DEIM2022 K34-5

空間データにおける重み付き三角形の効率的なTop-kモニタリング

谷口　凌亮† 　天方　大地†,†† 　原　隆浩†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1-5

†† JST さきがけ

E-mail: †{taniguchi.ryosuke,amagata.daichi,hara}@ist.osaka-u.ac.jp

あらまし　近年，位置情報サービスやIoTデバイスの普及により，空間データが大量に生成されている．また，空間データの解析は重要な課題である．空間データの解析方法として，各点が近傍への辺を持ち，各辺の重みが対応する点間の距離であるグラフに注目されている．本研究ではこのようなグラフにおいて，最も単純な部分グラフの1つである三角形に焦点を当てる．これは，コミュニティ検出，都市計画，およびコロケーションパターン検索など，実生活での応用で用いられる．ここで，一般的に空間データから成るグラフ中の三角形の数は膨大であり，全ての三角形の列挙は現実的でない．また，位置情報サービスやIoTサービスではデータがストリーム形式で発生する．そこで本論文では，上位k個の重み付き三角形をスライディングウィンドウ上で正確にモニタリングする問題に取り組む．実データを用いて実験を行い，提案アルゴリズムの有効性を示す．

キーワード　空間データ，重み付きグラフ，Top-kモニタリング

1 はじめに

近年, 位置情報サービスやIoTデバイスの普及に伴い, 空間データが大量に生成されている．空間データの分析は実世界で様々な応用例があり，多くの空間データ解析手法 [1–4, 10, 11, 16] やシステム [13, 18] が考案されている．最近では，空間データから成る近接グラフに基づいた解析手法が注目されている [5–8, 17, 19]．近接グラフは，空間データの集合Pと距離の閾値rが与えられたとき，P内の2点間の距離がr以下である場合にその2点間の距離が重みである辺が作られる．グラフを用いることにより，空間データ間の直感的な関係を発見可能であるが，そのためには近接グラフの部分グラフのマイニングが必要である．部分グラフの中でも三角形は最も単純な構造であるため，多くのアプリケーションに用いられている [12, 14]．空間データにおける三角形は，コミュニティ検出 [9]，コロケーションパターンマイニング [19]，および都市計画 [7, 8] などに利用可能である．近接グラフ内の三角形の数は一般に膨大であるため，全ての三角形を列挙することは不可能である．したがって，出力サイズはユーザが指定したパラメータkによって制御可能である必要がある [8, 12]．また，三角形の凝集性は空間データベースにおいて重要性を表す指標となる [9, 19]．

本論文では，上位k個の重み付き三角形に注目する．2点p_xおよびp_yのユークリッド距離を$dist(p_x, p_y)$で表すと，3点p_x，p_y，およびp_zで形成される三角形の重みは，$dist(p_x, p_y) + dist(p_y, p_z) + dist(p_x, p_z)$で表される．$P$と出力サイズ$k$が与えられたとき，$P$の近接グラフの三角形の中で最も重みが小さい$k$個の三角形を検索する．問題に対する単純な解法は，全ての三角形を列挙し，最小の重みをもつ上位k個の三角形を出力することである．しかし，近接グラフの三角形の数は$n = |P|$のとき，$O(\binom{n}{2})$であり，計算コストが大きいという問題がある．また，位置情報サービスやIoTサービスでは時々刻々とデータが発生するため，上位k個の三角形を常にモニタリングすることが求められる [1, 15]．そのため，データ集合が更新される毎に高速に結果を更新する技術が必要となる．

この課題を解決するため，本論文ではスライディングウィンドウ上で効率的に結果を更新するアルゴリズムを提案する．近接グラフの各点$p \in P$に対して，pを持つTop-k結果に含まれない最も重みの小さい三角形とその重みをそれぞれ$\triangle_p^{\min}$および$w(\triangle_p^{\min})$で表す．Top-k三角形を更新しなければならない状況は，(i) 新しく追加された点を持つ三角形の重みが現在のk番目の三角形の重みよりも小さい場合，および (ii) 削除された点がTop-k三角形に含まれている場合である．(i) の場合，Top-k番目の三角形の重み未満となる$w(\triangle_p^{\min})$を持つ点pだけに注目することにより，効率的にTop-k結果を更新できる．(ii) の場合，Top-k結果の不足分を$w(\triangle^{\min})$の昇順で補充することにより，タイトな閾値を得ることができ，Top-k結果の更新に不要な点や三角形の枝刈りに効果的である．一方，単純には点の挿入および削除毎にその影響を受ける点pに対して$\triangle_p^{\min}$および$w(\triangle_p^{\min})$を更新しなければならない．しかし，有向近接グラフを用いることにより，点の削除を行う場合は点pに対して$\triangle_p^{\min}$および$w(\triangle_p^{\min})$の更新を行う必要がないアルゴリズムを提案する．我々の主な貢献は以下の通りである．

