DEIM Forum 2013 F8-3

Microaggregationを利用したk-匿名化手法の高速化

須崎　亮† 上土井陽子†† 若林　真一††

† 広島市立大学情報科学部　〒 731–3194 広島市大塚東三丁目 4-1
†† 広島市立大学大学院情報科学研究科　〒 731–3194 広島市大塚東三丁目 4-1
E-mail: †suzaki@lcl.info.hiroshima-cu.ac.jp, ††{yoko,wakaba}@hiroshima-cu.ac.jp

あらまし　Microaggregation は統計的開示制御のためのデータ変換法の 1 つである．近年，microaggregation が k-匿名性を達成するために有効であることが知られてきた．既存の研究では近傍探索と最短経路探索を用いて小規模なデータに対して効率よく良質な k-匿名性を満たすデータを出力する手法が提案されている．本研究では上記の従来手法の高速化，及び，解の質の向上を目的として，microaggregation を用いた k-匿名化手法を提案する．さらに提案手法と従来手法の計算複雑さ，及び，解の質を理論的，実験的に比較する．

キーワード　microaggregation, k-匿名化，統計的開示制御

Ryo SUZAKI†, Yoko KAMIDOI††, and Shin'ichi WAKABAYASHI††

† Faculty of Information Sciences, Hiroshima City University
3-4-1 Ohzuka-higashi, Asaminami-ku, Hiroshima, 731–3194 Japan
†† Graduate School of Information Sciences, Hiroshima City University
3-4-1 Ohzuka-higashi, Asaminami-ku, Hiroshima, 731–3194 Japan
E-mail: †suzaki@lcl.info.hiroshima-cu.ac.jp, ††{yoko,wakaba}@hiroshima-cu.ac.jp

Key words　microaggregation, k-anonymization, statistical disclosure control

1. はじめに

近年，情報技術の進化により，企業に大量に蓄積した情報を統計的に解析することで新しい有用な情報が得られるようになった．企業等に集められた情報を第三者が解析できるように提供する際，元々の情報からの損失を減らしつつ，かつ，個人のプライバシーが守られるように情報を変換し利用する必要がある．つまり，プライバシーに対する個人の権利と企業の知る要求のバランスを取ることが重要である． 情報をそのまま第三者に提供すると個人が特定されてしまう．したがって，個人のプライバシーを保護するために情報を変換し提供しなければならないが，情報を変換すると元の情報が変化するので情報損失が起こる．企業等の第三者はできるだけ情報損失の少ないデータを要求する一方で，提供する側はプライバシー保護の条件を満たさなければならない．この両者のトレードオフのバランスを取ることが必要となる．このバランスのとり方の 1 つの方法に統計的開示抑制 (statistical disclosure control : SDC) がある． 本論文では，SDC の方法の一つである microaggregation を利用した k-匿名化に注目する．k-匿名化とは守りたい属性情報の値の組合せが少なくとも k 個存在するという性質である k-匿名性を満たしているデータに元々のマイクロデータを加工することである．データ管理者によって k が与えられたとき，k-匿名性を満たしているデータに加工する方法の 1 つに microaggregation がある．Microaggregation は k-匿名性を満たし，かつ，情報損失も少ないデータに変換する方法の 1 つとして知られている．本研究では microaggregation の一手法で文献 [1] で提案された手法に着目する．Microaggregation 手法ではプライバシー保護しなければならないマイクロデータを構成するレコードの集合が与えられたとき，集合をいくつかのグループに分割する．このとき，各グループは k 個のレコードを含む．分割した各グループで重心を計算し，その重心値にグループ内のレコードの値を置き換える．この k-匿名化されたデータは集約情報なのでプライバシー保護されている．しかし，公開するデータの情報損失は最小化されなければならないが，大規模で多属性のデータセットに対しては問題が複雑となり，情報損失が最小な解を求めることは困難と予測される．文献 [1] の手法もヒューリスティックである．またこの手法は，大規模なデータセットに対して変換データを作成するのに多大な時間がかかる．したがって，本稿では，従来手法の情報損失の解の高品質化，及び，microaggregation を用いた既存 k-匿名化手法の高速化の両立を目指す．

2. Microaggregation を利用した k-匿名化

2.1 k-匿 名 化

匿名化の尺度に安全性を表す k-匿名性がある．k-匿名性とは，データから個人が直接特定できる識別子を取り除いたうえで，守りたい準識別子の値の組合せのレコードが少なくとも k 個あるような状態であることを示す．k はデータ管理者によって前もってセットされる定数であり，プライバシー保護の評価尺度を示す．本稿では，特に明記しない場合を除いて各レコードは準識別子の属性値のみからなるものと仮定する．

2.2 Microaggregation の概要

Microaggregation とは，マイクロデータに対する SDC の既存方法の 1 つであり，連続数値データのために設計されたデータ変換法である．Microaggregation の動作は，大まかに分割と集約の 2 つに分類される．

Partition (分割)　分割とは，元のレコードの集合 (マイクロデータ) を，いくつかのグループに分けることである．分けられたグループは以下の特性を持つ．(1) 同じグループ内のレコードは互いに類似していること，(2) 各々のグループ内のレコードの数が少なくとも k 個以上であること．最小のグループのサイズがこの k 個以上であるという条件 (2) を満たしている分割を本論文では k-分割と呼ぶ．

Aggregation (集約)　集約とは，分割された各グループ内で各レコードを数値ベクトルと見なしたときの重心を計算し，グループの各々のレコードを，グループの重心によって置き換えることである．属性の数が 1 つの場合は，1 次元で表すことができるので平均と呼ぶが，本研究で扱うマイクロデータの属性数は 2 以上であるため各レコードは多次元数値ベクトルで表現され，この場合は平均とは呼ばず重心と呼ぶ．

