Aggregate Nearest Neighborhood Queries

高木　颯汰†　　陳　　漢雄††　　古瀬　一隆†††　　北川　博之††

† 筑波大学コンピュータサイエンス専攻　〒 305–8577 茨城県つくば市天王台 1 丁目 1–1
†† 筑波大学システム情報系　〒 305–8577 茨城県つくば市天王台 1 丁目 1–1
††† 白鴎大学経営学部　〒 323–8586 栃木県小山市駅東通り 2 丁目 2–2
E-mail: †s1920672@u.tsukuba.ac.jp, ††chx@cc.tsukuba.ac.jp, †††furuse@fc.hakuou.ac.jp,
††††kitagawa@cs.tsukuba.ac.jp

キーワード　Nearest neighborhood query, Spatial database, R-tree structure, Grid index.

1　はじめに

空間データベースにおける近傍クエリは、パターン認識や POI(Point of Interest) を活用したサービス、LBS(Location-based Services) などのさまざまなアプリケーションにおいて重要な問題である。 近傍探索について多くの研究が行われ、その中近年 Balanced Nearest Neighborhood Query(BNNH) [1] が提案されている。 与えられたクエリ点に最も近いオブジェクトを返す伝統的な最近傍探索と比較して、BNNH は最近傍のクラスタをレスポンスとして返す。 つまり、BNNH クエリは最近傍クエリのグループバージョンである。また Aggregate Nearest Neighbor Queries(ANN) [2] はクエリ点を複数受け取り、そのクエリ群に対して最近傍の点を返す。これらの研究では複数クエリに対する最近傍のクラスタを探索することはできないという問題があった。これを踏まえて本研究では複数クエリに対する最近傍クラスタの探索を行う。

以下に本研究の活用例を挙げる。図 1 は旅行行程を組む際の例である。出発地点と到着地点が決まっていて、立ち寄る観光地を決めようとする際に、移動距離を最短にして多くの観光地を巡りたいとする。この場合、出発地点と到着地点がクエリとなり、観光スポットがデータセットとなる。提案手法ではクエリからクラスタまでの距離の合計が小さくなり、かつより密集しているクラスタを探索する。

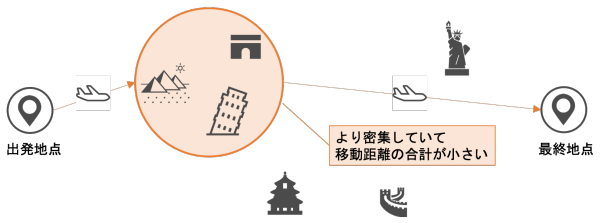

図 1　合計距離を最小とする例

図 2 は集合場所を決める際の例である。全員が集まりやすい場所であり、アクティビティが多くある地点を集合場所としたい。この場合各ユーザーの地点がクエリとなり、アクティビティスポットがデータセットとなる。提案手法ではクエリからクラスタまでに距離の最大値が最小となり、かつより密集しているクラスタを探索する。図 3 は解とならないクラスタの例である。クラスタの密集度は図 2 のクラスタと同程度だが、クエリとクラスタの距離の最大値が図 2 のクラスタの最大値よりも大きいため集合場所としては不適である。

図 2　最大距離を最小とする例

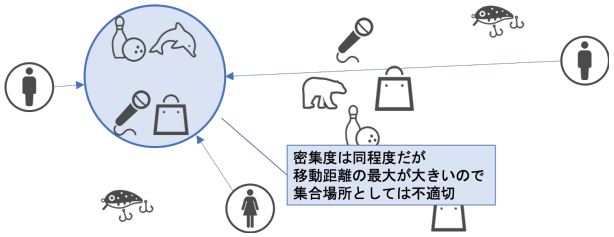

図 3　最大距離が最小になっていない例

2　関連研究

2.1　R-Tree を用いた最近傍探索

データベース管理システム、データマイニングおよび情報検索などの多くの分野において、基本的かつ重要な問題は、(k) 最近傍クエリ問題 (NN、または kNN) である。クエリ点 q とデータセット P が与えられると、NN はデータセット P から q に最も近い点を返す。 NN クエリを優れたパフォーマンスで解決するために多くの研究が行われてきた。最も一般的な解決策の 1 つは、R ツリーベースの処理である。[3] [4]。 R-tree [5] は、空間内の近接点を最小外接矩形 (MBR) でグループ化し、MBR をツリー構造で管理する空間インデックスである。Hjaltason et

