情報処理学会論文誌　データベース　Vol.6　No.5　1–10 (Dec. 2013)

時々刻々と成長するグラフのための直径モニタリング

藤原 靖宏[1,a)]　鬼塚 真[1,b)]　喜連川 優[2,3,c)]

受付日 2013年6月21日, 採録日 2013年8月9日

概要：本論文では時々刻々と成長するグラフの直径を検出し続ける問題を対象とする．グラフの直径とはグラフにおいて最も長いノード間の距離である．この問題は P2P ネットワークにおいて検索性能を向上させるなど，様々なアプリケーションに応用することができる．提案手法は時々刻々と成長するグラフの直径を効率的かつ正確に検出し続けることができる．提案手法は，(1) 直径を構成しえないノードを枝刈りするアイデアと，(2) 新たにグラフに追加されたノードからの距離を用いて直径を構成するノードのペアを更新するアイデアから構成される．本論文では理論解析によって提案手法が正確に直径を検出し続けることができることを示す．また大規模なグラフを用いた実験により提案手法が従来用いられている手法より大幅に高速な検出が可能であることを示す．

キーワード：直径，グラフマイニング，検出，正確

Diameter Monitoring for Time-evolving Graphs

Yasuhiro Fujiwara[1,a)]　Makoto Onizuka[1,b)]　Masaru Kitsuregawa[2,3,c)]

Received: June 21, 2013, Accepted: August 9, 2013

Abstract: The goal of this work is to identify the diameter, the maximum distance between any two nodes, of a large graph that evolve over time. This problem is useful for many applications such as improving the quality of P2P networks. Our solution can detect the diameter of time-evolving graphs in the most efficient and correct manner. Our solution is based on two ideas: (1) It estimates the maximal distances to filter unlikely nodes that cannot be associated with the diameter, and (2) It maintains answer node pairs by exploiting the distances from a newly added node. Our theoretical analyses show that it guarantees exactness in identifying the diameter. We perform several experiments on real and large datasets that contain one million nodes and verify the efficiency of our solution.

Keywords: diameter, graph mining, monitoring, exact

1. まえがき

グラフは何らかのオブジェクト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．

グラフにおけるノード間の距離はグラフ理論の中で基本的な特徴量の１つである．グラフの**直径**とはノード間の距離に基づいた特徴量であり，グラフを解析するうえで非常に重要である．しかし従来のグラフ理論が対象としてきたグラフは静的であり，ノードおよびエッジの数は変わらないという前提があった．

近年，時間とともにノード数が時々刻々と増大し，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

[1] NTT ソフトウェアイノベーションセンタ
NTT Software Innovation Center, Musashino, Tokyo 180–0012, Japan
[2] 東京大学生産技術研究所
Institute of Industrial Science, The University of Tokyo, Meguro, Tokyo 153–8505, Japan
[3] 国立情報学研究所
National Institute of Informatics, Chiyoda, Tokyo 101–0003, Japan
[a)] fujiwara.yasuhiro@lab.ntt.co.jp
[b)] onizuka.makoto@lab.ntt.co.jp
[c)] kitsure@tkl.iis.u-tokyo.ac.jp

続けるグラフデータの特徴が明らかになりつつある [12]．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直径を検出し続ける問題を対象に正確かつ高速なアルゴリズムを提案する．従来手法では非常に大きな計算時間が必要であるため，大規模なグラフの直径を計算するのは困難であった．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直径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

定式的に本論文で扱う問題は以下のように表現される．

問題　（直径のモニタリング）

Given：　時刻 t における重みなし無向グラフ $G[t] = (V, E)$.

Find：　$G[t]$ における直径の値とそれを与えるノードのペア．

この問題は重みなし無向グラフを対象とするが，提案手法は重みありグラフに対しても有向グラフに対しても適用することができる．また提案手法はノードの追加のみでなくエッジの追加に対してもノード/エッジの削除に対しても適用することができる．

提案手法では莫大な計算量を減らすために，(1) ノードからその他のノードまでの最大距離を推定し，直径に関係ないノードを枝刈りし，(2) 直径となるノードのペアを新たに追加になったノードからの距離により更新する．提案手法は以下のような特徴がある．

- **効率的**：従来手法は $O(n^2 + nm)$ の計算時間（n と m はぞれぞれノードとエッジの数）が必要であり大規模なグラフの解析には実用的ではなかった．しかし提案手法は直径が縮まない場合 $O(n + m)$ の計算時間で，直径が縮む場合 $O(kn + km)$（ここで $k \ll n$）の計算時間で直径を求めることができ，従来手法より大幅に高速である．
- **正確**：提案手法では近似によって解になりえないノードを枝刈りするが，近似は解になるノードを枝刈りしないという理論的保証を持つ．そのため得られる解は正確である．
- **パラメータなし**：提案手法は自動的に必要となるパラメータを決定するため，ユーザはパラメータ調整する必要がない．

高速な検出を行うため提案手法はまず新たに追加したノードの距離を用いて解ノードのペアを更新する．もし追加したノードによって解ノードのペアがすべて無効になる場合は，ノードからの距離の最大値を推定し解ノードのペアの再計算を行う．結果的にノード間の距離の計算は最小限に抑えられるため，高速な検出が可能になる．実験を行ったところ，提案手法は従来手法より最大 990,000 倍以上高速であることが確認された．

1.2　応用例

近年 Gnutella，Seti@home，OceanStore など P2P サービスに対する注目が集まっている．P2P は中心にサーバを置くのではなくネットワークを介して計算リソースを共有するところに大きな特徴がある [1]．

