グラフ構造に基づく信頼性クエリの高速推定

柳澤　隼也†　　塩川　浩昭††,†††

† 筑波大学 システム情報工学研究群　〒 305-8577 茨城県つくば市天王台 1–1–1
†† 筑波大学計算科学研究センター　〒 305-8577 茨城県つくば市天王台 1–1–1
††† JST さきがけ
E-mail: †y.junya@kde.tsukuba.ac.jp, ††shiokawa@cs.tsukuba.ac.jp

あらまし　信頼性は不確実グラフ (確率的グラフ) においてノード間の接続確率を評価する重要な指標のひとつである．信頼性の計算は# P 完全であるため，モンテカルロサンプリングを用いた近似解を計算する手法が提案されている．しかしながら，高精度な結果を得るためには多くのサンプルを必要となるため，依然として計算時間が膨大であり，大規模な不確実グラフを効率的に分析することが難しい．一般的にサンプルの生成には m 本のエッジの有無をサンプリングによって確認するため，グラフ集約は高速化において重要である．そこで本研究では，トライアングルに着目した構造情報の集約により，一回のサンプリングでトライアングル内の接続関係を得ることを可能にする．実グラフにおいてトライアングルは頻出するため，少ない計算回数で信頼性推定を可能にした．

キーワード　不確実グラフ，s-t 信頼性

1 序　　論

実世界を表すデータ構造としてグラフが存在する．例えばソーシャルネットワーク，生物学的ネットワーク，モバイルネットワークなどが代表的なグラフとして挙げらる [12]．グラフは一般的にデータオブジェクトをノードとし，データオブジェクト間の関係をエッジとして表す．ところが，ノイズの多い測定や推論および予測モデルに由来するデータが多く存在している．このようなデータに対してグラフ表現を構築する場合，エッジに存在確率が付与された不確実グラフを用いることが一般的である．不確実グラフは従来のグラフ表現とは異なり，表現力が高く多様なデータを表現可能であるため，多くのアプリケーションにおいて近年注目を集めている．

不確実グラフの解析における基礎的なトピックとして，接続性を表すネットワーク信頼性 (k 端子信頼性) の計算がある [引用]．ネットワーク信頼性は不確実グラフにおいて，与えられたクエリノード集合 N 内の任意の 2 ノード間にパスが存在する確率を表す．特に N が全てのノードである場合に全端子信頼性，$|N| = 2$ である場合に 2 端子信頼性と呼ぶ．2 端子信頼性は，類似度尺度として用いることで，不確実グラフに対するクラスタ分析 [2], [5] や k 近傍検索 [14] を実現している．ネットワーク信頼性は，従来コンピューター回路において信頼できる回路を構築する問題 [9] として考えられたが，最近では，ソーシャルネットワークや生物学的ネットワーク，通信ネットワーク，都市計画など，不確実グラフとして自然に表現できる他の種類のネットワークにも注目が集まっている [6], [10], [13].

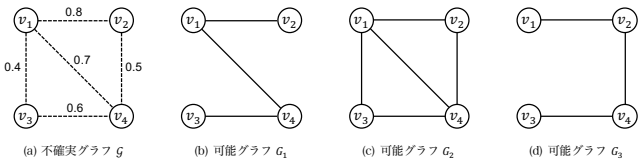
このネットワーク信頼性の計算は# P 完全 [1] であることが知られており，多くの計算コストが必要となる．これは厳密なネットワーク信頼性はひとつの不確実グラフから生成することができる全ての種類のグラフインスタンス (可能グラフ) を列挙し，その中でクエリを満たす可能グラフの生成確率を足し合わせることで求めることができる．しかしながら，全ての可能グラフの列挙は不確実グラフのエッジ数を m としたときに，$O(2^m)$ の計算量を必要とするため，厳密なネットワーク信頼性の計算は現実的ではない．

上述の問題を解決するために，サンプリング技術を用いて近似的にネットワーク信頼性を求める高速化手法がこれまで提案されている．代表的な手法として，モンテカルロ法を用いた手法 [4] が挙げられる．この手法では，与えられた不確実グラフを基に，2^m 個存在する可能グラフの中から少数の可能グラフをサンプリングし，サンプリングした可能グラフを用いて，s-t 信頼性を近似的に計算する．この手法では，K 個のサンプルが与えられ，ノード数を n，エッジ数を m とした時，任意のクエリノード集合 N に対して $O(K(n+m))$ 程度の計算量でネットワーク信頼性を近似的に求めることができるため，厳密解を求める場合と比較して高速に計算できる．

