段階的一般化法によるミスマッチクラスタを表現する最小汎化集合の効率的抽出

田村　慶一[†]　　木村　浩明[††*]　　荒木康太郎[†††**]　　北上　始[†]

An Efficient Method for Extracting Minimum Generalized Set of Mismatch Clusters by Step-Wise Generalization

Keiichi TAMURA[†], Hiroaki KIMURA[††*], Kotaro ARAKI[†††**], and Hajime KITAKAMI[†]

あらまし　配列データベースに対するあいまいな問合せ結果として返されるミスマッチクラスタから，あいまい文字表現を含む最小汎化集合を効率的に抽出する方法を提案する．この提案手法は，ミスマッチクラスタの部分集合のすべてについて，小さなサイズの部分集合から順に列挙し，列挙木を探索するボトムアップアプローチをとっている．この探索過程において，列挙された各集合に対する最汎パターンの計算，不要な部分列挙木の枝刈り，冗長なパターンの除去などを実施することにより，ミスマッチクラスタの最小汎化集合を抽出する．提案手法の有効性を確認するために，10 種類のデータセットを用いて実験を行ったので，その実験結果についても報告する．

キーワード　テキストマイニング，情報抽出，バイオインフォマティクス

1. まえがき

あいまいな問合せ処理は，テキストデータベースや配列データベースから類似する部分文字列の検索をさし，Web 文書，オンライン文書，分子配列データなどに対する情報検索をはじめとして，クラスタリングや配列データマイニングなどの多くの分野で重要な要素技術である．あいまいな問合せ処理は，人為的過誤による誤字を含む単語や熟語をはじめとしてカタカナ語の異表記同義語などが含まれるテキストデータ，同じメッセージが通信路ノイズとともに繰り返し現れるストリームデータ，わずかな揺らぎをもつモチーフが含まれるアミノ酸の分子配列データなどに対する類似検索に有用である．モチーフとは，PROSITE [1] や Pfam [2], [3] などで見られる生物学的に重要な機能をもつ特徴的なパターンである．

例えば，脳梗塞，血栓などの治療薬として利用 [4], [5] されている $Kringle$ ドメインは，最近，癌治療薬の有力な候補として注目されている．本論文ではそのドメインを特徴づけるモチーフを $Kringle$ モチーフと呼ぶ．また，古くからヘビ毒に含まれていることが知られている $Kringle$ ドメインは，いまだ十分には解明されていないがアルツハイマー病患者 [6], [7] の脳からも検出されている．ここでは，そのドメインを特徴づけるモチーフを $Kunitz$ モチーフと呼ぶ．$Kringle$ ドメインにおける多様なモチーフの全体像を明らかにすることは，このような治療薬の開発や難病の解明に大変重要である．

$Kringle$ モチーフは，以下の正規表現の配列パターンとして知られている．

$$<[FY]\text{-}C\text{-}[RH]\text{-}[NS]\text{-}x(7,8)\text{-}[WY]\text{-}C>$$

上記の $[FY]$ は，あいまい文字と呼ばれ，集合 $\{F, Y\}$

† 広島市立大学大学院情報科学研究科，広島市
Graduate School of Information Sciences, Hiroshima City University, 3–4–1 Ozuka-Higashi, Asa-Minami-ku, Hiroshima-shi, 731–3194 Japan
†† 広島市立大学情報科学部，広島市
Faculty of Information Sciences, Hiroshima City University, 3–4–1 Ozuka-Higashi, Asa-Minami-ku, Hiroshima-shi, 731–3194 Japan
††† 広島市立大学大学院情報科学研究科，広島市
Graduate School of Information Sciences, Hiroshima City University, 3–4–1 Ozuka-Higashi, Asa-Minami-ku, Hiroshima-shi, 731–3194 Japan
* 現在，(株) コア四国カンパニー
** 現在，(株) 日立製作所

の中のどの要素文字でも配置が許されることを示している．あいまい文字 $[NS]$ とあいまい文字 $[WY]$ の間の記号 $x(7,8)$ は，ワイルドカード文字（任意の 1 文字を意味する特殊文字）の許容数を表現しており，この場合は，$[NS]$ とあいまい文字 $[WY]$ の間に 7 文字から 8 文字までから成るワイルドカード列の配置が許されていることを示している．

