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

中村朋健[†] 上土井陽子[††]
若林真一[††] 吉田典可[††]

高次元な大規模データを対話的に処理する場合，従来のクラスタリング手法では高い処理コスト，不十分なクラスタリング精度などが問題となっていた．我々はデータ空間の部分空間であるセルを動的に構築・削除可能なデータ構造を用いることで従来手法の問題点を軽減し，従来手法では困難であった高次元な大規模データセットを高精度かつ高速にクラスタリング可能な FlexDice を開発した．FlexDice は局所的なデータ密度を計算し，データ密度の高い連続した部分空間のデータ要素を集めるクラスタリング手法であり，2 つのフェーズからなる．第 1 フェーズでは局所的にデータ密度の高い部分空間を不均一なサイズのセルとして階層的に構築しながらセルの隣接関係を作成し，セルを節点，隣接関係を枝とするグラフ構造を作成する．第 2 フェーズでは，隣接したセルを結合することでクラスタを形成する．ベンチマークデータを用いたシミュレーション実験により，FlexDice が高精度にクラスタリング可能な従来手法と同等のクラスタリング結果を出力すること，高次元な大規模データに対して高速にクラスタリング可能なことを示す．

FlexDice: A Fast Clustering Method for Large High Dimensional Data Sets

Tomotake Nakamura,[†] Yoko Kamidoi,[††]
Shin'ichi Wakabayashi[††] and Noriyoshi Yoshida[††]

When processing large high dimensional data interactively, existing methods have issues of high processing cost or bad accuracy of clustering. We resolve those issues of existing methods by using a data structure, in which constructing and deleting cells are dynamically possible, and develop a new clustering method FlexDice which can perform clustering at high speed with high accuracy for large high dimensional data sets. FlexDice is a clustering method which groups data objects in adjacent dense data space. It consists of two phases. The first phase constructs hierarchically cells in a top-down fashion, and constructs a graph structure which has irregular size cells as nodes and neighboring links as edges by creating neighboring links every layer. The second phase constructs clusters by merging a set of cells having a neighboring relation with each other transitively. Simulation experimental results using benchmark data shows that it is possible for FlexDice to output clustering results having as same accuracy as ones found by existing methods which can perform clustering for large high dimensional data with high accuracy, and that FlexDice is faster than one of the fastest clustering methods, OptiGrid.

1. はじめに

近年の情報化社会の進展と記憶装置の低価格化，大容量化によりデータベースに蓄積されるデータは高次元であり大規模なものとなっている．今後はさらに高次元な大規模データベースが構築されることが予想される．大規模データベースが構築される一方で，蓄積された多くのデータを有効に活用できていないのが現状である．そのようなデータベースからデータの規則，特徴，そしてパターンを見つけ出すことで有用な情報を抽出するプロセスはデータマイニングと呼ばれ，その 1 つの技法にデータ要素をグループ化するクラスタリングがある．クラスタリングとは類似性の高いデータ要素を同じグループ（クラスタ）に集め，類似性の低いデータ要素を別々のクラスタに分ける操作である．多くの古典的なクラスタリング手法では，データ

† 広島市立大学大学院情報科学研究科
Graduate School of Information Sciences, Hiroshima City University
†† 広島市立大学情報科学部
Faculty of Information Sciences, Hiroshima City University

要素間の類似性をユークリッド距離やマンハッタン距離などの距離尺度によって定義し，それに基づいてクラスタを抽出する．我々の提案するクラスタリング手法 FlexDice はデータ空間における局所的なデータ密度を計算し，データ密度の高い連続した部分空間に含まれるデータ要素をクラスタとして集め，データ密度の低い部分空間に含まれるデータ要素をノイズとして 1 つのクラスタに集めるクラスタリング手法である．

我々はクラスタリング手法を意思決定支援に用いることを考えている．意思決定には様々な形態があるが，ここでは人とクラスタリングアルゴリズムが対話的に情報をやりとりすることで人の意思決定を支援することを想定している．たとえば，本を購入したいがどの本を購入してよいか分からない場合，人とコンピュータが対話的に情報をやりとりすることがあったとする．過去に同じような状況に遭遇した人々をクラスタリングによって探し出し，過去の人々の行動を参考にし，かつ，ユーザの意志を反映させて購入すべき本の選択を支援することが考えられる．人の行動などを蓄積したデータウェアハウスは今後ますます高次元かつ大規模になることが予測され，そのようなデータセットから対話的に同じような状況に遭遇した人々を探し出す必要がある．我々は 10 秒以内の応答を対話的な応答と想定している．我々がデータベースから情報を抽出したいとき，一般に，クラスタリングに要求されるクエリは初めから完全ではないことが多い．人々は概略を尋ねるクエリから始め，得られた知識からのフィードバックに基づいてクエリを洗練させていく[6]．このとき，対話的データマイニングでは概略的な解を高速に出力し，クエリの洗練が進むにつれて詳細な解を出力することが要求される．

90 年代後半から今日に至るまでにクラスタリング手法は多くの研究者によって活発に開発されてきているが，従来クラスタリング手法は 50 属性を持つ 100 万データ要素を含むような高次元な大規模データをクラスタリングするとき，依然として高い処理コストや不十分な精度などの問題点を持つ．現在，高次元な大規模データに対して高速に処理できる手法としては OptiGrid[8] や O-Cluster[10] があげられる．しかし，O-Cluster は，使用した計算機は不明であるが 50 属性を持つ 100 万データ要素を含む合成データに対して約 1,000 秒費やすため，対話的な応答が難しい．

我々は高次元な大規模データに対して対話的に応答可能な FlexDice を開発した．FlexDice はデータ密度の高い部分空間のデータ要素を含む不均一なサイズのセルを階層的にトップダウン方式で構築しながらセルの隣接関係を作成し，セルを節点，セル間の隣接関係を枝とするグラフ構造を作成する．隣接したセルを結合することでクラスタを形成する．トップダウン方式でセルを構築することによりユーザの定めた時間内に可能な限り精度の高いクラスタリング結果を出力可能なアルゴリズムとなっている．

