Compact and Scalable Graph Neighborhood Sketching

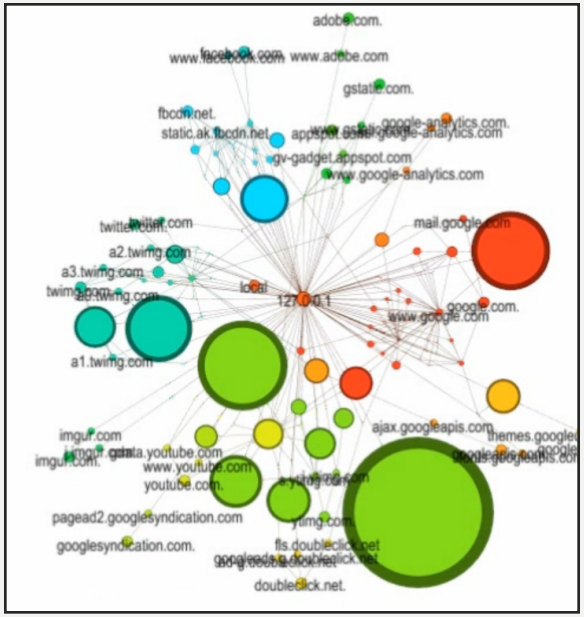
(KDD'16)

秋葉 拓哉 (Preferred Networks)

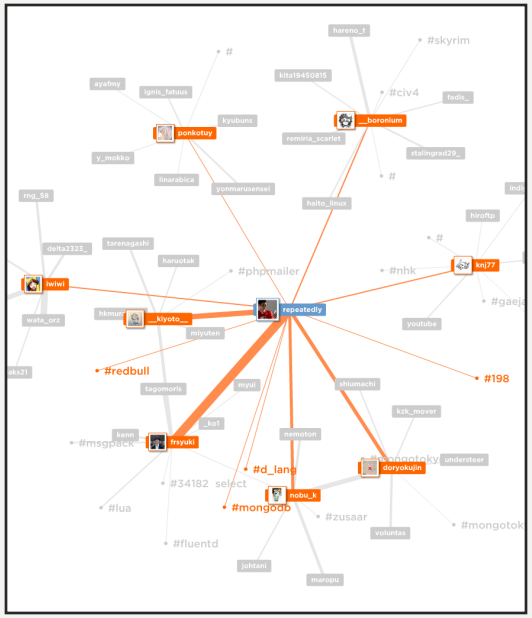
矢野 洋祐 (リクルートホールディングス)

