情報処理学会論文誌 Vol. 52 No. 4 1739–1754 (Apr. 2011)

時々刻々と成長するグラフのための中心性モニタリング

藤原靖宏[†1,†2] 鬼塚真[†1] 喜連川優[†2]

本論文では時々刻々とノードの数を増やしながら成長するグラフの中から，他のノードまでの距離の和が最小になるノード（最も距離中心性の高いノード）と他のノードまでの距離の最大値が最も小さくなるノード（最も離心中心性の高いノード）を効率的かつ正確に検出し続ける問題を対象とする．過去のこの問題に対する手法では最も中心性の高いノードを高速に求めることができる代わりに計算結果の正確性を犠牲にしていた．本論文は正確性を犠牲にしないで最も中心性の高いノードを求めることができないかということを動機にしている．提案手法は (1) 解の正確性を保ったまま元のグラフを縮退してグラフの近似を行い，その近似グラフにおける中心性の値を計算し，(2) 中心性を求めるときに不必要な計算を早期に打ち止めるという 2 つのアプローチで構成される．これらのアプローチにより数多くのノードを枝刈りできるために効率的な検出が可能になる．理論的解析により提案手法は検出結果が正確であり，また必要となる計算量は従来の手法より少ないオーダとなることが分かった．数十万ノードを含む実データで検証を行ったところ，提案手法は解の正確性を保証したまま従来手法と比較して非常に高速であることが確認された．

Centrality Monitoring for Time-evolving Graphs

Yasuhiro Fujiwara,[†1,†2] Makoto Onizuka[†1]
and Masaru Kitsuregawa[†2]

The goal of this work is to identify the nodes that have the smallest sum of distances to other nodes (the lowest closeness centrality nodes) and the nodes that minimizes the maximum distance to all other nodes (the lowest eccentricity nodes) from graphs that evolve over time. Previous approaches for this problem find the lowest centrality nodes efficiently at the expense of exactness. The main motivation of this paper is to answer the question, 'Is it possible to improve the search time without sacrificing the exactness?'. Our solution is based on two ideas: (1) It computes approximate centrality by reducing the original graph size while guaranteeing the answer exactness, and (2) It terminates unnecessary distance computations early when pruning unlikely nodes. Our theoretical analyses show that our approach guarantees exactness of the answer and that it has a lower order of space and time complexities than the previous approaches. We perform several experiments to verify the efficiency of our approach with real and large datasets that contain several hundreds of thousands of nodes. The results show that our approach can find the lowest centrality nodes significantly faster than the previous approaches while it guarantees the answer exactness.

1. はじめに

グラフは実世界に存在するものとそれらの関係をノードとエッジという形で表現するデータ構造である．その直感的な表現方法によりグラフは数学やコンピュータ科学の分野において広く使われているデータ構造である．

グラフ理論において与えられたノードの中から適切なノードを求める施設配置問題は非常に重要な問題の 1 つである．施設配置問題は最適化問題の典型的なものであり，経済や産業などの広い分野に応用が可能である．この問題は，たとえば輸送・通信・伝送システムなどにおいてその効率性を解析するために用いられている[1)]．この問題においては他のノードからの距離の合計が短いノードの方がより適切だと考えられている．それはその他のノードからの移動コストが小さいことが期待されるためである．グラフ理論においてこの適切さを測る尺度として**距離中心性**と**離心中心性**が用いられている．距離中心性は他のノードまでの距離の和の逆数と計算され，離心中心性は他のノードまでの距離の最大値の逆数と計算される．

これらの中心性を素直に計算するには幅優先探索を用いる．しかしグラフが大規模になると幅優先探索による手法は非常に高い計算コストがかかるため実用的ではない．そのため過去の手法[2)–4)] では中心性の近似値を計算する．これらの手法は解の正確性を犠牲にする代わりに幅優先探索による手法と比較して高速に解を求めることができる．しかし近似による手法は必ずしも多くのアプリケーションでは利用されない．これは近似による手法では最も中心性の高いノードを正確に計算することができないため，アプリケーションの面から好ましくないためである．さらに従来の研究においてはグラフは静的でありノード数は変化しないという大きな前提があった．しかし近年，時間とともにノード数が時々刻々と増大し，

†1 日本電信電話株式会社 NTT サイバースペース研究所
NTT Cyber Space Laboratories, NTT Corporation
†2 東京大学生産技術研究所
Institute of Industrial Science, The University of Tokyo

結果的にノード数が数十万以上になるグラフデータを扱うことが必要となってきた．これは巨大なデータベースとインターネットを用いる環境が整ってきた結果である．そして近年の研究により時々刻々とノードの数を増やしながら成長し続けるグラフデータの特徴が明らかになりつつある[5),6)]．そのため時々刻々と成長し，結果的に非常に大きなサイズになるグラフを効率的に処理する需要が高まりつつある．

本論文では時々刻々と成長し続けるグラフの中から最も高い中心性を有するノードを検出し続ける問題を扱う．定式的に述べると本論文では以下の問題を対象とする．

問題　（中心性のモニタリング）

Given：時刻 t におけるグラフ $G[t] = (V, E)$．なおここで V はノードの集合で E はエッジの集合とする．

Find：$G[t]$ において距離中心性の値が最も高くなるノードと離心中心性が最も高くなるノード

我々の知る限り本論文は時々刻々と成長するグラフの中から中心性の最も高いノードを正確に検出し続ける問題を対象とした初めてのものである．

1.1　提案手法の特徴

本論文では時々刻々と成長するグラフの中から中心性が最も高いノードを検出する手法を提案する．提案手法では計算コストを低減するために，(1) 元のグラフの近似を行い，近似グラフにおける中心性の値（以下**中心性の近似値**とする）によりノードを枝刈りする手法と，(2) 不必要な距離計算を早期に打ち止めして中心性の低いノードを効率的に枝刈りする手法を提案する．提案手法には以下のような特徴がある．

- **正確**：提案手法は中心性の低いノードを枝刈りするために近似計算を用いるが，検出結果は正確であることが保証される．従来の手法は解の正確性を犠牲にしていたが，提案手法は最も中心性の高いノードを正確に検出する．
- **効率的**：ノードの数とエッジの数をそれぞれ n，m としたとき，提案手法に必要な計算時間は $O(n^2 + nm)$ となる．しかし従来の近似による手法に必要な計算時間は $O(n^3)$ であり，大規模なグラフに適用することは困難である．
- **省メモリ**：提案手法に必要なメモリ量は従来の近似による手法と比較して少ない．提案手法に必要なメモリ量が $O(n+m)$ であるのに対して，近似による手法に必要なメモリ量は $O(n^2)$ である．

検出の高速性と検出結果の正確性を両立させるために，提案手法ではまず中心性の近似値により多くのノードを枝刈りする．正確に中心性を計算するノードは必要最低限になるために高速な検出が可能になる．提案手法の効率性を検証するため数十万ノードを含む実データを用いて従来の手法と比較を行った．その結果提案手法は最大 110 倍まで高速に最も中心性が高いノードを検出することができることを確認した．

1.2　応　用　例

本論文が対象とする問題の応用例として以下のようなアプリケーションがあげられる．

1.2.1　ソーシャルネットワーク

社会科学者たちによって様々な相互関係のネットワークについての研究が長い間行われてきた．これらのネットワークにおいてノードは研究者や論文や本やジャーナルを表し，エッジは共著関係や参照を表す[7)]．この学術的なネットワークは非常に速い速度で増大することが知られている．たとえばデータベース分野の共著関係のネットワークは数万以上の研究者で構成され，また研究者の数は年々数千以上増加していっていることが知られている[8)]．

今までの研究において「ネットワークにおいてどのノードが最も重要であるか？」という問題は社会科学者たちの大きな関心を集めてきた．直感的にはノードが多くのエッジとつながっているということは，関心や人気をそのネットワークにおいて集めているということである．具体的には論文や本が多くの参照を集めるほど，他の多くの研究者から有用であると判断されたということである．Garfield はジャーナルに対する重要度の尺度としてインパクトファクタを提案した．インパクトファクタは出版ごとの参照の数から定義される[9)]．基本的にインパクトファクタはグラフにおける次数に対応するシンプルな尺度である．

しかしこのような尺度はグラフ全体から見れば局所的な尺度である．それは次数がノードの隣接ノードの個数から決定されるからである．そのため次数の高いノードであっても，もしグラフ内の孤立したコミュニティに属するのであれば，そのノードの全体に対する影響は大きいとは限らなくなる．

距離中心性の値は他のノードまでの距離の和の逆数から計算されるため，局所的な尺度ではない．そのため距離中心性によってノードの重要度を測るのは次数より適切である可能性がある．時系列的に最も重要なノードは時々刻々と成長するグラフを距離中心性に基づきモニタリングすることで検出することができる．

