大規模グラフのコンパクトでスケーラブルな全距離スケッチ

Compressed All-Distances Sketches for Large Graphs

秋葉拓哉 [1*]　矢野洋祐 [1,2]
Takuya Akiba[1,2]　Yosuke Yano[1,2]

[1] 国立情報学研究所
[1] National Institute of Informatics
[2] JST さきがけ
[2] JST, PRESTO

Abstract: The *all-distances sketch* is a useful neighborhood sampling scheme for large-scale graph analysis. In this study, we propose a compression technique to improve its space efficiency.

1 はじめに

All-distances sketches (ADS) [4, 5] は近年注目されるグラフに対するスケッチ手法である．全頂点に対する ADS はグラフサイズの線形に近い時間で構築を行うことができる．そして，ADS を用いると，近傍関数 [4,9,5]，距離 [6,11]，近接中心性 [5]，近接類似性 [6]，実効直径 [2]，逆最近傍探索 [3]，時間を考慮した影響拡散 [10,8,7] といった，グラフ解析に用いる様々な指標が，効率的かつ高精度で推定可能である．

しかし，ADS は上述の通り理論的には優れた性質を持つものの，実際にはデータサイズが大きすぎ実用的ではないということが分かってきている．各頂点の ADS は長さが約 $k \ln n$ の配列である．ここで，k は推定精度とサイズのトレードオフを設定するパラメータであり，n は頂点数を表す．高精度な推定のためには，k は数十から数百に設定されるため，ADS のサイズは対象のグラフ自体よりも大幅に大きなサイズとなってしまう．

そこで，本研究では，新たなグラフのスケッチ手法である *sketch retrieval shortcuts (SRS)* を提案する．SRS は，ADS と類似した形式のデータ構造であるが，ADS よりサイズが小さい．そして，SRS から各頂点の ADS を，必要に応じて高速に復元できる．復元した ADS は，上述の様々な指標の推定に，通常の ADS と全く同様に利用できる．

構成　本論文の構成は以下の通りである．2 章で前提となる表記や概念を説明する．3 章で提案手法を説明する．4 章で実験結果を示す．5 章で結論を述べる．なお，本研究についての詳細は文献 [1] を参照されたい．

*連絡先： 国立情報学研究所　東京都千代田区一ツ橋 2-1-2
E-mail: takiba@nii.ac.jp

2 前準備

2.1 表記

$G = (V, E)$ を，頂点集合を V，辺集合を E とする有向重みつきグラフとする．$|V|$ と $|E|$ はそれぞれ n と m と表記する．辺の重みは 0 より大きいと仮定する．$d(u, v)$ により頂点 u から v への距離を表す．$P(u, v)$ は頂点 u から v への最短経路上の頂点を表す．即ち，$P(u, v) = \{w \in V \mid d(u, w) + d(w, v) = d(u, v)\}$ である．

2.2 全距離スケッチ

All-distances sketches (ADS) は整数 k と頂点へのランク割当て r に基づき定義される．パラメータ k はデータサイズと推定精度のトレードオフを調整する．ランク割当て関数として $r : V \to [0, 1]$ を用いる．$r(v) \sim U[0, 1]$ とする．即ち，$r(v)$ は $[0, 1]$ の一様分布より選択される．

頂点 $u, v \in V$ について，$N(u, v)$ を u に関して頂点 v よりも近い頂点の集合とする．頂点集合 $X \subseteq V$ について，$k_r^{\text{th}}(X)$ を X 中で k 番目に小さいランク値とする．$|X| < k$ である場合は $k_r^{\text{th}}(X) = 1$ とする．頂点 u, v について，$\pi(u, v)$ を $\pi(u, v) = k_r^{\text{th}}(N(u, v))$, と定義する．ADS は以下のように定義される．

定義 1 (ADS [5])**.** 頂点 u の all-distances sketch (ADS) は $\mathcal{A}(u) = \{(v, \delta_{uv}) \mid v \in V, r(v) < \pi(u, v)\}$ である．ここで，$\delta_{uv} = d(u, v)$ である．

$\mathcal{A}(u)$ の大きさの期待値は $O(k \log n)$ である [5].