2.3 標　準　化

Microaggregation では数十にも及ぶ属性の項目を持つデータのレコード集合を類似したレコード集合に分割しなければならない．中には身長や体重と言ったように，単位系が違い比較するのに適していない項目がある．そこで，多次元データの各属性の影響を均一にするため，属性の値を標準化する．標準化とは，データ等を一定のルールに基づいて変形し，単位系や数値の大きさによらない利用しやすいデータにすることである．データの標準化は，以下の流れで行う．

（1）　各属性ごとの平均値を計算する．

（2）　各属性のレコードの値から，属性に対応している平均値を引く．

（3）　各属性のレコードを 2 乗した上で全てを足し，レコード数で割ることで分散値を計算する．

（4）　分散値に平方根を使用し，標準偏差を計算する．

（5）　手順 2 の値を標準偏差で割ると，標準化される．

以後では，標準化されたデータが入力マイクロデータとして与えられると仮定する．また，表 1 を元々のマイクロデータ，表 2 をマイクロデータの準識別子 (この例では面積と従業員数) を標準化したデータとして表す．

表 1　元々のマイクロデータ

会社名	面積 (m^2)	従業員数	取引高 (Euros)	利益 (Euros)
A&ALtd	790	55	3,212,334	313,250
B&BSpA	710	44	2,283,340	299,876
C&CInc	730	32	1,989,233	200,213
D&DBV	810	17	984,983	143,211
E&ESL	950	3	194,232	51,233
F&FGmbH	510	25	119,332	20,333
G&GAG	400	45	3,012,444	501,233
H&HSA	330	50	4,233,312	777,882
I&ILLC	510	5	159,999	60,388
J&JCo	760	52	5,333,442	1,001,233
K&KSarl	50	12	645,223	333,010

表 2　標準化したマイクロデータ

会社名	面積 (m^2)	従業員数	取引高 (Euros)	利益 (Euros)
A&ALtd	0.777	1.296	3,212,334	313,250
B&BSpA	0.457	0.704	2,283,340	299,876
C&CInc	0.537	0.058	1,989,233	200,213
D&DBV	0.857	-0.748	984,983	143,211
E&ESL	1.416	-1.502	194,232	51,233
F&FGmbH	-0.341	-0.318	119,332	20,333
G&GAG	-0.781	0.758	3,012,444	501,233
H&HSA	-1.060	1.027	4,233,312	777,882
I&ILLC	-0.341	-1.394	159,999	60,388
J&JCo	0.657	1.135	5,333,442	1,001,233
K&KSarl	-2.179	-1.017	645,223	333,010

2.4 情 報 損 失

2.4.1　情報損失の定義

データ変換に伴う情報損失のことを以下では SSE と記述する．SSE とは，sum of squared Euclidean distances の略語である．SSE は以下のように定義される．

$$SSE = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (X_{ij} - \overline{X}_i)^2 \qquad (1)$$

- g はグループの数
- n_i は i 番目のグループ内に含まれるレコードの個数
- X_{ij} は i 番目のグループの j 番目のレコード
- $\overline{X}_i$? は i 番目のグループの平均

SSE の値が小さいほど情報損失は小さい．

2.4.2　情報損失の度合い

SSE を総平方和で割ることで，0 と 1 の間で囲まれている情報損失の尺度を導き出す．総平方和を英語表記にすると，total sum of squares であるので，以後，総平方和を SST と記述する．SST とは，全レコードを 1 つのグループにしたときの情報損失である．SSE/SST が 0 になるときは，マイクロデータを変換せずにそのまま第三者に公開した場合である．そのまま第三者にマイクロデータを提供しているので，情報損失が起きていない．つまり，SSE が 0 であるので SSE/SST は 0 となる．SSE/SST が 1 になるときは，マイクロデータの全てのレコードを同じグループにした場合である．したがって，SSE/SST

の値が 0 に近い程，情報損失は小さく，1 に近い程，情報損失は大きいことを示している．

2.5 Microaggregation 問題の定式化

本稿で対象とする microaggregation 問題は以下のように定式化される．

[入力]　n 個のレコードからなる標準化されたマイクロデータ，パラメータ k (k はデータ管理者によって前もってセットされる定数)

[出力]　情報損失が最小となる k-分割 microaggregation 結果

[評価関数]　情報損失度合い

3. Microaggregation の従来手法

本節では，文献 [1] の microaggregation 手法を詳細に説明し，その問題点を述べる．

3.1 従 来 手 法

p 個の準識別子を持つ n 個のレコードで構成されるデータセットは，R^p 中で n 個の点 $X_1, \ldots, X_n$ と表現できる (R^p とは，p 次元空間のことである)．

Microaggregation に対して，文献 [1] で提案されているデータ指向のヒューリステックは以下の 3 つの手続きにより構成される．

A　データセットの全ての n 個の点を通過する経路 T を探す．点が T によって探索される順序を πT として表す．πT は $\{1, \ldots, n\}$ の順列である．

B　Hansen-Mukherjee アルゴリズム（Multivariate version of Hansen-Mukherjee algorithm: MHM アルゴリズム）を使用し，手続き A で順序付けされたデータセット πT=$(X_{\pi T(1)}, \ldots, X_{\pi T(n)})$ に対するデータ指向 k-分割を出力する．

C　MHM アルゴリズムによって出力された k-分割を使って，microaggregation する．

microaggregation の一連の動作過程の例を図 1 に示す．

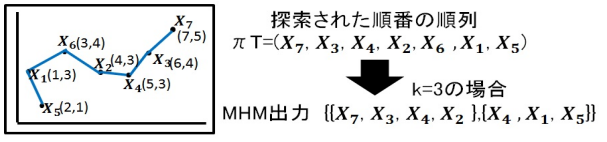

図 1　手続き A，B の適用例

以降の部分節で microaggregation の 3 つの手続きについて詳しく説明する．

3.2 経 路 作 成

microaggregation のためのヒューリスティックの最初の手続き A について説明する．まず経路作成には最近傍点:Nearest Point Next (NPN) 探索を使用する．NPN 探索とは，1 つの点に対して最も近い点を次の点として探していく探索方法である．全ての経路構築は，近くの点を繋ぐことで行われる．近くの点を繋ぐことは，点に対応するレコードに近いレコードを順に並べ経路を作成することになる．それは，microaggregation の目的に沿っているように思われる．なぜなら，データセットを分割し各グループに分ける際に，経路で隣接する 2 つのレコードは互いに類似した値になってないといけないので，隣接するレコードに対応する点と点の間が短いような経路は目的に沿っているように思われる．経路作成は以下の 3 つのステップにより行われる．