- 上位k個の重み付き三角形をスライディングウィンドウ上でモニタリングする問題に取り組む．
- 効率的かつ厳密なアルゴリズムを提案する．
- 実データを用いて実験を行い，提案アルゴリズムが高速に解を更新できることを示す．

本論文の構成． 2章では，本論文の問題を定義する．3章では，関連研究を紹介する．4章では，提案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 章で本論文のまとめを行う．

2 問題定義

P をユークリッド空間内の点集合とする．点 $p \in P$ は 2 次元であり，p および p' 間のユークリッド距離を $dist(p, p')$ で表す．ここで，点同士を近いとみなす閾値を r とすると，以下のような近接グラフを構築できる．

定義 1 （近接グラフ）．点集合 P と閾値 r が与えられたとき，P の近接グラフは，点 p_i および p_j 間が $dist(p_i, p_j) \leqq r$ である場合に p_i-p_j 間に辺が作られる無向グラフである．p_i および p_j 間の辺は $e_{i,j}$ と表され，$w(e_{i,j}) = dist(p_i, p_j)$ である重み $w(e_{i,j})$ を持つ．

近接グラフでは，互いに接続された 3 点から成る三角形が存在する．この三角形の重みを定義する．

定義 2 （三角形の重み）．3 点 p_x，p_y，および p_z から成る三角形 $\triangle_{x,y,z}$ があるとき，この三角形の重み $w(\triangle_{x,y,z})$ は式 (1) のようになる．

$$w(\triangle_{x,y,z}) = dist(p_x, p_y) + dist(p_y, p_z) + dist(p_x, p_z). \quad (1)$$

そして，上位 k 個の重み付き三角形を検索する問題は以下のように定義される．

定義 3 （重み付き三角形の Top-k 検索問題）．点集合 P，出力サイズ k，および閾値 r が与えられたとき，この問題は，P の近接グラフにおいて，最小の重みをもつ k 個の三角形を検索する．

ここで，r が小さすぎると P の近接グラフがスパースになり，グラフ内に k 個以下の三角形しか存在しないことがある．この場合，この問題は簡単であるため，r はグラフ内に多くの三角形を持つように適切な値であると想定する．本論文では P が動的であることを想定する．このとき，Top-k 結果を更新することが求められるため，本論文では以下の問題に取り組む．

定義 4 （スライディングウィンドウ上の重み付き三角形の Top-k モニタリング問題）．動的な点集合 P，出力サイズ k，閾値 r，およびウィンドウサイズ W が与えられたとき，本問題はウィンドウ上の P において最も重みの小さい k 個の三角形を探索する．

本論文では，カウントベースウィンドウを想定する．これは最新の W 個の点をウィンドウに含める．以降では，P をウィンドウに含まれる点集合とする．

3 関連研究

グラフによる空間データマイニング． グラフはデータ間の関係を表現するシンプルで効果的な構造であり，また，一般的に位置が近い点同士には関係がある．そのため，グラフを用いた空間データマイニングが注目されている．

文献 [7] では，空間パターンマッチングについて検討している．空間パターンマッチングとは，P およびグラフパターンであるクエリが与えられたとき，クエリにマッチする P のすべてのサブセットを求める問題である．この空間パターンマッチングでは，我々の問題とは異なり，P のサブグラフを指定する必要がある．しかし，P のグラフ構造は事前にわからないため，具体的なクエリを指定することは難しい．また，クエリの結果のサイズは制御できない．文献 [8] では，空間パターンマッチングの Top-k を考慮した手法について検討しているが，クエリに対する欠点は残っている．文献 [19] では，P の近接グラフにおける最大クリークについて検討している．この文献では，最大クリークの検索は最大凸多角形を作ることに対応していることを示したが，多角形の重みを考慮していないため，この手法を我々の問題に適用することはできない．

