高次元データに対するMicroaggregationを利用したk-匿名化手法

須崎 亮*† 橋本 翔太† 上土井 陽子‡ 若林 真一‡

† 広島市立大学情報科学部
広島市安佐南区大塚東三丁目４－１
shotah@lcs.info.hiroshima-cu.ac.jp

‡ 広島市立大学大学院情報科学研究科
広島市安佐南区大塚東三丁目４－１
(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，格子データ構造を用いた探索が低次元データのmicroaggregationに有効であることを示し，さらに，高次元データのmicroaggregationを対象とした格子の大きさが一定でない可変格子データ構造に基づくk-匿名化手法を提案する．

A Technique for k-Anonymity Based on Microaggregation Toward High Dimensional Datasets

Ryo Suzaki† Syota Hashimoto† Yoko. Kamidoi‡ Shin'ichi Wakabayashi‡

†Faculty of Information Sciences,
Hiroshima City University
3-4-1, Ohzuka-Higashi, Asaminami-ku,
Hiroshima, 731-3194, JAPAN
shotah@lcs.info.hiroshima-cu.ac.jp

‡Graduate School of Information Sciences,
Hiroshima City University
3-4-1, Ohzuka-Higashi, Asaminami-ku,
Hiroshima, 731-3194, JAPAN
(yoko,wakaba)@hiroshima-cu.ac.jp

Abstract Microaggregation is one of good masking methods for statistical disclosure control. Recently, microaggregation has been shown to be useful to achieve k-anonymity of microdata. A previous work shows, that a microaggregation method based on neighborhood queries and shortest path searches outputs good quality k-anonymized data for small input data. In this paper, we propose a new microaggregation technique for k-anonymity, in order to improve performance of the previous method and quality of solutions. In addition, we compare theoretically and experimentally computational complexities and qualities of solutions of the proposed and the previous methods.

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．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変換する方法の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．

本稿では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

* 現在，マイクロテクノ株式会社．

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．情報損失の最小化を目的とするデータ変換問題は単属性データの場合，最適解を多項式時間で求めることができる．一方，多属性のデータセットに対しては上記のデータ変換問題は NP-困難である [1]．このため，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効率よく求めることは困難と予測される．文献 [1] の手法は単属性データセットに対する最適解法 [3] を拡張した多属性データセットに対するヒューリスティックであるが，大規模データに対しては計算時間が大きく，情報損失も必要以上に大きい可能性が高い．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．さらに，高次元データにおいても高品質な解を高速に計算するために，クラスタリング問題を解決するために著者らが開発したデータ構造 [4] を Microaggregation に利用する．

2 Microaggregation を利用した k-匿名化

2.1 マイクロデータと k-匿名化

本稿で対象とする入力データベースであるマイクロデータに関する用語について定義する．ここでは，マイクロデータは表として表現されていると仮定する．このとき，属性とはマイクロデータの列に対応する項目にあたる．また，表の各行は一個人のデータであり，レコードと呼ぶ．入力である非保護のマイクロデータの属性は主に 3 つに分類することができる．以下に属性の 3 つのタイプを記述する．また，マイクロデータの例を表 1 に記す．

識別子 識別子は，データの持ち主を明白に識別する属性である．レコードの持ち主が第三者に特定されるのを防がなければならないので，この識別子は削除または暗号化される．

準識別子 準識別子は，識別子とは違い 1 つだけでは持ち主を特定することはできない．しかし，他の複数の属性を組み合わせることでレコードの持ち主を特定できる可能性がある．

機密の出力属性 3 つ目は持ち主に関する機密情報を含む属性である．

本稿で microaggregation を利用した k-匿名化を行うため変換の対象とする属性の値は，準識別子にあたる属性の値である．

データの安全性を表す尺度に 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．マイクロデータを k-匿名性を満たすように変換することを k-匿名化と呼ぶ．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り，各属性値は数値であると仮定する．

2.2 Microaggregation の概要

Microaggregation とは，マイクロデータに対する統計的開示制御 (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稿では 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．

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．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，標準偏差を計算する．
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き，その値を該当属性の標準偏差で割ると，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 Microaggregation 問題

2.4.1 情報損失

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．

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

ここで，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とする．

SSE を総平方和で割ることで，0〜1 の範囲の情報損失度合いという尺度を導き出す．総平方和 SST を全レコードを 1 つのグループにしたときの情報損失とする．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．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，情報損失度合い 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

