1 対全ノードに対する s-t 信頼性の高速推定

柳澤　隼也†　　塩川　浩昭††

† 筑波大学情報学群情報科学類　〒 305-8573 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
E-mail: †y.junya@kde.tsukuba.ac.jp, ††shiokawa@cs.tsukuba.ac.jp

あらまし　s-t 信頼性は不確実グラフにおいて 2 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．s-t 信頼性の計算は ♯P 完全であるため，近似解を計算する手法が提案されている．しかしながら，不確実グラフにおいて top-k 検索やクラスタリングなどの分析処理を実行するためには，1 対全ノードに対する s-t 信頼性計算を繰り返し実行する必要がある．この処理は，これまで提案されてきた近似的な計算手法を用いた場合でも，計算コストが膨大となり，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．そこで本研究では，1 対全ノードに対する高速な s-t 信頼性推定手法を提案する．提案手法では幅優先探索とグラフサンプリングに基づく s-t 信頼性推定手法を統合する．これにより，少ない計算回数で高精度に 1 対全ノードに対する s-t 信頼性を推定する．

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 [9]．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．

不確実グラフの尺度として s-t 信頼性 [1] が存在する．s-t 信頼性は不確実グラフにおいて与えられた始点ノードから終点ノードに到達可能である確率を計算したものである．既存の研究では s-t 信頼性によって求めた確率を 2 ノード間の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 [3], [5] や k 近傍検索 [10] を実現している．この s-t 信頼性の計算は ♯P 完全 [2] 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 s-t 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 (可能世界) を列挙し，その中で始点ノード s と終点ノード t が到達可能である可能世界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，全ての可能世界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 s-t 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 s-t 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 個存在する可能世界の中から少数の可能世界をサンプリングし，サンプリングした可能世界を用いて，s-t 信頼性を近似的に計算する．この手法では，サンプル数を N，ノード数を n，エッジ数を m とした時，任意の 2 ノード間に対して $O(K(n+m))$ 程度の計算量で s-t 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 s-t 信頼性を計算できる．

しかしながら，前述したクラスタリングや top-k 検索を扱う不確実グラフ分析処理では，1 つの始点ノードと全ノードに対する s-t 信頼性を繰り返し計算する場合が数多く存在する．すなわち，不確実グラフにおいて 1 対多の s-t 信頼性計算を必要とする分析処理を行う場合，従来提案されてきた s-t 信頼性の近似計算手法では依然として大きな計算量を必要とする．

1.1 本研究の貢献

本研究では不確実グラフにおける 1 対全ノードに対する s-t 信頼性の効率的な推定手法を提案する．提案手法では幅優先探索に基づく s-t 信頼性推定手法 [10] と不確実グラフにおける層化抽出法 [7] を統合する．層化抽出法はモンテカルロ法の標本誤差を減らし，サンプルの質を高める手法である．層化抽出法によって得られたサンプルを用い，幅優先探索に基づく手法で 1 対全ノードに対する到達可能性を効率的に調べることで，少ないサンプル数で高い精度の s-t 信頼性推定を達成できる．本研究では中・大規模な 5 つの実データを用いて既存手法と比較することで，提案手法は最も一般的な手法と比較して最大 200 倍，最先端の手法に対しても最大 14 倍高速であることを実験的に確認した．

2 事 前 知 識

本節では事前知識として不確実グラフ，s-t 信頼性および本研究で対象とする問題について述べる．

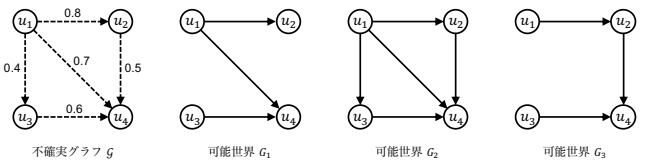

図 1: 不確実グラフの例

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有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世界と呼ばれ，$\mathcal{G}$ における可能世界を $G_i = (V, E_i)$ 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．$\mathcal{G}$ は各可能世界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 $\mathcal{G}$ の可能世界である時，$G_i \sqsubseteq \mathcal{G}$ 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，$\mathcal{G}$ における可能世界は 2^m 通り生成しうる．可能世界 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．

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

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 $\mathcal{G}$ および可能世界 G_1，G_2，$G_3 \sqsubseteq \mathcal{G}$ の例を示す．図 1 において，破線の矢印は $\mathcal{G}$ のエッジを表し，それぞれ生成確率を持つ．実線の矢印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，$\mathcal{G}$ は可能世界を $2^5 = 32$ 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 s-t 信頼性

s-t 信頼性は不確実グラフ $\mathcal{G}$ において始点ノード $s \in V$ から終点ノード $t \in V$ が到達可能である確率である．ある可能世界 G について指示関数 I_G を定義する．始点ノード s から終点ノード t が到達可能であるとき $I_G(s, t) = 1$ とし，そうでないときは $I_G(s, t) = 0$ とする．不確実グラフ $\mathcal{G}$ におけるノード s からノード t への s-t 信頼性 $R_{\mathcal{G}}(s, t)$ は次のように計算する．

$$R_{\mathcal{G}}(s, t) = \sum_{G \sqsubseteq \mathcal{G}} I_G(s, t) \cdot Pr(G) \tag{2}$$

しかしながら，全ての可能世界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．s-t 信頼性の計算は ♯P 完全 [2] であることがしられているため，厳密な s-t 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世界 G をサンプリングすることで近似的に s-t 信頼性を計算する．

2.3 問題定義

最後に本研究で対象とする問題を定義する．

定義 1 (one-to-many 信頼性)**.** 不確実グラフ $\mathcal{G}$ とノード $s \in V$ が与えられると，one-to-many 信頼性は，すべてのノード $t \in V$ の s-t 信頼性 $R_{\mathcal{G}}(s, t)$ を効率的に計算するための問題である．

s-t 信頼性を尺度として用いたクラスタ分析や top-k 検索などの不確実グラフ分析では，s-t 信頼性を尺度とした類似度の比較のために，one-to-many 信頼性を繰り返し計算する場合が数多く存在する．ゆえに，定義 1 に示した one-to-many 信頼性問題に対する高速な解法の提案は，不確実グラフ分析において重要な研究課題である．

