A-003

大規模グラフの *spanner* を生成するストリーミングアルゴリズムの実装

石島 正大 † 中野 眞一 †

1. まえがき

計算機の能力は年々向上しているが, 扱うデータの規模はそれ以上に拡大している．小規模なグラフならば, グラフ全体をメモリに格納できる．一方, メモリに全体を格納できないような大規模グラフを HDD に格納すると, アクセスに非常に長い時間がかかってしまう．

大規模グラフ G の距離を近似的に保存するように, 辺を適切に間引いた部分グラフを, グラフ G の ***spanner*** という．グラフ G の 2 点 x, y 間の距離を $dist_G(x, y)$ と書く．$G = (V, E)$ と G の *spanner* G' が, 任意の 2 点 $x, y \in V$ について,

$$dist_{G'}(x, y) \leq t \cdot dist_G(x, y)$$

を満たすとき, G' を G の ***t-spanner*** という．*spanner* を求める *streaming* アルゴリズム [1] が知られている．本文はこのアルゴリズムを実装し, 様々な大規模グラフに対して, 得られる *spanner* の特徴を調べる．

2. 定義

グラフ $G = (V, E)$ は, 点の集合 $V = \{v_1, v_2, ..., v_n\}$ と辺の集合 $E = \{e_1, e_2, ..., e_m\}$ からなる．

グラフ G の 2 点 $x, y \in V$ 間の経路とは, x から y への辺列である．経路の距離とは, 経路上の辺の本数である．G の 2 点, x, y 間の, 距離が最小の経路を, x, y 間の最短経路という．また, G の 2 点 x, y 間の最短経路の距離を, x, y 間の距離といい, $dist_G(x, y)$ と書く．

t-spanner の定義より, G において 2 点 x, y 間の距離が d のとき, G の *t-spanner* G' において 2 点 x, y 間の距離は高々td である.(図 1 参照.)

ループを持たない連結グラフを**木**という．1 点を根として指定した木を**根つき木**という．

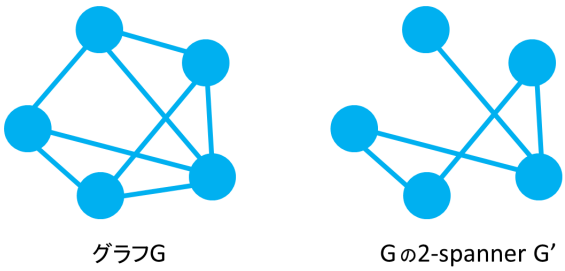

図 1: G と G の 2-*spanner* G' の例

3. アルゴリズム

グラフの $(2t-1)$-*spanner* を求めるストリーミングアルゴリズム [1] を説明する．このアルゴリズムは, 前処理ののち, グラフの各辺を順に入力として受け取る．各辺 e が入力される度に, e を *spanner* の辺として採用するか, それとも捨てるか, を高速に判定する．このアルゴリズムは, いくつかの木を保持する．また各点 v に, (1) 木の辺の集合 $T(v)$ と, (2) 木と点 v をつなぐ辺の集合 $X(v)$ を保持する．これら全ての辺から誘導されるグラフを $(2t-1)$-*spanner* として最後に出力する．

アルゴリズムは, グラフの各点 v に次の 4 つの情報を保持する．点番号 $I(v)$, 最大距離 $r(v)$, ラベル $P(v)$, 表 $M(v)$ である．

$I(v)$ と $r(v)$ の値はアルゴリズムの実行中は不変である．

$I(v) \in \{0, 1, ..., n-1\}$ とする．$I(v)$ は各点の一意的な識別子である．

アルゴリズムは, いくつかの根つき木を常に保持し, 少しづつ成長させる． v を根とする木 T_v は, T_v において根 v からの距離が最大 $r(v)$ までの点のみを含むことができる．非負の整数 $r(v)$ は, 次の幾何学的確率分布に従い, 前処理のときに各点 v に $r(v)$ を割り当てる．$r(v)$ に k を割り当てる確率を $P(r(v) = k)$ と書く．各 $k \in \{0, 1, ..., t-2\}$ について $P(r(v) = k) = p^k \cdot (1-p)$ とする．特に $k = t-1$ について $P(r(v) = t-1) = p^k$ とする．ここで $p = (\frac{\log n}{n})^{1/t}$ とする．

ラベル $P(v)$ は, 点 v が現在属している木の根の点番号 $B(v)$, およびその木における根から v までの距離 $L(v)$ からなる．$P(v) = n \cdot L(v) + B(v)$ とする．前処理のときに $L(v) = 0$, $B(v) = I(v)$ とし, すなわち $P(v) = I(v)$ とする．これは各点 v が, v を根とする 1 点のみからなる根つき木に属していることを示す．アルゴリズムはこれらの木を次第に成長させていく．

表 $M(v)$ は空集合に初期化する．アルゴリズム実行中に辺 (u, v) を $X(v)$ に追加する度に, $M(v)$ に $B(u)$ を追加する．すなわち点 v から点 u を根とする木 T_u への辺が $X(v)$ 中にすでにあるかどうかを記憶する.(図 4 参照.)

