CD-001

空間データにおける重み付き三角形の効率的な Top-k 検索アルゴリズム

谷口凌亮 *　天方大地 *　原隆浩 *
Ryosuke Taniguchi　Daichi Amagata　Takahiro Hara

抄録

近年，位置情報を扱うデバイスの増加に伴い，位置情報をもつデータが大量に生成されている．位置情報付きデータの分析は，コミュニティ検出によるグループチャットの推薦などに応用できる．位置情報付きデータの分析のため，位置が近いデータ同士をエッジで結んだグラフの部分グラフに注目し，データ同士の距離を重みとしたときに重みの小さい三角形を列挙することがある．そこで，本研究では空間データから辺の重みが最も小さい k 個の三角形を列挙する新しいアルゴリズムを提案する．実データを用いた実験により，提案アルゴリズムは既存アルゴリズムと比べて大幅に高速であることを確認した．

1　はじめに

近年，位置情報を扱うデバイスの増加に伴い，位置情報をもつデータが大量に生成されている．位置情報付きデータを分析するため，位置が近いデータ同士を結んだグラフを利用するものがある [16, 23, 26]．これにより，データ同士の関係を直感的に理解できる．グラフの分析手法は数多く存在するが，その一つに部分グラフを利用する方法がある．部分グラフを分析することにより，大規模なグラフの構成要素や構造が分かる [6, 14, 15]．また，様々な部分グラフの中でも三辺の重みの和が最も小さい三角形を列挙することは重要である．このような三角形に注目することによりコミュニティ検出 [13, 21] やネットワークの比較 [18, 22] などができる．

例 1. 位置情報を利用したゲーム（Pokemon GO[1] および Ingress[2] など）において位置情報が近いプレイヤ同士のチームを推薦することを考える．位置情報が近いプレイヤ同士でチームを組むことによりコミュニケーションが図りやすくなり，プレイヤのゲーム体験を向上できる．そのために，プレイヤの位置情報を空間データとして三辺の重みの和が小さい上位 k 個の三角形を列挙すれば，互いに位置が近いプレイヤを特定できる．

実データのグラフのエッジには重みが付加されている [5]．エッジの重みはソーシャルネットワークでは結びつきの強さ [17] を表していたり，交通ネットワークでは交通量 [11] を表していたりする．しかし，既存アルゴリズムの多くは重み無しグラフに対するアルゴリズムであるため，既存アルゴリズムを用いて三辺の重みの和が最も小さい三角形を列挙することは難しい [4, 8, 27, 28]．また，単純に空間データから三辺の重みの和が小さい上位 k 個の三角形を列挙するには，n 個の空間データに対して ${}_nC_3$ 個の三角形の重みを計算し，それらをソートする必要がある．しかし，この方法では n が大きいときに極めて低速である．既存研究において，重み付きグラフを入力とし，すべてのエッジを重みの降順でソートすることにより解となる三角形を高速に列挙する手法がある [12]．ソートは高速なアルゴリズム [9, 10] であるが，点集合から構築されたグラフのすべてのエッジをソートするコストは非常に大きい．したがって，この既存アルゴリズムでは点集合から重みの小さい三角形を高速に列挙できない．そこで本研究では，空間データから三辺の重みの和が最も小さい三角形を高速に列挙するアルゴリズムを提案する．

提案アルゴリズムでは，前処理としてグラフを構築し，全ての点に対して自身と自身から最も近い距離の点および二番目に近い点で構成される三角形を構成してその三角形の辺の重みの和を昇順でソートしておく．そして，ユーザから与えられるパラメータにより解の閾値を設定する．次に，解に含まれない三角形の頂点となる点を閾値を用いて枝刈りし，候補内の全ての点に対して解になり得る三角形が存在するか調べる．提案アルゴリズムは，点の枝刈りと解の候補となる三角形の存在の調査を繰り

*大阪大学大学院情報科学研究科マルチメディア工学専攻
[1] https://www.pokemongo.jp/
[2] https://www.ingress.com/ja/

返すことにより，考慮する必要のない三角形を除外しながら上位 k 個の重みの小さい三角形を正確かつ効率的に列挙できる．

本研究の貢献は以下の通りである．

- 点集合を入力として，三辺の重みの和が小さい上位 k 個の三角形を高速に列挙できるアルゴリズムを提案する．
- 実世界のデータを用いた実験により，提案アルゴリズムの有効性を示す．

