完全動的索引によるグラフ上の影響力推定・影響最大化クエリ

Estimating and Maximizing the Spread of Influence on Graphs with Fully-Dynamic Indices

大坂直人 *1　秋葉拓哉 *1　吉田悠一 *2　河原林健一 *2
Naoto Ohsaka　Takuya Akiba　Yuichi Yoshida　Ken-ichi Kawarabayashi

*1東京大学　*2国立情報学研究所
The University of Tokyo　National Institute of Informatics

We propose the first real-time fully-dynamic index data structure for influence analysis under the independent cascade model. To the best of our knowledge, this is the first time to deal with any kind of dynamic graph changes for the independent cascade model. For this purpose, we carefully redesign the data structure of the sketching method due to Borgs et al. and devise update algorithms. Using this index, we present query algorithms for two kinds of queries, influence estimation and influence maximization, which are highly motivated from practical applications such as viral marketing. In addition, we provide theoretical analysis that guarantees the non-degeneracy of our update algorithms and the solution accuracy of our query algorithms. Through our experiments on real dynamic networks, we demonstrate the efficiency, scalability, and accuracy of the proposed indexing scheme.

1. はじめに

バイラルマーケティング [3, 10] は，ソーシャルネットワーク上の「口コミ」による情報拡散現象を利用したマーケティング戦略である．企業は商品の割引／試供品を少数集団に提供し，商品に関する評判の間接的な拡大を通して費用効果的な販売促進を狙う．マーケティング成功の鍵はネットワーク上に存在する影響力が強い少数集団を特定することである．

Kempe，Kleinberg，Tardos [7] は，バイラルマーケティングを動機づけとする**影響最大化問題**を，グラフと情報拡散モデルを入力とし，影響力最大の頂点集合を選択する離散最適化問題として定式化した．厳密解の計算は NP-hard である [7] が，各頂点の影響力を評価し最良の頂点を適応的に選択する貪欲アルゴリズムが約 63%近似解を出力する [7]．貪欲アルゴリズム高速化のため，影響力の効率的推定手法が提案されてきた [1, 2, 6, 9] が，それらは静的なグラフを対象としている．

Web グラフやソーシャルネットワークといった現実のグラフは大規模かつ動的である．**動的グラフ**における頂点の影響力遷移の追跡や高影響力頂点の実時間抽出といった需要を満たすためには新たな手法が不可欠である．静的手法はグラフ変化の度にグラフサイズの線形時間以上を要するためである．

本論文は**独立カスケードモデル** [4, 5] における実時間影響解析を実現するための索引手法を提案する．提案手法は，まず，ある時点のグラフから**索引**と呼ばれるデータ構造を作成する．そして，**頂点追加・頂点削除・辺追加・辺削除・辺確率変更**からなるグラフの変化を索引に反映させ，最新のグラフにおける**影響力推定・影響最大化**という影響解析クエリに索引を用いて答える．我々は，Borgs，Brautbar，Chayes，Lucier [1] の静的スケッチ手法に基づく索引とその更新手法及び索引を用いた影響解析クエリ手法を開発し，索引更新を続けても影響解析クエリの精度が保たれることを証明した．計算機実験により，100 万辺を有するグラフに対して，提案手法が索引を再構築の 10–10,000 倍の速度で更新し，影響解析クエリを既存手法と同等の精度かつ 10 倍以上高速に計算できることを示した．

連絡先: 大坂直人，東京大学大学院情報理工学系研究科コンピュータ科学専攻，東京都文京区本郷 7-3-1，ohsaka@is.s.u-tokyo.ac.jp

2. 予備知識

2.1 表記

辺確率付き有向グラフを $G = (V, E, p)$ $(p : E \to [0, 1])$ で表す．G における頂点 v の入近傍・出近傍を $N_G^-(v)$ と $N_G^+(v)$，入次数・出次数を $\mathrm{d}_G^-(v)$ と $\mathrm{d}_G^+(v)$ で表す．**動的グラフ**をグラフの列 $(G^1, G^2, \ldots, G^T)$ で定義する．$G^\tau = (V^\tau, E^\tau, p^\tau)$ は時刻 τ のグラフのスナップショットである．1 から k までの整数からなる集合を $[k] = \{1, 2, \ldots, k\}$ で表す．