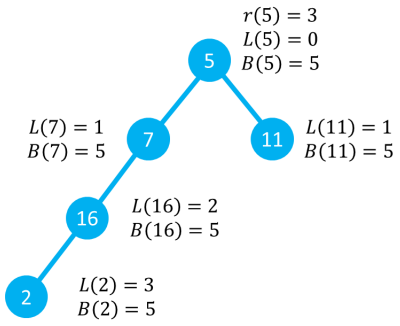

図 2: $L(v)$ と $B(v)$ の例

(1)$P(v) > P(v')$, または, (2)$P(v) = P(v')$ かつ

†群馬大学大学院工学研究科

$I(v) > I(v')$, のとき $P(v) \succ P(v')$ と書く．$I(v)$ は一意であるので, 任意の 2 点 v, v' について, $P(v) \succ P(v')$ または $P(v') \succ P(v)$ のいずれかとなる．

論文 [1] のアルゴリズムを次に示す．

Read_Edge $e = (u, v)$

1: *Let* u *be the vertex s.t.* $P(u) \succ P(v)$
2: **if** $L(u) < r(B(u))$ **then**
/ もし u の属する木に点 v を追加できるならば /
3: $P(v) \leftarrow P(u) + n$
/ 点 u の属する木に点 v を追加する．
$B(v) \leftarrow B(u)$, $L(v) \leftarrow L(u) + 1$ となる．/
4: $T(v) \leftarrow T(v) \cup \{e\}$
/ 辺 e を *spanner* に含める /
5: **else if** $B(u) \notin M(v)$**then**
6: $M(v) \leftarrow M(v) \cup \{B(u)\}$
/ 点 v に u の属する木への辺を追加 /
7: $X(v) \leftarrow X(v) \cup \{e\}$
8: **end if** / 辺 e は捨てる /

まず, 辺 $e = (u, v)$ について, $P(u) \succ P(v)$ のように u, v を定める．直観的には, 根からの距離が大きい方の点を u とし, u の属する木に点 v を追加する．ただし v の追加後に, u の属する木の根から v までの距離が最大距離 $r(B(u))$ 以下である場合に限り, u の属する木に点 v を追加する．このとき辺 e を $T(v)$ に追加し, *spanner* の辺とする (図 3 参照.)

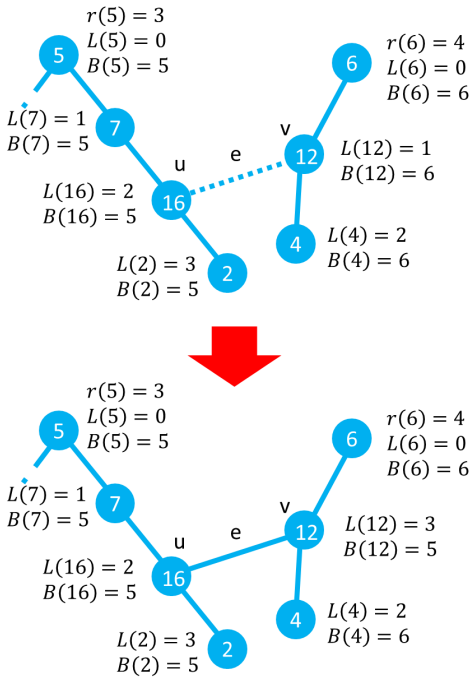

図 3: アルゴリズムの説明図

一方, u の属する木に v を追加すると, u の属する木の根から v までの距離が最大距離 $r(B(u))$ より大きくなってしまうときは, u の属する木と点 v をつなぐ辺が $X(v)$ 中にない時に限り, e を $X(v)$ に追加し, *spanner* の辺とする．もし点 u の属している木と点 v をつなぐ辺が $X(v)$ の中にすでにあるのであれば, 辺 e は捨てる.(図 4 参照.)

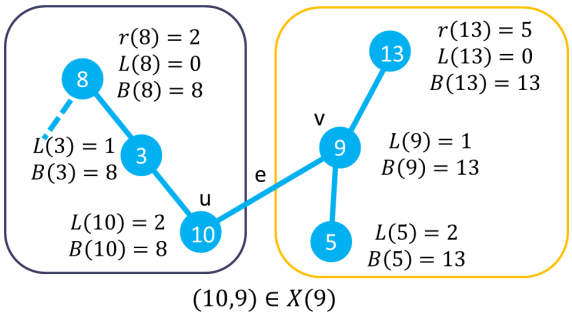

図 4: アルゴリズムの説明図

最後に得られた $\cup_{v \in V} T(v) \cup \cup_{v \in V} X(v)$ を, $(2t-1)$-*spanner* として出力する．

4. 実装と結果

このアルゴリズムを C 言語で実装した．実行環境は *Core i7*(2.00*GHz*), メモリ 8*GB* である．大規模グラフデータとして [2][3] を用いた