以降では，2 章では予備知識を説明し，本研究の問題を定義する．3 章では，大規模なグラフから部分グラフを列挙する手法，および大規模なグラフから三角形の数を推定する手法などの関連研究を紹介する．4 章では，提案したアルゴリズムを詳細に説明する．5 章では，実データを用いて提案アルゴリズムと既存アルゴリズムとの比較実験を行い，その結果について報告する．最後に，6 章で本研究のまとめを行う．

2 予備知識

与えられた点集合を $V = \{v_0, v_1, ..., v_{n-1}\}$，二点間の距離の閾値を θ とする．$v_i = (x_i, y_i)(i = 0, ..., n-1)$ とし，x_i および y_i はそれぞれ v_i の x 座標と y 座標を表している．

定義 1（二点間の距離）． v_i および v_j 間の距離 $dist(v_i, v_j)$ を以下に定義する．

$$dist(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

点（ノード）集合 V，エッジ集合 E，およびエッジの重みの集合 W で構成されるグラフを $G(V, E, W)$ と表記する．$v_i, v_j \in V$ 間のエッジを $e_{i,j}$，$e_{i,j}$ の重みを $w_{i,j}$ と表記する．また，二点間の距離が閾値 θ 以下であるときにのみエッジが存在し，その重みが二点間の距離であるグラフを $G_\theta(V, E, W)$ と表記する．さらに，それぞれの点に対して k 番目までの近傍の点にエッジが存在し，重みが二点間の距離であるグラフを $G_k(V, E, W)$ と表記する．

定義 2（三角形の重み）． v_i，v_j，および v_k を頂点とする三角形 (v_i, v_j, v_k) の重み $m(v_i, v_j, v_k)$ を以下に定義する．

$$m(v_i, v_j, v_k) = w_{i,j} + w_{j,k} + w_{i,k}$$

問題定義． 本問題は，k，θ，および V が与えられたとき，重み m が小さい上位 k 個の三角形を列挙することである．

3 関連研究

これまでに大規模なグラフから特定の部分グラフを列挙する手法 [4, 8, 12, 27, 28] や，特定の部分グラフをカウントする手法 [1–4, 19, 20, 24, 25] が多く提案されている．部分グラフの列挙手法として，文献 [27] では，大規模なグラフから DN-graph という密なグラフを三角測量で見つけるアルゴリズムを提案している．また，文献 [28] では，メインメモリに収まらないグラフ G 内の三角形を列挙するアルゴリズムを提案している．しかし，これらの手法は辺の重みを考慮していないため本問題に適用できない．

部分グラフの数をカウントできる手法として，文献 [20] では，大規模なグラフ G から頂点が 5 個以下のすべての連結する部分グラフの数を正確にカウントするためのアルゴリズム ESCAPE（Efficient Subgraph Counting Algorithmic PackagE）を提案している．さらに文献 [19] では，ESCAPE を任意の点 v を含む部分グラフのカウントである局所的なカウントに拡張したアルゴリズムを提案している．また，文献 [2] では，大規模なグラフ G 内の三角形の数をカウントするアルゴリズム PATRIC を提案している．しかし，これらの手法は部分グラフの数を正確にカウントするものであるため，本問題には適用できない．

重み付き三角形を列挙できる手法として，与えられた重み付きグラフから辺の和が最も大きい k 個の三角形を高速に列挙する DHL（Dynamic Heavy Light）アルゴリズムがある [12]．入力データを用いてグラフ G_θ を構築した後に DHL アルゴリズムを実行することにより，本論文の問題を正確に解くことができる．しかし，DHL アルゴリズムはグラフを構築した後にすべてのエッジをソートする必要があり，そのソートに要する時間が大きいという欠点がある．

4 提案アルゴリズム

4.1 アイデア

三点 v_i，v_j，および v_k がエッジ $e_{i,j}$，$e_{j,k}$，および $e_{i,k}$ によって接続されているときに三角形 (v_i, v_j, v_k) ができる．ここで，v_i を頂点とする三角形が解に含まれないこ

とがわかれば，エッジ $e_{i,j}$ および $e_{i,k}$ にアクセスする必要がなくなる．提案アルゴリズムでは候補となる点を減らすことによって解になり得ない三角形を除外し，高速に上位 k 個の三角形を列挙する．