さて，*Kringle* モチーフが含まれる *Kringle* データセットには現在 90 件のデータが収集されており，現在までに知られている 16 種類のモチーフ配列のうち，$<$*Y-C-R-N-x*$(7,8)$*-W-C*$>$:64, $<$*F-C-R-N-x*$(7,8)$*-W-C*$>$:22, $<$*F-C-R-S-x*$(7,8)$*-W-C*$>$:9, $<$*Y-C-R-N-x*$(7,8)$*-Y-C*$>$:2 が含まれている．ただし，コロン (:) の後の数字は，モチーフ配列が異なる配列データに現れる数，すなわち支持数を示す．*Kringle* データセットに対するあいまいな問合せ処理の結果として得られる類似部分配列の集合，すなわちミスマッチクラスタには，モチーフとは無関係な部分配列が非常に多く含まれている．このミスマッチクラスタから汎化パターン集合を抽出する仕組みがあると，例えば 2 種類のモチーフ配列を含む汎化パターン $<$[*FY*]*-C-R-N-x*$(7,8)$*-W-C*$>$: 81 を抽出することができる．これにより抽出される汎化パターンは支持数に関して上位にランキングされる．また，現在の *Kringle* データセットには登録されていないが，支持数が極めて低いモチーフ配列 $<$*Y-C-R-S-x*$(7,8)$*-W-C*$>$ が登録されると，4 種類のモチーフ配列を含む汎化パターンとして $<$[*FY*]*-C-R-*[*NS*]*-x*$(7,8)$*-W-C*$>$ を抽出することができる．これにより抽出される汎化パターンは更に上位にランキングされる．すなわち，この汎化パターンは，モチーフに関係しない屑パターンと更に容易に区別できるようになる．このような汎化パターンを人手で見落しなく抽出しようとすると，モチーフとは無関係な部分配列が多数含まれるミスマッチクラスタの中から目立たない，すなわち $<$*Y-C-R-N*$^{-}-x(7,8)$*-W-C*$>$:64 に比べて頻出ではない類似配列 $<$*F-C-R-N-x*$(7,8)$*-W-C* $>$: 22, $<$ *F-C-R-S-x*$(7,8)$*-W-C*$>$: 9 を見つけ出し，汎化を行う必要があるので，大変な手間がかかる．ミスマッチクラスタから汎化パターンを抽出する仕組みは，*Kringle* データセットに限らず他のデータセットにも応用可能である．例えば，*Zinc Finger* データセットには 1893 件のデータが収集されているが，高い支持数をもつモチーフ配列として

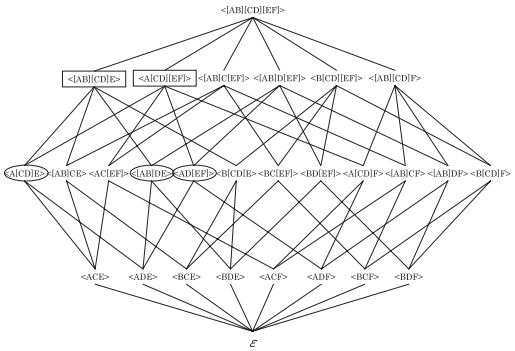


図 1　汎化パターンの列挙例
Fig. 1　An example of enumeration of generalized patterns.

$<$*C-x*$(2,4)$*-C-x*(3)*-F-x*(8)*-H-x*$(3,5)$*-H*$>$: 1678 が存在するのに対して，極めて低い支持数をもつモチーフ配列 $<$*C-x*$(2,4)$*-C-x*(3)*-M-x*(8)*-H-x*$(3,5)$*-H* $>$: 14 が含まれており，汎化パターン集合を抽出する仕組みがなければ，この低い支持数をもつモチーフ配列を見落としてしまう．このような仕組みを未知のデータセットに利用すると，モチーフ配列を含む汎化パターンが上位にランキングされる傾向があるので，新たなモチーフ発見の機会を与えることにつながる．また，このような仕組みは，PROSITE や Pfam などのモチーフデータベースに存在するモチーフの正規表現に見落としや誤りがないかどうかを検証する仕組みとしても重要な役割を与えるものと期待されている．

しかしながら，従来のあいまいな問合せ処理の研究（文献 [8]〜[16]）では，長さ k の問合せ文字列に対して誤差半径 r 以内に入る k-部分文字列をすべて求める研究にとどまっている．本論文では，あいまいな問合せ処理の結果として得られる k-部分文字列集合をミスマッチクラスタと呼び，このミスマッチクラスタの効率的な汎化に着目する．以後，ミスマッチクラスタの汎化により得られる配列パターンを汎化配列パターンあるいは単に汎化パターンと呼び，汎化配列パターンが異なる配列データに現れる数を汎化配列パターンの支持数と呼ぶ．

