D-003

類似軌跡探索における高速かつ高精度なスケッチ間類似度

電気通信大学　情報理工学研究科　情報・ネットワーク工学専攻

河野 大督　　古賀 久志

1　はじめに

現在，GPS やセンサー，携帯電話のデータなどから様々な形で位置情報を得ることができ，それらの情報は社会科学を始め，通信ネットワークや交通機関に至るまで広い領域で応用されている. 得られた位置情報のデータは応用する際に，位置分析と呼ばれる分析が必要となる．その中で類似軌跡探索は，得られた軌跡データベースから似ている軌跡を見つけ出す問題であり，トラフィック分析やライドシェアリングのスケジューリング最適化などの応用領域を持つ．近年は，取得可能な軌跡の数が増加し，さらに各軌跡のデータ量も巨大化しているため，類似軌跡検索の計算量が大きくなりがちである．そのため，大規模なデータを扱うには，類似軌跡探索を行うアルゴリズムの高速化が非常に重要である．

通常，軌跡間の距離はハウスドルフ (Hausdorff) 距離，フレシェ (Fréchet) 距離，Dynamic time warping(DTW) などで測定される．しかし，ハウスドルフ距離は軌跡を頂点の集合としか捉えておらず，頂点の順序を考慮しないという問題がある．また, フレシェ距離と DTW は計算量が大きく，大規模なデータセットにそのまま適用することが出来ない．そこで，現在は高速に類似軌跡探索を行うための手法として近似検索が用いられている．特に，確率的な類似検索アルゴリズムである Locality sensitive hashing(LSH) を軌跡データに拡張するアプローチがいくつか提案されている．例えば，Anne らは領域をグリッドで分割し，軌跡内の点をグリッドの格子点に量子化することで可変長のハッシュ値を算出する手法を提案した [1]．また，Maria らは軌跡が存在する領域にランダムにディスクを設置し，ディスクに対する軌跡の出入りを観測して可変長のハッシュ値を算出する手法を提案した [4]．こうしたハッシュ値は元の軌跡データをコンパクトに要約したものと見なせるので，ハッシュスケッチとも呼ばれる．

上記の既存研究では，2 つの軌跡 p と q のハッシュ値 $h(p) = h(q)$ となる際に，p と q の軌跡間距離が小さくなる確率が高いことを理論的に証明している．しかし，ハッシュが一致した場合のみにしか注目していない．すなわち，$h(p) \neq h(q)$ となり p と q のスケッチが一致しない時，$h(p)$ と $h(q)$ のスケッチ間類似度から p と q の軌跡間類似度を推定するというアプローチについては，研究が進んでいない．Maria らはスケッチ間の文字列編集距離をスケッチ間類似度としたが，フレシェ距離や DTW との相関が高くないことも述べている [4]．スケッチ間類似度が軌跡間距離と近似的に対応するような手法には，Jeff らの手法が存在する [2]．この手法は，軌跡の存在領域に複数のランドマークを設けて，各ランドマークから軌跡の最も近い位置までのベクトルをスケッチとする．スケッチ間類似度はベクトルの差の大きさの二乗平均平方根を用いているため，スケッチ間類似度が実際の軌跡間距離と近似的に対応する．しかし，この手法はランドマークによって値が大きく影響されるが，最適なランドマークの配置を見つけ出すのが難しいという問題点がある．

