# 高次元データに対する近似最近傍探索手法とストリームデータに対する $k$-匿名化への応用

山吉　勇輝†　　若林　真一††　　上土井　陽子††

† 広島市立大学情報科学部　〒 731–3194 広島市安佐南区大塚東 3-4-1
†† 広島市立大学大学院情報科学研究科　〒 731–3194 広島市安佐南区大塚東 3-4-1
E-mail: †yuki55145514@gmail.com, ††{wakaba,yoko}@hiroshima-cu.ac.jp

**あらまし**　本稿では，多次元データに対するハッシングに基づく近似最近傍探索手法の既存手法である Flexible Distance-based Hashing (FDH) を拡張し，Flexible Distance-based Hashing with Quadrisection (FDHQ) を提案する．FDH が探索領域を定義する各アンカーに対してデータ空間を 2 分割するのに対し，FDHQ はデータ空間を 4 分割することで探索対象データの絞り込みと部分空間ごとのデータ数の均等化を実現する．さらに，FDHQ の応用として高次元ストリームデータに対する FDHQ に基づく $k$-匿名化手法を提案する．FDHQ と提案ストリームデータ $k$-匿名化手法をプログラムとして実現し，計算機実験により評価する．

**キーワード**　FDH, FDHQ, $k$-匿名化, ストリームデータ

## 1　はじめに

インターネット，クラウドコンピューティングなどの ICT 技術の著しい発達により，様々な形態のデータが大量に扱われるようになった [8]．こうしたデータはビッグデータと呼ばれる．ビッグデータの解析により新たな有用な情報や知見を得られる可能性がある．一方，ビッグデータには多くの個人情報が含まれる場合が多いため，ビッグデータの解析を目的にデータを第三者に公開する場合, 個人が特定されないようにデータを加工した上で公開する必要がある．こうしたデータの加工は匿名化と呼ばれる．データを匿名化する際に生じる情報量の誤差を情報損失と呼ぶ. 情報損失をできるだけ少なくすることで，より有用な解析結果を得ることができるが，一方で十分な匿名化が行われないままデータを公開すると，個人情報が漏洩する可能性が増大する．このため，個人情報を含むデータの公開においては, 個人情報保護と情報損失のバランスが重要であり，これまでに様々な個人情報保護を目的としたデータ変換手法が提案されている [2], [7], [9], [11], [12].

個人情報保護のための代表的なデータ変換手法として $k$-匿名化が知られている [8]．$k$-匿名化とは，個人情報を含むデータの集合を，データの属性値が同じである個人が少なくとも $k$ 人以上いるようにデータを変換する手法である．$k$-匿名化手法はデータ変換モデルによっていくつかの手法に分類することができる．代表的な $k$- 匿名化手法としては，非数値属性に対する一般化階層木に基づく手法や，数値属性に対する Microaggregation に基づく手法が知られている [2].

一方，近年では SNS やオンラインショッピングの普及に伴い，連続的に流れ続けるビッグデータに対する高速データ解析が求められるようになってきた．このようなデータはストリームデータと呼ばれる．近年，個人情報を含むストリームデータに対する高速 $k$-匿名化も注目されており，非数値属性に対する一般化階層木に基づくストリームデータ $k$-匿名化手法が提案されているが [1], [6], [9], [11], [12], [14]，数値属性を持つストリームデータに対する $k$-匿名化手法は著者らが調べた限りでは，著者らが以前に発表した手法 [3] 以外はほとんど知られていない.

数値属性を持つストリームデータに対する $k$-匿名化の応用としては, 個人の移動履歴データの集合に対する匿名化がある [10]．移動履歴は位置情報の時系列と考えることができ, $\{(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n)\}$ で表すことができる．ただし, $x_a$, $y_a$ は時刻 $t_a$ におけるある個人の地理的な位置を表す 2 次元座標である．この時系列データは $3n$ 次元の数値属性を持つデータとみなすことができる．多数の個人の移動履歴がオンラインでストリームデータとして入力され, $k$-匿名化処理されて出力される場合, この匿名化処理は数値属性を持つストリームデータの $k$-匿名化として定式化できる.

そこで，本研究ではストリームデータに対する Microaggregation に基づく $k$-匿名化手法を提案する．まず, 高次元データ空間における近似最近傍探索の既存手法である FDH (Flexible Distance-based Hashing) [4], [13] を拡張した FDHQ (Flexible Distance-based Hashing with Quadrisection) を提案する．FDH が探索領域を定義する各アンカーに対してデータ空間を 2 分割するのに対し，FDHQ はデータ空間を 4 分割することで探索対象データの絞り込みと部分空間ごとのデータ数の均等化を実現する．また，FDHQ のアンカー集合を複数にすることで計算時間と正答率のトレードオフの柔軟な制御を可能にする手法を提案する．次に, FDHQ を用いてストリームデータの $k$-匿名化を実現する．提案手法を C 言語を用いてプログラムとして実装し, 計算機実験をすることで, 提案手法の有効性を示す.

本論文の構成は以下のとおりである．2 では準備として, ストリームデータに対する $k$-匿名化問題を定義し, FDH を説明す

る．3 では FDH を拡張した近似最近傍探索手法 FDHQ と，アンカー集合を複数に拡張した MA-FDHQ を提案する．4 では MA-FDHQ に基づく新しいストリームデータ $k$-匿名化手法を提案する．5 では提案手法に対する実験結果に基づいて提案手法の有効性を示す．最後に, 6 で 本論文をまとめる．

## 2 準　　備

### 2.1 データ解析における個人情報保護 [8]

個人情報を含むデータの解析を目的として, 個人に関するデータがデータ所有者から第三者に提供される場合, 提供された各個人情報データがどの個人に関する情報であるのかを特定されないように, 適切にデータを変換しておく必要がある．このようなデータ変換を匿名化という．本研究では個人情報データに含まれる属性情報を以下の 3 つに種別する．