al [4] は、NN クエリを解決するために R ツリーを用いた最良優先探索を提案した。最良優先探索は次に探索する最も望ましいノードを選択するように探索をする。具体的には、クエリ点 q までの最小距離の順序で R 木の MBR を含む優先待ち行列を作成する。

2.2 複数問い合わせ点の集約最近傍

従来の NN クエリのように単一点間の関係を評価するだけでは実際のアプリケーションでは活用できないため、さまざまな種類の NN クエリの変種が研究されてきた。 Tao らは複数のクエリ点までの最小集約距離を持つ点を取得するために、Aggregate Nearest Neighbor Queries(ANN) [2] を提案した。 ANN は質問点群 Q を入力し、Q までの最小の集合距離を有する最も近い単一点を見出すことを目的としている。したがって、本研究は ANN と比べてクエリを複数受け取る部分は合致しているが、クラスタを探索する点で異なっている。 Deng らは、Group nearest neighbor queries(GNG) [6] を提案した。これは、クエリ群を入力に受け取り、グループを返す。GNG ではレスポンスとなるグループの評価を各点のクエリからの距離で行なっているのに対して、本研究では点をクラスタと捉え、クラスタの密集度も評価に入れている部分で異なっている。Nearest neighborhood search(NNH) [1] は NN クエリを拡張し、クエリ点 q に対して最近傍クラスタを返す。具体的には半径 ρ の円の中心を返す。この円は k の点を含むクエリから最近傍の円である。Balanced Nearest Neighborhood Query(BNNH) [7] は NNH を拡張し、半径を固定長から変更可能にし、平滑パラメータを用いてクラスタの大きさとクエリ点からの距離のバランスを考慮した。

3 提 案 手 法

ANN [2] は複数クエリを入力にとり、単一点を出力としていた。また BNNH [7] は単一点を入力にとり、クラスタを出力としていた。本研究ではこれらの研究では探索できない、複数クエリを入力として、出力をクエリの最近傍のクラスタとする。BNNH [7] ではクラスタを円と定義し、密集度を円の半径としていたが、円を形成するのに計算が必要な上に、クラスタ内部のデータの偏りを検知できない問題があった。本研究ではクラスタの形を円に限定せず、クラスタ内の分散で密集度を評価することで、計算量の削減とより密集したクラスタを探索することを実現している。

以下の説明で使用する変数について表 1 に示す。ユーザーからクエリ群 Q、クラスタサイズ k、クラスタの距離と密集度の好みのバランスを設定する α,β が与えられ、最近傍のクラスタを返す。

3.1 集約最近傍

クエリが複数になることにより、クエリ群 Q と点 p の最近傍について定義が必要になる。本研究では”合計”、”最大”を最近傍の指標として挙げる。

3.1.1 合計値の集約最近傍

各クエリからクラスタまでの距離の合計が最小となるクラスタを最近傍とする。図 1 で示した例のように、ユーザーは 1 人でクエリとして経路の地点を設定することで移動距離が最小となるような問題を解決できる。点 p とクエリ Q の集約距離を求める式を (1) に示す。

要素名	説明
p, P	点, 点のクラスタ
q, Q	クエリ点, クエリ群
C, V_C	クラスタ, クラスタ内の分散
O_C	クラスタの重心
$dist(p, q)$	点 p と点 q のユークリッド距離
$f(p, C)$	点 p とクラスタ C の集約距離
α,β	ユーザー定義の平滑パラメータ
k	ユーザー定義のクラスタサイズ
$\Delta(C, Q)$	C と Q の複合類似度
MBR	R-tree の最小外接矩形
n	グリッドを用いた手法におけるセル数