たとえば Nascimento らはデータベースにおける著名な国際会議である SIGMOD の共著者関係のグラフを解析した[10)]．そして彼らは L.A. Rowe と M. Stonebraker と M.J. Carey がそれぞれ 1986 年から 1988 年，1989 年から 1992 年，1993 年から 2002 年まで最も影響力のある研究者であることを検出することに成功した．これら 3 人の研究者はデータベースコミュニティにおいて非常に著名な研究者として知られている．なお他の研究者と共著関

係が最もある著者（最も次数が高いノード）は必ずしも距離中心性の値が最も高かったわけではなかったことは，情報検索の分野の国際会議の共著者関係を解析した結果から示されている[11]．

1.2.2　P2P ネットワーク

近年プロセッサ，メモリ，無線技術の発達により安価に P2P センサネットワークを構築することが可能になってきた．その結果センサネットワークによって人が入ることが困難である環境であっても容易にセンサからの情報を集めることができるようになった．

多くのセンサネットワークのモデルが今までに提案されているが，基本的な特徴は共通している．センサネットワークは多くのセンサノードと一握りのマスタノードから構成される．センサノードは制限された計算しか行わず，他のノードに情報を伝達する際に多くの電力を消費する．一方マスタノードはすべてのセンサノードからの情報を集約してそれらの解析を行う．センサネットワークの設計において最も重要なのはいかに省電力で処理を行うかということである．これは電源の消費量がネットワーク全体の寿命に大きく影響するためである．

離心中心性が最も高いノードからはその他のノードまでの距離が少なくなることが期待される．そのため離心中心性の最も高いノードをマスタノードとすればネットワークの寿命を延ばすことができると期待される．この考えに基づき Wang らはセンサネットワークにおける効果的なマスタノードの決定方法を提案した[12]．彼らは離心中心性が最も高いノードをマスタノードとすることでセンサネットワークにおける電力消費量を大きく低減させることができることを示した．

時々刻々と成長するグラフを解析することは多くのアプリケーションにおいて有用であるが，その処理に多くの計算コストがかかるため実用的ではなかった．しかし時々刻々と成長するグラフを正確かつ高速に解析する技術により今後多くのアプリケーションが開発されることが期待される．

本論文の構成は以下のとおりである．2 章で関連研究を述べる．3 章で本研究における背景知識を説明する．4 章で提案手法における 2 つのアプローチの説明を行い，提案手法を説明する．5 章で理論的な解析を行う．6 章で提案手法の拡張を行う．7 章で実験の結果を示す．8 章で本論文をまとめる．

2. 関 連 研 究

実際のネットワークにおける特徴について近年研究が行われてきた．その一方データ工学の分野においてはグラフの処理についての研究が行われてきた．しかし我々の知る限りにおいて 1 章にあげた条件をすべて満たす研究は本研究が初めてである．

2.1　ネットワークサイエンス

近年大規模なネットワークについての研究が進み，それについての様々な特性が明らかになっている．

社会科学の分野で早くから明らかになっている特性として，多くのエッジにつながっているノードほど新しくエッジを得るというものがあげられる[5]．この特性によりネットワークの次数は結果的に傾斜分布になることが知られている[13]．

また，すでに知られている特性として，共通のノードにつながっているノードどうしほど新しくエッジを得るというものがあげられる[14]．この特性は電子メールのログを解析した研究によって確認されている[15]．

その他の特性として，ネットワークが成長するほどノードあたりの平均次数が増加していくものがあげられる．結果的に，ネットワークにおいて最も距離の離れているノードの距離がネットワークの成長とともに減少していくことが知られている[6]．

2.2　データ工学

グラフのノード間の距離を近似する多くの研究が行われてきた．過去に提案された近似手法は，分割による手法と埋め込みによる手法の 2 つに大別することができる．

Rattigna らは中心性を高速に計算する手法として network structure index という分割による手法を提案した[2]．これはグラフを小さな領域に分割して，分割された領域までの距離によって距離を近似する手法である．彼らは埋め込みによる手法と比較してより良い精度でグラフのノード間の距離を推定できることを示した．しかし彼らの手法はその論文中に述べられているように最も中心性の高いノードを求めるために $O(n^2)$ のメモリ量と $O(n^3)$ の計算時間が必要である．

ランドマークによる手法は埋め込みによる手法の 1 つであり，ノード間の距離を $O(n)$ の計算時間で推定する[3),16]．グラフ中で複数のランドマークとして選ばれたノードの中からそれらの 1 つを経由した距離の最小値がノード間の距離として用いられる．Ng らは埋め込みによる手法として L_p ノルムをノード間の距離として推定する手法を提案した[4]．これらの埋め込みによる手法は $O(n^2)$ のメモリ量が必要である．これはすべての n 個のノードに対して $O(n)$ 個の選択されたノードまでの距離を保持する必要があるためである．さらに最も中心性の高いノードを計算するためには $O(n^3)$ の計算時間が必要である．これはすべての n 個のノードに対して中心性を計算するための計算時間が $O(n^2)$ だからである．

表 1　主な記号の定義
Table 1　Definition of main symbols.

記号	定義
V	グラフ G におけるノードの集合
E	グラフ G におけるエッジの集合
n	ノードの数
m	エッジの数
t	タイムスタンプ $t \geq 1$
$d(u,v)$	ノード u から v までの距離
θ	解候補のノードの正確な中心性
C_u	ノード u の距離中心性
N_u	ノード u の近接ノードの集合

Sun らはテンソル解析を用いてグラフのマイニングを行う手法を研究した[17]．静的なグラフは隣接行列により表現できるため，特異値分解により効果的に近似することができる．時々刻々と成長するグラフは静的なグラフの重なりとして表現できるが，彼らはそれを小さなサイズのテンソルと行列として表現する方法を提案した．

3. 前　準　備

この章では本論文で用いる記号を定義し，必要となる背景知識を説明する．**表 1** に主な記号とその定義を示す．

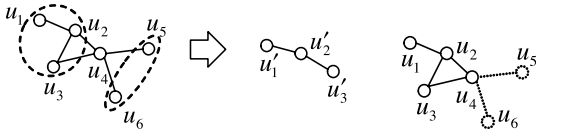
実世界のネットワークはグラフ $G = (V, E)$ として表現することができる．ここで V はノードの集合とし，E はエッジの集合とする．また n と m はそれぞれノードとエッジの数とする．すなわち $n = |V|$ であり $m = |E|$ である．

ノード u からノード v までの**パス**とはノード u から始まりノード v で終わるエッジの並びである．パスの中に含まれるノードが最も少ないものは**最短パス**と呼ばれる．ノード u とノード v の**距離**は最短パスに含まれるエッジの数の合計であり，$d(u,v)$ と定義される．定義からノード $u \in V$ に対して $d(u,u) = 0$ であり，ノード $u, v \in V$ に対して $d(u,v) = d(v,u)$ となる．

ノードの距離中心性はその他のノードまでの距離の和の逆数であり，以下のように定義される．

定義 1（ノードの距離中心性）　ノード u の距離中心性 C_u は以下のように定義される．

$$C_u = \frac{1}{\sum_{v \in V} d(u,v)} \tag{1}$$



(1) ノードの集約　　(2) 探索木による推定

図 1　提案手法
Fig. 1　Proposal approach.

ノードの離心中心性はその他のノードまでの距離の最大値の逆数であり，以下のように定義される．

定義 2（ノードの離心中心性）　ノード u の離心中心性 E_u は以下のように定義される．

$$E_u = \frac{1}{\max\{d(u,v) : v \in V\}} \tag{2}$$

4. 提 案 手 法

この章では提案手法において用いる 2 つの手法について説明する．提案手法は時々刻々と成長するグラフに対して高速かつ正確に中心性の最も高いノードを求めることに大きな特徴がある．この章ではまず距離中心性の最も高いノードを求める手法について説明する．離心中心性が最も高いノードを求める手法については 6 章で説明する．また重みなし無向グラフを前提に議論を進めるが，6 章で他の種類のグラフに対する手法を説明する．また 6 章では範囲検索（与えられた閾値以上の中心性を持ったノードを求める）および K-上位検索（中心値が上位 K 個のノードを求める）の処理方法について述べる．この章では簡単のため各ノードの中心性の値はそれぞれ異なるものとし，また各時刻においては 1 つのノードのみがグラフに追加されるとする．

まず 2 つの手法の概要を説明してから，それぞれの手法の詳細な説明を行う．

4.1 概　　要

提案手法は**図 1** に示す以下の 2 つの手法で構成される．

ノードの集約　従来の手法における莫大な計算量を削減するために，提案手法では近似計算を行う．すべてのノードに対して正確な中心性を計算する代わりに，高速に中心性の近似値を計算し中心性の低いノードを効率的に枝刈りする．

はじめのアプローチはグラフのサイズを小さくするというものである．元のノードが n 個

でエッジが m 個のグラフに対して，似たノードどうしをまとめることによりノードが n' 個でエッジが m' 個（$n' < n$，$m' < m$）であるようなグラフを新たに計算する．この近似のグラフにおいて中心性の近似値を計算するための計算時間は $O(n' + m')$ である．これは従来の近似による手法における中心性の計算時間 $O(n^2)$ と比較すると大きな削減である．元のグラフにおいて似たノードを計算するためには Jaccard 係数を用いる．本論文において元のグラフをまとめることを**集約**すると呼ぶこととする．