3 先行研究

本節では定義 1 に示した問題に対する先行研究を述べる．3.1 節では，最も一般的な推定手法について述べ，それに対する探索処理に着目した高速化手法を 3.2 節，サンプルの抽出法に関する技術を 3.3 節で述べる．

3.1 モンテカルロ法を用いた手法

最も素朴な手法はモンテカルロ法を用いた近似アルゴリズム [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 個の可能世界 G_1，G_2，$\cdots$，G_K を生成し，s-t 信頼性の推定値 $\hat{R}_{\mathcal{G}}(s, t)$ を次のように計算する．

$$\hat{R}_{\mathcal{G}}(s, t) = \frac{1}{K} \sum_{i=0}^{K} I_{G_i \sqsubseteq \mathcal{G}}(s, t) \tag{3}$$

これは確率 $R_{\mathcal{G}}(s, t)$ のベルヌーイ試行であり，分散は二項分布に従う．よって推定値 $\hat{R}_{\mathcal{G}}(s, t)$ の分散は以下のとおりとなる．

$$\begin{aligned} Var(\hat{R}_{\mathcal{G}}(s, t)) &= Var\left(\frac{1}{K} \sum_{i=0}^{K} I_{G_i}(s, t)\right) \\ &= \frac{1}{K^2} \cdot Var\left(\sum_{i=0}^{K} I_{G_i}(s, t)\right) \\ &= \frac{1}{K^2} \cdot K \cdot R_{\mathcal{G}}(s, t) \cdot (1 - R_{\mathcal{G}}(s, t)) \\ &= \frac{R_{\mathcal{G}}(s, t) \cdot (1 - R_{\mathcal{G}}(s, t))}{K} \end{aligned} \tag{4}$$

ここでモンテカルロ法を用いた one-to-many 信頼性の推定手法 (以降，MC) を考える．MC では各可能世界にて全てのノードの到達可能性を同時に探索する．この手法はサンプル数を K，ノード数を n，エッジ数を m とした時，ひとつの可能世界の生成に $O(m)$ 必要であり，到達可能を調べるために $O(n + m)$ 必要である．従って，MC の計算量は $O(K \cdot (n + m))$ である．

一般的に推定器の精度は平均二乗誤差 (MSE) によって評価する．ただし，本推定器は不偏である．つまり推定器の期待値が厳密解に等しいため推定器の精度は分散で評価できる．

MC は式 (4) より，分散がサンプル数 K に反比例して減少することがわかる．つまり標準偏差は $\sqrt{K}$ に反比例して減少するため，結果の誤差はサンプル数を 4 倍にしても 1/2 にしかならず良い精度を得るには多くのサンプル数を必要とするという問題点がある．

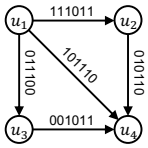

図 2: サンプル数 $K = 6$ の時の BFS Sharing における可能世界の表現

3.2 BFS Sharing

Zhu らは MC において各可能世界探索時の探索処理に大きな重複があることに着眼し，一回の幅優先探索で 1 対全ノードに対する s-t 信頼性を推定する手法 BFS Sharing [**?**] (以降，BFSS) を提案した．BFSS は MC を拡張し，各エッジに可能世界を表すビットベクトルを与える．そして，始点ノード s からビットベクトルに対してビット演算を行いながら幅優先探索することで，s から他のノードに対する s-t 信頼性の推定値を一回の幅優先探索で求める．

BFSS の詳細な処理の流れは以下のとおりである．

Step 1.
不確実グラフ $\mathcal{G}$ から可能世界 G_1，G_2，$\cdots$，G_K をサンプリングする．
この処理では MC と同様，$\mathcal{G}$ から各エッジの発生確率に従ってランダムに K 個の可能世界 G_1，G_2，$\cdots$，G_K をサンプリングする．

Step 2.
可能世界 G_1，G_2，$\cdots$，G_K を用いて全エッジ $e \in E$ のビットベクトル $\mathcal{B}_e$ を構築する．
この処理では可能世界を表すエッジのビットベクトルを構築する．BFSS ではサンプル数を K とした時，図 2 に示すように，不確実グラフの各エッジに K bit のビットベクトルを与える．ビットベクトルの各要素は 1 つの可能世界におけるエッジの有無を表している．要素が 1 の時は対応する可能世界においてエッジが存在し，0 の時は存在していないことを示す．可能世界は全てのエッジの存在によって決定されるため，全てのビットベクトルの i 番目の要素で構成されるグラフがサンプリングされた可能世界 G_i を表す．

Step 3.
ビット演算を用いた幅優先探索で到達可能性を調べる．
この処理では 1 対全ノードに対する到達可能性を Algorithm 1 によって調べる．各ノード $v \in V$ は，ノード s からノード v の到達可能性を保持するために K ビットベクトル $\mathcal{B}_v$ を持つ．i 番目のビット $\mathcal{B}_v[i]$ はノード v が可能世界 G_i のノード s から到達可能な場合のみ 1 であり，それ以外の場合は $\mathcal{B}_v[i]$ は 0 である．そのため，全ての i について $\mathcal{B}_s[i] = 1$ かつ $\mathcal{B}_v[i] = 0$ に設定しておく (2-3 行目)．次にノード s から幅優先探索 (BFS) を行う．ここで U は訪問済ノード集合，worklist は BFS 用のキューである．隣接ノードに対する処理は in-neighbors および out-neighbors によって分かれており，in-neighbors に対してはビット演算によって効率的に到達可能性を伝搬する (12-15 行目)．out-neighbors に対しては訪問済ノードの場合は worklist に追加する．それ以外の場合は，訪問済のノードに対して更新の可能性があるため Algorithm 2 によって訪問済のノードに対しての更新処理を行う．

Algorithm 1 Sharing BFS

Input: 不確実グラフ $\mathcal{G} = (V, E, p)$，サンプル数 K，始点ノード s，全エッジ $e \in E$ のビットベクトル $\mathcal{B}_e$
Output: 全ノード $v \in V$ のビットベクトル $\mathcal{B}_v$

```
1:  U ← {s}
2:  B_s ← 1
3:  B_v ← 0 for all v in V \ {s}
4:  worklist ← ∅
5:  worklist.enqueue(s の全ての out-neighbors)
6:  while worklist ≠ ∅ do
7:      v ← worklist.dequeue()
8:      if v ∈ U then
9:          continue
10:     if v ∉ U then
11:         U ← U ∪ {v}
12:     for each in-neighbors in of v do
13:         if in ∈ U then
14:             e ← (in, v)
15:             B_v ← B_v OR (B_in AND B_e)
16:     for each out-neighbors out of v do
17:         if out ∉ U then
18:             worklist.enqueue(out)
19:         else
20:             update(v, out, U)
21: return 全ノード v ∈ V のビットベクトル B_v
```