P2P を用いたアプリケーションのなかでもインターネットを用いたコンテンツ配信は多くの研究者から注目される最も重要なものの 1 つである．たとえば Kazaa とその亜種のネットワークは急激に成長していて，450 万人以上のユーザが 7 ペタバイト以上のコンテンツを共有していることが知られている [2]．これらアプリケーションにおいて各コンピュータはデータを互いに転送し合いコンテンツ配信を行っている．

各コンピュータが直接通信するコンピュータの数はネットワークで用いられるプロトコルによって制限される．ネットワークを設計するとき最も重要な問題として，各コンピュータが直接通信するコンピュータの数を何台とするかというものがある [17]．この問題は 2 つの理由から重要である．1 つ目の理由としてネットワークに接続するコンピュータの数は膨大であり直接通信するコンピュータの数は非常に大きくなりうることがあげられる．また 2 つめの理由としてコンピュータ間のデータ転送はオーバヘッドであるため，効率的な配信のためには転送の数を減らす必要があることがあげられる．

ネットワークにおける直径は効率的なコンテンツ配信のための重要な指標である．それは直径がネットワークにおけるコンピュータ間の最大の転送回数と対応しているからである [9]．直径が大きくなったときにのみ各コンピュータが直接通信するコンピュータを増やすことによって効率的なコンテンツ配信が可能になる．

上記以外にネットワークの頑健性の向上も重要なアプリケーションとしてあげられる．Koppula らは動的に変化するグラフの直径をプロットすることにより，どのようにエッジを張り替えればネットワークの頑健性の向上させることができるかを示した [7]．さらに提案手法は鉄道ネットワークの頑健性の測定 [13]，インターネットの成長の監視 [11]，共著関係のネットワークの大きさの測定 [10] などのアプリケーションに用いることができる．

時々刻々と成長するグラフの直径を検出することは様々なアプリケーションに適用することができるが，それを求めるために莫大な計算時間が必要であるために実用的ではなかった．しかし高速かつ正確に直径を検出する手法を提供することによって，時々刻々と成長するグラフにおいて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 章で実験の結果を示す．7 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 章にあげた特性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 [12]．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 [19]．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 [16]．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 [8]．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ということがあげられる．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 [10]．

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．

Rattigan 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 [18]．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．

ランドマークによる手法は埋め込みによる手法の 1 つである [6], [15]．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 [14]．

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 [21]．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. 事前準備

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．

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 からはじまりノード 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．

グラフの直径は以下のようにノード間の最大の距離として定義される [4]．

定義 1（直径） グラフ $G = (V, E)$ においてノード間の最大の距離を $\max(d(u, v)|u, v \in V)$ としたとき，直径 D は以下のように定義される．

$$D = \max(d(u, v)|u, v \in V) \tag{1}$$

提案手法はグラフの直径のみでなく，直径となるノードのペアの集合 $\mathcal{D}$ も計算する．$\mathcal{D}$ は定式的に以下のように定義される．

定義 2（解ノードのペア） ノード間の距離が直径となるノードのペア $\mathcal{D}$ は以下のように定義される．

$$\mathcal{D} = \{(u, v)|d(u, v) = D\} \tag{2}$$

定義 2 に従い素直にグラフの直径を計算するには全ノードのペアの距離を計算する必要がある．これらの距離を求めるには，(1) 単一始点最短路問題による手法と，(2) 全点対最短路問題による手法がある．

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
D	グラフ G における直径
$\mathcal{D}$	直径となるノードペアの集合

単一始点最短路問題による手法とは，すべての n 個のノードごとに単一始点最短路問題における手法を適用するものである．すなわち重みなしグラフに対する単一始点最短路問題の解法は幅優先探索であるため，正確にグラフの直径を計算するにはすべてのノードに対して幅優先探索を用いればよい．幅優先探索を用いて直径を計算するためには幅優先探索には $O(n^2+nm)$ の計算量と $O(n+m)$ のメモリ量が必要となる．

また全点対最短路問題による手法は全ノードのペアの距離を $n \times n$ の大きさの行列積から求めるものである．行列積アルゴリズムとして最も有名なのはフロイド・ウォーシャルのアルゴリズムである [5]．このアルゴリズムの計算量は $O(n^3)$ である．これまでに行列積を用いて最短路を計算するアルゴリズムの改良について様々な研究がなされている．Fredma らは全点対最短路問題がエッジの重みの和を $O(n^{5/2})$ 回比較することによって解けることを示し，$O(n^3(\log\log n/\log n)^{1/3})$ の計算量で実行できるアルゴリズムを提案した [5]．また Roditty らは動的に変化するグラフにおいてならし計算量が $O(m\sqrt{n})$ となるアルゴリズムを提案した [20]．

全点対最短路問題による手法によって直径を求めるには上記の計算量のほかにさらに $O(n^2)$ の計算量が必要である．これは求められた n^2 個の距離から最も大きいものを求める処理がさらに必要があるからである．また $n \times n$ の大きさの行列を保持する必要があるため，全点対最短路問題による手法におけるメモリ量は $O(n^2)$ である．本研究では大規模なグラフを対象とするため，$O(n^2)$ のメモリが必要となる全点対最短路問題による手法を計算機で実行することは事実上不可能である[*1]．

そのため大規模なグラフの直径を求めるには単一始点最短路問題による手法を用いるほかないが，その計算量は $O(n^2+nm)$ であるため高速に実行することができない．また時々刻々と成長するグラフに対してはグラフが成長するためにこの処理を繰り返すこととなるために，幅優先探索により直径をノードを求めるのは現実的ではない．