4.2 アルゴリズム

任意の点 v の隣接ノードを v との距離の昇順で 1-NN$_v$，2-NN$_v$，... と表記する．提案アルゴリズムは以下の手順で構成されている．また，提案アルゴリズムの疑似コードをアルゴリズム 1 に示す．

1. 点集合 V からバッチサイズ B によって $b = B$ として b-NN グラフ G_b をオフラインで構築する．
2. 候補の点集合 C の各点 c に対して，c，1-NN$_c$，および 2-NN$_c$ で構成される三角形を列挙し，その三角形の辺の重みの和を昇順にソートし，与えられた k，および θ を用いて解の閾値 τ を設定する．
3. C の各点 c に対して，エッジの重みの閾値 D を用いて，解に含まれない点を C から除く．
4. C の各点 c に対して，手順 2 および 4 を行った合計の回数を t とするとき，$e_{c,(t+2)\text{-NN}_c}$ を一辺とする三角形を追加で列挙する．
5. $C \neq \emptyset$ かつ $t = b - 1$ の場合，$b = b + B$ として b-NN グラフ G_b を更新する．
6. $C = \emptyset$ となるまで 3，4，および 5 を繰り返す．
7. 列挙した三角形のうち，三辺の重みの和が小さい上位 k 個の三角形を解として返却する．

次に提案アルゴリズムを詳細に説明する．

4.2.1 グラフの構築

バッチサイズを B とする．$b = B$ として与えられた点集合 V から重み付きグラフ $G_b(V, E, W)$ を構築する．グラフを構築するため，kd 木 [7] を用いて，V の各点 v に対して b-NN 検索を行う．また，各ノード v に対して $v.neighbors$ に v の隣接ノードを 1-NN$_v$，2-NN$_v$，...，b-NN$_v$ の順に格納する．この操作はユーザからの入力に依存しないため，オフラインで実行できる．

Algorithm 1: PROPOSED-ALGORITHM

Input: weighted graph $G_b(V, E, W)$, edge threshold θ, number of triangles k and batch size B
Output: k triangles with the smallest weight

```
b ← B
T ← ∅
C ← {v_0, v_1, ··· , v_{n−1}}
t ← 1
DETERMININGTHRESHOLD(G_b(V, E, W), k, T, C, θ)
while C ≠ ∅ do
    REDUCINGCANDIDATENODES(G_b(V, E, W), t, C, D)
    t ← t + 1
    ENUMERATINGTRIANGLES(G_b(V, E, W), k, T, τ, t, C)
    D ← T_k.weight − max(w_{x_k,y_k}, w_{y_k,z_k}, w_{x_k,z_k})
    if t = b − 1 then
        b ← b + B
        Build G_b(V, E, W) from C and b
```

4.2.2 三角形の閾値の決定

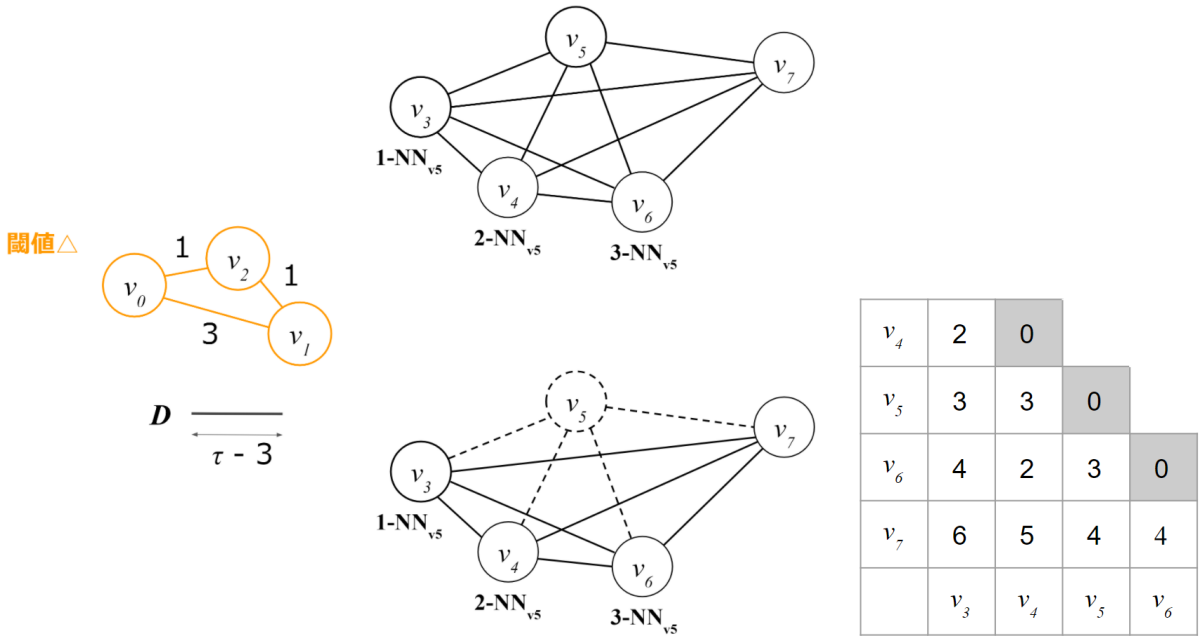
解の候補の三角形に含まれる点の集合を $C = \{c_0, c_1, ..., c_{n-1}\} = \{v_0, v_1, ..., v_{n-1}\} = V$ とする．候補の点集合 C の各点 c に対して三角形 (c，1-NN$_c$，2-NN$_c$) を列挙する．このとき，三角形 (v_i, v_j, v_k) と三角形 (v_j, v_i, v_k) は同じであるため，このような三角形が重複しないように注意する．列挙した三角形を三辺の重みの和で昇順にソートする．ここまではオフラインで実行できる．

