていくことで最終的に k 個 (計算機台数個) のクラスタを導出する．本手法は，2 つのステップからなる．一つ目のステップは Modularity クラスタリングステップである．このステップでは，クラスタの等粒度性を保ちつつ Modularity の値が大きくなるようにクラスタをマージしていく．このステップでは k 個よりも多いクラスタを導出し，次のステップで最終的な k 個のクラスタを得る．二つ目のステップは等粒度クラスタリングステップである．このステップでは，最終的に得られるクラスタの等粒度性が高くなるように隣接するクラスタをマージしていく．以下でそれぞれのステップについて説明する．

1) Modularity クラスタリングステップ

Modularity クラスタリングステップでは，等粒度性を保ちつつ Modularity が高くなるようにクラスタをマージしていく．このステップでは，あるクラスタ i とマージするクラスタを，以下のような式で定義される指標 $\Delta Q'_{ij}$ [13] を用いて決定する．

$$\Delta Q' = min\left(\frac{|E_i|}{|E_j|}, \frac{|E_j|}{|E_i|}\right) \times \Delta Q \tag{6}$$

ここで，E_i はクラスタ i が内包するエッジの集合である．Modularity クラスタリングステップでは，上記の $\Delta Q'_{ij}$ の値が最も大きくなるようなクラスタの組 i, j をマージする．なお，あるクラスタ i に対して，$\Delta Q'$ が 0 より大きくなるようなクラスタ j が存在しなかった場合，クラスタ i はどのクラスタともマージを行わない．ここで，ΔQ は，2.2 節で述べたクラスタ i, j をマージした場合の Modularity の変化量 (式 (5)) である．一方，$min\left(\frac{|E_i|}{|E_j|}, \frac{|E_j|}{|E_i|}\right)$ は等粒度性を評価するためのものであり，クラスタ i, j の内包エッジ数の差が少ないほど高い数値を示す．Modularity を評価するための指標と等粒度性を評価するための指標を統合することにより，等粒度性を保ちつつ Modularity が高くすることができる．

Modularity クラスタリングステップでは，塩川らによるクラスタの逐次集約手法 [5] を用いる．この手法は，クラスタのマージを逐次的に行い，かつ隣接クラスタの少ないクラスタから順にマージ判定を行うことで，Louvain 法 [8] のように複数クラスタをまとめてマージし，かつマージ判定の際に参照するエッジの数を削減することができる．これにより，ランダムにクラスタをマージしていく場合よりも高速にグラフクラスタリングを実行できる．

なお，Modularity クラスタリングステップでは k 個のクラスタを導出するということはせず，Modularity の値を大きくするようなクラスタの組が存在しなくなるか，クラスタ数が $a * k$ になった時点で二つ目のステップに移る．ここで，a はユーザが指定する 1 以上の実数である．

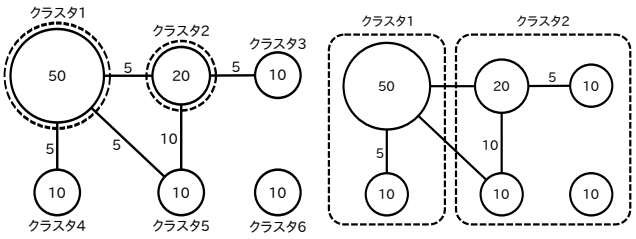
2) 等粒度クラスタリングステップ

等粒度クラスタリングステップでは，最終的に得られるクラスタの等粒度性が高くなるようにクラスタをマージしていく．具体的な手順は以下のとおりである．ここで，等粒度クラスタリングステップの説明において，クラスタの内包エッジ数の大きさをクラスタのサイズと表現することとする．まず，サイズが大きいクラスタの上位 k 個を選択し，それらを種クラスタとする．この種クラスタと隣接するクラスタをマージさせていくことで，最終的に k 個のクラスタを導出する．サイズの大きなものを種クラスタとして選択するのは，ステップの終盤で種クラスタとサイズの大きなクラスタがマージされた結果，最終的な等粒度性が大きく損なわれてしまうという状況を防ぐためである．また，種クラスタと隣接するクラスタとをマージすることで，クラスタ間に存在していたエッジが種クラスタに内包されるため，切断エッジ数を削減できる．