1.　データセット中のすべての点の重心 $\overline{X}$ を計算する．
2.　$\overline{X}$ から最も遠い点を計算し，最も遠い点を経路の最初の点 r とする．
3.　最初の点 r から (残りの点の間で)，最も近いものを 2 点目とする．1 番目と 2 番目を除いた点の中で，2 番目の点に最も近い点を 3 点目とし，そして，すべての n 個の点が経路に追加されるまで探索を繰り返す．

以上の 3 つのステップからデータセットの全ての n 個の点を通過する経路 T を見つけ，経路 T を探索されたレコードの順列 πT として表現する．経路作成の例を図 1 に示す．ステップ 1 より，データセット内のすべての点の重心を計算し，ステップ 2 で重心から最も遠い点を計算し，最も遠い点を経路の最初の点 r とする．図 1 の最初の点は X_7 である．ステップ 3 より，NPN 探索で経路を作成する．最初の点 r から残りの点の中で最も近い点を 2 点目とする．2 点目を決定するために最初の点以外の点に対し距離計算をする．図 1 の 2 点目は X_3 である．3 点目も同様にして探索し，すべての n 個の点が経路に追加されるまでこの探索を繰り返して経路を作成する．そして探索された順番の順列 πT を求める．図 1 で経路の探索された順列を表す πT は，$\pi T = (X_7, X_3, X_4, X_2, X_6, X_1, X_5)$ となる．

次に，経路作成の実行時間について述べる．経路作成の実行時間は $O(n^2)$ である．n 点のデータセットについては，以下の計算が必要となる．

- n 個のデータセットの重心を計算する．
- データセットの重心から最も遠い点 r を探す．
- r で始まっている経路を作成する．経路の 2 点目を決定するのに，$n-1$ 個の点に対する距離計算が必要となる．3 点目を決定するのに，$n-2$ 個の点に対する距離計算を必要とする．そして，$n-1$ 番目の点を決定するのに，残りの 2 つの点に対する距離計算が必要であり，最後の n 番目の点は，残りの点である．

4. MHM アルゴリズム

手続き B は一次元データ用の microaggregation 手法 [2] で用いられた手続きを多次元データ用に拡張した方法であり，MHM アルゴリズム (Multivalue version of Hansen-Mukherjee algorithm) と呼ぶ．以下に MHM アルゴリズムの出力が満たす理論的性質について説明する．

[定義 1]　順列 πT に矛盾しない k-分割：p-次元点の順列 πT=$(X_1, \ldots, X_n)$ が与えられたとき，k-分割中の任意のグループ G と，$i \leqq j \leqq k$ を満たす任意の 3 点 X_i, X_j, X_k において，$X_i, X_k \in G$ に含まれるなら，X_j も G に含まれるという条件を満たすなら，k-分割は順列 πT に矛盾しないとする．

[定理 1]　MHM アルゴリズムは，与えられた順列に矛盾しない k-分割の中で最も情報損失の小さい k-分割を出力する．

MHM アルゴリズムは，有向グラフの作成と最短経路探索という，以下の 2 つのフェーズで構成される．

フェーズ 1　　有向グラフの作成

1.1　順列 πT 中の各々の値 X_i に対して，ラベル i のノードを作成．また，ラベル 0 の追加ノードを作成．

1.2　$i+k \leqq j < i+2k$ を満たすノードの各々の組 (i,j) に対して，ノード i からノード j の有向枝 (i,j) を作成．

1.3　各枝（i,j）に対して，$C_{(i,j)}$=$\{X_h : i < h \leqq j\}$ を対応づけ，$L_{(i,j)}$ を枝 (i,j) の長さとし，グループ $C_{(i,j)}$ に属する各値 X_h からグループの重心 $\overline{X}_{(i,j)}$ までの距離の二乗和とする (つまり，$\overline{X}_{(i,j)}$ は $C_{(i,j)}$ のレコードの重心として計算)．

$$L_{(i,j)} = \sum_{h=i+1}^{j} (X_h - \overline{X}_{(i,j)})^2 \qquad (2)$$

ここで，有向グラフ作成の実行時間を説明する．有向グラフ作成の実行時間は $O(k^2n)$ である．グラフの枝数を E としたとき，枝の個数は，$|E| < kn$ であり，1 本の枝の長さの計算にかかる演算数は，$O(k)$ である．したがって，ステップ 1.3 の計算時間は，$O(kn)$ と $O(k)$ の乗算より $O(k^2n)$ となる．

フェーズ 2　　最短経路探索 フェーズ 1 で作成した有向グラフのノード 0 と n の間の最短経路を探索する．最短経路の各枝に対応するグループが，MHM によって出力される k-分割としてヒューリスティックの手続き C で microaggregation される．

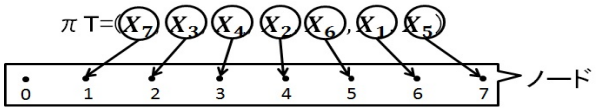

図 2　フェーズ 1.1 の例

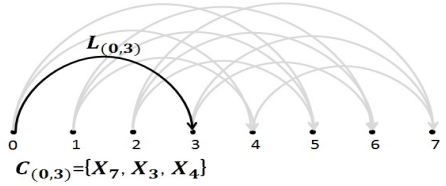

図 3　フェーズ 1.2 とフェーズ 1.3 の例

最短経路探索の実行時間は一般的な場合，$O(n\log n)$ である．一般的なグラフに対する最短経路アルゴリズム (ダイクストラ [3]) より，n 回の優先キュー探索があるので $O(n\log n)$ となる．