**直接識別情報** マイナンバーや運転免許証番号など, そのデータ単体で直接に個人の特定を可能にする情報．匿名化を行う際には, 削除または暗号化される．

**間接識別情報** 年齢や性別, 生年月日など, 個人に関する不変な情報であり, そのデータ単体では個人を識別可能な状態にはしないが, 複数のデータを組み合わせることで個人を識別可能にする情報．

**機密情報** 個人の病歴，個人の購買行動や Web 検索の記録など, 個人情報や個人の活動にかかわる情報．データ解析の対象となるため，匿名化処理の対象としない．

### 2.2 Microaggregation による $k$-匿名化

Microaggregation [2] による $k$-匿名化とは, 各属性が連続数値を持つデータの匿名化に利用されるデータ変換方法の 1 つである．元のデータのタプル集合を分割し, 類似した値の組み合わせを持つグループに集約する．この時, すべてのグループは $k$ 個以上から $2k-1$ 個以下のタプルを保持する．分けられたグループ内のすべてのタプルの値を, 各タプルを数値ベクトルと見なした時の重心で置き換える．最後に直接識別情報を削除することで $k$-匿名化の結果が得られる．

Microaggregation によるデータ変換に伴って発生する情報損失を SSE(Sum of Squared Euclidean distances) で定義する．以下に定義を示す．

$$SSE = \Sigma_{k=1}^{g} \Sigma_{i \in G_k} \Sigma_{j=1}^{s} (X_{i_j} - \overline{X_{k_j}})^2$$

ここで, $g$ は匿名化するときに出来たグループの数, $G_k$ は $k$ 番目のグループのタプルインデックスの集合, $s$ は間接識別情報の数, $X_{i_j}$ は $i$ 番目のタプルの $j$ 番目の間接識別情報, $\overline{X_{k_j}}$ は $k$ 番目のグループ $G_k$ の平均タプルの $j$ 番目の間接識別情報である．

### 2.3 $l$-多 様 性

$k$-匿名化の問題点として，$k$-匿名化しても機密情報が特定される可能性が残ることがある．例えば，3-匿名化が実現されていたとしても，同じ間接識別情報を持つように加工されたデータの機密情報がすべて同じ場合，機密情報は特定されてしまう．この問題を解決する匿名化手法として，$l$-多様性がある．$l$-多様性とは，$k$-匿名化後の各グループの機密情報が $l(1 < l \leqq k)$ 種類以上になることを保証するものである．$l$ の値が増加するほど機密情報が何であるかを特定することが困難になる．

### 2.4 ストリームデータに対する $k$-匿名化問題

ストリームデータとは, 無限に生成され続けるデータ (タプル) の系列であり, 本研究ではストリームデータを構成する各データは複数の属性を持つものとし, 各属性は実数値とし, すべての属性を $k$-匿名化の対象とする．属性数（数値データの次元数）を $d$ とする．ストリームデータに対する $k$-匿名化問題は以下のように定式化される．

**入力:** 無限に生成され続けるデータ系列 $SD =< T_1, T_2, \ldots >$, 各 $T_i$ は $d$ 個の実数値属性を持つ．$T_1, T_2, \ldots$ の順に, 逐次入力される ．正の整定数 $k$, $W$ ($W$ をウィンドウサイズという).

**出力:** 条件を満たし, 目的関数を最小とするデータ系列 $AD =< U_1, U_2, \ldots >$, 各 $U_i$ は $T_i$ の匿名化データ．$U_1, U_2, \ldots$ の順に, 逐次出力される ．

**条件:** 各 $U_i$ に対し, $U_1, U_2, \ldots, U_i, \ldots, U_{i+W-1}$ の中に, すべての属性値が $U_i$ と同じ匿名化データが $k$ 個以上ある ($k$-匿名化の条件).

**目的関数:** $T_i$ と $U_i$ のユークリッド距離の 2 乗 ( 情報損失 ) を $IL(T_i, U_i)$ とするとき, 最新の出力タプル $U_x$ に対し, $\Sigma_{i=1}^{x} IL(T_i, U_i)/x$, すなわち, 匿名化データの平均情報損失を目的関数とする．

2.3 で示した $l$-多様性の保証も求められる場合，上記において, 同じ匿名化データに加工されたデータの機密情報が $l$ 種類以上あることが条件として追加される．

### 2.5 Flexible Distance-based Hashing(FDH) [13]

FDH は大規模多次元データの近傍探索のために開発されたハッシング手法である．与えられたデータ集合から $A$ 個のアンカー $a_i (1 \leqq i \leqq A)$ を決め, $a_i$ に対する任意に決められた半径 $r_i$ によって空間を分割し, 分割された部分空間 ( 領域 ) ごとにデータを管理することで, 距離が近いデータを領域ごとに管理できるようになる．FDH に基づく近傍探索の対象となるデータ集合を $D = \{p_1, p_2, \ldots, p_n\}$ とする．ただし, 各 $p_i$ は $d$ 次元ベクトルとし, データ集合 $D$ を $d$ 次元空間に分布するデータ点の集合とみなす．$D$ に対して, $A$ 個のアンカー $a_i$ を定め, アンカー集合を $A_n$ で表す．アンカーは $d$ 次元空間 ($d$ はデータの属性数 ) においてできるだけ分散するように決定する．詳細は紙面の都合上, 省略する．

各アンカー $a_i$ に対して距離 $r_i$ を定め, 半径と呼ぶ．アンカー $a_i$ に対する半径 $r_i$ は以下のように決定する．各 $a_i$ に対し, $r_i$ によって定義される $d$ 次元空間中の超球 ($d = 2$ の場合は円 ) の内と外に存在するデータ点が半々になるように, 半径 $r_i$ を与える．具体的には, データ集合 $D$ のデータ数を $n$ とするとき, $n$ が奇数の場合は, $D$ の要素を $a_i$ から近い順に並べた時の $\lceil n/2 \rceil$ 番目のデータ点に対する $a_i$ からのユークリッド距離を $r_i$ とする．$n$ が偶数の場合は, $D$ の要素を $a_i$ から近い順に

