# 部分グラフ検索高速化のためのハブ節点を考慮した関係辺の再帰走査法

楠　　和馬†　　波多野賢治††

† 同志社大学大学院文化情報学研究科　〒 610–0394 京都府京田辺市多々羅都谷 1–3
†† 同志社大学文化情報学部　〒 610–0394 京都府京田辺市多々羅都谷 1–3
E-mail: †{kusu,hatano}@mil.doshisha.ac.jp

**あらまし**　柔軟な表現力の高さからさまざまな分野において，データがグラフ構造で表現されている．グラフデータベースには，膨大で複雑なグラフを一定コストでグラフ走査を行えるインデックスフリー隣接性という特性がある．しかし，同一関係性を再帰的に辿るような問合せでは，グラフ全体を走査することになり同一の節点に到達することが多くなる．また，辺の次数が多いハブ節点の影響を受けることで，辺の走査を必要とする節点候補が増加し，問合せ結果の導出コストが高くなる非効率性がある．そこで本研究では，次数が外れ値的に多いようなハブ節点の存在を考慮し，ハブ節点へのパスおよびハブ節点からの再帰的な同一関係性の経路情報をインデックスフリー隣接性に拡張することで効率的なグラフ走査方法を提案する．

**キーワード**　グラフデータベース，再帰経路走査，経路インデクス

## 1　はじめに

さまざまな分野において実体間の関係性に着目したデータ問合せや分析が行われている．そのような実体間の関係性を取り扱う実世界におけるデータ例として，ソーシャルネットワーキングサービス (SNS) における人間同士の友好関係を表すソーシャルネットワーク，研究者の共著関係や論文の引用関係，メーリングリストの投稿，タンパク質の構造などがある [1]．このようなネットワーク構造をそのままデータ表現可能な構造にグラフがあり，これは実体を表現する節点（点，node, vertex）と関係性を表現する辺（線，edge, link）で構成される．基本的なグラフの構成要素としては上記の二つであるが，アプリケーションの目的に応じてグラフの諸要素が持つ属性情報やそれらの種類を管理できるようなこれまで多様な形状のグラフが提案されてきた [2]．

数十年にも渡り研究されて今や世の中に普及しているデータベースシステムにリレーショナルデータベース (RDB) が存在し，リレーションという表形式のデータモデルでグラフの管理方法が確立されてきた [3]．しかし，実体間の関係性を辿ること（グラフ走査）で必要な部分グラフを抽出し分析を行うといったグラフデータの処理特性上，RDB 上での管理ではリレーション同士の自己結合が発生するため高コストになる．こういったグラフに対する問合せの特性を考慮して，汎用的にグラフデータを管理でき，かつ膨大なグラフに対しても一つの関係性を一定コストで辿ることが可能なネイティブ・グラフデータベース (GDB) が多数開発され研究されている [1] [4]．グラフ走査の高速化は，データベースから部分グラフを抽出することや特定の開始節点から対話的に問い合わせることの高速化につながる．また基本的に問合せはグラフ走査による部分グラフ抽出のようなトランザクション処理 (Online Transaction Processing: OLTP) 後に集約演算やグラフ分析のような分析処理 (Online Analysis Processing: OLAP) が実行されるため，分析を実行するためにも高速に部分グラフの問合せを完了させる必要がある．ネイティブ GDB は各節点が隣接している節点へのポインタを持っているという特性（インデックスフリー隣接性）を持ち合わせており，この特性がデータベースに格納されているデータ量に影響を受けず，一定の計算コストによりグラフ走査の実行を可能にしている．

しかし，ネイティブ GDB のインデックスフリー隣接性ではグラフがどのような形状か把握できないため，どのような節点に到達するかグラフ走査中に知ることができない．グラフ走査において避けるべき処理としては，同じ節点を何度も処理することや次数の多い節点に対する処理は辿るべき辺数が多くなるためコストが高くなる．次数が外れ値的に多い節点のことをネットワーク科学ではハブ (*Hub*) と呼ばれており [1]，そのような節点に対して何度も処理の対象にすることは避けるべきである．ネットワーク科学では現実におけるグラフの次数分布は冪乗則に従うとされており全節点内でハブに値する節点は極少数ではあるが，現実ネットワークの諸性質から再帰的にグラフ走査を行うほどハブに到達する可能性が徐々に高くなる．また，グラフ問合せにおいては関係性指向の問合せ性質上，同一関係性を再帰的に辿ることはあらゆる分野のデータにおいて行われる．そのため，グラフ問合せにおいてはハブから受けるグラフ走査性能への影響が大きくなることが容易に予測できる．