[**出力**] 情報損失が最小となる k-分割の 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，p 次元空間 R^p 中で n 個の点 $X_1, \ldots, X_n$ として表現できる．

Microaggregation に対して，文献 [1] で提案されている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 [3] 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 手続き B 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．

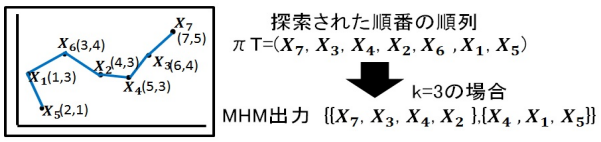


図 1: 手続き A，B の適用例

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．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．その後，経路上にない点で経路上最後の点に最も近い点を経路に追加する．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．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．

上記の経路作成の実行時間は $O(n^2)$ 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，以降，このアルゴリズムを 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，$i \leq j \leq k$ を満たす任意の 3 点 X_i, X_j, X_k において，$X_i, X_k \in G$ を満たすとき，$X_j \in G$ という条件を満たすなら，k-分割は順列 πT に矛盾しないと言う．

[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 $i + k \leq j < i + 2k$ 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，$C_{(i,j)}$=$\{X_h : i < h \leq j\}$ を対応づけ，$L_{(i,j)}$ を枝 (i, j) の長さとし，グ

ループ $C_{(i,j)}$ に属する各レコード 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．

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

ここで，有向グラフ作成の実行時間は $O(k^2n)$ である．

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．

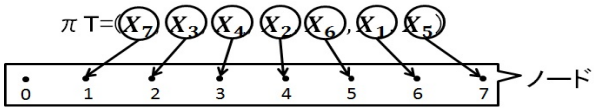


図 2: フェーズ 1.1 の例

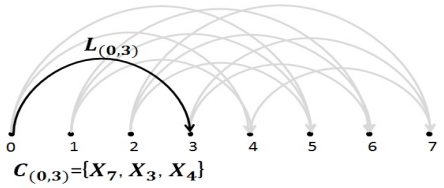


図 3: フェーズ 1.2 とフェーズ 1.3 の例

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 [2]) 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], [3]．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．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，p 次元空間 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，p 次元空間 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サイクルを含まず，トポロジカルソートされているという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) を使用せず動的計画法にて計算する (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 $C_{(i,j)} = \{X_{i+1}, \ldots, X_{j-1}, X_j\}$ である．このとき，重心 $\overline{X}_{(i,j)}$ を $(y1_{(i,j)}, y2_{(i,j)}, \ldots, yp_{(i,j)})$ と表すとする．ここで，$X_{i+1} = (x1_{i+1}, x2_{i+1}, \ldots, xp_{i+1})$ と表し，$X_{i+2}, \ldots, X_j$ に対して成分に同様な表現を与えたとき，重心 $\overline{X}_{(i,j)}$ の l 番目 $(1 \le l \le p)$ の成分は $yl_{(i,j)} = (xl_{i+1} + \ldots + xl_{j-1} + xl_j)/(j-i)$ を満たす．枝コスト $L_{(i,j)}$ のレコードの l 番目 $(1 \le l \le p)$ の成分による増加分を $Ll_{(i,j)}$ と表すと $L_{(i,j)}$ は以下のように表現できる．

$$
\begin{aligned}
L_{(i,j)} &= \sum_{h=i+1}^{j} (X_h - \overline{X}_{(i,j)})^2 \\
&= \sum_{l=1}^{p} ((xl_{i+1} - yl_{(i,j)})^2 + \ldots + (xl_j - yl_{(i,j)})^2) \\
&= \sum_{l=1}^{p} Ll_{(i,j)}
\end{aligned}
\tag{3}
$$

以降では，枝コスト $L_{(i,j)}$，任意の l $(1 \le l \le p)$ におけるレコードの l 番目の成分による増加分 $Ll_{(i,j)}$ のみに注目し，$Ll_{(i,j)}$ を展開する．また，以降では簡単のため，$C_{(i,j)}$ に属するレコードの l 番目の成分を x で，$C_{(i,j)}$ の重心の l 番目の成分を $y_{(i,j)}$ と表す．このとき，$Ll_{(i,j)}$ を展開整理すると，

$$
\begin{aligned}
Ll_{(i,j)} &= \sum_{h=i+1}^{j} (x_h - y_{i,j})^2 \\
&= x_{i+1}^2 + \ldots + x_{j-1}^2 + x_j^2 - 2y_{(i,j)}(x_{i+1} + \ldots + x_{j-1} + x_j) \\
&\quad + (j-i)y_{(i,j)}^2
\end{aligned}
\tag{4}
$$