本研究では，Maria らによるハッシュスケッチ [4] よりも，高速に算出可能でフレシェ距離とスケッチ間類似度との相関が高いという 2 つの良い性質を持った新たなハッシュスケッチの生成法を提案する．提案手法は軌跡のディスクに対する出入りを記録するのではなく，軌跡の各点に対して，その点を包含する最近傍のディスクを記録しスケッチとする．同時に各ディスクの中心座標を記憶しておき，必要に応じてスケッチ番号列から中心座標列への変換を可能にする．1 頂点につき最大で 1 つのディスクにしか関連付けられないため，ディスクに対する出入りを記録する場合に比べ, スケッチが短縮化され, スケッチ間類似度を高速に計算できる．また，Maria らの手法 [4] ではスケッチにした時点で位置情報を捨てていたが，提案手法ではディスクの中心座標列に変換可能なスケッチにすることによって，スケッチが 2 次元情報を持つ．スケッチが 2 次元情報を持つと，スケッチ間類似度の比較にフレシェ距離を用いることができ，実軌跡のフレシェ距離との相関が向上する．実軌跡データを用いた実験評価により，提案するスケッチ間類似度が Maria らによるスケッチ間類似度 [4] よりも，フレシェ

距離との相関を劇的に改善できることを示す．

2 予備知識

本節では，本研究に関する基礎的な知識となる軌跡に対する離散フレシェ距離と Locality Sensitive Hashing(LSH) について述べる．軌跡間距離は，1 節でも述べたようにいくつか手法があるが，本研究では点数に影響を受けず，頂点数がバラバラな実軌跡データに応用できるフレシェ距離を扱う．

2.1 離散フレシェ距離

フレシェ距離には，順序情報をもつ曲線に対する連続フレシェ距離と順序情報をもつ頂点の集合に対する離散フレシェ距離の 2 種類が存在する．本研究では軌跡を 2 次元の位置情報を持つ頂点の順序付き集合としてとらえるため，後者の離散フレシェ距離について述べる．

2 つの軌跡を $A = \{p_1, p_2, p_3, ..., p_n\}, B = \{q_1, q_2, q_3, ..., q_m\}$ と表し，軌跡の順序情報を崩さないという条件の下で，軌跡 A, B 上の各頂点をもう一方の軌跡の頂点とマッチングさせる．軌跡の順序情報を崩さないというのは，軌跡 A の p_a と軌跡 B の q_b がマッチングした際に，軌跡 A の次の頂点 p_{a+1} は軌跡 B の $q_{b+z}(z = 0, 1, 2, ..., n-b)$ としかマッチングできないことを指す．マッチングした頂点のペアを $(p_{\alpha(t)}, q_{\beta(t)})$ とするとき，すべてのマッチングしたペアの中での頂点間距離 $d(p_{\alpha(t)}, q_{\beta(t)})$ の最大値が最小になるようなマッチングを見つけ出し，その時の頂点間距離 $d(p_{\alpha(t)}, q_{\beta(t)})$ の最大値がフレシェ距離となる．マッチングの数を l とすると $l = \max\{m, n\}$ となるため，フレシェ距離を $d_F()$ として式 (1) で求められる．

$$d_F(A, B) = \min_{\alpha,\beta} \max_{t\in[1,l]} \{d(p_{\alpha(t)}, q_{\beta(t)})\}. \quad (1)$$

2.2 Locality Sensitive Hashing(LSH)

Locality Sensitive Hashing とは，似ているデータであるほどハッシュが一致する確率が高いハッシュ関数を用いて，データを整理する手法である．データ A, B 間の距離関数を $d(A, B)$ とし，距離の閾値を $R > 0$，パラメータを $c > 1$，確率を $k_1, k_2(k_1 > k_2)$ とすると，似ているデータであるほどハッシュが一致する確率 P_r が高いハッシュ関数 h は以下の性質をもつ．

- $if\ d(A, B) \leq R\ then\ P_r[h(A) = h(B)] \geq k_1$;
- $if\ d(A, B) \geq cR\ then\ P_r[h(A) = h(B)] \leq k_2$.