種クラスタとその隣接クラスタとのマージは，次のような手順で行う．(1) サイズが最少の種クラスタを取得する．(2) 取得した種クラスタと隣接クラスタとをマージする．(3) 取得したクラスタよりもサイズの小さな種クラスタが存在しない場合，処理 (2) を繰り返す．(4) 取得したクラスタよりもサイズの小さな種クラスタが存在した場合，処理 (1) に戻る．このような処理を行うことで，種クラスタの等粒度性を保ちながらクラスタをマージしていくことができる．最終的に得られる k 個のクラスタの平均内包エッジ数はグラフの切断エッジ数によって異るためにこの方法をとる．

しかし，この方法でクラスタのマージを続けていくと，種クラスタからエッジを通じて辿りつけないクラスタはマージを行うことができない．それらのクラスタは通常，種クラスタと比べるとサイズがきわめて小さいことから，以下のような方法で種クラスタとのマージを行う．(1) 種クラスタから辿りつけないクラスタを一つ取得する．(2) 取得したクラスタが隣接クラスタを持っていた場合，それらをマージする．(3) 取得したクラスタが隣接クラスタを持たなくなるまで処理 (2) を繰り返す．(4) 取得したクラスタが隣接クラスタを持たなくなったら，そのクラスタをサイズが最小の種クラスタとマージする．(5) 種クラスタから辿りつけないクラスタが存在していた場合，処理 (1) に戻る．このような処理を行うことによって，切断エッジ数を削減しつつ種クラスタの等粒度性を高めることができる．

等粒度クラスタリングステップにおけるマージの例を図 1 に示す．図中の円がクラスタを表し，円内の数値が内包エッジ数を，線分上の数がクラスタ間エッジ数を表す．例では，$k = 2$ であり，図 1(a) のグラフを 2 つのクラスタに分割したいとす



(a) 初期状態　　(b) 終了状態

図 1　等粒度クラスタリングステップのマージ例

る．まず，種クラスタとしてサイズが大きいクラスタ上位 2 つであるクラスタ 1 とクラスタ 2 が選ばれる．次に，種クラスタ内でサイズが最小であるクラスタ 2 と隣接クラスタとのマージを行う．添字の数値が小さいクラスタから優先してマージを行うとすると，クラスタ 2 とクラスタ 3 がマージされる．その結果クラスタ 2 の内包エッジ数が 35 となるが，クラスタ 2 よりサイズの大きい種クラスタが存在しないため，クラスタ 2 のマージ処理を続けクラスタ 5 とマージする．クラスタ 2 の内包エッジ数が 55 となりクラスタ 1 の内包エッジ数を上回るため，今度はクラスタ 1 がサイズが最小の種クラスタとなる．その後，クラスタ 1 とクラスタ 4 がマージされ，クラスタ 1 の内包するエッジ数が 65 となる．種クラスタから辿りつけるクラスタはもう存在しないため，残ったクラスタ 6 とサイズが最小の種クラスタであるクラスタ 2 とをマージする．最終的に，図 1(b) のような状態になる．内包するエッジ数はクラスタ 1，クラスタ 2 ともに 65 であり，切断エッジ数は 10 となる．

2.3.2　vertex-cut 方式への変換

本手法は，edge-cut 方式のグラフ分割手法である．しかし，本稿では本手法を vertex-cut 方式を採用しているフレームワークである PowerGraph に組み込み性能評価を行っている．edge-cut 方式から vertex-cut 方式への変換は，クラスタ間のエッジをどちらかのクラスタに振り分けることにより簡単に行うことができる．本手法では，クラスタ間に存在するあるエッジに着目した時点で，内包エッジ数が少ない方のクラスタに振り分けるという方針を取っている．これにより，最終的なクラスタの等粒度性を高めることができる．また，切断エッジ（クラスタ間エッジ）を振り分ける処理によって頂点が切断される．したがって，切断エッジ数を削減することは切断される頂点を削減することにもつながり，分析処理中の通信コストを抑えることが可能となる．