表 1 記号の説明

$$f_{sum}(p, Q) = \sum_{i=1}^{n} dist(p, q_i) \tag{1}$$

3.1.2 最大値の集約最近傍

各クエリからクラスタまでの距離の最大値が最小となるクラスタを最近傍とする。図 2 で示した例のように、クエリがユーザーの位置で、集合場所を決める際に中間地点を選択する問題を解決できる。

$$f_{max}(p, Q) = max_{i=1}^{n} dist(p, q_i) \tag{2}$$

3.2 クラスタ内分散

クラスタの密集度をクラスタ内分散 V で表す。点 $p_1, \cdots p_n$ が構成するクラスタ P の重心が O_P の時、P の分散を求める式を以下に示す。

$$V_P = \frac{1}{|P|} \sum_{i=1}^{n} \{dist(p_i, O_P)\}^2 \tag{3}$$

3.3 クラスタの評価関数

クラスタの密集度はクラスタ内分散 V_C で表すとする。また、クエリとクラスタの距離の評価は式 (1)-(2) で行う。BNNH [7] で提案されている合成関数を変形し、本研究で用いる評価関数 Δ(.) を以下に示す。

$$\Delta(C, Q) = \alpha \cdot f(O_C, Q) + \beta \cdot V_C \tag{4}$$

式 (4) を用いて平滑パラメータ α,β の値を変えることによってユーザーの好みを反映できる。例えば $\alpha > \beta$ の場合はクラスタの位置が近い方が好ましいことを表し、$\alpha < \beta$ の場合はクラスタの密度が高い方が好ましいことを表し、$\alpha = \beta$ の場合はクラスタの位置と密度が同程度重要であることを表す。図 4 には 3 つのクラスタ C_1、C_2、C_3 がある。この例では C_2 がクラスタの密度が一番大きく、クエリに近いので解となる。C_1 と C_3 を比較すると密度は C_3 の方が高いが、クエリとの距離は C_1 の方

が小さい値になっている。このような場合にクラスタを単純比較できないため、評価関数を用いて比較を行う。

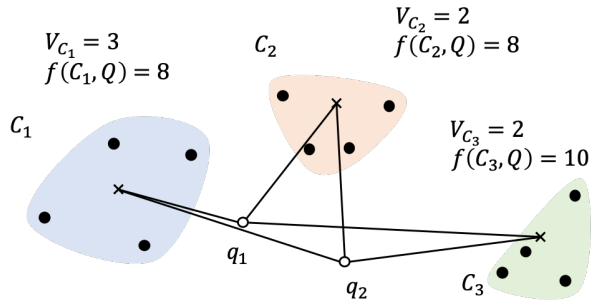

図 4　クラスタの距離と密集度の関係

3.4　フィルタリング

全てのデータ点 p に対してクラスタを作成し、$\Delta(.)$ の値を計算することはとても非効率である。これを省略するために、計算を行う前にフィルタリングを行う。評価式 (4) の両辺を $\alpha+\beta$ で割ると次の式となる。

$$\frac{\Delta(C,Q)}{\alpha+\beta} = \frac{\alpha}{\alpha+\beta}\cdot f(O_C,Q) + \frac{\beta}{\alpha+\beta}\cdot V_C \tag{5}$$

$\frac{1}{\alpha+\beta}$ はデータセットに関係なく一定であるので、$\frac{\Delta(C,Q)}{\alpha+\beta}$ を考えることは $\Delta(C,Q)$ を考えることと同じであると言える。以降、$\frac{\Delta(C,Q)}{\alpha+\beta}$ は $\Delta'(C,Q)$ と表す。