文献 [16] では，位置情報サービスプロバイダの集合 Q と位置情報を持つユーザの集合 U が与えられたときに，Q と U の 2 部グラフの最大マッチングに取り組んでいる．この問題は，グラフを構築することに焦点を当てている．文献 [17] では，マルチコアプロセッサを用いて，与えられた P から近接グラフを構築するシステムを設計した．また，近接グラフ上でのクラスタリングや最小全域木の計算など，ほかの操作もサポートしている．しかし，重み付き三角形の Top-k 検索はサポートされておらず，このシステムを本問題へ適用することはできない．

重み付き三角形の Top-k 検索． 静的な空間データに対して効率的に解を求めるアルゴリズムを紹介する [20]．我々は P の近接グラフ内で重みの小さい上位 k 個の三角形を含む部分グラフをオフラインで構築可能であることを確認した．また，この部分グラフから，各点 $p \in P$ に対して，p を持つ三角形を $O(1)$ 時間で列挙可能であり，これらの n 個の三角形の重みから Top-k のためのタイトな閾値が計算可能であり，不要な点や三角形の枝刈りに効果的である．不要な点を枝刈りした後に残った点に対して重みが小さい可能性がある三角形を列挙して解を更新し，さらに不要な点を枝刈りする．これを繰り返し行うことにより，効率的に解を求めることができる．このアルゴリズムは本問題へ適用可能であるが，Top-k 結果を更新するには，ウィンドウのスライド毎に近接グラフを再構築し，このアルゴリズムにより再び解を求める必要があるため，リアルタイムモニタリングには適していない．そのため，本論文ではインクリメンタルに解を更新するアルゴリズムを設計する．

4 提案アルゴリズム

4.1 アイデア

Top-k 結果を更新しなければならない状況は，(i) 新しく追加された点を持つ三角形の重みが現在の k 番目の三角形の重みよりも小さい場合，および (ii) 削除された点が Top-k であった三角形に含まれている場合である．Top-k 結果の更新を効率的に行うためには，現在の Top-k となる三角形およびこれまで

に列挙した三角形を活用すればよい．しかし，これまでに列挙した三角形を全て保持すると，メモリ使用量が膨大になってしまう．ここで，各点 $p \in P$ について，p を持つ三角形で，現在の Top-k に含まれていない最も重みの小さい三角形は今後の Top-k に含まれる可能性がある．提案アルゴリズムはこのアイデアを利用する．

定義 5 点 $p \in P$ を持ち，Top-k に含まれない三角形があるとき，その中で重みが最も小さい三角形を $\triangle_p^{\min}$ およびその重みを $w(\triangle_p^{\min})$ とする．

各点に対する $\triangle^{\min}$ は点の挿入および削除により変化するが，Top-k 結果の更新において効果的である．(i) の場合では，Top-k 番目の三角形の重み未満となる $w(\triangle_x^{\min})$ を持つ点 p_x だけに注目することにより，効率的に Top-k 結果を更新でき，(ii) の場合では，Top-k 結果の不足分の三角形を $w(\triangle^{\min})$ の昇順で補充することにより，タイトな閾値を得ることができ，Top-k 結果の更新に不要な点や三角形の枝刈りに効果的である．また，各点に対して $\{\triangle^{\min}, w(\triangle^{\min})\}$ のみを保持するため，メモリ使用量は $O(W)$ となる．

提案アルゴリズムは，点の削除，点の挿入，および Top-k 結果の更新により構成されている．以降では，これらの詳細を説明する．

4.2 点の削除

点 p_x が P から削除されるとき，p_x が $\triangle^{\min}$ に含まれているか確認する．p_x の近傍の点集合を $N(p_x)$ とすると，点 $p_y \in N(p_x)$ について，p_x が $\triangle_y^{\min}$ に含まれる場合，$\triangle_y^{\min}$ を更新しなければならない．$\triangle_y^{\min}$ は，p_y を持ち，Top-k に含まれない最も重みの小さい三角形であるため，更新するには p_y を持つ三角形を列挙する必要がある．p_y の近傍の点のうち p_y との距離が昇順で i 番目の点を p_y^i としたとき，$\triangle_y^{\min}$ の更新は，以下の手順で行う．

1. 三角形の重みの閾値 τ および辺の重みの閾値 θ を設定する．
2. $\theta > w(e_{p_y,p_y^{i+2}})$ のとき，$e_{p_y,p_y^{i+2}}$ を 1 辺とする三角形 $\triangle_{p_y,p_y^j,p_y^{i+2}}$ を列挙する（$1 \leqq i \leqq |N(p_y)|, 1 \leqq j \leqq i+1$）．
3. $\triangle_y^{\min}$ を更新し，τ および θ も更新する．
4. 手順 2 および 3 を繰り返し行う．

