DEIM Forum 2015 E6-3

# 動的グラフに対する密度ベースクラスタリング

塩川　浩昭† 　藤原　靖宏† 　飯田　恭弘† 　鬼塚　真††

† 日本電信電話株式会社 NTT ソフトウェアイノベーションセンタ　〒 180–8585 東京都武蔵野市緑町 3-9-11
†† 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 2-1
E-mail: †{shiokawa.hiroaki,fujiwara.yasuhiro,iida.yasuhiro}@lab.ntt.co.jp, ††onizuka@ist.osaka-u.ac.jp

**あらまし**　グラフはエンティティ同士の関連性を表現する最も基本的なデータ構造であり，我々の身の回りに数多く存在している．特に近年では，短期間に構造を大きく変化させる動的グラフが登場しており，動的グラフに対する高速かつ高精度なクラスタ分析は重要な要素技術となっている．本稿では，動的に構造が変化するグラフに対する密度ベースのクラスタリング手法を提案する．提案手法は，(1) ある時刻 $t$ に入力されたグラフ構造から，グラフ構造に含まれるクラスタ構造を要約した cluster skeleton を抽出し，(2) 時刻 $t+1$ におけるグラフ構造の変化に併せて，事前に構築した cluster skeleton を逐次更新する．その後，(3) cluster skeleton から時刻 $t+1$ におけるクラスタ構造を復元する．以上の手続きに基づく提案手法を用いることで，動的なグラフからパラメータの設定を介さず高速かつ高精度にクラスタを抽出することが可能になる．本稿では提案手法の詳細と実験結果について述べる．

**キーワード**　グラフ，クラスタリング

## 1. はじめに

グラフ構造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．とりわけ近年では，時間経過とともに構造を大きく変化させる動的グラフが数多く登場しており，動的グラフに対する高速かつ高精度なデータ分析技術は大規模なデータに隠れた構造や特性，相互作用を理解する上で重要な要素技術となっている．例えば，マイクロブログサービス Twitter では 2015 年現在で 1 日あたり約 5 億件の投稿があることが報告されている (注1)．これは 1 秒間あたりに約 6,000 件の投稿がされていることを示しており，時間経過とともに Twitter のもつグラフ構造が極めて大きく変化していることを確認できる．このような動的グラフは Twitter のみに限定されず，例えば道路ネットワークや通信ネットワーク，文書の引用関係，ソーシャルネットワーク，マイクロアレイデータなど様々存在しており，動的グラフに対する高速かつ高精度な分析技術への要求が大きなものであることが認識できる．

大規模なグラフ構造の分析技術のひとつとして，グラフクラスタリングが挙げられる．グラフ構造には，クラスタと呼ばれる相互に密な接続を有する部分ノード集合が存在する．例えば，Web グラフではトピックや関心の近いページ集合が互いにリンクすることで，トピックや関心の類似したページ群がコミュニティを形成する傾向にある．グラフ構造中のクラスタは互いに共通した性質を持つことから，グラフ構造の理解や様々なアプリケーションに利用され，非常に重要な要素技術となっている．そのため，これまで Modularity に基づく手法 [1], [2] や最小カットに基づく手法 [3], [4] など様々なグラフクラスタリング手法の研究が行われてきた．

(注1)：https://about.twitter.com/company

その中でも構造的類似度を用いた密度ベースのクラスタリング手法 [5], [6] は特に注目を集めているクラスタリング手法である．これらの手法はグラフ構造からクラスタを抽出するのみでなく，グラフ構造中において複数のクラスタに接続するようなハブとなるノードや，クラスタに接続したノイズとして扱われる外れ値となるノードを合わせて抽出することが可能である．このことから，クラスタのみを抽出する従来手法に対して，密度ベースクラスタリング手法はより細やかな分析を可能にする手法として期待されている．しかしながら，既存の密度ベースクラスタリング手法の多くはグラフのもつ構造が変化しない静的なグラフを前提としており，本稿で述べた動的なグラフのクラスタリングは対象とはしていない．

この背景から，動的グラフを対象としたクラスタリング手法 [7], [8], [9], [10] がこれまで提案されてきたが，これらの既存手法は依然として下記に示す課題を有する．

- 高速性: 動的グラフは単位時間あたりにその構造を大きく変化させる．ところが既存の密度ベースクラスタリングは，グラフに含まれる全てのノードとエッジに対して計算を行う必要があり，クラスタリングに要する時間的コストが大きい．
- 高精度性: 大規模な動的グラフにおいて，高速なクラスタリングを実現するために計算対象をサンプリングする手法などが考えられる．ところが，文献 [11] で報告されている通り，サンプリングによる手法は出力結果の精度を劣化させる．
- パラメータ設定: 文献 [5], [6] で報告されている通り，パラメータ設定は出力結果の精度に大きく影響を与える．また，動的グラフではグラフの構造が大きく変化していくため，時間経過とともに最適なパラメータ設定が異なってくる．

理想的には，逐次的に変化するグラフ構造に追従するために，動的グラフに対するクラスタリング手法は高速かつ高精度であることが望ましい．さらに，大規模なグラフ構造に対するパラメータチューニングは多くの時間的コストを要することから，

ユーザによるパラメータ設定を介さない手法が望ましい．

### 1.1 本研究の貢献

本稿では動的グラフに対する，パラメータ設定を必要としない高速かつ高精度な密度ベースクラスタリング手法を提案する．より具体的には，(1) ある時刻 $t$ に入力されたグラフ構造から，グラフ構造に含まれるクラスタ構造を要約した cluster skeleton を抽出し，(2) 時刻 $t+1$ におけるグラフ構造の変化に併せて，事前に構築した cluster skeleton を逐次更新する．その後，(3) cluster skeleton から時刻 $t+1$ におけるクラスタ構造を復元する．さらに重要な点としてこれらの処理は，(4) ユーザによるパラメータ設定を必要とせず，既存の密度ベースクラスタリング手法に対して高速かつ精度を劣化させずに行う．その結果として，提案手法は以下の特性を有する．

- 高速性: 提案手法は動的グラフに対する既存の密度ベースクラスタリング手法に対して高速である．
- 高精度性: 提案手法は動的グラフに対する既存の密度ベースクラスタリング手法に対して高精度である．
- パラメータフリー: 提案手法はユーザによるパラメータ設定を必要としない．