Step 4.
得られた到達可能性から推定値を導出する．
この処理では Step 3 によって得られる到達可能性を示した全ノード $v \in V$ のビットベクトル $\mathcal{B}_v$ を用いて推定値 $\hat{R}_{\mathcal{G}}(s, v) = \frac{||\mathcal{B}_v||_1}{K}$ によって求める．ここで $||\mathcal{B}_v||_1$ は $\mathcal{B}_v$ における 1 の数を意味する．

BFSS では事前サンプリングを行うことでより高速な s-t 信頼性推定が可能である．事前サンプリングとは Step 1，Step 2 を事前に実行し，全エッジのビットベクトルを静的ファイルとして保存しておくことで，信頼性推定時には Step 3，Step 4 および静的ファイルの読み込み時間のみで推定を可能にする手法である．BFSS はビットベクトルを用いるため，事前サンプリングした可能世界をコンパクトに表現・格納することができる．ただし，事前サンプリング時に対象の不確実グラフ $\mathcal{G}$ およびサンプル数 K が既知であることに注意されたい．

BFSS は 1 対全ノードに対する到達可能性を効率的に調査できることに優位性がある．しかしながら，分散は MC と同じであるため，良い精度を得るには多くのサンプル数を必要とする．したがって，one-to-many 信頼性を求めるために依然として大きな計算時間を要する．

3.3 不確実グラフにおける層化抽出法

不確実グラフではいくつかのエッジの有無を決定することで，確率空間 $\mathcal{G}$ を存在が確定したエッジのパタンに従って複数の確率部分空間に分割できる．ただし，生成される各確率空間は互いに重複せず，和空間は分割前の確率空間となる必要がある．

Algorithm 2 update(v, u, U)

```
e ← (v, u)
B_u ← B_u OR (B_v AND B_e)
u を更新済にする
Q.enqueue(u)
while Q ≠ ∅ do
    w ← Q.dequeue()
    for each out-neighbor x of w do
        if x が更新済でないかつ，U に含まれている then
            e ← (w, x)
            B'_x ← B_x OR (B_w AND B_e)
            x を更新済にする
            if B'_x ≠ B_x then
                B_x ← B'_x
                Q.enqueue(x)
```

確率部分空間は以下に定義する．

定義 2 (確率部分空間)**.** 不確実グラフ $\mathcal{G} = (V, E, p)$，存在エッジ集合 E_i^1，非存在エッジ集合 E_i^0，未確定エッジ集合 E_i^* が与えられた時，$\mathcal{G}_i = (V, E_i^1, E_i^0, E_i^*, p_i)$ は $\mathcal{G}$ の確率部分空間である．ただし，$E_i^* = E \setminus \{E_i^1 \cup E_i^0\}$ であり，p_i は，$e \in E_i^*$ に対して $p_i(e) = p(e)$ となる確率関数 $p_i : E_i^* \to (0, 1]$ である．

$Pr(\mathcal{G}_i)$ を確率部分空間 $\mathcal{G}_i$ が生成される確率とした時，確率 $Pr(\mathcal{G}_i)$ は以下で定義できる．

$$Pr(\mathcal{G}_i) = \prod_{e \in E_i^1} p_i(e) \prod_{e \in E_i^0} (1 - p_i(e)) \tag{5}$$

ここで層化抽出法に従って N 個の確率部分空間に分割した場合，s-t 信頼性の厳密解は以下で与えられる．ただし，$R_{\mathcal{G}_i}(s, t)$ は確率部分空間 $\mathcal{G}_i$ における s-t 信頼性の厳密解である．

$$R_{\mathcal{G}}(s, t) = \sum_{i=1}^{N} Pr(\mathcal{G}_i) \cdot R_{\mathcal{G}_i}(s, t) \tag{6}$$

この式は，$\mathcal{G}$ における s-t 信頼性問題を N 個の確率部分空間 $\mathcal{G}_1$，$\mathcal{G}_2$，$\cdots$，$\mathcal{G}_N$ における s-t 信頼性問題へと小問題に分割することを意味する．

また，各確率部分空間に対してサンプリングによる推定を行った場合，推定値は以下のように表すことができる．ただし，K_i は $\mathcal{G}_i$ における s-t 信頼性推定に用いたサンプル数である．

$$\hat{R}_{\mathcal{G}}(s, t) = \sum_{i=1}^{N} Pr(\mathcal{G}_i) \cdot \frac{1}{K_i} \sum_{j=1}^{K_i} I_{G_j \sqsubseteq \mathcal{G}_i}(s, t) \tag{7}$$

したがって，この式に対する分散は以下のとおりである．ただし，$\tau = R_{\mathcal{G}}(s, t)$，$\tau_i = R_{\mathcal{G}_i}(s, t)$ とする．

$$Var(\hat{R}_{\mathcal{G}}(s, t)) = \sum_{i=1}^{N} Pr(\mathcal{G}_i)^2 \cdot \frac{\tau_i(1 - \tau_i)}{K_i} \tag{8}$$

しかし，実際には $\tau_i(1 - \tau_i)$ は未知であるため定数として考えると，$\sum_{i=1}^{N} K_i = K$ の条件下で最大値を取る K_i は $Pr(\mathcal{G}_i)$ に従ってサンプル数を比例分配した $K_i = Pr(\mathcal{G}_i) \cdot K$ である．