この手法には 2 つの優位性がある．はじめの優位性として中心性の近似値によってノードの枝刈りを行っても最も中心性の高いノードを枝刈りすることはないことがあげられる．これは中心性の近似値が正確な中心性の値よりも小さくなることがないためである．この手法により正解になりえないノードを効率的に枝刈りすることができる．2 つめの優位性は近似を行うことにより中心性を計算する回数を減らすことができることがあげられる．ノードの数を減らすことにより 1 回あたりの近似の計算を高速に行うことができるだけでなく，中心性の計算回数も減らすことができる．そのためグラフが大規模になっても効率的な処理が可能となる．

探索木による推定　上記のノードを集約するアプローチにより低い中心性を持つノードの大部分を枝刈りすることができるが，検出結果が正確になるためには正確な中心性の計算を行う必要がある．本手法はこの計算量を削減するためのものである．

正確な中心性を計算するためには始点ノードからその他のすべてのノードまでの距離を計算する必要がある．この距離は幅優先探索により求めることができるが，すべてのノードを探索するのはグラフが大規模になることを考えると好ましくない．本手法では始点ノードが最も高い中心性になりえなくなった時点で距離の計算を打ち切る．

提案手法では検出結果を出力する前に高い中心性を持つ解候補のノードを保持している．またあるノードの中心性を計算する途中で探索していないノードの距離を推定しそのノードの中心性の上限値を計算している．中心性の上限値は 1 つの幅優先探索木から計算される．検出処理においてもしあるノードの中心性の上限値が解候補の中心性より小さければ，そのノードは解になることはない．そのため不必要な距離の計算を打ち切ることができる．

中心性の推定を行うことにより幅優先探索においてグラフ全体の探索を行う必要がなくなるため，結果的に効率的な処理が可能となる．

本手法は正確な中心性の計算のみでなく，中心性の近似値の計算においても適用できる．すなわちこの手法によって中心性の近似値の計算をより高速に行うことができる．

4.2　ノードの集約

はじめのアプローチは元のグラフのノードを集約するものである．高速に中心性の上限値を計算することにより，誤りなく中心性が低いノードを枝刈りすることができる．

4.2.1　グラフの近似

元のグラフの集約をして中心性の近似値を高速に計算を行う．高速な検出を行うために n 個のノードと m 個のエッジを持つグラフ G から n' 個のノードと m' 個のエッジを持つグラフ G' を計算する．すなわちこの処理によってグラフ $G = (V, E)$ は近似グラフ $G' = (V', E')$ に縮約される．

ここではまず近似グラフにおけるエッジの計算方法を述べ，それから元のグラフのノードを集約する手法について述べる．

元のグラフにおいて集約されているノードがエッジを持つときにのみ，近似グラフにおいてノード u' と v' はエッジ $\{u', v'\} \in E'$ を持つとする．また近似グラフは輪（self-loop）となるエッジは持たないとする．これらの定義は後に述べるように中心性の上限値を計算するために必要なものである．形式的には近似グラフがエッジを持つための必要十分条件は以下のとおりである．

定義 3（ノードの集約）　近似グラフ G' においてノード u' と v' は以下の条件を満たすときにのみエッジを持つものとする．

$$(1)\ u' \neq v', \quad (2)\ \exists \{u, v\}, u \in u' \text{かつ } v \in v' \tag{3}$$

ここで $u \in u'$ は集約ノード u' が元のノード u を含むことを示す．

グラフの近似により，元のグラフにおいて存在しないエッジを近似グラフにおいて張ることが起こりうる．そのようなエッジが多くなるほど近似グラフにおける中心性と元のグラフにおける中心性の差が大きくなる．そのためそのようなエッジの本数を減らすために元のグラフにおいて似たノードどうしを集約する．すでに述べたとおり集約ノードどうしは少なくとも元のノードが 1 つでもエッジを持つときにのみエッジを持つ．その結果元のノードの隣接ノードの集合が似ているほど，1 つのノードに集約したときに元のグラフにおいて存在しないエッジを近似グラフにおいて張ることを少なくできる．本アプローチでは集合の類似性の評価基準としてよく用いられる Jaccard 係数を用いて類似のノードを集約する．しかし提案手法は Jaccard 係数だけでなく Dice 係数や Simpson 係数も類似性の評価基準として用いることができる．類似性の評価基準としてどれを用いても提案手法の高速性にほとんど影響がないことは 7 章に示す．

N_u と N_v をそれぞれノード u と v の隣接ノードの集合とする．すると Jaccard 係数は

$|N_u \cap N_v|/|N_u \cup N_v|$ として計算することができる[18]．これはノード u とノード v に共通する隣接ノードの数を，ノード u とノード v のいずれかに隣接するノードの数で割ったものである．本手法においてはノード u に対してノード v が最も似ているときそれらのノードを集約する．ただし似ていないノードを集約しないために，もし共通するノードの数が共有するノードの半分より少ない場合は自ノードを最もよく似たノードとしてその他のノードと集約しない．

1 つのノードがグラフに追加されたとき近似グラフを更新するために最も似たノードを更新する必要がある．加えられたノードに対して素直に最も似たノードを求めるには，加えられたノードと他のすべてのノードの間の類似度を計算する必要がある．しかし提案手法では類似度の計算を高速に行うために以下の補助定理を用いる．

補助定理 1（類似ノードの更新） 追加されたノードに対して最も似ているノードまでの距離はせいぜい 2 である．

証明 共通する隣接ノードを持たない 2 つのノードの Jaccard 係数はその定義より 0 である．共通する隣接ノードを介した長さ 2 のパスが存在するため，Jaccard 係数が正になるノードまでの距離はせいぜい 2 である． □

なおこの補助定理は Jaccard 係数だけでなく Dice 係数や Simpson 係数にも適用可能である．

図 2 に近似グラフを更新するアルゴリズムを示す．このアルゴリズムは上記の補助定理を元にしている．またこの図においてノード v に対して最も似たノードを v_{sim} とする．アルゴリズムではまず加えられたノードから距離が 2 以下であるようなノードを求める．求めたそれぞれのノードに対して加えられたノードとの類似度を計算し，最も類似したノードを更新する．そして最も似たノードを集約することで近似グラフのノードを求める．集約ノード間のエッジは定義 3 によって計算する．

このアルゴリズムは 1 つのノードがグラフに追加されたことを想定しているが，補助定理 1 は 1 つのノードが削除された場合にも適用することができる．すなわち削除されたノードがその他のノードに対して最も似たノードであった場合，新たに最も似たノードを再計算する必要があるが，補助定理 1 を用いることで Jaccard 係数を計算するノードの数を減らし，高速に最も似たノードの再計算を行う．またもし複数のノードが加えられた場合，上記のアルゴリズムを複数回適用することで対応することができる．

4.2.2　中心性の上限値

本アプローチでは中心性の近似値を計算するが，その近似値は元の中心性の値より小さくなることはない．まず近似値を計算する方法を述べる．それから近似グラフにおけるノード間の距離はそれに対応する元のグラフにおける距離より長くなることはないことを述べる．そしてこの特性によって近似グラフから中心性の上限値を求めることができることを示す．

近似グラフが与えられたとき，距離中心性の近似値を以下のように求める．

定義 4（距離中心性の近似値） 近似グラフにおけるノード u' の距離中心性の近似値を以下のように計算する．

$$C_{u'} = \frac{1}{\sum_{v' \in V'} \{d(u', v') \cdot |v'|\}} \tag{4}$$

ここで $|v'|$ は集約ノード v' に含まれる元のグラフにおけるノードの数とする．

中心性の近似値が元の中心性の上限値となる性質を示すために，以下の補助定理を示す．

Algorithm Update
input: $G[t] = (V, E)$, a time-evolving graph at time t
　　u_{add}, the added node at time t
output: $G'[t] = (V', E')$: the approximate graph.
$V' = \emptyset$;
$E' = \emptyset$;
compute distances from u_{add};
for each node v s.t. $d(u_{add}, v) \leq 2$ **do**
　compute the similarity coefficient;
　update the most similar node of v, v_{sim}, from $\{V, u_{add}\}$;
end for
for each node $v \in V$ **do**
　if $\nexists v'_{sim}$ s.t. $v_{sim} \in v'_{sim}$ in V' **then**
　　add node v'_{sim} to V';
　end if
　aggregate node v to v'_{sim};
end for
for each node $v' \in V'$ **do**
　if $v' \neq w'$かつ $(\exists\{v, w\}, v \in v'$かつ $w \in w')$ **then**
　　add edge $\{v', w'\}$ to E';
　end if
end for
return $G'[t]$;

図 2　近似グラフの更新
Fig. 2　Update of approximate graph.