ここで，ユーザから与えられる k および θ を用いて列挙した三角形の中で一辺の重みが θ を超える三角形を除外し，三角形の三辺の重みの和が最小の k 個だけを保持する．さらに，保持した k 番目の三角形の三辺の重みの和を閾値 τ とする．

4.2.3 候補の点の削減

手順 2 では，候補の点集合 C に含まれる各点 c に対して三角形 (c，1-NN$_c$，2-NN$_c$) を列挙している．ここで，手順 2 および 4 を行った合計回数を t と想定する．閾値 τ と暫定解の k 番目の三角形の最大のエッジの重みの差 D について考えたとき，追加で三角形を列挙する必要がある点 c は，$D > w_{c,(t+2)\text{-NN}_c}$ となるエッジ $e_{c,(t+2)\text{-NN}_c}$ を持つものに限定できる．

定理 1． 三角形 (v_i, v_j, v_k) が $w_{i,j} \leq w_{j,k} \leq w_{i,k}$，三角形 (v'_i, v'_j, v'_k) が $w'_{i,j} \leq w'_{j,k} \leq w'_{i,k}$ および $w_{i,j} + w_{j,k} \leq w'_{i,k}$ のとき，$w_{i,j} + w_{j,k} + w_{i,k} \leq w'_{i,j} + w'_{j,k} + w'_{i,k}$ である．



v_4	2	0		
v_5	3	3	0	
v_6	4	2	3	0
v_7	6	5	4	4
	v_3	v_4	v_5	v_6

(a) エッジの重みの閾値 D　(b) 点 v_5 の削減　(c) エッジの重み

図 1: 候補の点の削減の例

証明．三角形 (v_i, v_j, v_k) が三角形であるためには，三角不等式から $w_{i,j}+w_{j,k} \geq w_{i,k}$ である必要がある．三角形 (v'_i, v'_j, v'_k) についても同様に $w'_{i,j}+w'_{j,k} \geq w'_{i,k}$．また，$w_{i,j}+w_{j,k} \leq w'_{i,k}$ であるから，$w_{i,j}+w_{j,k} \leq w'_{i,j}+w'_{j,k}$ および $w_{i,k} \leq w'_{i,k}$ となる．つまり，$w_{i,j}+w_{j,k}+w_{i,k} \leq w'_{i,j}+w'_{j,k}+w'_{i,k}$ である．□

点 c について，$e_{c,(t+2)\text{-NN}_c}$ の重み $w_{c,(t+2)\text{-NN}_c}$ が D よりも大きい場合，定理 1 から，$e_{c,(t+2)\text{-NN}_c}$ を一辺とする三角形の辺の重みの和が τ を超える．つまり，まだ列挙されていない三角形の中で c を含む三角形はこれ以降必ず三辺の重みの和が τ より大きくなる．そうでない場合は，$e_{c,(t+2)\text{-NN}_c}$ を一辺とする三角形が解となる可能性があるため，c は候補の点となる．

