内積縮退MC：類似行の検出と類似列の検出を組み合わせたマトリクスクラスタリングアルゴリズム

上原子 正利[†] 小柳 滋[†]

本論文は，2 値行列から密な部分行列を発見する操作であるマトリクスクラスタリングの新しいアルゴリズム，内積縮退 MC を提案するものである．このアルゴリズムは，行列中の 1 つの 1 要素（基準要素）を指定されることで，その要素の列（基準列）と多くの 1 要素を共有する列の中に多くの 1 要素を共有する行と，その要素の行（基準行）と多くの 1 要素を共有する行の中に多くの 1 要素を共有する列からなる部分行列を発見する．そのために，部分行列から基準行に類似した行を発見する操作と基準列に類似した列を発見する操作を組み合わせ，徐々に部分行列のサイズを小さくしてゆく．このアルゴリズムは，処理過程で得られる部分行列のサイズが単調減少するため，出力部分行列に対するいくつかの重要な要求を満たすことができる．我々はこのアルゴリズムが密な部分行列を発見できることと十分な速度が得られることを実験で確認した．

Inner-product Degeneration MC: A Matrix-clustering Algorithm Combining Detections of Similar Rows and Similar Columns

Masatoshi Kamiharako[†] and Shigeru Oyanagi[†]

Matrix-clustering is operation on binary matrix to find dense submatrix. We propose a new matrix-clustering algorithm, *inner-product degeneration MC*. By being specified a one-element (base element) in a binary matrix, this algorithm finds an dense submatrix whose rows are similar to the base element's row (base row) by the columns those are similar to the base element's column (base column), and the columns of the submatrix are similar to the base column by the rows those are similar to the base row. To find the submatrix, this algorithm combines the operations of finding the rows similar to the base row and finding the columns similar to the base column, and gradually decreases the size of submatrix monotonously. By this monotonicity, this algorithm has ability for meeting some requirements for output submatrix. We confirmed that this algorithm can find dense submatrix and is fast enough for practical use.

1. はじめに

大規模なデータセットを対象とするデータマイニングの問題には，データセットを行列として表現できるものが多い．そのような行列上で行われる操作の代表的なものに，関連する行の発見がある．たとえば文書処理では，文書を行，文書中に出現する単語を列，文書中の単語の出現回数を要素とした行列を用い，行ベクトル間の内積などを関連性の尺度として関連文書を求めることが行われる．

行列上での関連する行の発見だけでなく，関連する行と列を同時に発見する操作がマトリクスクラスタリング[1),2)]（matrix-clustering，以下 MC とする）である．MC は 2 値行列を対象にし，1 要素が密に存在する部分行列を発見することでこれを実現する（**図 1**）．

MC の概念を示した文献 1) は，十分な速度を持つ MC アルゴリズムも同時に提案している．しかし，その後の我々の実験において，そのアルゴリズムは出力部分行列に対するいくつかの一般的な要求を満たさないことが明らかとなった．そのため我々は，それらの要求を満たす新しい MC アルゴリズム，内積縮退 MC を開発した．本論文はこのアルゴリズムについて述べるものである．

以下では，2 章で MC の特徴と MC アルゴリズムの出力に対するいくつかの一般的な要求を述べ，3 章で既存の MC アルゴリズムであるピンポン法の出力がそれらの要求を必ずしも満たさないことを確認する．4 章では，2 章の要求を満たす MC アルゴリズム，内積縮退 MC を定義し，5 章でその性質と処理速度を実

† 立命館大学
Ritsumeikan University

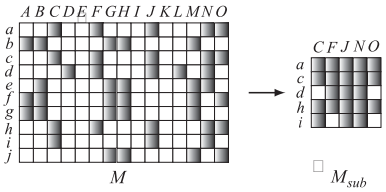

図 1 マトリクスクラスタリング
Fig. 1 Matrix-clustering.

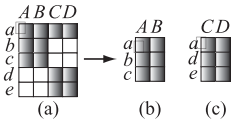

図 2 行と列の類似性を同時に考える効果
Fig. 2 The effect of simultaneous consideration of rows' similarity and columns' similarity.

験によって確認する．6 章と 7 章で高速化手法を導入し，8 章で今後解決すべき点について議論する．

用語と表記法について

以下では，複数の行の間で共通して 1 となる列をそれらの行の共通要素と呼ぶ．列についても同様である．文中での行列の表記法は，行の集合と列の集合の間に × を置くか，行数と列数の間に × を置くものとする．行列の 1 つの要素は，行の名前と列の名前を並べ，() で括ることで表記する．

2. MC の特徴と出力に対する要求

2.1 MC の特徴

MC とは，0 要素と 1 要素からなる 2 値行列から，1 要素の割合の高い部分行列，すなわち密な部分行列を取り出す操作である☆．この操作の目的は，関連する行と列の集合を同時に求めることである．対象が 2 値行列であるため，行と列が関連しているとは，それらの行がそれらの列で類似している，すなわちそれらの行がそれらの列に多くの 1 要素を持ち，同時にそれらの列もそれらの行で類似していることを意味する．**図 2** は，2 値行列中の類似行を求める操作において，行のみを考慮する場合と列も同時に考慮する場合の違いを示している．(a) において行 a の類似行を探す場合，共通要素数を類似性の尺度とすると，b，c，d，e はいずれも a と 2 つの共通要素を持つため，行のみでは b，c，d，e を区別できない．それに対して MC は，2 つの部分行列 (b)，(c) によって b，c と d，e を区別する．

文献 1) では，MC を行うアルゴリズムとして行・列置換法とピンポン法の 2 つが示され，後者が前者を上回る性能を持つことが示された．これらのアルゴリズムは，対象となる行列が同じでも異なる部分行列を出力し，同じアルゴリズムを異なるパラメータで実行した場合も異なる部分行列を出力する．このように出力が異なる原因は，どのような部分行列を密であるかと定義する基準が一意でないことと，1 つの基準の下でもそれを満たす部分行列が一般に 1 つの行列中に複数存在することにある．たとえば，図 1 の行列 M から密度 0.8 以上で面積 20 以上の部分行列を取り出すなら，密度 0.84 である図中の M_{sub} のほかに，密度 0.88 の $\{b, e, f, g, j\} \times \{A, B, G, H, M\}$ などが可能である．このため，MC を行う場合は出力部分行列に対する要求を明確にする必要がある．

2.2 出力部分行列に対する要求

文献 1) で扱われた出力部分行列に対する要求は，指定された最小密度，最小行数，最小列数，最小面積を満たすというものである．密度は部分行列の行と列の関連の強さを示すため，最小密度は関連の強さに対する制約である．最小行数・最小列数・最小面積は，関連する行・列の集合の大きさに対する制約である．

なお，これらの要求は同時に満たされることが保証されない．たとえば MC アルゴリズムに対して，図 1 の M から面積 100 以上，密度 0.5 以上の部分行列を見つけるよう指示したとする．しかし，M の 1 要素の総数は 48 であるため，面積 100 の部分行列にそれらをすべて収めたとしても密度は 0.5 に及ばず，この 2 つの要求は同時に達成できない．このような要求間の矛盾は一般にアルゴリズムの実行前には判定できない．問題によってはこのような場合に出力がないことが許されるが，何らかの出力を必要とする場合は要求間に優先順位を設定する必要がある．