しかしながら，近似精度の高い結果を得るには大きなサンプル数を必要とし，R を乱数生成に必要な計算量とした場合，サンプルあたりの計算量は $O(mR)$ となる．一般的に実行時間は乱数生成に大きく依存するため，先行研究では，サンプル数の削減やサンプルあたりの計算量の改善による高速化をを行っている．

1.1 本研究の貢献

本研究では，ネットワーク信頼性の高速化のためのグラフ集約手法を提案する．具体的には，トライアングル構造に着目した集約により，サンプルあたりの計算量を削減する．また，既存のインデックスに基づいたクエリ高速化手法である ProbTree と組み合わせやサンプリングに基づくクエリ高速化手法の適用可能性を残した手法を提案した．実データを用いた評価実験では，提案手法は必要な乱数生成回数が最大約 47% 削減される



(a) 不確実グラフ $\mathcal{G}$　(b) 可能グラフ G_1　(c) 可能グラフ G_2　(d) 可能グラフ G_3

図 1: 不確実グラフの例

ことを示した．

2 事 前 知 識

本節では事前知識として不確実グラフ，ネットワーク信頼性，モンテカルロ法を用いた推定手法について述べる．

2.1 不確実グラフ

不確実グラフを $\mathcal{G} = (V, E, p)$ として表す．V と E はそれぞれノード集合と無向エッジ集合である．p は確率関数で $p : E \to (0, 1]$ である．不確実グラフから生成される可能性のあるグラフは可能グラフと呼ばれ，$\mathcal{G}$ における可能グラフを $G_i = (V, E_i)$ と定義する．ここで E_i は G_i に存在するエッジの集合で $E_i \subseteq E$ である．各エッジの存在は，他のエッジの存在とは独立に確率 $p(e)$ で生成する．$\mathcal{G}$ は各可能グラフを生成する確率を示す確率空間と考えることができる．また便宜上，G_i が $\mathcal{G}$ の可能グラフである時，$G_i \sqsubseteq \mathcal{G}$ と表す．m 個のエッジを持つ不確実グラフ $\mathcal{G}$ を考えた時，$\mathcal{G}$ における可能グラフは 2^m 通り生成しうる．可能グラフ G_i の生成確率を $Pr(G_i)$ で表すと確率 $Pr(G_i)$ は以下で定義できる．

$$Pr(G_i) = \prod_{e \in E_i} p(e) \prod_{e \in E \setminus E_i} (1 - p(e)) \qquad (1)$$

このとき，$\sum_{G_i \sqsubseteq \mathcal{G}} Pr(G_i) = 1$ であることに注意されたい．

図 1 に不確実グラフ $\mathcal{G}$ および可能グラフ G_1，G_2，$G_3 \sqsubseteq \mathcal{G}$ の例を示す．図 1 において，破線は $\mathcal{G}$ のエッジを表し，それぞれ生成確率を持つ．実線は生成されたエッジを表す．式 (1) に示すように $\mathcal{G}$ は確率 $Pr(G_1) = 0.8 \times 0.7 \times 0.6 \times (1-0.4) \times (1-0.5) = 0.1008$ で生成する．同様に G_2 と G_3 はそれぞれ $\mathcal{G}$ から $Pr(G_2) = 0.0672$ および $Pr(G_3) = 0.0432$ で生成される．また，$\mathcal{G}$ は可能グラフを $2^5 = 32$ 通り生成しうるため，G_1，G_2 および G_3 に限られない．

2.2 ネットワーク信頼性

ネットワーク信頼性は不確実グラフ $\mathcal{G}$ においてクエリノード集合 $N \subseteq V$ が与えられた時，任意の 2 ノード $u, v \in N$ 間にパスが存在する確率である．ある可能グラフ G について指示関数 I_G を定義する．任意の 2 ノード $u, v \in N$ 間にパスが存在するとき $I_G(N) = 1$ とし，そうでないときは $I_G(N) = 0$ とする．不確実グラフ $\mathcal{G}$ におけるネットワーク信頼性 $R_{\mathcal{G}}(N)$ は次のように計算する．

$$R_{\mathcal{G}}(N) = \sum_{G \sqsubseteq \mathcal{G}} I_G(N) \cdot Pr(G) \qquad (2)$$