この MHM アルゴリズムに順列 πT が与えられたときに，その順列 πT に対しては最適な k-分割 microaggregation 出力を導出することが定理 1 より言える [1], [2]．しかし，全体の microaggregation 問題に対して，常に情報損失最小の結果が出力できるかどうかは保証されない．なぜなら，手続き A の作成経路が手続き B において最適な解を導出可能な入力かどうかわからないからである．よって，従来手法は全体としては発見的手法 (ヒューリスティック) となる．

4.1　従来手法における課題

文献 [1] の microaggregation 手法における問題点を述べる．まず計算量が $O(n^2)$ と計算時間が大きいことが挙げられる．大規模なデータセットに対して microaggregation を利用して変換データを作成するのに多大な時間がかかる．特に手続き A の経路作成ではデータ数が増加するほど 2 乗オーダで計算時間が増加してしまう．次に，情報損失の解の質が手続き A の出力された順列 πT にどの程度依存しているか不明確であることが挙げられる．一般に，点と点が近いような NPN 探索によって形成される順列 πT が最適であるとは言えないからである．したがって，microaggregation の高速化と，より適した経路探索方法を模索する必要がある．さらに，プライバシー保護レベル k に大きく依存する計算量 $O(k^2n)$ を持つ有向グラフ作成も幅広い保護レベルに対応させようとしたとき，計算時間が急激に増大する要因となる．

5. 提 案 手 法

本研究では，従来手法より高速に，より microaggregation の目的に沿った経路を作成すると共に，MHM アルゴリズムの入力有向グラフを高速に作成し，最短経路探索も高速化することを目的として新しい手法を提案する．

5.1　提案手法の概要

従来手法の手続き A に対して，空間 R^p を複数のブロックに分割しブロック数倍の高速化と，ブロック内に限定した経路探索による類似した点集合の順列化を目標とした手続き A’ を提案する．同時に，従来手法の手続き B の有向グラフ作成の高速化を目標とした手続き B’ と最短経路探索の高速化を目標とした手続き B” を提案する．提案手法の概要を以下に示す．

A’　データセットを R^p で表現したとき，空間 R^p を複数のブロックに分割する．ある点の次に通過する点をブロック内の未通過点の中で最も近い点のみとすることで n 個の点を通過する経路 T を求める．このとき，ブロック内に未通過点がなければ隣接ブロックを探索する (計算量 $O(n^2)$ のブロック数倍の高速化が目標)．

B’　A’ で求めた順列 πT で有向グラフを作成する．有向枝のコストを順列 πT に従った順序で計算する．以前に計算した枝コストを用いて，各有向枝のコストを帰納的に定数回の演算適用で計算する (計算量 $O(kn)$)．

B”　B’ で求めた有向グラフの最短経路を探索する．A’ で求めた順列より作成する有向グラフがある条件を満たしているため，一般的なグラフに対する最短経路アルゴリズム (ダイクストラ) を使用せず計算する (計算量 $O(kn)$)．

5.2　有向グラフ作成の高速化

この部分節では MHM アルゴリズムのフェーズ 1 の有向グラフ作成に注目し，提案手続き B’ について説明する．従来手法の計算量が $O(k^2n)$ であるのに対し，提案手法を利用して計算量を $O(k^2n)$ から $O(kn)$ に削減する．具体的には，1 本の枝の長さの計算にかかる演算数 $O(k)$ を，以前に計算した枝コストを用いて，各有向枝のコストを帰納的に定数回，厳密には 4 回の加減算の適用で計算することで $O(kn)$ とする．以下に 1

本の有向枝のコストを定数回で演算するための一般式を示す．図 4 は一般式を考える際のノード番号と各ノードに対応しているレコードを示している．まず枝 (i,j) の枝コストを従来手法を用いて計算する場合を考える．(i,j) に対応するグループは $C_{i,j}=\{x_{i+1},\ldots,x_{j-1},x_j\}$ でありこのときの枝コスト $L_{i,j}$ の重心を $y_{i,j}=(x_{i+1}+\ldots+x_{j-1}+x_j)/(j-i)$ とし，$L_{i,j}$ を展開整理すると，

$$\begin{aligned}L_{i,j} &= \sum_{n=i+1}^{j}(X_n-y_{i,j})\\ &= (x_{i+1}-y_{i,j})^2+\ldots+(x_{j-1}-y_{i,j})^2+(x_j-y_{i,j})^2\\ &= x_{i+1}^2-2x_{i+1}y_{i,j}+y_{i,j}^2+\ldots+x_{i-1}^2-2x_{j-1}y_{i,j}+y_{i,j}^2\\ &\quad +x_j^2-2x_jy_{i,j}+y_{i,j}^2\\ &= x_{i+1}^2+\ldots+x_{j-1}^2+x_j^2-2y_{i,j}(x_{i+1}+\ldots+x_{j-1}+x_j)\\ &\quad +(j-i)y_{i,j}^2 \qquad (3)\end{aligned}$$

となる．(i,j) の直前の枝を $(i,j-1)$ としたときの重心を $y_{i,j-1}=(x_{i+1}+\ldots+x_{j-1})/(j-1-i)$ とし同様に展開整理すると，

$$\begin{aligned}L_{i,j-1} &= \sum_{n=i+1}^{j-1}(X_n-y_{i,j-1})\\ &= (x_{i+1}-y_{i,j-1})^2+\ldots+(x_{j-1}-y_{i,j-1})^2\\ &= x_{i+1}^2-2x_{i+1}y_{i,j-1}+y_{i,j-1}^2+\ldots+x_{j-1}^2-2x_{j-1}y_{i,j-1}\\ &\quad +y_{i,j-1}^2\\ &= x_{i+1}^2+\ldots+x_{j-1}^2-2y_{i,j-1}(x_{i+1}+\ldots+x_{j-1})\\ &\quad +(j-1-i)y_{i,j-1}^2 \qquad (4)\end{aligned}$$