2.2 情報拡散モデル

情報拡散モデルは**シード**と呼ばれる頂点集合から発生した影響の拡散過程を定める．本論文では，最も基本的な**独立カスケードモデル** [4, 5] を採用する．各頂点は**活性**または**非活性**の状態をとる．非活性頂点は活性になりうるが逆は起こらない．グラフ $G = (V, E, p)$ とシード $S \subseteq V$ を入力とし，次の確率的過程が繰り返される：(i)S 内の頂点の状態を活性，残りを非活性に設定する．(ii) 新しく活性化した各頂点 u は u の非活性な各出近傍 v を確率 p_{uv} で活性化させる．(iii) 新しく活性化した頂点が存在すれば (ii) に戻り，そうでなければ終了する．

2.3 影響力推定問題と影響最大化問題

独立カスケードモデルにおけるシード S の**影響拡散** [7] を「S をシードとする独立カスケードモデルの情報拡散過程終了時の活性頂点数の期待値」で定義し，$\sigma(S)$ と表記する．**影響力推定問題**を $G = (V, E, p)$ と $S \subseteq V$ を入力とし，$\sigma(S)$ を推定する問題と定義する．**影響最大化問題** [7] を $G = (V, E, p)$ と正整数 k を入力とし，$\sigma(\cdot)$ を最大化する大きさ k の頂点集合を求める問題と定義する．

影響力推定問題は#P-hard に属する [2] が，シミュレーションによる推定値が良い近似値を与えることが知られている．

影響最大化問題は NP-hard に属する [7] が，$\sigma(\cdot)$ の性質を活用した貪欲アルゴリズムが定数近似比を達成する．集合関数 $f : 2^V \to \mathbb{R}$ は，任意の $S \subseteq T \subseteq V$ と $v \in V \setminus T$ について $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$ ならば**劣モジュラ**であるという．Kempe ら [7] の影響最大化に対する**貪欲アルゴリズム**は，空のシード $S = \emptyset$ から開始し，$|S| < k$ である間，影響拡散の増加量が最大の頂点 $t = \mathrm{argmax}_{v \in V \setminus S} \sigma(S \cup \{v\}) - \sigma(S)$

を S に追加する．Theorem 1, 2 は，貪欲アルゴリズムが影響最大化問題の約 63%近似解を出力することを保証する．

Theorem 1 ([8])**.** 非負かつ単調な劣モジュラ関数 f について，貪欲アルゴリズムにより得られた大きさ k の集合を S，f の最大化問題の大きさ k の最適解を S^* とすると，$f(S) \geq (1-1/e)f(S^*) \approx 0.63 \cdot f(S^*)$ が成立する．

Theorem 2 ([7])**.** 独立カスケードモデルにおいて影響拡散関数 $\sigma(\cdot)$ は非負，単調，劣モジュラである．

3. 提案手法

3.1 索引構造

$G = (V, E, p)$ に対する索引は 3 つ組の集合 $I = \{(z_i, x_i, H_i)\}_i$ からなる．ここで，$z_i \in V$ は**目標頂点**，$x_i \in E \to [0,1]$ は各辺の状態を表す乱数，H_i は $x_i(e) < p_e$ なる辺のみを用いて z_i に到達可能な頂点からなる G の誘導部分グラフである．以後，x_i の下の辺 e の状態を $x_i(e) < p_e$ ならば**生**と呼び，$x_i(e) \geq p_e$ ならば**死**と呼ぶ．さらに，x_i の下で状態が生の辺のみを用いて頂点 u が頂点 v に到達可能ならば u は x_i **の下で** v **に到達可能**であるという．また，v が x_i の下 z_i に到達不可能ならば，H_i の構造は $x_i(uv)$ の値に依存しない．そこで，メモリ使用量削減のため，$x_i(uv)$ の値は必要時にのみ決定し，v の削除時に索引から削除する．したがって，索引に保存されている $x_i(e)$ の個数は $\sum_{i\in[|I|]} \sum_{v\in V(H_i)} \mathrm{d}^-_G(v)$ である．部分グラフ H の**重み**を $w(H) = |V(H)| + \sum_{v\in V(H)} \mathrm{d}^-_G(v)$ と定義する．I 中の 3 つ組の個数は，R をパラメータとして，