$\alpha < \beta$ の時、α と β は以下の式を満たす。

$$\frac{\beta}{\alpha} > 1 \tag{6}$$

式 (6) より式 (5) は以下の通りに変形することができる。

$$\Delta'(C,Q) = \frac{\alpha}{\alpha+\beta}\cdot f(O_C,Q) + \frac{\beta}{\alpha+\beta}\cdot V_C$$

$$\frac{\alpha+\beta}{\alpha}\cdot\Delta'(C,Q) = f(O_C,Q) + \frac{\beta}{\alpha}\cdot V_C$$

$$\frac{\alpha+\beta}{\alpha}\cdot\Delta'(C,Q) > f(O_C,Q) + V_C \tag{7}$$

式 (7) より以下の定理が導出できる。

定理 1. クエリ Q と最善のクラスタ C があった時、クラスタとの集約距離が閾値 $\frac{\alpha+\beta}{min(\alpha,\beta)}\cdot\Delta(C,Q)$ よりも離れている点は最善のクラスタにはなり得ない。

証明 1. クエリ Q、現時点での最善のスコアを持つクラスタ C_1、C_1 に属さない点 p があり、$\alpha < \beta$ であるとする。式 (7) により、p と他の点で任意のクラスタ C_2 を作成すると、以下の式が成り立つ。

$$f(p,Q) \leqq f(O_{C_2},Q) + V_{C_2} < \frac{\alpha+\beta}{\alpha}\cdot\Delta(C_2,Q)$$

したがって、p が現在の最善のクラスタ C_1 によって作成された閾値 $\frac{\alpha+\beta}{\alpha}\cdot\Delta(C_1,Q)$ よりも遠くにある場合、$\Delta(C_1,Q) < \Delta(C_2,Q)$ と推測できる。したがって、p を含むクラスタは C_1 と比較して良いスコアを持たない。 □

3.5　ポイントベースの手法

以上の式を用いてクラスタを探索する手法を示す。まずはシンプルに探索するポイントベースの手法である。これは各データセットの点に対して $k-1$ 点の最近傍点を探し出し、クラスタを計算する。このクラスタの評価関数の値がその段階での最善解よりも良ければ解として更新する。クラスタを作成する前に式 (**??**) を用いてフィルタリングを行う。ポイントベースの手法の擬似アルゴリズムを Algorithm1 に示す。

Algorithm 1 ポイントベースの手法

Input: P, Q, k, α, β
Output: C

```
bound ← ∞
C ← ∅
for each p_i ∈ P do
    if f(p_i, Q) < bound then
        pSet ← p_i and its k − 1 nearest points
        C_t ← create Cluster with pSet
        if Δ(C_t, Q) < Δ(C, Q) then
            C ← C_t
            update bound with Δ(C_t, Q)
return C
```

Algorithm 2 R-tree を用いる手法

Input: P, Q, k, α, β
Output: C

```
p_a ← the nearest point to q from P.
pSet ← p_a and its k − 1 nearest points
C ← create Cluster with pSet
bound ← calculate the filter condition with Δ(C, Q)
heep ← Rtree.root
while heap is not empty do
    E ← heap.front()
    if mindit(E, Q) < bound then
        if E is none-leaf node then
            heap ∩ E.children
        if E is leaf node then
            for each p ∈ E do
                if dist(p_i, Q) < bound then
                    pSet ← p_i and its k − 1 nearest points
                    C_t ← create Cluster with pSet
                    if Δ(C_t, Q) < Δ(C, Q) then
                        C ← C_t
                        update bound with Δ(C_t, Q)
return C
```

3.6　R-tree を用いる手法

ポイントベースの手法ではフィルタリングが点単位であり、R-tree の *MBR* 単位でフィルタリングを行えば効率的にフィルタリングができ、計算量の減少が見込める。データセットを R-tree 上に配置し、Q も *MBR* として配置する。各ノードと Q の距離でフィルタリングを行うことで、一度にフィルタリング