となる．(i,j) の直前の枝 $(i,j-1)$ の重心 $y_{i,j-1}$ を使用し $y_{i,j}$ の重心を計算すると，$y_{i,j}=((j-i-1)y_{i,j-1}+x_j)/(j-1)$ と表すことができる．さらに，式 (3) に (4) と (i,j) と $(i,j-1)$ の重心を利用し変形すると，

$$\begin{aligned}L_{i,j} &= L_{i,j-1}+2y_{i,j-1}(x_{i+1}+\ldots+x_{j-1})-(j-1-i)y_{i,j-1}^2+x_j^2\\ &\quad -2y_{i,j}(x_{i+1}+\ldots+x_{j-1}+x_j)+(j-i)y_{i,j}^2\\ &= L_{i,j-1}+2y_{i,j-1}(j-1-i)y_{i,j-1}-(j-1-i)y_{i,j-1}^2+x_j^2\\ &\quad -2y_{i,j}(j-i)y_{i,j}+(j-i)y_{i,j}^2\\ &= L_{i,j-1}+2(j-1-i)y_{i,j-1}^2-(j-1-i)y_{i,j-1}^2+x_j^2\\ &\quad -2(j-i)y_{i,j}^2+(j-i)y_{i,j}^2\\ &= L_{i,j-1}+(j-1-i)y_{i,j-1}^2+x_j^2-(j-i)y_{i,j}^2 \qquad (5)\end{aligned}$$

と表現することができる．この式 (5) を利用すると以前の枝 $L_{i,j-1}$ と重心 $y_{i,j-1},y_{i,j}$ が求まっていたとき，それらの値を利用し以降の枝のコストを計算することができる．しかし，この式 (5) で求めることができるのは始点の枝が同じ場合の枝コストのみである．すなわち，式 (5) は 1 つのノードからは最大で $k-1$ 本の枝が作成され 1 本目の枝のコストが算出されていたら，それ以降の枝コストは 1 本目の枝コストの解を利用して求めることができることを示した式である．始点のノードが隣に移動するとまた 1 本目の枝コストを $O(k)$ で計算する必要がある．したがって，次に始点が隣に移動した場合の一般式を考える．始点が隣に移ったときの枝を (i,j) と仮定する．(i,j) に対応するグループは $C_{i,j}=\{x_{i+1},\ldots,x_{j-1},x_j\}$ でありこのときの枝コスト $L_{i,j}$ の重心を $y_{i,j}=(x_{i+1}+\ldots+x_j)/(j-i)$ とし $L_{i,j}$ を展開整理すると，

$$\begin{aligned}L_{i,j} &= \sum_{n=i+1}^{j}(X_n-y_{i,j})\\ &= (x_{i+1}-y_{i,j})^2+\ldots+(x_j-y_{i,j})^2\\ &= x_{i+1}^2-2x_{i+1}y_{i,j}+y_{i,j}^2+\ldots+x_j^2-2x_jy_{i,j}+y_{i,j}^2\\ &= x_{i+1}^2+\ldots+x_j^2-2y_{i,j}(x_{i+1}+\ldots+x_j)+(j-i)y_{i,j}^2 \qquad (6)\end{aligned}$$

となる．次にノードが 1 つ前で終点が同じ枝 $(i-1,j)$ を考える．このときの重心を $y_{i-1,j}=(x_i+x_{i+1}+\ldots+x_j)/(j-i+1)$ とすると，

$$\begin{aligned}L_{i-1,j} &= \sum_{n=i}^{j}(X_n-y_{i-1,j})\\ &= (x_i-y_{i-1,j})^2+(x_{i+1}-y_{i-1,j})^2+\ldots+(x_j-y_{i-1,j})^2\\ &= x_i^2-2x_iy_{i-1,j}+y_{i-1,j}^2+x_{i+1}^2-2x_{i+1}y_{i-1,j}+y_{i-1,j}^2\\ &\quad +\ldots+x_j^2-2x_jy_{i-1,j}+y_{i-1,j}^2\\ &= x_i^2+x_{i+1}^2+\ldots+x_j^2-2y_{i-1,j}(x_i+x_{i+1}+\ldots+x_j)\\ &\quad +(j-i+1)y_{i-1,j}^2 \qquad (7)\end{aligned}$$

となる．(i,j) の 1 つ前のノードで終点が同じ枝 $(i-1,j)$ の重心 $y_{i-1,j}$ を使用し $y_{i,j}$ の重心を計算すると，$y_{i,j}=((j-i+1)y_{i-1,j}+x_{i-1})/(j-i)$ と表わせる．式 (6) に (7) と (i,j) と $(i-1,j)$ の重心を利用し整理すると，

$$\begin{aligned}L_{i,j} &= L_{i-1,j}-x_i^2+2y_{i-1,j}(x_i+x_{i+1}+\ldots+x_j)-(j-i+1)y_{i-1,j}^2\\ &\quad -2y_{i,j}(x_{i+1}+\ldots+x_j)+(j-i)y_{i,j}^2\\ &= L_{i-1,j}-x_i^2+2y_{i-1,j}(j-i+1)y_{i-1,j}-(j-i+1)y_{i-1,j}^2\\ &\quad -2y_{i,j}(j-i)y_{i,j}+(j-i)y_{i,j}^2\\ &= L_{i-1,j}-x_i^2+2(j-i+1)y_{i-1,j}^2-(j-i+1)y_{i-1,j}^2\\ &\quad -2(j-i)y_{i,j}^2+(j-i)y_{i,j}^2\\ &= L_{i-1,j}-x_i^2+(j-i+1)y_{i-1,j}^2-(j-i)y_{i,j}^2 \qquad (8)\end{aligned}$$