これらの要求のほかに，出力部分行列に対する一般的に重要な要求がいくつか存在することに我々は着目した．それらは以下のようなものである．

2.2.1 孤立した行・列を含まない

図 3 (a) は，他の行と共通要素を持たない行 d と他の列と共通要素を持たない列 F を含んでいる．(b) には他の行と共通要素を持たない行の一種である空行が存在する．部分行列が関連した行と列の集合を表すな

☆ MC は一般的な意味での「クラスタリング」，すなわち，与えられた集合から類似した要素の部分集合を列挙する操作ではない．クラスタ列挙の操作は MC と独立の概念である．なお，本論文での MC の定義は文献 1) のものと異なる．文献 1) では MC を，指定された面積以上で指定された密度以上の部分行列を取り出すことと定義している．しかし，この定義では面積と密度の条件が同時に満たされなくてはならないが，後に見るようにこの 2 つの条件が同時に満たされることは保証されない．この点を考慮して，本論文では文献 1) の定義を緩めたものを採用する．

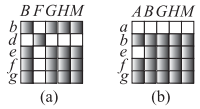

図 3 孤立した行・列の存在する行列
Fig. 3 Matricies containing isolated rows and columns.

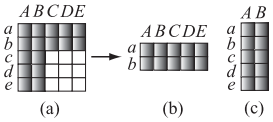

図 4 横長部分行列と縦長部分行列
Fig. 4 A wide submatrix and a tall submatrix.

ら，孤立した行と列を含むことは許されない．

2.2.2 行数と列数の比が指定された値に近い

図 4 (a) から密度 1 で可能な限り大きい部分行列を取り出すと，横長の部分行列 (b) と縦長の部分行列 (c) の 2 つが得られる．横長の部分行列は列の間よりも行の間の共通要素数を多くすることを優先した結果であり，縦長の部分行列は行よりも列を優先した結果である．たとえば，行を文書，列をそのキーワードとして類似文書を探すことを考えると，縦長の部分行列が表すものは，頻繁に使われるキーワード，つまり大きな分野を示すキーワードで一致する文書の集合であり，横長の部分行列が表すものは，使用される頻度の低いキーワード，つまりより細分化された分野を示すキーワードまで一致する文書の集合である．このような区別をするためには，MC アルゴリズムが出力部分行列の行数と列数の比を制御する必要がある．

2.2.3 注目すべき行と列を同時に含む

図 2 (a) において，行 a に注目したうえに (b) と (c) を区別して取り出すなら，注目すべき列も指定する必要がある．類似文書検索で考えると，文書 a は複数の分野にまたがったものであるため，どのキーワードに注目するかによって得られる類似文書の分野が異なる．このような区別をするためには，MC アルゴリズムが注目すべき行だけでなく列も明示的に扱わなければならない．

2.2.4 特定の行・列が 0 要素を含む

ある 2 値行列を特徴づける概念の 1 つに，0 要素と 1 要素の存在する位置がある．この特徴に関する要求を考える．MC の出力部分行列は 1 要素が密に存在する行列であるため，相対的に少ない 0 要素の位置にのみ注目すると，この要求は 0 要素の存在する位置に関するものになる．そのような要求の 1 つとして，特定の行・列が 0 要素を含まなければならないというものを考える．この要求が現れる例にはリコメンダシステム[3] があり，ここではオンライン書店でユーザに書籍を推薦することを考える．

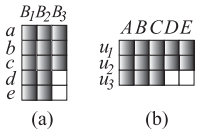

図 5 0 要素の位置が限定された行列
Fig. 5 Matricies containing zero-elements only in particular position.

ユーザを行，書籍を列とし，ユーザが購入した書籍に該当する要素を 1 とした行列を用いると，これから得られる密な部分行列は関連の強いユーザと書籍を表す．このとき，図 1 の M に対して MC を用いてユーザ a に推薦を行うなら，M_{sub} を出力してはいけない．なぜなら，M_{sub} での行 a は 0 要素を含まないため，この部分行列の列に相当する書籍はすべてユーザ a にすでに買われたものだからである．このような場合，MC アルゴリズムは出力部分行列の特定の行に 0 要素を含むことを保証できなければならない．列についても同様である．

2.2.5 特定の行・列にのみ 0 要素が存在する

0 要素の存在する位置に関する他の要求として，0 要素が特定の行・列のみに存在しなければならないというものがあげられる．この要求の現れる例として再びオンライン書店を用いる．前述の例での部分行列は，関連の強いユーザと書籍を示すだけであり，ユーザ間あるいは書籍間の差異を考慮していなかった．それに対して，ユーザ間あるいは書籍間の差異を考慮すると，この要求が現れる．書籍 B_1，B_2 の両方を買ったユーザが高い割合で書籍 B_3 も買っている場合，B_1，B_2，B_3 は類似した書籍と見なせ，B_1，B_2 と B_3 には役割の違いがある．ユーザに注目した場合も同様に，ユーザ u_1，u_2 の両方が買った書籍が高い割合でユーザ u_3 にも買われている場合，u_1，u_2，u_3 は類似したユーザと見なせ，u_1，u_2 と u_3 の間には役割の違いがある．これらの場合の部分行列を示すのが**図 5** である．(a) の $a, \cdots, e$ は B_1 と B_2 が 1 要素を同時に持つ行のすべてであり，(b) の $A, \cdots, E$ も u_1 と u_2 について同様である．B_1，B_2 と B_3 および u_1，u_2 と u_3 の役割の違いは，0 要素の存在する位置によって表現される．

図 5 (a) の B_1，B_2，B_3 の間の関係は相関規則[4] と呼ばれ，$\{B_1, B_2\} \Rightarrow \{B_3\}$ と表記される．相関規則はリコメンダシステムでよく用いられる[5] ため，2 値行列からの推薦の一般的な概念ととらえることができる．以下では相関規則を示す部分行列を相関規則部分

行列と呼ぶ．

相関規則部分行列は 2 つの値で特徴づけられる．1 つはサポートで，これは図 5 (a) の密度 1 の部分行列 $\{a,b,c\}\times\{B_1,B_2,B_3\}$ の行数である．このときの列集合は頻出アイテムセットと呼ばれるため，以下ではこのような部分行列を頻出アイテムセット部分行列と呼ぶ．頻出アイテムセット部分行列として許容できる最小の行数は最小サポートと呼ばれる．図 5 (a) が頻出アイテムセット部分行列となるのは，最小サポートが 3 以下の場合である．

もう 1 つの値はコンフィデンスである．これは，相関規則部分行列の行数に対する頻出アイテムセット部分行列の行数の割合である．この値に対して相関規則が許容できる下限が最小コンフィデンスである．図 5 (a) のコンフィデンスは 0.6 であり，最小コンフィデンスが 0.6 以下なら $\{B_1,B_2\}\Rightarrow\{B_3\}$ は相関規則になる．なお，最小サポートは最小行数であり，最小コンフィデンスは密度の下限となる☆．

相関規則は一般にアイテム間の関係であるため，ユーザを行，アイテムを列とした場合，列の関係のみを示す．しかし，行と列を入れ替えることでユーザの間の関係を得ることもできる．図 5 (b) がその例である．