4. 直径の検出

この章では提案手法における主要なアイデアを述べ，それらのアルゴリズムについて詳細な説明を行う．提案手法の大きな利点として時々刻々と成長するグラフに対して高速かつ正確に直径を求められることである．

4.1 手法の概要

提案手法は以下に述べる 2 つのアイデアによって構成される．

参照ノードによる枝刈り　はじめのアイデアは高い計算時間を減らすために直径になりえないノードを高速に枝刈りするものである．従来手法は n 個のノードすべてから幅優先探索を行うため非常に高い計算コストになる．提案手法におけるアイデアはすべてのノードから距離を計算する代わりに，選ばれたノードからのみ距離を計算するというシンプルなものである．このアイデアにおいて選択されたノードを**参照ノード**と呼ぶこととする．

検出処理において提案手法は 1 つ 1 つ参照ノードを選択する．参照ノードから他のノードへの最大距離を計算し，それを用いて参照ノードの周辺のノードの最大距離を推定する．この最大距離の推定値によってその周辺のノードが解ノードのペアとなりうるかを求める．すなわち参照ノードによってその周辺のノードが枝刈りできるかの判定を行う．それぞれの周辺のノードに対して推定は $O(1)$ の計算時間で行うことができる．そのため参照ノードの個数を k $(k \ll n)$ としたとき，$O(kn+km)$ の計算時間で直径を求めることができる．

この手法は 2 つの利点を有する．はじめの利点は周辺のノードを推定値によって枝刈りするが，正確に直径を求められることがあげられる．すなわちこの手法は少ない計算時間で解になりえないノードを安全に枝刈りできる．また参照ノードの個数 k は自動的に決定される．一般的にアルゴリズムのパラメータはその性能に大きく影響するためその決定は煩雑であるが，提案手法はパラメータを自動的に決定するためユーザに対してパラメータの決定を求めない．

逐次的更新　時々刻々と成長するグラフにノードが追加した後，直径は**大きくなる**か，**縮む**か，**変化しない**．ノードが追加した後に直径を効率的に更新する手法を提案する．

参照ノードによる枝刈りによって高速に直径を求めることができるが，グラフが成長する場合はそのたびに参照ノードによる再計算が必要になる．本手法はグラフが成長するたびに参照ノードによる枝刈りを避けるためのものである．後に述べるようにもし直径がノードの追加によって縮まなければ，直径は追加されたノードからの距離だけで更新することができる．

この手法は特に時々刻々と成長するグラフに対して有効である．それは追加されるノードは元々存在するノードに比べて少ないため，追加の前後で大きくグラフは変化しないためである．

4.2 参照ノードによる枝刈り

はじめのアイデアは参照ノードを選択し直径になりえないノードを枝刈りするものである．

参照ノードによる枝刈りは以下のように行う．(1) その他のノードまでの距離が長くなることが期待される候補距離を計算し，(2) その他のノードまでの距離が短くなることが期待される参照ノードを選択し，(3) 参照ノードの周辺ノードに対して，その他のノードまでの距離の最大値を

[*1] 100 万個のノードを有するグラフ場合，1 つの値を 4 バイトで保持するのであれば，必要なメモリは約 4 TB になる．

推定し，(4) 推定された最大値が候補距離より小さければそのノードを枝刈りする．

この章ではまずノードに対して最大の距離を推定する方法について述べ，その推定値が最大の距離の上限値となることを示す．そして参照ノードを選択する方法と候補距離を求める方法について述べる．

定式的にノード u から他のノードまでの距離の最大値を以下のように定義する．

定義 3（最大距離） グラフ G においてノード u からその他のノードまでの距離の最大値 $d_{max}(u)$ を以下のように定義する．

$$d_{max}(u) = \max(d(u,v)|v \in V) \tag{3}$$

最大距離の推定値は以下のように定義する．

定義 4（推定値） グラフ G のノード v に対して参照ノードを u_r としたとき，最大距離の推定値 $\hat{d}_{max}(v)$ を以下のように定義する．

$$\hat{d}_{max}(v) = d_{max}(u_r) + d(u_r, v) \tag{4}$$

推定値は以下に述べるように距離の最大値の上限値となる性質がある．

補助定理 1（上限値） グラフ G のすべてのノードに対して以下の性質が成り立つ．

$$d_{max}(v) \leq \hat{d}_{max}(v) \tag{5}$$

証明　ノード v からの距離が $d_{max}(v)$ であるノードを w とすると以下の式が成り立つ．

$$\begin{aligned} d_{max}(v) &= d(v,w) \leq d(u_r,w) + d(u_r,v) \text{ ([5] 参照)} \\ &\leq d_{max}(u_r) + d(u_r,v) \text{ (定義 3)} = \hat{d}_{max}(v) \end{aligned} \tag{6}$$

よって成り立つ．　□

提案手法は後に述べるようにこの性質を用いて正確な直径を求めることができる．

もし推定値が候補距離より短ければそのノードを枝刈りする．定義 4 に見られるように推定値は $O(1)$ の計算時間で求めることができるため，効率的に直径を計算することができる．

もし参照ノードの最大距離が長く，また候補距離が短ければ効果的に枝刈りすることができない．そのため効果的に枝刈りを行うためには参照ノードと候補距離の選択が重要である．提案手法では枝刈りされていないノードの中で最も次数の高いノードを参照ノードとして選択する．これはそのようなノードの最大距離が小さいことが期待されるからである．また提案手法では 1 時刻前の解ノードのペアから候補距離を計算する．これはノードが追加されてもグラフはほとんど変化しないため，これらのノードのペアは長い距離となることが期待されるからである．これらの選択により効果的な枝刈りが可能になることは 6 章に示す．