3.　評価 · 分析

先行研究で提案したグラフ分割手法 [17] についてのさらなる検証を行うために，通信コストとタスク量の偏りの 2 つの観点から本手法の性能を評価するための実験を行った．本実験では，本手法を既存の分散グラフ処理フレームワークである PowerGraph [3] に組み込み，以下のような性能評価を行った．

（1）　グラフ分割指標による評価

グラフ分割の精度を評価するための指標であるレプリケーションファクタ (式 (1)) とロードバランスファクタ (式 (3)) による評価を行った．

表 1　比較実験で用いたデータセット

データ名	$\|V\|$	$\|E\|$	Modularity
email-EuAll [15]	265,214	420,045	0.779
web-Stanford [15]	281,903	2,312,497	0.914
com-DBLP [15]	317,080	1,049,866	0.806
web-NotreDame [15]	325,729	1,497,134	0.931
amazon0505 [15]	410,236	3,356,824	0.852
web-BerkStan [15]	685,230	7,600,595	0.930
web-Google [15]	875,713	5,105,039	0.974
soc-Pokec [15]	1,632,803	30,622,564	0.633
roadNet-CA [15]	1,965,206	2,766,607	0.992
wiki-Talk [15]	2,394,385	5,021,410	0.566
soc-LiveJournal1 [15]	4,847,571	68,993,773	0.721
uk-2002 [16]	18,520,486	298,113,762	0.986
webbase-2001 [16]	118,142,155	1,019,903,190	0.976

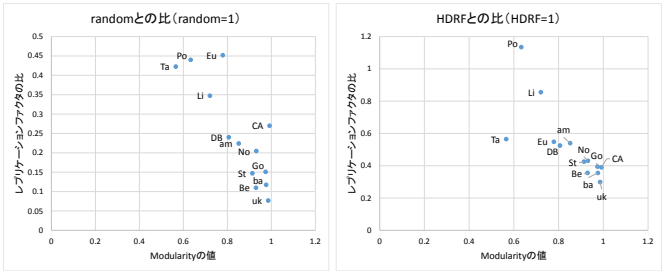


図 2　Modularity と通信コストの関係性

（2）　分析処理時の通信コストと負荷の偏りの評価

実際に分析処理を行った場合の通信バイト量と各計算機の処理時間の偏り，分析処理に要した時間を計測し比較を行った．

（3）　スケーラビリティの評価

計算機台数を変化させた場合の分析処理時間，グラフ分割時間および合計実行時間を計測し，スケーラビリティの評価を行った．

上記の性能評価では，本手法に対する比較対象として，PowerGraph に用いられている既存のグラフ分割手法である random, oblivious [3], HDRF [9] についての性能評価も行った．random は，頂点およびエッジの割り当て先となる計算機をハッシュ値を用いて決める最も単純な手法である．oblivious は，高い等粒度性を実現しつつ random よりもレプリケーションファクタが小さくなるようにグラフデータを分割するヒューリスティックな手法である．HDRF は oblivious を改良した手法であり，多くのグラフデータにおいて oblivious と同等かそれ以上の精度でグラフデータを分割できる．また，本手法に関しては，Modularity クラスタリングステップの終了条件となるクラスタ数 $a * k$(a はユーザ指定の 1 以上の実数，k は計算機台数) を変化させて上記の性能評価を行った．

本実験で用いたデータセットを表 1 に示す．いずれも，ソーシャルグラフや Web グラフなどの実世界のデータに関するグラフデータである．

本実験では，実験環境として Amazon EC2 を使用する．使用したインスタンスは r3.2xlarge の linux インスタンスである．CPU は Intel(R) Xeon(R) CPU E5-2670 v2 であり，ク