補助定理 2（ショートカット） 近似グラフのすべてのノードに対して以下の不等式が成立する．

$$d(u', v') \leq d(u, v) \tag{5}$$

証明 元のグラフのノード u と v の間の最短パスを $u \rightsquigarrow v$ とする．またその最短パスの一部分を $w_i \rightsquigarrow w_j$ とする．もし最短パスにおけるすべてのノードが異なる集約ノードに集約されたとすると，その最短パスの長さと近似グラフで対応するパスの長さは等しくなる．これは近似グラフ上におけるパス $u' \rightsquigarrow w'_i \rightsquigarrow w'_j \rightsquigarrow v'$ は元のグラフにおける最短パス $u \rightsquigarrow w_i \rightsquigarrow w_j \rightsquigarrow v$ と完全に一致するからである．もしそうでなければ最短パスにおいて少なくとも 2 つ以上のノードが 1 つの集約ノードに集約されていることとなる．すなわち元のグラフのノード w_i と w_j がノード w' に集約されていることとなる．そのため元のグラフにおける最短パス $u \rightsquigarrow w_i \rightsquigarrow w_j \rightsquigarrow v$ は近似グラフにおいてパス $u' \rightsquigarrow w' \rightsquigarrow v'$ に短縮される．結果として $d(u', v') \leq d(u, v)$ となる． □

例 1 図 1 (1) のグラフにおいて元のグラフのノード u_1 と u_2 と u_3 は近似グラフのノード u'_1 に，ノード u_4 はノード u'_2 に，ノード u_5 と u_6 はノード u'_3 にそれぞれ集約されている．

元のグラフのノード u_2 とノード u_5 の最短パスにはノード u_2 とノード u_4 とノード u_5 があるが，これらのノードはそれぞれノード u'_1，ノード u'_2，ノード u'_3 とそれぞれ異なるノードに集約されている．そのため元のグラフのノード u_2 とノード u_5 の最短パスの長さは，近似グラフにおいて対応するノード u'_1 とノード u'_3 の最短パスの長さと等しい．

また元のグラフのノード u_1 とノード u_3 の最短パスにはノード u_1 とノード u_2 とノード u_3 があるが，ノード u_1 とノード u_3 は同じノード u'_1 に集約されている．そのため元のグラフにおける最短パスの長さは近似グラフにおける最短パスの長さより短い．

補助定理 2 から中心性の近似値について以下の定理を示すことができる．

補助定理 3（距離中心性の近似値） 近似グラフのすべてのノードについて以下の不等式が成り立つ．

$$C_{u'} \geq C_u \tag{6}$$

証明 すべての集約ノードについて補助定理 2 から以下の不等式が成り立つ．

$$d(u', v') \cdot |v'| \leq \sum_{v \in v'} d(u, v) \tag{7}$$

そのため以下の不等式が成り立つ．

$$C_{u'} = \frac{1}{\sum_{v' \in V'} \{d(u', v') \cdot |v'|\}} \geq \frac{1}{\sum_{v \in V} d(u, v)} = C_u \tag{8}$$

よって定理は成り立つ． □

補助定理 3 を用いることで提案手法は検出結果が正確であることを保証している．この証明は 5 章で述べる．

4.3 探索木による推定

元のグラフにおいて中心性を高速に計算するための手法を述べる．この手法ではまず 1 つのノード（以下**基準ノード**とする）を選択し，そのノードをルートノードする幅優先探索木を構築する．そしてグラフの中から 1 つずつノードを選択し（以下**推定対象ノード**とする）中心値の値を推定する．もし推定された値が検出処理の最初に解ノードの候補とするノード（以下**候補ノード**とする）の中心値の値より小さい場合，その推定対象ノードを枝刈りする．この推定は推定対象ノードを起点とした幅優先探索で未探索なノードの距離を推定することで行う．この節ではまずノードの中心性を推定する時に用いる記号の定義をする．

探索処理においては基準ノードをルートノードとする 1 つの幅優先探索木を構築する．その結果基準ノードは第 0 層を構成する．基準ノードに隣接するノードは第 1 層を構成する．基準ノードからの距離が i であるノードによって第 i 層が構成される．基準ノードの選択方法については後述する．

探索処理では推定対象ノード u の中心性が候補ノードの中心性より小さいかを確認する．推定対象ノード u から幅優先探索によって探索済みなノードの集合を V_{ex} とし，未探索なノードの集合を $V_{un}(= V \setminus V_{ex})$ とする．$d_{max}(u)$ を推定対象ノード u から探索済みのノードまでの距離の最大値とする．すなわち $d_{max}(u) = \max\{d(u, v) : v \in V_{ex}\}$ である．さらにある層におけるノードが 1 つでも探索済みであれば，その層を探索済みの層 L_{ex} とする．またある層において探索済みのノードがまったく存在しなければ，その層を未探索な層 L_{un} と定義する．ノード u の層の番号は $l(u)$ として表記する．

これらの記号を用いて推定対象ノード u の中心性を以下の式によって推定する．

定義 5（距離中心性の推定） 幅優先探索において計算を打ち切るために，元のグラフにおいて推定対象ノード u の距離中心性の推定値 $\hat{C}_u$ を以下のように推定する．

$$\hat{C}_u = \frac{1}{\sum_{v \in V_{ex}} d(u,v) + \sum_{v \in V_{un}} e(u,v)} \tag{9}$$

$$e(u,v) = \begin{cases} d_{max}(u) & (v \in V_{un} \cap L_{ex}) \\ d_{max}(u) + \min\{|l(v) - l(w)|\} - 1 & (v \in L_{un}, w \in L_{ex}) \end{cases}$$

もしすべてのノードを探索した場合，この推定値は正確な中心性の値と同じになる．距離の推定値の性質を以下に示す．

補助定理 4（距離中心性の推定） 元のグラフにおいて幅優先探索を行う過程で以下の不等式が成り立つ．

$$\hat{C}_u \geq C_u \tag{10}$$

証明 まず推定対象ノードから幅優先探索によって探索済みの層にある未探索なノードの距離は d_{max} より短くならないことを示す．幅優先探索において未探索なノードの距離は探索済みノードの距離から計算される．この処理のため未探索なノードの距離は幅優先探索において単調増加となり，d_{max} より短くなることはない．

次に未探索な層のノードの距離は $d_{max} + \min(|l(v) - l(w)|) - 1$ より短くならないことを示す．ここでノード w は探索済みの層にあるとする．もしある未探索な層に対して，1 つ上/下の層が探索済みであれば，未探索な層のノードの距離は d_{max} より短くなることはない．これは未探索なノードの距離が d_{max} より短くならないからである．さらにある層に対して 2 つ上/下の層が探索済みでかつ 1 つ上/下の層が未探索であれば，その層にあるノードの距離は $d_{max} + 1$ より短くなることはない．これは 1 つ上/下の層のノードの距離が d_{max} より短くなることがなく，かつその層にあるノードは 2 つ上/下の層のノードにはつながっていないからである．同様にもしある層の i 個上/下の層が探索済みでかつ 1 つから $i - 1$ 個までの層がすべて未探索であれば，その層にあるノードの距離は $d_{max} + i - 1$ より短くなることはない．□

例 2 図 1 (2) のグラフにおいて基準ノードを u_6，推定対象ノードを u_1 とする．基準ノードが u_6 なので，ノード u_1 の層の番号は 3，ノード u_2，u_3，u_5 の層の番号は 2，ノード u_4 の層の番号は 1，ノード u_6 の層の番号は 0 となる．またノード u_1 から幅優先探索を行った場合，ノードを u_2，u_3，u_4，u_5，u_6 の順番で探索するとする．またノード u_1 の中心性の値は 1/11 である．

ノード u_2 まで探索したとき，ノード u_1，u_2，u_3，u_5 が探索済みの層にあり，また $d_{max} = 1$ であるので，$\hat{C}_{u_1} = 1/6 > 1/11$ となる．ノード u_4 まで探索したとき，ノード u_1，u_2，u_3，u_4，u_5 が探索済みの層にあり，また $d_{max} = 2$ であるので，$\hat{C}_{u_1} = 1/8 > 1/11$ となる．ノード u_6 まで探索したとき，すべてのノードが探索済みの層にあり $\hat{C}_{u_1} = 1/11 = C_{u_1}$ となる．

Algorithm Centrality Computation
input: $G = (V, E)$, a graph
　　u, a source node
　　θ, exact centrality of the candidate node
output: $\hat{C}_u$, estimate centrality
$V_{ex} \leftarrow$ empty set;
while $V_{ex} \neq V$ **do**
　compute distance of node v, $d(v,u)$, by breadth-first search;
　append node $v \rightarrow V_{ex}$;
　compute the estimation $\hat{C}_u$;
　if $\hat{C}_u < \theta$ **then**
　　return $\hat{C}_u$;
　end if
end while
return $\hat{C}_u$;

図 3 中心性の計算
Fig. 3 Centrality computation.