と表現することができる．この式 (8) を利用すると 1 つのノードから出ている有向グラフの枝コストを全て計算し終え隣のノードに移ったとしても，再び $O(k)$ の計算を行うことなく終点が同じで 1 つ前のノードの枝コストの値を利用し計算することができる．この 2 つの一般式 (5) と (8) を利用することで以前に計算した枝コストを用いて，各有向枝のコストを帰納的に定数回の演算適用で計算することで $O(kn)$ で計算可能となる．以上を補題 1 としてまとめる．

[補題 1] 枝コスト $L_{i,j}$ のコストは，枝コスト $L_{i,j-1}$ と重心 $y_{i,j-1}$ か枝コスト $L_{i-1,j}$ と重心 $y_{i-1,j}$ を利用することにより 4 回の加減算の適用により計算することができる．

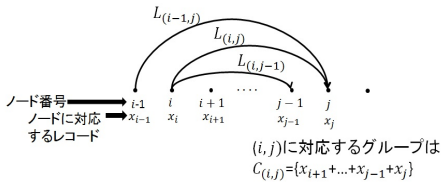

図 4　着目した枝コスト間の関係

補題 1 より以下の定理が成り立つ．

[定理 2] MHM アルゴリズムのフェーズ 1 の有向グラフは $O(kn)$ の計算量で作成できる．

5.3　経路作成の高速化と高品質化

本節では提案手続き A’ について説明する．提案手続き A’ は経路作成に注目し，従来手法より高速に，より microaggregation の目的に沿った経路作成を目標とする．従来手法の microaggregation 問題において全体の計算時間を支配している箇所はこの経路作成である．計算量は $O(n^2)$ であるのでデータ数が多くなればなるほど計算時間の大部分を支配することとなる．そのため経路作成を改善すれば大幅に速度を向上できると考えられる．経路作成での従来手法の問題点を述べる．r で始まっている経路を作成する際，経路の 2 点目を決定するのに最初の点以外の残りの点 $n-1$ 個分の距離計算が必要となり，3 点目を決定するのに 1 点目 2 点目以外の残りの点 $n-2$ 個の距離計算をしなければならない．次の点を探索するのに経路に追加されていない残りの点全てに対して距離計算をしているところに大きな無駄が生じている．経路作成は点と点の間が短くなるような経路を探索するのだが，この方法では明らかに次点を決定するのに必要ない遠い点まで探索しているため計算時間増加の問題が生じる．次点を探索する量を減らすことができれば，計算時間に関する問題を解消できる．

次点を探索する計算量を減らすため，データセットを R^p 中で表現できることを利用する．データセットをプロットした空間 R^p を複数の空間に分割する．探索された点の次に通過する点をブロック内の未通過点の中で最も近い点のみとすることで従来手法の残りの点の全ての探索を防ぐことができる．このとき，ブロック内に未通過点がなければ周囲の隣接ブロックを探索する．隣接ブロック内のみの探索となるので，ここでも残りの点全ての探索は不要となる．周囲の隣接ブロック内に 1 つもレコードがなければ探索範囲を 1 段階大きくする．これを繰り返して経路を探索する．最大まで探索範囲を広げてもレコードがなければ探索を終了する．R^p 空間を複数の空間に分割することで最初の 1 点目の探索数も削減することができる．1 点目は全体の重心からもっとも遠い点なので分割したブロックの外周りに探索を絞ることで探索数を削減できる．この R^p 空間を分割する方法を使用することで大幅な時間短縮が可能になると考えられる．この節のもう一つの目標である microaggregation の目的に沿った経路作成について説明する．文献 [1] の microaggregation のためのヒューリスティックスである経路作成に使用される NPN 探索は microaggregation の目的に沿っていると考えられていた．なぜなら，データセットを分割する際に，レコードは類似した値になってないといけないので，この NPN 探索の点と点の間が短いような経路は目的に沿っていると考えられるからである．しかしここで，図 5 の簡単な例で従来手法の NPN 探索を用いて形成された経路を k=5 で分割するときと，図 5 の空間を 16 個のブロックに分割し手続き A’ で作成した経路を表す図 6 を k=5 で分割するときを比較する．そうすると，最初の点 r から分割していくと明らかに図 6 の方がグループの重心値に類似したレコードが集まりそうなことがわかる．したがって，実際には従来手法の経路探索は microaggregation の目的に沿っているとは言えない．提案手法のブロック分割を用いた NPN 構成で作成した経路の方が従来手法よりも経路をグループに分割するときグループの重心に近いレコードをより集めることができる可能性が高い．なぜなら，従来手法とは違い提案手法では類似したレコードが集まっているブロック内で経路を作成するからである．

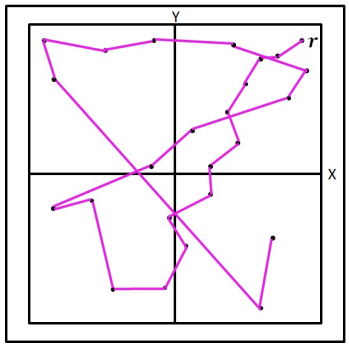

図 5　従来手法の経路

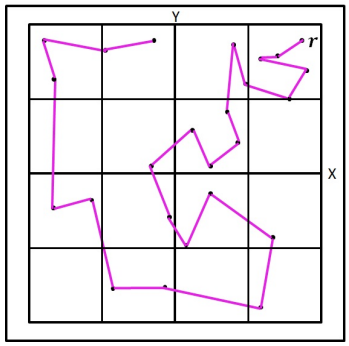

図 6　提案手法での経路

6.　最短経路探索の高速化

文献 [1] の最短経路探索の実行時間は $O(n\log n)$ である．一般の有向グラフに対する最短経路アルゴリズム (ダイクストラ) を適応しているので $O(n\log n)$ となる．この $O(n\log n)$ を以下に示す独自に開発したアルゴリズムを使用することで $O(nk)$ とする．このアルゴリズムは，開発後にトポロジカルソートされ