データ空間を分割した後に結合させる DENCLUE[9] や STING[13] はボトムアップ方式でセルを結合させることでクラスタを形成する手法であるため，多次元な入力に対して隣接するセル数を制限することが難しく高精度な結果を得にくい[8]．OptiGrid と O-Cluster はトップダウン方式でクラスタの存在しないデータ空間を分割することで，その問題点を避けて高次元な入力をクラスタリング可能にした手法である．しかし，これらの手法はデータ空間の分割に高い処理コストを費やすため，対話的な処理が難しい．我々はトップダウン方式に密でも疎でもないセルのみに対して分割面を選択せずに分割することにより，データ要素の再走査を減らすことで高速な処理を可能にし，データ空間の過度に冗長な分割を避けることで高精度なクラスタリングを実現した．分割面の選択は行わないが，セルの密度を調べることにより分割すべきセルと分割すべきでないセルを選択し，さらに，セル間の隣接関係を作成して類似したデータ要素を含むセルを結合可能にすることにより，OptiGrid や O-Cluster と同様の精度を保っている．

シミュレーション実験では，クラスタリングの精度が高く，ユーザの欲するクラスタに近いクラスタを構築可能な Agglomerative 手法と FlexDice を比較し，FlexDice が同程度の精度でクラスタリング可能であることを示す．また，現在，高次元な大規模データに対して最も高速にクラスタリング可能な手法の 1 つである OptiGrid よりも高次元な大規模データに対して高速に処理可能であることを示す．

本論文では，2 章において従来手法と FlexDice の関係，3 章において FlexDice のアルゴリズムを説明する．4 章において，FlexDice の特徴として FlexDice のクラスタリング精度，処理のコスト，メモリ使用量，そしてパラメータの設定の困難さを見積もる．5 章において，FlexDice を実験的に評価する．

2. 従来手法と FlexDice の関係

既存のクラスタリング手法は分割手法（Partitioning methods），階層的手法（Hierarchical methods），密度に基づく手法（Density-based methods），格子に基づく手法（Grid-based methods），そしてモデル

に基づく手法（Model-based methods）に分類できる[7]．2.1 節において，FlexDice に特に関連した階層的手法，密度に基づく手法，そして格子に基づく手法と FlexDice との関連について述べ，2.2 節において，FlexDice と近年のクラスタリング手法との関係について述べる．

2.1 FlexDice に関連した手法

階層的手法はデータ要素集合に対してあらかじめ与えられた基準を満たすまで分割または結合を繰り返しクラスタを形成する．この手法に分類される手法には BIRCH[14] や CURE[4] などがある．階層的手法は結合または分割のどちらを基本とするかにより，さらに分類される．データ要素集合の分割には一般に高い処理コストを必要とするため，従来のほとんどの階層的手法は結合を基本とする手法である[7]．

密度に基づく手法は低密度の領域によって分けられる連続する高密度の領域のデータ要素を集める．この手法に分類される手法には DBSCAN[2]，OPTICS[1]，そして DBRS+[12] などがある．一般に密度に基づく手法の利点は任意の形のクラスタを判別できることやクラスタとノイズの区別が可能となることなどである．

格子に基づく手法は入力データ空間を格子状に分割したセルを用いてクラスタを形成する．この手法には STING[13] や WaveCluster[11] などが分類され，複数のデータ要素を含むセルを結合，または分割しクラスタを生成する．格子に基づく手法の利点は高速にクラスタリングできることである．データ要素のセルへの割当てを除くと，これらの手法の一般的な計算複雑さは $O(g)$ である．ここで g はセル数である．一般に，低次元な入力データであるとき $g \ll N$ であるが，高次元な大規模データが入力であるとき，構築するセル数の増大によりデータ空間の細かい分割が困難になる．また，これらの手法は高次元な入力であると隣接するセルが非常に多くなり隣接関係を作成することが困難になる．類似したデータ要素群を含む隣接したセルを結合できないことにより，十分な精度が得られない問題点を持つ．

我々は，高次元なデータを処理するために分割を基本とする階層的手法を採用し，さらに，任意の形のクラスタを抽出でき連続する高密度の領域のデータ要素をクラスタとして集めるため密度情報に基づき，高速にクラスタリングするために格子構造に基づく新しいクラスタリング手法である FlexDice を提案する．

2.2 FlexDice と近年のクラスタリング手法の比較

近年，クラスタリングの質を高める研究や処理コストを減らす研究が活発になされている[3),8),10)]．

階層的手法においては高次元なデータセットでは結合させるデータ要素群を探すことが困難であるため，データ空間の分割を繰り返すだけでクラスタを形成する OptiGrid や O-Cluster などの手法が開発された．OptiGrid[8] と O-Cluster[10] はクラスタが存在しないデータ空間を超平面で分割しクラスタを発見する手法であり，高次元な大規模データに適用可能な手法である．これらの手法と FlexDice はトップダウンにデータ空間を分割するという共通点を持つ．OptiGrid と O-Cluster の違いは OptiGrid は分割面の選択に入力パラメータを必要とするが O-Cluster はそれを必要とせず，OptiGrid はデータ空間を任意の角度で分割するが，O-Cluster はデータ空間の座標面に直交の分割だけである．O-Cluster は OptiGrid よりデータ空間の分割に費やすコストは少ないが，$O(D \cdot b)$ の計算複雑さを持つ．ここで D は入力データの属性数であり，b はデータ要素の写像をヒストグラムに変換したときの量子化の数である．文献 8) はあらかじめクラスタを形成した合成データを用いて，OptiGrid が DENCLUE や BIRCH などの従来手法よりもクラスタリングの質が高いことを示している．また，それらの従来手法より高速に処理可能であることを示している．文献 10) は O-Cluster も合成データを用いて正しくクラスタリングできた割合などを示すことでクラスタリング結果の質を示している．OptiGrid の計算コストは $O(N \cdot D)$ と近似でき，シミュレーション実験により入力データ要素数と計算時間の関係と入力データセットの属性数と計算時間の関係がともにほぼ線形関係であることを示している．ここで，N は入力データ要素数である．しかしながら，OptiGrid では分割面の選択に高い処理コストを費やすため，40 属性の 50 万データ要素を含むデータセットに対して対話的に処理できるほど高速には処理できていない．