しかしながら，全ての可能グラフの列挙は不確実グラフのエッジ数を m としたときに，$O(2^m)$ の計算量を必要とする．ネットワーク信頼性の計算は# P 完全 [1] であることがしられているため，厳密なネットワーク信頼性の値を計算することは現実的ではない．したがって一般には可能グラフ G をサンプリングすることで近似的にネットワーク信頼性を計算する．

2.3 モンテカルロ法を用いた推定手法

最も素朴な推定手法はモンテカルロ法を用いた近似アルゴリズム (以降，MC) [4] である．この手法では，$\mathcal{G}$ から各エッジの発生確率に従ってランダムにサンプリングした K 個の可能グラフ G_1，G_2，$\cdots$，G_K を生成し，ネットワーク信頼性の推定値 $\hat{R}_{\mathcal{G}}(N)$ を次のように計算する．

$$\hat{R}_{\mathcal{G}}(N) = \frac{1}{K} \sum_{i=1}^{K} I_{G_i \sqsubseteq \mathcal{G}}(N) \qquad (3)$$

この手法は K 個のサンプルが与えられ，ノード数を n，エッジ数を m，乱数生成の計算量を R とした時，全体で $O(KmR + K(n + m))$ 必要である．また，エッジの有無を探索と同時に調べることで，$O(K(n + mR))$ の計算量で求めることができる．

3 先 行 研 究

本節ではネットワーク信頼性問題に適応可能なグラフ集約に関する先行研究を述べる．

ProbTree

Maniu ら [8] は有向の不確実グラフにおける信頼性クエリ高速化のためのインデックス手法，ProbTree を提案した．

[8] では，任意の数の内部ノード集合 S と最大 2 つのエンドポイントノード v_1, v_2 からなる接続された誘導部分グラフを独立部分グラフとして定義した．また，元の不確実グラフから独立部分グラフの内部ノード集合 S を取り除き，エンドポイントノード v_1, v_2 間に双方向に 2 本のエッジを挿入することで $N \subseteq V \setminus S$ に対して損失なくクエリ結果を得ることを示した．ただし，挿入されたエッジの確率はそれぞれ独立部分グラフから厳密に計算される．無向の不確実グラフにおいては，v_1, v_2 間に 1 本のエッジを挿入する．

この性質を用いて，3 つのインデックス構造を開発している．その中で，FWD (fixed-width tree decomposition) は，木分解幅 $w \leqq 2$ で，(1) インデックスの構築時間，スペース，クエリ処理時間が入力グラフサイズに線形であり，(2) 損失のないクエリ処理結果を得ることができる．実際, [7], [8] にあるように $w = 2$ の FWD は高品質の結果を生成する．本稿では，ProbTree は $w = 2$ の FWD を指すものとする．

ProbTree の問題点としては，以下が挙げられる．

- 次数の低い部分構造にのみ適用可能である．
- クエリノード集合 N によっては，元のグラフサイズで推定を行う必要がある．

4 提案手法

4.1 提案手法の概要

本研究では，ネットワーク信頼性の高速化のためのグラフ集約手法を提案する．具体的には，以下の観察に基づいてトライアングルの集約を考える．

- 実グラフはクラスタ係数が高いため，トライアングルが頻出する．
- ネットワーク信頼性において，距離情報は必要ない．
- 各可能グラフの生成確率は互いに独立である．

上記の観察に基づいて，無向の不確実グラフにおいて，トライアングルを構成する 3 ノード間の接続関係は 5 パターンのみであり互いに独立していることがわかる．それにより，少ない時間計算量と空間計算量で集約が可能であり，トライアングル内の接続関係が 1 回のサンプリングで得られることを導ける．また，実グラフにおいて本手法が有用であると考えられる．

グラフ集約は 4.2 節にて述べ，集約後の不確実グラフにおける MC によるネットワーク信頼性推定手法を 4.3 節にて述べる．

4.2 トライアングル集約

トライアングルの集約を定義する前に，トライアングルについて定義する．

定義 1 (トライアングル)**.** 不確実グラフ $\mathcal{G} = (V, E, p)$ が与えられた時，トライアングルは $G^t = (V^t, E^t)$ である．ただし，$V^t = \{v_a, v_b, v_c\} \subseteq V$ かつ $E^t = \{(v_a, v_b), (v_a, v_c), (v_b, v_c)\} \subseteq E$ である．