そこで本研究では，グラフ内に存在するハブとその他節点との位置関係を管理することにより，グラフ走査中に何度もハブ節点を再読み込みする必要の無いグラフ管理方法を提案する．また，グラフ走査の計算コストを最小限にすることに有効なインデックスフリー隣接性の隣接節点情報と，本研究で追加するハブとその他節点の位置情報を使い分けることによりグラフ走査を実現する．さらに，本研究の提案手法とグラフ走査性能

1 : DB-Engines: “DB-Engines Ranking of Graph DBMS.” https://db-engines.com/en/ranking/graph+dbms （2020 年 3 月 19 日 閲覧.）

が高いネイティブ GDB との比較により，同一関係性の再帰的なグラフ走査性能に対して評価実験を行い，本研究の有効性を示す．

2 節では，既存のネイティブ GDB で採用されているインデックスフリー隣接性の管理方法について，グラフ走査の性能向上を目的とした研究について紹介する．3 節では，ハブを考慮したグラフの管理方法およびグラフ走査方法について説明する．また，インデックスフリー隣接性の隣接節点情報との併用をどのように行うかについて，その方法も提案する．4 節では，本研究で提案したグラフ走査方法により，同一関係性を再帰的に辿るグラフ走査の性能について評価を行い，本研究の有効性について評価する．

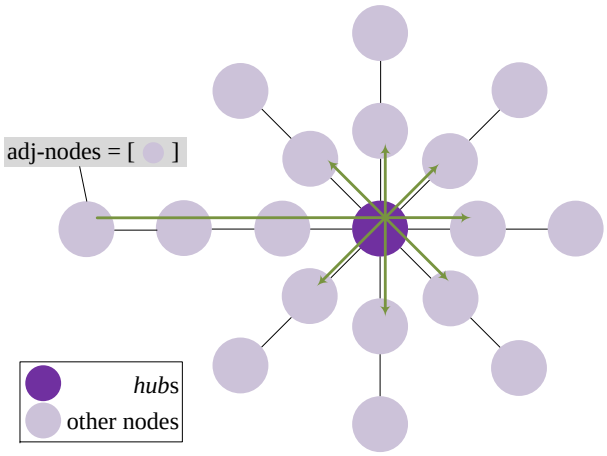


図 1　既存 GDB のグラフ走査

## 2　前 提 知 識

節点間の辺を効率的に辿ることは部分グラフ抽出の高速化に繋がるため，グラフの管理方法やインデックス構築手法などのさまざまな方法で経路探索の効率化を目的とした研究は多く存在する．その中でも本研究に大きく関わるインデックスフリー隣接性を持つグラフ管理方法についてと，特定パタンに対する経路インデックスの構築方法について紹介する．

2.1 項では，本研究で密接に関わるネイティブ GDB がどういった管理方法であるのか，ネイティブ GDB が持つ特性であるインデックスフリー隣接性のグラフ管理方法について詳説する．2.2 項では，経路の問合せ高速化に向けて提案された経路インデックスについて紹介する．最後に 2.3 項では，既存のグラフ走査効率化に関する手法の特徴を鑑みて，本研究におけるグラフ管理方法の目的について明記する．

### 2.1　インデックスフリー隣接性

現在ではグラフデータの管理に需要が高まってきたことからさまざまな GDB が開発されている．RDB 上においてグラフのデータ表現は隣接節点リストを管理する方法が提案されており，その方法を GDB のデータ管理方法に採用した GDB が存在する．このような管理方法では，節点同士の接続状態にあるか検索を効率化するために，隣接節点に対してインデックスを作成して検索しグラフ走査を実現している．他にもグラフに対するインデックス単位はいくつか提案されている一方で，隣接節点の接続状況を各節点が保持することでグラフを管理する方法がある．この管理方法はグラフの形状を保存し，データベースに格納された関係性に基づいてデータ問合せを得意とすることから，ネイティブグラフストレージと呼ばれている．また，ネイティブグラフストレージを採用する GDB のことをネイティブ GDB と呼称している [5]．