例 2. 図 1 を用いて候補点の削減の例を説明する．図 1a において三角形 (v_0, v_1, v_2) の三辺の重みの和が閾値 τ であるとし，D は三角形 (v_0, v_1, v_2) の重みと三辺の中で最大のエッジの重みとの差を表しており，$D = 5-3 = 2$ である．また，図 1c は図 1b におけるそれぞれの点同士の辺の重みを表している．図 1b において v_5 を候補の点集合から削減可能か考える．エッジ $e_{5,(t+2)\text{-NN}_{v_5}}$ の重みは $w_{5,(t+2)\text{-NN}_{v_5}}$ であり，$t = 1$ のとき，$w_{5,3\text{-NN}_{v_5}} = w_{5,6} = 3$ となる．$D = 2$ であるため，定理 1 を用いると，v_5 を含む三角形でまだ列挙されていない三角形の重みは閾値 τ を必ず超えてしまうことがわかる．つまり，v_5 を含む三角形でまだ列挙していない考慮すべき三角形は存在しない．したがって，これ以降の三角形の列挙において v_5 に注目する必要はない．

4.2.4　追加の三角形の列挙

手順 4 でまだ候補の点がある場合，追加で三角形を列挙できるか調べる必要がある．また，追加で三角形を列挙するたびに閾値 τ は更新する．手順 2 で閾値を求める際に，C の各点 c に対して三角形 (c,1-NN$_c$,2-NN$_c$) を列挙している．手順 4 では，C の各点 c に対してエッジ $e_{c,(t+2)\text{-NN}_c}$ を一辺とする三角形の重みを閾値 τ と比較し，より小さい三角形があれば解を更新する．つまり，$t = 1$ のときは三角形 (c,1-NN$_c$,3-NN$_c$) および (c,2-NN$_c$,3-NN$_c$) の二つの三角形を τ と比較し，より重みが小さい三角形を解の候補とする．同様に，K 回目の三角形の列挙では，C の各点 c に対してエッジ $e_{c,(K+1)\text{-NN}_c}$ をもつ三角形のそれぞれの重みを閾値 τ と比較し，より重みが小さい三角形を保持する．このように $e_{c,(t+2)\text{-NN}_c}$ を一辺とする三角形を τ と比較するとき，$e_{c,h\text{-NN}_c}(h \in \{1, ..., K\})$ をもう一辺とする三角形を h が 1 から順に K まで列挙し比較する．このとき，K 個のすべての三角形を閾値と比

較すると，比較回数が $K \cdot |C|$ となってしまう．しかし，途中で $w_{c,(K+1)\text{-NN}_c} + w_{c,h\text{-NN}_c} \geq \tau (h \in \{1, ..., K\})$ となると，c に対する三角形の列挙を中断できる．

候補の点集合 $C = \emptyset$ となるまで C の削減と追加の三角形の列挙を繰り返すことにより，三辺の重みの和が小さい上位 k 個の三角形を列挙することができる．ただし，$t = b - 1$ のときはこれ以上列挙できる三角形が残っていない．$C \neq \emptyset$ にもかかわらず列挙する三角形がなくなった場合，C に含まれる点に対して $b = b + B$ としてグラフ G_b を更新する．このグラフ G_b に対して C の削減と追加の三角形の列挙および G_b の更新を同様に繰り返すことにより，解を特定できる．

5　評価実験

本章では，提案アルゴリズムの性能を評価するための実験とその結果を示す．

5.1　設定

本実験では以下の二つのアルゴリズムの評価を行った．

- DHLA[12].
- 本研究における提案アルゴリズム (Proposed Algorithm：PA と表記).

DHLA[3]および PA はすべて C++で実装されており，g++の-O3 最適化オプションを付けてコンパイルされている．すべての実験を Windows10 が搭載された 3.6GHz Intel Core i9 および 128GB RAM で構成される計算機上で行った．

5.1.1　評価指標

本実験では，ユーザから入力データとして k および θ が与えられてから解を特定するまでの時間を評価した．

5.1.2　データセット

実験で用いるデータは place[4]および CaStreet[5]の二つである．

place このデータはアメリカの公共施設の位置情報が緯度経度で記録されたものである．データ総数は 9,356,750 である．

CaStreet このデータはカリフォルニアの道の位置情報が緯度経度で記録されたものである．また，データが最小外接矩形であるため，二つの位置座標を別のデータとして扱っている．データ総数は 4,499,454 である．