4.1. 実行結果

入力グラフを $G = (V, E)$ とし, *spanner* を $G' = (V, E')$ とする． $|E'|/|E| \times 100$ を圧縮率とする．$\sum_{(u,v) \in E} (dist_{G'}(u, v)/dist_G(u, v))$ を平均 *stretch* 値とする．$(dist_{G'}(u, v)/dist_G(u, v)) = k$ なる辺 $e \in E$ を k-*stretch* 辺と呼ぶ．このとき辺 e の *stretch* 値は k であるという．

いろいろな大規模グラフに対して, 3-*spanner* では辺数を 76〜98% に, 5-*spanner* では辺数を 33〜79% に圧縮できた．代表的な結果を以下に示す．

```
データファイル名          :Slashdot0902
元のグラフの点の個数      :82168
元のグラフの辺の本数      :504230
元のグラフの平均次数      :12.3

3-spanner
    辺の本数              :473935
    圧縮率                :94.0 %
    平均次数              :11.5
    平均 stretch 値       :1.07
    1-stretch 辺の本数    :473935
    2-stretch 辺の本数    :24983
    3-stretch 辺の本数    :5312
5-spanner
    辺の本数              :367259
    圧縮率                :72.8 %
    平均次数              :8.9
    平均 stretch 値       :1.44
```

```
    1-stretch 辺の本数  :367259
    2-stretch 辺の本数  :54115
    3-stretch 辺の本数  :80220
    4-stretch 辺の本数  :2619
    5-stretch 辺の本数  :17
7-spanner
    辺の本数            :267880
    圧縮率              :53.1 %
    平均次数            :6.5
    平均 stretch 値     :1.95
    1-stretch 辺の本数  :267880
    2-stretch 辺の本数  :38099
    3-stretch 辺の本数  :155932
    4-stretch 辺の本数  :41059
    5-stretch 辺の本数  :1260
    6-stretch 辺の本数  :0
    7-stretch 辺の本数  :0
9-spanner
    辺の本数            :216940
    圧縮率              :43.0 %
    平均次数            :5.3
    平均 stretch 値     :2.25
    1-stretch 辺の本数  :216940
    2-stretch 辺の本数  :38331
    3-stretch 辺の本数  :158870
    4-stretch 辺の本数  :83975
    5-stretch 辺の本数  :6078
    6-stretch 辺の本数  :38
    7-stretch 辺の本数  :0
    8-stretch 辺の本数  :0
    9-stretch 辺の本数  :0
```

```
データファイル名        :Amazon0601.txt
元のグラフの点の個数    :403394
元のグラフの辺の本数    :2443408
元のグラフの平均次数    :12.1

3-spanner
    辺の本数            :2394348
    圧縮率              :98.0 %
    平均次数            :11.9
    平均 stretch 値     :1.02
    1-stretch 辺の本数  :2394348
    2-stretch 辺の本数  :48092
    3-stretch 辺の本数  :968
5-spanner
    辺の本数            :2237050
    圧縮率              :91.6 %
    平均次数            :11.1
    平均 stretch 値     :1.09
    1-stretch 辺の本数  :2237050
    2-stretch 辺の本数  :191369
    3-stretch 辺の本数  :14042
    4-stretch 辺の本数  :927
    5-stretch 辺の本数  :20
7-spanner
    辺の本数            :2073020
    圧縮率              :84.8 %
    平均次数            :10.3
    平均 stretch 値     :1.17
    1-stretch 辺の本数  :2073020
    2-stretch 辺の本数  :327907
    3-stretch 辺の本数  :39377
    4-stretch 辺の本数  :2993
    5-stretch 辺の本数  :110
    6-stretch 辺の本数  :1
    7-stretch 辺の本数  :0
9-spanner
    辺の本数            :1938520
    圧縮率              :79.3 %
    平均次数            :9.6
    平均 stretch 値     :1.24
    1-stretch 辺の本数  :1938520
    2-stretch 辺の本数  :428951
    3-stretch 辺の本数  :68345
    4-stretch 辺の本数  :7110
    5-stretch 辺の本数  :457
    6-stretch 辺の本数  :24
    7-stretch 辺の本数  :1
    8-stretch 辺の本数  :0
    9-stretch 辺の本数  :0
```

5. 考察

$(2t-1)$-*spanner* 中に, $(2t-1)$-*stretch* 辺はほとんどなく, また *stretch* 値が大きい辺は極めて少ないことを確認した. よって後処理として, 得られた *spanner* に, *stretch* 値の大きな少量の辺を追加することにより, 最大の *stretch* 値の値を下げた *spanner* を得ることができる.

平均次数がほぼ等しいグラフであっても, 圧縮率が非常に異なるグラフがあった. 辺を入力する順は, 得られる *spanner* に顕著な影響を与えないことも確認できた.

参考文献

[1] Michael Elkin. “Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners”, ACM Transactions on Algorithms, Vol.7, Issue 2, Article 20, 2011

[2] “Stanford Large Network Dataset Collection” http://snap.stanford.edu/data/index.html

[3] “Laboratory for Web Algorithmics” http://law.dsi.unimi.it/datasets.php