定義 4 に基づいて，集約トライアングルとスーパーノードについて定義する．

定義 2 (集約トライアングルとスーパーノード)**.** トライアングル $G^t = (V^t, E^t), V^t = \{v_a, v_b, v_c\}, E^t = \{(v_a, v_b), (v_a, v_c), (v_b, v_c)\}$ が与えられた時，集約トライアングルは $G^{ta} = (V^{ta}, E^{ta})$ である．ただし，$V^{ta} = \{s, v_a, v_b, v_c\}$ かつ $E^{ta} = \{(s, v_a), (s, v_b), (s, v_c)\}$ である．また，トライアングルから集約トライアングルを得ることをトライアングル集約といい，トライアングル集約で新たに追加されるノード s は，スーパーノードである．

また，スーパーノードに対して t_func を以下で定義する．

定義 3 (t_func)**.** スーパーノード s と $r \in X = \{x \in \mathbb{R} \mid 0 \leqq x < 1\}$ が与えられた時，t_func は以下に示されるエッジの集合を返す関数である．

$$t\_func(s, r) = \begin{cases} \{e_a, e_b, e_c\} & (r < p^t_{\{a,b,c\}}) \\ \{e_a, e_b\} & (p^t_{\{a,b,c\}} \leqq r < p^t_{\{a,b\}}) \\ \{e_a, e_c\} & (p^t_{\{a,b\}} \leqq r < p^t_{\{a,c\}}) \\ \{e_b, e_c\} & (p^t_{\{a,c\}} \leqq r < p^t_{\{b,c\}}) \\ \emptyset & (otherwise) \end{cases} \tag{4}$$

ここで $p^t_{\{a,b,c\}}, p^t_{\{a,b\}}, p^t_{\{a,c\}}, p^t_{\{b,c\}}$ は不確実グラフ $\mathcal{G} = (V, E, p)$ とトライアングル $G^t = (V^t, E^t), V^t = \{a, b, c\}, E^t = \{e_1, e_2, e_3\}$ から事前計算される確率であり，$e_1 = (a, b), e_2 = (a, c), e_3 = (b, c)$ とした時，

$$\begin{aligned} p^t_{\{a,b,c\}} &= p(e_1) \cdot p(e_2) \cdot p(e_3) \\ &\quad + (1 - p(e_1)) \cdot p(e_2) \cdot p(e_3) \\ &\quad + p(e_1) \cdot (1 - p(e_2)) \cdot p(e_3) \\ &\quad + p(e_1) \cdot p(e_2) \cdot (1 - p(e_3)) \\ p^t_{\{a,b\}} &= p^t_{\{a,b,c\}} + p(e_1) \cdot (1 - p(e_2)) \cdot (1 - p(e_3)) \\ p^t_{\{a,c\}} &= p^t_{\{a,b\}} + (1 - p(e_1)) \cdot p(e_2) \cdot (1 - p(e_3)) \\ p^t_{\{b,c\}} &= p^t_{\{a,c\}} + (1 - p(e_1)) \cdot (1 - p(e_2)) \cdot p(e_3) \end{aligned} \tag{5}$$

によって求めることができる．またエッジ e_a, e_b, e_c は，トライアングル G^t の集約トライアングルを $G^{ta} = (V^{ta}, E^{ta}), V^{ta} = \{s, a, b, c\}, E^{ta} = \{(s, a), (s, b), (s, c)\}$ とした時，$e_a = (s, a), e_b = (s, b), e_c = (s, c)$ である．

式 5 はトライアングルから生成されうる $2^3 - 1 = 7$ 個の可能グラフの発生確率の和で計算されている．可能グラフの発生確率は互いに独立しており，これらと全てのエッジが存在しない確率 $(1 - p(e_1)) \cdot (1 - p(e_2)) \cdot (1 - p(e_3))$ の和は 1 になる．そのため，式 4 から 0 以上 1 未満の一様乱数によって 1 回の計算で接続関係が得られることができる．