3 提案手法

3.1 定義

提案手法 *Sketch retrieval shortcuts (SRS)* を定義する．ADS と同様に，SRS はグラフ $G = (V, E)$，パラメータ k, ランダムランク割当て $r : V \to [0, 1]$ に基づき定義される．

$\Delta = \{d(u, v) \mid u, v \in V\}$ とする．$d_0 < d_1 < \cdots < d_h$ とし $\Delta = \{d_0, d_1, \ldots, d_h\}$ とおく．ここで，$d_0 = 0$ であり d_h はグラフの直径である．$\mathcal{B}_i$ $(i = 0, 1, \ldots, h)$, $\mathcal{C}_i, \mathcal{D}_i$ $(i = 1, 2, \ldots, h)$ を以下のように定義する．

- $\mathcal{B}_0(u) = \emptyset$ かつ $\mathcal{B}_i(u) = \mathcal{B}_{i-1}(u) \cup \mathcal{D}_i(u)$ $(i > 0)$.
- $\mathcal{C}_i(u, v) = \{w \in P(u, v) \mid w \in \mathcal{A}(u), v \in \mathcal{B}_{i-1}(w)\}$.
- $\mathcal{D}_i(u) = \{(v, \delta_{uv}) \in \mathcal{A}(u) \mid \delta_{uv} = d_i, \mathcal{C}_i(u, v) = \emptyset\}$.

SRS は以下のように定義される．

定義 2 (SRS)**.** 頂点 u の Sketch retrieval shortcuts (SRS) は $\mathcal{B}_h(u)$ である．

以下，簡単のため，$\mathcal{B}_h(u)$ を $\mathcal{B}(u)$ と表記する．

定義より，$\mathcal{B}(u)$ は $\mathcal{A}(u)$ の部分集合である．従って，大きさは以下のように評価できる．

補題 3. $\mathcal{B}(u)$ の大きさの期待値は $O(k \log n)$ である．

3.2 SRS からの ADS の取得

SRS の主な機能は，任意の頂点の ADS を高速に再構築することである．SRS から ADS を取得することにより，前述の通り，グラフ解析のための様々な指標が，通常の ADS と同様に推定可能である．取得アルゴリズム Retrieve-ADS はアルゴリズム 1 である．頂点 u の ADS の取得は，SRS 上で $\mathcal{A}(u)$ に含まれる頂点のみに訪問するような最短経路探索に相当する．

アルゴリズム 1: 頂点 u の ADS の取得

```
Procedure Retrieve-ADS(B, u, k)
1   A   an empty all-distances sketch;
2   Q   a priority queue with only one element (0, u);
3   while Q is not empty do
4       (δ_uv, v) ← Q.Pop;
5       if u ∉ A and r(v) < π(u, v) then
6           Add (v, δ_uv) to A;
7           for all (δ_vw, w) ∈ B(v) do
8               Q.Push(δ_uv + δ_vw, w);
9   return A;
```

アルゴリズム Retrieve-ADS の期待計算量は $O(k^2 \log^2 n \log(k \log n))$ 時間である．

3.3 ADS からの SRS の構築

まず，ADS から SRS を構築するアルゴリズムを説明する．詳細はアルゴリズム 2 として記述した．SRS は距離について再帰的に定義されているため，SRS の構築も距離が小さいエントリから順に行う．そのため，全頂点の ADS の全エントリを，距離が小さいものから順に，SRS に必要か否かを判断してゆく．エントリが SRS に必要か否かの判断に ADS 取得アルゴリズム Retrieve-ADS を利用する点がこのアルゴリズムの興味深い点である．

アルゴリズム 2: ADS からの SRS の構築

```
Procedure Construct-SRS(G = (V, E), 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;
```