Algorithm 1 参照ノードによる枝刈り

Require: $G_t = (V, E)$，時刻 t におけるグラフ
　　$\mathcal{D}_{t-1}$，1 時刻前の解ノードのペア
Ensure: D_t，グラフ G_t の直径
　　$\mathcal{D}_t$，解ノードのペア

```
1: D_t := max(d(u,v)|(u,v) ∈ D_{t-1});
2: D_t := ∅;
3: V' := V;
4: while V' ≠ ∅ do
5:    u_r := argmax(deg(v)|v ∈ V');
6:    最大距離 d_max(u_r) を計算;
7:    if d_max(u_r) = D_t then
8:       D_t に {(u_r,v)|d(u_r,v) = D_t} を追加;
9:    end if
10:   if d_max(u_r) > D_t then
11:      D_t := d_max(u_r);
12:      D_t := {(u_r,v)|d(u_r,v) = D_t};
13:   end if
14:   V' から u_r を削除;
15:   V' から {v|d̂_max(v) < D_t} を削除;
16: end while
```

アルゴリズム 1 に参照ノードによる枝刈りの手法を示す．ここでノード u の次数を $deg(u)$ とする．アルゴリズムではまず 1 時刻前の解のペアを用いて候補距離を計算する（1 行目）．そして次数に基づいて参照ノードを求める（5 行目）．もし参照ノードの最大距離が候補距離以上であれば解ノードのペアを更新する（7〜13 行目）．アルゴリズムは候補距離を枝刈りのために用いる．すなわちもしあるノードにおける推定値が候補距離より小さければ，そのノードを枝刈りする（15 行目）．これらの処理はすべてのノードが処理されるまで繰り返される．このアルゴリズムにおいて参照ノードの数は自動的に決定される．また参照ノードを求めるのに必要な計算量は $O(kn)$ である．これは 5 行目に示すとおり $O(n)$ 個の枝刈りされていないノードから次数の最も大きいノードを選択することを k 回繰り返すからである．

4.3　逐次的更新

2 つめのアイデアは直径を高速に検出するために解ノードのペアを逐次的に更新するというものである．参照ノードによる枝刈りをグラフが成長するたびに行う必要がなくなるため，より効率的な検出が可能になる．

4.3.1　直径の変化

この項ではまずグラフが成長した後のノード間の距離の性質について述べる．そして直径の（1）大きくなる，(2) 縮む，(3) 変化しないことの必要十分条件を明らかにする．この項では 1 時刻ごとにノード u_a が新たに加わるとする．

また時刻 t におけるグラフを G_t，時刻 t において直径となるノードのペアの集合を $\mathcal{D}_t$ とする．

まずノードが追加された後，追加前に存在するノード間の距離は増えることがないことを示す．

補助定理 2（ノード追加後のノード間の距離） 時刻 t におけるノード間の距離は時刻 $t-1$ におけるグラフ G_{t-1} におけるノード間の距離より長くなることはない．

証明　もし時刻 t におけるノード u と v の最短パスが追加ノードを通るとすると，時刻 $t-1$ においてそれに対応するパスは存在しない．すなわち時刻 $t-1$ における最短パスは追加ノードによって短くなる．そうでなければ時刻 t において追加ノードを通らないノード u と v の最短パスが存在する．すなわち時刻 $t-1$ においても同じパスが存在する．結果ノード u と v の距離はノードの追加によって長くなることはない．□

ノードが追加した後直径は変化するが，上記の性質を用いてグラフの直径が変化することの条件を明らかにする．まずグラフの直径が大きくなることの条件を述べる．グラフの直径が大きくなることの必要十分条件は追加したノードの最大距離が 1 時刻前の直径より大きいことである．

補助定理 3（直径が大きくなることの必要十分条件） グラフの直径が大きくなることの必要十分条件は以下のとおりである．

$$d_{max}(u_a) > D_{t-1} \tag{7}$$

証明　もし $D_t > D_{t-1}$ であれば補助定理 2 より元から存在するすべてのノードの最大距離は D_{t-1} より長くなることはないため，ノード u_a の最大距離が直径となる．またもし $d_{max}(u_a) > D_{t-1}$ であれば明らかに $d_{max}(u_a) = D_t$ であり $D_t > D_{t-1}$ が成り立つ．□

グラフの直径が縮むことの必要十分条件は，追加ノードの最大距離が 1 時刻前の直径より短くかつノードの追加によって 1 時刻前のすべての解ノードのペアの距離が短くなることである．

補助定理 4（直径が縮むことの必要十分条件） グラフの直径が縮むことの必要十分条件は以下のとおりである．

$$\begin{aligned}&(1)\ d_{max}(u_a) < D_{t-1}, \text{かつ}\\ &(2)\ \forall (v,w) \in \mathcal{D}_{t-1}, d(v,w) < D_{t-1}\end{aligned} \tag{8}$$

証明　もし $D_t < D_{t-1}$ であれば定義 2 より $d_{max}(u_a) < D_{t-1}$ でありかつ 1 時刻前のすべての解ノードのペアの距離が D_{t-1} より短くなる．またもし式 (1) と (2) が成り立つのであれば定義 2 と補助定理 2 よりグラフの直径は時刻 t で短くなる．□