$$\sum_{i\in[|I|-1]} w(H_i) < R \quad \text{かつ} \quad \sum_{i\in[|I|]} w(H_i) \geq R, \qquad (*)$$

を満たすよう唯一に定める．

3.2 静的グラフからの索引構築

$G = (V, E, p)$ から索引を構築する手法を述べる．まず，空の索引 I から始め，I の総重みが R 以下である間，次を反復する：$z \in V$ を一様無作為に抽出し，x の下で生の辺のみを使った逆幅優先探索を z から行う．訪問した頂点からなる誘導部分グラフ H を求め，3 つ組 (z, x, H) を I に挿入する．

3.3 影響解析クエリ手法

提案索引を用いた影響力推定・影響最大化手法を提案する．

影響力推定クエリ 頂点集合の影響拡散推定の近似手法を述べる．I_S を $S \cap V(H_i) \neq \emptyset$ なる添字 i の集合とする．提案手法は $\sigma(S)$ の値を $|V| \cdot |I_S|/|I|$ により推定する．素朴な実装は時間 $\Theta(|I| \cdot |S|)$ を要するが，各 $v \in V$ の $I_{\{v\}}$ を保持するヒューリスティクスは時間が $\Theta(\sum_{v\in S} |I_{\{v\}}|)$ に抑えられる．

影響最大化クエリ 影響拡散を最大化する大きさ k の頂点集合を求める近似手法を述べる．索引 I に対する頂点 v の次数 $\mathrm{d}_I(v)$ を $v \in V(H_i)$ なる添字 i の個数とする．また，$S \subseteq V$ に対し $I - S$ を $S \cap V(H_i) \neq \emptyset$ なる (z_i, x_i, H_i) を I から削除した索引とする．提案手法は，影響拡散の増加量を $\sigma(S \cup \{v\}) - \sigma(S) \approx |V| \cdot \mathrm{d}_{I-S}(v)/|I|$ で近似し，貪欲アルゴリズムに基づきシードを選択する．

以上のクエリ手法は Borgs ら [1] の手法と等しい結果を返すため，次の定理が成立する．

Theorem 3 (影響力推定 [1])**.** $R = \Theta(\frac{1}{\epsilon^3}(|V| + |E|) \log |V|)$ とする．提案手法は，$1 - \frac{1}{|V|}$ 以上の確率で $\sigma(S)$ の推定値を付加誤差 ϵ 以内で返す．

Theorem 4 (影響最大化 [1])**.** $R = \Theta(\frac{1}{\epsilon^3}(|V| + |E|) \log |V|)$ とする．提案手法は，$1 - \frac{1}{|V|}$ 以上の確率で最適解に対する近似比 $(1 - 1/e - \epsilon)$ 以上の大きさ k の頂点集合を返す．