このアルゴリズムの期待計算量は $O(nk^3 \log^3 n \log(k \log n))$ 時間である．また，毎回 Retrieve-ADS を呼び出す代わりに，SRS の各エントリの挿入時に，取得可能な SRS エントリを陽に生成する工夫により，$O(nk \log n \log(nk \log n) + |\mathcal{B}| k \log n)$ 期待時間に改善する．ここで，$|\mathcal{B}|$ は SRS の合計サイズを表す．

3.4 直接的な SRS の構築

前述のアルゴリズム Construct-SRS は，一度 ADS の陽な構築を経由してしまうため，大きな作業領域をメモリ上に要求してしまう．そこで，以下では，ADS を陽に構築することのないアルゴリズム Construct-SRS-Direct を考える．アルゴリズムの詳細はアルゴリズム 3 として記述した．ここでは，簡単のため，グラフは重み無しを仮定する．このアルゴリズムは，伝搬による ADS 構築アルゴリズムを仮想的に実行しながら，前述の Retrieve-ADS によるエントリの要不要判定を行っている．

このアルゴリズム Construct-SRS-Direct の期待計算量は $O(D(n + m)k^2 \log^2 n \log(k \log n))$ 時間と $O(n + m + |\mathcal{B}| + k \log n)$ 空間である．ここで，D はグラフの直径を表す．ボトルネックとなる 7 行目の Retrieve-ADS の結果をキャッシュすることにより，期待時間計算量は $O(Dnk^2 \log^2 n \log(k \log n) + mk \log n)$ に改善する．一方，期待空間計算量は $O(n + m + |\mathcal{B}| + nk \log n)$ となるが，キャッシュするのは距離 $i - 1$ のエントリのみでよく，実験的には小さい．

アルゴリズム 3: SRS の直接的な構築

```
Procedure Construct-SRS-Direct(G = (V, E), k)
1   B[u] ← ∅ for all u ∈ V;
2   for i = 1, 2, ... do
3       f ← False;
4       for u ∈ V do
5           T ← ∅;
6           for v ∈ V such that (u, v) ∈ E do
7               A ← Retrieve-ADS(B, v, k);
8               for (w, δ_vw) ∈ A do
9                   if δ_vw = i − 1 then Add w to T;
10          Sort T;
11          A ← Retrieve-ADS(B, u, k);
12          for w ∈ T do
13              if r(w) ≥ π(u, w) then continue;
14              if w ∉ A then Add (w, i) to B[u];
15              f ← True;
16      if f = False then break;
17  return B;
```

3.5 近傍の省略

SRS を用いてグラフを解析する際には，元のグラフ自体もメモリ上に存在しアクセス可能である場合が多いと考えられる．そのような場合には，そのグラフ自体の情報を併用することにより，SRS のサイズを更に減らすことが可能である．即ち，SRS に含まれるエントリの中から，元のグラフの辺に合致するものは削除できる．近傍を削除した場合， ADS 取得時の最短経路探索では，SRS のエントリだけでなく元のグラフの辺も遷移に用いる．

4 評価実験

4.1 実験方法

本実験には CPU が Intel Xeon 2.67 GHz 2 ソケット，メモリが 96GB の Linux サーバを利用した．全てのアルゴリズムは C++ で実装され，gcc 4.8.4 を用いて最適化オプション `-O3` の設定でコンパイルされた．データセットとして実際のソーシャルグラフとウェブグラフを用いた（表 1）．SRS 構築アルゴリズムは並列化され 24 スレッドで実行した．$k = 16$ とした．

手法として以下の 5 つを比較する．*(1)* ADS は通常の ADS である．*(2)* ADS-c は LZ 系圧縮アルゴリズム (google-snappy) を適用した ADS である．*(3)* SRS は ADS 経由で SRS を構築する手法である (アルゴリズム 2)．*(4)* SRS-d は SRS の直接的な構築アルゴリズムである (アルゴリズム 3)．*(5)* SRS-i は 3.5 章で述べた近傍の省略を用いた SRS である．

表 1: 実験で用いたデータセット

名称	種別	$\|V\|$	$\|E\|$
email-Enron	Social (u)	36,692	367,662
com-dblp	Social (u)	317,080	2,099,732
web-Google	Web (d)	875,713	5,105,039
in-2004	Web (d)	1,382,870	16,917,053
flickr-links	Social (u)	1,715,256	31,101,563