手順 1 では，三角形の重みの閾値および辺の重みの閾値をそれぞれ $\tau = w(\triangle_{p_y,p_y^1,p_y^2})$ および $\theta = \tau - \max\{dist(p_y,p_y^1), dist(p_y,p_y^2), dist(p_y^1,p_y^2)\}$ と設定する．手順 2 は，三角不等式に基づいており，条件を満たさない場合はそれ以降に列挙する三角形が全て τ よりも重みの大きい三角形となる．手順 3 では，よりタイトな閾値にするために，列挙した三角形の重みが τ より小さい場合に更新する．そして，列挙した三角形のうち，Top-k に含まれない最も重みの小さい三角形を $\triangle_y^{\min}$ に設定する．p_y を持つ三角形の列挙は，手順 2 の条件を満たさない場合に終了する．点の削除の疑似コードをアルゴリズム 1 に示す．

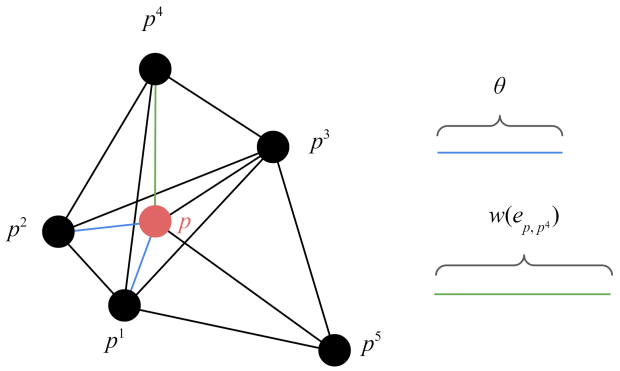


図 1: $\triangle_p^{\min}$ の更新の例

Algorithm 1: REMOVE

Input: P (set of points) and p_x (removed node)

```
for ∀p_y ∈ N(p_x) do
    if p_x ∈ △_y^min then
        △_y^min ← UPDATE-△^min(P, p_y)
```

Algorithm 2: INSERT

Input: P (set of points), p_x (insert node) and r (threshold)

```
△_x^min ← UPDATE-△^min(P, p_x)
for ∀p_y ∈ N(p_x) do
    if p_x ∈ △_y^min then
        △_y^min ← UPDATE-△^min(P, p_y)
```

例 1. 図 1 を用いて $\triangle_p^{\min}$ の更新の例を説明する．手順 1 に従って，三角形の重みの閾値 τ および辺の重みの閾値 θ をそれぞれ $\tau = w(\triangle_{p,p^1,p^2})$ および $\theta = \tau - dist(p^1,p^2)$ と設定する．手順 2 では，$\theta > w(e_{p,p^3})$ のとき，e_{p,p^3} を 1 辺とする三角形 $\triangle_{p,p^1,p^3}$ および $\triangle_{p,p^2,p^3}$ を列挙する．手順 3 では，$\triangle_{p,p^1,p^2}$，$\triangle_{p,p^1,p^3}$，および $\triangle_{p,p^2,p^3}$ のうち，Top-k に含まれない最も重みの小さい三角形を $\triangle_p^{\min}$ に設定する．そして，τ と $w(\triangle_{p,p^1,p^3})$ および $w(\triangle_{p,p^2,p^3})$ を比較し，最も重みの小さい三角形により τ および θ を更新する．p^4 および p^5 に対しても同様に行う．しかし，p^4 について，$\theta \leqq w(e_{p,p^4})$ となり手順 2 の条件を満たさない．このとき，三角不等式により，e_{p,p^4} を 1 辺とする三角形の重みは τ よりも大きくなり，p^4 以降の点に対しても同様であるため，p^4 および p^5 について三角形を列挙せずに終了する．

4.3 点の挿入

点 p_x が P に挿入されるとき，p_x を持つ三角形が $\triangle^{\min}$ になるかどうか確認する．まず，p_x を中心とした半径 r の範囲検索により $N(p_x)$ を求め，P の近接グラフを更新する．そして，点の削除と同様に p_x を持つ三角形および点 $p_y \in N(p_x)$ を持つ三角形を列挙し，$\triangle_x^{\min}$ および $\triangle_y^{\min}$ を更新する．この操作の疑似コードをアルゴリズム 2 に示す．

Algorithm 3: UPDATE-TOP-k

Input: P (set of points), r (threshold), T (set of $|P|$ triangles with $\triangle^{\min}$) and R (set of k triangles with the minimum weight)