となる．(i,j) の直前の枝を $(i,j-1)$ としたときの重心 $\overline{X}_{(i,j-1)}$ の l 番目の成分を $y_{i,j-1} = (x_{i+1} + \ldots + x_{j-1})/(j-1-i)$ とし同様に展開整理すると，

$$
\begin{aligned}
Ll_{(i,j-1)} &= \sum_{h=i+1}^{j-1} (x_h - y_{(i,j-1)})^2 \\
&= x_{i+1}^2 + \ldots + x_{j-1}^2 - 2y_{(i,j-1)}(x_{i+1} + \ldots + x_{j-1}) \\
&\quad + (j-1-i)y_{(i,j-1)}^2
\end{aligned}
\tag{5}
$$

となる．(i,j) の直前の枝 $(i,j-1)$ の重心 $\overline{X}_{(i,j-1)}$ の l 番目の成分 $y_{i,j-1}$ を使用し，$y_{i,j}$ を表現すると，$y_{i,j} = ((j-i-1)y_{i,j-1} + x_j)/(j-1)$ と表すことができる．さらに，式 (4) に (5) と (i,j) と $(i,j-1)$ の重心の l 番目の成分を利用し変形すると，

$$
\begin{aligned}
Ll_{(i,j)} &= Ll_{(i,j-1)} + 2y_{(i,j-1)}(x_{i+1} + \ldots + x_{j-1}) - (j-1-i)y_{(i,j-1)}^2 \\
&\quad + x_j^2 - 2y_{(i,j)}(x_{i+1} + \ldots + x_j) + (j-i)y_{(i,j)}^2 \\
&= Ll_{(i,j-1)} + (j-1-i)y_{(i,j-1)}^2 + x_j^2 - (j-i)y_{(i,j)}^2
\end{aligned}
\tag{6}
$$

と表現することができる．この式 (6) を利用すると以前の枝の l 番目の増加分 $Ll_{(i,j-1)}$ と重心 $\overline{X}_{(i,j-1)}$, $\overline{X}_{(i,j)}$ の l 番目の成分 $y_{(i,j-1)}, y_{(i,j)}$ 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の l 番目の増加分を計算することができる．しかし，この式 (6) 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 l 番目の増加分のみである．すなわち，式 (6) 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．さらに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．注目する枝を (i,j) と仮定し，グループ $C_{(i,j)}$, 枝コスト $L_{(i,j)}$ に関する表現は上記と同様とする．

このとき，始点が 1 つ前のノードで終点が同じ枝 $(i-1,j)$ を考える．重心 $\overline{X}_{(i-1,j)}$ の l 番目の成分を $y_{(i-1,j)} = (x_i + x_{i+1} + ... + x_j)/(j-i+1)$ とすると，

$$\begin{aligned} Ll_{(i-1,j)} &= \sum_{h=i}^{j} (X_h - y_{(i-1,j)})^2 \\ &= x_i^2 + \ldots + x_j^2 - 2y_{(i-1,j)}(x_i + x_{i+1} + \ldots + x_j) \\ &\quad +(j-i+1)y_{(i-1,j)}^2 \end{aligned} \tag{7}$$

となる．枝 (i,j) において，始点が 1 つ前のノードで終点が同じ枝 $(i-1,j)$ の重心 $\overline{X}_{(i-1,j)}$ の l 番目の成分 $y_{i-1,j}$ を使用し，重心 $\overline{X}_{(i,j)}$ の l 番目の成分 $y_{i,j}$ を表現すると，$y_{(i,j)} = ((j-i+1)y_{i-1,j} + x_{i-1})/(j-i)$ と表わせる．式 (6) に (7) と重心の l 番目の成分 $y_{(i,j)}$ と $y_{(i-1,j)}$ を利用し整理すると，