この定理から提案手法は検出結果が正確になることが示されるが，その証明は 5 章に示す．

効果的なノードの枝刈りには基準ノードの選択方法が重要である．提案手法では 1 時刻前の最も高い中心性の値を持つノードを基準ノードとして選択する．これには 2 つの理由がある．1 つ目の理由として，このノードとその周辺のノードが 1 時刻経過しても高い中心性を持ち，ノードが加わった後のグラフにおいて解ノードとなることが期待されることがあげられる．これは時々刻々と成長するグラフにおいては少数のノードがすでに存在する多数のノードから構成されるグラフに加わるため，ノードが加わる前後においてグラフに対する変化は小さいからである．2 つ目の理由として，基準ノードに近いノードほど中心性の推定が精度良く行うことができることがあげられる．これは中心性の推定が基準ノードからの距離に基づいているためである．

図 3 に中心性の計算方法を示す．ここで候補ノードの正確な中心性の値を θ とする．このアルゴリズムでは解になりえないノードを θ を用いて枝刈りする．もし中心性の推定値が θ より小さければ，そのノードは最も高い中心性の値を持つことはない．そのため幅優先

Algorithm Centrality Computation
input: $G[t] = (V, E)$, a time-evolving graph at time t
u_{add}, the added node at time t
$u_{best}[t-1]$, the previous best centrality node
output: $u_{best}[t]$: the best centrality node
//Update the approximate graph
update the approximate graph by the update algorithm;
//Search the best centrality node
$V_{exact} \leftarrow$ empty set;
compute θ, the exact centrality of node $u_{best}[t-1]$;
for each node $v' \in V'$ **do**
compute $C_{v'}$, the approximate centrality of node v';
if $C_{v'} \geq \theta$ **then**
for each node $v \in v'$ **do**
append node $v \rightarrow V_{exact}$;
end for
end if
end for
for each node $v \in V_{exact}$ **do**
compute C_v, the exact centrality of node v;
if $C_v > \theta$ **then**
$\theta \leftarrow C_v$;
$u_{best}[t] \leftarrow v$;
end if
end for
return $u_{best}[t]$;

図 4　中心性の検出
Fig. 4　Centrality monitoring.

探索の途中であっても安全に枝刈りすることができる．この推定値はすべてのノードを探索した場合正確な中心性の値となるため，アルゴリズムの最後で $\hat{C}_u$ を返す．

ここでは元のグラフにおいて幅優先探索を打ち切る方法について述べたが，同様の議論は近似グラフにおいても適用できる．

4.4 検 出 処 理

提案手法ではノードの中心性の近似値が候補ノードの中心性の値より低ければ枝刈りを行う．効果的なノードの枝刈りを行うためには高い中心性を持つ候補ノードを選択することが重要である．そのため提案手法では候補ノードとして 1 時刻前の解ノードを選択する．このノードは 4.3 節で述べたとおり高い中心性を持つことが期待されるからである．すなわち提案手法において基準ノードと候補ノードは同じノードになる．

図 4 に最も中心性が高いノードを検出するアルゴリズムを示す．ここで $u_{best}[t]$ を最も中心性の高いノード，$u_{best}[t-1]$ を 1 時刻前の最も中心性の高いノード，u_{add} を新たに加わったノードとする．また V_{exact} は正確な中心性の値を計算するノードの集合とする．

アルゴリズムは更新と検出の 2 つの段階に分けることができる．更新段階においては図 2 に示したアルゴリズムによって近似グラフを更新する．検出段階においてはまず 1 時刻前の最も中心性の高いノードを用いて θ を計算する．もし中心性の近似値が θ より小さければその集約ノードに集約されるノードは解にはなりえない．そのためその集約ノードを枝刈りする．近似値が θ より小さくなければ V_{exact} に集約されているノードを加える．その集約ノードに集約されるノードは解になりうるからである．そして V_{exact} に含まれるノードの正確な中心性の値を計算し，解であるかを確認する．

5. 理論的解析

この章では提案手法における検出結果の正確性および最悪時の計算量についての理論的な解析を行う．この章で行う理論的な解析は距離中心性および離心中心性に対して成り立つ．なお n はノードの数であり，また m はエッジの数である．

5.1 正　確　性

以下に示すとおり提案手法は正確に最も中心性の高いノードを検出することを示す．

定理 1（正確な検出）　提案手法は正確に最も中心性の高いノードを検出することを保証する．

証明　u_{best} を元のグラフにおいて最も中心性の高いノードとし，θ_{max} をノード u_{best} の正確な中心性の値とする．また θ を解候補の正確な中心性の値とする．
近似グラフにおいて $\theta_{max} \geq \theta$ であるためノード u_{best} の中心性の近似値は θ より小さくなることはない（補助定理 3）．同様に元のグラフにおいてノード u_{best} の中心性の推定値は θ より小さくなることはない（補助定理 4）．検出処理においてノードが枝刈りされるのは中心性の近似値または中心性の推定値が θ より小さくなる場合のみである．そのため u_{best} が枝刈りされることはありえない．　□

5.2 計　算　量

提案手法における計算量について議論する．なお既存の近似による手法においては $O(n^2)$ のメモリ量と $O(n^3)$ の計算時間が必要である．また幅優先探索を用いる場合は $O(n+m)$

のメモリ量と $O(n^2+nm)$ の計算時間が必要である．幅優先探索を用いる場合 $O(n^2+nm)$ の計算時間になるのは，計算量が $O(n+m)$ である幅優先探索をすべての n 個のノードに対して行うからである．

定理 2（提案手法のメモリ量） 提案手法は最も中心性の高いノードを求めるために $O(n+m)$ のメモリ量が必要である．

証明 提案手法は元のグラフと近似グラフを保持する．近似グラフにおいてノードとエッジの数はそれぞれ元のグラフのノードとエッジの数を超えることはない．そのため近似グラフを保持するために必要となるメモリ量は $O(n+m)$ である．また元のグラフを保持するためのメモリ量は $O(n+m)$ であるため，提案手法が必要とするメモリ量は $O(n+m)$ である． □

定理 3（提案手法の計算時間） 提案手法は最も中心性の高いノードを求めるために $O(n^2+nm)$ の計算時間が必要である．

証明 最も中心性の高いノードを求めるため，提案手法は近似グラフを求めてから近似グラフと元のグラフでノードの枝刈りを行う．

Jaccard 係数を計算するノードの集合を V_{update} とすると，Jaccard 係数を計算する計算時間が m であるため，近似グラフを更新するために必要な計算時間は $m \cdot |V_{update}|$ となる．ここで $|V_{update}|$ の最悪時（すべてのノードに対して Jaccard 係数を計算する場合）における大きさは $O(n)$ であるため，近似グラフを更新するには $O(nm)$ の計算時間が必要である．また近似グラフと元のグラフのノードとエッジの数はそれぞれ n' と m' であるため，近似グラフによってノードを枝刈りする計算時間 $(n')^2+n'm'$ となり，元のグラフで中心性を計算するコストは $|V_{exact}|(n+m)$ となる．ここで $n' \leq n$，$m' \leq m$，$|V_{exact}| \leq n$ であるため，最悪時（近似グラフのノード数またはエッジ数が元のグラフと同じで，近似グラフを用いてもノードが枝刈りできない場合）においてノードの枝刈りに必要な計算時間は $O(n^2+nm)$ となる．結果，提案手法の計算時間は $O(n^2+nm)$ となる． □

定理 2 と定理 3 は理論的に最悪時において提案手法は既存手法より計算時間が少なく，また幅優先探索と計算コストが同等であることを述べている．しかし提案手法における実際の計算時間は $m \cdot |V_{update}| + (n')^2 + n'm' + |V_{exact}|(n+m)$ となる．これは近似グラフの更新に必要な計算時間が $m \cdot |V_{update}|$ であり，ノードの枝刈りに必要な計算時間が $(n')^2+n'm'+|V_{exact}|(n+m)$ だからである．すなわち提案手法における実際の計算時間は Jaccard 係数を計算するノード数や枝刈りされたノード数に大きく影響を受ける．そのため 7 章において実証実験を行い，提案手法におけるアプローチが有効であることを示す．