グラフの直径が変化しないことの必要十分条件は追加ノードの最大距離が 1 時刻前の直径と同じまたはノードの追加によって距離が縮まない 1 時刻前の解ノードのペアが存在することである．

補助定理 5（直径が変化しないことの必要十分条件） グラフの直径が変化しないことの必要十分条件は以下のとおりである．

$$\begin{aligned}&(1)\ d_{max}(u_a) = D_{t-1}, \text{または}\\ &(2)\ \exists (v,w) \in \mathcal{D}_{t-1}\ \ s.t.\ \ d(v,w) = D_{t-1}\end{aligned} \tag{9}$$

証明　補助定理 3 と 4 より明らか．□

4.3.2 検出処理

逐次的に解ノードのペアを更新することにより効率的に直径を検出できる．まず 1 時刻前の解ノードのペアの距離が短くなるのかの確認は追加ノードからの距離を用いて可能であることを示し，逐次的更新を用いた提案手法の詳細について述べる．

解ノードのペアを更新するため提案手法では最短パスにおける以下の性質を用いる [3]．

補助定理 6（Bellman criterion [3]） ノード u がノード v と w の最短パス上にあることの必要十分条件は以下のとおりである．

$$d(u,v) + d(u,w) = d(v,w) \tag{10}$$

定理 6 を用いてノードの追加によって 1 時刻前の解ノードのペアの距離が短くなるのかの確認ができる．

補助定理 7（距離の確認） 時々刻々と成長するグラフにおいてノード u_a が *1* 時刻前の解ノードのペア (v,w) の距離を縮めることの必要十分条件は以下のとおりである．

$$d(v,u_a) + d(w,u_a) < D_{t-1} \tag{11}$$

証明　もしノード u_a が距離を縮めるのであればノード u_a はノード v と w の最短パス上にある．そのため補助定理 6 より $D_{t-1} > d(v,w) = d(u_a,v) + d(u_a,w)$ が成り立つ．またもし $d(v,u_a) + d(w,u_a) < D_{t-1}$ であれば追加ノード u_a は $D_{t-1} > d(u_a,v) + d(u_a,w) \geq d(v,w)$ が成り立つため距離を縮める（文献 [5] 参照）．□

定理 7 よりもしグラフの直径が大きくなるか変わらない場合は解ノードのペアを追加ノードの距離から逐次的に更新することができる．すなわちもし追加ノードが直径となるのであれば追加ノードからの距離を用いて解ノードのペアを更新する．またもし追加ノードが 1 時刻前の解ノードのペアの距離を縮めるのであれば，そのことを補助定理 7 を用いて効率的に求める．またもし 1 時刻前の直径と同じ長さのノードのペアがなければ参照ノードによる枝刈りを用いて直径を求める．

アルゴリズム 2 に提案手法の詳細を示す．提案手法ではまず追加したノードの最大距離を求める（1 行目）．もしその最大距離が 1 時刻前の直径より大きければそれが新しい直径になるため（補助定理 3），直径と解ノードのペアを初

Algorithm 2 Proposed method

Require: $G_t = (V, E)$，時刻 t におけるグラフ
　　D_{t-1}，1 時刻前の直径
　　$\mathcal{D}_{t-1}$，1 時刻前の解ノードのペア
　　u_a，時刻 t における追加ノード
Ensure: D_t，グラフ G_t の直径
　　$\mathcal{D}_t$，解ノードのペア
1: 最大距離 $d_{max}(u_a)$ を計算;
2: //直径が大きくなる場合
3: **if** $d_{max}(u_a) > D_{t-1}$ **then**
4:　$D_t := d_{max}(u_a)$;
5:　$\mathcal{D}_t := \{(u_a, v) | d(u_a, v) = D_t\}$;
6: **else**
7:　//直径が変化しない場合
8:　$D_t := D_{t-1}$;
9:　$\mathcal{D}_t := \mathcal{D}_{t-1}$;
10:　**if** $d_{max}(u_a) = D_{t-1}$ **then**
11:　　$\mathcal{D}_t$ に $\{(u_a, v) | d(u_a, v) = D_{t-1}\}$ を追加;
12:　**end if**
13:　$\mathcal{D}_t$ から $\{(v, w) | d(v, u_a) + d(w, u_a) < D_{t-1}\}$ を削除;
14:　//直径が縮む場合
15:　**if** $\mathcal{D}_t = \emptyset$ **then**
16:　　アルゴリズム 1 を用いて D_t と $\mathcal{D}_t$ を計算;
17:　**end if**
18: **end if**

期化する（3〜5 行目）．もし求めた最大距離が 1 時刻前の直径と同じかまたは 1 時刻前の直径と同じ長さのノードのペアがあればノードが追加されても直径は変わらない（補助定理 5）．そのため提案手法は追加ノードからの距離を用いて解ノードのペアを更新する（8〜13 行目）．またもし 1 時刻前の直径と同じ距離のノードのペアがなければ直径は縮むため（補助定理 4），アルゴリズム 1 によって直径を求める（15〜17 行目）．

ノードの数を増やしながらグラフは成長していくが，簡単のため時刻 $t = 1$ においてノードは 1 つあるとする．そのため時刻 $t = 1$ において $D_{t-1} = 0$ であり $\mathcal{D}_{t-1} = \emptyset$ である．

今まで 1 つのノードが追加されることを前提に議論を進めてきたが，提案手法は複数のノードが追加されても対応することができる．またもしエッジが追加されたらそのエッジに接続されたノードが追加されたものとして処理を行う．またノード/エッジが削除された場合はアルゴリズム 1 によって直径を求める．