OptiGrid と O-Cluster はデータ要素をノイズと判別できないアルゴリズムであり，クラスタ内に類似性の低いデータ要素が混在している可能性が高い．一方で FlexDice はノイズをノイズクラスタとして 1 つのクラスタに集めるため，ノイズクラスタ内でのデータ要素間の類似性は低いが他のクラスタに含まれるデータ要素間には高い類似性が保たれている．

文献 3) はいくつかのクラスタリングアルゴリズムと，それらのアルゴリズムの出力を集約して結果を出力する Clustering Aggregation を提案し，クラスタリング結果の精度を評価する基準を示している．Clustering Aggregation はベンチマークデータに対して高精度にクラスタリング可能であると評価されている ROCK[5]

と同等の精度でクラスタを出力でき，ROCK で扱うことができない 14 属性を持つ 32,561 データ要素を含むベンチマークデータ（census データ）を扱うことが可能である．Clustering Aggregation アルゴリズムの計算複雑さは $O(N^2 \cdot \log N)$ であり，クラスタ間の距離計算に高い処理コストを費やす手法である．文献 3) では，Aggregation アルゴリズムにおいて，合成データを入力とするシミュレーション実験より，要素数と計算時間が線形関係であり，このときの精度がほぼ一定であることが示されている．また，サンプリングを用いることで高速化できると結論付けている．

これらの近年に発表されているクラスタリング手法は高次元な大規模データを対象としている手法が多いが，どの手法も高い処理コストによって我々が目標とする対話的な処理の実行が難しい．5 章において，近年に発表されたクラスタリングの精度が高い Clustering Aggregation と現在最も高速な手法の 1 つである OptiGrid と提案手法 FlexDice をクラスタリング精度や処理時間について比較し，FlexDice の処理能力を評価する．

3. FlexDice アルゴリズム

提案手法 FlexDice は階層構造，密度情報，そして格子構造を用いるクラスタリング手法である．我々は様々なサイズのサイコロ（dice）のようなセルを階層的に構築し，さらにセルが柔軟に（flexibly）変化することから，提案手法を FlexDice と名付けた．ここで，セルとは入力データ空間の超直方体である部分空間と定義する．FlexDice は D 次元線形空間のデータに対して，以下の 2 つの事項を達成することを目標としたクラスタリング手法である．ただし，本論文における密度とは 1 辺が単位長さである D 次元超立方体あたりの要素数のことである．

- 低密度（疎）な領域によって分けられる高密度（密）な領域のデータ要素を集める．
- 高次元な大規模データに対して，高精度，かつ，対話的な応答時間でクラスタリングする．

FlexDice は 3.1 節と 3.2 節に示す 2 つのフェーズ（Ph.1，Ph.2）から構成される．Ph.1 では階層的に第 0 層から最下位層まで順次セルを構築・削除しながらセル間の隣接リンクを作成する．Ph.2 では Ph.1 で作成した密なセルをノード，隣接リンクを枝とするグラフから密なセルを結合してクラスタを形成する．FlexDice はクラスタリング結果に影響を及ぼす 4 つの入力パラメータ（P_{MAX}，P_{MEAN}，P_{MIN}，P_{LAY}）を持つ．最下位層以外における密セル（dense cell）はセルの密度が P_{MAX} 以上のセルであり，中セル（middle cell）はセルの密度が P_{MIN} 以上かつ P_{MAX} より小さいセルであり，疎セル（sparse cell）はセルの密度が P_{MIN} より小さいセルのことである．最下位層では中セルは作成されず，P_{MEAN} のみで密セルと疎セルに分類する．セルの密度が P_{MEAN} 以上のセルを密セル，P_{MEAN} より小さいセルを疎セルとする．P_{LAY} は最大の最下位層数であり，階層数はたかだか P_{LAY} である．

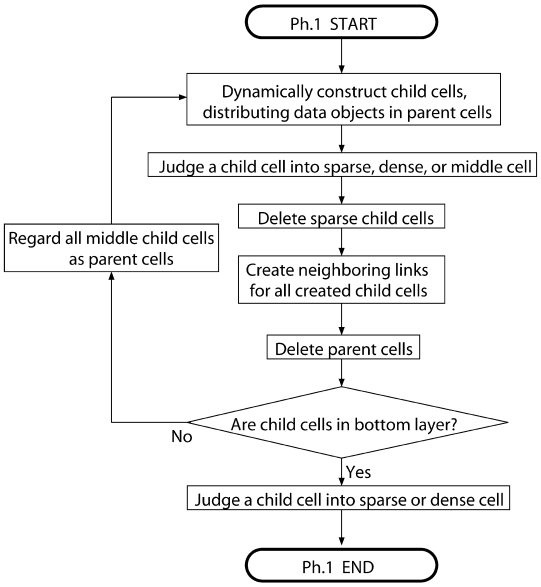

図 1 FlexDice アルゴリズム：Ph.1 のフローチャート
Fig. 1 A FlexDice algorithm: Flowchart of Ph.1.