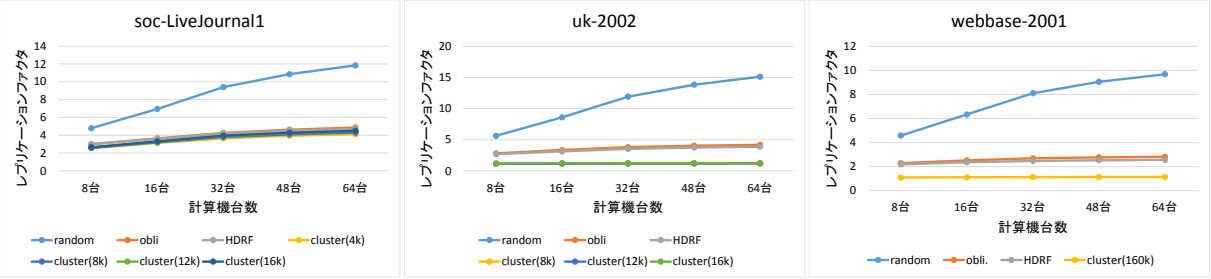


図 3　レプリケーションファクタ

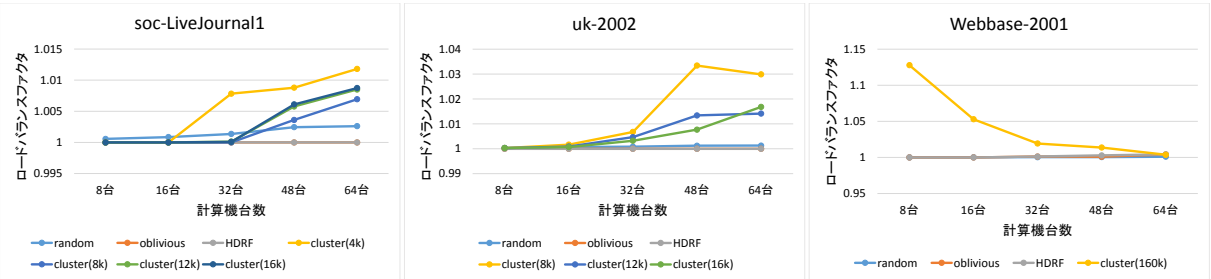


図 4　ロードバランスファクタ

ロック数は 2.50GHz，コア数は 4 である．メモリは 60GB のものを使用した．また，インスタンス間のデータ伝送速度はおよそ 1.03Gbps であり，hdparm を-t オプションをつけ実行することで得られたディスク読み込み速度はおよそ 103MB/sec である．先行研究で提案した手法の実装には C++を用いた．コンパイルに使用した g++のバージョンは 4.8.1 であり，最適化オプションとして-O3 を使用している．また，PowerGraph も C++で実装されており，上記と同様の条件でコンパイルを行った．

3.1　グラフ分割指標による性能評価

本実験では，表 1 の各グラフデータに対し，各グラフ分割手法を用いた場合のレプリケーションファクタおよびロードバランスファクタを計測した．本節では，レプリケーションファクタおよびロードバランスファクタを用いた評価 · 分析の結果を示す．

3.1.1　Modularity と通信コストとの関係性

本手法は分析処理中の通信コストを抑えるために，Modularity の値が大きくなるようにグラフを分割することで切断エッジ数を削減するという方法をとっている．そこで本節では，グラフデータの持つ Modularity の値と通信コストとの関係性についてまとめた結果について示す．通信コストの評価にはレプリケーションファクタの値を用い，従来手法である random および HDRF に対してどれだけレプリケーションファクタの値を抑えられたかによって本手法の効果を評価した．図 2 に，表 1 の各グラフデータにおける Modularity の値と通信コストとの関係性を示す．横軸はグラフの持つ Modularity の値である．縦軸は，従来手法を 1 とした場合の従来手法と本手法とのレプリケーションファクタの比である．なお，計算機台数は 64 台である．図 2 から，多くのグラフデータにおいて，グラフデータの持つ Modularity の値が大きくなるほど従来手法に対する本手法の通信コストに関する優位性は高くなることが分かる．Modularity とレプリケーションファクタの比の相関係数は，random の場合は −0.846，HDRF の場合は −0.758 であった．この結果から，本手法による通信コスト削減の効果が高いグラフデータとは，Modularity の値が高いデータであるということが分かった．