層化抽出法はこの比例分配によってモンテカルロ法と比べ分散が低減できる．具体的にはモンテカルロ法を用いた推定器を $\hat{R}_{MC}$ とし，層化抽出法を用いた手法を $\hat{R}_{SS}$ とすると，

$$\begin{aligned} & Var(\hat{R}_{MC}(s, t)) - Var(\hat{R}_{SS}(s, t)) \\ &= \frac{\tau(1 - \tau) - \sum_{i=1}^{N} Pr(\mathcal{G}_i) \cdot \tau_i(1 - \tau_i)}{K} \\ &= \frac{\tau - \tau^2 - \sum_{i=1}^{N} Pr(\mathcal{G}_i) \cdot \tau_i + \sum_{i=1}^{N} Pr(\mathcal{G}_i) \cdot \tau_i^2}{K} \\ &= \frac{\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} Pr(\mathcal{G}_i) \cdot Pr(\mathcal{G}_j) \cdot (\tau_i - \tau_j)^2}{K} \geqq 0 \end{aligned} \tag{9}$$

ただし，$\tau = \sum_{i=1}^{N} Pr(\mathcal{G}_i) \cdot \tau_i$ であり，$\tau^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} Pr(\mathcal{G}_i) \cdot Pr(\mathcal{G}_j) \cdot \tau_i \cdot \tau_j = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} Pr(\mathcal{G}_i) \cdot Pr(\mathcal{G}_j) \cdot (\tau_i - \tau_j)^2 - \sum_{i=1}^{N} Pr(\mathcal{G}_i) \cdot \tau_i^2$ である．よってモンテカルロ法と比較して分散が小さくなることがわかる．

4 提案手法

4.1 提案手法の概要

本研究では 1 対全ノードに対する s-t 信頼性を効率的に計算する手法を提案する．提案手法の流れを図 3 に示す．提案手法は (a) 事前サンプリング部と (b) 推定部から構成されている．(a) 事前サンプリング部では不確実グラフにおける層化抽出法を用いたエッジのビットベクトルの構築を行う．その後，(b) 推定部では BFSS を用いて到達可能性を調べ，one-to-many 信頼性推定値の計算を行う．提案手法の実行時には (a-1)，(a-2)，(b-1) および (b-2) を実行する．事前サンプリングを用いる場合は，事前に (a-1)，(a-2) を実行し，(a-3) の静的ファイルを生成しておく．ただし，事前サンプリング時に対象の不確実グラフ $\mathcal{G}$，サンプル数 K，始点ノード s が既知であることに注意されたい．実行時には (a-3) の読み込みと (b-1) および (b-2) の実行を行う．

図 3 の (a-1) に示した確率部分空間の計算は，4.2 節にて新たなサンプリング方式 RCSS+を提案する．RCSS+は，Li ら [7] が提案した既存のサンプリング方式 RSS-II と RCSS を統合および改良した新たなサンプリング方式である，このフレームワークにおける (a-1) に RCSS+を用いた手法を Sharing RCSS+とし，4.3 節で説明する．

提案手法では既存手法と比較して，少ないサンプル数で分散の少ない高精度な推定値を高速に求めることを可能とする．

4.2 RCSS+

RCSS+は，Li ら [7] が提案したサンプリング方式 RSS-II と RCSS を統合および改良した新たな手法である，

RSS-II は $\mathcal{G}$ の r 本のエッジの状態を決定することで確率空間 $\mathcal{G}$ を $r + 1$ 個の非重複の確率部分空間 $\mathcal{G}_0$，$\mathcal{G}_1$，$\cdots$，$\mathcal{G}_r$ に分割し，各確率部分空間から可能世界を抽出するサンプリング方式である．また RCSS は，カットセットという概念に基づき，s-t 信頼性計算に無関係な部分確率空間の処理を省略することで RSS-II の高速化を図ったサンプリング方式である．ここで one-to-many 信頼性におけるカットセットは以下で定義される．

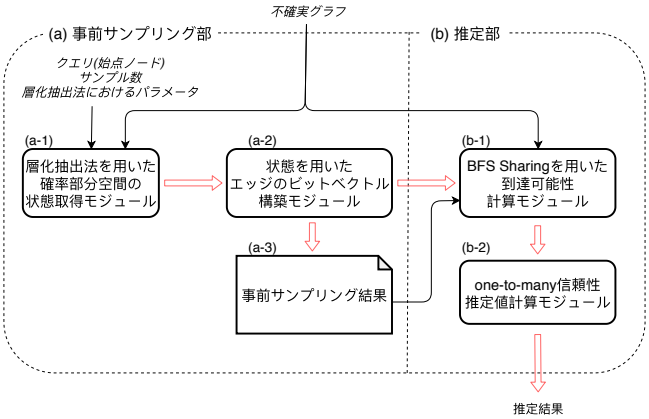

図 3: 提案手法の概略図

定義 3 (カットセット)**.** 確率部分空間 $\mathcal{G}_i = (V, E_i^1, E_i^0, E_i^*, p_i)$ およびノード $s \in V$ が与えられ，可能世界 $G = (V, E_i^1)$ において s から到達可能なノード集合を V' とした時，カットセット C は E_i^* 内の V' から $V \setminus V'$ に接続されているエッジの集合を示す．ここで E_i^1 は存在エッジ集合，E_i^0 は非存在エッジ集合，E_i^* は未確定エッジ集合である．

RCSS+はまず，幅優先探索を行いながらカットセット C を取得する．ここで，カットセットの大きさ $|C|$ がパラメータ r より大きい場合と r 以下の場合で処理は分岐する．r より大きい場合は，昇順にソートされたカットエッジの要素であるエッジを上位 r 本選択する．これ以降は RSS-II と同様の処理を行う．具体的には，r 本のエッジ $\{e_1, e_2, \cdots, e_r\} \subseteq E$ を選択すると仮定した場合，表 1 に示すエッジの選択パタンを採用することで重複のない確率部分空間を生成する．表 1 において，確率部分空間で e_i が存在することが確定している場合は 1，存在しないことが確定している場合は 0，存在が確定していない場合は $*$ で示している．ここで，$\sum_{i=0}^{r} Pr(\mathcal{G}_i) = 1$ である．また r 以下の場合は，昇順にソートされたカットセットの要素であるエッジを用いて RCSS と同様の処理を行う．ここでカットセットの大きさを $|C| = u$ とする．まず，$\mathcal{G}$ の u 本のエッジの状態を表 1 に従って決定することで確率空間 $\mathcal{G}$ を $u+1$ 個の非重複の確率部分空間 $\mathcal{G}_0$，$\mathcal{G}_1$，$\cdots$，$\mathcal{G}_u$ に分割する．ここで $\mathcal{G}_0$ はカットセットの全ての要素が存在しないので，$\mathcal{G}_0$ から生成される可能世界では始点ノード s から到達可能なノードは E_0^1 によって接続されているノード集合 V' のみである．よって確率 $Pr(\mathcal{G}_0)$ で V' に接続，$V \setminus V'$ に非接続であることを求めておくことで，$\mathcal{G}_0$ における分割および推定を避けることができる．RCSS+は上記の分割処理を再帰的に行い，更なる分散低減を行う．