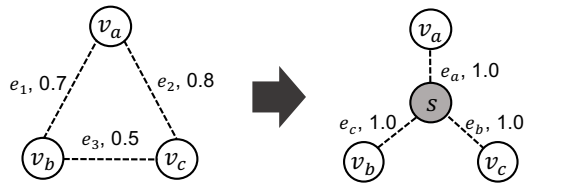
最後に，トライアングル集約によるグラフ集約後の不確実グラフについて定義する．

定義 4 (集約不確実グラフ)**.** 不確実グラフ $\mathcal{G} = (V, E, p)$ から導かれるトライアングル集合 T により $\mathcal{G}$ をグラフ集約を行った時，集約後の不確実グラフは集約不確実グラフ $\mathcal{G}' = (V, E', p', S, t\_func)$ である．ここで，E' は E から全てのトライアングル間に存在したエッジを削除し，トライアングル集約によって得られたエッジを追加したエッジ集合であり，p' は確率関数で $p' : E' \to (0, 1]$ である．S はスーパーノード集合であり，関数 t_func は全ての $s \in S$ について定義されている．

トライアングルの集約の例を図 2 に示す．(a) はトライアングルの例であり，(b) は (a) の集約トライアングルである．また，(c) は (b) のスーパーノード s に対して定義される t_func である．これは，定義 3 から求められる．

次に，提案手法のグラフ集約アルゴリズムについて述べる．本手法は，Algorithm 1 と Algorithm 2 から構成される．Algorithm 1 では，トライアングルの取集を行い，千葉ら [3] が提案したトライアングルの数え上げアルゴリズムを基に作成している．ただし，トライアングル集約後はトライアングルでなくなるため，トライアングルを構成するエッジを削除することに注意されたい．このトライアングルの収集と削除を繰り返すことで集約可能なトライアングル集合を求める．Algorithm 2 では，トライアングル集合を用いて集約不確実グラフを求める．

Algorithm 1 は，入力として不確実グラフを受け取り，トラ



(a) 不確実グラフにおける トライアングル　(b) 不確実グラフにおける 集約トライアングル

$$t\_func(s,r) := \begin{cases} \{e_a, e_b, e_c\} & (r < 0.75) \\ \{e_a, e_b\} & (0.75 \le r < 0.82) \\ \{e_a, e_c\} & (0.82 \le r < 0.94) \\ \{e_b, e_c\} & (0.94 \le r < 0.97) \\ \emptyset & (otherwise) \end{cases}$$

(c) スーパーノードsに対して定義されるt_func

図 2: トライアングル集約と t_func

イアングル集合を返す．このアルゴリズムでは，各ノードについてトライアングルを構成するか隣接ノードを調べる．調べたノードは，自身含まれる全てのトライアングルを列挙しているため，削除することができる．そのため，ノードを次数降順に調べることで効率的に探索できる．3 行目では，ノードを次数降順にソートしている．4-17 行目でトライアングルの収集を行う．5 行目で v_i の隣接ノードを候補ノードとしてマークしておく．6-16 行目で候補ノード内で互いに隣接している 2 ノードが存在しないかを調べ，存在した場合はトライアングルであるため 10-15 行目で収集を行う．v_i, u, w から誘導されるグラフがトライアングルである時，トライアングル集約によって u, w が v_i の隣接ノードでなくなるため，14，16 行目で候補ノードから除外する．また，16 行目は v_i, u で構成されうるトライアングルを全て調べた後にも実行される．15 行目は，トライアングル集約によって u が候補でなくなっているための break 文である．17 行目は，v_i が探索済であるため，削除している．

Algorithm 2 は，入力として不確実グラフとトライアングル集合を受け取り，集約不確実グラフを返す．このアルゴリズムでは，各トライアングルについて集約トライアングル，t_func を求め，集約不確実グラフを構築する．

[3] では，時間計算量が $O(m^{1.5})$ であることが示されており，Algorithm 1 についても同等である．一つのトライアングル集約に必要な時間と空間を定数と見なし，トライアングル集合を T とした時，Algorithm 2 の時間計算量および空間計算量は $O(|T|)$ である．ここで $|T|$ がとりうる最大値は $|T| = \frac{m}{3}$ であり，その場合の計算量は $O(\frac{m}{3})$ である．

4.3 トライアングル集約のネットワーク信頼性推定への適用

集約後の不確実グラフ上でネットワーク信頼性を推定するにはスーパーノードに対して特殊な処理を行う必要がある．そのため本稿では，MC への適用を考える．スーパーノードの接続関係は探索済であるかと幅優先探索に用いるキューによってエッジの削除を行わずに表現可能であることに注意されたい．