本稿で提案する手法は我々の知る限り，動的グラフに対する密度ベースクラスタリングにおいて，高速性，高精度性およびパラメータフリーの全てを同時に満たす最初の手法である．既存の密度ベースクラスタリング手法は高い計算コストを有するものの，クラスタだけでなくハブや外れ値を抽出できることから既に幅広いアプリケーションで利用されている．本稿で提案するグラフクラスタリング手法は，既に従来技術が利用されているアプリケーションや将来的に利用が予測される分野において，その処理性能の向上に貢献する．

本稿の構成は，次のとおりである．2. 節において，本研究で扱う問題を定義する．4. 節において提案手法の概要を述べ，5. 節において提案手法の評価と分析を行う．6. 節にて，本研究に対する関連研究について述べる．そして 7. 節において，本稿をまとめ今後の課題について論ずる．

## 2. 問題定義

ここでは本論文で対象とする問題を定義する．

動的グラフとは時刻経過とともにノードとエッジが更新されるグラフ構造である．本稿では，$\mathbb{G}_t = \{\mathbb{V}_t, \mathbb{E}_t\}$ を時刻 $t$ における動的グラフのスナップショットとし，$\mathbb{V}_t$ および $\mathbb{E}_t$ はそれぞれ時刻 $t$ における $\mathbb{G}_t$ のもつノード集合およびエッジ集合とする．このとき，動的グラフを下記のように定義する．

［**定義** 1］（動的グラフ $\mathcal{G}_{ij}$） 時刻 $i$ から時刻 $j$ までの動的グラフを $\mathcal{G}_{ij} = (\mathbb{G}_i; \triangle\mathbb{G}_{i+1}, \ldots, \triangle\mathbb{G}_j)$ とする．ただし，時刻 $(i \leqq t < j)$ における $\triangle\mathbb{G}_{t+1} = \{\triangle\mathbb{V}_{t+1}, \triangle\mathbb{E}_{t+1}\}$ は時刻 $t+1$ で更新される部分グラフである．$\triangle\mathbb{V}_{t+1}$ と $\triangle\mathbb{E}_{t+1}$ はそれぞれ時刻 $t+1$ において追加/削除されたノードおよびエッジ集合を表す．

本稿では説明の簡単化のため動的グラフ $\mathcal{G}$，および時刻 $t$ におけるグラフ $\mathbb{G}_t$ は無向重みなしグラフとする．

定義 3 で与えた動的グラフに対して，動的クラスタを以下に定義する．

［**定義** 2］（動的クラスタ） 時刻 $t$ におけるグラフ $\mathbb{G}_t$ に含まれるクラスタ集合を $\mathbb{C}_t$ とするとき，動的グラフ $\mathcal{G}_{ij} = (\mathbb{G}_i; \triangle\mathbb{G}_{i+1}, \ldots, \triangle\mathbb{G}_j)$ に対する動的クラスタを $\mathcal{C}_{ij} = (\mathbb{C}_i; \triangle\mathbb{C}_{i+1}, \ldots, \triangle\mathbb{C}_j)$ とする．ただし，時刻 $(i \leqq t < j)$ において $\mathbb{C}_t + \triangle\mathbb{C}_{t+1} = \mathbb{C}_{t+1}$ である．

定義 3,4 より，本稿で取り組む問題は，動的グラフ $\mathcal{G}_{ij}$ が与えられた時に，動的クラスタ $\mathcal{C}_{ij}$ を獲得することである．本稿ではユーザによるパラメータ設定を介さず，精度の高いクラスタリング結果を求めるための手法を提案する．

## 3. クラスタモデル

本節では本稿で求める密度ベースクラスタについて定義する．本稿で求める密度ベースクラスタは，静的なグラフ構造に対する密度ベースクラスタリング手法 SCAN [5] で求めるクラスタである．従来手法では 2 つのノード間で共有される隣接ノード集合の割合を評価することで構造類似度を計算していく．ここで用いる隣接ノード集合は以下のように定義される．

［**定義** 3］（構造的隣接ノード集合） $u \in \mathbb{V}$ とするとき，隣接ノード集合はノード $u$ にエッジで接続するノードとノード $u$ 自身から構成される集合 $\mathbb{N}[u]$ で与えられる．

$$\mathbb{N}[u] = \{v \in \mathbb{V} : \{u, v\} \in \mathbb{E}\} \cup \{u\}. \tag{1}$$

また先に述べたように，クラスタリングで用いられる 2 ノード間の構造的類似度は定義 3 に基づき以下のように定義される．

［**定義** 4］（構造的類似度） $u, v \in V$，$|\mathbb{N}[u]|$ を隣接ノード集合に含まれるノード数とするとき，ノード $u$，$v$ 間の構造的類似度は $\sigma(u, v)$ となる．

$$\sigma(u, v) = \frac{|\mathbb{N}[u] \cap \mathbb{N}[v]|}{\sqrt{|\mathbb{N}[u]||\mathbb{N}[v]|}}. \tag{2}$$

ノード $u$，$v$ 間の $\sigma(u, v)$ は共有されるノードがない場合 $\sigma(u, v) = 0$，互いに全て共有する場合には $\sigma(u, v) = 1$ となる．

構造的類似度に基づくクラスタリング手法はクラスタを構成するための類似度の閾値として $\epsilon$ を導入し，類似度 $\epsilon$ 以上で接続する隣接ノード集合 $\epsilon$-neighborhood を定義する．

［**定義** 5］（$\epsilon$-neighborhood） $u \in \mathbb{V}$，$\epsilon \in \mathbb{R}$ とするとき，$\epsilon$-neighborhood $\mathbb{N}_\epsilon[u]$ は以下のように定義される．

$$\mathbb{N}_\epsilon[u] = \{v \in \mathbb{N}[u] : \sigma(u, v) \geqq \epsilon\}. \tag{3}$$

ここで，クラスタを構成する最小ノード数としてパラメータ $\mu$ を導入し，特別なノードのクラスとして *core* を定義する．

［**定義** 6］（Core） $u \in \mathbb{V}$，$|\mathbb{N}_\epsilon[u]|$ をノード $u$ における $\epsilon$-neighborhood のノード数とするとき，*core* は以下のように定義される．

$$K_{\epsilon,\mu}(u) \Leftrightarrow |\mathbb{N}_\epsilon[u]| \geqq \mu. \tag{4}$$

従来手法では定義 6 を満たす core ノード $u$ をクラスタの中心として，$\mathbb{N}_\epsilon[u]$ をクラスタのメンバとして同一のクラスタに所属させる．この手順により，パラメータ $\epsilon$，$\mu$ を用いてクラ

スタの形を決定し，$\mu$ を用いてクラスタの最小サイズを定める．この考えは direct structure reachability として以下に定義される．