ネイティブ GDB の特性であるインデックスフリー隣接性は，効率的に隣接節点へアクセスできるようにストレージの格納位置（ポインタ）を各節点に保持させている．グラフ問合せにおいて，グラフ走査の開始となる節点をデータベースから読み出し後，それら節点からクエリ内で指定された関係性を割り当てられている辺を辿ることで，効率的に隣接節点を次々と読み出すことで，クエリで指定された部分グラフを導出している．つまりネイティブ GDB において，隣接節点情報を各節点が保持していることからインデックス計算時間は生じず，その節点の次数に依存した探索コストでグラフ走査を実現している．そのため，ある節点からのグラフ走査にかかる処理コストはデータ量の影響を受けにくく，膨大なグラフにおいても一定のコストで問合せが可能になる [6]．

ただ，各節点に保持させているのは隣接節点のリストのみであることから，隣接していない節点の接続状況は把握できていない．また，愚直に隣接節点リストに基づいて図 1 のようにデータを次々と問い合わせるため，ハブのような次数が多い節点に対しても同じように隣接節点リストに従ってグラフ走査をせざる負えない．

以上で述べた通り，インデックスフリー隣接性は局所的な問合せに非常に強力な特性といえるが，どのような節点に到達するかは到達するまで不明なため，節点群を起点とした大域的なグラフ走査の場合に機能しない特性といえる．

### 2.2　経路インデックス

逐次的に節点を辿る方法ではなく，アプリケーションの機能を実装する上でよく問い合わせられる関係性の組合せを事前に導出してインデックスの構築を方法も研究されている [7–10]．GDB のデータ管理事例として，巨大な一つのグラフを管理する場合と膨大な数の小さなグラフを管理する場合の 2 種類がある [7]．巨大な一つのグラフを管理する場合に該当する現実のグラフには，ソーシャルネットワークや Web グラフ，研究者の共著関係および引用関係などが挙げられる．一方，膨大な数の小さなグラフを管理する場合に該当する現実のグラフには，化学構造，タンパク質構造などが挙げられる．

膨大な数の節点と辺で構成される一つのグラフを管理する場合，節点が複雑に関係し合っているためグラフ内の経路候補が膨大な数になる傾向にある．それに伴い，そのグラフに対して構築したインデックスのデータ量も膨大になり，問合せ対象となる経路の導出コストはデータ量に依存することになる．

### 2.3 本研究の目的

現実のグラフには，次数分布が冪乗則に従うというスケールフリー性や 6 次の隔たりと呼ばれるスモールワールド性といった特性が存在する [11]．既存のネイティブ GDB ではこういった現実のグラフが持つ諸特性が考慮されていないため，他の節点よりも次数が桁外れに多いハブに対しても同様の方法でグラフ走査が行われる．また，次数が大きければグラフ走査のコストはその分だけ大きくなるが，ハブか否か関係無く節点のキャッシングを行い，グラフ走査を実現している現状にある．

そこで本研究では，グラフ問合せにおいて同一関係性を再帰的に辿る際にグラフのデータ量に影響を受けにくく，現実のグラフの特性を加味したグラフ管理を目的とする．したがって，データ量の影響を受けにくいインデックスフリー隣接性におけるグラフ管理方法に拡張する形で，効率的に再帰的なグラフ走査が可能なグラフ管理方法について提案する．

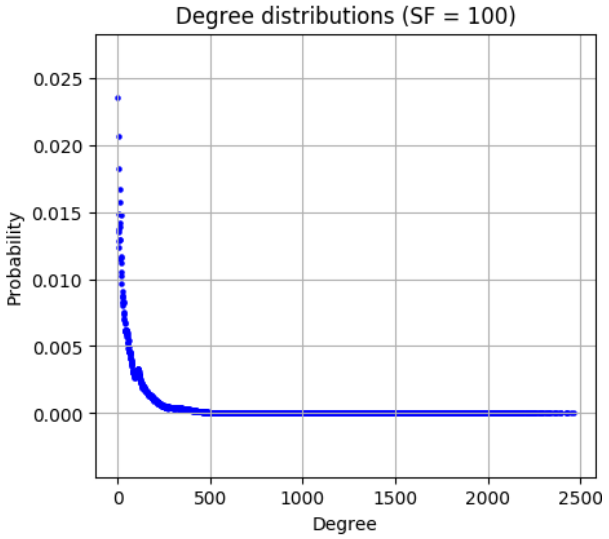


図 2　次数分布の例

## 3 提 案 手 法

実社会で発生しうるグラフデータは大抵の場合が関係の密度が疎で，外れ値的に次数が多いようなハブの数は少数とされている [1] つまり，人間の友好関係のような実社会で生じる現象を形成したグラフはその次数分布が冪分布になる傾向にある．したがって，そういったグラフを管理する汎用的な GDB において，隣接節点の情報のみだけではなくハブの存在を考慮することが妥当である．