4.2 実験結果

表 2 が実験結果を表す．

スケッチサイズ ADS-c が改善を達成していないことから，一般的な圧縮アルゴリズムはあまり効果的でないことが確認できる．一方，SRS は ADS に対し大幅な改善を達成している．

構築の時間と空間 SRS は ADS より構築に時間が掛かる．SRS-d は更に時間がかかる．一方，一般に SRS-d が最も小さな空間で構築を達成する．SRS の所要空間は ADS に比例するが効率的な操作のためのデータ構造により数倍大きくなってしまっている．

取得時間 SRS による ADS の平均取得時間は概ね 1 ミリ秒以下であり非常に高速である．SRS-i は SRS と比較すると低速になるものの数ミリ秒程度であり許容範囲であると考えられる．

5 おわりに

本研究では，all-distances sketches (ADS) の高速な取得のためのデータ構造 sketch retrieval shortcut (SRS) を提案した．各頂点の ADS は SRS より高速に復元でき，通常の ADS と同様に，様々なグラフ解析の指標の推定に利用可能である．

謝辞

本研究は日本学術振興会科学研究費補助金 (15H06828) 及び JST さきがけの支援を受けたものである．ここに記して謝意を表す．

参考文献

[1] T. Akiba and Y. Yano. Compact and scalable graph neighborhood sketching. Manuscript, 2016.

[2] P. Boldi, M. Rosa, and S. Vigna. HyperANF: Approximating the neighbourhood function of very large graphs on a budget. In *WWW*, pp. 625–634, 2011.

表 2: 左側は，スケッチのサイズと ADS の平均取得時間を表す．右側は，構築時の所要時間と所要空間を表す．

データセット	サイズ (MB)				構築空間 (MB)			
	ADS	ADS-c	SRS	SRS-i	ADS	ADS-c	SRS	SRS-d
email-Enron	19.46	19.63	1.53	0.56	59.11	59.40	182.85	40.57
com-dblp	222.15	223.73	22.30	14.66	529.67	532.26	1929.51	303.22
web-Google	451.38	455.01	78.42	58.45	1055.01	1052.66	3956.93	734.14
in-2004	597.66	603.18	138.63	92.68	1468.30	1489.94	5049.08	1272.42
flickr-links	1277.11	1285.42	59.56	16.84	2866.05	2893.68	11344.86	2045.52
	取得時間 (μs)				構築時間 (s)			
	ADS	ADS-c	SRS	SRS-i	ADS	ADS-c	SRS	SRS-d
email-Enron	—	0.84	182.71	538.69	2.26	2.40	5.32	11.46
com-dblp	—	1.33	413.76	494.32	45.83	46.12	84.60	324.12
web-Google	—	1.16	271.13	260.74	82.57	81.32	173.13	952.80
in-2004	—	0.91	160.90	189.82	64.67	66.12	171.73	3473.29
flickr-links	—	1.85	517.60	8163.60	341.40	319.07	599.38	1954.50

[3] E. Buchnik and E. Cohen. Reverse ranking by graph structure: Model and scalable algorithms. *CoRR*, abs/1506.02386, 2015.

[4] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. Syst. Sci.*, 55(3):441–453, 1997.

[5] E. Cohen. All-distances sketches, revisited: HIP estimators for massive graphs analysis. *IEEE TKDE*, 27(9):2320–2334, 2015.

[6] E. Cohen, D. Delling, F. Fuchs, A. V. Goldberg, M. Goldszmidt, and R. F. Werneck. Scalable similarity estimation in social networks: closeness, node labels, and random edge lengths. In *COSN*, pp. 131–142, 2013.

[7] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *CIKM*, pp. 629–638, 2014.

[8] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Timed influence: Computation and maximization. *CoRR*, abs/1410.6976, 2014.

[9] E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In *PODC*, pp. 225–234, 2007.

[10] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *NIPS*, pp. 3147–3155, 2013.

[11] M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.