2016/08/09 @ ERATO 感謝祭 Season III

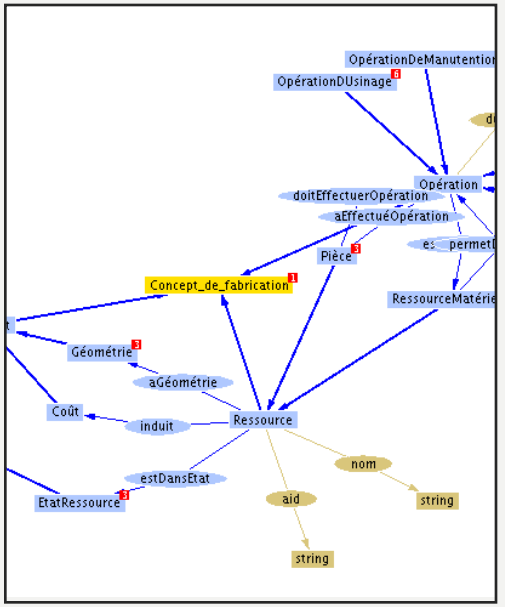
現代の大規模グラフの例



ウェブグラフ



ソーシャルグラフ



などなど……

グラフから得たい情報

1. 頂点の重要度
2. 頂点間の類似度・関連度
3. グラフの性質
4. ……

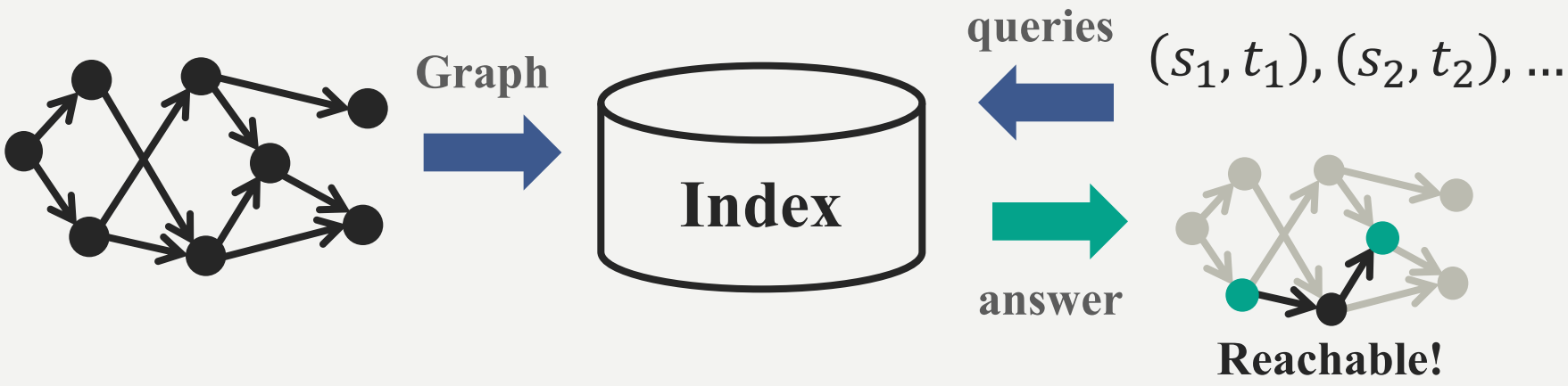
例：

- 検索サービス (Web 検索, SNS 内の検索)
- ソーシャルメディア解析

テキストのみからは得にくい情報がある

グラフ上のクエリ処理

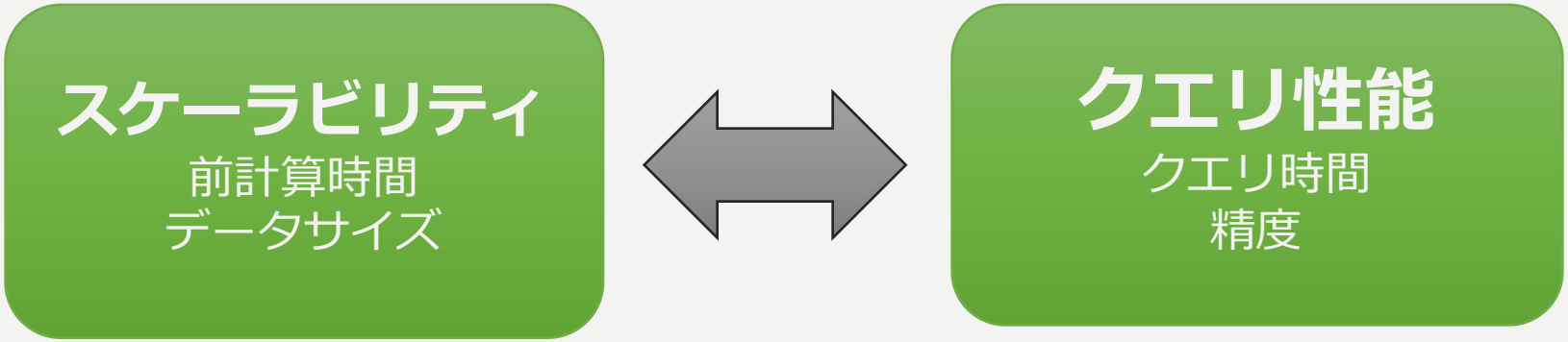
1. **索引構築**
   - グラフからデータ構造を前計算しておき
2. **クエリ応答**
   - それを用いて質問に高速に答える



グラフ上のクエリ処理

1. **索引構築**
   - グラフからデータ構造を前計算しておき
2. **クエリ応答**
   - それを用いて質問に高速に答える

目標: 良好なトレードオフ



グラフのインデクシング・スケッチング

これまでの多くの研究

- 特定の問題のみを扱う索引
  - 例：最短経路のみ，近接中心性のみ
- 理論的な bound なし
  - 計算量 and/or 精度

All-Distances Sketches (ADS) [Cohen'97, Cohen'15]

- **様々な値**を推定できる
- **精度保証有り**で答えられる
- ほぼ**線形時間**で計算・データはほぼ**線形空間**

理論的には極めて理想的な性質

ADS で推定可能な指標

- Neighborhood Function
- Shortest-Path Distance
- Closeness Centrality
- Closeness Similarity
- Average Distance, Effective Diameter
- Reverse Ranking, Reverse k-NN
- Continuous-Time Influence
- ……

ADS の問題点

理論的には「ほぼ線形」であっても

実用的にはかなりサイズが大きい

例

- 7M 頂点，200M 辺のウェブグラフ
- グラフ自体は **800MB**
- ADS は **4GB** $(k = 16)$

控えめな精度パラメータでも
グラフ自体より随分大きい

提案手法：Sketch Retrieval Shortcuts

ADS より小さく，ADS と同等の推定性能

Sketch Retrieval Shortcuts (SRS)

- ADS より小さいスケッチ：**1/3 ～ 1/10**
- 各頂点の ADS を瞬時に復元：**数 ms**
- 復元した ADS を用いて同様の推定が可能

目次

1: イントロダクション（ここまで）

2: All-Distances Sketches [Cohen+'15]