そこで本研究では，グラフ内に存在するハブとそれ以外の節点間の距離やそれまでの経路を隣接節点情報に加えて管理方法を提案する．また，グラフ走査時にその情報を利用することによりインデックスフリー隣接性の特性上避けるべきハブに対する処理を軽減することによりグラフ走査を効率化する方法についても提案する．

ハブの再読み込みを抑制するために，どの節点がハブであるかを決定して格納位置を把握しておく必要がある．したがって 3.1 項では，GDB に格納されたグラフデータの中でどの節点がハブであるか決定する方法について説明する．次に 3.3 項では，各節点からハブへの経路を各節点に保持させることにより問合せ時にハブの読込みを行わず，問合せ内の再帰的経路の導出を効率化する方法について説明する．さらに 3.2 項では，他の節点より次数が多いという特徴を活かし，そこからの再帰経路を管理することで問合せ時にハブからの再帰経路を導出する方法について説明する．最後に 3.4 項では，3.3 項および 3.2 項で保存しているグラフ要素を利用して再帰経路の導出方法について説明する．

### 3.1 ハブの決定方法

本研究で提案するグラフ管理方法において，基準となるハブ群を決定する方法について説明する．まずは，単一の関係性に対する再帰的なグラフ走査を効率化するため，対象となる関係性を一つ選択する．次に，該当する関係性が割り当てられた辺を GDB から取得し，それら辺に接続されている節点群を取得する．さらに，各節点の次数を計算し，図 2 に示すような対象の関係性に対応する次数分布を得る．図 2 の横軸は，次数 (Degree) の値であり，縦軸は各次数における確率 (Probability) であり，節点群の中でその次数の値である節点の割合である．したがって，横軸の右側に行けば行くほど次数が大きくなり，本研究においてハブとして扱う節点群である．この次数分布は，4.1 項で説明するベンチマークのデータセットを利用して作成している．

現実のネットワークにおいて，図 2 に示すような冪乗則に従う次数分布が得られる．このような冪乗則に従う場合は，パレートの法則（80/20 の法則とも呼ぶ）[12] に基づいて分析対象のセグメンテーションが行われる．また，ネットワーク科学においても現実ネットワークに対してパレートの法則が存在することは報告されている [13]．

したがって本研究において，節点群の中から確率の多い方から 8 割を普遍的な次数を持つ節点とし，それ以降の確率が極めて低い全節点群の 2 割をハブとして扱う．

### 3.2 ハブにおける再帰経路の管理方法

対象の関係性による再帰経路を管理する方法として，3.1 項で決定したハブを開始点とする再帰経路を事前に取得し，パス長毎に再帰経路を管理する．ハブからの再帰経路を事前に収集しておくことにより，問合せ中に複数の節点からグラフ走査を行う場合に処理コストが高くなるハブを何度も読み込むことを避けることができる．

図 3 に示す通り，ハブとして扱っている節点から再帰的経路長 $rp$ 毎にその経路で到達する節点集合を識別できるように管理する．ハブから対象の関係性による再帰的なグラフ走査を行ったときに到達する節点群を取得できるように，経路の管理方法としてはグラフ走査の開始点となるハブと，対象の関係性を何回辿るかというパス長のペアで再帰経路を保存する．これを各ハブで事前に収集しておくことによって，ハブと残り何回のグラフ走査を実施するかが定まれば，部分グラフの導出を実