Algorithm 1 提案索引の動的更新操作．

```
1: procedure AddVertex(I, v)                                   ▷ 頂点追加
2:     α ← 1/(|V|+1) and i ← 0
3:     V ← V ∪ {v}
4:     while i < |I| do
5:         x ←_R [0,1], k ← ⌈log(1/(1−x)) / log(1/(1−α))⌉, and i ← i + k
6:         if i ≤ |I| then
7:             H_i ← ∅ and z_i ← v
8:             Expand(I, i, z_i)
9:     Adjust(I)

10: procedure DeleteVertex(I, v)                               ▷ 頂点削除
11:     V ← V \ {v}
12:     E ← E \ [{vw | w ∈ N⁺(v)} ∪ {uv | u ∈ N⁻(v)}]
13:     for i = 1 to |I| do
14:         if z_i = v then
15:             H_i ← ∅ and z_i ←_R V
16:             Expand(I, i, z_i)
17:         else if v ∈ V(H_i) then
18:             Shrink(I, i)
19:     Adjust(I)

20: procedure Change(I, uv, p)                                 ▷ 辺確率変更
21:     p′ ← p_uv and p_uv ← p
22:     for all i ∈ [|I|] s.t. v ∈ V(H_i) do
23:         if p′ ≤ x_i(uv) < p then
24:             Expand(I, i, u)
25:         if p ≤ x_i(uv) < p′ then
26:             Shrink(I, i)
27:     Adjust(I)

28: procedure AddEdge(I, uv, p)                                ▷ 辺追加
29:     E ← E ∪ {uv} and p_uv ← 0
30:     for all i ∈ [|I|] s.t. v ∈ V(H_i) do
31:         x_i(uv) ←_R [0,1]
32:     Change(I, uv, p)
33:     Adjust(I)

34: procedure DeleteEdge(I, uv)                                ▷ 辺削除
35:     Change(I, uv, 0)
36:     E ← E \ {uv}
37:     Adjust(I)
```

3.4 索引の動的更新手法

時刻 $\tau - 1$ から τ への間には，頂点集合 $V^\tau \setminus V^{\tau-1}$ の $V^{\tau-1} \setminus V^\tau$ が追加・削除，辺集合 $E^\tau \setminus E^{\tau-1}$ の $E^{\tau-1} \setminus E^\tau$ が追加・削除，$p_e^{\tau-1} \neq p_e^\tau$ なる辺 e の確率変更が発生する．本節は，各変更を索引に反映させる更新手法を提案する．

3.4.1 補助手続

まず，索引更新手法が補助的に用いる 3 つの手続を説明する．

誘導部分グラフの拡大 $\textsc{Expand}(I, i, v)$

頂点 v が x_i の下で新たに目標頂点 z_i に到達可能となった時には，v を通り x_i の下で z_i に到達可能な頂点を H_i に追加する必要がある．手続 $\textsc{Expand}(I, i, v)$ は，$V \setminus V(H_i)$ からなる誘導部分グラフ上で v から逆幅優先探索を行い，訪問した頂点を H_i に追加する．

誘導部分グラフの縮小 $\textsc{Shrink}(I, i)$

ある辺の状態が死に変わった時には，x_i の下で z_i に到達不可能となった頂点を H_i から削除する必要がある．手続 $\textsc{Shrink}(I, i)$ は，z_i から逆幅優先探索を行い，x_i の下で z_i に到達可能な頂点集合 H_i を再計算する．

3 つ組の個数の調整 $\textsc{Adjust}(I)$

更新後の索引は式 $(*)$ に反することがある．手続 $\textsc{Adjust}(I)$ は，索引の総重みが R 未満である間，新たに生成した 3 つ組を索引の末尾に追加し，末尾の 3 つ組を除く索引の総重みが R 以上である間，末尾の 3 つ組を索引から取り除く．

3.4.2 更新操作

次に，Algorithm 1 に示す索引の更新操作を述べる．変化後のグラフから再構築した索引の分布と等しくなるように各更新

操作は現在の索引を更新する．

頂点追加 AddVertex(I, v)

新しい孤立頂点 v を追加したとする．この時，「各目標頂点が v を含む新しい頂点集合から一様無作為に抽出された」という性質を保つため，索引中の目標頂点を変更する．V を v の追加直前の頂点集合とする．v を追加後のグラフから索引を再構築したとすると，各目標頂点が V から選択される確率は $\frac{|V|}{|V|+1}$ であり，v となる確率は $\frac{1}{|V|+1}$ である．したがって，各目標頂点を確率 $\frac{1}{|V|+1}$ で v に変更すればよい．この手順は $|I|$ 個の 3 つ組を走査する．