Output: R

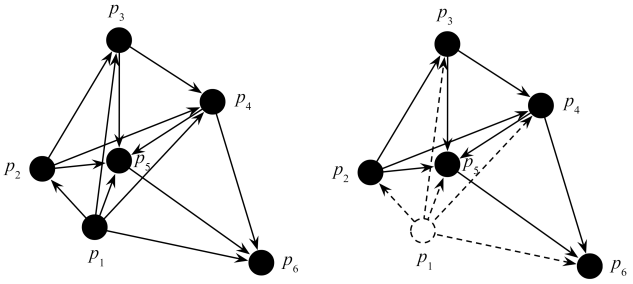
```
1  Sort the triangles △^min ∈ T in ascending order of w(△^min)
2  l ← k − |R|
3  if l > 0 then
4      R ← R ∪ {l triangles with the smallest weight in T}
5  △_{x,y,z} ← triangle with the k-th smallest weight in R
6  τ ← w(△_{x,y,z})
7  i ← 1
8  while i ≦ l do
9      △_p^min ←triangle with the i-th smallest weight in T
10     R ← UPDATE-TOP-k-TRIANGLES(P, p)
11     △_p^min ← UPDATE-△_p^min(P, p)
12     Execute lines 5-6
13     i ← i + 1
14 △_q^min ←triangle with the smallest weight in T
15 i ← 1
16 while w(△_q^min) < τ do
17     Execute lines 10-13
18     △_q^min ←triangle with the i-th smallest weight in T
```

4.4 Top-k 結果の更新

Top-k 結果を更新する必要があるのは，(i) 追加された点の $w(\triangle^{\min})$ が現在の k 番目の三角形の重みよりも小さい場合，および (ii) 削除された点が Top-k 三角形に含まれている場合である．これらを考慮し，以下の手順で Top-k 三角形の更新を行う．

1. Top-k 結果の不足している l 個の三角形を点 $p \in P$ の $\triangle_p^{\min}$ をその重み $w(\triangle_p^{\min})$ の昇順で Top-k 結果に追加する $(0 \leqq l \leqq k)$.
2. Top-k 結果に追加した $\triangle_p^{\min}$ を持つ点 p について，p から成る他の三角形を列挙し，Top-k 結果および $\triangle_p^{\min}$ を更新する．
3. 点 $q \in P(q \neq p)$ の $\triangle_q^{\min}$ の重みが Top-k 番目の重みよりも小さいとき，手順 2 と同様の操作を行う．

手順 1 では，点の削除により Top-k 結果のいくつかの三角形が削除されている可能性があるため，$\triangle^{\min}$ の重みの昇順で l 個の三角形を Top-k 結果に追加することにより，Top-k 結果の更新のためのタイトな閾値を得ることができる．手順 2 では，閾値 τ よりも $\triangle_p^{\min}$ の重みが小さい p を持つ三角形を列挙する．これは，p を持つ三角形の中に $\triangle_p^{\min}$ ではないが Top-k になり得る三角形があるためである．また，必要に応じて $\triangle_p^{\min}$ を更新する．ここで，点の挿入により，挿入前の Top-k 結果と比べて重みが小さい三角形が多数生成される場合がある．このとき，手順 1 および 2 のみでは正確な Top-k 結果を保証できない．そこで手順 3 では，手順 1 および 2 を行い，Top-k 結果を更新した後の Top-k 番目の三角形の重みよりも $\triangle_q^{\min}$ の重み $w(\triangle_q^{\min})$ が小さい q を持つ三角形を列挙し，Top-k 結果および $\triangle_q^{\min}$ を更新する．この操作をアルゴリズム 3 に示す．



(a) 有向近接グラフ　　(b) 点 p_1 の削除

図 2: 有向近接グラフにおける点の削除の例

4.5 提案アルゴリズムの最適化

ここで，点 $a \in P$ の $\triangle_a^{\min}$ が $\triangle_{a,b,c}$ であったとき，点 b および c の $\triangle^{\min}$ も $\triangle_{a,b,c}$ である可能性があり，その場合，重複して三角形を保持している．このとき，$\triangle_{a,b,c}$ が Top-k に含まれると a，b，および c について，それぞれを持つ三角形を列挙し $\triangle^{\min}$ を更新しなければならない．また，a が P から削除されると b および c を持つ三角形を列挙し，$\triangle^{\min}$ を更新しなければならない．これらは $\triangle^{\min}$ に重複が起きるために発生する無駄な操作である．そこで，$\triangle^{\min}$ が重複しないようにするため，以下の有向近接グラフを用いる．