［**定義** 7］（Direct structure reachability） $u, v \in \mathbb{V}$ とするとき，ノード $u$ とノード $v$ における direct structure reachability $u \mapsto_{\epsilon,\mu} v$ は以下のようになる．

$$u \mapsto_{\epsilon,\mu} v \Leftrightarrow K_{\epsilon,\mu}(u) \wedge v \in \mathbb{N}_{\epsilon}[u]. \tag{5}$$

定義 7 はノード $K_{\epsilon,\mu}(u) \wedge K_{\epsilon,\mu}(v)$ である場合対称であるが，どちらか一方が core でない場合には非対称となる．ゆえに，$u \mapsto_{\epsilon,\mu} v$ かつノード $v$ が core でない場合，ノード $v$ は $K_{\epsilon,\mu}(u)$ が構築するクラスタの境界に面したノードとなる．本稿ではこのようなノード $v$ を *border* と呼ぶ．定義 7 をより一般的な形に拡張した structure reachability を示す．

［**定義** 8］（Structure reachability） $u, v \in \mathbb{V}$ とすると，ノード $u$ とノード $v$ における structure reachability$u \rightarrow_{\epsilon,\mu} v$ は以下に定義される．

$$u \rightarrow_{\epsilon,\mu} v \Leftrightarrow \tag{6}$$

$$\exists u_1, \ldots, u_n \in V : u_1 = u \wedge u_n = v \wedge u_i \mapsto_{\epsilon,\mu} u_{i+1}. \tag{7}$$

ここで定義された structure reachability は推移律を満たし，非対称である．この structure reachability は direct structure reachability の推移閉包であり，定義 7 の対称性から，推移閉包の構成要素である $u_1, \ldots, u_{n-1}$ は core ノードである必要がある．言い換えると，core 同士が direct structure reachability を示す時，それらの $\epsilon$-neighbor は同一クラスタに属することになる．この考えは structure connected クラスタとして以下に定義される．

［**定義** 9］（Structure-Connected クラスタ） ノード $u \in \mathbb{V}$ とすると，$K_{\epsilon,\mu}(u)$ から求まるクラスタ $\mathbb{C}^u \in C$ が structure-connected クラスタである必要十分条件は (1) $u \in \mathbb{C}^u$; (2) $\forall v \in \mathbb{V}$, $u \rightarrow_{\epsilon,\mu} v \Leftrightarrow v \in \mathbb{C}^u$.

この定義より，structure-connected クラスタはその中に含まれる core により一意に決めることができることがわかる．従来手法では structure-connected クラスタの取得のために全てのエッジを走査するため，クラスタリングの計算量が $O(|V|^2)$ となる．ここで，border ノードは，複数のクラスタの core から direct structure reachability となる可能性があることに注意されたい．この場合，border ノードは複数のクラスタに属する．

パラメータ $\epsilon, \mu$ に対するクラスタリング結果が $\mathbb{C}$ で与えられる時，ノード集合 $\mathbb{V}$ にはいずれのクラスタ集合 $\mathbb{C}$ にも属さないノードが存在する場合がある．これらのノードはハブもしくは外れ値に分類される．

［**定義** 10］（ハブ） クラスタ集合 $\mathbb{C}$，ハブ集合 $\mathbb{H}$ とすると，ノード $h \in \mathbb{V}$ がハブである (*e.g.* $h \in \mathbb{H}$) 必要十分条件は (1) $h \notin \forall \mathbb{C}^i \in \mathbb{C}$, (2) $u, v \in \mathbb{N}[h]$ $s.t$ $u \in \mathbb{C}^i, v \in \mathbb{C}^j, i \neq j$.

［**定義** 11］（外れ値） クラスタ集合 $\mathbb{C}$，ハブ集合 $\mathbb{H}$，外れ値集合 $\mathbb{O}$ とすると，ノード $o \in \mathbb{V}$ が外れ値 (*e.g.* $o \in \mathbb{O}$) である必要十分条件は $o \notin \mathbb{C} \wedge o \notin \mathbb{H}$.

## 4. 動的グラフに対するクラスタリング

本節では提案手法について述べる．提案手法は，ユーザによる パラメータ設定を必要とせず, 既存の密度ベースクラスタリン グ手法に対して高速かつ精度を劣化させず．まず最初に提案手法を構成する基本アイデアについて述べた後，その詳細を 4.2 節，4.3 節，4.4 節にて述べる．

### 4.1 基本アイデア

提案手法は 2 つのアイデアに基づき動的グラフに対するクラスタリングの高速化と高精度化を行う．具体的には，従来手法で全てのノードとエッジに対して密度の回数を削減するために，cluster skeleton の構築によるクラスタ構造の情報要約手法と，cluster skeleton の差分更新手法を提案する．

a） Cluster Skeleton の構築

従来のグラフに対する密度ベースクラスタリング手法はグラフ構造に含まれる全てのエッジ $\mathbb{E}$ に対して計算を行う．この全網羅的な類似度計算は，動的グラフが構造を変化させる毎に高い計算コストを要求し，クラスタリングに膨大な時間を必要とする．そこで本稿では，従来手法とは異なり，前の時刻のクラスタリング結果を要約する Cluster Skeleton を構築し，Cluster Skeleton に対してのみ類似度計算を行うことで動的グラフに対するクラスタリングの計算コストを削減する．具体的には，前の時刻のクラスタリング結果から獲得されるグラフ構造の構造的類似度に基づく最大全域木を Cluster Skeleton として用いる．

b） Cluster Skeleton の差分更新

Cluster Skeleton に基づき動的グラフに対応するための差分更新手法を提案する．本稿で提案する差分更新手法では，既存研究である Lee らによる知見 [12] に基づき，密度ベースクラスタリングにおける差分更新ノード集合を獲得する．その後，獲得したノード集合に対してのみ，structural similarity の計算と全域木の再計算を行い Cluster Skeleton を更新する．

### 4.2 Cluster Skeleton の構築

ある時刻 $t$ におけるグラフ構造 $\mathbb{G}_t$ に含まれるクラスタ構造を要約するために Cluster Skeleton を構築する．本節では Cluster Skeleton として，Sun らによって提案された Core-Connected Maximal Spanning Tree (CCMST) [6] を用いる．CCMST とは互いに同一のクラスタとなる可能性の高い core 同士を接続した Maximal Spanning Tree である．CCMST を獲得するためには，まず Structure Core-Similarity とよばれる指標を各ノードに対して計算する必要がある．