3.1 Ph.1 —クラスタ領域の抽出と隣接リンクの作成

FlexDice の Ph.1 のフローチャートを**図 1** に示す．

Ph.1 の主な処理は密セルを抽出することと，密セルに隣接リンクを作成することである．密・中・疎のセル群は親セルを各属性に関して 2 等分，つまり 2^D 分割してできた空間にデータ要素が存在するとき子セルとして構築される．セルは入力データ空間である第 0 層から最下位層まで分割を繰り返し，構築される．ただし密セルまたは疎セルと判断されたセルはそれ以上分割を進めない．親セルと子セル間には双方向のリンクを作成する．ただし，親セルから子セルへのリンクの選択は $O(1)$ で選択可能にするためハッシュ関数を用いる．

第 k 層で作成された中セルに含まれるデータ要素を第 $(k+1)$ 層のセルへ振り分ける．第 $(k+1)$ 層が最下位層でないとき，第 $(k+1)$ 層へデータ要素の振り分けが終了すると，第 $(k+1)$ 層で作成されたセルの密度を調べ，P_{MAX} と P_{MIN} を用いてセルを疎セル，

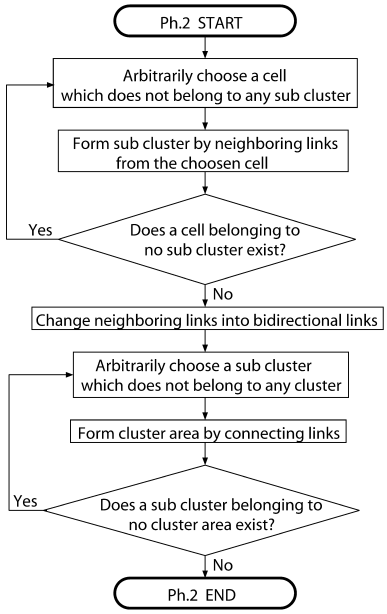

図 2 FlexDice アルゴリズム：Ph.2 のフローチャート
Fig. 2 A FlexDice algorithm: Flowchart of Ph.2.

中セル，そして密セルに分類する．$(k+1)$ が P_{LAY} に等しいとき，第 $(k+1)$ 層へデータ要素の振り分けが終了すると，第 $(k+1)$ 層で作成されたセルの密度を調べ，P_{MEAN} を用いてセルを疎セルと密セルに分類する．疎セルと判断されたセルはメモリ上から削除され，疎セルに含まれていたデータ要素はノイズとして 1 つのグループに集められる．第 $(k+1)$ 層で密・中セルと判断されたセルは第 $(k+1)$ 層のすべてのセルの構築の終了後に作成された隣接リンクにより隣接するセルとつながる．隣接リンクは入力データ要素が D 属性であったとき $(D-1)$ 次元超平面で接するサイズの等しい，または，大きいセルに対してのみ作成する．ここで $(D-1)$ 次元超平面とは入力データ空間が 1 次元データ空間 $(D=1)$ であるとき点，2 次元データ空間であるとき線，3 次元データ空間であるとき 2 次元平面のことである．隣接リンクは異なる階層に存在するセル間でも作成される．第 $(k+1)$ 層のセルの隣接リンクを作成し終わると第 k 層の中セルとそのセルが保持する隣接リンクをメモリ上から削除する．

3.2 Ph.2 —セル結合とクラスタの形成

図 2 に FlexDice の Ph.2 である密セルの結合方法のフローチャートを示す．Ph.2 の主な処理は，Ph.1 で構築した密セルと作成した隣接リンクから隣接する密セルを結合し，結合したセル群に含まれるデータ要素をクラスタとすることである．Ph.1 で作成した隣接リンクは単方向リンクと双方向リンクが存在するため，グラフの連結成分抽出アルゴリズムを直接適用することはできず，単方向リンクを双方向リンクに修正する必要がある．Ph.2 において提案するアルゴリズムは，密セル群をサブクラスタとしてある程度まとめ，サブクラスタ間の単方向リンクを双方向リンクに変換し隣接した密セル群を結合する．

4. FlexDice の特徴

FlexDice は低次元な小規模データから高次元な大規模データに対して，高精度，かつ，対話的な応答ができる高速なクラスタリングアルゴリズムである．本章において，FlexDice のクラスタリング精度，処理コスト，メモリ使用量，そしてパラメータの設定コストを見積もる．

高次元な大規模データセットを高精度にクラスタリングしたときの状況の再現として，50 属性の 100 万データ要素を含むデータセットに対して 10 階層までセルを構築したときの例を状況 1 とする．この例を用いて，費やすメモリ量や処理コストを以下で具体的に説明する．

4.1 クラスタリング精度

OptiGrid や O-Cluster はデータ空間上のデータ要素が少ない面を切断してクラスタリングするために高精度なものとなっている．一方で，FlexDice はトップダウン方式に高密度，または，低密度でないセルのみを分割することによりセルの冗長な分割を避けているが，データ空間の切断面を選択せずにデータ空間を分割するため，類似性の高いデータ要素を含むデータ空間を分割する可能性がある．分割すべきセルとそうでないセルを密度によって分類することで類似性の高いデータ要素を含むデータ空間の分割を極力避けている．分割すべきでない空間を分割した場合でも，セル間に隣接関係を作成し最後に類似したデータ要素を含むセルを結合させることで，高精度にクラスタリング可能となった．また，トップダウン方式に疎でも密でもないセルだけを分割する方法により，データ要素の再走査を減らすことで高速にクラスタリング可能である．

4.2 処理コスト

FlexDice はデータ空間の分割面の選択に計算時間を費やさず，分割後の結合にもほとんど計算時間を費やさないことにより高速化を実現した．OptiGrid や O-Cluster はデータ空間を分割する際にクラスタの存在しない領域を選択するが，FlexDice はデータ空間を各次元に関して 2 等分を繰り返す．このため，FlexDice での分割面の選択の計算複雑さは $O(1)$ である．OptiGrid や O-Cluster はセル間に隣接関係を