3: Sketch Retrieval Shortcuts

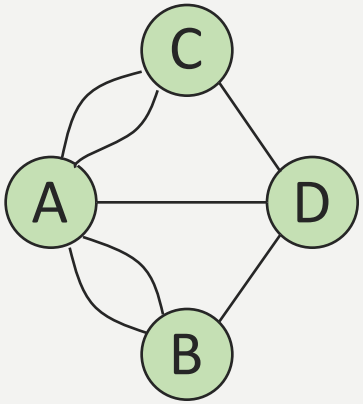
4: 実験結果

All-Distances Sketches [Cohen'15]

Part 2

前提

グラフ $G = (V, E)$ を持っている
（無向グラフと仮定）



$n = |V|, m = |E|$

全頂点 $v \in V$ に対して，
スケッチ $\mathbf{ADS}(v)$ を前計算＆保存する．
これを使って色々する．

ADS の使い方

前計算

持っているグラフデータから，**全頂点の ADS** を計算する．

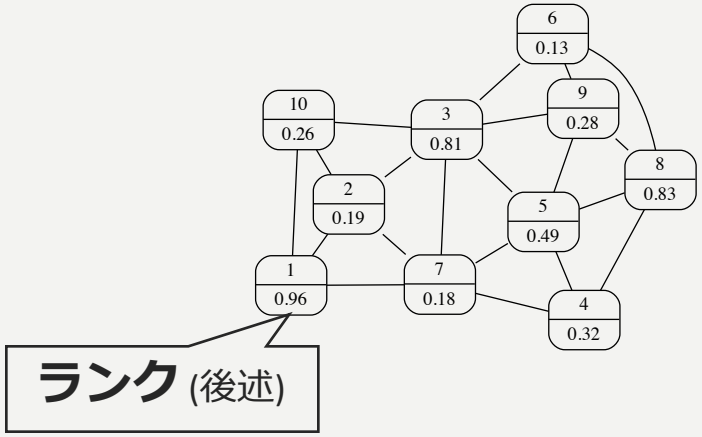
スケッチ: $\{\mathbf{ADS}(v)\}_{v \in V}$

使用

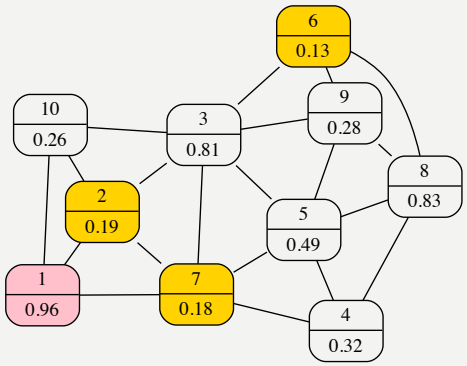
ADS を使って色々な値を高速に推定する．

推定可能：Neighborhood Function, Shortest-Path Distance, Closeness Centrality, Closeness Similarity, Average Distance, Effective Diameter, Reverse Ranking, Reverse k-NN, Continuous-Time Influence, ……