ミスマッチクラスタを表現する汎化配列パターンの計算には，トップダウンとボトムアップの二つの探索アプローチが考えられる．例を用いて，両方の違いについて考えてみよう．図 1 は，ミスマッチクラスタ $MIS = \{< ACE >, < ADE >, < BCE >, < BDE >, < ACF >, < ADF >, < BCF >, < BDF >\}$

から列挙される汎化パターンをハッセ図で表現したものである．このハッセ図上で MIS に含まれる二つのミスマッチクラスタ MIS_1，MIS_2 について考えてみよう．ただし，$MIS_1 = \{< ACE >,< ADE >,< BCE >,< BDE >,< ACF >,< ADF >\}$，$MIS_2 = \{<ACE>,<ADE>,<BDE>,<ADF>\}$ とする．図の最上位にある汎化パターン$<[AB][CD][EF]>$は，MIS，MIS_1，MIS_2 のどのミスマッチクラスタに対しても最も一般的な配列パターン（最汎パターンと呼ぶ）である．MIS を表現する汎化配列パターンは，この最汎パターンと一致する．MIS_1 を表現する汎化配列パターンは図の方形で囲まれた二つのパターンであり，MIS_2 を表現する汎化配列パターンは，図の丸で囲まれた三つのパターンである．

トップダウン探索のアプローチの具体的な方法については，筆者らは，既に，反復精密化法 [17], [18] と呼ぶ方法を提案している．反復精密化法は，図の最上位にある最汎パターン$<[AB][CD][EF]>$を列挙木のルートとみなし，ルートから下方に探索を進める方法である．MIS_1 の解は最上位付近に存在するので，短時間に解を見つけ出すことは可能だが，MIS_2 の解はミスマッチクラスタ付近に存在するので，解の探索に多くの時間を費やしてしまう．

本論文では，この問題を解決するために，ユーザの背景知識が得られていないという前提に立ち，ミスマッチクラスタを表現する最小汎化集合，すなわち少数の汎化配列パターンと汎化できなかった類似部分配列からなる集合を効率的に抽出する方法としてボトムアップ探索アプローチによる段階的一般化法 [19] を提案する．すなわち，この方法では，ミスマッチクラスタ（図 1 の例では下から 1 段目）付近に解が存在する場合に，図の最下位にある空文字 ϵ を列挙木のルートとみなし，ルートから上方に探索を進めることにより解を短時間に見つけ出せる．

本論文の構成は以下のとおりである．まず，**2.** で関連研究について述べ．**3.** で，本論文で重要な用語や問題の定義について説明する．**4.** で，段階的一般化法で重要な役割を果たすパターン切除操作について述べた後，**5.** で，段階的一般化法を提案する．**6.** で，提案手法の有効性を確認するための性能評価結果について報告し，**7.** で全体をまとめる．

2. 関 連 研 究

あいまいな問合せ処理の研究は，ある文字列に対して，文字の削除，挿入，置換をする操作に基づく類似性検索や近似検索の研究として数多く行われてきた [8]〜[16]．これらの研究では，ビットベクトル [9]，サフィックス木 [10], [13], [14]，V-木 [11]，MAT-木 [12] などが提案され，類似検索性能の向上が中心話題であった．このため，大量に返される類似検索結果を分析し規則的な特徴を把握する方法には十分な関心が寄せられていなかった．本論文では，この点に着目し，類似検索結果，ミスマッチクラスタを表現する最小汎化集合を抽出する方法について提案している．この抽出には文字データの集合に対して汎化を行う処理が重要となるので，以下では汎化に関す関連研究について述べる．

文字データの集合に対して汎化を行う処理は，汎化処理と呼ばれるが，述語論理に基づく帰納論理プログラミング [20], [21]，概念階層を用いたデータベースの属性指向帰納法 [22]，PrefixSpan 法に基づく可変長配列パターン抽出法 [23]，反復精密化法 [17], [18] などに見られる．

帰納論理プログラミングでは，正や負のファクトを含む集合を説明するホーン節ルール集合を抽出する問題が扱われている．これに対して，本論文では類似文字列集合を説明する正規表現を抽出する問題を扱っているので，この問題はホーン節ルール集合を抽出する問題とは形式的に異なる．このホーン節ルール集合を抽出する問題を解くためには，論理式の性質を用いた汎化処理が重要であり，トップダウン探索 [20] とボトムアップ探索 [21] により解くアプローチがある．トップダウン探索のアプローチは最も一般的な仮説から探索を始め仮説の特殊化を繰り返し進めることにより解を見つけ出す探索方法であり，ボトムアップ探索のアプローチはファクトと背景知識を組み合わせて特殊な仮説を段階的に生成することにより解を見つけ出す探索方法である．これに対して，本論文の提案手法は，このボトムアップ探索のアプローチに似ているが，汎化処理をするためにあいまい文字に関する正規表現の性質を利用して最汎パターンの計算やパターン切除の操作を実施している点が異なる．