［**定義** 12］（Structure Core-Similarity） ノード $u \in \mathbb{V}_t$ の Structure Core-Similarity $CS(u)$ を以下に定義する．

$$CS(u) = \begin{cases} \max\{\epsilon \in \mathbb{R} : |\mathbb{N}_{\epsilon}[u]| \geqq \mu\} & |\mathbb{N}[u]| \geqq \mu \\ 0 & otherwise \end{cases} \tag{8}$$

定義 12 に示した様に，Structure Core-Similarity とは core となるノードが持つ最大の $\epsilon$ の値を示している．この値を基に隣接する core 同士が同一クラスタになりやすさを定義 13 に示した Reachability-Similarity を用いて，定義 14 に示した 2 つ

の core 間の Core-Connectivity Similarity を計算することで求める．

［**定義** 13］（Reachability-Similarity） ノード $u, v \in \mathbb{G}_t$ とするとき，ノード $u$ に対するノード $v$ の Reachability-Similarity $RS(u, v)$ を以下に定義する．

$$RS(u, v) = \min\{CS(u), \sigma(u, v)\}. \tag{9}$$

［**定義** 14］（Core-Connectivity Similarity） ノード $u, v \in \mathbb{G}_t$ かつ $(u, v) \in \mathbb{E}_t$ とするとき，ノード $u, v$ 間の Core-Connectivity Similarity $CCS(u, v)$ は以下のように定義する．

$$\begin{aligned} CCS(u, v) &= \min\{RS(u, v), RS(v, u)\} \\ &= \min\{CS(u), CS(v), \sigma(u, v)\}. \end{aligned} \tag{10}$$

定義 14 に示した $CCS$ に基いて，提案手法で構築する CCMST は以下のように与えられる．

［**定義** 15］（CCMST） 時刻 $t$ のグラフ構造 $\mathbb{G}_t$ に含まれる全てのエッジ $e$ に対して，CCS の値を $\omega(e)$ で与えられるとするとき，CCS により重み付けされたグラフ構造を $\mathbb{G}'_t = (\mathbb{V}_t, \mathbb{E}_t, \omega)$ とする．このとき，CCMST は $\mathbb{G}'_t$ から獲得できる Maximum Spaning Tree である．

定義 15 で与えられる CCMST は以下の性質を持つことが文献 [6] において知られている．

［**補題** 1］ $\mathbb{G}'_t$ に対する CCMST において，$\omega(e) > \hat{\epsilon}$ で接続するノードは，$\mathbb{G}_t = (\mathbb{V}_t, \mathbb{E}_t)$ で与えられる時刻 $t$ のグラフに対して，$\hat{\epsilon} \geqq \epsilon$ となるパラメータ $\epsilon$ で SCAN [5] を実行した際に必ず同じクラスタに含まれる．

補題 1 の証明については，文献 [6] を参照されたい．本稿では，$\mathbb{G}'$ から CCMST を構築するために Prim 法 [13] を用いる．

提案手法では CCMST を Cluster Skeleton として構築し，CCMST を差分更新していくことにより，動的グラフに対するパラメータを必要としない密度ベースクラスタリングを実現する．CCMST の差分更新方式の詳細は 4.3 節，クラスタ抽出手順の詳細は 4.4 節で述べる．

### 4.3 Cluster Skeleton の差分更新

前節でも述べたとおり，提案手法は CCMST を更新していくことで動的グラフに対する密度ベースクラスタリングを実現する．本節では CCMST の更新について述べる．

時刻が $t$ から $t+1$ に推移しグラフ構造が変化した際に，時刻 $t+1$ における CCMST を構築するためには，構造的類似度を計算しそこから定義 14 で与えた Core-Connectivity Similarity を求める必要がある．時刻が $t$ から $t+1$ に推移した際のグラフ構造の変化は大きく分けて，(1) エッジの追加と (2) エッジの削除の 2 つに分類できる．そこで本稿では，時刻 $t+1$ において $(u, v) \in \mathbb{E}_{t+1}$ となるエッジが追加される場合および $(u, v) \in \mathbb{E}_t$ が削除される場合について構造的類似度の差分更新方式を与える．

#### 4.3.1 時刻 $t+1$ で $(u, v) \in \mathbb{E}_{t+1}$ が追加される場合

時刻 $t+1$ においてエッジ $(u, v) \in \mathbb{E}_{t+1}$ が追加される場合について考える．時刻変化とともにエッジが追加される場合，エッジ $(u, v)$ に対する構造的類似度計算のみならず，エッジの短点であるノード $u$ および $v$ に接続するエッジ $(u, v)$ 以外のエッジが持つ構造的類似度が更新されることになる．そこで本稿では，時刻 $t+1$ におけるエッジ $(u, w) s.t. w \in \mathbb{N}[u]$ および，エッジ $(v, w) s.t. w \in \mathbb{N}[v]$ に対する，構造的類似度の差分更新方式を定義 16 に与える．

［**定義** 16］（エッジ追加による差分更新） エッジ $(u, v)$ を時刻 $t+1$ で追加されたエッジとし，ノード $w$ を $w \in \mathbb{N}[u]$ とするとき，時刻 $t+1$ におけるエッジ $(u, w)$ の構造的類似度 $\sigma_{t+1}(u, v)$ は以下のように求める．

$$\sigma_{t+1}(u, w) = \begin{cases} \frac{\sigma_t(u,w)\sqrt{|\mathbb{N}[u]||\mathbb{N}[w]|}+1}{\sqrt{(|\mathbb{N}[u]|+1)|\mathbb{N}[w]|}} & (w \in \mathbb{N}[u] \cap \mathbb{N}[v]) \\ \frac{\sigma_t(u,w)\sqrt{|\mathbb{N}[u]||\mathbb{N}[w]|}}{\sqrt{(|\mathbb{N}[u]|+1)|\mathbb{N}[w]|}} & (Otherwise) \end{cases} \tag{11}$$

ただし，$\sigma_t(u, w)$ は時刻 $t$ におけるエッジ $(u, w)$ の構造的類似度である．

定義 16 において計算が必要となる $\mathbb{N}[u] \cap \mathbb{N}[v]$ は $\sigma_{t+1}(u, v)$ の計算結果から獲得することができる．また，$|\mathbb{N}[u]|$ および $|\mathbb{N}[v]|$ は時刻 $t$ における構造的類似度の計算結果から獲得可能である．したがって，エッジの追加に伴う構造的類似度の差分更新を少ない計算コストで実行することができる．