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

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定数回の計算で終点が同じで始点が 1 つ前のノードの枝コストの値を利用し，枝コストを計算することができる．この 2 つの一般式 (6) 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 $L_{(i,j-1)}$ の各次元 $l(1 \leq l \leq p)$ の増加分 $Ll_{(i,j)}$ と重心 $\overline{X}_{(i,j-1)}$ か枝コスト $L_{(i-1,j)}$ の各次元 $l(1 \leq l \leq p)$ の増加分 $Ll_{(i-1,j)}$ と重心 $\overline{X}_{(i-1,j)}$ を利用することにより $4p$ 回の加減算の適用により計算することができる．

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 $O(kpn)$ の計算量で作成できる．

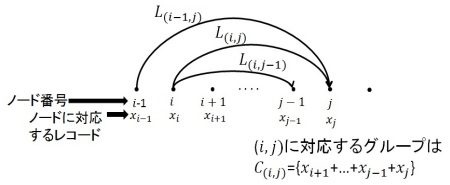


図 4: 着目した枝コスト間の関係

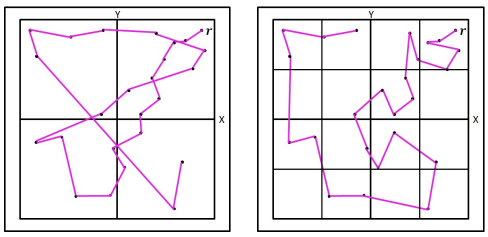
5.3 経路作成の高速化と高品質化

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．

次点を探索する計算量を減らすため，データセットを p 次元空間 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(a) 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(a) 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 5(b) 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 5(b) の方がグループの重心に類似したレコードを経路の短い範囲内に集め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．



(a) 従来手法　　(b) 提案手法

図 5: 従来手法と提案手法の経路作成の違い

さらに，経路作成の時間計算量を $O(n^2/k^2)$ に削減するため，1 つのブロック内のレコード数が定数以下となるようにブロックを作成できるか検討する．上記を実現するには，ブロック間にレコード分布の偏りが生じている場合，密なブロックのみ，各ブロック内のレコード数が定数以下になるまで，階層的に分割するブロック構成法が必要となる．我々は高次元空間でブロック数をレコード数に比べて，小さく抑えながら，ブロックの大きさが不均一な可変サイズ格子データ構造を構築するアルゴリズム FlexDice[4] を microaggregation に利用することを試みる．FlexDice は高次元データのクラスタリング向けに高次元空間を不均一なブロックに分割した後，ブロックを節点とし，ブロック間の隣接関係を枝として表現するグラフを構築する．ここで，各ブロックが隣接するブロック数を次元数 p の 2 倍以下に抑えることで，ブロック数を B としたとき枝数を $2pB$ 以下に抑えることが可能となる．また，FlexDice では高次元空間でブロックに割り当てられるレコードがない場合には，ブロックを作成しない．上記より，ブロック数がほぼ次元数に依存しないという特徴をもつブロック分割を実現できる．

本稿では，FlexDice において各ブロックが定数 C_1 より多いレコードを含んでいる場合には必ず，ブロックを分割し，さらに，ブロックが定数 C_1 以下の要素しか持たない場合でもブロック後のブロック数が定数 C_2 以下になる場合には分割を続けることによって，ブロック数とブロック内の要素数の双方を抑制する．上記により，各ブロック内でのレコードの順列化に $O(C_1^2)$, ブロック数を B としたとき，各ブロックからの隣接ブロックの探索に FlexDice で構築されるブロックに関するグラフ上での幅優先探索を利用するとグラフの枝数 $O(pB)$ の計算を必要とする．よって，経路作成の全体の計算量としては $O(C_1^2B + pB)$ となる．ここで，全体のブロック数を $B = O(C_2n/C_1)$ と表現できるとすると，経路作成の計算量は $O(C_1C_2n + p(C_2n/C_1)^2)$ と表現できる．定数 C_1, C_2 を $C_1 = pkC_2$ となるように調整することにより，経路作成の計算量を $O(n^2/(pk^2))$ にすることが可能となり，経路作成の高速化が可能になると予測される．

6　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．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．

今後の課題としては，数十属性を持つデータに対する提案手法の実験的性能評価が挙げられる．

参考文献

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

[3] 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).

[4] 中村 朋健, 上土井 陽子, 若林 真一, 吉田 典可, “FlexDice: 高次元な大規模データセットに対する高速クラスタリング手法”, 情報処理学会論文誌：データベース, Vol.46, No.SIG 18(TOD 28), pp.40-49 (2005).