た無閉路有向グラフの経路探索のための動的計画法と同じ計算時間を持つことが判明した．従来手法の手続き A での経路探索によって形成される順列 πT を使用して有向グラフを作成すると，トポロジカルソートされた無閉路有向グラフとなる．無閉路有向グラフとは閉路を持たない有向グラフである．また，トポロジカルソートとは無閉路有向グラフの各ノード (ここでは，順列 πT の各レコードに対応しているノード) を順序付けしたもである．順序付けとは，各ノードは出力辺の先のノードより前にくるように並べた状態である．手続き B で作成した有向グラフのどのノードの枝も，出力辺の先のノードより前にきておりトポロジカルソートされた無閉路有向グラフとなっているので，以下のアルゴリズムを使用して最短経路を探索することができる．

まず，ここで対象とする最短経路問題の入力とそれに伴う出力を定義する．

入力　有向グラフ $G = (V, E)$，枝コスト関数 $l : E \to R^+$，有向グラフ G のノード数 $n+1$，ノードの順序 πT(順番にノードを一対一に対応させる関数 πT: $0, \ldots, n \to v$)，ここで有向グラフ G では，各枝 (u, v) において $\pi T^{-1}(u) < \pi T^{-1}(v)$ が成り立つ

出力　節点 $\pi T^{-1}(0)$ から節点 $\pi T^{-1}(n)$ の最短経路 ST(逆順)

最短経路探索のアルゴリズムの概要は以下の通りである．

ステップ 1　$d(\pi T(0))$ に 0 を代入，$\pi T(0)$ ではない全ての頂点 v に対して $d(v)$ に ∞ を代入．

ステップ 2　$for(i = 0; i <= n; i++)$ 以下のステップ 2.1 を繰り返す．

ステップ 2.1　頂点 $\pi T(i)$ を選び，頂点 $\pi T(i)$ を始点とする各枝 $(\pi T(i), v)$ に対してステップ 2.1.1. を実行．

ステップ 2.1.1　$if(d(v) > d(\pi T(0)) + l(\pi T(i), v))$
$then(d(v) \leftarrow d(\pi T(i)) + l(\pi T(i), v), f(v) \leftarrow \pi T(i))$

ステップ 3　$ST[0] \leftarrow \pi T(n)$

ステップ 4　$for(i = 1; i <= n \;\&\&\; ST[i-1]! = \pi T(0); i++)$ 以下のステップ 4.1 を繰り返す．

ステップ 4.1　$ST[i] = f(ST[i-1])$

以上のステップより最短経路を求めることができる．5 節で論じる従来手法と提案手法の比較実験の際には，本節で説明した最短経路法を従来手法，提案手法共に適用し計算機上に実現した．したがって，従来手法の実装は文献 [1] の従来手法の実装より理論的に高速な実装となっている．次章で従来手法と提案手法に有向グラフ作成と経路作成の高速化を実装した場合の計算時間を比較する．また，提案の経路作成による SSE への影響についても比較する．

7. 評価と考察

7.1 実験環境

従来手法と提案手法を C 言語を用いて実装し，Borland C++ 5.5.1 コンパイラを用いてコンパイルし，CPU : Intel Core i7-3770 CPU 3.40GHz，メインメモリ : 8.00GB，OS : Windows7 Professional 64bit の計算機上に実現しシミュレーション実験を行った．実験で用いたデータは，UCI KDD アーカイブの Paralyzed Veterans of America の募金対象者ベンチマークデータ (データ数 50 万，属性数 36) と属性を限定して投影し作成したデータ (データ数 50 万，属性数 2) である．以後では，提案手法 1 を有向グラフ作成の高速化，提案手法 2 を提案手法 1 を用いて経路作成の高速化を行ったときの提案手法とする．

7.2 実験結果

表 3 と表 4 はベンチマークデータ (データ数 50 万，属性数 36) を用いて従来手法と提案手法 1 の実験結果を示している．表 5 はベンチマークデータ (データ数 50 万，属性数 2) を用いて従来手法と提案手法 2 の空間を 16 分割した場合と 64 分割した場合の実験結果である．表 3 では計算時間に関して提案手法 1 と従来手法を比較している．表 4 は従来手法，提案手法 1 両方で出力された解の情報損失を示している．提案手法 1 の解は従来手法の解と同じであるため，情報損失も同じである．また，提案手法 1，従来手法ともに適用した開発最短経路探索手法に要した計算時間も表 4 に記している．表 5 では，計算時間と出力された解の情報損失に関して提案手法 2 と従来手法を比較している．

表 3　従来手法と提案手法 1 の計算時間の比較

		従来手法の計算時間 [秒]			提案手法 1 の計算時間 [秒]		
要素数	k	全体	経路作成	枝コスト計算	全体	経路作成	枝のコスト計算
1 万	2	11.1	11.1	0.01	11.3	11.3	0.01
	5	11.2	11.1	0.1	11.3	11.3	0.03
	10	11.4	11.0	0.4	11.4	11.3	0.04
	50	20.6	11.1	9.5	11.5	11.3	0.2
	100	49.3	11.2	38.1	11.7	11.3	0.4
5 万	2	330.0	330.0	0.1	340.8	340.8	0.05
	5	332.8	332.2	0.5	340.7	340.5	0.1
	10	341.4	339.4	2.0	343.0	342.8	0.2
	50	384.2	335.4	48.8	342.0	340.9	1.0
	100	539.4	338.2	201.2	339.1	337.1	2.1
10 万	2	1446.0	1445.8	0.2	1424.3	1424.2	0.1
	5	1443.5	1442.4	1.1	1425.9	1425.7	0.2
	10	1450.4	1446.2	4.2	1427.3	1426.8	0.4
	50	1546.8	1445.8	100.9	1458.5	1456.3	2.2
	100	1855.4	1448.7	406.7	1464.8	1460.5	4.2
30 万	2	13184.9	13184.3	0.5	13401.3	13401.0	0.01
	5	13189.0	13185.9	3.1	13380.5	13379.8	0.7
	10	13420.3	13408.1	11.9	13364.0	13363.0	1.3
	50	13484.1	13186.6	297.3	13230.8	13224.4	6.2
	100	15065.5	13841.1	1224.2	13438.3	13425.5	12.6
50 万	2	36272.6	36271.7	0.9	37127.2	37126.7	0.6
	5	37411.3	37406.0	5.4	37164.8	37163.6	1.1
	10	36794.5	36774.5	19.9	37603.4	37601.2	2.2
	50	37217.4	36719.2	198.0	37590.7	37579.7	10.2
	100	38219.3	36253.3	1965.8	37149.7	36290.3	20.3