この場合のハッシュ関数 h の族は (R, cR, k_1, k_2)-sensitive であると呼ばれる．

3 従来手法

本節では，離散フレシェ距離に対する LSH の従来手法を 2 つ示す．1 つはハッシュ関数にグリッドを用いた演算を取り入れた LSH[1]，もう 1 つは Maria らの領域にディスクを配置し，それらに対する軌跡の出入りでスケッチを作成し，比較する手法である [4]．以下それぞれのアルゴリズムについて説明する．

3.1 グリッドを用いた LSH

Anne らは軌跡に対する LSH として，ハッシュ関数にグリッドを用いる手法を提案している [1]．この手法では，事前に軌跡の存在する領域 W に一辺の長さ δ のグリッドを配置し，このグリッドを用いて各軌跡のスケッチを算出する．以下で具体的な各軌跡に対するスケッチの導出方法を示す．

1. 軌跡の各頂点を最近傍なグリッドの格子頂点に置き換える．
2. 置き換えた後の軌跡の連続する頂点を取り除く．
3. 残った頂点列をハッシュ関数 h_g に対するハッシュ値とする．

ハッシュ値としては，グリッド格子の頂点列が得られる．そのため，2 つの軌跡がすべて同じ頂点に割り当てられた場合，ハッシュが一致したとみなす．ハッシュが一致する確率 P_g は，軌跡 A, B のハッシュ値を $h_g(A), h_g(B)$，頂点数を m, n として，$l_{min} = min\{m, n\}$，次元数を d，A, B のフレシェ距離を $d_F(A, B)$ とすると，式 (2) のようになる．

$$P_g(h_g(A) = h_g(B)) \geq 1 - \left(2dl_{min} \cdot \frac{d_F(A, B)}{\delta}\right). \quad (2)$$

このグリッドを用いた LSH の手法は，フレシェ距離や DTW に対する LSH の基礎的な考えになっている．

3.2 ディスクを用いた LSH

Maria らは，軌跡に対する LSH としてハッシュ関数にディスクを用いる手法をを提案している [4]．この手法では，事前に軌跡の存在する領域 W にランダムに半径 r のディスクを D 枚配置し，各ディスクにディスク番号 $d(d = 1, 2, 3, ..., D)$ を与える．このディスクに対

する軌跡 $T = (t_1, t_2, t_3, ..., t_m)$ の出入りを記録することでスケッチを算出する．以下で具体的な各軌跡に対するスケッチの導出方法を示す．

1. 軌跡の $j(j = 1, 2, 3, ..., m)$ 番目の頂点 t_j を含んでいるディスクを探索し，そのディスク番号の集合を $\mathbb{D}_j$ とする．
2. 軌跡の $j-1$ 番目の頂点 t_{j-1} を含んでいるディスクの集合 $\mathbb{D}_{j-1}$ と $\mathbb{D}_j$ を比較し，$\mathbb{D}_{diff} = \mathbb{D}_{j-1} \oplus \mathbb{D}_j$ となるディスク番号の集合 $\mathbb{D}_{diff}$ をスケッチに追加する．ただし，$j = 1$ の場合は，$\mathbb{D}_{j-1} = \varnothing$ とする．
3. 手順 2,3 を $j = 1$ から $j = m$ まで変化させて，スケッチを算出する．

手順 3 の比較を行うことで，軌跡が新しく入ったディスクの番号と外に出たディスクの番号がスケッチに格納されることになる．スケッチとしては，ディスクの番号列が得られる．

軌跡 A, B のスケッチ $H_d(A), H_d(B)$ が一致する確率 P_d は，軌跡 A, B の軌跡の頂点数 m, n として，$l_{min} = min(m, n)$ と，軌跡の存在する領域 W の広さ S, フレシェ距離 d_F を用いて，式 (3) のようにかける．

$$P_d[H_d(A) = H_d(B)] \geq 1 - \frac{4\pi r D l_{min}}{S} d_F(A, B). \quad (3)$$

また，定数 c を用いて，LSH の定義的に表すと，

- $if\ \ d_F(A, B) < \frac{2r}{c}\ \ then\ \ P_d[H_d(q) = H_d(p)] > 1 - \frac{8\pi r^2 D l_{min}}{cA}$;
- $if\ d_F(A, B) > 2r\ then\ P_d[H_d(q) = H_d(p)] = 0.$