並べた時の $n/2$ 番目と $n/2+1$ 番目のデータ点に対する $a_i$ からのユークリッド距離の平均を $r_i$ とする．アンカー $a_i$ と $d$ 次元空間の任意のデータ点 $p$ に対し，$a_i$ と $p$ の間のユークリッド距離を $dist(a_i, p)$ で表す．$d$ 次元空間のアンカー集合 $A_n$ と任意のデータ点 $p$ に対し，長さ $A$ のビット列（ビットマップ）$BM(p) = b_1, b_2, \ldots, b_A$ を以下のように定める．

$$BM(p)[i] = b_i = \begin{cases} 0 & \text{if } dist(a_i, p) \leqq r_i \\ 1 & \text{otherwise} \end{cases}$$

この時，$d$ 次元データ空間はビットマップによって $2^A$ 個の部分空間に分割される．すなわち，$d$ 次元空間の任意のデータ点 $p$ と $q$ に対し，$BM(p) = BM(q)$ であれば，$p$ と $q$ が同じ部分空間になるように $d$ 次元空間を分割する．分割された各部分空間を以下では領域という．領域は対応するビットマップによって識別される．$d = 2$, $A = 3$ の場合，図 1 に示すように，領域は 2 次元平面上の 3 個の円の交わりで表現され，8 つの領域に分割される．

与えられた $d$ 次元データ集合 $D$ を領域ごとに分けて記憶しておくことで，任意に与えられたデータ点 $p$ に対し，$D$ の要素で $p$ の近傍となるデータを高速に探索することが可能となる．この近傍データ探索手法を FDH という．例えば図 1 において，ビットマップ 10 の領域にはデータ点 $p_5$, $p_7$, $p_8$, $p_{11}$ が存在し，それらが直接，探索できるように記憶されている．データ点 $p$ が与えられたとき，$p$ は 010 の領域に属しているので，同じ領域に属する $p_5$, $p_7$, $p_8$, $p_{11}$ のみを探索対象として $p$ から各データ点へのユークリッド距離を計算すれば，$p$ の最近傍データ点である $p_8$ を効率よく見つけることができる．

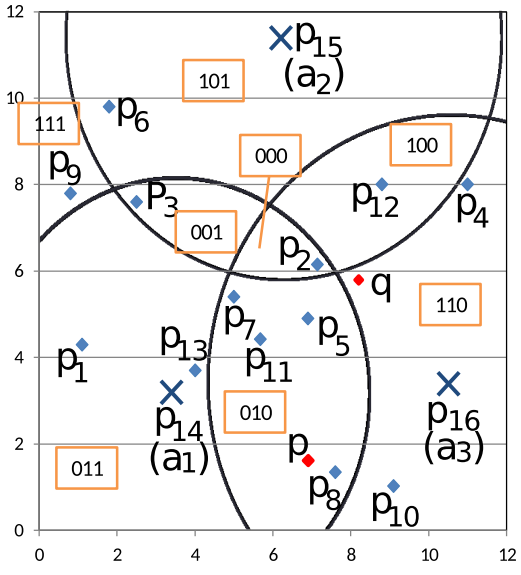


図 1　FDH の例

## 3　FDH の拡張

### 3.1　FDHQ

2.5 で述べた FDH は，5.2.1 で実験結果を示すように，領域ごとに含まれるデータ点が偏在しやすいという課題がある．この問題点を解消する手法として，本稿では FDHQ （Flexible Distance-based Hashing with Quadrisection）を提案する．FDHQ は FDH を拡張した手法であり，アンカー $a_i$ に対し，半径 $r_i$ に加えて半径 $r_i'$ と半径 $r_i''$ を取り入れる．FDHQ は FDH と同様，与えられたデータ集合 $D$={$p_1$, $p_2$, ..., $p_n$} から $A$ 個のアンカー $a_i$ を決定し，$a_i$ に対する任意に決められた半径 $r_i$ を定める．$r_i$ の定め方は FDH と同じである．さらに，FDHQ では半径 $r_i'$ と半径 $r_i''$ を定める．ただし，議論を簡単にするため，以下ではデータ数を $4n(n = 1, 2, 3, \ldots)$ 個と仮定する．まず，半径 $r_i'$ の長さは，アンカー $a_i$ に近い方から数えて $D/4$, $D/4$+1 番目のデータ点 $p_{D/4}$, $p_{D/4+1}$ の中間とアンカー間の距離とする．つぎに，半径 $r_i''$ の長さは，アンカー $a_i$ に近い方から数えて $3D/4$, $3D/4$+1 番目のデータ点 $p_{3D/4}$, $p_{3D/4+1}$ の中間とアンカー間の距離とする．

アンカー $a_i$ と $d$ 次元空間の任意のデータ点 $p$ の間のユークリッド距離を $dist(a_i, p)$ で表す．$d$ 次元空間のアンカー集合 $A_n$ と任意のデータ点 $p$ に対し，長さ $A$ の 4 値列 (“Q マップ” と呼ぶ)$QM(p) = q_1, q_2, ..., q_A$ を次のように定める．

$$QM(p)[i] = q_i = \begin{cases} 0 & \text{if } dist(a_i, p) \leqq r_i' \\ 1 & \text{if } r_i' < dist(a_i, p) \leqq r_i \\ 2 & \text{if } r_i < dist(a_i, p) \leqq r_i'' \\ 3 & \text{if } r_i'' < dist(a_i, p) \end{cases}$$

このとき，$d$ 次元データ空間は $4^A$ 個の部分空間に分割される．すなわち，任意のデータ点 $p$ とクエリ $q$ に対して $QM(p) = QM(q)$ が成り立つならば，$p$ と $q$ は同じ部分空間に属することになる．このように生成された部分空間を，FDH と同様に領域と呼ぶ．

例えば次元数 $d$=2，アンカー数 $A$=3 の場合，2 次元平面上にアンカーが 3 つ生成され，図 2 のように 3 つの真円によって $4^3$(=64) 個の領域に分けられる．また，同図には Q マップの一部を示す．FDHQ は FDH と比較してより細かく領域分けをするため，データ点をより平均的に分布させることが可能である．