計算時間の削減技法 目標頂点を初めて変更する 3 つ組の添字を表す確率変数を k とする．k の標本は，$[0,1]$ から一様無作為に抽出した x について，$\sum_{1\leq t\leq k'} \Pr[k=t] \geq x$ を満たす最小の k' で得られる．$\Pr[k=t] = (1-\alpha)^{t-1}\alpha$ $(\alpha = \frac{1}{|V|+1})$ であるから，$k' = \lceil \log \frac{1}{x-1} / \log \frac{1}{1-\alpha} \rceil$ として k を標本し，$z_{k'}$ を v に変更し，$k'+1$ 番目以降の 3 つ組からなる索引について，同じ処理を繰り返せばよい．走査する 3 つ組の期待個数は $\frac{|I|}{|V|+1}$ であり，$|I|$ に比べて非常に小さい．

頂点削除 DeleteVertex(I, v)

頂点 v を削除したとする．この時，$v \in V(H_i)$ なる添字 i について，もし $z_i = v$ であれば，目標頂点を $V \setminus \{v\}$ から一様無作為に抽出し H_i を再計算し，そうでなければ，頂点 v と v の隣接辺を H_i から削除し H_i を縮小する．

辺確率変更 Change(I, uv, p)

辺 uv の確率を p' から p に変更したとする．この時，$v \in V(H_i)$ なる添字 i について x_i の下で uv の状態が生に変化した $(p' < x_i(uv) \leq p)$ ならば，H_i を拡大し，死に変化した $(p < x_i(uv) \leq p')$ ならば，H_i を縮小する．

辺追加 AddEdge(I, uv, p)

確率 p の辺 uv を追加したとする．この時，E に辺 uv を挿入し，辺確率を $p_{uv} = 0$ に割り当てる．そして，$v \in V(H_i)$ なる添字 i について，$x_i(uv)$ を $[0,1]$ から一様無作為に抽出し，最後に uv の確率を 0 から p に変更する (Change(I, uv, p))．

辺削除 DeleteEdge(I, uv)

辺 uv を削除したとする．この時，uv の辺確率を 0 に変更し (Change$(I, uv, 0)$)，E から uv を削除する．

3.5 理論的解析

本節では，動的更新による索引分布の非退化性を証明する．$\mathfrak{I}_R^{\mathrm{sta}}(G)$ をグラフ G から静的に構築することで得られる索引の分布とし，$\mathfrak{I}_R^{\mathrm{dyn}}(G)$ を G を最終状態とする更新の列を適用することで得られる索引の分布とする．目的は，$\mathfrak{I}_R^{\mathrm{sta}}(G) = \mathfrak{I}_R^{\mathrm{dyn}}(G)$ を示すことである．

まず，$G = (V, E, p)$ に対する 3 つ組列を生成する次の確率的過程を考える：目標頂点 $z \in V$，辺乱数 $x : E \to [0,1]$ を一様無作為に抽出し，3 つ組 $(z, x, H(z,x))$ を列に加える，ただし，$H(z,x)$ は x の下で z に到達可能な頂点のみからなる G の誘導部分グラフとする．$\mathfrak{I}_\infty(G)$ を以上の手順で得られる 3 つ組の無限列の分布とする．3 つ組列の分布 $\mathfrak{I}$ が G について**正当**であるとは，$\mathfrak{I}_\infty(G)$ から一様無作為に抽出した（無限）列の接頭辞を残すことで $\mathfrak{I}$ が得られるときをいう．正整数 R について，$\mathfrak{I}_R(G)$ を $\mathfrak{I}_\infty(G)$ から $(z_1, x_1, H_1), (z_2, x_2, H_2), \ldots$ を標本し，総重みが R 以上となる最小の接頭辞として得られる正当な分布とする．