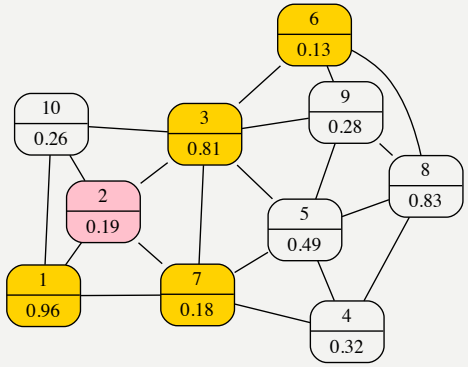
All-Distances Sketches の定義



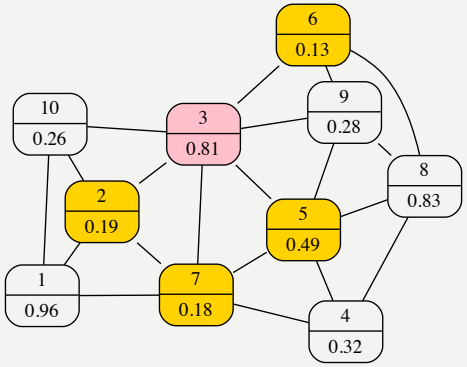
$$\{\mathrm{ADS}(v)\}_{v \in V}$$



$$\mathrm{ADS}(1) = \{2, 7, 6\}$$



$$\mathrm{ADS}(2) = \{1, 3, 6, 7\}$$



$$\mathrm{ADS}(3) = \{2,5,6,7\}$$

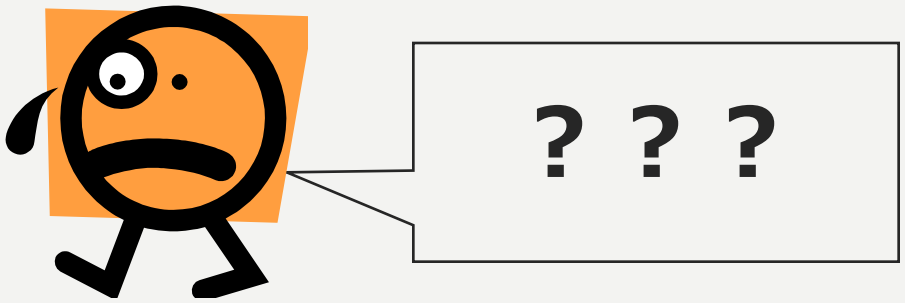
All-Distances Sketches の定義

定義 1

$$\mathrm{ADS}(v) \coloneqq \left\{ u \mid r(u) < \mathrm{k}_r^{\mathrm{th}}\left(\Phi_{<u}(v)\right) \right\}$$

定義 2

任意の距離 d に対する近傍関数 $N_d(v)$ の Bottom-k MinHash の和集合



ランク（ハッシュ関数）

- $r(v)$：各頂点の乱数ハッシュ値 [0, 1]

「ランク」と呼びます

イメージ：

ランクの小さい頂点ほど**偉い**！

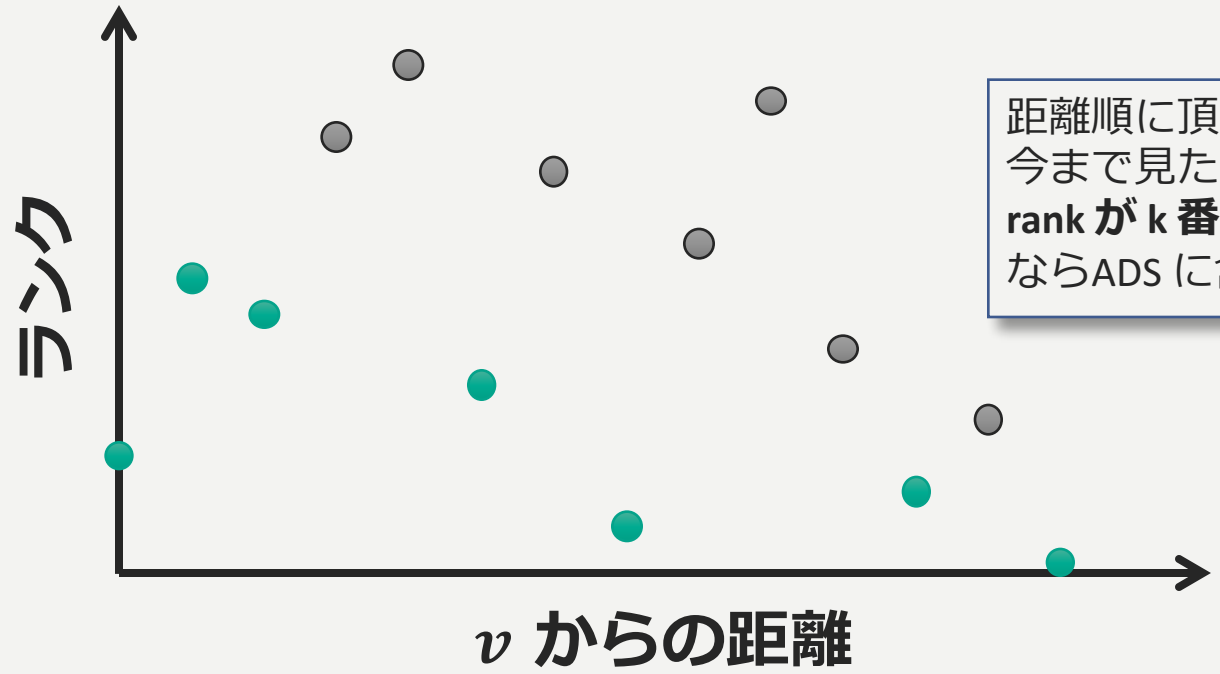
- 偉い＝保存されやすい
  （Min-Hash でもハッシュ値最低のやつが保存されていた）

$\mathrm{ADS}(v)$ の視覚的な説明

$k = 3$ の例

(MinHash 同様にパラメータがある)