を行うことができる。R-tree を用いる手法の擬似アルゴリズムを Algorithm2 に示す。

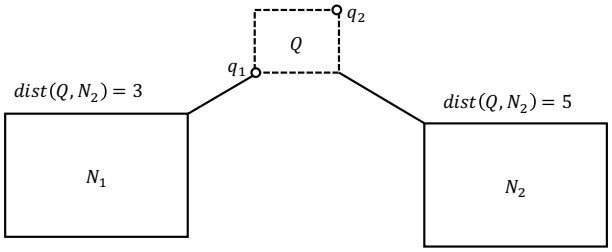

図 5　R-tree を用いる手法

3.7　グリッドを用いる手法

R-tree のを用いた手法の欠点は、走査が行われる際に毎回ルートノードから開始される点が挙げられる。また、NN 探索のフィルタリングにおいて、クエリ点近くから探索を行うと、フィルタリングの閾値が小さくなり、フィルタリング量が多くなることが報告されている [7]。これらより、既存研究 [7] と同様にデータセットを 2 次元空間のグリッドに配置する。クエリ点が 1 つの時の例を図 6 に示す。クエリ q' が存在するセルを特定し、その周辺のセルを探索することができる。これによって効率的に探索が行うことができる。閾値は図 6 では点線の円で表され、点線の円の中のセルのみ探索を行えば良いということになる。本研究では複数クエリの際に探索するセルの起点をクエリ群の重心とする。グリッドを用いる手法の擬似アルゴリズムを Algorithm3 に示す。

Algorithm 3 グリッドを用いる手法

Input: P, Q, k, α, β
Output: C

```
p_a ← the nearest point to q from P.
pSet ← p_a and its k − 1 nearest points
C ← create Cluster with pSet
bound ← calculate the filter condition with Δ(C, Q)
nextCells ← get surround cells of q
while nextCells locate in bound do
    pNext ← retrieve newarest point to q from nexCells
    pSet ← pNext and its k − 1 nearest points
    C_t ← create Cluster with pSet
    if Δ(C_t, Q) < Δ(C, Q) then
        C ← C_t
        update bound with Δ(C_t, Q)
    nextCells ← the next round of cells
return C
```

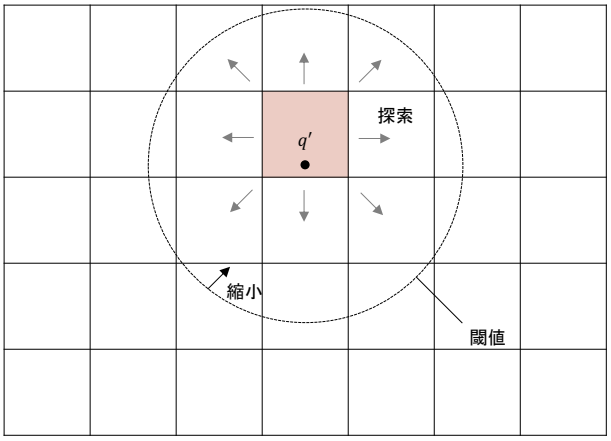

図 6　グリッドを用いる際のフィルタリング

4　実　　験

4.1　実 験 環 境

提案手法の有用性を示すために、実験を行った。実験環境をいかに示す。

- OS: macOS 10.15.2 (19C57)
- CPU: 2.7 GHz Intel Core i5
- Memory: 16 GB 1867 MHz DDR3
- 開発言語: C++

4.2　データセット

実データと合成データを用いて実験を行った。実データはアメリカ国勢調査 TIGER プロジェクト [8] から取得したデータ NE(123,593 点) を使用した。合成データは一様分布に基づくデータセットを作成し、使用した。

4.3　パラメータ

クエリはデータセットからランダムに選択した。出力結果となるクエリサイズ k のデフォルト値は 50、平滑パラメータ α, β は共に 0.5、グリットを用いる手法で分割するグリット数は 128^2 である。全ての実験結果は 10 回の試行の平均を用いている。

4.4　実 験 結 果

クエリサイズを変化させた結果を図 7 と図 8 に示す。集約距離をどちらに設定してもグリッドを用いた手法が他の 2 手法に比べて高速であることがいえる。これはグリッドを用いた手法ではクエリの近くから探索するので、フィルタリングに用いる閾値が早期に適切に設定され、フィルタリング量が大きかったためである。集約距離を最大に設定すると、ポイントベースと R-tree を用いた手法では集約距離合計の場合よりも速い結果となった。これはフィルタリングに用いる閾値と比較する値が、集約距離合計の際にはクエリからの距離の平均であるのに対して、集約距離合計の際にはクエリからの距離からの最大値であるので、フィルタリング量が大きかったと考えられる。