作成しないが，FlexDice は隣接関係を作成する．隣接関係作成の計算複雑さは $O(g)$ である．ここで g はセル数である．

FlexDice の Ph.1 では全データ要素を各階層ごとに走査する可能性があり，構築した全セルに隣接リンクを作成しなければならない．1 データ要素の走査は D 個の数値を操作する必要があるので，最悪の場合 $ND(K-1)$ 回の数値の走査が必要となる．FlexDice では要素数を多く含む密なセルや疎なセルに含まれるデータ要素の走査を避けることで，実用的には最悪の場合はほとんど起こらない．

隣接リンクの作成は，各階層ごとに隣接リンクを作成するため，D 本の隣接リンクを作成するために 2 本，さらに D 本の隣接リンクを作成するため 3 本のリンクをたどることで隣接リンクを作成できる．したがって 1 つのセルの隣接リンクの作成には最大で $5D$ 本のリンクをたどることで作成できる．4.3 節で述べたように，FlexDice は構築するセル数を削減しているため隣接リンクを作成するためのコストも削減できている．

FlexDice の Ph.2 は密セル数に依存した処理時間を必要とする．Ph.2 の処理コストは，密セル数は N であり結合するセルが見つからなかったときが最悪の場合である．このとき計算複雑さは $O(N+S^2)$ である．ここで S は Ph.2 で作成されるサブクラスタ数であり，一般に $S \ll N$ といえる．セルの不均一サイズ化により密セル数を削減し，単方向のリンクによってサブクラスタを形成することにより，ここでの計算量を少なく抑えることが可能となった．

4.3 メモリ使用量

一般に，格子に基づく手法は構築するセル数よりも入力データ要素数が多く，構築するセル数はメモリ使用量に影響を及ぼさないと考えられる．しかし，入力が高次元のデータセットであるとき，セル数が莫大な量になりメモリ使用量に大きな影響を及ぼすことがある．階層数を 1 つ増加させると各セルからは 2^D 個のセルが構築され，また構築されたセルが削除できない構造であるとき，構築セル数は $(2^D+1)^{K-1}$ 個となる．状況 1 のとき，構築されるセル数は 1.25×10^{27} 個となり現在の計算機環境ではとても処理できない．FlexDice はデータ要素が格納されている葉セルとその親セルだけのセル構築でよいため，最悪の場合でも構築するセル数は次元数や階層数に依存することなく $2N$ 個となる．高次元なデータ空間では均一な分布はほとんどないことが知られている[8]．したがって，高次元なデータセットに FlexDice を適用したとき構築するセル数は $2N$ よりも大幅に少なくなることが多いと予測される．

FlexDice が保持するセルはデータ要素が格納されたセルとその親セルだけであるが，Ph.2 において隣接した密セルを結合させなければならないため，セル間に隣接関係（隣接リンク）を作成する．隣接リンクは各セルに隣接するセルの数だけあればよいが，最悪の場合 1 つのセルに $2D \cdot 2^{(D-1)(K-3)}$ 個，かつ，$(N-1)$ 個以下のセルが隣接することがある（$K>3$）．状況 1 のとき，1 つのセルに隣接するセルの最大個数は 999,999 個となる．FlexDice は隣接リンクに費やすメモリ量を削減するために，入力データ要素が D 属性であったとき $(D-1)$ 次元超平面で接するサイズの大きい，または等しいセルに対してのみ隣接リンクを作成する．この条件により FlexDice では 1 つのセルが保持する隣接リンクは最大で $2D$ 個に限定できる．

FlexDice は階層的にセルを構築するが，FlexDice が用いるデータ構造は木構造ではなくグラフ構造であり，木構造のルートに対応する入力データ空間を表すセルや子セルを構築した後に不要となったセルを削除可能である．Quadtree や Octree では 1 度作成したセルをメモリから削除することが難しいため，FlexDice が用いるデータ構造は Quadtree や Octree のデータ構造よりセルに費やすメモリ使用量を削減できる．

4.4 パラメータの設定

3 章で述べたように，FlexDice はクラスタリング結果に影響を及ぼす入力パラメータが 4 つ存在する．他のクラスタリング手法において必要とされることがあるクラスタ数を定めるような難しいパラメータはないが，セルの疎密を判定する 3 つのパラメータは連続的に変化させユーザの欲する結果を抽出できる値に設定しなければならない．設定する順番は P_{MAX}，P_{MIN}，そして P_{MEAN} の順であり，出力結果に大きな影響を与えるパラメータから順に設定する．この 3 つのパラメータの設定はユーザの負担となると考えられるので，現在の設定方法よりも容易に設定できるように工夫することは今後の課題である．FlexDice の実行時間がユーザが結果を欲する時間に近づいたとき，P_{MIN} と P_{MAX} を P_{MEAN} に自動的に切り替え，階層数が P_{LAY} に達する前にクラスタリング結果を出力することで対話的な応答が可能になっている．この対話的な応答が可能になったのは，FlexDice がトップダウン方式にセルを構築するアルゴリズムであるためである．

5. FlexDice の評価実験

本章において，FlexDice を実験的に評価する．5.1 節

では評価実験のための準備として，扱う入力データやクラスタリング結果の評価方法を説明する．5.2 節において，FlexDice と従来手法を精度と処理速度について比較する．

5.1 評価実験の準備

5.2 節以降では，SUN Ultra60 Model1450（CPU: 450 MHz，Main memory: 1 GByte）を使用して実験した．提案手法 FlexDice と従来手法 OptiGrid は C++言語で実現し，従来手法 Clustering Aggregation は著者から提供されたソースコードを用いてシミュレーション実験した．結果を評価する指標として計算時間を定義する．計算時間は入力データを主記憶上に取り込んだ後から主記憶上にクラスタが形成されるまでの時間とする．以降の図において計算時間は Time で表される．