$\mathrm{ADS}(v) =$ 自分より左下に点が k 個未満の頂点



距離順に頂点を見て行って，
今まで見た頂点の中で
rank が k 番目以内
ならADS に含む

性質

- ランク小さい方から k 個は必ず含まれる
- 近い方から k 個も必ず含まれる
- 遠くなってゆくほど含まれにくくなっていく

ADS のサイズ

$$|\mathrm{ADS}(v)| \cong k(1 + \ln n - \ln k) = O(k \log n)$$

k を大きくするほど
スペースを使う
しかし精度が良くなる（後述）

よって，合計の使用メモリは：

$$O(kn \log n)$$

k を定数とすると，スペースは頂点数に**ほぼ線形**
（→ 理論的には高いスケーラビリティ）

ADS の計算

詳細は省略しますが難しくない

1. 枝刈り Dijkstra を全頂点から実行
2. 動的計画法

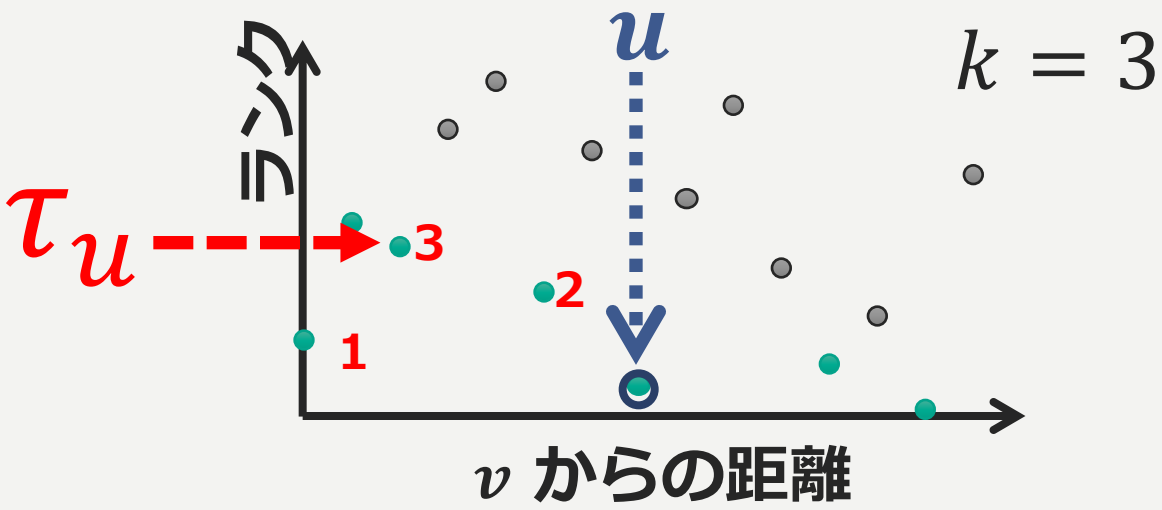
$O\big((m + n \log k)\, k \log n\big)$ 時間で構築可

ADS による近接関数の推定

HIP Estimator [Cohen, PODS'14]

$$\widetilde{n_d}(v) = \sum_{u \in \mathrm{ADS}(v),\, d_G(v,u) \le d} \frac{1}{\tau_u}$$

各 u について，
そいつより左で k 番目に
小さいランク



ADS による近接関数の推定

HIP Estimator

$$\widetilde{n_d}(v) = \sum_{u \in \mathbf{ADS}(v),\, d_G(v,u) \le d} \frac{1}{\tau_u}$$

v の ADS だけから推定可

精度

$$CV \le 1/\sqrt{2(k-1)}$$

（CV = 標準偏差 / 平均）

Sketch Retrieval Shortcuts（提案手法）

Part 3

概要：Sketch Retrieval Shortcuts

アイディア

1. ADS を新たなグラフ G_{ADS} とみなす
2. G_{ADS} から $\mathrm{ADS}(v)$ を取得するアルゴリズムを定義
3. 取得アルゴリズムが正しく動作するために
   必要十分な辺のみにする

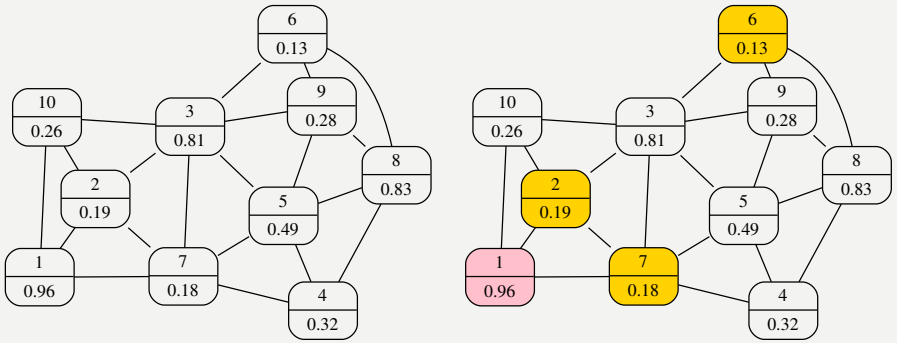
ADS の全体よりも遥かに小さいデータ構造となるが
ADS が瞬時に取得できる！

考え方：ADS 取得の難しさ

前提：頂点 v の ADS(v) を必要に応じて取り出したい

- **全頂点保存しておく** → 容量多すぎ
- **前処理なし** → グラフ全体の探索が必要，遅すぎ
  下図のように，「飛び地」があるので，グラフ全体を調べる必要がある