#### 4.3.2 時刻 $t+1$ で $(u, v) \in \mathbb{E}_t$ が削除される場合

本節ではエッジが削除される場合について述べる．前節と同様に，エッジ $(u, v)$ が削除されることにより，その短点であるノード $u$，$v$ に接続するエッジに関して構造的類似度の更新が必要となる．そこで本稿では，時刻 $t+1$ におけるエッジ $(u, w) s.t. w \in \mathbb{N}[u]$ および，エッジ $(v, w) s.t. w \in \mathbb{N}[v]$ に対する，構造的類似度の差分更新方式を定義 17 に与える．

［**定義** 17］（エッジ削除による差分更新） エッジ $(u, v)$ を時刻 $t+1$ で削除されたエッジとし，ノード $w$ を $w \in \mathbb{N}[u]$ とするとき，時刻 $t+1$ におけるエッジ $(u, w)$ の構造的類似度 $\sigma_{t+1}(u, v)$ は以下のように求める．

$$\sigma_{t+1}(u, w) = \begin{cases} \frac{\sigma_t(u,w)\sqrt{|\mathbb{N}[u]||\mathbb{N}[w]|}-1}{\sqrt{(|\mathbb{N}[u]|-1)|\mathbb{N}[w]|}} & (w \in \mathbb{N}[u] \cap \mathbb{N}[v]) \\ \frac{\sigma_t(u,w)\sqrt{|\mathbb{N}[u]||\mathbb{N}[w]|}}{\sqrt{(|\mathbb{N}[u]|-1)|\mathbb{N}[w]|}} & (Otherwise) \end{cases} \tag{12}$$

ただし，$\sigma_t(u, w)$ は時刻 $t$ におけるエッジ $(u, w)$ の構造的類似度である．

前節と同様に，定義 17 において計算が必要となる $\mathbb{N}[u] \cap \mathbb{N}[v]$ は $\sigma_{t+1}(u, v)$ の計算結果から獲得することができる．また，$|\mathbb{N}[u]|$ および $|\mathbb{N}[v]|$ は時刻 $t$ における構造的類似度の計算結果から獲得可能である．したがって，エッジの追加に伴う構造的類似度の差分更新を少ない計算コストで実行することができる．

提案手法では，定義 16 および定義 17 を用いて，時刻 $t$ における構造的類似度計算結果から時刻 $t+1$ における構造的類似度計算結果を差分更新により求める．その後，差分更新の結果に対して改めて Prim 法により時刻 $t+1$ の CCMST を構築する．

### 4.4 Cluster Skeleton からのクラスタ抽出

本節では，これまで述べた CCMST からクラスタ抽出を行う手法について述べる．クラスタを抽出するに際して重要とな

るのが，適切なクラスタリング結果を与えるパラメータ $\epsilon$ の自動決定である．本節では，文献 [6], [14] に基づく $\epsilon$ の決定方法とクラスタ抽出手法について述べる．

まず提案手法では，4.2 節で与えた CCMST から，CCMST のエッジに付与されている全ての重み ($\epsilon$) を列挙する．その後列挙した値をソートし，ひとつずつ $\epsilon$ の候補を選択する．選択した $\epsilon$ の候補に対して，まず補題 1 に基づくクラスタの抽出を行う．その後，クラスタに含まれなかったノードに関して以下に定義される Attachability-Similarity を求める．

［**定義** 18］（Attachability-Similarity） ノード $v \in \mathbb{V}_t$ としたとき，ノード $v$ の Attachability-Similarity $AS(v)$ は以下のように定義される．

$$AS(v) = \max\{RS(u,v) : u \in \mathbb{N}[v] \backslash \{v\}\}. \tag{13}$$

すなわち，Attachability-Similarity とは各ノードとその隣接ノードに対する最大の Reachability-Similarity を示したものである．Attachability-Similarity を計算後，クラスタに含まれないノードを Attachability-Similarity で与えられる隣接ノードと同一のクラスタに割り当てる．以上の処理により，選択した $\epsilon$ 候補に対するクラスタリングを終了する．

提案手法では，選択した $\epsilon$ 候補から獲得したクラスタリング結果に対して，以下に示した構造的類似度に基づく modularity の値を計算する．

［**定義** 19］（構造的類似度に基づく modularity） 時刻 $t$ のクラスタを $\mathbb{C}_t = \{\mathbb{C}_t^1, \ldots, \mathbb{C}_t^{|\mathbb{C}_t|}\}$ とし，$|\mathbb{C}_t|$ を時刻 $t$ におけるクラスタ数とするとき，構造的類似度に基づく modularity$Q_s$ は以下に定義される．

$$Q_s = \sum_{i=1}^{|\mathbb{C}_t|} \left\{ \frac{IS_i}{TS} - \left( \frac{DS_i}{TS} \right)^2 \right\}. \tag{14}$$

ただし，$IS_i = \sum_{u,v \in \mathbb{C}_t^i} \sigma(u,v)$，$DS_i = \sum_{u \in \mathbb{C}_t^i, v \in \mathbb{V}_t} \sigma(u,v)$，$TS = \sum_{u,v \in \mathbb{V}_t} \sigma(u,v)$ である．

その後，選択した $\epsilon$ 候補に対する $Q_s$ を保存し，次の $\epsilon$ 候補を選択する．上記の処理を $Q_s$ を最大化する $\epsilon$ 候補が決定するまで実行し，クラスタリング精度を最も向上させる $\epsilon$ を自動決定する．以上により，提案手法はパラメータフリーな動的グラフに対する密度ベースクラスタリングを実現する．

## 5. 評 価 実 験

提案手法の有効性を評価するために，我々の提案手法および Kim らによる手法 (Kim-Han09) [8] および Lee らによる手法 (eTrack) [12] に対し，処理の高速性とクラスタリング精度の観点から比較実験を行う予定である．本評価で比較に用いる Kim-Han09 [8] は Kim らにより 2009 年に提案された動的グラフに対する密度ベースのクラスタリング手法である．Kim-Han09 はクラスタリング結果の時間変化に対する類似度と各時刻におけるクラスタリングの精度の両方を最適化する手法である．また，提案手法と同様にパラメータ $\epsilon$ をヒューリスティックスを用いて最適化する手法も導入している．eTrack は Lee らにより 2014 年に提案された動的グラフに対する密度ベースクラスタリング手法である [12]．eTrack では，動的グラフの構造変化の影響を受けるノード集合およびエッジ集合を定義しており，そのノード集合とエッジ集合に対してのみ SCAN に基づくクラスタリングを実行し直すインクリメンタル型のクラスタリング手法である．なお，提案手法で導入しているパラメータ $\epsilon$ の最適化は eTrack では行わず，一度設定した *epsilon* の値を使い続ける手法である．これらの手法の詳細については 6. 節で述べる．