上記からも分かるようにこの手法では，2 つの軌跡のフレシェ距離が $2r$ より大きい場合，同じディスクを横切ることがないので，スケッチが一致する確率を 0 にすることができる．

3.1 節ではハッシュが一致するかどうかのみで類似しているかを判定していたのに対し，この手法では，文字列編集距離を用いることでスケッチ間類似度を計算することができ，それを近似的な軌跡の類似度として扱うことができる．したがって，スケッチが完全一致しないような軌跡間においてもその類似度を調べることができる．これによって，軌跡間の距離推定やクラスタリングに応用することを可能にしている．

本論文では，ディスクに対する出入りを記録してスケッチを作成し，文字列編集距離を用いてスケッチ間類似度を計算する手法を DiscNE(Disk Number Editdistance) と呼び，ベースライン手法とする．

4 提案手法

3.2 節の手法では，文字列編集距離でスケッチ間類似度を計算していたが，文字列編集距離はスケッチの長さに影響されるという性質があるため，軌跡の長さに影響を受けないフレシェ距離との相関が強くない結果が出ている．本研究では，3.2 節のスケッチ間類似度を改善する手法として，ディスクの中心座標列に変換可能なスケッチを作成し，フレシェ距離によってスケッチ間類似度を計算することを提案する．

4.1 ディスクの中心座標列に変換可能なスケッチ

3.2 節の従来手法と同様に，軌跡の存在する領域 W に半径 r のディスクを D 枚配置する．各ディスクはディスク番号と中心座標を保持する．その後，軌跡 $T = (t_1, t_2, t_3, ..., t_m)$ の各頂点が所属するディスクを調べ，その中で頂点に最近傍なディスクをスケッチに記録する．以下で具体的な各軌跡に対するスケッチの導出方法を示す．

1. 軌跡の $k(k = 1, 2, 3, ..., m)$ 番目の頂点 t_k を含んでいるディスクを探索し，そのディスクの集合を $\mathbb{D}_{ck}$ とする．$\mathbb{D}_{ck}$ に含まれるディスクの内，中心座標が k 番目の頂点 t_k と最も近いディスクを D_k とする．
2. 軌跡の $k-1$ 番目の頂点 t_{k-1} における D_{k-1} と D_k を比較して，$D_{k-1} \neq D_k$ であれば，スケッチに D_k を追加する．
3. 手順 2 と手順 3 を $k = 1$ から $k = m$ まで変化させて，スケッチを作成する．

上記の手順により得られたスケッチは，DiskNE 同様ディスク番号列となる．しかし，ディスクが中心座標を保持していることから，ディスク番号列からディスクの中心座標列に変換可能になる．位置情報を使用できるスケッチであるため，高精度なスケッチ間類似度が望める．

また，従来手法の DiskNE では，ディスクに入るときと出るときの 2 回をスケッチに追加していたため，スケッチ長が長くなってしまう問題点もあったが，この手法では頂点を含むディスクのうち，中心座標が最も近いものだけがスケッチに追加される．そのため，1 つの頂点につき最大で 1 つしかスケッチ長が増加しない．したがって，ディスクの配置や軌跡の長さに影響を受けるこ

となく，必ず元の軌跡の長さより短いスケッチを作り出すことが出来る．これにより計算時間の短縮も望める．

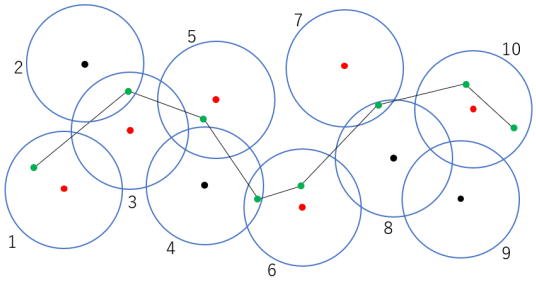

図 1　ディスクの中心座標を用いたスケッチ作成の例

図 1 の例では，従来手法の DiskNE でスケッチ作成した場合，$S = 1, 1, 2, 3, 2, 3, 5, 4, 5, 6, 4, 6, 7, 8, 7, 8, 10$ となるが，提案手法でスケッチを作成した場合，$S = 1, 3, 5, 6, 7, 10$ となり，かなりスケッチを短縮することができると分かる．提案手法におけるスケッチでは，軌跡の頂点が赤色で示されたディスクの中心座標列に変換できる．