5. 理論的解析

この章では提案手法が正確に直径を検出できることとその計算量を示す．なお参照ノードの数を k とする．

5.1 正確性

まず提案手法が正確に直径を検出できることを示す．

補助定理 8（正確な検出） 提案手法は正確に直径を検出できる．

証明 時刻 t（≥ 1）で提案手法が正確な検出を行えることを数学的帰納法を用いて証明する．まず時刻 $t = 1$ で正確な検出を行えることを示す．時刻 $t = 1$ において提案手法は $D_1 = 0$ で $\mathcal{D}_1 = (u_1, u_1)$ であることを（1）$D_{t-1} = 0$ で $\mathcal{D}_{t-1} = \emptyset$ であり，（2）$d_{max}(u_1) = 0$ であるため求めることができる（アルゴリズム 2 の 8〜12 行目参照）．また時刻 $t = k$ において正確な検出が行えるとした場合，時刻 $t = k + 1$ でも正確な検出が行えることを示す．もし時刻 $t = k + 1$ で直径が縮まなければ補助定理 7 を用いて提案手法は正確な検出を行うことができる．またもしそうでなければ提案手法は参照ノードによる枝刈りにより直径を求める．ここで推定値が候補距離より小さければノードが枝刈りされるが，推定値は上限値になる性質がある（補助定理 1）．そのため直径となるノードが枝刈りされることはありえない．よって成り立つ．　□

5.2 計算量

ここではまず従来手法の計算量について述べてから，提案手法の計算量について述べる．

補助定理 9（従来手法 [5]） 従来手法で直径を求めるには $O(n + m)$ のメモリ量と $O(n^2 + nm)$ の計算時間が必要である．

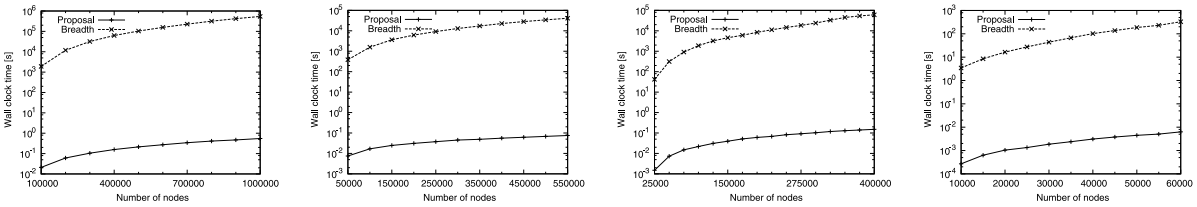
補助定理 10（提案手法のメモリ量） 提案手法は $O(n + m)$ のメモリ量が必要である．

証明 提案手法はグラフを保持するメモリ量と解ノードのペアを保持するメモリ量が必要であるが，解ノードのペアの数は 6 章に示すとおりノード/エッジの数に比較して非常に小さい．そのため提案手法は $O(n + m)$ のメモリ量が必要である．　□

補助定理 11（提案手法の計算時間） 提案手法においては直径が縮まない場合 $O(n + m)$ の計算時間が必要であり，直径が縮む場合 $O(kn + km)$ の計算時間が必要である．

証明 提案手法はまず追加されたノードからの距離を計算し直径が縮むかの確認を行う．このために必要な計算時間は $O(n + m)$ である．もし直径が縮む場合，提案手法は参照ノードによる枝刈りよって直径を求めるがこの場合の計算時間は $O(kn + km)$ である．　□

直径の検出における処理時間は参照ノードによる枝刈りと逐次的更新の効果に影響される．たとえば直径が縮む場合 $O(kn + km)$ の計算時間が必要であるが，参照ノードの数は実データの特性に影響される．そのため次の章では提案手法の有効性を実データで検証して確認する．



(1) Citation　(2) Web　(3) Amazon　(4) P2P

図 1　ノード数によると処理時間

Fig. 1　Wall clock time versus number of nodes.

6. 評価実験

提案手法の評価を行うために実験を行った．実験では幅優先探索による手法と network structure index [18] と比較を行った．network structure index はノード間の距離の近似計算を高速に行える手法である．

本実験では以下の 3 点を確認した．

- 高速性：提案手法は幅優先探索による手法より 5 桁以上高速であり，ノードの数が増えても高速に処理が可能である（6.1 節）．
- 各手法の有効性：提案手法で用いる参照ノードによる枝刈りや逐次的更新は直径を求めるのに効果的である（6.2 節）．
- 正確性：近似による手法と異なり，提案手法は直径を正確かつ高速に求めることができる（6.3 節）．

実験では公開されている実データ Citation，Web，Amazon，P2P を用いた．これらはそれぞれアメリカの特許の引用ネットワークのデータ，「berkely.edu」ドメインと「stanford.edu」ドメイン内のウェブデータ，アマゾンのウェブネットワークのデータ，Gnutella のネットワークのデータである．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 1 つ 1 つノードを加えていった．

実験は CPU が Intel Xeon Quad-Core 3.33 GHz，メモリが 32 GB の Linux サーバで行った．またすべてのアルゴリズムは GCC で実装した．

6.1　処理時間

提案手法と幅優先探索における手法とを比較した．実験では様々なノードの数に対する処理時間を調べた．**図 1** に結果を示す．

実験結果から提案手法は大幅に従来手法より高速であることが分かる．最大で提案手法は 990,000 倍以上高速であった．幅優先探索ではすべてのノードのペアに対して距離を計算する．しかし提案手法は直径が縮まなければ追加ノードからの距離のみで解ノードのペアを更新できる．直径が縮む場合は参照ノードからの距離を計算する必要があるが，実験結果にはほぼ影響がなかった．これはノードが追加されても直径が縮むことは少なく，また後に示すように参照ノードの数 k は非常に小さいからである．

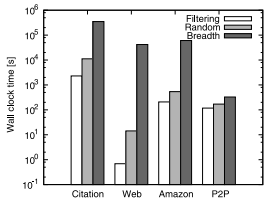


図 2　参照ノードによる枝刈り

Fig. 2　Reference node filtering.