6. 提案手法の拡張

この章では提案手法における離心中心性への拡張などについて述べる．

6.1 離心中心性

4 章では距離中心性に焦点を当てて説明を行った．この章では提案手法が離心中心性についても効率的に扱えることを述べる．

近似グラフにおいて離心中心性の近似は以下のように行う．

定義 6（離心中心性の近似） 近似グラフにおいてノード u' の離心中心性の近似値 $E_{u'}$ を以下のように計算する．

$$E_{u'} = \max\{d(u', v') : v' \in V'\} \tag{11}$$

この近似については以下の性質が成り立つ．

補助定理 5（離心中心性の近似） 近似グラフのすべてのノードに対して以下の性質が成り立つ．

$$E_{u'} \leq E_u \tag{12}$$

証明 これは補助定理 2 より明らかである． □

幅優先探索においてノード u の離心中心性の推定を以下のように行う．

定義 7（離心中心性の推定） 元のグラフにおいて幅優先探索を打ち切るために，ノード u の離心中心性の推定値 $\hat{E}_u$ は以下のように計算する．

$$\hat{E}_u = \max\{d(u, v), e(u, w) : u \in V_{ex}, u \in V_{un}\} \tag{13}$$

この推定における以下の性質によって，提案手法は正確に最も離心中心性の高いノードを推定可能である．

補助定理 6（離心中心性の推定） 元のグラフに対して以下の性質が成り立つ．

$$\hat{E}_u \leq E_u \tag{14}$$

証明 これは補助定理 4 の証明より明らかである． □

6.2 有向グラフおよび重み付きグラフ

これまで重みなし無向グラフの処理手法について議論を行ってきたが，提案手法が有向グラフおよび重み付きグラフに対しても処理を行うことができる．4.2 節で述べたとおり，集約されているノードにおいて 1 つでもエッジがあれば近似グラフにおいてもエッジを張る．有向グラフおよび重み付きグラフを扱う場合にはエッジの張り方を少し変えることで対応で

きる．

有向グラフに対してはエッジのそれぞれの方向に対して定義 3 を適用する．重み付きグラフに対しては元のエッジの重みの中で最も小さな値をを用いることで，元のグラフの中心性の近似を行う．

有向グラフに対しては定義 4 をそのまま適用することで中心性の推定を行うことができる．重み付きグラフにおいては多少の変更が必要である．ノード u と v の距離の推定値は $d_{max}(u) + \min\{\omega(v,w) : w \in V \setminus v\}$ として計算する．なおここで $\omega(v,w)$ はエッジ $\{v,w\}$ の重みとする．

6.3　問合せ処理

今まで最も中心性の高いノードを求めることを前提に議論を行ってきたが，ここでは提案手法が範囲検索および K-上位検索も取り扱えることを述べる．

4.4 節で述べたとおり提案手法はまず候補ノードの中心性の値 θ を 1 時刻前の解ノードを用いて計算する．範囲検索および K-上位検索における処理は最も中心性の高いノードを求める処理と基本的に同じである．しかし最も重要な違いは θ の求め方にある．

範囲検索においては与えられた閾値を θ として用いる（すなわち候補ノードは用いない）．すべてのノードに対して中心性の近似値を計算し，それが θ より小さければ枝刈りを行う．

K-上位検索においては 1 時刻前において中心性が上位 K だったノードに対して正確な中心性を再計算する．そしてそれらの中から K 番目の正確な中心性を求めて θ とする．それ以外の処理は最も中心性の高いノードを求める処理と同じである．

6.4　静的なグラフの処理

提案手法は時々刻々と成長するグラフに対する手法であるが，静的なグラフを処理することも可能である．静的なグラフを処理するためには最も似たノードの計算方法および候補ノードの選定方法についての検討が必要である．

動的なグラフを処理する場合，提案手法は近似グラフを求めるために追加されたノードに対してのみ Jaccard 係数を計算し逐次的に最もノードを更新した．しかし静的なグラフの処理に適用する場合，逐次的なアプローチをとることはできない．すなわち近似グラフを求めるためにはすべてのノードに対して最も似たノードを計算する必要があり，その計算時間が問題になる．そのため静的なグラフを対象とするとき場合，提案手法は Broder らが提案した手法[19] を用いて Jaccard 係数を高速に推定する．さらに補助定理 1 を用いて類似度を計算するノードの数を減らす．その結果静的なグラフに対しても高速に近似グラフを計算することができる．類似度を推定することによって近似の精度が低下する可能性があるが中心性の上限値は計算可能であるので，正確に最も中心性の高いノードを計算することができる．静的なグラフを対象とするとき候補ノードは最も次数の高いノードとする．この手法がランダムに候補ノードを選定するよりは高速に最も中心性の高いノードを求めることができることを 7 章の図 16 に示す．

7. 評 価 実 験

提案手法の有効性を調べるために実験を行った．実験では既存のグラフを分割して中心性の近似値を計算する手法および幅優先探索による手法（以後 **Breadth** と表記する）と比較を行った．実験では分割による手法として **Zone** による手法と **Distant to zone** による手法（以後 **DTZ** と表記する）と比較した．これらは本実験で用いたデータセットにおいて埋め込みによる手法より精度高く最も中心性の高いノードを推定できることを確認し，また同様の結果が Rattigna らの論文[2] でも報告されているためである．Zone による手法と DTZ による手法においては zones と dimensions の 2 つのパラメータがある．zones はグラフの分割数であり，dimensions は zones の集合である⋆1．

本実験では以下の 3 点を確認した．

- 高速性：提案手法は近似による手法より 110 倍まで高速であり，ノードの数が増えても高速に処理が可能である（7.1 節）．
- 正確性：近似による手法と異なり，提案手法は最も中心性の高いノードを正確かつ高速に求めることができる（7.2 節）．
- 各手法の有効性：提案手法で用いるノードの集約や探索木による推定は最も中心性の高いノードを求めるのに効果的である（7.3 節）．

実験では公開されている実データ *P2P*[20]，*Social*[21]，*WWW*[22] を用いた．これらはそれぞれ大学の P2P ネットワークのデータ，フリーのソーシャルネットワークサービス，「nd.edu」ドメイン内のウェブデータである．実験ではネットワークから最も大きな連結成分を抜き出した後に，その連結成分において追加されたノードどうしが連結成分になるように 1 つ 1 つランダムにノードを加えていった．これはノードが連結していない場合はノードどうしの距離は計算できないため，定義 1 と定義 2 から明らかなとおり中心性を計算できないからである．

実験は CPU が Intel Xeon Quad-Core 3.33 GHz，メモリが 32 GB の Linux サーバで

⋆1 Rattigna らの論文の条件に合わせるため，すべてのノードのペアの半分をサンプリングして中心性を推定した．

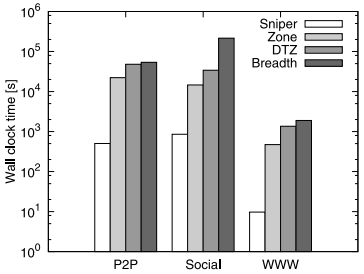


図 5　距離中心性に対する提案手法の高速性
Fig. 5　Efficiency of the proposal approach for closeness centrality.

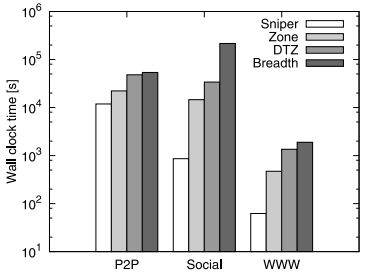


図 6　離心中心性に対する提案手法の高速性
Fig. 6　Efficiency of the proposal approach for eccentricity centrality.

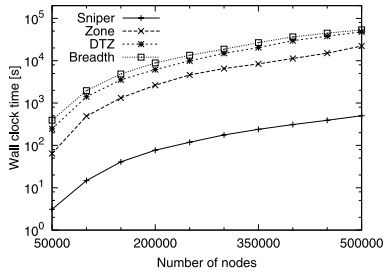


図 7　ノード数と処理時間の関係
Fig. 7　Wall clock time versus number of nodes.

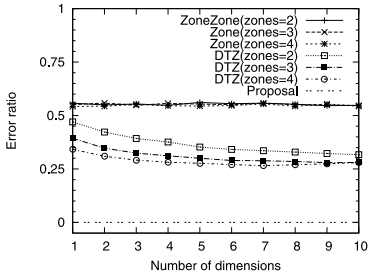


図 8　近似手法の誤差率
Fig. 8　Error ratio of the approximate approaches.

行った．またすべてのアルゴリズムは GCC で実装した．

7.1 処理時間

提案手法，近似手法および幅優先探索による手法の処理時間を比較した．**図 5** と**図 6** に最も中心性の高いノードを求めるために要した処理時間を示す．これらの図においてノードの数は P2P および Social において 500,000，WWW において 10,000 である．また**図 7** に距離中心性における処理時間とノードの数の関係を示す．これらの図で提案手法における処理時間とはノード追加後の更新段階と検出段階の両方を含むトータルの時間である．すなわちここで処理時間とはノード追加があった後から，近似グラフを更新し，近似グラフを用いて最も中心性の高いノードを検出したまでの時間を計測した．また近似手法において処理時間とは，提案手法と同様にノード追加があった後から，グラフを分割し，ノードごとに分割された領域までの距離を計算し，最も中心性の高いノードを検出したまでの時間を計測した．近似手法において zones と dimensions の値はそれぞれ 2 と 1 とした．近似手法はこのパラメータ設定で最も高速に中心性の高いノードを求めることができる．また幅優先探索による手法においてはノード追加があった後から，すべてのノード間の距離を計算し，ノードを検出するまでの時間を計測した．

これらの図から提案手法は近似手法および幅優先探索による手法と比較して大幅に高速に最も中心性の高いノードを求められることが分かる．提案手法はとくに近似手法と比較して 110 倍まで高速であることが確認された．