3.1.2　レプリケーションファクタ

図 3 に，計算機台数を 8 台，16 台，32 台，48 台，64 台と変化させた場合の各グラフ分割手法におけるレプリケーションファクタを示す．なお，表 1 の全てのグラフデータにおいて同様な傾向の結果が得られたため，本稿ではサイズの大きなグラフデータである soc-LiveJournal1，uk-2002，webbase-2001 についての結果のみ示す．図中のグラフ分割手法 cluster(4k) は，$a * k$ の値が 4000 であるような本手法のことを指しており，その他 cluster(*k) も同様の意味である．なお，uk-2002 および webbase-2001 において $a * k$ の値が小さい場合の結果が記されていないのは，データのサイズが大きいために，$a * k$ の値を小さくしすぎると Modularity クラスタリングステップの処理に膨大な時間を要してしまうからである．図 3 から，いずれのグラフデータにおいても，従来手法より本手法の方がレプリケーションファクタを小さくすることができていることが分かる．HDRF と比較すると，soc-LiveJournal1 では最大 12%，uk-2002 では最大 70%，webbase-2001 では最大 56% レプリケーションファクタを削減することができた．また，表 1 と合わせて，Modularity の値が大きいグラフデータほどレプリケーションファクタを削減できていることが分かる．

3.1.3　ロードバランスファクタ

図 4 に，計算機台数を 8 台，16 台，32 台，48 台，64 台と変化させた場合の各グラフ分割手法におけるロードバランスファクタを示す．図 4 から，従来手法はロードバランスファクタの値がほぼ 1 に近く最適な値となっているのに対し，本手法は従来手法よりもロードバランスファクタが大きくなっ

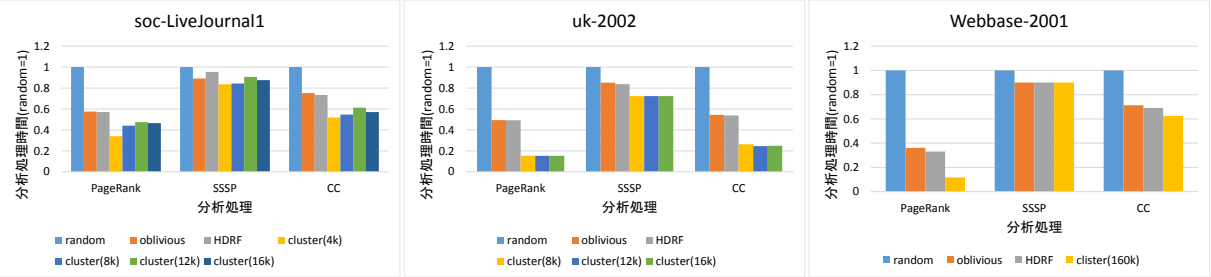


図 5　分析処理時間（random を 1 とした場合）

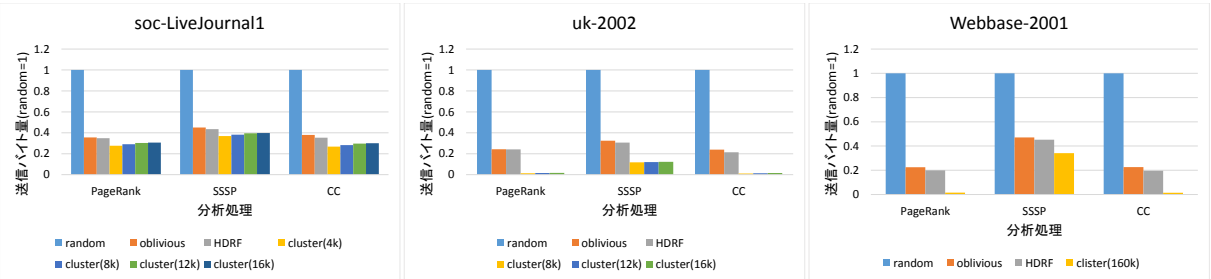


図 6　各計算機の平均送信バイト量（random を 1 とした場合）

ていることが分かる．表 1 のグラフデータでは，com-DBLP，amazon0505，web-Google，soc-Pokec，wiki-Talk に関しては本手法によって random よりもロードバランスファクタを抑えることができたが，それ以外のグラフデータに関しては，従来手法よりもロードバランスファクタが大きくなるという結果が得られた．HDRF と比較すると，soc-LiveJournal1 では最大 1.1%，uk-2002 では最大 3.3%，webbase-2001 では最大 13% ロードバランスファクタが増大した．