Algorithm 3 に，集約不確実グラフ上での MC によるネットワーク信頼性アルゴリズムを示す．Algorithm 4 は Algorithm 3

Algorithm 1 トライアングルの収集

Input: 不確実グラフ $\mathcal{G} = (V, E, p)$
Output: トライアングル集合 T

1: $T \leftarrow \emptyset$
2: $\mathcal{G}' = (V', E', p) \leftarrow \mathcal{G}$
3: ノード $v_1, v_2, ..., v_{|V|}$ を次数降順にソート
4: **for** $i = 1$ to $|V| - 2$ **do**
5: 　v_i の隣接ノードを候補としてマーク
6: 　**for** u in v_i の隣接ノード **do**
7: 　　**if** u が候補でない **then**
8: 　　　*continue*
9: 　　**for** w in u の隣接ノード **do**
10: 　　　**if** w が候補である **then**
11: 　　　　$V^t \leftarrow \{v_i, u, w\}, E^t \leftarrow \{(v_i, u), (v_i, w), (u, w)\}$
12: 　　　　$E' \leftarrow E' \setminus E^t$
13: 　　　　$T \leftarrow T \cup \{(V^t, E^t)\}$
14: 　　　　w の候補のマークを外す
15: 　　　　*break*
16: 　　u の候補のマークを外す
17: 　$V' \leftarrow V' \setminus \{v_i\}$
18: **return** T

Algorithm 2 トライアングルの集約

Input: 不確実グラフ $\mathcal{G} = (V, E, p)$，トライアングル集合 T
Output: 集約不確実グラフ $\mathcal{G}' = (V, E', p', S, t\_func)$

1: $S \leftarrow \emptyset$
2: $E' \leftarrow E, p' \leftarrow p$
3: **for** $(\{v_a, v_b, v_c\}, \{(v_a, v_b), (v_a, v_c), (v_b, v_c)\})$ in T **do**
4: 　$S \leftarrow S \cup \{s\}, E^{ta} \leftarrow \{(v_a, s), (v_b, s), (v_c, s)\}$
5: 　E' および p' に E^{ta} を確率 1 で追加
6: 　定義 3 に従って E^{ta} および $(v_a, v_b), (v_a, v_c), (v_b, v_c)$ から，s に対して t_func を定義する．
7: 　$E' \leftarrow E' \setminus \{(v_a, v_b), (v_a, v_c), (v_b, v_c)\}$
8: **return** (V, E', p', S, t_func)

のサブルーチンであり，スーパーノードに対する処理を示している．

Algorithm 3 に，入力として不確実グラフとクエリノード集合を受け取り，ネットワーク信頼性の推定値を返す．各イテレーションが一つの可能世界を表しているが，探索と同時にエッジの有無を確かめるため，クエリを満たしているかの確認に必要なサンプリングしか行わない．16 行目では，エッジ e の発生確率 $p'(e)$ が 1 であるか，確率 $p(e)$ でエッジのサンプリングを行い，エッジが発生するかを調べている．トライアングル集約で追加されるエッジは全て確率 1 であるため，サンプリングは必要ないことに注意されたい．スーパーノードのサンプリングは 20 行目でトラバーサル時に一回だけ行われる．以降は Algorithm 4 の処理について述べる．2 行目で，スーパーノード s と乱数を受け取り，発生したエッジ集合を返す．3 行以降はこのエッジ集合のパターンによって処理が異なる．発生したエッジ集合が空集合である時，s を探索済としてマークする．これにより，s は二度と調べられない．集約トライアングル内においてノードは全てスーパーノードを介して接続しているため，

Algorithm 3 集約不確実グラフにおける MC

Input: 集約不確実グラフ $\mathcal{G}' = (V, E', p', S, t\_func)$，クエリノード集合 N
Output: ネットワーク信頼性の推定値 $\hat{R}_{\mathcal{G}}(N)$