うまく中間を取れないか？



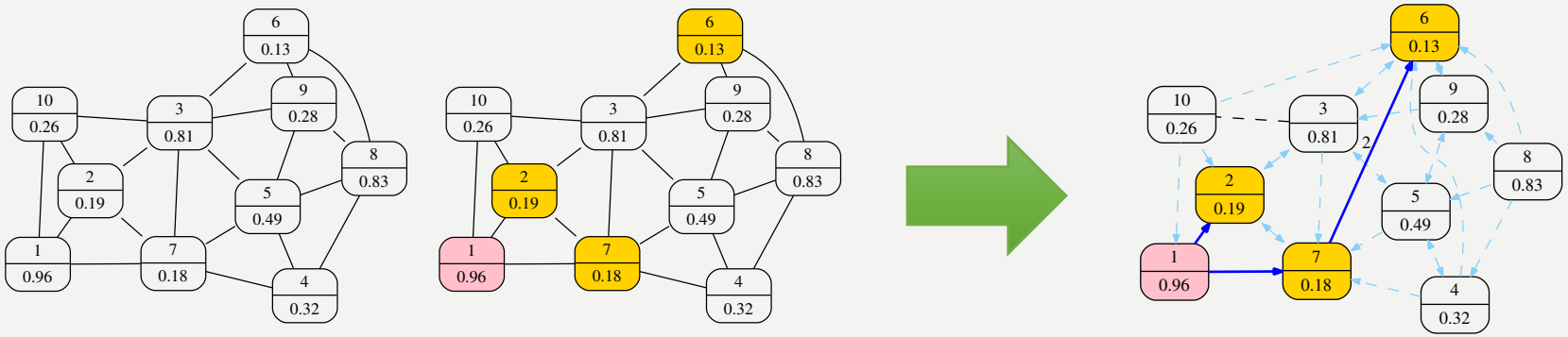
(a) A graph. (b) The vertices in $\mathcal{A}(1)$.

Figure 1: Examples of a graph and an ADS ($k = 2$).

考え方：制限された探索による ADS 取得

目標：ADS 取得時にちょっと計算をしてもいい

- グラフ全体の探索をしたくない
  大規模なグラフのサイズに比例した計算量は無謀
- ADS 内の頂点のみを訪れる探索で取得できるか？
  勿論，これは元のグラフ上では実現できない



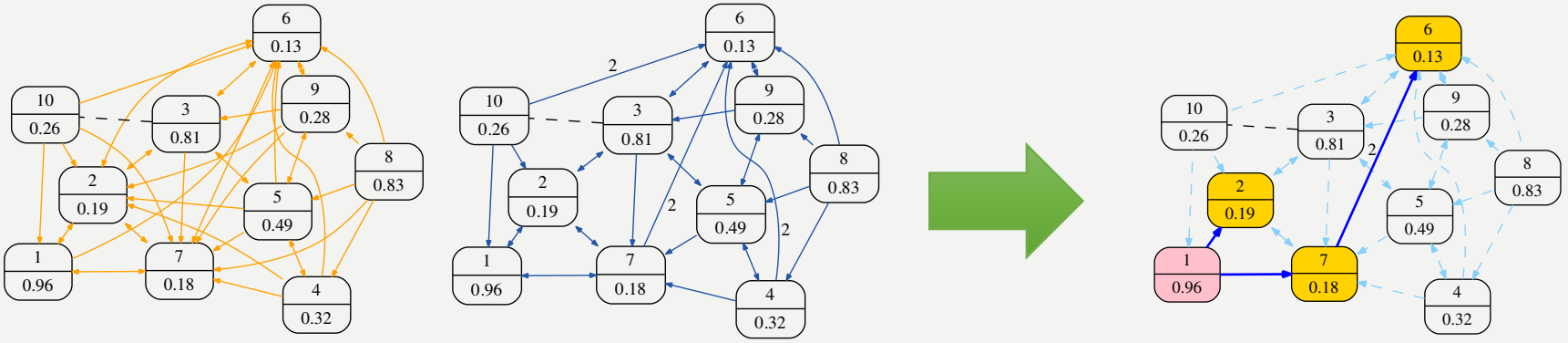
(a) A graph. (b) The vertices in $\mathcal{A}(1)$.

Figure 1: Examples of a graph and an ADS ($k = 2$).

考え方：グラフとしての ADS と SRS

ADS 内の頂点のみを訪れる探索をしたい？

- ADS をグラフだと思うと，これは**成功**する
  当然．全 ADS 内の頂点に 1 hop で到達．
- 無くても成功する辺がいっぱいある
- **必要最低限の辺に選別**したのが **SRS**（提案手法）



(a) ADSs as a graph. (b) SRSs as a graph.