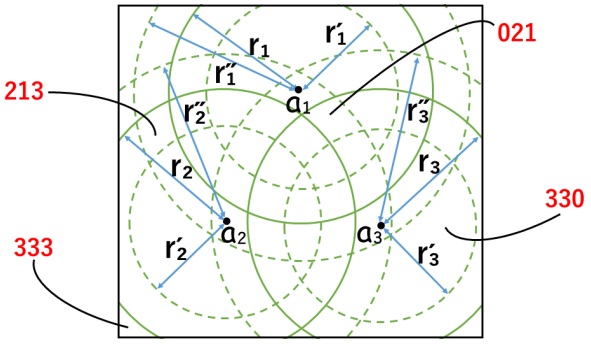


図 2　$d$=2, $A$=3 のときの領域分け (FDHQ)

### 3.2　半径 $r_i'$ と半径 $r_i''$ の位置変更

FDH から FDHQ へ拡張することによりデータ点の領域分布の偏りが少なくなったが，さらに偏りを少なくするために次の

手法を提案する．前節で示した FDHQ は，各アンカー $a_i$ に対してデータ点の数が 1:1:1:1 になるようにデータ空間を分割している．この提案手法ではその分割割合を等分から変更することで，領域に属するデータ点が偏在する可能性を減らす．5 の実験では 25:25:25:25(等分) のほかに，24:26:26:24，23:27:27:23，22:28:28:22，21:29:29:21，20:30:30:20 の計 6 種類を使用している．

### 3.3 マルチアンカー集合 FDHQ

本稿では FDHQ のさらなる拡張して, マルチアンカー集合 FDHQ ( Multi-Anchor Set FDHQ, MA-FDHQ ) を提案する．FDH に対し, 著者らは [5] において, 通常の FDH が 1 個のアンカー集合に基づいて多次元空間を複数の領域に分割し, 与えられたクエリに近い領域を探索領域として, 最近傍探索を行うのに対し, 複数のアンカー集合を用意し, アンカー集合ごとに多次元空間を領域に分割し, 与えられたクエリに対して, アンカー集合ごとのそれぞれの近傍領域において, 最近傍探索を行うことで,1 つのアンカー集合を用いる場合よりもよりよい近似解を得る MA-FDH を提案した．MA-FDHQ は，MA-FDH における FDH の代わりに FDHQ を用いる手法である．

MA-FDHQ のアンカー集合の集合を $MA = \{A_1, A_2, \ldots, A_s\}$ とする．ただし, 各 $A_i$ は $A$ 個のアンカーから構成されるアンカー集合である．MA-FDHQ の近傍解探索では, 各アンカー集合 $A_i$ $(1 \leqq i \leqq s)$ に対して FDHQ を実行し, 得られた近傍解の中で最も与えられたクエリに近い近傍解を MA-FDHQ の近傍解として出力する．MA-FDHQ は FDHQ を $s$ 回実行して, 得られた近似解の中でクエリに最も近いデータを出力することと等価なので, MA-FDHQ の近似解が FDHQ と同じか優れた解が得られることは自明であるが, 一方で, 計算時間は MA-FDHQ のほうが増加する．計算時間の短縮のため, MA-FDHQ ではデータ集合の各データにフラグを付加する．クエリに対して 1 度距離計算を行ったデータについては, フラグをクエリの識別番号に設定することで, 同じクエリに対しては距離計算を 1 度だけ行うことで計算時間の短縮を実現する．

## 4 ストリームデータに対する $k$-匿名化

### 4.1 $k$-匿名化手法の概要

著者らは [3] において，高次元ストリームデータに対する FDH に基づく $k$-匿名化手法を提案した．本稿では，この手法を FDHQ に基づく手法に変更するとともに，$l$-多様性も考慮できるように拡張した．

本稿で提案する高次元ストリームデータに対する $k$-匿名化の流れを図 3 に示す．外部から入力されるストリームデータは 1 タプルずつ入力バッファに格納される．入力バッファのサイズ ( バッファに格納されるタプル数 ) は $k$-匿名化問題の入力として与えられるウィンドウサイズ $W$ である．入力バッファに入ってきたタイミングで各タプルに 1 から始まり, 2, 3, 4 と増大する自然数をタイムスタンプとして付与する．入力バッファは FIFO(First-In First-Out) で管理されており, タプルが 1 つ処理され出力されるたびに, 新たに 1 つのタプルが入力されるものと仮定する．キャッシュは過去に匿名化されたデータを保持しておくものであり, FIFO で管理される．キャッシュでは, キャッシュ内の各匿名化データごとに, いくつのデータを匿名化した結果なのかを示すデータ数と情報損失も格納されている．

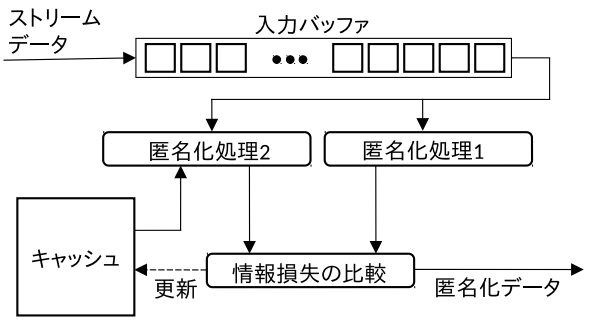


図 3　提案アルゴリズムの概要

入力バッファとキャッシュのデータは FDH あるいは FDHQ で管理されている．入力バッファについては外部から入力されたストリームデータ $q$ に対し，ビットマップ $BM(q)$ あるいは Q マップ $QM(q)$ が計算され, それらをインデックスとする領域ごとにデータはリスト構造で管理されている．キャッシュについては, 新しく匿名化データが作成されるごとに, そのデータを $q$ とすると，ビットマップ $BM(q)$ あるいは Q マップ $QM(q)$ が計算され, それらをインデックスとする領域ごとにデータはリスト構造で管理されている．入力バッファあるいはキャッシュに対する最近傍探索においては，MA-FDH あるいは MA-FDHQ を利用する．