[3]https://github.com/raunakkmr/Retrieving-Top-Weighted-Triangles-in-Graphs
[4]https://archive.org/details/2011-08-SimpleGeo-CC0-Public-Spaces
[5]http://chorochronos.datastories.org/?q=node/59

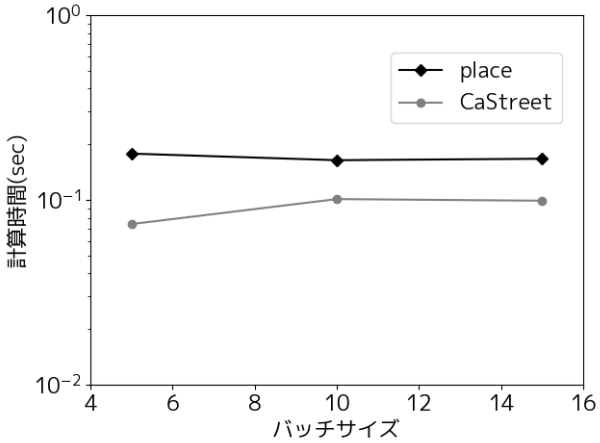


図 2: B の影響

5.1.3　パラメータ

本実験で用いたパラメータの設定を表 1 に示す．n は入力データのデータ数である．それぞれのパラメータにおける太字は本実験におけるデフォルト値である．

5.2　実験結果

5.2.1　バッチサイズ B による影響

バッチサイズを変化させたときの計算時間は図 2 の結果になった．バッチサイズが変化してもデータ計算時間に大きな差がないことがわかる．バッチサイズを変化させることにより，グラフ G_b の辺の数が変わる．また，G_b はオンラインで更新される場合がある．このことから，G_b を更新する場合，十分に候補の点集合 C が削減されているならば更新にかかる計算時間はわずかであることがわかる．この結果から，$B = 10$ とした．

5.2.2　列挙する三角形の数 k による影響

k を 10 から 1000 まで変化させたときの計算時間は図 3 の結果になった．図 3a および図 3b から，k が 10 のときには PA の計算時間が DHLA と比べて約 2000 倍早いことがわかる．PA において，place は k が大きくなるにつれて実行時間が少しずつ増えているが，CaStreet では計算時間が一定であった．これは place と比べて CaStreet のデータ分布が疎であり，閾値の更新やグラフの更新があまり行われなかったためであると考えられる．それに対して，DHLA ではデータにかかわらず計算時間が一定

表 1: パラメータ

データセット	パラメータ	値
place	B	$5/\mathbf{10}/15$
	θ	$\mathbf{0.01}$
	k	$10/50/\mathbf{100}/500/1000$
	n	$\mathbf{1,000,000}/2,500,000/5,000,000/7,500,000/9,356,750$
CaStreet	B	$5/\mathbf{10}/15$
	θ	$\mathbf{0.01}$
	k	$10/50/\mathbf{100}/500/1000$
	n	$\mathbf{1,000,000}/2,500,000/4,499,454$

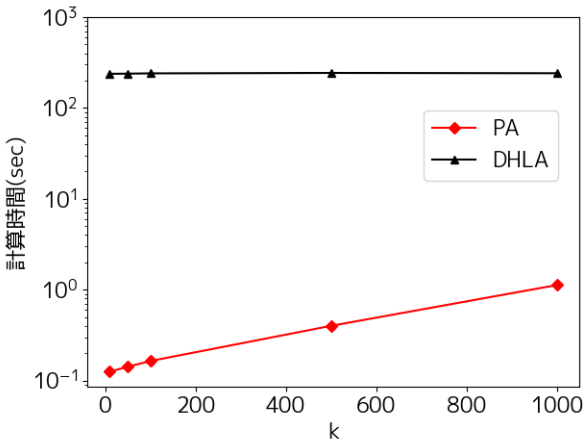
である．これは，DHLA にかかる時間のほとんどがエッジのソートであることが原因である．