6.2　各手法の有効性

以下の実験で提案手法で用いる 2 つの手法の有効性を示す．

6.2.1　参照ノードによる枝刈り

提案手法は参照ノードと距離候補を用いてノードを枝刈りする．4.2 節で述べたとおり提案手法は次数の高いノードを参照ノードとして選択し，1 時刻前の解ノードのペアから候補距離を求める．この手法の有効性を調べるため参照ノードによる枝刈りのみで直径を検出するために必要な処理時間を調べた．

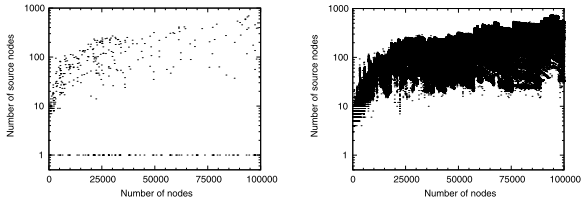
結果を**図 2** に示す．この図において参照ノードによる枝刈りのみの結果を Filtering，またランダムに参照ノードと候補距離を求めた結果を Random，幅優先探索の結果を Breadth とする．この図においてノード数は Citation で 1,000,000，Web で 550,000，Amazon で 400,000，P2P で 60,000 である．

実験結果から Filtering は Random より 20 倍以上高速であることが分かる．これは次数の高いノードほど最大距離が小さい傾向があり，また 1 時刻前の解ノードのペアの距離はノードの追加後も長い傾向があるからである．結果として参照ノードによる枝刈りにより直径を効率的に検出できる．

また 4.3.2 項で述べたとおりノード/エッジが削除された場合は参照ノードによる枝刈りにより直径を求めるが，この図の Filtering と Breadth を比較することによりこの方法が有効であることが分かる．

6.2.2　逐次的更新

逐次的に解ノードのペアを更新することでより効率的に直径を検出することができる．この手法の有効性を示すために距離計算を行ったノードの数を調べた．なお距離計算を行ったノードの数は追加ノードと参照ノードの数の和



(1) Proposal　(2) Filtering

図 3　距離を計算したノード数

Fig. 3　Number of distances computations nodes.

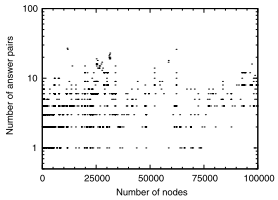


図 4　解ノードのペアの数

Fig. 4　Number of answer pairs.

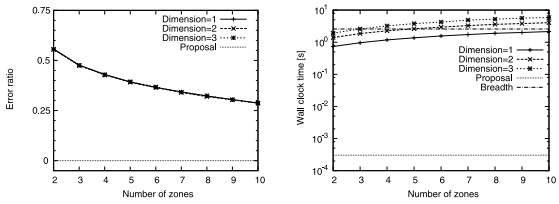
$(k+1)$ になる．**図 3** に結果を示す．この図において逐次的更新を行わずに枝刈りのみ行う手法の結果は Filtering である．また**図 4** に解ノードのペアの数を示す．なお実験データは Citation である．

図 3 から距離計算を行うノードの数を劇的に少なくできることが分かる．すなわち参照ノードの数 k はグラフのノード数より大幅に少ないが，ほとんどの場合提案手法は追加ノードからの距離を計算することで直径を検出できることが分かる．さらに図 4 から解ノードのペアの数は非常に小さいことが分かる．多くの場合解ノードのペアの数は 10 以下であり，実験において 30 より大きくなることはなかった．このため提案手法は直径を効率的に検出することができる．

6.3　検出の正確性

提案手法の利点の 1 つとして正確に直径を検出できることがあげられる．しかしながら，(1) 従来の近似手法はどれほどの精度で直径を計算できるのか，(2) 提案手法は従来の近似手法より高速に直径を計算できるのかということを検証する必要がある．

これらを検証するため Rattigan らが提案した network structure index との比較を行った．彼らは距離の近似手法としていくつか提案したが，本実験では distant to zone という手法と比較した．この手法は 2 章で示した手法も含めて他の手法よりより良い結果を示したからである．なお同様の結果は彼らの論文でも報告されている．network structure index は zones と dimensions の 2 つのパラメータを有する．zones とはグラフを領域に分割したときの分割数であり，dimensions とは zones の集合である．実験では手法の正確性をエラー率で評価した．ここでエラー率とは求められた直径の大きさの誤差を正確な直径の大きさで



(1) Error ratio　(2) Wall clock time

図 5　network structure index との比較

Fig. 5　Comparison to the network structure index.