RCSS はカットセット C が大きさが r の時，$r+1$ 個の確率部分空間に分割する．つまり，カットセット C が大きさが非常に大きい時，発生確率の非常に小さい確率部分空間を多く生成する可能性が高い．したがって，RCSS+では選択するエッジの本数に上限を設けることで一度の分割で生成する確率部分空間の数を制御する．また，表 1 において e_1 の発生確率が非常に高いとき，$\mathcal{G}_2$，$\mathcal{G}_3$，$\cdots$，$\mathcal{G}_r$ の発生確率が非常に低いことを意味する．したがって，選択したエッジを発生確率に従い，昇順にソートしておくが効率的な分割を可能とする．

表 1: エッジの状態決定 (1:存在，0:非存在，$*$:未確定)

Subspace	e_1	e_2	e_3	$\cdots$	e_r	e_{r+1}	$\cdots$	e_m
$\mathcal{G}_0$	0	0	0	$\cdots$	0	$*$	$\cdots$	$*$
$\mathcal{G}_1$	1	$*$	$*$	$\cdots$	$*$	$*$	$\cdots$	$*$
$\mathcal{G}_2$	0	1	$*$	$\cdots$	$*$	$*$	$\cdots$	$*$
$\mathcal{G}_3$	0	0	1	$\cdots$	$*$	$*$	$\cdots$	$*$
$\vdots$				$\ddots$				
$\mathcal{G}_r$	0	0	0	$\cdots$	1	$*$	$\cdots$	$*$

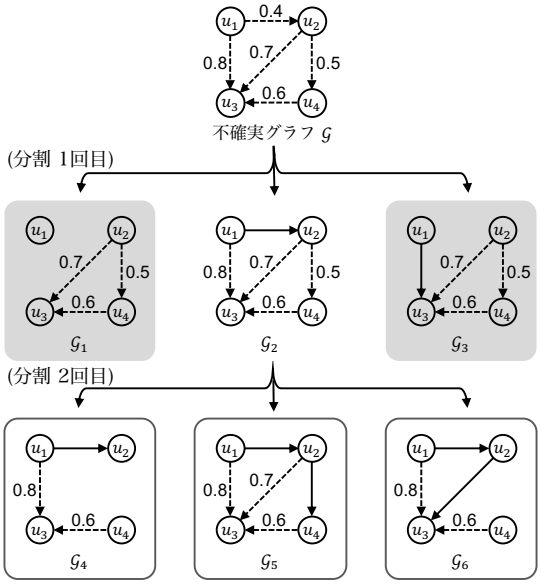

図 4: RCSS+における再帰的な確率空間分割

図 4 に始点ノードを u_1，確率空間の分割時に選択するエッジの本数の上限 $r = 2$，再帰的な分割の打切条件であるサンプル数の閾値 $\theta = 25$，サンプル数 $K = 100$ としたときの RCSS+による確率空間分割の例を示す．1 回目の確率空間の分割では $\mathcal{G}$ を 2 つのエッジ (u_1, u_2) および (u_1, u_3) を用いた 3 つの存在パターンによって，3 つの確率部分空間 $\mathcal{G}_1$，$\mathcal{G}_2$ および $\mathcal{G}_3$ を生成する．背景が灰色の確率部分空間は分割および推定処理を省略可能であることを示す．$\mathcal{G}_1$ ではカットセットの全ての要素が存在しないため，u_1 から到達可能なノードが存在しないことが分かる．そのため，$\mathcal{G}_1$ における分割および推定は省略できる．また，$\mathcal{G}_3$ は次回分割時のカットセットの大きさが $|C| = 0$ であることから，省略可能であることがわかる．$\mathcal{G}_2$ に割り当てられたサンプル数は $Pr(\mathcal{G}_2) \cdot K = 0.4 \cdot 100 = 40 \geq \theta$ であるため，2 回目の分割を行う．ここで $\mathcal{G}_2$ におけるカットセット C は，(u_1, u_3)，(u_2, u_3) および (u_2, u_4) から構成される．$|C| > r$ であるため，発生確率の小さい r 本のエッジ (u_2, u_3) および (u_2, u_4) を用いて分割を行い，$\mathcal{G}_4$，$\mathcal{G}_5$ および $\mathcal{G}_6$ を生成する．実線で囲まれた確率部分空間は推定を行うことを示す．$\mathcal{G}_4$ に割り当てられたサンプル数は $Pr(\mathcal{G}_4) \cdot K = 0.06 \cdot 100 = 6 < \theta$ であるため，サンプル数 6 で $\mathcal{G}_4$ から可能世界を生成する．$\mathcal{G}_5$，$\mathcal{G}_6$ についても同様である．最後に，処理を省略可能でなく，分割されていないすべての確率部分空間に割り当てられたサンプル数が θ よりも小さいため，分割を終了する．

Algorithm 3 層化抽出法を用いたエッジのビットベクトルの構築

Input: 不確実グラフ $\mathcal{G} = (V, E, p)$，サンプル数 K，確率部分空間の状態集合 $\mathcal{S}$

Output: 全てのエッジ $e \in E$ のビットベクトル $\mathcal{B}_e$

1: **for** e in E **do**
2: **for** (π, E^1, E^0) in $\mathcal{S}$ **do**
3: $K' \leftarrow \pi \cdot K$
4: **if** e in E^1 **then**
5: $\mathcal{B}_e$ に $\lceil K' \rceil$ ビット 1 を追加
6: **else if** e in E^0 **then**
7: $\mathcal{B}_e$ に $\lceil K' \rceil$ ビット 0 を追加
8: **else**
9: 発生確率 $p(e)$ に従って $\mathcal{B}_e$ に $\lceil K' \rceil$ ビット追加
10: **return** 全てのエッジ $e \in E$ のビットベクトル $\mathcal{B}_e$

RCSS+は入力サンプル数 100 に対してサンプル数 40 で推定を行うため，処理を大幅に省略できることがわかる．また，効率的な分割によって 2 回の分割処理しか要さない (RCSS の場合，3 回の分割が必要).