2.2.6 本論文で扱う出力部分行列に対する要求

以上の出力部分行列に対する要求を整理する．

(1) 最小密度，最小行数，最小列数，最小面積の条件を満たす．
(2) 孤立した行・列を含まない．
(3) 行数と列数の比が指定された値に近い．
(4) 注目すべき行と列を同時に含む．
(5) 特定の行・列が 0 要素を含む．
(6) 特定の行・列にのみ 0 要素が存在する．

これらのうち，(6) は既存の相関規則発見アルゴリズム[6] で満たされるため，以下では (1) から (5) のみを考慮する．次章では，これらの要求が既存の MC アルゴリズムであるピンポン法で満たされないことを見る．

☆ これは以下のように示される．$I_1\Rightarrow I_2$ において，$|I_1|$ がとりうる最小の値である 1 に近づき，$|I_2|$ が大きくなると，コンフィデンスが 1 でなければ，部分行列の密度は小さくなりコンフィデンスに近づくが，けっして等しくならない．たとえば図 5 (a) では，$\Rightarrow$ の右の列集合 $\{B_3\}$ の要素数が大きくなれば密度が小さくなり，その部分行列のコンフィンデンスに近づく．しかし，$\Rightarrow$ の左の列集合が存在するため，けっしてコンフィデンスに等しくならない．そして，コンフィデンスは最小コンフィデンスより小さくならない．

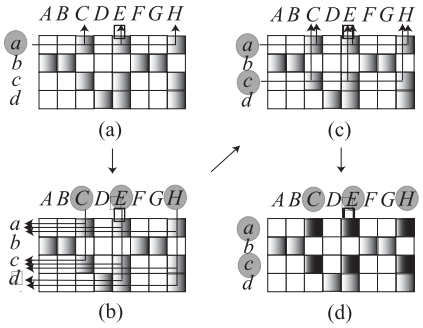

図 6 ピンポン法
Fig. 6 Ping-pong algorithm.

3. ピンポン法—既存の MC アルゴリズム

ピンポン法は文献 1) で提案された MC アルゴリズムである．このアルゴリズムは，開始時に行または列を指定されると，行からの列の選択と列からの行の選択を繰り返すことで密な部分行列を発見する．終了条件は，ある時点で選択された行集合または列集合が，その直前に選択された行集合または列集合と等しいことである．開始時に選択される行・列を初期行・初期列，行・列の選択を活性化と呼ぶ．

図 6 は行 a を初期行として活性化したときのピンポン法の実行例である．(a) では a の中で 1 要素を持つ列 C，E，H が活性化されている．(b) では C，E，H の中に 1 要素を持つ行 a，c，d を活性化の候補とし，これらをその 1 要素数によって a，c と d に分ける．このとき，もし a，c，d のすべてを活性化すると $\{a,c,d\}\times\{C,E,H\}$ が作られ，a，c のみなら $\{a,c\}\times\{C,E,H\}$ が作られる．行集合の選択基準として，その行集合と直前に選択された列集合から構成される部分行列の密度を用いるとし，密度 1 を指定すると，a，c のみが活性化される．(c) では C，E，H すべてが活性化されるが，これらはすでに (a) で活性化されているため，さらに行を活性化しても同じ過程が繰り返される．よって，アルゴリズムは停止し，(d) に示される部分行列 $\{a,c\}\times\{C,E,H\}$ を出力する．

ピンポン法の出力部分行列は，終了条件より，その行はその列を活性化するものであり，同時にその列はその行を活性化するものである．一般に，ある行集合が活性化する列集合は必ずしも元の行集合を活性化しない，すなわち，活性化関係は非対称であるため，行からピンポン法を実行した場合，初期行が処理の途中で活性化された行集合から外れることがある．この現象を我々は行ドリフトと呼んでいる．列についても同様である．例として，**図 7** の行列で a を初期行に指

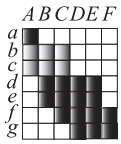

図 7 行ドリフトの発生する行列
Fig. 7 A matrix which raises row-drift.

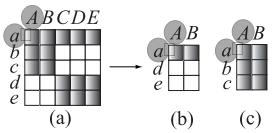

図 8 行と列で独立した類似検出と相互に依存した検出の違い
Fig. 8 The difference between independent and mutual dependent similarities detections of rows and columns.

定し，密度 0.75 でピンポン法を実行すると，11 回の活性化の末に $\{d, e, f, g\} \times \{B, C, D, E, F\}$ で停止する．a は 4 回目の活性化で部分行列から外れ，それ以降も含まれず，前章の要求 (4) は満たされない．

この問題に対処するため我々は，ある時点で活性化集合から外れた初期行を強制的にその集合に追加するという方法をとった．図 7 の行ドリフトはこれで解決し，ピンポン法は $\{a, b, c\} \times \{A, B, C\}$ を出力する．しかし，実際のデータを用いた場合には，この方法を用いても図 3 (b) のような初期行に 1 要素を含まない部分行列が出力されることがある．この現象の原因は，強制的に変更された活性化集合において，初期行以外の行が共通して多くの 1 要素を含む列と初期行が 1 要素を含む列の一致が保証されないことにある．

ピンポン法の出力である行と列が互いに活性化しあう部分行列を安定な部分行列と呼ぶと，初期行・列は注目すべき行・列と見なせるものではなく，安定な部分行列を探すための手がかりと見なすべきものである．そのため，ピンポン法は要求 (4) を満たさない．初期行強制追加時の空行発生により要求 (2) も満たさない．また，ピンポン法は初期行・列から到達可能である安定な部分行列を 1 つだけ発見するものであり，その行列が望ましい行数列数比を持っていることは保証されないため，要求 (3) も満たさない．さらに，安定な部分行列が指定された行に 0 要素を持つことも保証されないため，要求 (5) も満たさない．

これらの問題に対処するため，我々は新しい MC アルゴリズム，内積縮退 MC を開発した．

4. 内積縮退 MC

内積縮退 MC は 2 章で述べた要求を満たす MC アルゴリズムである．このアルゴリズムは，開始時に 1 つの 1 要素を指定されることで，それを元に疎でサイズの大きい初期部分行列を構成し，徐々にその部分行列を小さくすることで，最終的に密な部分行列を出力する．開始時に指定される 1 要素を基準要素，その行を基準行，列を基準列と呼ぶ．得られる部分行列の行は，基準行と共通要素を多く持つ行であり，列は基準列と共通要素を多く持つ列である．ただし，行どうしが共通要素を多く持つのは，すべての列の中ではなく，部分行列の列の中においてである．列についても同様である．このような部分行列を得るために，このアルゴリズムは類似行の検出と類似列の検出を合わせて考える．

行と列の類似検出を合わせて考える効果を示すのが**図 8** である．(a) からの類似行と類似列の検出を，行と列で独立に行うとする．行 a に最も似た行は d，e である．列 A に最も似た列は B である．それらを組み合わせて得られる部分行列は (b) の $\{a, d, e\} \times \{A, B\}$ となるが，これは 2.2.6 項の要求 (2) を満たさない．それに対して，(c) は望ましい部分行列の例である．内積縮退 MC は (b) を避け (c) を得ることを可能にする．