4.2　フレシェ距離を用いたスケッチ間類似度

DiskNE のスケッチはディスク番号列であったため，スケッチ間類似度の計算には，ハミング距離や文字列編集距離を使わざるを得なかった．しかし，4.1 節のスケッチでは，2 次元の位置情報を含むディスクの中心座標列に変換できる．そこで，スケッチをディスクの中心座標列に変換し，フレシェ距離でスケッチ間類似度を計算することを提案する．スケッチにしてからスケッチ間類似度のフレシェ距離を計算することで，元の軌跡のフレシェ距離を計算するより計算量がかなり削減される．当然だが元の軌跡のフレシェ距離との相関も高くなることが望める．

本論文では，ディスクの中心座標に変換可能なスケッチを算出し，フレシェ距離を用いてスケッチ間類似度を調べる提案手法のアルゴリズムを DiskCF(Disk Center Fréchet) と呼ぶことにする．

4.3　DiskCF の誤差範囲

DiskCF を用いてスケッチ間類似度を計算した場合，元の軌跡の各頂点は半径 r のディスクの中心座標列に変換可能なスケッチに置き換わっている．スケッチ間類似度の誤差範囲は，以下のように理論的に示すことが出来る．

Theorem 1. ランダムに配置された半径 r のディスクを用いて作成された軌跡 A, B に対するスケッチを S_A, S_B とする．ディスク数 D と半径 r が十分に大きく，A, B の各頂点は少なくとも 1 つのディスクに属するとき，フレシェ距離を用いた 2 つのスケッチ間類似度 $F(S_A, S_B)$ について，式 (4) が成り立つ．

$$F(A, B) - 2r \leq F(S_A, S_B) \leq F(A, B) + 2r \qquad (4)$$

この不等式は，元の軌跡のフレシェ距離とスケッチ間のフレシェ距離の誤差が，半径の 2 倍以内であることを示している．

5　実験評価

従来手法である DiskNE と提案した DiskCF について，スケッチ間類似度の算出を比較する評価実験を行った．実験を行った環境は，Intel(R) Core(TM) i7-12700, CPU 2.10GHz, メモリ 16GB, Ubuntu 20.04.4 である．

5.1　データセット

実験は 2 つの実データを用いて行った．

1 つ目は，3.2 節の手法を提案した Maria らの論文 [4] で使われていたローマのタクシーの GPS データを用いた [3]．1 か月間の 320 台のタクシーの軌跡データである．頂点は約 7 秒ごとに記録した位置情報である．このデータの中から，特定の南北 2km，東西 7km の領域内に存在する軌跡を取り出した後，長さが 5km〜8km になるように切った軌跡 1000 個をデータ，30 個をクエリとした．各軌跡の頂点数の平均値は 88.117 で中央値は 75 である．

2 つ目は，北京の 182 人の 5 年間の人流データを用いた．頂点は 1〜5 秒ごとか 5〜10m ごとに記録された位置情報を示す．このデータの中から，特定の南北 10km，東西 5km の領域内に存在する軌跡データを取り出した後，長さが 2km〜8km になるように切った軌跡 1000 個をデータ，30 個をクエリとした．各軌跡の頂点数の平均値は 268.812 で中央値は 217 である．