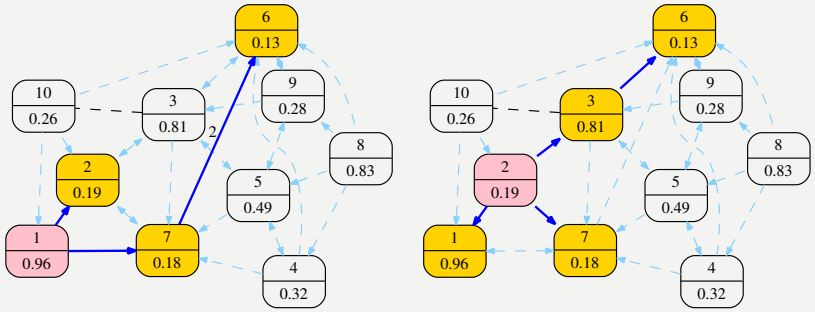
Figure 2: ADSs and SRSs as graphs.

アルゴリズム：SRS から ADS を取得

Algorithm 1: Retrieving the ADS of vertex u

Procedure Retrieve-ADS$(\mathcal{B}, u, k)$

1 $A \leftarrow$ an empty all-distances sketch;
2 $Q \leftarrow$ a priority queue with only one element $(0, u)$;
3 **while** Q is not empty **do**
4 $(\delta_{uv}, v) \leftarrow Q$.Pop;
5 **if** $u \notin A$ and $r(v) < \pi(u, v)$ **then**
6 Add (v, δ_{uv}) to A;
7 **for all** $(\delta_{vw}, w) \in \mathcal{B}(v)$ **do**
8 Q.Push$(\delta_{uv} + \delta_{vw}, w)$;
9 **return** A;



(a) Retrieving $\mathcal{A}(1)$. (b) Retrieving $\mathcal{A}(2)$.

Figure 3: Retrieval of ADSs from SRSs.

- SRS 上で Dijkstra のアルゴリズムを実行する
- ただし，ADS に含まれない頂点は訪問しない
  重要：ADS に含まれるか否かは探索中の情報のみで判断可能！
- $\tilde{O}(k^2)$ 時間で動作
  グラフサイズの依存が polylog!

アルゴリズム：グラフから SRS 構築

Algorithm 2: Constructing SRSs from ADSs (naive).

Procedure Construct-SRS-Naive($G = (V, E), \mathcal{A}, k$)

```
1  B[u] ← ∅ for all u ∈ V;
2  T ← {(δ_uv, v, u) | (v, δ_uv) ∈ A(u)};
3  Sort T;
4  for (δ_uv, v, u) ∈ T do
5      A ← Retrieve-ADS(B, u, k);
6      if (v, δ_uv) ∉ A then Add (v, δ_uv) to B[u];
7  return B;
```

構築時に取得のアルゴリズムを利用

- 全 ADS を距離が短い順にソート
- 各エントリを SRS に入れるかを判定
  SRS を小さい距離の部分から構築してゆく
- 判定には取得アルゴリズムを利用
  取得によって既にそのエントリが得られるのであれば必要ない