4.1 アルゴリズムの定義

このアルゴリズムの基本操作は，初期部分行列構成と縮退操作である．初期部分行列構成は，基準行と共通要素を持つ行と，基準列と共通要素を持つ列からなる部分行列を構成する操作である．縮退操作は内積計算と削除操作からなり，行方向と列方向の 2 つがある．行方向の内積計算は，部分行列の各行をベクトルと見なし，基準行との内積を求めることで，それらの行にスコアを付ける操作である．対象が 2 値行列であるため，2 つの行の内積はそれらの行の共通要素数に等しい．削除操作は，スコアの最も小さい行を部分行列から排除するものである．列についても同様である．

図 9 の行列 M で基準要素を (a, F) としたとき，内積縮退 MC の処理過程は次のようになる．まず，初期部分行列 M_0 を構成する．図の *init* がこの操作を示す．次に行縮退を行うとする．基準行が a なので，各行の内積は，c，h が 5，d，i が 3，e，f，j が 1 となる．この結果から削除操作を行うと，内積最小の行 e，f，j が取り除かれ，M_1 が作られる．図の r が行縮退を示す．次に列縮退を行うとする．基準列は F なので，内積は J，N が 4，C，O が 3，D，L が 1 となり，D，L が削除され，M_2 が得られる．図の c が列縮退を示す．さらに行縮退を行うと，密度 1 の M_3 が得られる☆．

☆ M_3 に対してさらに行と列の縮退操作を 1 回ずつ適用でき，その結果 $\{a\} \times \{F\}$ が得られる．図 10 はその様子を示す．

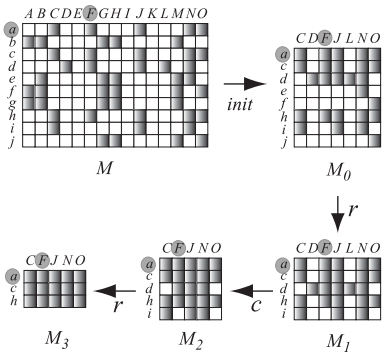

図 9 内積縮退 MC
Fig. 9 Inner-product degeneration MC.

図 9 に示されていない重要な点が 2 つある．1 つは行縮退と列縮退の順序の決定方法である．ここではそれらが交互に行われているが，必ずしもそのようにする必要はない．後に見るように，この順序は出力に影響を与える．もう 1 つは終了条件である．図 9 では密度 1 の M_3 が作られるまで処理が続いているが，何らかの終了条件を設定すれば，M_3 に至る前に処理は停止する．主な終了条件には最小密度がある．図 9 を最小密度 0.8 で実行すれば，密度 0.84 となった M_2 で停止する．これらの点を決定する関数を縮退方向決定関数と呼ぶ．この関数は，現在の部分行列から縮退できる場合はその方向を決定し，何らかの終了条件によって縮退できない場合は停止を出力する．

以上の説明をまとめると次のようになる．

(1) 2 値行列中の 1 要素を 1 つ指定し，その行を r_b，その列を c_b とする．
(2) r_b と c_b を用いて初期部分行列 M_0 を作る．M_0 の行集合を R_0，列集合を C_0 とする．
(3) $i = 0, 1, 2, \ldots$ に関して，終了条件を満たすまで以下を繰り返す：
  (a) 縮退方向決定関数が停止を出力すれば，M_i を返して終了する．それ以外なら，この関数が示す方向を縮退する．以下では行縮退を行う場合を示す．列縮退も同様である．
  (b) M_i の各行に対して，r_b との内積を求める．
  (c) 内積の最も小さい行（複数存在するならすべて）を R_i から取り除いたものを R_{i+1} とする．
  (d) C_i を C_{i+1} とする．
  (e) $R_{i+1} \times C_{i+1}$ を M_{i+1} とする．

初期部分行列の構成方法は一意ではない．行集合の構成方法には，

(1) M の行のうち，r_b と共通要素を持つすべての行の集合 R_0 を作る（この R_0 を R_l とする），
(2) M の行のうち，c_b に 1 要素を持つ行のみからなる集合 R_0 を作る（この R_0 を R_s とする），

という 2 つがある☆．列集合についても同様で，

(1) M の列のうち，c_b と共通要素を持つすべての列の集合 C_0 を作る（この C_0 を C_l とする），
(2) M の列のうち，r_b に 1 要素を持つ列のみからなる集合 C_0 を作る（この C_0 を C_s とする），

の 2 つがある．2 種類の行と 2 種類の列の組合せで，計 4 種類を作ることができる☆☆．これらは目的に応じて使い分けられるべきものである．たとえば，基準行や基準列が 2.2.6 項の要求 (5) を満たす必要があるなら，R_s や C_s は使えない．なぜなら，R_s を使うときは初期部分行列中の基準列が，C_s なら基準行が，すべて 1 要素で占められるからである．なお，$R_s \subseteq R_l$ および $C_s \subseteq C_l$ が成立するため，4 種類のうち，$R_s \times C_s$ が最も小さい部分行列となり，$R_l \times C_l$ が最も大きい部分行列となる．図 9 では，M_0 が $R_l \times C_l$ であり，$R_s \times C_s$ は $\{a, c, d, h\} \times \{C, F, J, N, O\}$ である．

4.2 内積縮退 MC の性質

内積縮退 MC の性質を理解するには，まず，列縮退を一度も行っていないときの類似行検出と，列縮退を限界まで行った後の類似行検出を考え，その後，それらの中間を考える必要がある．列縮退を一度も行っていないときの類似行検出は，行列全体から内積を計算して，その値の大きい行を検出することに等しい．列縮退を限界まで行った後の類似行検出は，基準列に 1 要素を持つ行を検出することに等しい．これらの中間は，ある程度の列縮退を行った後の類似行検出であり，これは基準列とある程度の内積を持つ列にのみ注目して類似行検出を行うことに等しい．同様の議論が類似列検出についても成立する．そのため，内積縮退 MC の縮退過程は，類似度計算のために注目する範囲を段階的に限定していく過程となる．

4.2.1 縮退グラフ

内積縮退 MC の縮退過程は，部分行列を節点，1 回の行縮退と列縮退を枝とした有向グラフで表現することができる．このグラフを縮退グラフと呼ぶ．**図 10** は図 9 の行列 M_0 を始点とする縮退グラフであり，M_0，M_1，M_2，M_3 は図 9 の部分行列を示す．以下では，縮退グラフ中の部分行列とそれに相当する節点を同一視する．図中の r と c はそれぞれ行縮退と列縮退による節点の遷移を示し，以下ではそれぞれ r 遷移，c 遷移と呼ぶ．行列 M に行縮退を施した行列を $r(M)$，

☆ 添字の l は loose を，s は strict をそれぞれ示す．
☆☆ これらを元にさらに多くの種類の初期部分行列を構成する方法を後に見る．

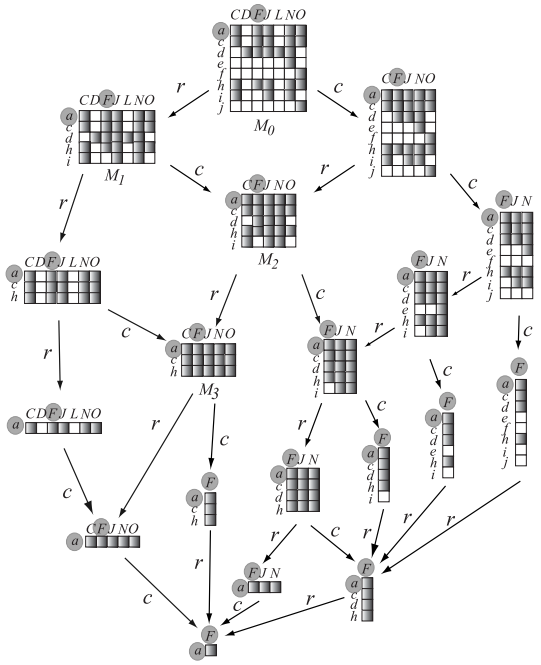

図 10 縮退グラフ
Fig. 10 A degeneration graph.