入力バッファがデータで満たされたら匿名化処理が開始され, 入力バッファの先頭データを匿名化の処理対象として匿名化処理 1 と匿名化処理 2 を行う．匿名化処理 1 では, 入力バッファに記憶されているデータに対して先頭データの匿名化処理を行う．匿名化処理 2 ではキャッシュに格納されている, すでに匿名化され, 出力された匿名化データに対して入力バッファの先頭データの匿名化処理を行う．得られた 2 つの匿名化の結果の中で情報損失が小さい方の結果が採用され, 先頭タプルの匿名化データとして出力される．匿名化処理 1 の結果が出力された場合はキャッシュに格納されている匿名化データが更新される．匿名化処理 1 と匿名化処理 2, および匿名化データ出力処理については 4.2, 4.3, 4.4 で説明する．

### 4.2 FDHQ を適用した匿名化処理 1

(1) 入力バッファの先頭タプル（$q$ とする）を選択する．$q$ のフラグが 0 でなければ，過去に匿名化の対象となったタプルなので，処理を終了する（匿名化処理 2 も実行しない）．
(2) $q$ に対し，入力バッファ内のフラグが 0 であり, $q$ に対して最近傍データの候補となるタプルを $2pk$ 個以上，MA-FDHQ を用いて探索し，先頭タプル $q$ との $q$-匿名化の対象候補とする．ここで，$p$ はあらかじめ与えられる正整数とする．
(3) (2) で選択した各タプルを $q$ との距離の非減少順に $2k-1$

個を対象に，タプル 1 つあたりの情報損失が最小となる $k'$ $(1 \leqq k' < 2k)$ 個のタプルを選択して $k$-匿名化の候補として採用し，タプル 1 個あたりの情報損失を匿名化処理 1 の結果 $IS_1$ として出力する．選択された $k'$ 個のタプルは一時バッファに入れて記憶しておく．$l$-多様性の保証も求められる場合，$k'$ 個のタプルの決定において，同じ匿名化データに加工されたデータの機密情報が $l$ 種類以上あることが条件として追加される．

### 4.3 FDHQ を適用した匿名化処理 2

(1) 匿名化したデータ数が $2k-1$ 以下であることを条件とし，入力バッファの先頭タプル $q$ との距離が最小となるキャッシュデータを MA-FDHQ を用いて求める．
(2) (1) で求めたキャッシュデータに $q$ を加えたときの情報損失を匿名化処理 2 の結果 $IS_2$ として出力する．

### 4.4 匿名化データの出力処理

入力バッファの先頭タプル $q$ のフラグが 1 の場合は過去にすでに匿名化されたデータなので，タプルをそのまま出力する．フラグが 0 の場合は以下の処理を行う．

$IS_1 \leqq IS_2$ ならば匿名化処理 1 の結果が採用される．匿名化処理 1 で得られた匿名化データを出力する．さらに，このデータをキャッシュデータとして新規にキャッシュに登録する．匿名化処理 1 で匿名化の対象として選択され一時バッファに記憶されている入力バッファデータを匿名化データに書き換え，それぞれのデータのフラグを 1 とする．

$IS_1 > IS_2$ ならば匿名化処理 2 の結果が採用される．キャッシュから選択された匿名化データを入力バッファの先頭データ $q$ の匿名化データとして出力すると共に，キャッシュ内の対応する匿名化データのデータ数を 1 増やし，情報損失を更新する．入力バッファの先頭タプルは削除する．

## 5 実験と評価

本節では，まず，領域ごとのデータ分布に関する FDH と FDHQ の比較を行う．次に，高次元のストリームデータに対する $k$-匿名化手法の解法として全探索，FDH および FDHQ を用いた場合の実験結果の比較と考察を行う．

### 5.1 実 験 環 境

提案手法をプログラム実装し，CPU:Intel Core i7 4.0GHz, OS:CentOS Linux 7，主記憶 64GB の PC を使用し，コンパイラ:gcc version 4.8.5，最適化オプション-O2 でコンパイルして実行した．なお，実装したストリームデータに対する $k$-匿名化プログラムにおいては，2.3 で説明した $l$-多様性については考慮していない．

### 5.2 実験結果と考察

#### 5.2.1 FDH と FDHQ のデータ点の領域分布の偏りの比較

FDH と FDHQ のデータ点の領域分布の偏りの比較を行う．条件を揃えるために，互いに生成される領域数が同じになるようにアンカー数を設定している．具体的には，領域数を 4,096 としたため FDH のアンカー数は 12，FDHQ のアンカー数は 6 としている．図 4 にデータ数 $n$=1,000,000，次元数 $d$=32 のデータ集合 $D$ を用いた場合について，FDH と FDHQ の双方の結果を示している．

まず FDH は，多くの領域に平均値付近のデータ数が入っているが，最大値が 6235 と，平均値の約 26 倍ものデータ点が入っている領域がある．一方で FDHQ は，同じく多くの領域に平均値付近のデータ数が入っていて，最大値も 1450 と平均値の 6 倍程度に収まっている．以上のことから，FDH と比較すると FDHQ の方が平均的な計算時間を実現できるといえる．

このような違いが現れた理由は，アンカー数の差だと考えられる．この節の実験で用いたアンカー数は，FDH が 12，FDHQ が 6 である．アンカー数が少ないほど，密集したデータ点が上手く区切れるのではないかと推測する．

#### 5.2.2 FDHQ の分割割合によるデータ点の領域分布の比較

FDHQ のデータ空間の分割割合の変更による領域ごとのデータ数分布の比較を行う．図 5 にデータ数 $n$=1,000,000，次元数 $d$=32 のデータ集合 $D$ を用いた場合について，25:25:25:25(等分)，24:26:26:24，23:27:27:23，22:28:28:22，21:29:29:21，20:30:30:20 の 6 通りを示している．いずれの分割割合も最大値付近以外では同程度の分布となっている．そこで，図 6 はそれぞれの分割割合の上位 20 データを取り上げている．また，表 1 には各分割割合の最大値を記載している．最大値が最小である分割割合は 21:29:29:21 となっていて，801 である．これは平均値の約 3 倍であり，FDH や FDHQ の等分と比較すると非常に優れているといえる．