割ったものである．エラー率は 0 から 1 の値になる．**図 5** にエラー率と処理時間の結果を示す．なお実験データは Citation であり，ノード数は 10,000 である．

図から分かるとおり提案手法のエラー率は 0 である．これは提案手法が正確に直径を検出できるからである．しかし network structure index においてエラー率は非常に高い結果となった．そのため network structure index で直径を求めることは実用的ではない．また図 5 から提案手法の処理時間は非常に少ないことが分かる．提案手法は network structure index より 2,400 倍以上高速であった．また network structure index においてはパラメータによって大きく異なり，エラー率と処理時間の間にトレードオフがあることが分かる．すなわちパラメータの値が小さくなるほど，処理時間も減るがエラー率は高くなる結果となった．提案手法も推定によりノードの枝刈りを行うが，network structure index と異なり直径となるノードを枝刈りすることはない．そのため提案手法は解の正確性と処理時間を両立させることができる．

7.　まとめ

本論文では時々刻々と成長するグラフの直径を検出する問題を扱った．提案手法は我々の知る限り正確に直径を検出し続ける初めてのものである．提案手法では，(1) 参照ノードにより直径になりえないノードを枝刈りし，(2) 追加ノードからの距離を用いて解ノードのペアの候補を更新する．実験を行ったところ提案手法は従来手法より 990,000 倍以上高速であることが分かった．

参考文献

[1] Androutsellis-Theotokis, S. and Spinellis, D.: A Survey of Peer-to-peer Content Distribution Technologies, *ACM Comput. Surv.*, Vol.36, No.4, pp.335–371 (2004).

[2] Bawa, M., Cooper, B.F., Crespo, A., Daswani, N., Ganesan, P., Garcia-Molina, H., Kamvar, S.D., Marti, S., Schlosser, M.T., Sun, Q., Vinograd, P. and Yang, B.: Peer-to-peer Research at Stanford, *SIGMOD Record*, Vol.32, No.3, pp.23–28 (2003).

[3] Brandes, U.: A Faster Algorithm for Betweenness Centrality, *Journal of Mathematical Sociology*, Vol.25, pp.163–177 (2001).

[4] Brandes, U. and Erlebach, T.: *Network Analysis: Methodological Foundations*, Springer (2008).

[5] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C.: *Introduction to Algorithms*, The MIT Press (2009).
[6] Goldberg, A.V. and Harrelson, C.: Computing the Shortest Path: Search Meets Graph Theory, *SODA*, pp.156–165 (2005).
[7] Koppula, H.S., Puspesh, K. and Ganguly, N.: Study and Improvement of Robustness of Overlay Networks, *HiPC*, pp.1–5 (2007).
[8] Kossinets, G. and Watts, D.J.: Empirical Analysis of an Evolving Social Network, *Science*, Vol.311, No.5757, pp.88–90 (2006).
[9] Kumar, A., Merugu, S., Xu, J. and Yu, X.: Ulysses: A Robust, Low-Diameter, Low-Latency Peer-to-Peer Network, *ICNP*, pp.258–267 (2003).
[10] Leskovec, J., Kleinberg, J.M. and Faloutsos, C.: Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations, *KDD*, pp.177–187 (2005).
[11] Magoni, D. and Pansiot, J.-J.: Analysis of the Autonomous System Network Topology, *Computer Communication Review*, Vol.31, No.3, pp.26–37 (2001).
[12] Newman: The Structure and Function of Complex Networks, *SIREV: SIAM Review*, Vol.45, pp.167–256 (2003).
[13] Ng, A.K.S. and Efstathiou, J.: Structural Robustness of Complex Networks, *NetSci*, pp.1–2 (2006).
[14] Ng, T.S.E. and Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches, *INFOCOM*, pp.170–179 (2002).
[15] Potamias, M., Bonchi, F., Castillo, C. and Gionis, A.: Fast Shortest Path Distance Estimation in Large Networks, *CIKM*, pp.867–876 (2009).
[16] 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).
[17] Ratnasamy, S., Stoica, I. and Shenker, S.: Routing Algorithms for DHTs: Some Open Questions, *IPTPS*, pp.45–52 (2002).
[18] Rattigan, M.J., Maier, M. and Jensen, D.: Using Structure Indices for Efficient Approximation of Network Properties, *KDD*, pp.357–366 (2006).
[19] Reka, A. and Barabási: Statistical Mechanics of Complex Networks, *Rev. Mod. Phys.*, Vol.74, pp.47–97 (2002).
[20] Roditty, L. and Zwick, U.: On Dnamic Shortest Paths Problems, *ESA* (2004).
[21] Sun, J., Tao, D. and Faloutsos, C.: Beyond Streams and Graphs: Dynamic Tensor Analysis, *KDD*, pp.374–383 (2006).

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 年早稲田大学大学院理工学研究科電気工学専攻修士課程修了．2003 年日本電信電話株式会社入社．2011 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修了，現在，NTT ソフトウェアイノベーションセンタ研究主任．博士（情報理工学）．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．本会平成 2007 年度および 2012 年度論文賞，第 27 回テレコムシステム技術賞，第 9 回上林奨励賞等受賞，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 NTT 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）ソフトウェアイノベーションセンタ特別研究員，電気通信大学客員教授，博士（工学）．大規模グラフデータのデータ処理に関する研究開発に取り組んでいる．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博士課程修了（1983 年）工学博士．現在，国立情報学研究所長，東大生産技術研究所教授，東大地球観測データ統融合連携研究機構長，文科省「情報爆発」特定研究領域代表，経済産業省「情報大航海」戦略会議委員長．データベース工学が専門．ACM SIGMOD Edgar F Codd Innovation Award 受賞，ACM/IEEE フェロー，本会/電子情報通信学会フェロー，本会会長，内閣府最先端研究開発支援プログラムを推進．

（担当編集委員　伊藤 貴之）