5 章で述べたとおり，近似手法では検出に $O(n^3)$ の計算時間，幅優先探索による手法では $O(n^2 + nm)$ の計算時間が必要である．一方提案手法は $m \cdot |V_{update}|$ の計算時間で近似グラフを更新し，$(n')^2 + n'm' + |V_{exact}|(n+m)$ の計算時間で中心性の計算を行う．7.3.1 項および 7.3.5 項に示すとおり，近似グラフを更新するために Jaccard 係数を計算するノード数 $|V_{update}|$，および正確な中心性を計算するノード数 $|V_{exact}|$ はグラフのノード数 n と比較して非常に小さい．そのため実際の処理において提案手法は近似手法および幅優先探索による手法より大幅に高速に検出が行えた．

7.2 検出結果の正確性

提案手法の優位性の 1 つとして検出結果が正確であることがあげられる．既存の近似による手法は検出結果が正確でないにしろ，どの程度正確に検出を行うことができるのかを示すことは提案手法の優位性を検証するために必要である．

解の正確性を評価する指標として誤差率を用いた．誤差率は，最も中心性が高いと推定されたノードの正確な中心性の値を，最も中心性が高いノードの正確な中心性の値で割ったものである．**図 8** と**図 9** にそれぞれ近似手法においてパラメータ変化させた場合の誤差率と処理時間を示す．なおこれらの図は距離中心性に対する結果であり，ノードの数は 10,000 であり実験データは P2P を用いた．

図 8 に示されているように提案手法の誤差率は 0 である．これは最も中心性の高いノードを正確に検出することができるからである．しかし近似手法の誤差率は非常に高く，実質的に最も中心性の高いノードを検出することはできない．また図 9 から近似手法は精度を犠牲にしているにもかかわらず提案手法より計算時間が大幅に長いことが分かる．さらに近似手法の高速性は大きくそのパラメータに依存する．

また結果から近似手法は計算時間と適合率のトレードオフがあることが分かる．すなわち

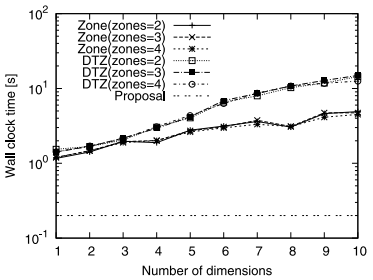


図 9　近似手法の処理時間
Fig. 9　Wall clock time of the approximate approaches.

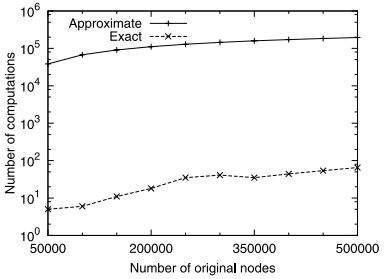


図 10　ノードの計算回数
Fig. 10　Number of centrality computations.

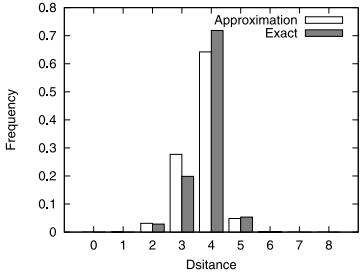


図 11　距離の分布
Fig. 11　Distribution of distances.

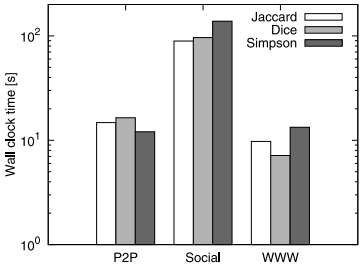


図 12　類似性の比較
Fig. 12　Comparison of similarity measures.

zones や dimensions のパラメータの値が小さくなると，計算時間は少なくなるが誤差率は高くなる．近似手法では最も中心性の高いノードが枝刈りされることがあるため，正確に検出を行うことはできない．提案手法においても近似を用いるが，近似手法と異なり最も中心性の高いノードを枝刈りすることはない．結果的に提案手法は近似手法に対して正確性だけでなく，計算時間に対しても優位であることが示された．

7.3　各手法の有効性

以下の実験では提案手法で用いる各手法（ノードの集約，探索の枝刈り）の有効性を調べた．なお以降においては距離中心性に対する結果を示す．

7.3.1　中心性の近似値

提案手法は中心性の近似値を用いて解になりえないノードを枝刈りする．中心性の近似値の計算回数 n' と正確な値の計算回数 $|V_{exact}|$ は処理時間に大きな影響を与えるため調査を行った．**図 10** に結果を示す．この図において *Approximate* は中心性の近似値の計算回数 n' であり，*Exact* とは正確な中心性の値の計算回数 $|V_{exact}|$ である．なおここでノードの数は 100,000 であり，実験データは P2P である．

図から，近似値と正確な値の計算回数（すなわち近似グラフのノード数）は元のグラフのノード数に対して非常に少ないことが分かる．また提案手法において正確な中心性の計算を行うのはほとんどないことが分かる．これは近似グラフのノードの枝刈り効果が非常に高いからである．このため提案手法は図 5 と図 6 に見られるような高速な性能を実現している．

7.3.2　近似の正確さ

探索処理においては近似グラフを用いて解になりえないノードの多くを枝刈りする．近似グラフにおける近似の正確さを，元のグラフと近似グラフにおける距離の分布を調べることで示す．**図 11** に P2P データにおいてノードの数が 100,000 である場合の結果を示す．

この図から近似グラフと元のグラフにおいてノード間の距離の分布はほとんど同じであり，近似グラフによって非常に正確に元のノード間の距離を近似できていることが分かる．近似グラフにおいてはノード間の距離の下限値を計算するために，集約されているノードが 1 つでもつながっていればエッジを張るが，この処理によっても近似の誤差はほとんど発生しないことが分かる．これは 4.2 節で述べたとおり似ていないノードは集約しないためである．そのため図 10 に見られるように近似グラフによって多くのノードを枝刈りすることができる．

7.3.3　類 似 尺 度

4.2 節で述べたとおり，提案手法は Jaccard 係数をノードを集約するときの類似尺度として用いる．しかし提案手法は他の係数を類似尺度として用いることができる．そのためこの実験では Jaccard 係数のほかに Dice 係数と Simpson 係数の処理時間を調べた．**図 12** にノードを数を 100,000 としたときの結果を示す．

結果に見られるように提案手法は Dice 係数および Simpson 係数を用いても Jaccard 係数と同等に高速に検出が行えることが分かる．Dice 係数および Simpson 係数は集合間の類似度を測るための尺度である．そのため 4.2 節で述べたとおり，集約されるノードが共通した隣接ノードを持つほどこれらの類似度は高くなる．そのためこれらの係数を用いても提案手法は高速に最も中心性の高いノードを計算することができた．

7.3.4　係 数 の 値

4.2 節で述べたとおり，提案手法は似ていないノードを集約するのを防ぐため共通する

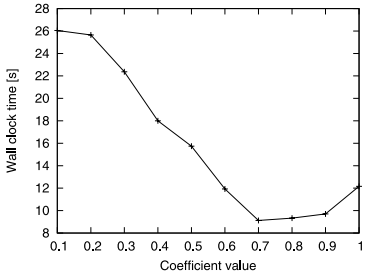


図 13　係数の値と処理時間
Fig. 13　Wall clock time versus coefficient values.

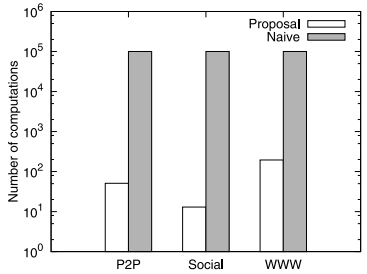


図 14　更新における計算回数
Fig. 14　Number of computations for update.

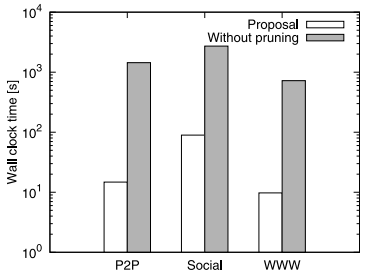


図 15　探索の枝刈りの効果
Fig. 15　Effect of exploration pruning.

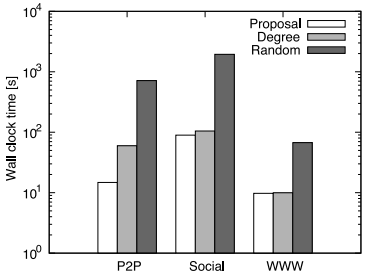


図 16　候補ノードの選択方法の比較
Fig. 16　Comparison of candidate selections.