このような違いが現れた理由は，領域分けによるデータ点分布の特徴にあると考えられる．分割割合が等分のときよりも，21:29:29:21 の方が均等になるということは，全アンカーに対して最も内側の領域か，最も外側の領域に入るデータ点が多いのではないかと考えられる．

表 1　FDHQ の分割割合によるデータ数分布の変化

| 分割割合 | 領域内のデータ数の最大値 |
|---|---|
| 25:25:25:25 | 1450 |
| 24:26:26:24 | 1241 |
| 23:27:27:23 | 1243 |
| 22:28:28:22 | 883 |
| 21:29:29:21 | 801 |
| 20:30:30:20 | 956 |

### 5.3 ストリーミングデータに対する $k$-匿名化手法における情報損失および計算時間の比較

$k$-匿名化に用いる近傍探索手法として，全探索と近似最近傍探索である MA-FDH および MA-FDHQ を用いたストリーミングデータに対する $k$-匿名化の比較を行う．評価基準は情報損失と計算時間で，この 2 つの基準にはトレードオフの関係

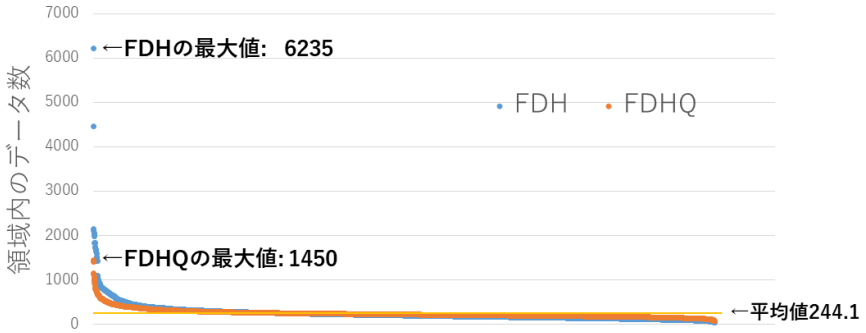


図 4　4096 領域に空間分割したときの FDH および FDHQ のデータ数分布 (降順)

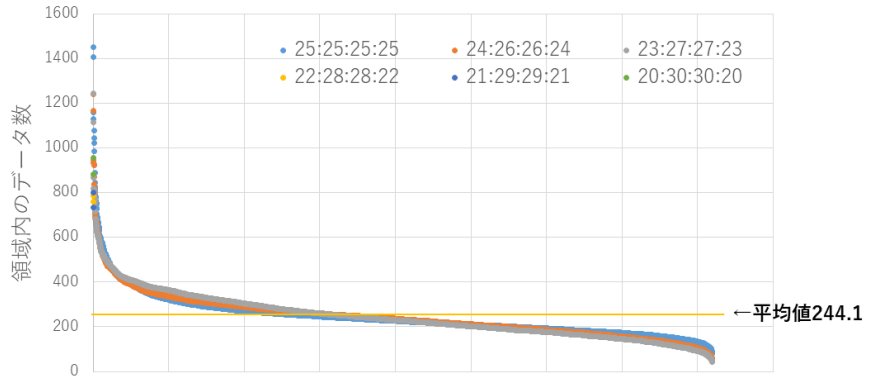


図 5　分割割合を変更した場合のデータ数の分布 (降順)

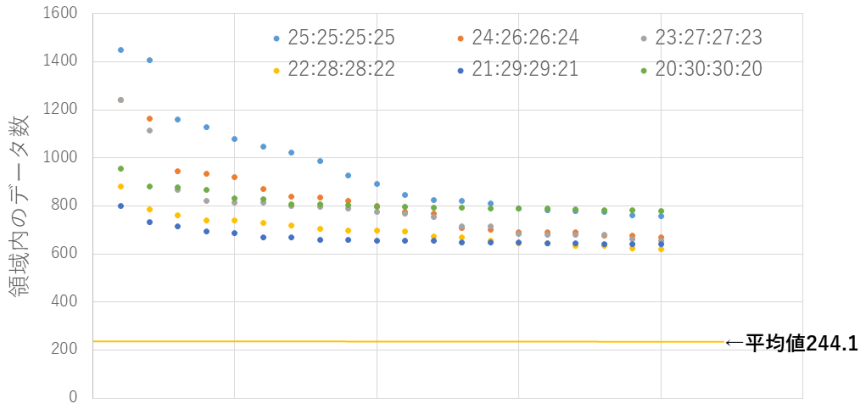


図 6　図 5 のうち上位 20 データを抜き出したもの

がある．比較する手法は全 4 種類で，全領域を探索する全探索，MA-FDH と MA-FDHQ，そして MA-FDHQ の分割割合を 22:28:28:22 に変更した MA-FDHQ′ である．

入力バッファのサイズ $W$ を 10,000，最近傍データ候補のパラメータ $p$ を 2 とした．データの次元数 $d$ は 16, 32, 64 とし，データの値は-0.999 から 0.999 の範囲で一様乱数に基づいて発生させた．また，MA-FDH のアンカー数を 10，MA-FDHQ のアンカー数を 5 とし，ともにデータ空間を 1024 領域に分割するようにしている．さらに，MA-FDH と MA-FDHQ ともにアンカー集合数を 100 としている．$k$-匿名化の $k$ の値を 3, 4, 5 とした場合の実験結果について，表 2 に計算時間，表 3 に情報損失を示す．表中の FDH, FDHQ, FDHQ' はそれぞれ MA-FDH, MA-FDHQ, MA-FDHQ' の結果を示す．計算時間はクエリ 10,000 個に対する計算時間の総和，情報損失はクエリ 1 個あたりの情報損失の平均である．情報損失のカッコ内の数値は全探索に対する相対誤差を示す．