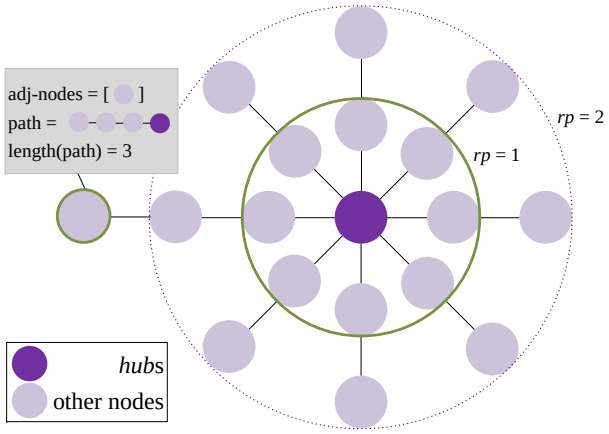


図 3　本研究のグラフ走査

行することができる．

### 3.3　ハブへの経路管理方法

ネイティブ GDB のインデックスフリー隣接性は各節点がその隣接節点のリストのみを保持しており，逐次的に辺を辿ることでグラフ走査をせざる負えず，その際にハブを辿る可能性もあった．そこで本研究で提案する手法では，各節点から各ハブまでの最短経路を保持しておくことにより，再帰的なグラフ走査において何度もハブを読み込むことを避けることができる．

各節点から各ハブまでの最短経路の導出方法には，任意の二点間における最短経路およびその経路長を導出する一般的な手法であるダイクストラ法によって導出する．実装は GDB にハブ以外の節点群と各ハブ群をそれぞれ始点と終点のペアを作成し，そのペアにおける最短経路および経路長を各節点に記録して問合せ時に利用する．

### 3.4　再帰経路探索の実現方法

ここまででハブに対する再帰的経路の管理方法とハブ以外の各節点に対するハブまでの最短経路の管理方法を説明した．各節点からの再帰的なグラフ走査を実現するために，各節点からハブまでの経路とハブからの再帰的な経路を利用する．

再帰的経路の探索手順としては次の通りである．

（1）　グラフ走査の開始節点を取得
（2）　問合せ内の指定再帰回数内にハブが存在するか確認
（3）　ハブまでの最短経路と経路長を取得
（4）　ハブまでの経路長と問合せの再帰回数の差分を算出
（5）　差分の値を再帰経路長 $rp$ に設定
（6）　ハブと再帰経路長 $rp$ のペアで再帰経路を取得
（7）　ハブに届かない場合は隣接節点に基づきグラフ走査

以上のように，インデックスフリー隣接性によるグラフ走査と本研究で拡張したハブを起点とする再帰経路を基に，同一関係性による再帰的なグラフ走査を実現する．これにより，グラフ走査の開始節点が複数の場合，本来であればハブに到達するたび隣接節点に基づきグラフ走査を行っていたが，一度収集したハブからの再帰経路を再利用することが可能になる．

## 4　評 価 実 験

本研究で提案したグラフ管理方法によって，同一関係性の再帰的なグラフ走査の効率化にどの程度貢献できているか評価するため，既存のネイティブ GDB と比較して性能評価を行う．比較対象とするネイティブ GDB には，広く利用され [2]グラフ走査が高速である [14] Neo4j[3] を利用する．また，実験環境は表 1 の通りである．

表 1　実 験 環 境

| 対象 | スペック，バージョン |
|---|---|
| OS | CentOS 7.6.1801 (x86 64bit) |
| CPU | Intel Xeon Silver 4114 CPU @ 2.20GHz |
| RAM | 256 GB |
| Neo4j | ver. 3.4.7 |

上記の目的で評価実験を行うために，GDB の性能を測るために開発されたベンチマークを用いる．そのため 4.1 項では，評価実験で利用するベンチマークに関する説明を行う．また 4.2 項では，本研究で提案したグラフ管理方法が，同一関係性を再帰的にグラフ走査を行う問合せにおいて処理の効率化が実現できたか評価する方法について説明する．

### 4.1　GDB ベンチマーク

実運用可能な GDB が次々と開発されて行く上で，どの GDB がどういった問合せにおいて性能が優れているのか，またデータ量が膨大であっても安定した性能を発揮できるか検証することに需要が出てきている．そういった検証を可能にするために，非営利団体である Linked Data Benchmark Council (LDBC)[4] は現実に存在する SNS におけるデータ管理を模した Social Network Benchmark (SNB) を開発している [5]．