3.2　分析処理を行った場合の性能評価

3.1 節では，グラフ分割の精度を評価するための指標であるレプリケーションファクタとロードバランスファクタを用いて各グラフ分割手法の性能比較を行った．本節では，本手法を実装した PowerGraph を用いて実際に分析処理を行い，分析処理に要した時間や各計算機間の通信コスト，各計算機の実行時間の偏りなどを計測した．なお，計算機台数は 64 台固定とし，結果を載せるグラフデータは 3.1 節で扱ったものと同じとする．本実験では，以下の 3 つの分析処理を実行した．

1)　PageRank [14]：ウェブページの重要度を計算するために考えられた分析処理

2)　SSSP(single-source shortest path)：ある一つの頂点から他の全ての頂点への最短距離を求める分析処理

3)　CC(Connected Component)：グラフデータ内から頂点が連結された部分グラフを抽出する分析処理．

3.2.1　分析処理時間

各分析処理を実行した場合の分析処理時間を図 5 に示す．縦軸は random の分析処理時間を 1 とした場合の比となっている．図 5 から，いずれの分析処理においても本手法によって分析処理を高速化できていることが分かる．uk-2002 の場合が最も分析処理を高速化できており，HDRF と比較すると，PageRank では最大 3.2 倍，SSSP では最大 1.2 倍，CC では最大 2.2 倍分析処理を高速化できた．図 3 の結果と合わせて，従来手法に対するレプリケーションファクタ削減の効果が大きいグラフデータほど，分析処理を高速化できていることが分かる．また，分析手法ごとの高速化の効果の違いは，各分析手法における通信コストの大きさによって違うと考えられる．詳しくは 3.2.2 節で述べる．

3.2.2　送信バイト量

各分析処理における分析処理中の送信バイト量を図 6 に示す．縦軸は各計算機の送信バイト量の平均値であり，random を 1 としている．図 6 から，本手法は，従来手法を用いた場合に比べて送信バイト量を大幅に削減できていることが分かる．uk-2002 の場合が最も送信バイト量を削減できており，HDRF と比較すると，PageRank では最大 94%，SSSP では最大 62%，CC では最大 95% 送信バイト量を削減できた．図 3 とも合わせて，従来手法に対するレプリケーションファクタ削減の効果が大きいグラフデータほど送信バイト量を削減できていることが分かる．また，各分析処理の実際の送信バイト量には大きな差があり，グラフデータを uk-2002，グラフ分割手法を random とした場合，各計算機間の平均送信バイト量は，PageRank では 21GB，SSSP では 0.17GB，CC では 1.5GB であった．この結果と図 5 から，通信コストが高くなるような分析処理では，レプリケーションファクタを削減することによる高速化の効果が高くなると考えられる．

3.2.3　各計算機の分析処理時間の偏り

本実験では，各計算機の分析処理に要した時間を計測した．この実験では，同期処理を行う際に発生する待ち時間は含めず，実際の計算処理に要した時間のみを計測した．そして，各計算機の分析処理時間の偏りを，分析処理時間の最大値を平均値で割るというロードバランスファクタと同様の方法で評価した．その結果，本手法による分析処理時間の最大値と平均値の比は 1.2〜1.4 程度であった．また，ロードバランスファクタは最適値に近い従来手法においても，似たような結果が得られた．本