7.3 考　　察

表 3 と表 4 の実験結果より，有向グラフ作成の提案手法を適用することで，解の質に影響を与えることなく高速化が達成できていることがわかる．しかし，全体の実行時間を比較すると要素数が増加するにつれて大きな時間短縮とはなっていない．このことからも，microaggregation 問題の実行時間を支配しているのは経路作成だと言える．

表 4　従来手法と提案手法 1 の出力結果の情報損失と最短経路探索の計算時間

要素数	k の値	情報損失 [$\times 10^2$]	最短経路探索計算時間 [秒]
1 万	2	4.091	0.001
	5	15.221	0.001
	10	21.874	0.001
	50	34.981	0.004
	100	41.598	0.005
5 万	2	5.441	0.002
	5	12.472	0.002
	10	19.342	0.004
	50	34.865	0.014
	100	39.422	0.026
10 万	2	4.200	0.104
	5	9.401	0.229
	10	15.387	0.436
	50	32.217	2.092
	100	38.923	4.151
30 万	2	3.290	0.315
	5	6.272	0.689
	10	8.90	1.302
	50	20.801	6.249
	100	27.143	12.597
50 万	2	4.337	0.531
	5	7.689	1.148
	10	10.061	2.176
	50	17.887	10.234
	100	22.979	20.345

表 5　従来手法と提案手法 2 の比較結果

		計算時間 [秒]		
n	k	従来手法	提案手法 (16 分割)	提案手法 (64 分割)
10 万	25	133	36	30
	100	157	36	30
30 万	25	1588	310	233
	100	1675	314	231
50 万	25	4703	1030	893
	100	4868	1026	889
		情報損失 [$\times 10^4$]		
n	k	従来手法	提案手法 (16 分割)	提案手法 (64 分割)
10 万	25	5.64	4.22	3.52
	100	38.17	23.55	20.66
30 万	25	3.72	1.78	1.63
	100	14.58	9.46	7.79
50 万	25	1.29	1.11	1.00
	100	7.05	6.24	5.04

表 5 の実験結果より提案手法 2 は経路作成の高速化と高品質化を達成している．高速化については分割数が多い方がより高速化を達成している．これは，ブロック数が多い方が各ブロックにレコードがばらけているので，ブロック内の未通過点の経路探索とブロック内の未通過点がない場合の周囲のブロック内の探索の際に探索回数が削減できているからである．しかし，ブロック数倍の高速化を目標としていたが 16 分割 64 分割ともに約 5 倍程度の高速化しか達成できなかった．これはレコードが均等にブロック内になかったからである．実際は，1 つのブロックにレコードの片寄りが生じていたためブロック数倍の高速化までとはならなかった．また，提案手法 2 では類似しているレコードが集まるブロック内に限定した探索により従来手法の点と点の近い経路とは違い，提案手法ではすでに探索された集団に近いレコードを次の点として選択するため情報損失も低減できた．情報損失についてもブロック数が多い方がより低減できている．16 分割の場合は平均 29 %情報損失を改良し，64 分割の場合は平均 40 %情報損失を改良している．これは，ブロック数が多い方がブロック内のレコードの重心と各レコードの値がより近い値となるため，内で経路を作成しグループに分割するときにより類似したレコードを含むグループ分けをすることができるためと考えられる．以上の結果より，microaggregation を用いた k-匿名化に対し，提案手法は従来手法と比較して高速化を実現し，さらに，解の質の向上も同時に達成することができたと結論できる．

8. まとめと今後の課題

本研究では，microaggregation を用いた k-匿名化手法の高速化，及び，解の質の向上を目的として，新たな手法を提案した．主な提案の 1 つ目として，有向グラフの作成時に 1 本の枝の長さの計算にかかる演算回数 $O(k)$ を，以前に計算した枝コストを用いて各有向枝のコストを帰納的に定数回で計算するように工夫したことで，有向グラフ作成の計算時間を従来手法の $O(k^2n)$ から $O(kn)$ に改良した．理論的に計算の正当性を確認するとともに，計算機実験により解の質を落とさず高速化を達成していることを確認した．2 つ目の提案として，経路作成時に空間 R^p を複数のブロックに分割することで経路作成の NPN 探索数を削減した．また，各ブロック内で経路を生成することにより，グループに分割するときにより類似したレコードを含むグループ分けが可能となり，従来手法と比較して情報損失の低減も達成した．計算機実験により，解の質を向上させかつ高速化も達成していることを確認した．

今後の課題としては，空間 R^p を複数のブロックに分割した場合に，もし，1 つのブロック内にレコードの片寄りが生じていた場合は，そのブロックをレコードの個数が一定値以下になるまで階層的に分割するブロック構成方法の拡張が挙げられる．また，本研究では空間 R^p の分割手法が現在，2 属性のデータにしか対応していないため，数十属性を持つデータにも対応できるように手法を拡張することが求められる．

文　献

[1] J. Domingo-Ferrer, A. Martinez-Balleste, JM. Mateo-Sanz and F. Sebe “Efficient multivariate data-oriented microaggregation,” The VLDB Journal, Vol.15, pp. 355-369 (2006).

[2] S. L. Hansen and S. Mukherjee, “A polynomial algorithm for optimal univarate microaggregation,” IEEE Trans. Kowl. Data Eng., Vol.15, No.4, pp. 1043-1044 (2003).

[3] S. Even, “Graph Algorithms,” Computer Science Press, (1979).