LDBC SNB のデータセット生成プログラム [6]によって出力されるグラフスキーマを図 4 に示す．これは LDBC SNB のマニュアル [15] に掲載されており，各ラベルの節点がどの程度生成されるか，関係性の生成方法，グラフ要素へどのように属性を付与しているかに関しても記載されている．また，さまざまなデータ量における問合せ性能の検証を行えるようにするために，Scale Factor (SF) の値を変更することによってデータ量を制御することが可能になっている．さらに LDBC SNB には，Interactive Workload [16] と Business Intelligence Workload [17] と名称付けられている 2 種類のワークロード

2 : DB-Engines: “DB-Engines Ranking of Graph DBMS.” https://db-engines.com/en/ranking/graph+dbms （2020 年 3 月 19 日 閲覧.）
3 : Neo4j, Inc.: Neo4j HP. https://neo4j.com/ （2020 年 3 月 19 日 閲覧.）
4 : LDBC: “Social Network Benchmark” http://ldbcouncil.org （2020 年 3 月 19 日 閲覧.）
5 : LDBC: “Social Network Benchmark” http://ldbcouncil.org/benchmarks/snb （2020 年 3 月 19 日 閲覧.）
6 : LDBC: ldbc_snb_datagen.git. https://github.com/ldbc/ldbc_snb_datagen （2020 年 3 月 19 日 閲覧.）

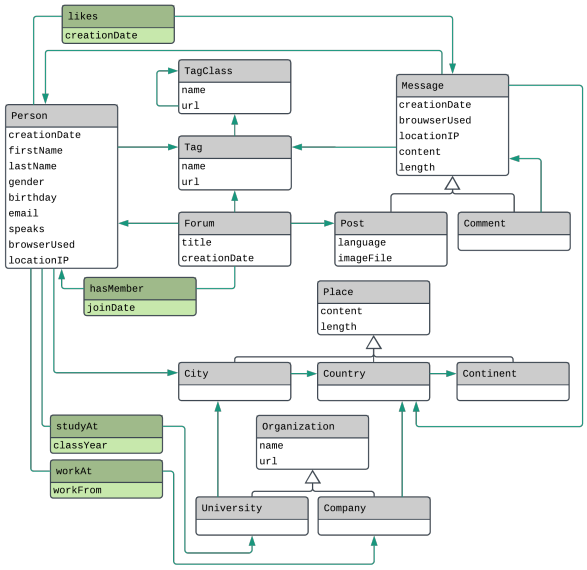


図 4　LDBC SNB データセットのグラフスキーマ

が用意されている．Interactive Workload は，サービス上で実際に利用されることを想定された問合せおよびデータ更新に関するトランザクション処理に関するクエリがまとめられている．その中には，数回のグラフ走査を行う Interactive Short クエリが 7 種類，再帰的なグラフ走査や属性条件を含んでいる Interactive Complex クエリが 14 種類，データの更新を行うクエリが 8 種類ある．Business Intelligence Workload は，サービスの運営に役立つ情報を得る目的で行うようなデータ分析に関するクエリセットであり，部分グラフの集計や集約演算など行うような分析的なグラフ問合せが 25 種類ある．

本研究では簡易な性能評価の実験として，LDBC SNB のデータセット生成プログラムを用いて，SF の値を 0.1 に設定してデータセットを生成した．これにより生成できた節点・辺・属性それぞれの数を表 2 に示す．次に，生成した LDBC SNB

表 2　グラフデータセット

| SF | #節点 | #辺 | #属性 |
|---|---|---|---|
| 0.1 | 327,588 | 1,477,965 | 1,866,081 |

データセットは，LDBC SNB のデータ格納およびベンチマーク実行用プログラム [7]を用いて，データセットを Neo4j へ格納し，実験時に利用する問合せパラメタの生成を行う．

### 4.2　再帰経路に対するグラフ走査性能の評価方法

本実験では，同一の関係性を繰り返しグラフ走査する問合せの処理性能について評価する．グラフ走査自体の高速化が行えているか確認するため，ランダムに選択した単一の節点を始点とする再帰的経路のグラフ走査が行えているか評価する．本実験により，本研究で提案したグラフ走査の方法と，ネイティブ GDB のインデックスフリー隣接性のみによる方法のどちらが再帰的経路の問合せを高速に行えたか評価することができる．