列縮退を施した行列を $c(M)$ とする．縮退グラフ中の経路を縮退経路と呼ぶ．縮退グラフは必ず，始点が初期部分行列，終端が基準要素である束になる．このように表現すると，内積縮退 MC の処理過程は，縮退グラフ上で望ましい部分行列を探索するグラフ探索ととらえることができる．

縮退グラフ上のグラフ探索アルゴリズムは，縮退方向決定関数に実現される．この関数をどのように構成するかについて，内積縮退 MC は制約を課さない．この点については後に述べる．

4.2.2 縮退グラフの性質

縮退グラフは，2.2.6 項の要求に関係する以下の性質を持つ．

(1) 開始節点以外の任意の節点の行集合は親節点の行集合の部分集合である．列集合も同様である．

(2) 開始節点以外の任意の節点の面積は親節点の面積より小さい．

(3) 内積 0 の行と列を同時に含む節点は縮退グラフ上に存在しない．

(4) 内積 0 の行を含む節点から 1 回の r 遷移を行うと，内積 0 の行も内積 0 の列も含まない節点に至る．行と列を入れ換えても同様である．

(5) 開始節点以外の任意の節点の任意の行で 0 要素を持つ列の集合は，親節点の同一行に 0 要素を持つ列の集合の部分集合である．行と列を入れ換えても同様である．

(6) 初期部分行列が $R_s \times C_s$ のとき，縮退操作の度に部分行列の密度は増加する．

これらのうち，(1)，(2)，(5) は明らかであるため，(3)，(4) および (6) を証明する．

(3) は帰納法により以下のように示される．初期部分行列はその構成方法より，すべての行と列が 0 より大きい内積を持つ．次に，開始節点を含む連結部分グラフ D_{sub} を考え，そのすべての節点が内積 0 の行と列を同時に持たないとする．D_{sub} の任意の端点 n をとる．n が r 遷移可能なら，内積 0 の行を含んでいたとしても r 遷移で削除されるため，$r(n)$ は内積 0 の行を持たない．同様に，$c(n)$ は内積 0 の列を持たない☆．以上より (3) は示された．

(4) は次のように示される．縮退グラフ上で内積 0 の行を含む任意の節点を n とする．(3) より，n は内積 0 の列を含まない．n の行で最小の内積は 0 であり，その行は r 遷移で削除されるため，$r(n)$ は内積 0 の行を含まない．また，この縮退が内積 0 の列を新たに発生させることはない．もし内積 0 の列が新たに発生するなら，削除された行に 1 要素を含む列のうち，基準列と共通要素を持つものが存在することになる．しかし，これはありえない．なぜなら，このような列が存在すれば，削除された行の内積は 0 ではなく，前提に矛盾するからである．以上より (4) は示された．

(6) は次のように示される．行縮退について考えると，$R_s \times C_s$ を初期部分行列とする縮退グラフ上で，任意の部分行列の基準行は 1 要素のみからなる．そのため，その部分行列で内積の最も小さい行は，1 要素数の最も少ない行である．1 要素数の最も少ない行を除けば，残りの行からなる部分行列の密度は高くなる．よって，縮退操作のたびに密度は上昇する．列縮退についても同様である．以上より (6) は示された．

次に，これらの性質と出力部分行列に対する要求の関係を見る．

4.2.3 縮退グラフの性質と出力部分行列に対する要求の関係

内積縮退 MC は，2.2.6 項で述べた要求のうち，注目すべき行と列を同時に扱う能力を基準行と基準列の指定で実現している．この指定の効果を示すのが**図 11** である．ここでは，同一の基準行 a に対して，基準列を A，B，E からとる場合で異なる縮退グラフが構成

☆ $r(n)$ は内積 0 の列を持ちうる．これは，図 10 の M_1 と $r(M_1)$ の間の遷移に見られるように，r 遷移の結果として内積 0 の列が発生する場合があるからである．同様に，c 遷移が可能なら，$c(n)$ は内積 0 の列を持ちうる．

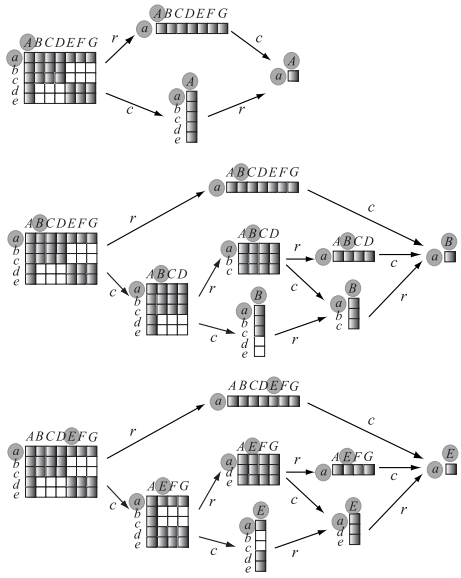

図 11 基準列を変えることによる出力の違い
Fig. 11 Various outputs of various base columns.

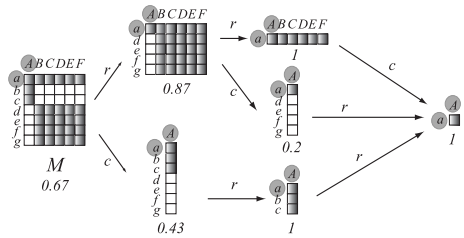

図 12 縮退経路によって密度が減少する例
Fig. 12 An example of density decrease according to degeneration path.

され，その結果得られる部分行列も異なる．その他の要求は，縮退グラフの性質を縮退方向決定関数の探索戦略で利用することによって満たされる．

最小行数・列数の要求は，性質 (1) より，その要求に違反する節点に到達した時点でそれ以降の探索を打ち切り，それまでに展開した節点のみを出力の候補とすることで満たされる．同様に，最小面積の要求は性質 (2) によって，特定の行・列での 0 要素の存在に対する要求は性質 (5) によって，それぞれ満たされる．

孤立した行・列の排除の要求は，内積 0 の行または列を含む節点に到達したとき，縮退方向決定関数がその行または列を排除するように指示することで満たされる．このとき，性質 (4) より，1 回の遷移のみでこの要求を満たす部分行列に到達することが保証される．