1: $cnt \leftarrow 0$
2: $n \leftarrow v \in N$
3: **for** $i = 1$ to K **do**
4: $\forall v \in S$ を未サンプリングでマーク
5: $\forall v \in V \cup S$ を未探索でマーク
6: $Q \leftarrow \emptyset$
7: $Q.enqueue(n)$
8: $N' \leftarrow N \setminus \{n\}$
9: n を探索済でマーク
10: **while** $Q \neq \emptyset$ **do**
11: **if** $N' = \emptyset$ **then**
12: $cnt \leftarrow cnt + 1$
13: $v \leftarrow Q.dequeue()$
14: **for** e in v の接続エッジ集合 **do**
15: $0 \leqq rand < 1$ の一様乱数 $rand$ を生成
16: **if** $p(e) = 1$ **or** $check\_exist(e, p', rand) = True$ **then**
17: $w \leftarrow v$ と e で接続しているノード
18: **if** w が未探索 **then**
19: **if** w in S **and** w が未サンプリング **then**
20: $check\_supernode(w, e, Q, N', t\_func)$
21: **else**
22: **if** w in N' **then**
23: $N' \leftarrow N' \setminus \{w\}$
24: $Q.enqueue(w)$
25: w を探索済でマーク
26: **return** $\frac{cnt}{K}$

s を探索済みとすることで全てのエッジが発生していないことを表現できる．5-13 行目では，発生したエッジ集合に e が含まれる場合の処理を示している．この場合，e を除く各発生したエッジを辿り，接続しているノードを探索用キューに追加する．7-13 行目の for 文のイテレーション回数は高々2 回であることに注意されたい．発生したエッジ集合に e が含まれない場合は特別な処理を必要としない．

Algorithm 3 の時間計算量はノード数を n，エッジ数を m，乱数生成の計算量を R，スーパーノード集合を S とした時，$O(K(n+|S|+(m-2|S|)R))$ である．ここで $|S|$ がとりうる最大値は $|S| = \frac{m}{3}$ であり，その場合の計算量は $O(K(n+\frac{m}{3}+\frac{m}{3}R))$ であり，最小値は $|S| = 0$ で元のグラフ上での MC と同等である．このように R の計算量に依存するが，実験によって集約により実行時間が短縮されることを確認し，手法の有意性を示す．

5 評価実験

5.1 実験概要

本章では，実データに対して Algorithm 1 を適用し，最大で削減可能なエッジのサンプリング回数を比較した．

データセットは以下の 3 つの実データを用いた．

Algorithm 4 $check\_supernode(s, e, Q, N', t\_func)$

1: $0 \leqq rand < 1$ の一様乱数 $rand$ を生成
2: $exist\_edges \leftarrow t\_func(s, rand)$
3: **if** $exist\_edges = \emptyset$ **then**
4: s を探索済でマーク
5: **else if** e in $exist\_edges$ **then**
6: s を探索済でマーク
7: **for** nbr_e in $exist\_edges \setminus \{e\}$ **do**
8: $nbr\_v \leftarrow s$ と nbr_e で接続しているノード
9: **if** nbr_v が探索済でない **then**
10: **if** nbr_v in N' **then**
11: $N' \leftarrow N' \setminus \{nbr\_v\}$
12: $Q.enqueue(nbr\_v)$
13: nbr_v を探索済でマーク
14: s をサンプリング済でマーク

- **NYC** [1]

NYC はニューヨークの道路ネットワークである．エッジの存在確率は $\frac{log(\alpha+1)}{log(\alpha_M+2)}$ によって得られる．ここで α，α_M はそれぞれ道路の長さとデータセットにおける道路の長さの最大値を示す．

- **Hit-d**

Hit-d は，ヒトゲノム解析センターから抽出されたタンパク質間相互作用ネットワーク (PPIs) である．相互作用のスコア $\in (0, 1]$ をエッジ存在確率として使用する．

- **DBLP**

DBLP は DBLP の共著ネットワークである．各ノードは著者であり，2 人の著者が少なくとも一つのジャーナル論文で共著者である場合，エッジによって接続される．エッジの確率は，x を共著のジャーナル論文とした時, $1 - exp\{-x/2\}$ で得られる．

NYC および Hid-d は文献 [11] の著者らから提供して頂いたものを使用した．DBLP は [2] の著者らが公開しているものを使用した．使用したグラフの詳細は表 1 に示す．

Dataset	#vertices	#edges	#triangles
NYC	180,187	208,440	881
Hit-d	18,256	248,770	631,119
DBLP	636,751	2,366,453	5,638,676

表 1: データセット

5.2 削減可能な最大のエッジサンプリング回数に関する実験

削減可能な最大のサンプリング回数に関する実験を実データを用いて行った．結果を表 2 に示す．#collect_triangles は，収集したトライアングルの数である．reduceible_rate は削減可能な最大サンプリング回数の割合を示しており，$reduceible\_rate = \frac{2 \cdot \#collect\_triangles}{\#edges} \cdot 100$ で求める．