どちらのデータもクエリはデータに含まれていないものを選んでおり，データとクエリに被りはない．

5.2　相関の比較

DiskNE と DiskCF について，元の軌跡間距離であるフレシェ距離との相関を調べた．縦軸にフレシェ距離，横軸をスケッチ間類似度として，クエリ 30 個に対する

各軌跡の値をプロットして散布図にするとともに相関係数も計算した．すべての頂点が少なくとも 1 つのディスクに含まれるように半径 r とディスク枚数 D を設定し，ローマのタクシーのデータセットでは，$r = 1, D = 50$，北京の人流データでは $r = 1, D = 100$ を採用した．

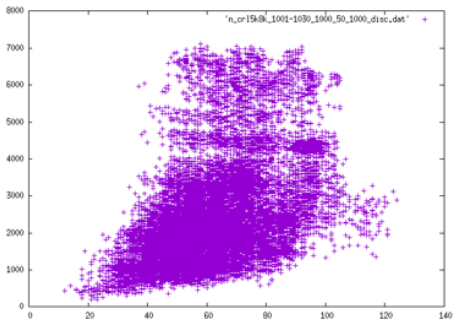

図 2　ローマデータにおける DiskNE($r = 1, D = 50$)

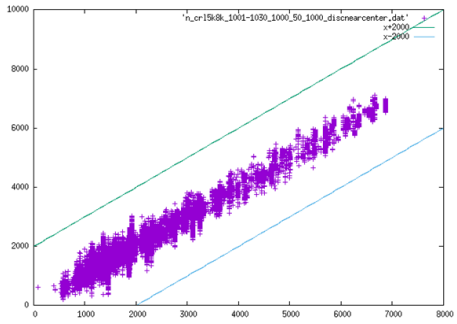

図 3　ローマデータにおける DiskCF($r = 1, D = 50$)

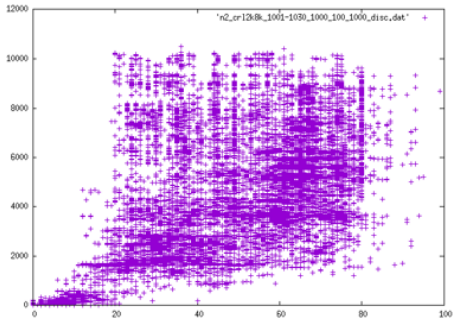

図 4　北京データにおける DiskNE($r = 1, D = 100$)

図 2 と図 3，図 4 と図 5 を比較すると，どちらも DiskNE が弱い正の相関であるのに対して，DiskCF はかなり強い正の相関になっていることが分かる．表 1 においても両データセットにおける DiskCF の相関係数が非常に高く，フレシェ距離に対する高精度なスケッチを作成できたと言える．

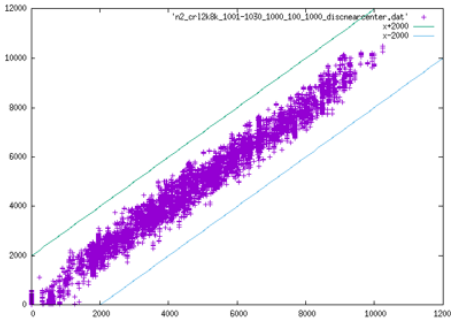

図 5　北京データにおける DiskCF($r = 1, D = 100$)

表 1　各手法の相関係数の比較

手法	ローマデータ ($r = 1, D = 50$)	北京データ ($r = 1, D = 100$)
DiskNE	0.4839	0.5010
DiskCF	0.9723	0.9892

また，DiskCF における 2 本の直線は式 (4) による誤差の範囲を示す．データが定理通り誤差の範囲内に収まっていることが分かる．

5.3　計算時間の比較

DiskNE と DiskCF の実行時間の比較を行った．半径を $r = 1$ に固定し，ディスク枚数 D を変化させてスケッチ長の変化を調べた．また，スケッチ長の変化が計算時間の変化につながっているかを確認するため，スケッチ間類似度の計算時間を調べた．変化させるディスク枚数 D は，いずれのデータセットの場合も，すべての頂点が少なくとも 1 枚のディスクに含まれるような値を用いている．スケッチの作成時間，スケッチ間類似度の計算時間は 10 回計測を行い，その平均を値とした．