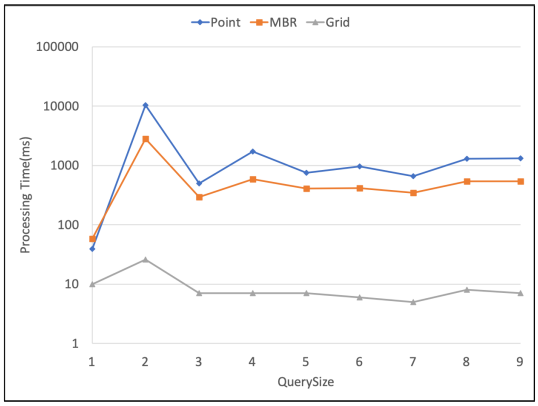

図 7　f = *sum*, NE, α = 0.5 , β = 0.5 , |P| = 123,593, k = 50, n = 128^2

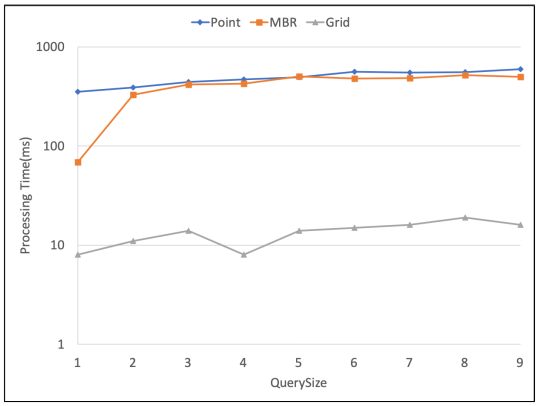

図 8　f = *max*, NE, α = 0.5 , β = 0.5 , |P| = 123,593, k = 50, n = 128^2

クラスタサイズを変化させた結果を図 9 と図 10 に示す。両方の集約距離でグリッドが最速となった。

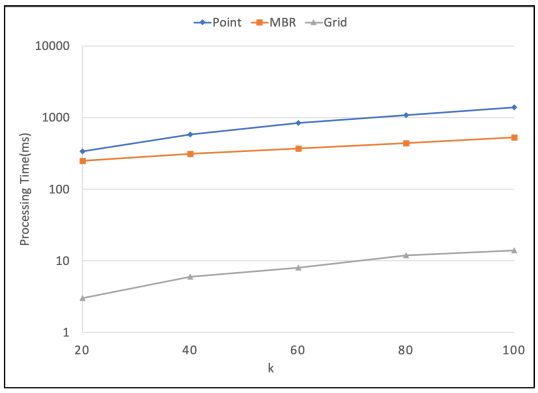

図 9　f = *sum*, NE, α = 0.5 , β = 0.5 , |P| = 123,593, |Q| = 5, n = 128^2

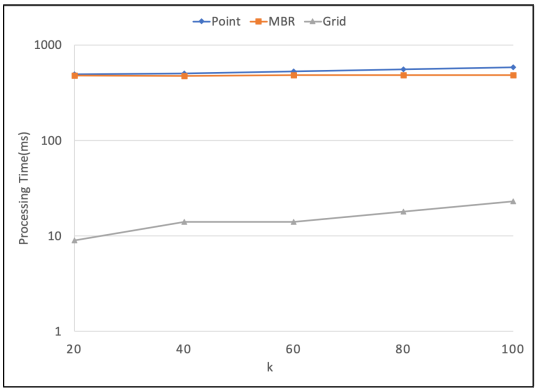

図 10　f = *max*, NE, α = 0.5 , β = 0.5 , |P| = 123,593, |Q| = 5, n = 128^2

データサイズを変化させた結果を図 11 と図 12 に示す。グリッドを用いた手法が最速となった。3 手法ともデータサイズが増えるとに処理時間が増えているが、グリッドの増加は緩やかで合った。これは探索をクエリの近くから行なっているため、データサイズ の影響を受けにくいからと考えられる。