最小密度の要求は，初期部分行列の構成方法によって議論が異なる．初期部分行列が $R_s \times C_s$ のときは，性質 (6) より密度が単調増加するため，最小密度要求を満たす節点とそれ以降の節点のみを出力の候補とすることで満たすことができる．しかし，初期部分行列が $R_s \times C_s$ 以外のときは，密度の単調増加が保証されないため，この考え方はできない．**図 12** はそのような場合を示す．図中の M から $c(M)$ および $r(M)$ から $c(r(M))$ への縮退では，密度が大きく減少している．そのため，$R_s \times C_s$ 以外の初期部分行列を用いる場合は，個々の節点について最小密度要求を満たすかどうかを確認する必要がある．また，$c(M)$ と $c(r(M))$ から到達できる密な部分行列は，行が基準行のみ，あるいは列が基準列のみからなる．このような自明な部分行列は必ず縮退グラフ上の存在するため，どのような縮退経路でも必ず最小密度要求を満たすといえるが，このような部分行列は関連する行と列の集合を求めるという MC の目的に適さない．そのため，自明な部分行列に到達しなければ密度要求を満たさない経路は，実質的に密度要求を満たさないと考えるべきである．

行数と列数の比に対する要求は，性質 (1) を利用し，望ましい比から遠ざかる方向の節点の探索を避けることで満たすことができる．しかし，任意の縮退経路は行数列数比に関する単調変化を保証しないため，つねに縮退グラフ上で最善の行数列数比を満たす部分行列を出力するには，縮退グラフの全探索を行い，最善の節点を選択する必要がある．

4.2.4 縮退方向決定関数の例

前述のように，内積縮退 MC では縮退方向決定関数の構成方法を定めない．これは，応用によって変わる出力に対する要求や処理時間に対する制約などによって，この関数を構成する方針が変わるためである．

例として，処理時間に対する制約が弱く，出力部分行列に対する要求が厳しい場合を考える．このときの縮退方向決定関数は，縮退グラフを全探索して，最善の部分行列を選択し，その部分行列への経路を順に返すものになるだろう☆．さらに，要求間に矛盾が生じたとき，どの要求を優先して達成し，どの要求を緩めるかの方針が問題によって変わるが，その方針もこの関数に含める必要がある．

反対に，処理時間に対する制約が強く，出力に対する要求が緩い場合を考える．このときの縮退方向決定関数は，可能な限り少数の部分行列のみを構成することで，要求をできるだけ満たす出力をする必要がある．部分行列の構成回数を最も少なくする関数は，現在の

☆ 密部分行列を求めるだけなら，望ましい部分行列を選択した時点でそれを出力すればよい．経路を順に返すのは，前述のアルゴリズムの表記に合わせるためである．

部分行列のみから縮退方向を決定するものである．この関数を用いると，単一の縮退経路上の部分行列しか構成されない．このような関数のうち最も単純だと我々が考えているものは，現在の部分行列の行数と列数の比を，要求として与えられた行数と列数の比と比較し，それらを近づける方向に縮退するものである．すなわち，与えられた比に比べて現在の行が大きければ行を縮退し，列が大きければ列を縮退する．この関数に必要な処理時間は部分行列のサイズによらず一定であり，除算と減算 1 回ずつのみで実現できる．

この単純な縮退方向決定関数は，処理速度の点で望ましいうえに，縮退グラフの単調変化する性質を利用する要求，すなわち最小行数，最小列数，最小面積，0 要素の存在保証を必ず満たすことができる．しかし，最小密度と行数列数比の制御に関しては，必ず満たすことを保証できない．前述の図 12 で問題となったような経路が縮退グラフ上に数多く存在すれば，この単純な縮退方向決定関数では密な部分行列を出力することが難しくなる．しかし，そのような経路の数が少なく，縮退操作のたびに密度が上昇するなら，この関数でも十分な密部分行列発見能力を持つことになる．行数列数比についても同様に，この単純な関数で実際にどの程度制御できるかが分からない．次章では，この単純な関数のこれらの能力について実験的に確認する．

4.3 データ構造と記憶量

内積縮退 MC では，行に対する操作と列に対する操作方法が等しく，頻度も同程度である．そのため，1 つの行列を行から見たものと列から見たものの 2 つのデータ構造を用意することで処理速度を上げることができる．**図 13** は図 11 の行列のデータ構造である．行から見た行列は行エントリ，列から見た行列は列エントリとして表現される．行・列エントリの要素は各行の 1 要素列・行番号の配列であり，ソートされている．行列 $R \times C$ に対して必要な記憶量は，行列内の総 1 要素数を n_e とすると $O(|R|+|C|+2n_e) = O(|R|+|C|+n_e)$ となり，n_e は密度 d を用いて $d|R||C|$ と表せるため，$O(|R|+|C|+d|R||C|)$ となる．MC が扱う行列は一般に疎であるため☆，d はこの値を大幅に下げる．

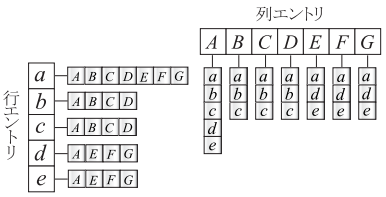

図 13 図 11 の行列のデータ構造
Fig. 13 A data structure for the matrix of Fig. 11.

4.4 データ構造上でのアルゴリズムと計算量

元となる行列の行エントリを `rowEntry`，列エントリを `columnEntry`，基準行を r_b とすると，初期部分行列構成，縮退操作および終了条件判定のアルゴリズムと計算量は次のようになる．

4.4.1 初期部分行列構成

前述のように初期部分行列の構成方法は 1 つではないが，ここでは最も多くの処理時間が必要になる $R_l \times C_l$ について考える．以下は行集合の構成方法である．列についても同様である．

(1) 初期部分行列の行番号を格納する配列 `rows` を用意する．`rows` は空で初期化する．
(2) `rowEntry`[r_b] 中の各要素 c_i に対して `columnEntry`[c_i] の全要素を `rows` にコピーする．
(3) `rows` をソートし，重複要素を削除する．

1 つの行・列の平均 1 要素数をそれぞれ n_r，n_c とすると，(2) のコピーは n_r 回実行され，1 回にコピーされる要素数は n_c である．コピーが終了した時点で，`rows` には $n_r n_c$ 個の要素が格納されている．$n_r n_c = N$ とすると，コピーは $O(N)$ となり，ソートを $O(N \log N)$，重複要素の削除を $O(N)$ とすると，この計算量はソートに支配され，$O(N \log N)$ となる．

4.4.2 縮退操作

部分行列 $R \times C$ に対する 1 回の内積計算のアルゴリズムは次のようになる．R の要素を格納した配列を `rows`，C の要素を格納した配列を `columns` とする．以下は行についてである．列についても同様である．

(1) `rows` の各要素に対応する内積の配列 `ip` を用意する．各要素の値は 0 とする．
(2) `columns` と `rowEntry`[r_b] の共通要素を求める．その結果を配列 `common` に格納する．
(3) `rows` の各要素 r_i について，`rowEntry`[r_i] と `common` の共通要素を求め，`ip` の r_i に相当する要素の値とする．

`rowEntry` の 1 行あたりの平均要素数を n_r，集合 A と B の共通要素を列挙する計算量を $O(|A|+|B|)$ とすると，(2) の計算量は $O(|C|+n_r)$ となる．`common` の要素数を $\min(|C|, n_r)$ とすると，(3) では，1 つの r_i に対して $O(n_r+\min(|C|, n_r)) = O(|C|+n_r)$ となる．これを $|R|$ 回繰り返すので，計算量は $O(|R|(|C|+n_r)) = O(|R||C|+|R|n_r)$ となり，これが行の内積計算の計算量を支配する．同様に，列の内積計算は $O(|R||C|+|C|n_c)$ となる．なお，内積の計算が必要