本稿では，LFR benchmark [15] を用いてノード数 10 万，エッジ数 200 万のグラフを生成した．上記のグラフを動的グラフの初期状態（初期グラフ）とし，4 パターンの動的グラフを作成し評価を行った．データセットの詳細は以下のとおりである．

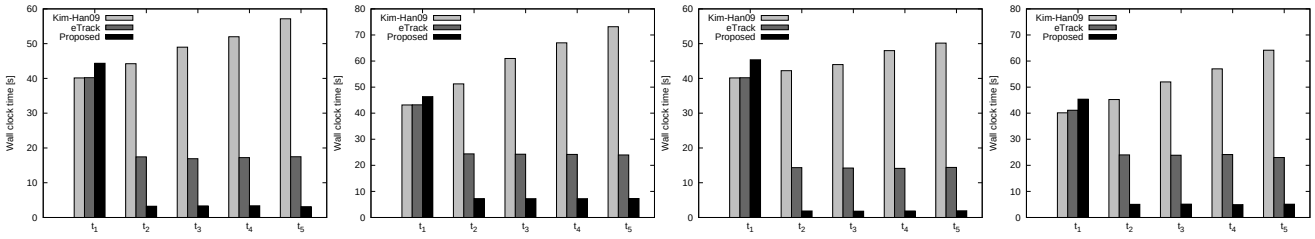
- **Graph(1%, Inner):** 初期グラフに存在するノードについてのみ，初期グラフの持つエッジ数の 1%に相当する 2 万エッジを各時刻毎に追加した動的グラフ．ただし，エッジを追加するノード対はランダムに選択するものとする．
- **Graph(5%, Inner):** 初期グラフに存在するノードについてのみ，初期グラフの持つエッジ数の 5%に相当する 10 万エッジを各時刻毎に追加した動的グラフ．ただし，エッジを追加するノード対はランダムに選択するものとする．
- **Graph(1%, Outer):** 初期グラフの持つエッジ数の 1%に相当する 2 万エッジを各時刻毎に追加した動的グラフ．ただし，エッジの追加に伴い，初期グラフに含まれないノードに接続するエッジも追加されるものとする．また，エッジを追加するノードはランダムに選択するものとする．
- **Graph(5%, Outer):** 初期グラフの持つエッジ数の 5%に相当する 10 万エッジを各時刻毎に追加した動的グラフ．ただし，エッジの追加に伴い，初期グラフに含まれないノードに接続するエッジも追加されるものとする．また，エッジを追加するノードはランダムに選択するものとする．

本実験では，これらのデータセットに対して，各タイムスタンプに対する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および $Q_s$ を評価した．本実験には CPU が Intel Core i7 1.8GHz，メモリが 4GB の計算機を利用した．また，提案手法および Kim-Han09，eTrackSCAN は gcc–g++_4.1.2 を用いて実装した．

### 5.1 実行時間の比較

本節では提案手法の実行速度を評価する．本実験では全ての評価において $\mu = 2$ を用いた．また，eTrack ではパラメータ $\epsilon$ の値として，初期グラフに対して提案手法が自動検出した $\epsilon$ の値をそれぞれのデータセットで用いるものとした．

本研究で用いた 4 種類の動的グラフに対する実行速度の評価結果を図 1 に示す．図 1 に示したように，いずれのデータセットに対しても提案手法は従来手法と比較して高速に処理できている．具体的には，Kim-Han09 に対しては約 20 倍，eTrack に対しては約 6 倍の高速化に成功している．特に，Kim-Han09 は $t_5$ に近づくにつれて処理時間が増加しているのに対し，提案手法はほぼ一定の処理時間でクラスタリングができていることが確認できる．この理由は，提案手法の差分更新方式にあると考えられる．提案手法は追加されたエッジとそのエッジに類




(a) Graph(1%, Inner) (b) Graph(5%, Inner) (c) Graph(1%, Outer) (d) Graph(5%, Outer)

図 1 実行速度の比較

似度が影響を受ける周辺エッジについてのみ計算を行う．ゆえに，提案手法は構造が変化した部分に対してのみ計算を行えばクラスタリング結果を獲得することができる．これに対して，Kim-Han09 は各 $t_1, \ldots, t_5$ においてグラフ構造全てを再計算する必要がある．したがって，時刻変化に伴い計算時間が増加している．また，同様に提案手法は eTrack と比較しても高速であることが確認できる．eTrack は提案手法と同様にグラフクラスタの差分更新手法であるが，差分更新を行う際に，追加・削除されたエッジや追加削除に伴い類似度を変化させるエッジの構造的類似度を全て再計算し直す必要がある．これに対して，提案手法は構造的類似度の再計算を必要とせず，前時刻の計算結果から現在時刻の類似度の値を差分計算により求めることができる．ゆえに，各エッジに対する類似度計算にかかるコストが小さくなることから，提案手法は eTrack よりも高速に動作していると考えられる．

### 5.2 クラスタリング精度の比較

本節では提案手法のクラスタリング精度を評価する．本研究では精度の評価も定義 19 で与えた，構造的類似度に基づく Modularity を用いる．なお，本実験で用いた各種パラメータの設定は前節と同様である．

本研究で用いた 4 種類の動的グラフに対するクラスタリング精度の評価結果を図 2 に示す．図 2 に示したように，提案手法はいずれのデータセットにおいても $Q_s$ の値が大きく良いクラスタリング結果を示していることが確認できる．一方で，Kim-Han09 や eTrack は提案手法よりも $Q_s$ の値が小さくなっていることがわかる．特に，eTrack は時刻経過とともに $Q_s$ の値が次第に低下している．

この理由は，各手法のパラメータ $\epsilon$ の決定方式にある．まず，提案手法と Kim-Han09 のクラスタリング精度の違いについて議論する．提案手法および Kim-Han09 はパラメータ $\epsilon$ を自動決定する手法である．提案手法は，Cluster Skeleton を構築することで，入力されたグラフがクラスタを作ることができる全ての $\epsilon$ を列挙し，構造的類似度に基づく Modularity$Q_s$ を最大化する $\epsilon$ を決定している．これにより $Q_s$ の値が高くなるクラスタリングを実現している．これに対して，Kim-Han09 ではパラメータ $\epsilon$ の最適化を行うものの，全ての $\epsilon$ を列挙しないため局所最適な $\epsilon$ を用いる．ゆえに，両手法とも動的グラフの構造変化に追従した $\epsilon$ の自動選択を行うものの，提案手法のほうが高い精度を示す結果となっている．次に，提案手法と eTrack のクラスタリング精度に関して議論する．eTrack はパラメータを自動調整することが出来ない手法である．したがって，初期グラフにおいて高い $Q_s$ の値を示していたとしても，時間経過とともに $\epsilon$ の値が最適ではなくなる可能性がある．その結果として，図 2 に示したように，eTrack の $Q_s$ の値は時間経過にしたがって低下していると考えられる．