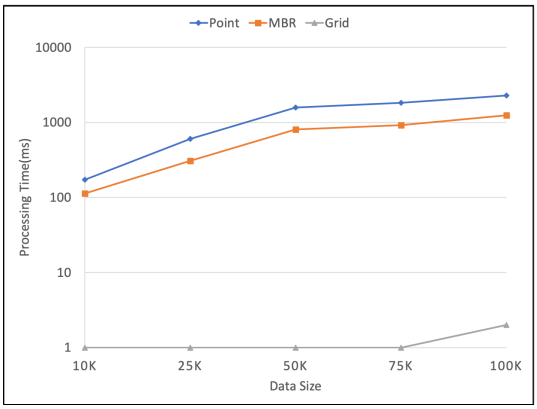

図 11　f = *sum*, UN, α = 0.5 , β = 0.5 ,k = 50, |Q| = 5, n = 128^2

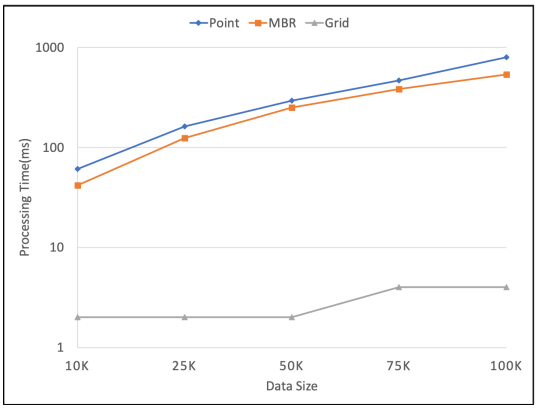

図 12　f = *max*, UN, α = 0.5 , β = 0.5 ,k = 50, |Q| = 5, n = 128^2

5　結論と今後の課題

最近傍クラスタを見つけることは、クエリ処理とデータマイニングにおいて重要な問題である。既存研究では複数のクエリ

の最近傍のクラスタを探索することはできなかった。本研究では集約距離を定義することで、複数クエリの最近傍のクラスタ探索を実現した。クラスタを探索する手法として、ポイントベース、R-tree、グリッドを用いる 3 つの手法を提案した。合成データと実データを用いた実験では集約距離合計ではグリッドを用いた手法が最も効率が良いことを示した。

今後の課題として、平滑パラメータの値によってパフォーマンスが落ちる傾向にあり、これを解消したいと考えている。

謝　　辞

本研究は JSPS 科研費 JP19K12114 の助成を受けたものである.

文　　献

[1] D. Choi and C. Chung. Nearest neighborhood search in spatial databases. In *2015 IEEE 31st International Conference on Data Engineering*, pages 699–710, April 2015.

[2] Dimitris Papadias, Yufei Tao, Kyriakos Mouratidis, and Chun Kit Hui. Aggregate nearest neighbor queries in spatial databases. *ACM Trans. Database Syst.*, 30(2):529–576, June 2005.

[3] King Lum Cheung and Ada Wai-Chee Fu. Enhanced nearest neighbour search on the r-tree. *ACM SIGMOD Record*, 27(3):16–21, 1998.

[4] Gísli R Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems (TODS)*, 24(2):265–318, 1999.

[5] Antonin Guttman. *R-trees: A dynamic index structure for spatial searching*, volume 14. ACM, 1984.

[6] Dimitris Papadias, Qiongmao Shen, Yufei Tao, and Kyriakos Mouratidis. Group nearest neighbor queries. In *Proceedings. 20th International Conference on Data Engineering*, pages 301–312, April 2004.

[7] S. Le, Y. Dong, H. Chen, and K. Furuse. Balanced nearest neighborhood query in spatial database. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–4, Feb 2019.

[8] US Census Bureau. Chorochronos.datastories.org. http://chorochronos.datastories.org/?q=node. (参照 2020-01-09).