5.1.1 入力データ

入力データセットとして，UCI KDD アーカイブ[15] の “Census Income Database”（収入データ），“Mushrooms Database”（キノコデータ），“Forest Cover Type”（森データ），そして “1990 US Census Data”（US データ）の 4 種のベンチマークデータを使用する．収入データは 14 属性を持つ 32,561 データ要素から構成され，キノコデータは 22 属性を持つ 8,124 データ要素から構成され，森データは 54 属性を持つ 581,012 個のデータ要素から構成され，US データは 68 属性を持つ 2,458,285 個のデータ要素から構成される．これらのデータを扱うとき，とりうる値が少ない属性はその属性における値の持つ意味（重要度など）を理解したうえで，値を変換する必要がある．しかし，データを解析し，各属性に重要度を定めることは難しい問題である．したがって，本論文では 2 値，または，3 値の属性を除外したデータを使用する．このとき収入データは 14 属性，キノコデータは 14 属性，森データは 10 属性，そして US データは 51 属性を持つ．

5.1.2 クラスタリング結果の評価基準

入力データが Classification に関するデータであり，データ要素の分類がクラスラベルとしてすでにラベル付けされているとき，出力結果の質を評価する方法が提案されている．4 つのベンチマークデータのうち，収入データ，キノコデータが Classification に関するデータである．

クラスタリング結果の評価基準として Classification エラー E_C，クラスタ平均 Precision P_{avg}，そしてクラスタ平均 Recall R_{avg} を用いる．クラスタリングアルゴリズムが要素数 $s_1, \ldots, s_k$ の k 個のクラスタを出力し，それぞれのクラスタの最多共通クラスのラベルが $L_1, \ldots, L_k$ とする．また，$s_1, \ldots, s_k$ の各クラスタの最多共通ラベルを持つ要素数が $m_1, \ldots, m_k$ であり，入力データにおいてラベル L_i のデータ要素の総数を N_{L_i} としたとき，E_C，P_{avg}，そして R_{avg} は以下の式によって定義される．

$$E_C = \frac{\sum_{i=1}^{k}(s_i - m_i)}{\sum_{i=1}^{k} s_i} = \frac{\sum_{i=1}^{k}(s_i - m_i)}{N}$$

$$P_{avg} = \frac{\sum_{i=1}^{k} \frac{m_i}{s_i}}{k}$$

$$R_{avg} = \frac{\sum_{i=1}^{k} \frac{m_i}{N_{L_i}}}{k}$$

E_C は入力データ要素全体に対して誤って分類された要素数の割合である．P_{avg} は各クラスタが正確に 1 つのラベルだけを集められているかの評価であり，各クラスタの正確さの平均値である．R_{avg} は同じラベルのデータ要素を 1 つのクラスタに集められているかを評価する値である．E_C は 0 に近いほど，P_{avg} と R_{avg} は 1 に近いほど高精度な出力と評価できる．以下のすべての実験において FlexDice と OptiGrid の入力パラメータは準備的な実験において E_C，P_{avg} と R_{avg} に関して高精度なクラスタリング結果を出力可能な値を使用した．

5.2 評 価 実 験

5.2.1 項において FlexDice と従来手法のクラスタリング精度について比較し，5.2.2 項において FlexDice と OptiGrid の計算時間について比較する．

5.2.1 従来手法との精度の比較

クラスタリング精度の比較には Classification に関するデータである収入データとキノコデータを用いる．

収入データは 14 属性を持つ 32,561 データ要素から構成される Classification に関するベンチマークデータである．収入データの各要素は人を表し，クラスラベルとして “収入が $50,000 以上” または “収入が $50,000 より少ない” を持っている．入力の時点ではこの情報を除き，クラスタリング後に情報に基づいて E_C，P_{avg}，そして R_{avg} を求める．

文献 3) の Clustering Aggregation のサンプリングを用いた Agglomerative 手法を収入データに適用したとき，サンプル数が 300 から 5,000 の間においてほぼ等しい出力を得た．サンプル数を 300 より小さくすると妥当なクラスタリング結果が出力されなかった．ここで，サンプル数によるクラスタリング結果は 100 から 5,000 の間で 9 通り調べている．Agglomerative 手法と OptiGrid と FlexDice の精度と計算時間につ

表 1 収入データに関する従来手法と FlexDice の精度と計算時間
Table 1 Accuracy and processing time for the census data by Agglomerative, OptiGrid, and FlexDice.

	E_C	P_{avg}	R_{avg}	Time [s]
Agglomerative	0.21	0.87	0.003	942
OptiGrid	0.21	0.82	0.029	9.29
FlexDice	0.21	0.86	0.011	2.81

表 2 キノコデータに関する従来手法と FlexDice の精度と計算時間
Table 2 Accuracy and processing time for the mushroom data by Agglomerative, OptiGrid and FlexDice.

	E_C	P_{avg}	R_{avg}	Time [s]
Agglomerative	0.10	0.89	0.22	76.4
OptiGrid	0.048	0.96	0.025	0.051
FlexDice	0.025	0.99	0.029	0.062

いての比較を**表 1** に示す．Agglomerative 手法のクラスタリング結果はサンプル数が 300 から 5,000 の間はほぼ等しい精度を保っているため，一例として最も高速にクラスタリングできたサンプル数が 300 での結果を示した．

表 1 より，収入データに関して，3 手法の E_C はすべて同じ値であった．P_{avg} は Agglomerative 手法が最も良い結果であったが，FlexDice とほぼ等しい値であった．R_{avg} は OptiGrid が最も良い結果であり，FlexDice よりも 2 倍以上良い値であった．計算時間に関しては Agglomerative 手法が極端に遅かった．

キノコデータは 22 属性を持つ 8,124 データ要素から構成される Classification に関するベンチマークデータである．データ要素はキノコを表し，属性はキノコの形状，色，においなどを表す．各データ要素はクラスラベルとして “毒入りである”（poisonous），または “食用である”（edible）が記されている．クラスラベルの属性を除いた入力データをクラスタリングした後で，クラスラベル属性に基づいて E_C を求める．