4.3 Sharing RCSS+

Sharing RCSS+は RCSS+と BFSS を統合した手法である．前節で述べた RCSS+を図 3 の (a-1) に用いることで BFSS との統合を行った．

Algorithm 5 は，(a-1) にあたる擬似コードを示す．カットセットの大きさがパラメータ r を以下の場合は RCSS と同様の処理を行い (8-14 行目)，カットセットの大きさがパラメータ r をより大きい場合はカットセット内の r 本数のエッジを用いて RSS-II と同様の処理を行う (17-22 行目)．ただし，選択したエッジを昇順にソートしていることに注意されたい (8,18 行目).

次に，Algorithm 3 において図 3 の (a-2) にあたる擬似コードを示す．確率部分空間の存在エッジ/非存在エッジおよびサンプル数に従って全エッジの状態を示すビットベクトルを構築する．はじめにエッジが存在エッジ/非存在エッジに該当するかを調べ，0/1 でビットを確率部分空間におけるサンプル数 $\lceil K' \rceil$ ビット追加する (4-7 行目)．該当しない場合は，エッジの発生確率 $p(e)$ に従って $\lceil K' \rceil$ ビット追加する (8-9 行目)．これを全てのエッジについて行い，全エッジのビットベクトルを結果として返す．Sharing RCSS+における事前サンプリングは，全エッジのビットベクトル，各確率部分空間の発生確率を静的ファイルとして保存しておく．

また，(b-1) は Algorithm 1 と同じであるため省略する．

Algorithm 4 は図 3 の (b-2) にあたる擬似コードである．式 (7) に示すように各確率部分空間における s-t 信頼性の推定値 $\hat{R}_{\mathcal{G}_i}(s, v)$ を確率部分空間の発生確率によって正規化し，one-to-many 信頼性の推定値を得る．

Sharing RCSS+における事前サンプリングは全エッジのビットベクトル，各確率部分空間の発生確率および Algorithm 5 で既に得られた s-t 信頼性を静的ファイルとして保存しておく．

MC および BFSS と比較して少ないサンプル数で同等の分散での推定が可能であり，カットセットを用いることにより省略できる確率部分空間について分割および推定を必要としないので，より高速な推定を可能とする．また，選択するエッジの本数に上限を設けることで RCSS の問題点であった分割数の制御を可能とし，効率的に分割可能である．

Algorithm 4 ノードのビットベクトルによる one-to-many 信頼性推定値の導出

Input: 全てのノード $v \in E$ のビットベクトル $\mathcal{B}_v$

Output: 全てのエッジ $e \in E$ のビットベクトル $\mathcal{B}_e$

1: **for** v in V **do**
2: $point \leftarrow 0$
3: **for** (π, E^1, E^0) in $\mathcal{S}$ **do**
4: $K' \leftarrow \pi \cdot K$
5: $count \leftarrow point$ ビット目から $point + \lceil K' \rceil$ ビット目までの 1 の数
6: $\hat{R}_{\mathcal{G}}(s, v) \leftarrow \hat{R}_{\mathcal{G}}(s, v) + \pi \cdot \frac{count}{\lceil K' \rceil}$
7: $point \leftarrow point + \lceil K' \rceil$
8: **return** one-to-many 信頼性の推定値 $\hat{R}_{\mathcal{G}}(s, v)$

Algorithm 5 Get Intermediate State^{RCSS+} (π, E^1, E^0)

Input: 不確実グラフ $\mathcal{G} = (V, E, p)$，サンプル数 K，始点ノード s，パラメータ θ, r

Output: 確率部分空間の状態集合 $\mathcal{S}$

1: $\mathcal{S} \leftarrow \emptyset$
2: $K' \leftarrow \pi \cdot K$
3: **if** $K' < \theta$ **then**
4: **return** $\{(\pi, E^1, E^0)\}$
5: **else**
6: カットセット C を取得
7: **if** $|C| \leqq r$ **then**
8: C を発生確率に従って昇順にソート
9: C を用いて E_0^1, E_0^0 を生成し，s から E_0^1 によって接続されているノード集合 V' を取得
10: V' 内の全てのノード v に対して $\hat{R}_{\mathcal{G}}(s, v) \leftarrow \hat{R}_{\mathcal{G}}(s, v) + Pr(\mathcal{G}_0)$
11: **for** $i = 1$ to $|C|$ **do**
12: C を用い，表 1 に従って E_i^1, E_i^0 を生成.
13: $\mathcal{S}_i \leftarrow$ Get-Intermediate-State^{RCSS+} $(Pr(\mathcal{G}_i), E_i^1, E_i^0)$
14: $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_i$
15: **return** $\mathcal{S}$
16: **else**
17: C を発生確率に従って昇順にソート
18: C から上位 r 本のエッジを選択
19: **for** $i = 0$ to r **do**
20: T を用い，表 1 に従って E_i^1, E_i^0 を生成.
21: $\mathcal{S}_i \leftarrow$ Get-Intermediate-State^{RCSS+} $(Pr(\mathcal{G}_i), E_i^1, E_i^0)$
22: $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}_i$
23: **return** $\mathcal{S}$

5 評価実験

5.1 実験概要

本章では，実データに対して提案手法 Sharing RCSS+と既存のサンプリング方式 RSS-II および RCSS を図 3 の (a-1) に用いた手法 (それぞれ Sharing RSS，Sharing RCSS)，既存手法の MC

表 2: データセットの特徴

Dataset	#Nodes	#Edges	Edge Prob: Mean, SD, Quarities
LastFM	6 899	23 696	0.29 ± 0.25, {0.13, 0.20, 0.33}
NetHEPT	15 233	62 774	0.04 ± 0.04, {0.001, 0.01, 0.10}
NYC	180 188	416 880	0.29 ± 0.13, {0.20, 0.28, 0.38}

および BFSS を実行することで実行速度の観点から提案手法の有効性を検証する．

データセットは以下の 3 つの実データを用いた．

- **LastFM** [1]