明らかに $\mathfrak{I}_R(G) = \mathfrak{I}_R^{\mathrm{sta}}(G)$ であるから，$\mathfrak{I}_R^{\mathrm{dyn}}(G) = \mathfrak{I}_R(G)$ を示すことで $\mathfrak{I}_R^{\mathrm{sta}}(G) = \mathfrak{I}_R^{\mathrm{dyn}}(G)$ を証明する．

分布 $\mathfrak{I}$ から標本した列に Adjust を適用することで得られる列の分布を Adjust$(\mathfrak{I})$ と表記する．同様の表記を AddVertex, DeleteVertex, Change, AddEdge, DeleteEdge に用いる．G をグラフ，$\mathfrak{I}$ を G に対する正当な分布とする時，Lemma 5, 6, 7, 8, 9, 10 が成立する．

Lemma 5. AddVertex$(\mathfrak{I}, v)$ は G に新しい頂点 $v \notin V(G)$ を追加したグラフについて正当である．

Lemma 6. DeleteVertex$(\mathfrak{I}, v)$ は G から頂点 $v \in V(G)$ を削除したグラフについて正当である．

Lemma 7. Change$(\mathfrak{I}, e, p)$ は G 中の辺 $e \in E(G)$ の確率を p に変更したグラフについて正当である．

Lemma 8. AddEdge$(\mathfrak{I}, e, p)$ は G に確率 p の新しい辺 $e \notin E(G)$ を追加したグラフについて正当である．

Lemma 9. DeleteEdge$(\mathfrak{I}, e)$ は G から辺 $e \in E(G)$ を削除したグラフについて正当である．

Lemma 10. Adjust$(\mathfrak{I})$ は $\mathfrak{I}_R(G)$ に等しい．

Theorem 11. $\mathfrak{I}_R^{\mathrm{sta}}(G) = \mathfrak{I}_R^{\mathrm{dyn}}(G)$.

Proof. Lemma 5, 6, 7, 8, 9 より，更新列を適用後の索引の分布は現在のグラフについて正当である．更新の最後に Adjust を呼び出すため，Lemma 10 より，索引の分布 $\mathfrak{I}_R^{\mathrm{dyn}}(G)$ は $\mathfrak{I}_R(G)$ に一致する．ゆえに，$\mathfrak{I}_R^{\mathrm{sta}}(G) = \mathfrak{I}_R^{\mathrm{dyn}}(G)$. □

3.3 節の手法と Borgs ら [1] の手法の結果が等しいことと Theorem 11 とから任意の時点で Theorem 3, 4 が成立する．

4. 評価実験

4.1 実験設定

データセット The Koblenz Network Collection[*1] より，辺の生成時刻をもつ有向ソーシャルネットワーク Epinions $(|V| = 114,222, |E| = 717,129)$ と無向ソーシャルネットワーク Facebook $(|V| = 63,731, |E| = 1,634,070)$ を用いる．

確率設定 辺 uv の確率 p_{uv} は，**一様カスケードモデル** (UC$_P$) [7] においてはパラメータ P を，**三価モデル** (TR) [2] においては $\{0.1, 0.01, 0.001\}$ から無作為に選択した確率を，**重み付きカスケードモデル** (WC) [7] においては $1/\mathrm{d}_G^-(v)$ を割り当てる．

提案手法と比較対象の設定 提案手法のパラメータ R は $R = \beta(|V| + |E|)\log_2 |V|$ と定める．以下の手法を影響最大化の性能比較に用いる．

PMC [9]： シミュレーションに基づく手法である．部分グラフの個数は 200 に設定した．

RIS [1]： スケッチに基づく手法である．逆幅優先探索は走査した辺数が $32(|V| + |E|)\log_2 |V|$ に達するまで行う．

IRIE [6]： 影響拡散を連立線形方程式として表す手法である．パラメータの値は $\alpha = 0.7, \theta = 1/320$ に設定した [6]．

Degree： k 頂点を次数の降順に選択する．

Random： k 頂点を無作為に選択する．

環境・実装 C++で実装された各手法を Linux サーバー（CPU: Intel Xeon X5670 (2.93 GHz)，メモリ: 48GB）上で実行した．IRIE は Kyomin Jung[*2] より提供された実装を使用し，その他の手法は本論文の著者による実装を使用した．

4.2 結果

索引構築 Table 1 に索引構築時間を示す．100 万辺を有するグラフの索引構築は 1,000 秒程度を要する．静的手法 RIS は

*1 http://konect.uni-koblenz.de/networks/

*2 [6] の著者の一人．

表 1: 提案手法 ($\beta = 32$) の索引構築時間と平均索引更新時間.