4.1 項で説明した LDBC SNB のデータセットにおいて，各種 Workload [16, 17] 内のクエリで再帰経路を問い合わせているのは，節点ラベル `Person` 間の関係性 `KNOWS` と，節点ラベル `Message` 間の関係性 `REPLY_TO` である．`REPLY_TO` 関係は投稿に対するコメントやそれに対する返信を意味する関係であるため，`REPLY_TO` 関係のみを抽出すると複数のコメントスレッドが得られるため複雑ではない．したがって，本実験では `Person` 間の友好関係を意味する `KNOWS` 関係を利用する．

### 4.3　実験結果および考察

本研究の提案手法はハブの決定およびハブの収集，ハブから普遍節点への再帰経路の収集，普遍節点からハブまでの最短経路の収集を行う必要があるため，その収集に要した時間について計測した．その結果を表 3 に示す．表 3 より，ハブの決定および収集と，ハブから普遍節点までの再帰経路収集は軽量な時間コストで行えており，総経過時間の 1 %未満の時間で収集できた．一方で，99 %以上が普遍節点からハブの再帰経路収集に時間を要しており，これは普遍節点とハブの組合せ数の多さによる影響を受けている．

表 3　経路収集に要した時間 [sec.]

| | ハブ収集 | ハブ → 普遍 | 普遍 → ハブ | 計 |
|---|---|---|---|---|
| SF 0.1 | 1.51 | 0.40 | 295.36 | 297.55 |

次に，再帰経路に対するグラフ走査の性能を測るためにグラフ走査時間を計測した．その結果が図 5 の箱ひげ図であり，縦軸には手法および再帰経路の Hop 数ごと層別しており，横軸には処理に要した時間 (msec.) であり，層ごとに処理時間の統計を取っている．各 Hop 数において 5〜10%程度のグラフ走査に要する時間を削減することができたが，これらの平均値の差には有意差がないという結果となった．しかし，これは単一の節点を起点とするグラフ走査であったため，起点となる節点数が多くなるほどハブに到達する可能性が高くなり，効率化の機会が多くなると考えられる．

## 5　おわりに

本稿では，既存の GDB において次数が多いハブの影響を受けグラフ走査が非効率になっている可能性に着目し，本研究では，ハブとそれら節点からの再帰経路を管理することでグラフ走査を効率的に行えるようなグラフ管理方法を提案した．また評価実験の結果，再帰経路に対するグラフ走査の処理時間を削減することができたため，グラフ走査の効率化に有効であると結論付けた．

しかし SF を大きくしていくと，普遍節点からハブまでの経路管理が膨大になるためスケールしないことが予想された．そのため今後の課題としては，スケーラビリティの改善のために効率化に重要なハブの選定基準の拡張や普遍節点からハブまでの経路収集の効率化を考える必要がある．

7 : LDBC: ldbc_snb_implementation.git, https://github.com/ldbc/ldbc_snb_implementation （2020 年 3 月 19 日 閲覧.）