5.2.3 データ数 n による影響

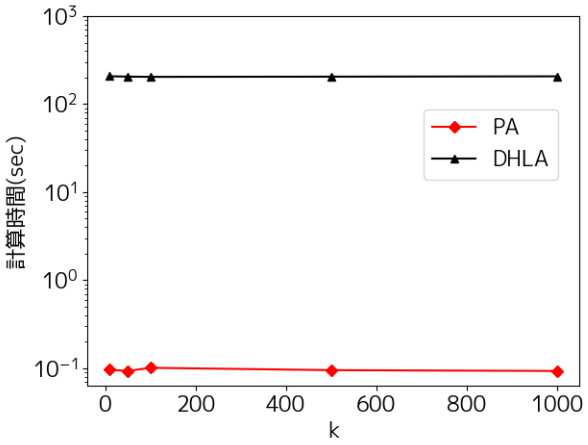
データ数を変化させたときの計算時間は図 4 の結果になった．図 4a および図 4b から，データ数が増加するに伴い計算時間は増加していることがわかる．データ数が 1,000,000 のときの計算時間を比較すると PA の方が place では約 1000 倍早く，CaStreet では約 2000 倍早い．

6 結論

本研究では，点集合から三辺の重みの和が最も小さい k 個の三角形を正確かつ効率的に列挙できるアルゴリズムを提案した．提案アルゴリズムは考慮するべき三角形を減らすことにより効率よく上位 k 個の三角形を列挙することができる．実データを用いた評価実験により，提案アルゴリズムが比較手法に比べて非常に高速に三角形を列挙することを確認した．



(a) place



(b) CaStreet

図 3: k の影響

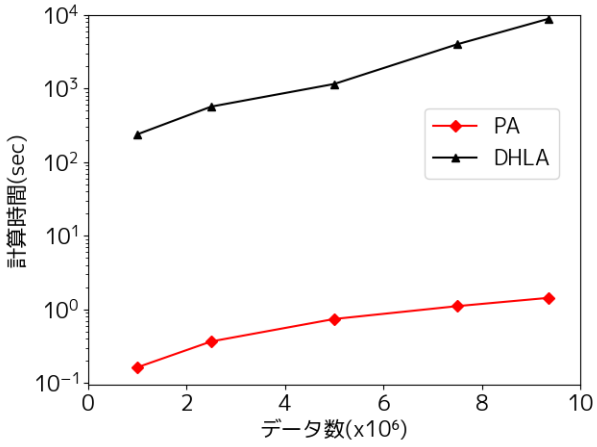
謝辞

本研究の一部は，文部科学省科学研究費補助金・基盤研究 (A)(18H04095)，JST さきがけ（JPMJPR1931），および JSTCREST（J181401085）の支援を受けたものである．

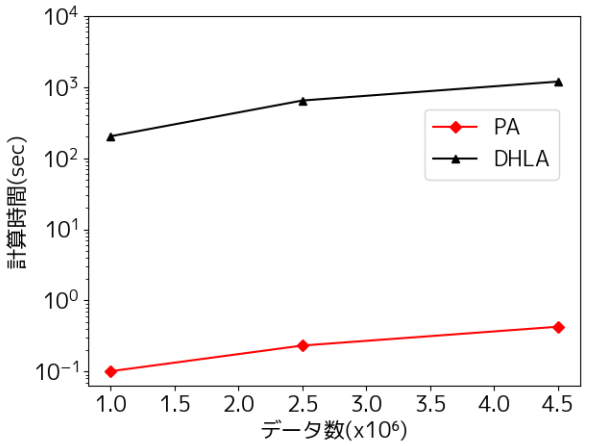
参考文献

[1] M. Al Hasan and V.S. Dave, “Triangle counting in largenetworks: a review,” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol.8, no.2, p.e1226, 2018.

[2] S. Arifuzzaman, M. Khan, and M. Marathe, “Patric: A parallel algorithm for counting triangles in massive networks,” Proceedings of the ACM International Conference on Information & Knowledge Management, pp.529–538, 2013.



(a) place



(b) CaStreet

図 4: n の影響

[3] H. Avron, “Counting triangles in large graphs using randomized matrix trace estimation,” Proceedings of the Workshop on Large-scale Data Mining: Theory and Applications, vol.10, pp.10–9, 2010.

[4] A. Azad, A. Buluç, and J. Gilbert, “Parallel triangle counting and enumeration using matrix algebra,” Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshop, pp.804–811, 2015.

[5] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani, “The architecture of complex weighted networks,” Proceedings of the National Academy of Sciences, vol.101, no.11, pp.3747–3752, 2004.