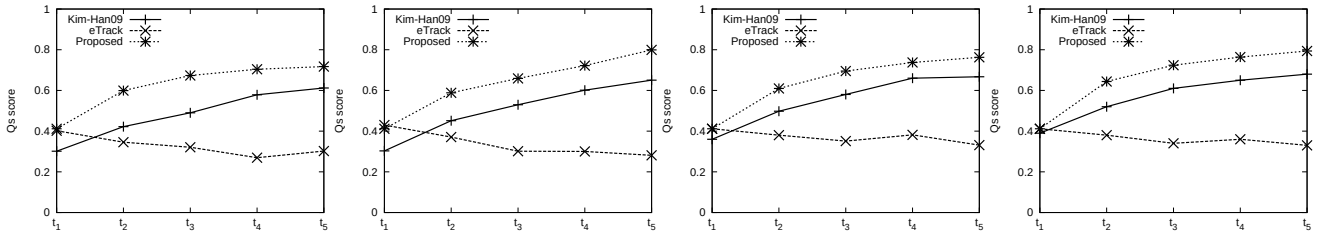
## 6. 関 連 研 究

グラフ構造中からクラスタを抽出するグラフクラスタリングはデータマイニング分野において重要な技術であり，これまで min-max cut [3] や normalized cut [4]，Modularity に基づく手法 [16], [17], [1], [2] など，数多く研究されてきた．本節ではこれらの中でも，本稿の議論の対象である構造的類似度に基づく密度ベースクラスタリング手法 [5], [18], [19], [6], [14], [20] は近年注目を集めている手法のひとつである．

### 6.1 静的なグラフ構造に対する手法

静的なグラフ構造を対象とした，構造的類似度に基づく密度ベースクラスタリング手法 [5], [18], [19], [6], [14], [20] は，データマイニング分野において近年利用され始めているグラフクラスタリング手法である．4. 節で述べたとおり，この手法では事前にクラスタを構成するための類似度の閾値 $\epsilon$ とクラスタを構成する最小ノード数 $\mu$ をパラメータとして与える．これにより任意の大きさ，形状のクラスタを抽出できるだけではなく，クラスタに併せてハブや外れ値といったノードを抽出することが可能である．

代表的な手法として Xu らによる SCAN [5] が挙げられる．SCAN は多次元ベクトルデータに対する密度ベースのクラスタリング手法としてよく知られている DBSCAN [21] の概念をグラフ構造に適用した手法である．1. 節で述べたように，事前に全てのエッジに対して構造的類似度を計算し，パラメータ $\epsilon$, $\mu$ に従い，クラスタの核となる core と定義されるノードを見つける．その後 SCAN は検出された core を中心にクラスタを拡張し，最終的にクラスタに属するノード集合とそれ以外の集合にノードを分類する．クラスタに属さなかったノード集合のうち，2 つ以上のクラスタに接続しているノードについてはハブ，それ以外のノードは外れ値と判定される．SCAN では 1 つのエッジに対する構造的類似度の計算に $O(|\mathbb{V}|^2)$，グラフ構造全体で $O(|\mathbb{V}|^4)$ の計算量が必要となる．さらにクラスタリング自体にかかる計算量も踏まえると，SCAN 全体で $O(|\mathbb{V}|^4)$ から $O(|\mathbb{V}|^4 + |\mathbb{V}|^2)$ 程度の計算コストが必要となる．この理由から，大規模な動的グラフに対し SCAN を適用することは難しい．




(a) Graph(1%, Inner) (b) Graph(5%, Inner) (c) Graph(1%, Outer) (d) Graph(5%, Outer)

図 2 クラスタリング精度の比較

また，SCAN の拡張手法として Bortner らによる SCOT+HintClus [19] および Huang らによる gSkeletonClu [14] が挙げられる．これらの手法では，SCAN の問題点としてパラメータ $\epsilon$ 決定の難しさを指摘し，$\epsilon$-free なクラスタリング手法をそれぞれ提案している．SCOT+HintClus では DBSCAN の拡張手法として知られる OPTICS [22] の概念を用いて，構造的類似度に基づくノードの走査順を与えることで準最適な $\epsilon$ を見つけ出す．gSkeletonClu では，構造的類似度をエッジの重みとして与えた最小全域木を構築し，この全域木の上でパラメータ $\mu$ の制約を満たすようなパラメータ $\epsilon$ の候補を抽出する．いずれの手法も SCAN と同様に，事前に全てのエッジに対して構造的類似度が計算されていることを前提としている．そのため，従来手法である SCAN と同程度かそれ以上の計算時間を必要とする．

### 6.2 動的なグラフ構造に対する手法

時間変化とともに構造を動的に変化させるグラフ構造に対する密度ベースクラスタリング手法 [7], [8], [9], [10] が近年提案されつつある．動的グラフに対する密度ベースクラスタリング手法は大きく分けて 2 種類に分類される．

ひとつは，Chakrabarti らによって提案された temporal smoothing [23] を用いる手法である．temporal smoothing とは，(1)snapshot quality と (2)history quality と呼ばれる 2 つの指標を動的グラフの各時刻毎に評価し，両指標を極大化させる解を各スナップショットのクラスタリング結果として獲得するアプローチである．snapshot quality は，各時刻におけるグラフ構造を静的なグラフ構造とみなして，密度ベースクラスタリングを適用した結果との類似度であり，history quality は直前の時刻におけるクラスタリング結果と現時刻におけるクラスタリング結果の類似度を示したものである．この両指標を満たすクラスタリング結果を各時刻に対して求めることで，動的グラフのクラスタリングを実現する．