☆ 次章の実験で用いる行列では $d = 2.7 \times 10^{-4}$ である．

になるのは，縮退方向がその直前と入れ替わったときのみである．同じ方向の縮退が続くときは，その前の内積計算の結果をそのまま用いることができる．

削除操作では ip の最小値を求める必要がある．行方向の場合，要素数 $|R|$ の配列を 1 つずつ見てゆくことで実現できるため $O(|R|)$，列なら $O(|C|)$ となる．よって，縮退操作の計算量は内積計算に支配される．

この計算量は 1 回の縮退操作のものである．終了までの縮退操作の回数はアルゴリズムの実行前には分からないため，初期部分行列構成から停止までの実際の計算時間も分からない．次章の実験でこれを確認する．

4.4.3 終了条件判定

ここでは終了条件のうち，密な部分行列を発見するという MC の目的にとって最も基本的なものである，密度条件の判定アルゴリズムを取り上げる．部分行列 $R \times C$ の密度を計算するアルゴリズムは次のようになる．R の要素を格納した配列を rows，C の要素を格納した配列を columns とする．

(1) $R \times C$ 内の 1 要素数を記録する変数 ones を用意し，0 で初期化する．
(2) rows の各要素 r_i について，rowEntry[r_i] と columns の共通要素を数え，ones に追加する．
(3) ones を面積 $|R||C|$ で割った値を返す．

ここで得られた値を要求された最小密度と比較することで，終了条件を判定できる．縮退操作の場合と同様に考えると，この計算量も $O(|R||C| + |R|n_r)$ となる．

5. 実　　験

前述のように，単純な縮退方向決定関数は，処理速度の点と単調変化する性質を利用する要求を満たす点で望ましいが，密行列発見能力を持たない可能性がある．また，4 章ではこの関数の行数列数比に関する保証も与えられなかった．本章ではこれらの点を実験で確認する．さらに，この関数を用いた場合の内積縮退 MC の処理時間を計測する．

実験に用いた行列は，行数 21,544，列数 182,505，1 要素数 1,077,501，密度 0.027%の実際のデータである．各行の平均 1 要素数は 50.0（最小 1，最大 387），各列の平均 1 要素数は 5.9（最小 1，最大 5473）である．以下ではこの行列を M_{exp} とする．

実験の設定は次のとおりである．初期部分行列は前章の $R_l \times C_l$ とし，M_{exp} のうちの 1,000 行をそれぞれ基準行とし，それらの各行に含まれる 1 要素列から 1 つを選んで基準列とし，内積縮退 MC を計 1,000 回実行する．このときの基準列の選び方は，基準行に 1 要素を持つ列をその 1 要素数でソートし，1 要素数が 2 以上のものから中央のインデックスのものを選ぶというものである．2 以上のものから選ぶ理由は，M_{exp} の列の約 60%が 1 要素を 1 つしか持たないものであり，そのような列を選んだ場合には 1 回の列縮退で基準列以外のすべての列が部分行列から取り除かれるからである．すべての 1 要素列の 1 要素数が 1 である行は基準行として選ばない．縮退方向決定関数の目標比は 1 対 1，終了条件は最小密度 0.5 とする．

5.1 密行列発見能力と行数列数比制御能力

まず，この関数の密行列発見能力を見る．内積縮退 MC を 1,000 回実行した結果，縮退操作は 27,279 回行われ，そのうち 593 回（2.2%）で密度が減少した．よって，この関数でのほとんどの縮退操作で密度は上昇している．また，密度減少が発生したペアについて，減少後の密度を減少前の密度で割った値の平均を求めると，0.93 であった．よって，密度が減少した場合でも減少の程度は小さいといえる．平均面積は 472 であり，十分な面積を持っている．ただし，面積が大きくても行数と列数の比が極端であれば，図 12 のような基準行のみや基準列のみの部分行列ばかりになっている可能性があるため，行数列数比や平均行数・列数も合わせて考える必要がある．

行数列数比として列数を行数で割った値を用いると，1,000 回の実行結果の平均は 0.98（最小 0.036，最大 19），出力部分行列の行数の平均は 21（最小 1，最大 164），列数の平均は 19（最小 1，最大 107）であった．なお，行数列数比の平均値は，相加平均では横長部分行列の影響が過度に現れるため，相乗平均を用いている．指定した行数列数比は 1 であったため，全体としてほぼ望みどおりの比が得られている．

5.2 処 理 時 間

1 回あたりの平均処理時間は，Pentium Ⅳ 2.5 GHz の PC を用いた場合，1.2 秒（最小 0，最大 7.6）であった．そのうち，初期部分行列構成が 0.011 秒，縮退操作の合計が 0.23 秒，終了条件判定の合計が 0.53 秒であった．

この速度は応用目的によっては問題となる．たとえば，ユーザとの対話的な処理でこのアルゴリズムを用いる場合，1.2 秒は体感的に遅いであろう．また，最大の処理時間と平均の処理時間の差の大きさも問題となる．この差の大きさは，内積縮退 MC が処理時間を制御するパラメータを持っていないことに起因する．これらの問題に対応するため，我々は高速化手法を考案した．

6. 内積縮退 MC の高速化

4 章で述べたように，1 回の縮退操作の計算量は $O(|R||C|+|R|n_r)$ または $O(|R||C|+|C|n_c)$ である．縮退の回数によってこの計算量はさらに増大する．また，我々が用いた終了条件判定関数は $O(|R||C|+|R|n_r)$ である．これらから分かるように，処理の高速化方法として，$|R|$ と $|C|$ を小さくすることと，縮退回数を少なくすることが考えられる☆．

6.1 初期部分行列を小さくする方法

初期部分行列を小さくする方法は 2 つある．1 つは，$R_l \times C_l$ ではなく $R_s \times C_s$ を利用するものである．これは前述のように，応用目的によっては用いることができない場合がある．もう 1 つの方法は，R_l のうち，基準行との共通要素数が多いものを上から一定の個数まで取り出して初期部分行列の行とし，列についても同様にするものである．この方法は次の仮定に基づく．すなわち，初期部分行列で内積の少ない行や列は早い段階の縮退で取り除かれるので，これらを初期部分行列に含めなくても出力に大きな影響はないであろう，というものである．この処理のために，4.4.1 項のアルゴリズムを変更し，`rows` から重複要素を削除するかわりに，各行番号の重複回数を数え，行番号を重複回数でソートし，`rows` の端から一定の個数の行番号を取り出す．列についても同様である．このときも元のアルゴリズムと同様に計算量は $O(N \log N)$ となる．

この方法は，高速化を達成することのほかに，処理時間を一定に近づける効果を持つ．計算量の議論より，内積縮退 MC で計算時間を多く消費するのは初期部分行列やそれに近い部分行列であることが分かる．そのため，初期部分行列が行列中の 1 要素の分布によって決められるなら，計算時間も行列中の 1 要素の分布によって決められることになる．それに対してこの方法では，部分行列のサイズをつねに一定以下に抑えることができるため，処理速度をアルゴリズムで制御することができる．