[6] A.R. Benson, D.F. Gleich, and J. Leskovec, “Higher-order organization of complex networks,” Science, vol.353, no.6295, pp.163–166, 2016.

[7] J.L. Bentley, “Multidimensional binary search trees used for associative searching,” Communications of the ACM, vol.18, no.9, pp.509–517, 1975.

[8] S. Chu and J. Cheng, “Triangle listing in massive networks and its applications,” Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp.672–680, 2011.

[9] M.S. Hossain, S. Mondal, R.S. Ali, and M. Hasan, “Optimizing complexity of quick sort,” Proceedings of the International Conference on Computing Science, Communication and Security, pp.329–339, 2020.

[10] F. Hu, G. Wang, and L. Feng, “Fast knowledge reduction algorithms based on quick sort,” Proceedings of the International Conference on Rough Sets and Knowledge Technology, pp.72–79, 2008.

[11] J. Jia, M.T. Schaub, S. Segarra, and A.R. Benson, “Graph-based semi-supervised & active learning for edge flows,” Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp.761–771, 2019.

[12] R. Kumar, P. Liu, M. Charikar, and A.R. Benson, “Retrieving top weighted triangles in graphs,” Proceedings of the ACM International Conference on Web Search and Data Mining, pp.295–303, 2020.

[13] J. Leskovec, K.J. Lang, and M. Mahoney, “Empirical comparison of algorithms for network community detection,” Proceedings of the International Conference on World Wide Web, pp.631–640, 2010.

[14] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon, “Superfamilies of evolved and designed networks,” Science, vol.303, no.5663, pp.1538–1542, 2004.

[15] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, “Network motifs: simple building blocks of complex networks,” Science, vol.298, no.5594, pp.824–827, 2002.

[16] P. Mohan, S. Shekhar, J.A. Shine, J.P. Rogers, Z. Jiang, and N. Wayant, “A neighborhood graph based approach to regional co-location pattern discovery: A summary of results,” Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp.122–132, 2011.

[17] T. Murata and S. Moriyasu, “Link prediction of social networks based on weighted proximity measures,” Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, pp.85–88, 2007.

[18] K. O’Hara, N.S. Contractor, W. Hall, J.A. Hendler, and N. Shadbolt, “Web science: Understanding the emergence of macro-level features on the world wide web,” Foundations and Trends in Web Science, vol.4, no.2-3, pp.103–267, 2013.

[19] N. Pashanasangi and C. Seshadhri, “Efficiently counting vertex orbits of all 5-vertex subgraphs, by evoke,” Proceedings of the ACM International Conference on Web Search and Data Mining, pp.447–455, 2020.

[20] A. Pinar, C. Seshadhri, and V. Vishal, “Escape: Efficiently counting all 5-vertex subgraphs,” Proceedings of the International Conference on World Wide Web, pp.1431–1440, 2017.

[21] A. Prat-Pérez, D. Dominguez-Sal, and J.L. Larriba-Pey, “High quality, scalable and parallel community detection for large real graphs,” Proceedings of the International Conference on World Wide Web, pp.225–236, 2014.

[22] N. Pržulj, “Biological network comparison using graphlet degree distribution,” Bioinformatics, vol.23, no.2, pp.e177–e183, 2007.

[23] K.J. Supowit, “The relative neighborhood graph, with an application to minimum spanning trees,” Journal of the ACM, vol.30, no.3, pp.428–448, 1983.

[24] C.E. Tsourakakis, “Fast counting of triangles in large real networks without counting: Algorithms and laws,” Proceedings of the IEEE International Conference on Data Mining, pp.608–617, 2008.

[25] C.E. Tsourakakis, U. Kang, G.L. Miller, and C. Faloutsos, “Doulion: counting triangles in massive graphs with a coin,” Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp.837–846, 2009.

[26] J. Wang and S. Li, “Query-driven iterated neighborhood graph search for large scale indexing,” Proceedings of the ACM International Conference on Multimedia, pp.179–188, 2012.

[27] N. Wang, J. Zhang, K.L. Tan, and A.K. Tung, “On triangulation-based dense neighborhood graph discovery,” Proceedings of the International Conference on Very Large Data Bases, vol.4, no.2, pp.58–68, 2010.

[28] C. Zhang, Y. Zhang, W. Zhang, L. Qin, and J. Yang, “Efficient maximal spatial clique enumeration,” Proceedings of the IEEE International Conference on Data Engineering, pp.878–889, 2019.