この概念に基づくクラスタリング手法として代表的な手法は Lin らによる手法 [7] や Kim らによる手法 [8] が挙げられる．ところがこれらの手法では，各時刻に対して都度，密度ベースクラスタリング (*e.g.* SCAN) を実行する必要があり，各時刻において $O(|\mathbb{V}|^4)$ 程度の計算量が必要となる．また，密度ベースクラスタリング手法および temporal smoothing のために複数のパラメータ設定を必要とすることから，大規模な動的なグラフ構造に対する適用が難しい．近年，temporal smoothing におけるパラメータ設定を除去するために Folino らが遺伝的アルゴリズムによる手法 [10] も提案しているが，その計算量は既存手法 [7], [8] よりも極めて大きく依然として大規模なグラフへの適用が難しい．

temporal smoothing とは異なるもうひとつの動的グラフクラスタリングアプローチとして，クラスタの差分更新による手法がある．Lee らは動的に変化するクラスタから SCAN に基づく core ノードのみを Skeleton として獲得し，グラフの時系列変化に併せて Skeleton を更新していく手法 *eTrack* を提案した [9]．Skeleton を構築し，それを差分更新するという点において，提案手法と同様のアプローチであるが，Lee らによる手法は密度ベースクラスタリング実行の際に複数のパラメータ設定を要求する．これに対して，提案手法では Skeleton の構築方法に Core-Connected Similarity による Maximum Spanning Tree を用いている．これにより，密度ベースクラスタリングに必要なパラメータの候補をリストアップし，構造的類似度に基づく modularity を用いたパラメータフリーな動的クラスタリングを実現している．

## 7. おわりに

本稿では，既存のクラスタリング手法のアプローチを融合し，大規模な動的グラフ構造に対する高速，正確かつパラメータフリーな密度ベースグラフクラスタリングの実現に向けた手法を提案した．提案手法では (1) ある時刻 $t$ に入力されたグラフ構造から，グラフ構造に含まれるクラスタ構造を要約した cluster skeleton を抽出し，(2) 時刻 $t+1$ におけるグラフ構造の変化に併せて，事前に構築した cluster skeleton を逐次更新する．その後，(3) cluster skeleton から時刻 $t+1$ におけるクラスタ構造を復元する．さらに重要な点としてこれらの処理は，(4) ユーザによるパラメータ設定を必要とせず，既存の密度ベースクラスタリング手法に対して高速かつ精度を劣化させずに行う．

その結果として，提案手法は以下の特性を有する．

- 高速性: 提案手法は動的グラフに対する既存の密度ベースクラスタリング手法に対して高速である．
- 高精度性: 提案手法は動的グラフに対する既存の密度ベースクラスタリング手法に対して高精度である．
- パラメータフリー: 提案手法はユーザによるパラメータ設定を必要としない．

上記の特性により，提案手法はこれまでグラフクラスタリングを用いていたアプリケーションにおける分析の幅や処理性能の向上に本手法は貢献できる．

## 文　献


[1] 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.

[2] Hiroaki Shiokawa, Yasuhiro Fujiwara, and Makoto Onizuka. Fast Algorithm for Modularity-based Graph Clustering. In *Proc. AAAI*, pages 1170–1176, 2013.

[3] Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A Min-max Cut Algorithm for Graph Partitioning and Data Clustering. In *Proc. ICDM*, pages 107–114, 2001.

[4] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE TPAMI*, 22(8):888–905, 2000.

[5] Xiaowei Xu, Nurcan Yuruk, Zhidan Geng, and Thomas A. J. Schweiger. SCAN: A Structural Clustering Algorithm for Networks. In *Proc. KDD*, pages 824–833, 2007.

[6] Heli Sun, Jianbin Huang, Jiawei Han, Hongbo Deng, Peixiang Zhao, and Boqin Feng. gSkeletonClu: Density-Based Network Clustering via Structure-Connected Tree Division or Agglomeration. In *Proc. ICDM*, pages 481–490, 2010.

[7] Yu-Ru Lin, Yun Chi, Shenghou Zhu, Hari Sundaram, and Belle L. Tseng. FacetNet: A Framework for Analyzing Communities and Their Evolutions in Dynamic Networks. In *Proc. WWW*, pages 685–694, 2008.

[8] Min-Soo Kim and Jiawei Han. A Particle-and-density Based Evolutionary Clustering Method for Dynamic Networks. *PVLDB*, 2(1):622–633, August 2009.

[9] Pei Lee, Laks V.S. Lakshmanan, and Evangelos E. Milios. Incremental Cluster Evolution Tracking from Highly Dynamic Network Data. In *Proc. ICDE*, pages 3–14, 2014.

[10] Francesco Folino and Clara Pizzuti. An Evolutionary Multiobjective Approach for Community Discovery in Dynamic Networks. *IEEE TKDE*, 26(8):1838–1852, 2014.

[11] Sungsu Lim, Seungwoo Ryu, Sejeong Kwon, Kyomin Jung, and Jae-Gil Lee. LinkSCAN*: Overlapping Community Detection Using the Link-space Transformation. In *Proc. ICDE*, pages 292–303, 2014.

[12] Pei Lee, Laks V. S. Lakshmanan, and Evangelos E. Milios. Incremental Cluster Evolution Tracking from Highly Dynamic Detwork Data. In *Proc. ICDE*, pages 3–14, 2014.

[13] R. C. Prim. Shortest Connection Networks and Some Generalizations. *Bell System Technology Journal*, 36:1389–1401, 1957.

[14] Jianbin Huang, Heli Sun, Qinbao Song, Hongbo Deng, and Jiawei Han. Revealing Density-Based Clustering Structure from the Core-Connected Tree of a Network. *IEEE TKDE*, 25(8):1876–1889, 2013.

[15] Andrea Lancichinetti and Santo Fortunato. Community Detection Algorithms: A Comparative Analysis. *Phys. Rev. E*, 80:056117, Nov 2009.

[16] M. E. J. Newman. Fast Algorithm for Detecting Community Structure in Networks. *Phys. Rev. E - PHYS REV E*, 69:066133, Jun 2004.

[17] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding Community Structure in Very Large Networks. *Phys. Rev. E*, 70:066111, Dec 2004.

[18] Nurcan Yuruk, Mutlu Mete, Xiaowei Xu, and Thomas A. J. Schweiger. AHSCAN: Agglomerative Hierarchical Structural Clustering Algorithm for Networks. In *Proc. ASONAM*, pages 72–77, 2009.

[19] Dustin Bortner and Jiawei Han. Progressive Clustering of Networks using Structure-Connected Order of Traversal. In *Proc. ICDE*, pages 653–656, 2010.

[20] Weizhong Zhao, Venkata Swamy Martha, and Xiaowei Xu. PSCAN: A Parallel Structural Clustering Algorithm for Big Networks in MapReduce. In *Proc. AINA*, pages 862–869, 2013.

[21] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. KDD*, pages 226–231, 1996.

[22] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering Points to Identify the Clustering Structure. In *Proc. SIGMOD*, pages 49–60, 1999.

[23] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary Clustering. In *Proc. KDD*, pages 554–560, 2006.