定義 6 (有向近接グラフ). 点集合 P と閾値 r が与えられたとき，点 $p \in P$ が P に挿入された順序を $o(p)$ で表すと，P の有向近接グラフは，点 p_i および p_j 間が $dist(p_i, p_j) \leqq r$ であり，$o(p_i) \prec o(p_j)$ である場合に p_i および p_j 間に有向辺 $(p_i \to p_j)$ が作られる有向グラフである．p_i および p_j 間の有向辺は $e_{i,j}$ と表され，$w(e_{i,j}) = dist(p_i, p_j)$ である重み $w(e_{i,j})$ を持つ．

有向近接グラフを用いることにより，点の削除，挿入，および Top-k 結果の更新を近接グラフよりも効率的に行うことができる．

点の削除. 点 p_x を P から削除するとき，この問題では，P 内の全ての点 $p(\neq p_x)$ に対して p_x は $o(p_x) \prec o(p)$ であるため，p_x は p_x の近傍の点 $p_y \in N(p_x)$ の $\triangle_y^{\min}$ に含まれない．したがって，点の削除において $\triangle_y^{\min}$ の更新は必要ない．

例 2. 図 2 を用いて有向近接グラフにおける点の削除の例を説明する．図 2a のグラフについて考える．$P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ であり，各点の P への挿入された順序が $o(p_i) \prec o(p_j)(i < j)$ であるとき，削除される点は p_1 である．p_1 を削除したとき，図 2b のように p_1 から外向きの辺のみ削除され，残った点の $\triangle^{\min}$ に影響を与えない．したがって，$\triangle^{\min}$ の更新の必要はない．

点の挿入. 点 p_x を P に挿入するとき，まずは p_x を中心に半径 r で範囲検索を行い，範囲内の点 p_y との辺 e_{p_y,p_x} を作り，P の有向近接グラフを更新する．次に，各 p_y について e_{p_y,p_x}

表 1: パラメータ

パラメータ	値
k	10/50/**100**/500/1000
r	**0.01**
W	500,000/**1,000,000**/1,500,000/2,000,000

を 1 辺とする三角形を列挙し，$\triangle_y^{\min}$ を更新する．これらの操作では，有向近接グラフを用いているため，三角形を重複なしで列挙できる．

Top-k 結果の更新．近接グラフの場合と同様に行うが，三角形の列挙を重複なく効率的に行うことができる．

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 3 つのアルゴリズムの評価を行った．

- 文献 [20] のアルゴリズムをスライド毎に行う単純なアルゴリズム (Naive Algorithm：NA と表記).
- 本研究における提案アルゴリズム (Proposed Algorithm：PA と表記).
- 提案アルゴリズムを最適化したアルゴリズム (Optimization Proposed Algorithm：OPA と表記).

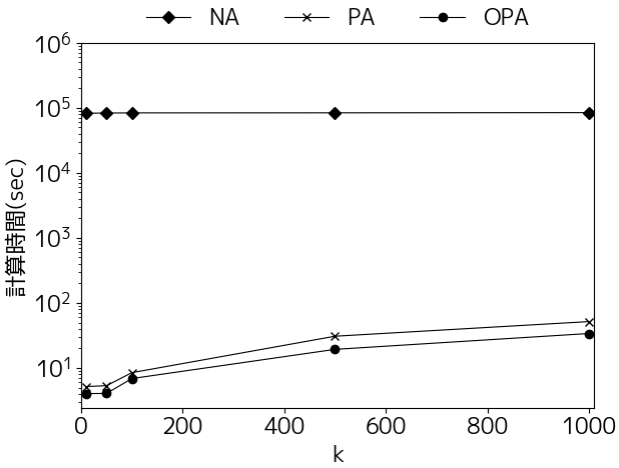
NA，PA，および OPA はすべて C++で実装されており，g++9.3.0 の-O3 最適化オプションを付けてコンパイルされている．また，すべての実験を ubuntu(WSL) が搭載された 3.6GHz Intel Core i9-9900K CPU および 128GB RAM で構成される計算機上で行った．

評価指標．本実験は，カウントベースウィンドウを用いており，新しい点を 1 点ずつ挿入して Top-k 結果の更新を行う．そして，10000 回のウィンドウのスライドで生じた計算時間を評価した．