この方法は，初期部分行列に対して，図 8 のような行と列で独立した類似検出を行うことに等しい．その結果，図 8 (b) のように空行や空列を含むことがある．そのため，4.2.2 項での縮退グラフの性質 (3) を保証するには，小さくなった初期部分行列の行と列の内積を計算し，内積 0 のものは排除したうえで，それを初期部分行列とする必要がある．この処理は，すでに部分行列が小さいため，計算時間を多く消費することはない．

なお，これらの 2 つの方法は組み合わせることができる．すなわち，R_s の中から基準行と多くの共通要素を持つものを初期部分行列の行として選ぶことも可能である．

6.2 縮退回数を減らす方法

縮退操作の回数を減らすために我々は，削除操作において内積が最小の行・列だけでなく，内積の小さい行・列を一定の割合で取り除く方法を導入する．この方法では，たとえば 1,000 × 2,000 の部分行列に対して 8 割を削除すると指定すれば，行ならば 800 行が，列ならば 1,600 列が一度の操作で取り除かれる．800 番目の行と同じ内積の行があれば，それらの行も同時に削除する．列についても同様である．この方法を縮退グラフの上で考えると，r 遷移か c 遷移を一度にまとめて行うことに相当する．以下ではこの縮退方法を割合縮退と呼び，内積最小の行・列のみを削除する方法を最小値縮退と呼ぶ．割合縮退で一度に取り除かれる行数・列数の割合を縮退率と呼ぶ．

なお，割合縮退を用いる場合，終了条件を満たす部分行列を発見したとしても，より適切な部分行列が縮退前の部分行列との間に存在する可能性がある．そのため，ある部分行列が終了条件を満たした場合，直前の部分行列から最小値縮退を行うことで，最も適切な部分行列を求める必要がある．

次に，これらの高速化手法の効果を実際の行列を用いて確認する．

7. 高速化実験

7.1 初期部分行列のサイズ変更の効果

まず，初期部分行列のサイズの変化にともなう処理時間の変化を見る．実験の設定は 5 章と同様にする．初期部分行列のサイズは $p \times p$ とし，p の値をさまざまに変える．それにともなう内積縮退 MC の速度変化を**表 1** に示す．処理時間の単位は秒である．この表より，初期部分行列のサイズを小さくすることで高速化を達成でき，処理時間の幅も減らせることが分かる．

さらに，初期部分行列のサイズの変化で出力部分行

表 1 初期部分行列サイズの変化による速度の変化
Table 1 The change of processing speed corresponding to the change of initial submatricies' sizes.

行数	列数	処理時間（平均，最小，最大）
$\|R_l\|$	$\|C_l\|$	1.21, 0, 7.59
2,500	2,500	0.52, 0, 1.82
1,000	1,000	0.20, 0, 0.82
500	500	0.092, 0.01, 0.39

☆ n_r や n_c を小さくすることでも高速化できるが，これらを変更するには元の行列の変更が必要であり，これは行列の意味の問題をともなう．よって，これらについては考えない．

列がどのように変わるかを確認するため，初期部分行列を $R_l \times C_l$ として得られた 1,000 個の部分行列と 500×500 として得られた 1,000 個の部分行列を比較する．この比較は，同じ基準要素から得られた 2 つの部分行列について，$R_l \times C_l$ の出力部分行列の行集合と列集合の要素のうち，500×500 の出力部分行列の行集合と列集合に含まれるものの割合を調べることで行う．その結果，行は 91%が含まれ，列は 94%が含まれていた．これらより，初期部分行列のサイズの変更は出力に大きな影響を与えていないといえる．

7.2 割合縮退の効果

次に，割合縮退の効果を調べる．初期部分行列は 500×500，縮退率は行・列ともに 50%とし，5 章と同様の設定で内積縮退 MC を 1,000 回実行した．

1,000 回の平均をとると，出力部分行列の行数の平均は 19.7（最小 1，最大 106），列数の平均は 18.8（最小 1，最大 61），行数と列数の比の平均は 1.00（最小 0.07，最大 19）平均面積は 459（最小 1，最大 3266）となった．平均処理時間は 0.027 秒であった．

また，前節と同様に，最小値縮退で得られた部分行列と割合縮退で得られた部分行列のそれぞれを比較した．その結果，最小値縮退の出力部分行列の行のうち 88%が割合縮退の出力部分行列に含まれ，列は 99%が含まれていた．これらより，割合縮退は密な部分行列の検出能力を下げることなく処理速度を向上させるといえる．

8. おわりに

本論文では，類似行検出と類似列検出を組み合わせた MC アルゴリズム，内積縮退 MC を定義し，このアルゴリズムが密部分行列の発見能力と実際の行列データに対する十分な速度を持つことを確認した．

今後の課題として，まずこのアルゴリズムを実際の問題に適用する場合の設定の考察があげられる．本論文では説明の例として類似文書検索と推薦を用いたが，これらの応用で生じる問題点，たとえば行列の適切な構成方法や基準列の選択方法などを考える必要がある．

縮退方向決定関数についての議論も必要である．本論文で述べたこの関数は，全解探索を行うものと現在の部分行列からのみ縮退方向を決定するものであり，これらは，出力に対する制御能力を重んじるか処理速度を重んじるかの両極端である．それらの中間の関数を用いれば，出力に対する制御能力と処理速度のバランスがとれると期待できる．そのため，そのような関数の構成方法を考える必要がある．

謝辞 本研究は文部科学省大学等発ベンチャー創出支援制度（課題番号 19）によるものである．

参考文献

1) 小柳 滋，久保田和人，仲瀬明彦：Matrix Clustering：CRM 向けの新しいデータマイニング手法，情報処理学会論文誌，Vol.42, No.8, pp.2156–2166 (2001).
2) 小柳 滋，上原子正利，久保田和人，仲瀬明彦：WWW アクセスシーケンスの新しいマイニング手法の提案，信学論誌，Vol.J87-D-I, No.2, pp.232–243 (2004).
3) Resnick, P. and Varian, H.R.: Recommender System, *Comm. ACM*, Vol.40, No.3, pp.56–58 (1997).
4) Agrawal, R., Imielinski, T. and Swami, A.: Mining Association Rules between Sets of Items in Large Databases, *Proc. 1993 ACM SIGMOD International Conference on Management of Data*, pp.207–216 (1993).
5) Schafer, J.B., Konstan, J.A. and Riedl, J.: E-Commerce Recommendation Applications, *Data Mining and Knowledge Discovery*, Vol.5, No.1/2, pp.115–153 (2001).
6) 福田剛志，森本康彦，徳山 豪：データマイニング，共立出版 (2001).

(平成 15 年 12 月 20 日受付)
(平成 16 年 4 月 12 日採録)

（担当編集委員 高須 淳宏）

上原子正利
1997 年京都大学工学部情報工学科卒業．1999 年同大学大学院工学研究科情報工学専攻修士課程修了．現在，立命館大学客員研究員．

小柳 滋（正会員）
1972 年京都大学工学部数理工学科卒業．1977 年同大学大学院工学研究科数理工学専攻博士課程修了．同年（株）東芝入社．2002 年より立命館大学教授．工学博士．データマイニング，並列処理，コンピュータアーキテクチャに関する研究に従事．電子情報通信学会，ACM，IEEE-CS 各会員．