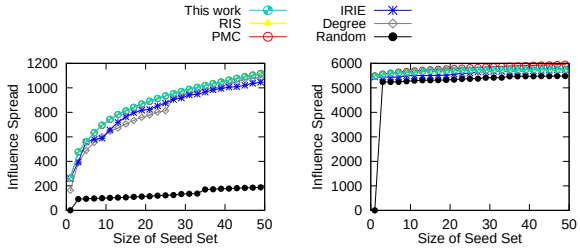
データセット	モデル	メモリ使用量	索引構築	頂点追加	頂点削除	辺確率変更	辺追加	辺削除
Epinions	$UC_{0.01}$	9.5 GB	290.2 s	6.9 ms	3,304.2 ms	72.3 ms	78.1 ms	314.2 ms
	$UC_{0.1}$	12.2 GB	350.5 s	3.2 ms	7,569.0 ms	329.5 ms	494.9 ms	1,268.2 ms
	TR	12.5 GB	316.7 s	2.9 ms	6,243.5 ms	554.7 ms	397.9 ms	602.0 ms
	WC	9.6 GB	298.7 s	4.4 ms	571.9 ms	7.9 ms	8.6 ms	26.1 ms
Facebook	$UC_{0.01}$	19.7 GB	856.0 s	13.0 ms	20,472.2 ms	194.8 ms	53.4 ms	100.5 ms
	$UC_{0.1}$	29.4 GB	864.1 s	10.4 ms	69,730.9 ms	824.2 ms	1,193.6 ms	2,026.8 ms
	TR	24.1 GB	754.7 s	10.2 ms	54,298.3 ms	866.3 ms	536.4 ms	1,662.3 ms
	WC	26.1 GB	698.0 s	12.9 ms	5,005.7 ms	44.2 ms	16.6 ms	121.8 ms

表 2: 単一頂点の影響力推定の平均時間.

データセット	モデル	提案手法 ($\beta = 32$)	RIS [1]	シミュレーション
Epinions	$UC_{0.01}$	0.016 ms	6.5 s	0.072 s
	$UC_{0.1}$	0.038 ms	6.9 s	10.1 s
	TR	0.024 ms	6.7 s	3.6 s
	WC	0.033 ms	6.7 s	0.043 s

表 3: 影響最大化 ($k = 50$) の計算時間.

データセット	モデル	提案手法 ($\beta = 32$)	RIS [1]	PMC [9]	IRIE [6]
Epinions	$UC_{0.01}$	0.311 s	7.0 s	5.1 s	9.5 s
	$UC_{0.1}$	0.814 s	7.9 s	10.1 s	10.0 s
	TR	0.412 s	7.6 s	8.4 s	9.7 s
	WC	0.505 s	7.5 s	13.5 s	9.7 s



(1.1) Epinions ($UC_{0.01}$) (1.2) Epinions (TR)

図 1: 提案手法 ($\beta = 32$) と既存手法の影響拡散の比較.

グラフ変化の度に索引再構築のためこの程度の時間を要するが，提案手法は以下の通り索引を高速に更新する.

索引の動的更新 各更新操作の処理時間を次の手順で計測した.