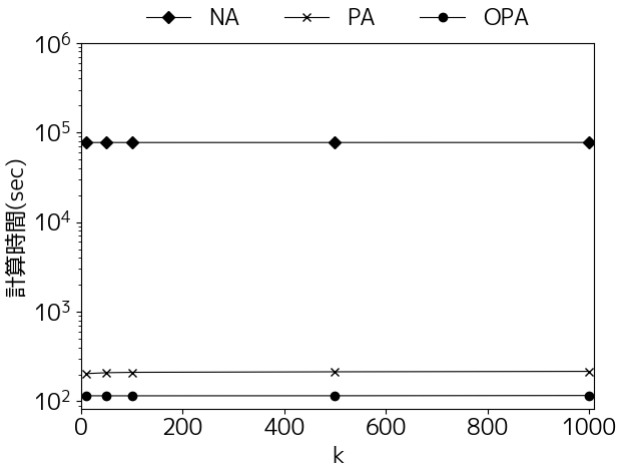
データセット．実験で用いるデータは Places[1] および CaStreet[2] の 2 つである．Places はアメリカの公共施設の位置情報が緯度経度で記録されたものである．データ総数は 9,356,750 である．CaStreet はカリフォルニアの道の位置情報が緯度経度で記録されたものである．また，データが最小外接矩形であるため，2 つの位置座標を別のデータとして扱っている．データ総数は 4,499,454 である．

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．提案アルゴリズムの性能への影響がないため，r の値は変化させない．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．

1：https://archive.org/details/2011-08-SimpleGeo-CC0-Public-Spaces
2：http://chorochronos.datastories.org/?q=node/59



(a) Places



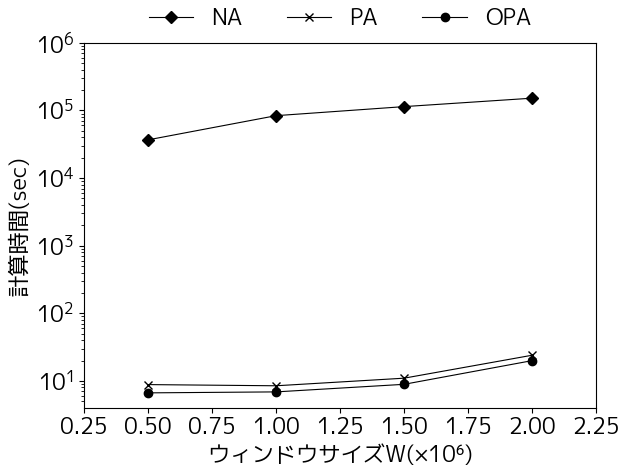
(b) CaStreet

図 3: k の影響

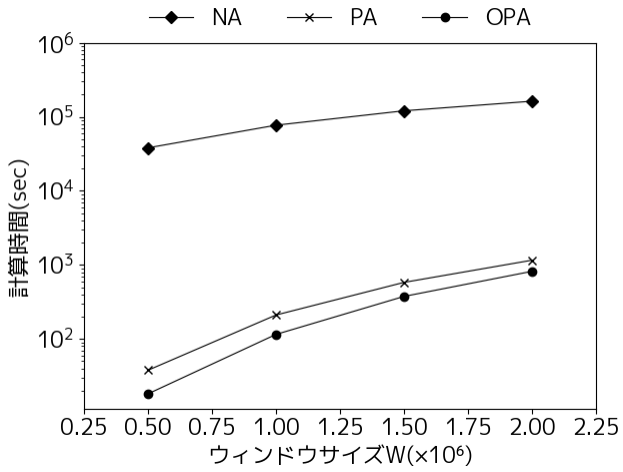
5.2 実 験 結 果

出力サイズ k の影響．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，OPA が最も効率的に Top-k 結果を更新できることがわかる．NA は Places および CaStreet のどちらのデータセットに対しても同程度の計算時間であるのに対して，PA および OPA は CaStreet の方が Places よりも計算時間が大きくなった．これは，CaStreet の方が Places よりも Top-k 結果の更新が多く行われたためである．また，PA および OPA が CaStreet では計算時間が k に関係なく一定であった．これは，CaStreet のデータ分布が均等であり，$k = 10$ のときと $k = 1000$ のときで Top-k 番目の三角形の重みに差がないためであると考えられる．

ウィンドウサイズ W による影響．W 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，W 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．W が 1,000,000 のときの計算時間を比較すると OPA の方が NA よりも Places では約 10000 倍早く，CaStreet では約 700 倍早い．