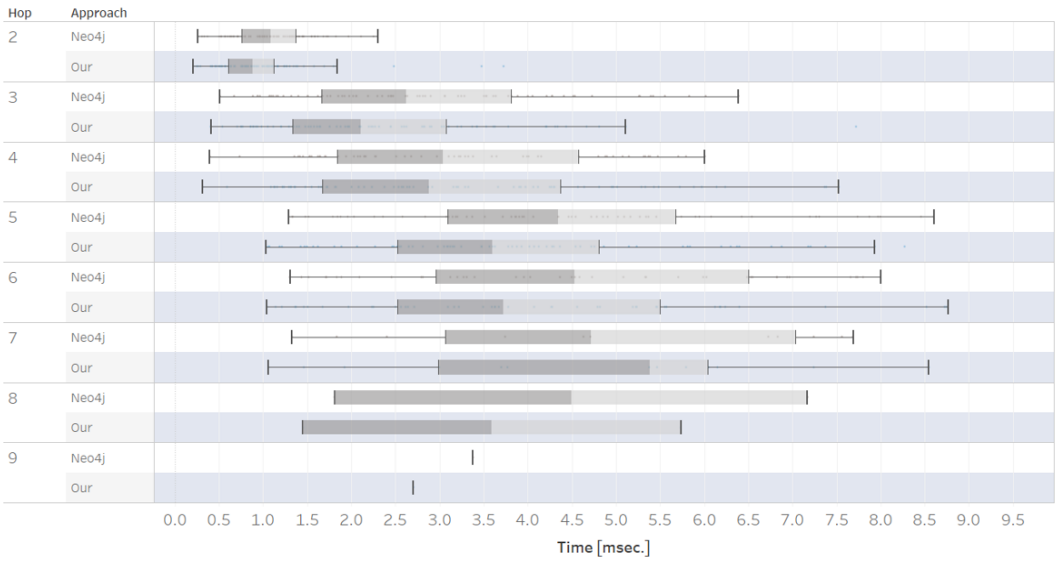


図 5　2　9 Hop の再帰経路に対するグラフ走査の経過時間

## 謝　　辞

本研究の一部は日本学術振興会特別研究員奨励費（課題番号：JP19J15498），JSPS 科研費 JP18H03342，JP19H01138，同志社大学大学院文化情報学研究科の研究推進補助金を受けて遂行された．

## 文　　献

[1] Albert-László Barabási and Márton Pósfai. *Network Science*. Cambridge University Press, 2016.

[2] Marko A. Rodriguez and Peter Neubauer. Constructions from Dots and Lines. *Bulletin of the American Society for Information Science and Technology*, Vol. 36, No. 6, pp. 35–41, 2010.

[3] Joe Celko. *Joe Celko's Trees and Hierarchies in SQL for Smarties*. Morgan Kaufmann, second edition edition, 2012.

[4] Sherif Sakr and Eric Pardede. *Graph Data Management: Techniques and Applications*. IGI Global, 2011.

[5] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases*. O'Reilly Media, Inc., 2015.

[6] Sherif Sakr and Eric Pardede. The Graph Traversal Pattern. In Marko A. Rodriguez and Peter Neubauer, editors, *Graph Data Management: Techniques and Applications*, chapter 2, pp. 29–46. IGI Global, 2012.

[7] Sherif Sakr and Ghazi Al-Naymat. The Overview of Graph Indexing and Querying Techniques. In Sherif Sakr and Ghazi Al-Naymat, editors, *Graph Data Management: Techniques and Applications*, chapter 3, pp. 71–88. IGI Global, 2012.

[8] Jaroslav Pokorný, Michal Valenta, and Martin Troup. Indexing Patterns in Graph Databases. In *Proceedings of the 7th International Conference on Data Science, Technology and Applications*, DATA 2018, p. 313–321. Science and Technology Publications, 2018.

[9] David Luaces, José R.R. Viqueira, Tomás F. Pena, and José M. Cotos. Leveraging Bitmap Indexing for Subgraph Searching. In *22nd International Conference on Extending Database Technology (EDBT)*, pp. 49–60. OpenProceedings.org, 2019.

[10] Andrew Carter, Andrew Rodriguez, Yiming Yang, and Scott Meyer. Nanosecond Indexing of Graph Data With Hash Maps and VLists. In *Proceedings of the 2019 International Conference on Management of Data*, SIGMOD '19, p. 623–635. Association for Computing Machinery, 2019.

[11] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[12] Vilfredo F. D. Pareto. La courbe des revenus. In *Cours d'économie politique*, Vol. (II), chapter (I), pp. 299–345. Librairie Droz, 1964.

[13] Albert-László Barabási and Jennifer Frangos. *Linked: The New Science Of Networks Science Of Networks*. Perseus Books Group, 2002.

[14] Matteo Lissandrini, Martin Brugnara, and Yannis Velegrakis. Beyond Macrobenchmarks: Microbenchmark-based Graph Database Evaluation. *Proceedings of the VLDB Endowment*, Vol. 12, No. 4, pp. 390–403, 2018.

[15] Linked Data Benchmark Council. *LDBC The graph & RDF benchmark reference The LDBC Social Network Benchmark (version 0.3.1)*.

[16] Orri Erling, Alex Averbuch, Josep Larriba-Pey, Hassan Chafi, Andrey Gubichev, Arnau Prat, Minh-Duc Pham, and Peter Boncz. The LDBC Social Network Benchmark: Interactive Workload. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pp. 619–630. ACM, 2015.

[17] Gábor Szárnyas, Arnau Prat-Pérez, Alex Averbuch, József Marton, Marcus Paradies, Moritz Kaufmann, Orri Erling, Peter Boncz, Vlad Haprian, and János Antal Benjamin. An Early Look at the LDBC Social Network Benchmark's Business Intelligence Workload. In *Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, GRADES-NDA '18, pp. 9:1–9:11. ACM, 2018.