Clustering Aggregation のサンプリングを用いた Agglomerative 手法をキノコデータに適用したとき，サンプル数が 200 から 2,000 の間においてほぼ等しい出力を得た．サンプル数を 200 より小さくすると妥当な結果が出力されなかった．ここで，サンプル数によるクラスタリング結果は 100 から 2,000 の間で 8 通り調べている．Agglomerative 手法と OptiGrid と FlexDice の精度についての比較を**表 2** に示す．Agglomerative 手法のクラスタリング結果はサンプル数が 200 から 2,000 の間はほぼ等しい精度を保っているため，一例として最も高速にクラスタリングできたサンプル数が 200 での結果を示した．

キノコデータに関して，FlexDice で E_C と P_{avg} の精度に関する評価関数において最も高精度な結果が得られた．R_{avg} に関してはクラスタ数を多く作成しているため Agglomerative 手法よりも悪い結果となった．計算時間に関してはキノコデータにおいても Agglomerative 手法が極端に遅かった．

表 1，2 の結果から，OptiGrid と FlexDice は E_C と P_{avg} に重きを置いたパラメータ設定であるとき，Agglomerative 手法にほぼ等しい精度，または，それ以上の精度を持つ結果を出力できた．Agglomerative 手法の R_{avg} の値は入力データによってばらつきがあり，Agglomerative 手法が他の 2 手法に比べて 10 倍近く良い結果を出力できる例があった．アルゴリズムの精度は入力データに依存していることが分かる．FlexDice は同じラベルを持つ要素を 1 つのクラスタに集めることが得意ではないが，各クラスタ内に 1 つのラベルだけを集めることについては他の手法と同程度かそれ以上の精度を達成できた．以上の FlexDice の結果はノイズクラスタを 1 つのクラスタとした精度の評価となっている．ノイズを検出したときの評価が十分でないため，その評価方法は今後の課題としている．

5.2.2 従来手法との計算時間の比較

5.2.1 項で簡単に示したように，Clustering Aggregation を用いた Agglomerative 手法の計算時間は FlexDice と OptiGrid の計算時間に比べて非常に遅い．したがって，計算時間に関して本項においては従来手法の中で最も高速に処理できる手法の 1 つである OptiGrid と FlexDice を森データと US データを用いて比較する．

FlexDice と OptiGrid の要素数と計算時間の関係を調べるために，10 属性を含む森データを使用した．FlexDice と OptiGrid の要素数と計算時間の関係を**図 3** に示す．入力データ要素数の変化は，入力データセットからランダムに抽出する要素数を変化させることで実現した．10 万データ要素から 58 万データ要素までの各出力結果は指定した要素数の 10 種類の入力データに関する結果の平均である．

図 3 から，FlexDice は要素数に大きな影響を受けず，要素数に対して計算時間はなだらかに上昇していることが分かる．一方，OptiGrid は FlexDice に比べて急激に上昇していることが分かる．FlexDice が要素数の多い 30 万から 58 万の間で OptiGrid よりも高速であったのは，データ要素を多く含んだ密セルを作成することにより，データ要素の再走査が少ないことが 1 つの要因であると考えられる．

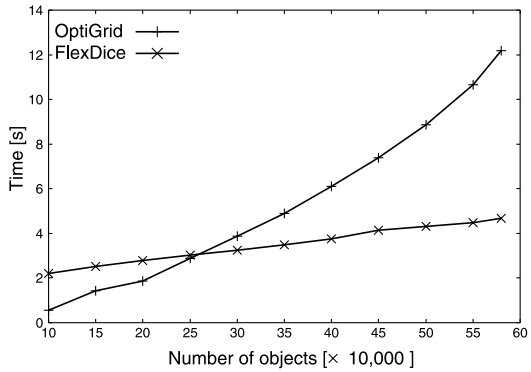

図 3 森データに対する FlexDice と OptiGrid の要素数と計算時間の関係
Fig. 3 Relation between the number of objects and Time for the Forest cover type data set.

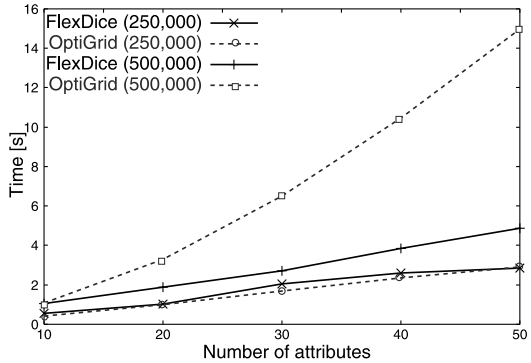

図 4 2 種類の US データ（25 万データ要素，50 万データ要素）に対する FlexDice と OptiGrid の属性数と計算時間の関係
Fig. 4 Relation between the number of attributes and Time for the US census data set.

FlexDice と OptiGrid の属性数と計算時間の関係を調べるために，US データから抽出した 25 万データ要素を含むデータと 50 万データ要素を含むデータを使用した．FlexDice と OptiGrid における属性数と計算時間の関係を**図 4** に示す．入力データの属性数 d' の変化は，入力データの d 属性（$d \geq d'$）からランダムに d' 個だけ属性を選び，d' 属性の部分データを構成することで実現した．図 4 での 10 属性から 50 属性までの各計算時間は，指定した属性数 d' において，10 種類の異なる属性選択パターンを持つ部分データに関する結果の平均である．

図 4 から，入力データの属性数が 10 属性であると FlexDice と OptiGrid の計算時間は要素数に関係なくほぼ等しく，要素数が 25 万程度であると FlexDice と OptiGrid の計算時間はほぼ等しいことが分かる．要素数が 50 万であるとき，FlexDice では OptiGrid に比べて属性数の増加に対し計算時間がなだらかに上昇し，かつ，高速であることが分かる．