(a) Places



(b) CaStreet

図 4: W の影響

6 結　　論

本論文では，上位 k 個の重み付き三角形をスライディングウィンドウ上で効率的かつ正確にモニタリングするアルゴリズムを提案した．提案アルゴリズムは，Top-k 結果になり得る三角形のみを保持することにより，Top-k 結果の更新を効率的に行うことができる．また，有向近接グラフを用いることにより削除に頑健である．実データを用いた評価実験により，提案アルゴリズムが単純な手法に比べて非常に効率的に Top-k モニタリングを行うことができることを確認した．

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 JST CREST（JPMJCR21F2）の支援を受けたものである．

文　　献

[1] D. Amagata and T. Hara, “Monitoring maxrs in spatial data streams,” In EDBT, pp.317–328, 2016.

[2] D. Amagata and T. Hara, “A general framework for maxrs and maxcrs monitoring in spatial data streams,” ACM Transactions on Spatial Algorithms and Systems (TSAS), vol.3, no.1, pp.1–34, 2017.

[3] D. Amagata and T. Hara, “Identifying the most interactive object in spatial databases,” In ICDE, pp.1286–1297, 2019.

[4] T. Chan, R. Cheng, and M. Yiu, “Quad: quadratic-bound-based kernel density visualization,” In SIGMOD, pp.35–50, 2020.

[5] S. Chatterjee, M. Connor, and P. Kumar, “Geometric minimum spanning trees with geofilterkruskal,” In SEA, pp.486–500, 2010.

[6] D. Choi, C. Chung, and Y. Tao, “A scalable algorithm for maximizing range sum in spatial databases,” PVLDB, vol.5, no.11, pp.1088–1099, 2012.

[7] Y. Fang, R. Cheng, G. Cong, N. Mamoulis, and Y. Li, “On spatial pattern matching.,” In ICDE, pp.293–304, 2018.

[8] Y. Fang, Y. Li, R. Cheng, N. Mamoulis, and G. Cong, “Evaluating pattern matching queries for spatial databases,” The VLDB Journal, vol.28, no.5, pp.649–673, 2019.

[9] Y. Fang, Z. Wang, R. Cheng, X. Li, S. Luo, J. Hu, and X. Chen, “On spatial-aware community search,” IEEE Transactions on Knowledge and Data Engineering, vol.31, no.4, pp.783–798, 2018.

[10] K. Feng, G. Cong, C. Jensen, and T. Guo, “Finding attribute-aware similar region for data analysis,” PVLDB, vol.12, no.11, pp.1414–1426, 2019.

[11] T. Guo, K. Feng, G. Cong, and Z. Bao, “Efficient selection of geospatial data on maps for interactive and visualized exploration,” In SIGMOD, pp.567–582, 2018.

[12] R. Kumar, P. Liu, M. Charikar, and A. Benson, “Retrieving top weighted triangles in graphs,” In WSDM, pp.295–303, 2020.

[13] V. Pandey, A. Kipf, T. Neumann, and A. Kemper, “How good are modern spatial analytics systems?,” PVLDB, vol.11, no.11, pp.1661–1673, 2018.

[14] H. Park, S. Myaeng, and U. Kang, “Pte: Enumerating trillion triangles on distributed systems,” In KDD, pp.1115–1124, 2016.

[15] S. Shaikh, A. Matono, and K. Kim, “A distance-window approach for the continuous processing of spatial data streams,” International Journal of Multimedia Data Engineering and Management (IJMDEM), vol.11, no.2, pp.16–30, 2020.

[16] Y. Tong, J. She, B. Ding, L. Chen, T. Wo, and K. Xu, “Online minimum matching in real-time spatial data: experiments and analysis,” PVLDB, vol.9, no.12, pp.1053–1064, 2016.

[17] Y. Wang, S. Yu, L. Dhulipala, Y. Gu, and J. Shun, “Geograph: A framework for graph processing on geometric data,” ACM SIGOPS Operating Systems Review, vol.55, no.1, pp.38–46, 2021.

[18] J. Yu and M. Sarwat, “Geosparkviz: a cluster computing system for visualizing massive-scale geospatial data,” The VLDB Journal, vol.30, no.2, pp.237–258, 2021.

[19] C. Zhang, Y. Zhang, W. Zhang, L. Qin, and J. Yang, “Efficient maximal spatial clique enumeration,” In ICDE, pp.878–889, 2019.

[20] 谷口凌亮, 天方大地, 原隆弘, “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,” 情報科学技術フォーラム (FIT), vol. 第 2 分冊, pp.1–8, 2021.