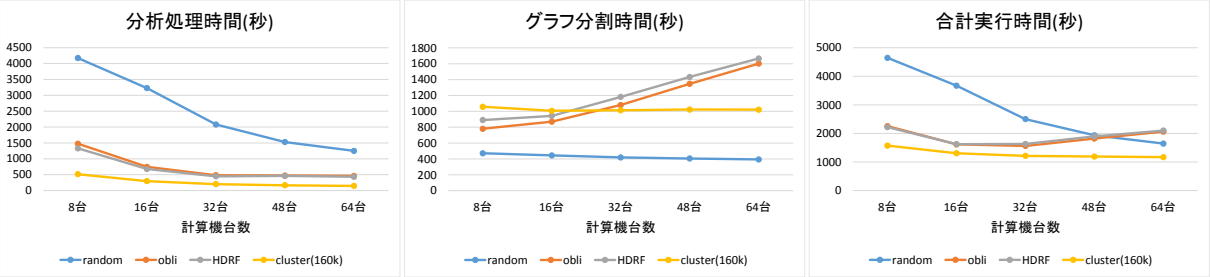


図 7　計算機台数を変化させた場合の PageRank の分析処理時間，グラフ分割時間および合計実行時間 (webbase-2001)

手法のロードバランスファクタは従来手法よりは大きいものの，1 に近い値となっているために，各計算機の分析処理時間の偏りを従来手法と同程度に抑えることができたと考えられる．また，実際に各計算機の分析処理時間を見てみると，いずれの分析手法およびグラフ分割手法の組み合わせにおいても，一部の計算機の分析処理時間が他の計算機に比べて長くなっていることがわかった．PowerGraph では頂点ごとに処理が割り当てられ，それらが並列に実行される．そして，紐付けられたデータへの変更を行う必要のない頂点に関しては処理を行わないことで効率化を図っている．分析処理時間が長くなる計算機は，他の頂点に比べてデータの収束が遅い頂点を有していると考えられる．

3.3　スケーラビリティの評価実験

本実験では，各グラフ分割手法において計算機台数を変化させた場合の分析処理時間，グラフ分割時間および合計実行時間を計測し比較を行った．ここで，合計実行時間とは，グラフデータの読み込みを始めてから分析処理が終了するまでに要した時間のことであり，グラフ分割時間は合計実行時間から分析処理時間を引いて求めた時間である．webbase-2001 を入力とし，PageRank を実行した場合の分析処理時間，グラフ分割時間および合計実行時間を図 7 に示す．図 7 から，本手法を用いた場合は計算機台数の増加に伴って分析処理が速くなっていることが分かる．また，全ての計算機台数において，本手法が従来手法より分析処理を高速化できていることが分かる．グラフ分割時間に関しては，oblivious と HDRF ではグラフ分割時間が計算機台数の増大に合わせてほぼ線形に増大しているのに対し，本手法はわずかにグラフ分割時間が短くなっていることが分かる．そのため，計算機台数が多い場合では，本手法のほうが従来のヒューリスティックな手法よりも高速にグラフ分割が可能であることが示された．そして，合計実行時間に関しては，全ての計算機台数において本手法により合計実行時間を短縮できている．これらの結果から，本手法は分析処理を高速化することが可能であり，かつグラフ分割も高速に行うことができるため，全体の処理を高速化できるということが示された．

4.　関 連 研 究

分散グラフ処理におけるグラフ分割方法は，大きく二種類に分けられる．一つ目は，頂点を各計算機に割り当ててエッジを切断する edge-cut 方式である．二つ目は，エッジを各計算機に割り当てて頂点を切断する vertex-cut 方式である．

edge-cut 方式によりグラフデータを分割する方法として METIS [10] を用いる方法が挙げられる．METIS はグラフを分割するための様々なアルゴリズムが実装されたソフトウェアパッケージである．METIS はグラフデータを精度よく分割することができるものの，データのサイズが大きくなると分割処理に膨大な時間がかかってしまうという欠点がある．vertex-cut 方式のグラフ分割手法として，oblivious [3] や HDRF [9] が挙げられる．これらの手法は，頂点やエッジのデータが読み込まれた時点で割り当て先となる計算機を逐次決定することにより，大規模なグラフデータを高速に分割することができる．oblivious は，過去のエッジの割り当て先を記録しておき，エッジの割り当てにより生成される頂点の複製の数が少なくなるようにエッジの割り当て先を決める．HDRF(High-Degree (are) Replicated First) は，次数の多い頂点を優先して切断することで，全体の頂点の複製数を削減する．これらの手法はグラフデータを高速に分割することができるものの，グラフ分割の精度があまり良くないという欠点がある．