まず，計算時間について考察を行う．計算時間が最も長いのは全探索であり，FDH, FDHQ, FDHQ' の順に短くなっている．例えば 3-匿名化において，全探索手法では 2.76 秒の時間を要している．これと比較して，MA-FDH は 60.9%，MA-FDHQ は 40.9%，MA-FDHQ′ は 39.5% の計算時間となっている．この傾向は，$k = 4$, $k = 5$ の場合でも同様である．すなわち，計算時間に関して，MA-FDHQ は MA-FDH より優れているといえる．また，22:28:28:22 に分割した MA-FDHQ′ は MA-FDHQ

表 2　$W = 10,000$，$d = 16, 32, 64$ のときの計算時間 [s]

| $d = 16$ | $k = 3$ | $k = 4$ | $k = 5$ |
|---|---|---|---|
| 全探索 | 2.76 | 2.05 | 1.66 |
| FDH | 1.68 | 1.33 | 1.14 |
| FDHQ | 1.13 | 0.85 | 0.74 |
| FDHQ’ | 1.09 | 0.85 | 0.74 |
| $d = 32$ | $k = 3$ | $k = 4$ | $k = 5$ |
| 全探索 | 3.52 | 2.55 | 1.94 |
| FDH | 1.84 | 1.50 | 1.33 |
| FDHQ | 1.21 | 0.97 | 0.87 |
| FDHQ’ | 1.21 | 0.95 | 0.85 |
| $d = 64$ | $k = 3$ | $k = 4$ | $k = 5$ |
| 全探索 | 5.11 | 3.71 | 3.04 |
| FDH | 2.60 | 2.07 | 1.81 |
| FDHQ | 1.59 | 1.29 | 1.14 |
| FDHQ’ | 1.55 | 1.27 | 1.10 |

表 3　$W = 10,000$，$d = 16, 32, 64$ のときの情報損失

| $d = 16$ | $k = 3$ | $k = 4$ | $k = 5$ |
|---|---|---|---|
| 全探索 | 1.027 | 1.371 | 1.612 |
| | (0.00%) | (0.00%) | (0.00%) |
| FDH | 1.036 | 1.389 | 1.645 |
| | (0.88%) | (1.31%) | (2.05%) |
| FDHQ | 1.061 | 1.413 | 1.676 |
| | (3.31%) | (3.06%) | (3.97%) |
| FDHQ’ | 1.061 | 1.416 | 1.684 |
| | (3.31%) | (3.28%) | (4.47%) |
| $d = 32$ | $k = 3$ | $k = 4$ | $k = 5$ |
| 全探索 | 3.524 | 4.460 | 5.123 |
| | (0.00%) | (0.00%) | (0.00%) |
| FDH | 3.574 | 4.543 | 5.190 |
| | (1.42%) | (1.86%) | (1.31%) |
| FDHQ | 3.654 | 4.619 | 5.291 |
| | (3.69%) | (3.57%) | (3.28%) |
| FDHQ’ | 3.666 | 4.639 | 5.315 |
| | (4.03%) | (4.01%) | (3.75%) |
| $d = 64$ | $k = 3$ | $k = 4$ | $k = 5$ |
| 全探索 | 9.327 | 11.417 | 12.774 |
| | (0.00%) | (0.00%) | (0.00%) |
| FDH | 9.528 | 11.615 | 13.000 |
| | (2.16%) | (1.73%) | (1.77%) |
| FDHQ | 9.684 | 11.817 | 13.192 |
| | (3.83%) | (3.50%) | (3.27%) |
| FDHQ’ | 9.679 | 11.807 | 13.233 |
| | (3.77%) | (3.42%) | (3.59%) |

より同等か優れているといえる．このような傾向になるのは，5. 2. 1 と 5. 2. 2 の結果から，データ点の領域分布が平均化されるほど，多くの計算時間を必要とする領域が減少し，クエリひとつあたりの計算時間が早くなるからだと考えられる．

つぎに，情報損失について考察を行う．3-匿名化において，全探索手法では情報損失は 1.027 となっている．これと比較して，MA-FDH は 0.88%，MA-FDHQ は 3.31%，MA-FDHQ′ は同じく 3.31%の情報損失の増加となっている．この傾向は，他の $k$ の値でも同様である．すなわち，情報損失において，MA-FDHQ は MA-FDH より劣っているといえる．また，22:28:28:22 に分割した MA-FDHQ′ は MA-FDHQ より若干優れているといえる．このような傾向になるのは，計算時間と情報損失の間のトレードオフが関係していると考えられる．

表 2 で示す計算時間について，全探索を使った場合と MA-FDH 等の近似最近傍探索手法を用いた場合とで大きな計算時間の差がないのは，近似最近傍探索手法における領域探索で生じるオーバヘッドが大きいためであり，近似最近傍探索手法を用いる利点は少ないと考えられる．領域に含まれるデータ数が少なくなれば，相対的にオーバヘッドが小さくなり，全探索と比較して計算時間の短縮が大きくなることが予想される．そこで，入力バッファサイズ $W$ を 100,000 とした場合の実験を行った．MA-FDH の各アンカー数を 12，MA-FDHQ の各アンカー数を 6 とし，ともにデータ空間を 4096 領域に分割するようにした．データの次元数 $d$ を 64 とした場合のクエリ 10,000 個に対する計算時間と情報損失の実験結果を表 4，表 5 にそれぞれ示す．

表 4　$W = 100,000$，$d = 64$ のときの計算時間 [s]

| $d = 64$ | $k = 3$ | $k = 4$ | $k = 5$ |
|---|---|---|---|
| 全探索 | 187.47 | 170.77 | 154.76 |
| FDH | 14.64 | 13.13 | 11.83 |
| FDHQ | 8.87 | 7.53 | 6.74 |
| FDHQ’ | 7.73 | 7.10 | 6.63 |