以上の結果から，FlexDice は従来手法で最も高速な手法の 1 つである OptiGrid よりも高次元な大規模データに対して高速に処理できることが分かる．

6. おわりに

本論文では高次元な大規模データに対して対話的に処理可能なクラスタリング手法 FlexDice を提案し，FlexDice をベンチマークデータを用いたシミュレーション実験で評価した．

シミュレーション実験において，FlexDice は近年に発表された高精度にクラスタリング可能な Clustering Aggregation とほぼ同等な精度のクラスタリング結果を出力できた．計算時間に関して，高次元な大規模データに対して FlexDice は従来手法の中で最も高速な手法の 1 つである OptiGrid よりも高速にクラスタリングできることを示した．低次元な要素数の少ない入力データに対して FlexDice よりも OptiGrid のほうが高速に処理できているが，どちらも高速であり対話的な応答が可能である．

FlexDice はクラスタ数の指定などの直感的に設定しにくい入力パラメータの設定を必要としないが，クラスタリング結果に影響する入力パラメータが 4 つ存在する．この 4 つの入力パラメータの変化に応じた出力結果の変化を解析することやクラスタリング結果のクラスタの特徴を抽出する後処理が今後の課題である．

参 考 文 献

1) Ankerst, M., Breunig, M.M., Kriegel, H.-P. and Sander, J.: OPTICS: Ordering points to identify the clustering structure, *Proc. 1999 ACM-SIGMOD Int. Conf. on Management of Data* (*SIGMOD '99*), pp.49–60 (1999).
2) Ester, M., Kriegel, H.-P., Sander, J. and Xu, X.: A density-based algorithm for discovering clusters in large spatial Databases with Noise, *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining* (*KDD '96*), pp.226–231 (1996).
3) Gionis, A., Mannila, H. and Tsaparas, P.: Clustering aggregation, *Proc. 2005 IEEE Int. Conf. on Data Engineering* (*ICDE '05*), pp.441–352 (2005).
4) Guha, S., Rastogi, R. and Shim, K.: CURE: An efficient clustering algorithm for large database, *Proc. 1998 ACM-SIGMOD Int. Conf. on Management of Data* (*SIGMOD '98*), pp.73–84 (1998).
5) Guha, S., Rastogi, R. and Shim, K.: ROCK: A Robust Clustering Algorithm for Categorical

Attributes, *Information Systems*, Vol.25, No.5, pp.345–366 (2000).
6) Hellerstein, J.M., et al.: Interactive data analysis: the control project, *IEEE Computer*, Vol.32, No.8, pp.51–59 (1999).
7) Han, J. and Kamber, M.: *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Francisco (2001).
8) Hinneburg, A. and Keim, D.A.: Optimal Grid-Clustering: Towards breaking the curse of dimensionality in high-dimensional clustering, *Proc. 25th Int. Conf. on Very Large Data Bases* (*VLDB '99*), pp.506–517 (1999).
9) Hinneburg, A. and Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise, *Proc. 1998 Int. Conf. Knowledge Discovery and Data Mining* (*KDD '98*), pp.58–65 (1998).
10) Milenova, B.L. and Campos, M.M.: O-Cluter: Scalable clutering of large high dimensional data sets, *Proc. 2002 IEEE Int. Conf. on Data Mining* (*ICDM '02*), pp.290–297 (2002).
11) Sheikholeslami, G., Chatterjee, S. and Zhang, A.: WaveCluster: A multi-resolution clustering approach for very large spatial databases, *Proc. 24th Int. Conf. on Very Large Data Bases* (*VLDB '98*), pp.289–304 (1998).
12) Wang, X., Rostoker, C. and Hamilton, H.J.: Density-based spatial clustering in the presence of obstacles and facilitators, *Proc. 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases* (*PKDD '04*), pp.446–458 (2004).
13) Wang, W., Yang, J. and Muntz, R.: STING: A statistical information grid approach to spatial data mining, *Proc. 1997 Int. Conf. Very Large Data Bases* (*VLDB '97*), pp.186–195 (1997).
14) Zhang, T., Ramakrishnan, R. and Livny, M.: BIRCH: An efficient data clustering method for very large databases, *Proc. 1996 ACM-SIGMOD Int. Conf. on Management of Data* (*SIGMOD '96*), pp.103–114 (1996).
15) The University of California, Irvine Knowledge Discovery in Databases Archive: The insurance company benchmark (COIL 2000). http://kdd.ics.uci.edu/

(平成 17 年 6 月 21 日受付)
(平成 17 年 10 月 5 日採録)

(担当編集委員 片山 紀生)

中村 朋健（学生会員）

2002 年広島市立大学情報科学部卒業．2004 年同大学大学院情報科学研究科博士前期課程修了．現在，同大学院情報科学研究科博士後期課程在学中．

上土井陽子

1989 年広島工業大学電子工学科卒業．1991 年広島大学大学院工学研究科博士課程前期修了．1994 年同大学院工学研究科博士課程後期修了．博士（工学）．1994 年より広島市立大学情報科学部助手．

若林 真一（正会員）

1979 年広島大学工学部電気工学科卒業．1984 年同大学大学院工学研究科博士課程後期修了．同年日本アイ・ビー・エム（株）入社．東京基礎研究所副主任研究員．1988 年広島大学工学部助教授．2003 年より広島市立大学情報科学部教授．工学博士．主として，VLSI CAD，VLSI 設計，組合せ最適化，遺伝的アルゴリズムに関する研究に従事．電子情報通信学会，IEEE，ACM 各会員．

吉田 典可（正会員）

1955 年九州大学工学部通信工学科卒業．1956 年同大学工学部通信工学科助手．同大学工学部電子工学科講師，助教授を経て，1969 年広島大学工学部電子工学科教授（電子回路工学）．1995 年同大学退官，同年広島市立大学情報科学部情報工学科教授（論理回路学）．2003 年同大学退職．工学博士（九州大学）．この間，電子回路とディジタルシステム，論理回路とそのシステム，コンピュータハードウェア，ネットワークシステム等の教育研究に従事．