生物学的ネットワークである Hit-d では，約 30% 削減可能，共著ネットワークである DBLP では，約 47% 削減可能でありネットワーク信頼性推定の大幅な高速化が見込める．しかし道

1：https://www.openstreetmap.org

Dataset	#edges	#collect_triangles	reducible_rate (%)
NYC	208,440	844	0.81
Hit-d	248,770	37,337	30.02
DBLP	2,366,453	557,219	47.09

表 2: グラフ集約後の最大エッジサンプリング回数

路ネットワークである NYC では，約 1%しか削減できない．これは，道路ネットワークが平面グラフに近く，エッジ数に対してトライアングル数が非常に少ないためである．

6 結　　論

本稿ではネットワーク信頼性推定高速化のためのグラフ集約手法を提案した．提案手法では実グラフにおいて頻出するトライアングルの接続関係を 1 回のサンプリングで得ることで高速化を図った．現状では，実データにおいて削減可能な最大サンプル回数を検証し，大幅な高速化の可能性を示した．

今後は，ProbTree とのネットワーク信頼性推定の実行時間の比較，ProbTree と組み合わせることによる更なる高速化を検討する．

謝　　辞

本研究の一部は JSPS 科研費 JP18K18057 ， JST ACT-I，及び JST さきがけ (JPMJPR2033)の助成を受けたものである．

文　　献

[1] M. O. Ball. Computational Complexity of Network Reliability Analysis: An Overview. *IEEE Transactions on Reliability*, Vol. 35, No. 3, pp. 230–239, Aug 1986.

[2] M. Ceccarello, C. Fantozzi, A. Pietracaprina, G. Pucci, and F. Vandin. Clustering Uncertain Graphs. *PVLDB*, Vol. 11, No. 4, pp. 472–484, December 2017.

[3] Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, Vol. 14, No. 1, pp. 210–223, 1985.

[4] G. S. Fishman. A Comparison of Four Monte Carlo Methods for Estimating the Probability of s-t Connectedness. *IEEE Transactions on Reliability*, Vol. 35, No. 2, pp. 145–155, June 1986.

[5] Kai Han, Fei Gui, Xiaokui Xiao, Jing Tang, Yuntian He, Zongmai Cao, and He Huang. Efficient and effective algorithms for clustering uncertain graphs. *Proceedings of the VLDB Endowment*, Vol. 12, No. 6, pp. 667–680, February 2019.

[6] R. Jin, L. Liu, B. Ding, and H. Wang. Distance-constraint Reachability Computation in Uncertain Graphs. *PVLDB*, Vol. 4, No. 9, pp. 551–562, June 2011.

[7] X. Ke, A. Khan, and L. L. H. Quan. An In-depth Comparison of S-t Reliability Algorithms over Uncertain Graphs. *PVLDB*, Vol. 12, No. 8, pp. 864–876, April 2019.

[8] Silviu Maniu, Reynold Cheng, and Pierre Senellart. An Indexing Framework for Queries on Probabilistic Graphs. *ACM Transactions of Database Systems*, Vol. 42, No. 2, May 2017.

[9] E.F. Moore and C.E. Shannon. Reliable circuits using less reliable relays. *Journal of the Franklin Institute*, Vol. 262, No. 3, pp. 191–208, September 1956.

[10] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. K-nearest Neighbors in Uncertain Graphs. *PVLDB*, Vol. 3, No. 1-2, pp. 997–1008, September 2010.

[11] Y. Sasaki, Y. Fujiwara, and M. Onizuka. Efficient Network Reliability Computation in Uncertain Graphs. In *Proc. EDBT 2019*, pp. 337–348, 2019.

[12] Hiroaki Shiokawa and Makoto Onizuka. Scalable Graph Clustering and Its Applications. *Encyclopedia of Social Network Analysis and Mining*, pp. 2290–2299, 2018.

[13] Ke Zhu, Wenjie Zhang, Gaoping Zhu, Ying Zhang, and Xuemin Lin. BMC: An efficient method to evaluate probabilistic reachability queries. In *Database Systems for Advanced Applications*, pp. 434–449. Springer Berlin Heidelberg, 2011.

[14] R. Zhu, Z. Zou, and J. Li. Top-k Reliability Search on Uncertain Graphs. In *Proc. ICDM 2015*, pp. 659–668, Nov 2015.