5.　ま　と　め

本稿では，我々が先行研究により提案した，分散グラフ処理フレームワークにおける分析処理を高速化するグラフ分割手法についてのさらなる検証を行った．まず，本手法がどのような特徴を持ったグラフデータに対して効果的なのか否かについて分析するために，本手法においてグラフデータの持つ Modularity の値が通信コストにどのような影響を与えるのかについて検証した．その結果，グラフデータの持つ Modularity の値が大きくなるほど，通信コストを抑えることができることが分かった．また，分散グラフ処理フレームワークにおけるグラフ分割の精度を評価するための指標であるレプリケーションファクタ [3] とロードバランスファクタの 2 つを計測するとともに，PowerGraph で分析処理を行った場合の通信バイト量および各計算機の処理時間の偏りを計測し分析を行った．その結果，レプリケーションファクタを削減することで通信バイト量を削減することができ，分析処理を高速化できることを示した．今後の課題として，ロードバランスファクタと各計算機の負荷の偏りとの関係性をより詳しく分析する必要がある．また，本手法のロードバランスファクタのさらなる改善についても取り組む必要がある．

文　　献

[1] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser and G. Czajkowski, “Pregel: a system for large-scale graph processing,” Proceedings of SIGMOD, 2010.

[2] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola and J. M. Hellerstein, ”Distributed GraphLab: a framework for machine learning and data mining in the cloud,” PVLDB, 2012.

[3] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson and C. Guestrin, “PowerGraph: distributed graph-parallel computation on natural graphs,” Proceedings of OSDI, 2012.

[4] R. S. Xin, J. E. Gonzalez, M. J. Franklin and I. Stoica, “GraphX: a resilient distributed graph system on Spark,” Proceeding of GRADES, 2013.

[5] H. Shiokawa, Y. Fujiwara, and M. Onizuka, “Fast algorithm for modularity-based graph clustering,” Proceedings of the 27th AAAI Conference on Artificial Intelligence, 2013.

[6] M. E. J. Newman and M. Girvan. “Finding and evaluating community structure in networks,” Phys. Rev. E, 2004.

[7] J. Leskovec, K. J. Lang, A. Dasgupta and M. W. Mahoney, ”Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters,” Internet Mathematics 6, 2008.

[8] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. “Fast unfolding of communities in large networks,” Journal of Statistical Mechanics: Theory and Experiment, 2008.

[9] F. Petroni, Leonardo Querzoni, K. Daudjee, S. Kamali and G. Iacoboni. “HDRF: Stream-Based Partitioning for Power-Law Graphs,” Proceeding of CIKM, 2015.

[10] G. Karypis and V. Kumar, “A fast and high quality multilevel scheme for partitioning irregular graphs,” SIAM Journal on Scientific Computing.

[11] S. Suri and S. Vassilvitskii, “Counting triangles and the curse of the last reducer,” Proceedings of WWW, 2011.

[12] A. Clauset, M. E. J. Newman and C. Moore, “Finding community structure in very large networks,” Phys. Rev. E, 2004.

[13] K. Wakita and T. Tsurumi, “Finding community structure in mega-scale social networks,” Proceedings of WWW, 2007.

[14] L. Page, S. Brin, R. Motwani and T. Winograd. “The pagerank citation ranking: Bringing order to the web,” Technical Report, 1999.

[15] Stanford Large Network Dataset Collection. http://snap.stanford.edu/data/

[16] Laboratory for Web Algorithmics. http://law.di.unimi.it.

[17] 藤森俊匡, 塩川浩昭, 鬼塚真, “分散グラフ処理におけるグラフ分割の最適化,” DEIM, 2015.