表 2　ローマデータのスケッチ長

D	DiskNE		DiskCF	
	平均値	中央値	平均値	中央値
30	44.1	43	12.1	12
50	67.4	66	13.8	13
100	136.8	135	19.3	19

表 2，3 から DiskNE から DiskCF に変えることでスケッチ長を約 5 分の 1 以下にするという大幅な短縮化に成功していることが分かる．DiskNE のスケッチ長は，

表 3　北京データのスケッチ長

D	DiskNE		DiskCF	
	平均値	中央値	平均値	中央値
100	40.2	35	8.4	7
200	69.4	55	10.1	7
500	165.6	124	17.3	13

ディスク枚数 D が増えるにつれて，大幅に長くなっているのに対し，DiskCF のスケッチ長は，ディスク枚数 D が増えても，小さな変化にとどまっている．このことから，D の枚数が多いほど，DiskCF が有用な手法であると言える．

5.1 節で示した通り，ローマデータより北京データの方が元の軌跡の長さが長いが，軌跡の存在する範囲が広いため通過するディスクの数が少なく，スケッチに変換した際の長さの短縮率が大きいと考えられる．

表 4　ローマデータのスケッチ間類似度の計算時間 (ms)

D	DiskNE	DiskCF
30	1452	277
50	3120	344
100	12481	662

表 5　北京データのスケッチ間類似度の計算時間 (ms)

D	DiskNE	DiskCF
100	1230	148
200	3776	212
500	21276	601

表 4，5 の計算時間の変化をみると，スケッチ長を短縮したためスケッチ間類似度の計算時間を約 5 分の 1 以下に短縮できていることが分かる．また，スケッチ長と計算時間の関係を調べると，スケッチ長を L，スケッチ間類似度の計算時間を T_{cal} として，DiskNE も DiskCF もおおよそ $T_{cal} = cL^2$(c は各手法とデータセットによる定数) の形が成り立っていることが分かる．このことから，文字列編集距離とフレシェ距離の計算量が $\mathcal{O}(L^2)$ であることが確認でき，スケッチ長がスケッチ間類似度の計算時間に影響を与えていることが分かる．

6　結論

LSH を用いた類似軌跡探索において，ディスクの中心座標列に変換可能なスケッチを作成し，フレシェ距離によってスケッチ間類似度を算出する DiskCF を提案した．実験では実データを用いて，DiskCF がベースラインとなる Maria らの手法より高精度なスケッチ間類似度を高速に算出できることを示した．

今後の課題としては，層状のデータ構造をもち，フレシェ距離の計算やクラスタリングを高速に行える Multi Resolution Trajectory Sketch(MRTS) への DiskCF の導入，フレシェ距離と同じ軌跡間距離である DTW とスケッチ間類似度の相関の向上などが挙げられる．

謝辞

本研究は JSPS 科研費 JP21K11901 の助成を受けたものである．

参考文献

[1] D. Anne and S. Francesco. Locality-sensitive hashing of curves. In *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77, pages 31:1–37:16, 2017.

[2] M. P. Jeff and T. Pingfan. Simple distances for trajectories via landmarks. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 468–471, 2019.

[3] B. Lorenzo, B. Marco, L. Pierpaolo, B. Giuseppe, A. Raul, and R. Antonello. Crawdad dataset roma/taxi (v. 2014-07-17), 2014. `https://ieee-dataport.org/open-access/crawdad-romataxi`.

[4] A. Maria, C. Paul, K. Panagiota, G. Mayank, and S. Rik. Multi-resolution sketches and locality sensitive hashing for fast trajectory processing. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 279–288, 2018.