LastFM はユーザーがお気に入りのトラックを聴き，音楽の好みに基づいて相互に通信する音楽ソーシャルネットワークである．LastFM のローカルネットワークをクロールし，2 人のユーザーが少なくとも 1 回通信した場合は 2 人のユーザーを接続して双方向グラフを作成した. 任意のエッジの確率は，発信元ノードの出次数の逆数に対応する．

- **NetHEPT** [2]

このグラフは e-print arXiv の「High Energy Physics Theory」セクションから 1991 年から 2003 年までの論文とともに抽出されたグラフである．ノードとエッジはそれぞれ著者と共著関係を示す．ノードは，少なくとも一度共著した場合，双方向のエッジで接続される．各エッジは，(0.1, 0.01, 0.001) からランダムに均一に選択される確率で割り当てられる．

- **NYC** [3]

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．本実験ではこれを双方向グラフとして用いる．

LastFM および NetHEPT は文献 [6] の著者らが公開しているもの [4]を使用し，NYC は文献 [8] の著者らから提供して頂いたものを使用した．使用したグラフの詳細は表 2 に示す．

本実験は先行研究 [6] と同様にそれぞれのデータセットにおいて，最短パスが 2，4，6 ホップのノードペアを N 個用意する．用意したペアは s-t 信頼性推定値の精度評価に用いる．LastFM および NetHEPT では $N = 100$，NYC では $N = 10$ で実験を行った．RSS-II，RCSS，RCSS+に用いるパラメータ θ および RSS-II，RCSS+に用いるパラメータ r は先行研究 [6] で効果的とされる $\theta = 5$，$r = 50$ とする．

全てのアルゴリズムは C++で実装し，-O3 オプションを使用して GNU gcc 8.2.0 でコンパイルを行った，すべての実験は，Intel Xeon CPU（3.50GHz）および 128 GiB RAM を搭載したサーバーで行った．また，擬似乱数生成器には Mersenne twister を用いた．

推定器の評価指標として収束を定義する．収束の定義について，調査論文に従い正規化された分散 $\rho_K = \frac{V_K}{R_K}$ を用いる．ここで V_K は N 個の s-t ペアによる s-t 信頼性の分散の平均であり，$V_K = \frac{1}{N}\sum_{i=1}^{N} V(s_i, t_i, K)$ で表される．$V(s_i, t_i, K)$ は s_i，t_i によって求められる s-t 信頼性を 100 回繰り返し計算し，その不偏分散を求めたものであり，$V(s_i, t_i, K) = \frac{1}{99}\sum_{j=1}^{100}(R_j(s_i, t_i, K) - \overline{R}(s_i, t_i, K))^2$ で表される．ここで $R_j(s_i, t_i, K)$ は j 回目に求めた s_i，t_i 間の s-t 信頼性の推定値であり，$\overline{R}(s_i, t_i, K)$ は 100 回の s_i，t_i 間の s-t 信頼性の平均である．また，N 個の s-t ペアによる s-t 信頼性の平均 R_K は $R_K = \frac{1}{N}\sum_{i=1}^{N}\overline{R}(s_i, t_i, K)$ で表される．データセットと推定器，サンプル数 K が与えられ $\rho_K = \frac{V_K}{R_K} < 0.001$ である時，そのデータセットで収束しているとする．

5.2 収束に必要なサンプル数に関する実験

はじめに収束に必要なサンプル数に関する実験を行った．提案手法は層化抽出法を用いた分散低減により少ないサンプル数で既存手法と同等の精度を達成できる．

BFSS の分散は MC と同じであるため省略している．この実験ではデータセットと推定器に対して入力としてサンプル数 K を与え，収束に必要なサンプル数を調べる．ただし，与えるサンプル数 K は初期値を 100 とし，収束に達するまで 100 のステップで増加させる．各データセットにおける収束時のサンプル数を比較した結果を図 5 〜 7 に示す．ただし，層化抽出法を用いた手法において実際に用いたサンプル数がサンプル数 K 以上であることに注意されたい．

全てのデータセットにおいて層化抽出法を用いた手法が少ないサンプル数で収束に達していることがわかる．これは層化抽出法がモンテカルロ法に対して分散が低減できていることを意味する．また，Sharing RCSS+は他の層化抽出法を用いた手法以下のサンプル数で収束に達している．これは発生確率の非常に小さい確率部分空間を多数の生成を避けることが効率的な分散低減を可能にするためだと考えられる．それは s-t 信頼性が一般的に低くなる 6 ホップで顕著に現れることを確認した．また，Sharing RCSS が Sharing RSS の必要サンプル数が多い LastFM および NetHEPT のホップでは RCSS で用いるカットセットの大きさがパラメータ r より大きいことが多く見られたためだと考えられる．

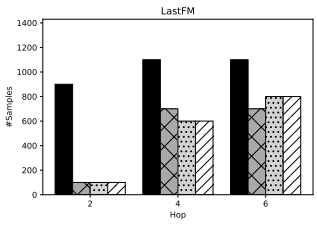

図 5: 収束サンプル数 (LastFM)

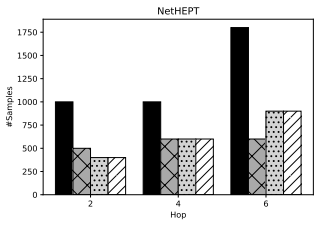

図 6: 収束サンプル数 (NetHEPT)

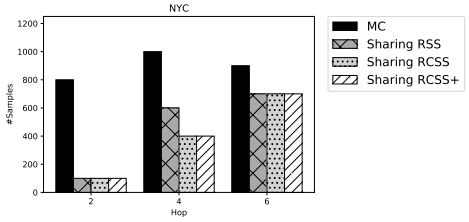

図 7: 収束サンプル数 (NYC)

1：www.last.fm

2：www.arXiv.org

3：https://www.openstreetmap.org

4：https://github.com/5555lan/RelComp

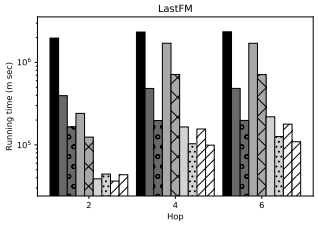

図 **8:** 実行時間 (LastFM)