表 5　$W = 100,000$，$d = 64$ のときの情報損失

| $d = 64$ | $k = 3$ | $k = 4$ | $k = 5$ |
|---|---|---|---|
| 全探索 | 7.844 | 9.560 | 10.717 |
| | (0.00%) | (0.00%) | (0.00%) |
| FDH | 8.257 | 10.058 | 11.273 |
| | (5.27%) | (5.21%) | (5.19%) |
| FDHQ | 8.479 | 10.313 | 11.557 |
| | (8.10%) | (7.88%) | (7.84%) |
| FDHQ’ | 8.495 | 10.339 | 11.580 |
| | (8.30%) | (8.15%) | (8.05%) |

この実験ではアンカー数を増やしたために領域数が表 2 の 4 倍になったため，表 4 からわかるように, 全探索と比較して，近似最近傍探索手法を用いた場合は計算時間が大幅に短縮された．例えば，$k = 3$ の場合，MA-FDH は 12.8 倍，MA-FDHQ は 21.1 倍，MA-FDHQ’ は 24.3 倍の高速化を達成している．一方，情報損失については，領域数の増加に伴って，各領域に含まれるデータ数が相対的に減少したため，表 3 の場合と比較して，相対誤差は増加している．

最後に，提案手法 MA-FDHQ および MA-FDHQ′ について総合的な評価を行う．まず，MA-FDHQ は MA-FDH と同じ領域数にも関わらず，計算時間が MA-FDH の 60~70%のため有用であるといえる．また，情報損失も全探索と比較して 2~3%程度の増加のため，実用的であると考えられる．さらに，MA-FDHQ′ は MA-FDHQ と比較して計算時間も精度も優れており，MA-FDHQ よりもさらに実用的であると考えられる．

## 6 まとめ

本稿では, 数値属性をもつストリームデータに対する個人情報保護を目的とした $k$-匿名化に対し, FDH(Flexible Distance-based Hashing) を拡張した高次元データ空間における高速近傍データ探索手法 FDHQ を導入することで, 計算時間の短縮を実現した $k$-匿名化手法を提案した.

既存手法の FDH と比較し，FDHQ はデータ点の領域分布の偏りが少なく，計算時間のばらつきも抑えられた．また，MA-FDHQ を $k$-匿名化へ応用すると，アンカー数が 6（探索領域数が 4096）の場合は全探索と比較して大幅な計算時間の短縮を実現した．MA-FDH と比較しても，情報損失は多少劣化するが，計算時間は短縮されている．

今後の課題としては, 並列処理の導入による処理時間の短縮, 数値データと非数値データを共に属性として持つストリームデータに対する $k$-匿名化への提案手法の拡張等が上げられる.

本研究は JSPS 科研費 JP17K00188 の助成を受けたものである.

### 文　献

[1] Jianneng Cao, Barbara Carminati, Elena Ferrari, Kian-Lee Tan, “ASTLE: Continuously anonymizing data streams,” IEEE Trans. Dependable and Secure Computing, Vol.8, No.3, pp.337-352, 2011.

[2] J. Domingo-Ferrer, A. Martinez-Ballests, J.M. Mateo-Sanz, F. Sebe, “Efficient multivariate data-oriented microaggregation,” The VLDB Journal, Vol.15, pp.355-369, 2006.

[3] 糸谷 友里, 若林 真一, 永山 忍, 稲木 雅人, “距離に基づくハッシングを用いた高次元ストリームデータに対する効率の良い $k$-匿名化手法”, 第 19 回 IEEE 広島支部学生シンポジウム論文集, pp.28–33, 2017.

[4] Yuri Itotani, Shin’ichi Wakabayashi, Shinobu Nagayaya, Masato Inagi, “An Approximate Nearest Neighbor Search Algorithm Using Distance-based Hashing,” Proc. 29th International Conference on Database and Expert Systems Applications (DEXA2018), pp.203–213, 2018.

[5] 糸谷 友里, 森 啓輔, 若林 真一, 永山 忍, 稲木 雅人, 上土井 陽子, “大規模多次元データ集合に対する近似最近傍探索手法の改良”, DEIM フォーラム 2019 論文集, A7-1, 2019.

[6] Jianzhong Li, Beng Chin Ooi, Weiping Wang, “Anonymizing streaming data for privacy protection,” Proc. International Conf. on Data Engineering 2008, pp.1367-1369, 2008.

[7] Esmaeil Mohammadian, Morteza Noferesti, Rasool Jalili, “Fast anonymization of big data streams,” Proc. BigData-Science ’14, 2014.

[8] 佐久間 淳, データ解析におけるプライバシー保護, 講談社, 2016.

[9] 澤田 純一, 井上 恒一, 西 宏章, “低遅延匿名化処理機構における情報損失度改善手法の提案,” 情報処理学会研究報告, Vol.2011ARC196, No.9, 2011.

[10] 高橋 勝巳, 正木 彰伍, 濱田 浩気, “個人データの匿名化とその限界”, 電子情報通信学会誌, Vol.98, No.3, pp.193–201, 2015.

[11] 山岡 裕司, 伊藤 孝一, “ストリームデータに対する遅延のない $k$-匿名化方式”, 暗号と情報セキュリティシンポジウム 2016(SCIS2016) 論文集, 2A3-3, 2016.

[12] 山岡 裕司, 伊藤 孝一, “ストリームデータに対する遅延のない $k$-匿名化の情報劣化を改善する方式”, コンピュータセキュリティシンポジウム 2016(CSS2016) 論文集, 3A2-3, 2016.

[13] Man Lung Yiu, Ira Assent, Christian S. Jensen, Panos Kalnis, “Outsourced similarity search on metric data assets,” IEEE Trans. Knowledge and Data Engineering, Vol.24, No.2, pp.338-352, 2012.

[14] Bin Zhou, Yi Han, Jian Pei, Bin Jiang, Yufei Tao, Yan Jia, “Continuous privacy preserving publishing of data streams,” Proc. International Conference on Extending Database Technology, pp.648-659, 2009.