ノードの数が共有するノードの半分未満である場合ノードを集約しない．すなわちノードの Jaccard 係数が 0.5 以上である場合のみノードの集約を行う．この値は近似グラフの大きさとノードの枝刈り効果に影響するが，提案手法ではノードの集約において他の値を基準として用いることができる．**図 13** に基準の値と処理時間の関係を示す．なおここでノードの数は 100,000 であり，データセットは P2P である．

図から Jaccard 係数の値が小さすぎても大きすぎても処理時間は低下することが分かる．係数の値が大きくなるほど近似グラフの大きさは大きくなり，ノードの枝刈り性能は向上する．しかし中心性の近似値の計算時間も増加する．そのため Jaccard 係数の値が大きくなりすぎると中心性の近似値の計算に多くの時間が必要になる．しかしもし Jaccard 係数の値が小さすぎると中心性の近似値によって効果的なノードの枝刈りが行えない．

7.3.5　更新コスト

提案手法では近似のグラフを計算するために最も類似しているノードを計算する．4 章で述べたとおり，提案手法では新たに加えられたノードの近くのノードに対してのみ類似性を計算し，高速に近似グラフを更新する．実験では提案手法で類似性を計算したノード数 $|V_{update}|$ と，素朴に更新する方法において類似性を計算したノード数を比較した．素朴に更新する方法では加えられたノードとそのほかのすべてのノードに対して類似性を計算した．**図 14** にノードの数が 100,000 のときの結果を示す．

5 章に述べたとおり，提案手法は最悪の場合すべてのノードと類似性を計算する．しかし実際に類似性を計算したノード数 $|V_{update}|$ は非常に少ない結果となった．提案手法では加えられたノードの距離が 2 以下のノードとのみ類似性を計算するが，実際のデータにおいてこの条件を満たすノードは非常に少ないため高速な更新を行うことができる．

7.3.6　探索木による推定

4.3 節で述べたとおり，提案手法は検出処理において必要のない距離計算を打ち切る．この手法の効果を調べるために，提案手法からこの手法を取り除き処理時間を調べた．**図 15** にノードの数を 100,000 としたときの結果を示す．この図では枝刈りの手法を取り除いたものを *Without pruning* とした．

実験結果から探索の枝刈りを行うことにより検出処理が最大 98 倍高速になることが分かる．提案手法では近似グラフと元のグラフにおいて中心性の計算を行うが，探索の枝刈りを行うことによってこれらの計算をより高速に行えることが分かる．

7.3.7　候補ノードの選択

提案手法では 1 時刻前の最も中心性の高いノードを候補ノードとする．この手法の有効性を確認するために実験を行った．実験結果を**図 16** に示す．この図において *Degree* は最も次数の高いノードを候補ノードとした結果，また *Random* はランダムなノードを候補ノードとした結果を示している．ここでノードの数は 100,000 とした．

実験結果から提案手法における候補ノードの決定方法は，最も次数の高いノードを候補ノードとした結果より 4 倍，ランダムなノードを候補ノードとした結果より 40 倍高速であることが分かる．1 時刻前の最も中心性の高いノードは再び最も中心性の高いノードになることが非常によくあるため，時刻とともに成長するグラフに対して効果的に検出を行うことができる．

8. 結　　論

本論文では時々刻々と成長するグラフの中から最も距離中心性の高いノードと離心中心性

の高いノードを検出し続ける問題を対象にした．我々の知る限り本論文はこの問題において正確性と高速性を両立させる取り組みとして初めてのものである．提案手法は，(1) 元のグラフノードを集約して中心性の近似値を計算し，(2) 不必要な距離の計算を打ち切ることを特徴とする．理論的解析により提案手法は解の正確性を保証したまま既存の近似による手法より少ない計算量で最も中心性の高いノードを計算できることが分かった．実験を行ったところ提案手法は既存手法より 110 倍まで高速であることが確認された．ソーシャルネットワークや P2P センサネットワークにおいて中心性の最も高いノードを計算することは重要である．提案手法により正確かつ高速に中心性の最も高いノードを検出することが可能になり，今後多くのアプリケーションが開発されることが期待される．

参 考 文 献

1) Brandes, U. and Erlebach, T.: *Network Analysis: Methodological Foundations*, Springer (2008).
2) Rattigan, M.J., Maier, M. and Jensen, D.: Using structure indices for efficient approximation of network properties, *KDD*, pp.357–366 (2006).
3) Potamias, M., Bonchi, F., Castillo, C. and Gionis, A.: Fast shortest path distance estimation in large networks, *CIKM*, pp.867–876 (2009).
4) Ng, T.S.E. and Zhang, H.: Predicting internet network distance with coordinates-based approaches, *INFOCOM* (2002).
5) Newman: The structure and function of complex networks, *SIREV: SIAM Review*, Vol.45 (2003).
6) Leskovec, J., Kleinberg, J.M. and Faloutsos, C.: Graph evolution: Densification and shrinking diameters, *TKDD*, Vol.1, No.1 (2007).
7) Wasserman, S. and Faust, K.: *Social Network Analysis: Methods and Applications*, Cambridge University Press (1994).
8) Elmacioglu, E. and Lee, D.: On six degrees of separation in dblp-db and more, *SIGMOD Record*, Vol.34, No.2, pp.33–40 (2005).
9) Garfield, E.: Citation Analysis as a Tool in Journal Evaluation, *Science*, Vol.178, pp.471–479 (1972).
10) Nascimento, M.A., Sander, J. and Pound, J.: Analysis of sigmod's co-authorship graph, *SIGMOD Record*, Vol.32, No.3, pp.8–10 (2003).
11) Hiemstra, D., Hauff, C., de Jong, F. and Kraaij, W.: Sigir's 30th anniversary: An analysis of trends in ir research and the topology of its community, *SIGIR Forum*, Vol.41, No.2, pp.18–24 (2007).
12) Wang, D.: A graph-center-based scheme for energy-efficient data collection in wireless sensor networks, *MSN*, pp.579–587 (2006).
13) Reka, A. and Barabási, A.-L.: Statistical mechanics of complex networks, *Rev. Mod. Phys.*, Vol.74, pp.47–97 (2002).
14) Rapoport, A.: Spread of information through a population with socio-structural bias: I. assumption of transitivity, *Bulletin of Mathematical Biology*, Vol.15, No.4, pp.523–533 (1953).
15) Kossinets, G. and Watts, D.J.: Empirical analysis of an evolving social network, *Science*, Vol.311, 5757, pp.88–90 (2006).
16) Goldberg, A.V. and Harrelson, C.: Computing the shortest path: Search meets graph theory, *SODA*, pp.156–165 (2005).
17) Sun, J., Tao, D. and Faloutsos, C.: Beyond streams and graphs: Dynamic tensor analysis, *KDD*, pp.374–383 (2006).
18) Gibson, D., Kumar, R. and Tomkins, A.: Discovering large dense subgraphs in massive graphs, *VLDB*, pp.721–732 (2005).
19) Broder, A.Z., Glassman, S.C., Manasse, M.S. and Zweig, G.: Syntactic clustering of the web, *Computer Networks*, Vol.29, No.8-13, pp.1157–1166 (1997).
20) http://kdl.cs.umass.edu/data/canosleep/canosleep-info.html
21) http://snap.stanford.edu/data/soc-LiveJournal1.html
22) http://vlado.fmf.uni-lj.si/pub/networks/data/ND/NDnets.htm

(平成 22 年 3 月 15 日受付)
(平成 23 年 1 月 14 日採録)

藤原　靖宏（正会員）

2001 年早稲田大学理工学部電気電子情報工学科卒業．2003 年同大学大学院理工学研究科修士課程修了．2003 年日本電信電話株式会社入社．2008 年東京大学大学院情報理工学系研究科電子情報学専攻博士課程入学．DEWS2006 優秀論文賞，電子情報通信学会平成 19 年度論文賞，本会平成 19 年度論文賞，KDD 2008 best paper award runner-up 受賞．時系列データ処理およびグラフマイニングの研究開発に従事．電子情報通信学会，日本データベース学会各会員．

鬼塚　　真（正会員）

1991 年東京工業大学工学部情報工学科卒業．同年日本電信電話（株）入社．2000〜2001 年ワシントン州立大学客員研究員．現在，日本電信電話（株）サイバースペース研究所主幹研究員（特別研究員）．博士（工学）．これまでオブジェクトリレーショナルデータベース管理システム，XML データベース管理・ストリーム処理技術，大規模分散データ管理・マイニングに関する研究・開発に従事．2004 年度山下記念賞受賞．2008 年上林奨励賞．ACM，電子情報通信学会，日本データベース学会各会員．

喜連川　優（フェロー）

東京大学大学院工学系研究科情報工学専攻博士課程修了（1983 年），工学博士．東京大学生産技術研究所講師，助教授を経て，現在，同教授．東京大学地球観測データ統融合連携研究機構長，東京大学生産技術研究所戦略情報融合国際研究センター長．文部科学官．文部科学省「情報爆発」特定研究領域代表（2005〜2010 年），経済産業省「情報大航海プロジェクト」戦略会議委員長（2007〜2009 年），情報処理学会フェロー，副会長（2008〜2009 年），データベース工学の研究に従事．ACM SIGMOD Edgar F Codd Innovation Award 受賞．