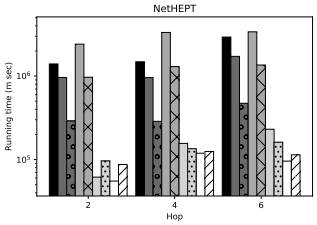

図 **9:** 実行時間 (NetHEPT)

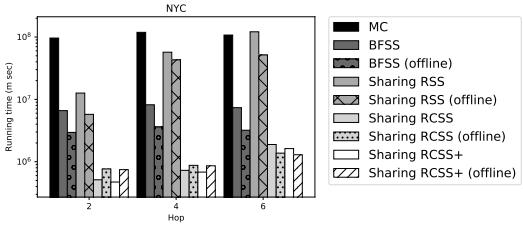

図 **10:** 実行時間 (NYC)

5.3 推定速度に関する実験

次に収束時のサンプル数を与えた際の実行時間の比較を行った．提案手法は最も一般的な手法と比較して最大約 200 倍の高速化を実現し，最先端の手法に対しても最大約 14 倍の高速化を実現した，

各データセットにおける収束時の実行時間を比較した結果を図 8 〜 10 に示す．

LastFM，NetHEPT，NYC において Sharing RCSS および Sharing RCSS+が最も高速な推定を可能．これはカットセットを用いた層化抽出法における確率部分空間の省略が非常に効果的であることがわかる．また全てのデータセットにおいて Sharing RCSS+は Sharing RCSS より高速な推定を可能にしている．これは同サンプル数であっても発生確率の非常に小さい確率部分空間を多数の生成を避けることが効率的な分散低減を可能にするためだと考えられる．また，事前サンプリングは必要サンプル数が多いほど効果的であるため，必要サンプル数が一般的に少ない低ホップの推定では事前サンプリングを用いない方が高速な場合も見られた．

5.4 事前サンプリングに関する実験

最後に収束時のサンプル数を与えた際の事前サンプリングに必要な実行時間の比較を行った．提案手法は最先端の手法に対して最大約 10 倍の高速化を実現した．

各データセットにおける収束時の事前サンプリング時間を比較した結果を図 11 〜 13 に示す．

全てのデータセットにおいて Sharing RCSS+は他の手法より高速な事前サンプリングを可能にしている．これは層化抽出法による収束に必要なサンプル数の削減と効率的な確率空間の分割によるものだと考えられる．

6 結　　論

本稿では 1 対全ノードに対する s-t 信頼性推定の高速化手法を提案した．提案手法では BFS Sharing の事前サンプリングとして RSS を用いることで推定した s-t 信頼性の分散の低減を図った．実データを用いた評価実験では，提案手法は BFSS と比較して高速に s-t 信頼性推定が可能になることを示した．また，様々な観点から比較し優位性を示した．

今後の課題として，より大規模なグラフでの比較および収束の基準に用いたクエリのホップ数を増加させた場合の比較実験を行いたい，

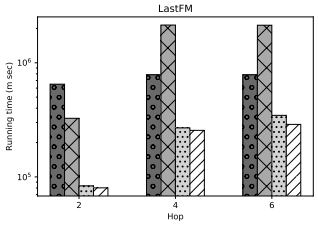

図 **11:** 事前サンプリング時間 (LastFM)

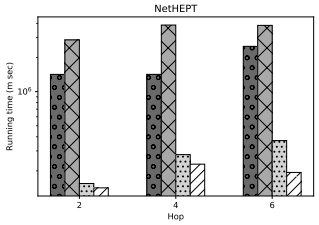

図 **12:** 事前サンプリング時間 (NetHEPT)

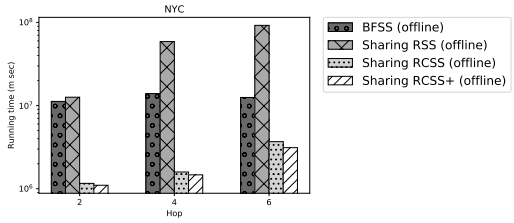

図 **13:** 事前サンプリング時間 (NYC)

謝　　辞

本研究の一部は JSPS 科研費 JP18K18057 ならびに JST ACT-I の助成を受けたものである．

文　　献

[1] Aggarwal, K.K., Misra, K.B., Gupta, J.S.: Reliability Evaluation A Comparative Study of Different Techniques. Microelectronics Reliability **14**(1), 49–56 (Feb 1975)

[2] Ball, M.O.: Computational Complexity of Network Reliability Analysis: An Overview. IEEE Transactions on Reliability **35**(3), 230–239 (Aug 1986)

[3] Ceccarello, M., Fantozzi, C., Pietracaprina, A., Pucci, G., Vandin, F.: Clustering Uncertain Graphs. PVLDB **11**(4), 472–484 (Dec 2017)

[4] Fishman, G.S.: A Comparison of Four Monte Carlo Methods for Estimating the Probability of s-t Connectedness. IEEE Transactions on Reliability **35**(2), 145–155 (Jun 1986)

[5] Han, K., Gui, F., Xiao, X., Tang, J., He, Y., Cao, Z., Huang, H.: Efficient and Effective Algorithms for Clustering Uncertain Graphs. PVLDB **12**(6), 667–680 (Feb 2019)

[6] Ke, X., Khan, A., Quan, L.L.H.: An In-depth Comparison of S-t Reliability Algorithms over Uncertain Graphs. PVLDB **12**(8), 864–876 (Apr 2019)

[7] Li, R., Yu, J.X., Mao, R., Jin, T.: Recursive Stratified Sampling: A New Framework for Query Evaluation on Uncertain Graphs. IEEE Transactions on Knowledge and Data Engineering **28**(2), 468–482 (Feb 2016)

[8] Sasaki, Y., Fujiwara, Y., Onizuka, M.: Efficient Network Reliability Computation in Uncertain Graphs. In: EDBT (Mar 2019)

[9] Shiokawa, H., Onizuka, M.: Scalable Graph Clustering and Its Applications. Springer New York, New York, NY (Jan 2017)

[10] Zhu, R., Zou, Z., Li, J.: Top-k Reliability Search on Uncertain Graphs. In: Proc. ICDM 2015. pp. 659–668 (Nov 2015)