- 頂点追加 : グラフ全体から索引を構築後，孤立した 1,000 頂点を追加する.
- 頂点削除 : グラフ全体から索引を構築後，無作為に選択した 1,000 頂点を削除する.
- 辺追加 : 辺を時刻の昇順に整列し，最後 1,000 辺を除くグラフ全体から索引を構築後，除かれた 1,000 辺を追加する.
- 辺削除 : グラフ全体から索引を構築後，辺追加の逆順に 1,000 辺を削除する.
- 辺確率変更:グラフ全体から索引を構築後，1,000 辺を一様無作為に選択する．各辺 e の確率 p_e を，TR では $\{0.1, 0.01, 0.001\} \setminus \{p_e\}$ から選択した確率に，それ以外のモデルでは $p_e \times 2$ と $p_e/2$ のいずれかに更新する.

Table 1 に平均処理時間を示す．頂点追加は最も速く再構築の 10,000 倍以上高速である．辺に関する操作は再構築の 1,000 倍程度速いが，$UC_{0.1}$ と TR においては $UC_{0.01}$ と WC に比べて 10–50 倍遅い．$UC_{0.1}$ と TR は辺確率が高いため，Expand と Shrink 中の逆幅優先探索が時間を消費したことが原因といえる．最も遅い操作は頂点削除である．一頂点の削除で多数の辺が削除され，部分グラフの大幅な変更が発生するからである.

影響力推定クエリ 1,000 頂点を頂点集合から無作為に選択し，各頂点の影響拡散の推定時間を計測した．Table 2 に平均推定時間示す．提案手法は 1 ミリ秒未満であり，RIS やシミュレーションの 1,000 倍以上高速である.

影響最大化クエリ Table 3 に大きさ $k = 50$ の頂点集合の選択に要する時間を示す．提案手法は 1 秒以下であり，既存手法の 20 倍程度高速である．Figure 1 にシードの大きさ k ごとの影響拡散値を示す．提案手法と RIS とは，索引分布が等しいため，ほぼ同等の性能を示している．提案手法は PMC と比較してわずかに劣っているが，最大の隔たりは Epinions (TR, $k = 50$) における高々4%である．IRIE は提案手法よりも Epinions ($UC_{0.01}$, $k = 3$) において 17%劣る．また，Degree や Random は明らかに影響拡散の値が低い.

5. おわりに

本論文は，動的グラフ上の影響解析クエリのための索引手法を提案した．実験により，提案手法はグラフの変化に伴う索引の動的更新を再構築の 10–10,000 倍高速に行い，影響力推定・影響最大化クエリに既存手法より高速かつ同等精度で答えることを示した．今後は，より大規模なグラフへ適用するためのメモリ使用量の削減や索引更新の一括処理，新たな影響解析クエリの検討を行う.

参考文献

[1] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing Social Influence in Nearly Optimal Time. In *SODA*, pages 946–957, 2014.

[2] W. Chen, C. Wang, and Y. Wang. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In *KDD*, pages 1029–1038, 2010.

[3] P. Domingos and M. Richardson. Mining the Network Value of Customers. In *KDD*, pages 57–66, 2001.

[4] J. Goldenberg, B. Libai, and E. Muller. Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth. *Marketing Letters*, 12(3):211–223, 2001.

[5] J. Goldenberg, B. Libai, and E. Muller. Using Complex Systems Analysis to Advance Marketing Theory Development: Modeling Heterogeneity Effects on New Product Growth through Stochastic Cellular Automata. *Academy of Marketing Science Review*, 9(3):1–18, 2001.

[6] K. Jung, W. Heo, and W. Chen. IRIE: Scalable and Robust Influence Maximization in Social Networks. In *ICDM*, pages 918–923, 2012.

[7] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the Spread of Influence through a Social Network. In *KDD*, pages 137–146, 2003.

[8] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[9] N. Ohsaka, T. Akiba, Y. Yoshida, and K. Kawarabayashi. Fast and Accurate Influence Maximization on Large Networks with Pruned Monte-Carlo Simulations. In *AAAI*, pages 138–144, 2014.

[10] M. Richardson and P. Domingos. Mining Knowledge-Sharing Sites for Viral Marketing. In *KDD*, pages 61–70, 2002.