枝刈りラベリング法 [Akiba+,SIGMOD'13]と類似

省略すること

定義と性質

- SRS の数学的定義とその性質

構築アルゴリズム

- より高速な構築アルゴリズム
- ADS を経由しない省メモリな構築アルゴリズム
- 並列化

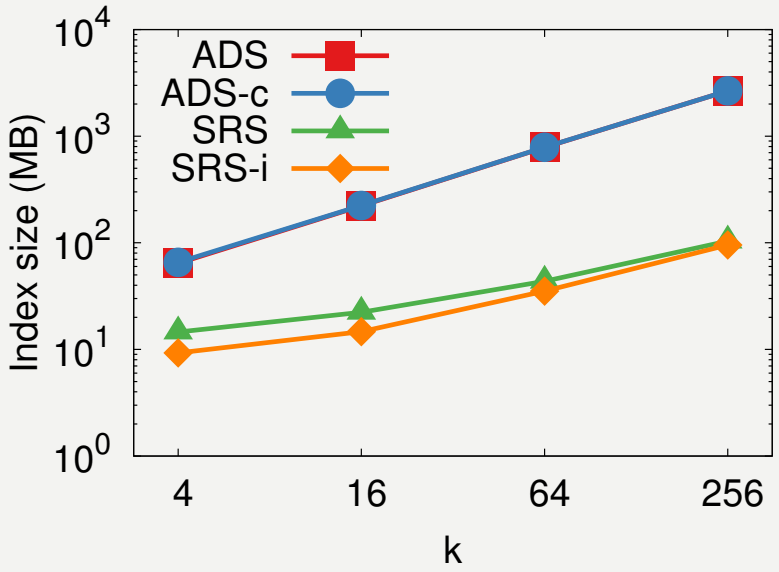
更なるサイズ縮減

- 元グラフでの近傍は除去する

実験結果

Part 4

サイズ

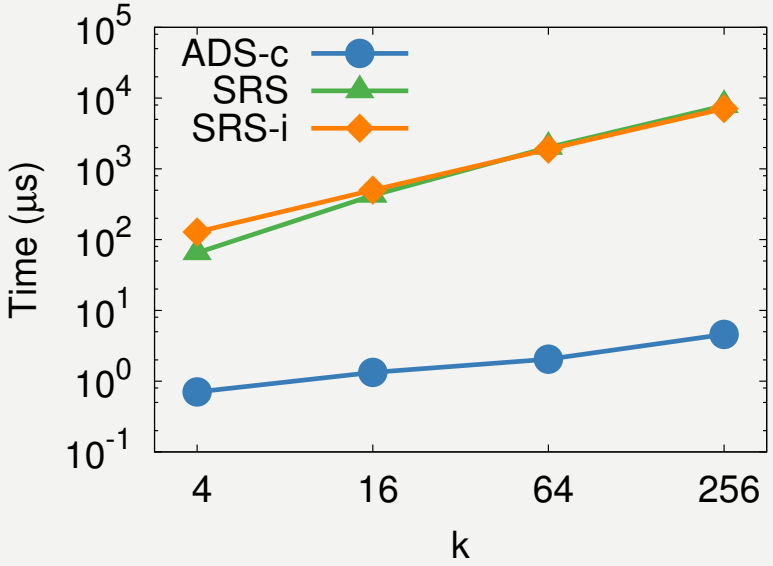


- **ADS**：通常の ADS
- **ADS-c**：ADS + LZ 圧縮 (snappy)
- **SRS**：通常の SRS
- **SRS-i** ：SRS + implicit neighbor

データ：com-DBLP ($|V| = 0.3$M, $|E| = 2$M)

- 汎用圧縮アルゴリズムはほぼ効果なし
  各 ADS が短く，反復が少ないため
- SRS は k が大きいほど ADS との差が大きい
  $k = 256$ で 10 倍以上の差

復元時間

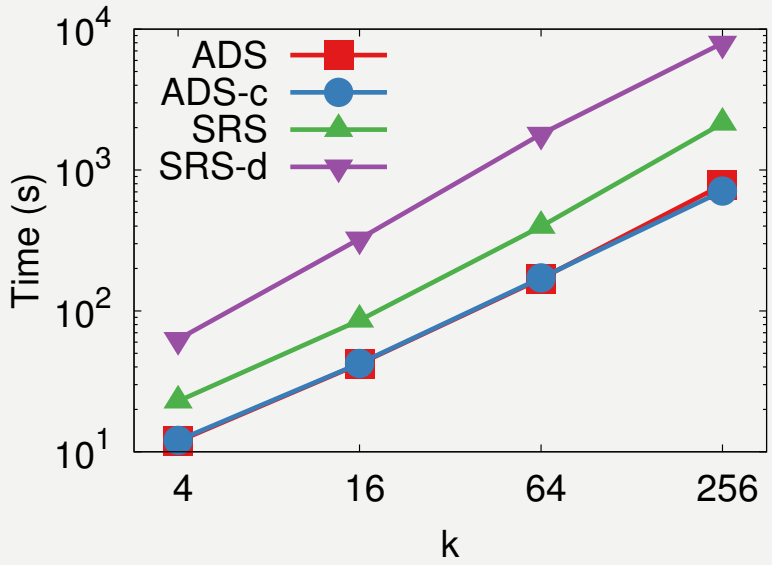


- **ADS-c**：ADS + LZ 圧縮 (snappy)
- **SRS**：通常の SRS
- **SRS-i** ：SRS + implicit neighbor

データ：com-DBLP ($|V| = 0.3$M, $|E| = 2$M)

- 大きい k でも数 ms 程度
  $k = 256$ で 8ms 程度
- 復元後は通常の ADS と同様に指標の推定が可能

構築時間



- **ADS-c**：ADS + LZ 圧縮 (snappy)
- **SRS**：ADS 経由の構築
- **SRS-d** ：直接構築（省スペース）

データ：com-DBLP ($|V| = 0.3$M, $|E| = 2$M)

スケーラビリティ

- **3M** 頂点，**200M** 辺のソーシャルグラフ
- **7M** 頂点，**200M** 辺のウェブグラフ
- 数時間～10時間 (k=16)

まとめ

まとめ

背景：All-Distances Sketches の有用性と課題

- 理論的には優れた性質を持つグラフスケッチ
- しかし，実用的には，サイズがとても大きい

提案手法：Sketch Retrieval Shortcuts

- ADS より小さいスケッチ：**1/3 ～ 1/10**
- 各頂点の ADS を瞬時に復元：**数 ms**
- 復元した ADS を用いて元と同様の推定が可能