属性指向帰納法では，データベースリレーションの各属性値を一般化することにより属性ルール集合を抽出する問題を扱っている．これに対して，本論文の問題として扱われている正規表現の集合は属性ルール集合とは異なる．また，属性指向帰納法では属性値の汎化処理のために概念階層木を用いているのに対して，本論文の提案手法ではこのような概念階層木の代わり

に，最汎パターンの計算とパターン切除の操作を用いて汎化処理を行っている．

PrefixSpan 法は，述語論理に基礎を置く帰納論理プログラミングの分野とは異なり購買データのバスケット解析から生まれた計算手法である [24]．PrefixSpan 法では，配列データベースから頻出する部分文字列を抽出する問題を扱っているが，この問題を解くためには汎化処理は不要である．Prefixspan 法を拡張した可変長パターン抽出法 [23] では，配列データベースから可変長ワイルドカード領域を有する頻出配列パターンを抽出する問題を扱っており，この問題を解くには，可変長ワイルドカード領域の候補を見つけ出すための汎化処理が必要になる．この汎化処理は，配列データ上のアルファベット文字をワイルドカードに置き換えることにより実施されている．これに対して，本論文では，可変長ワイルドカード領域とは異なるあいまい文字領域を有する配列パターンを抽出する問題を扱っており，この問題を解くには，複数の類似部分配列を一つの配列パターンにまとめる汎化処理が必要になる．したがって，この汎化処理は，アルファベット文字を一つのワイルドカードに置き換えるような汎化処理とは異なる．

反復精密化法 [17], [18] では，本論文と同じ問題を扱っており，トップダウン探索のアプローチによりミスマッチクラスタを表現する最小汎化集合を抽出している．しかしながら，**1.** で述べたように，最小汎化集合がミスマッチクラスタの最汎パターンとは大きくかけ離れている場合には，大きな計算時間を要するという問題があった．本論文では，ボトムアップ探索のアプローチによる段階的一般化法 [19] を提案し，効率的な枝刈り処理などを導入することにより，この問題を解決している．

3. 用語と問題の定義

ミスマッチクラスタから最小汎化集合を抽出する方法を説明するためには多くの用語や記号を定義しておかなければならない．本章では，用語や記号の定義，及び本論文で扱う問題の定義を行う．

3.1 あいまいな問合せ

部分文字列 K と許容誤差 $r(\geq 0)$ が，問合せ Q として与えられているとしよう．この問合せ Q による処理で，ハミング距離 $d(K, K') \leq r$ を満たす部分文字列 K' がすべてを配列データベース DB から選択されるとき，問合せ Q をあいまいな問合せと呼び，この問合せ Q による処理をあいまいな問合せ処理と呼ぶ．K は検索円の中心と呼ばれ，r は検索円の半径とも呼ばれる．

また，この問合せ Q では，検索文字列 K の中にどんな文字とも誤差が 0 で一致するワイルドカード文字の表記を許す．

3.2 ミスマッチクラスタ

あいまいな問合せでは，ユーザが長さ k をもつある文字列（k-文字列）と許容誤差 r を検索条件として与える．この問合せにより配列データベース DB から検索条件を満たす長さ k の部分文字列 $<inst>$ の集合が得られる．この集合をミスマッチクラスタ MIS と呼ぶ．以後，ミスマッチクラスタ MIS を次の形式で表現する．

$$MIS = \{<inst_1>, <inst_2>, \cdots, <inst_n>\} \qquad (1)$$

3.3 あいまい性を表現する汎化配列パターン

Σ_i をアルファベット Σ の部分集合とするとき，次式のように k 個の Σ_i を並べたパターンを k-汎化配列パターンと呼び，$<pat^k>$ と表記する．

$$<pat^k> = <\Sigma_1\text{-}\Sigma_2\text{-}\cdots\text{-}\Sigma_{k-1}\text{-}\Sigma_k>: cnt \qquad (2)$$

ただし，Σ_i は，たびたび括弧 [ ] の中に Σ_i の全要素を列挙した表記をする．式 (2) 中の，$\Sigma_i \subseteq \Sigma$ が存在する場所をあいまい文字領域と呼ぶ．また，$|\Sigma_i| \geq 2$ のとき，集合 Σ_i はあいまい文字ドメインと呼ばれ，Σ_i 内に存在する任意の 1 文字の配置が許されることを示している．あいまい文字ドメインが 1 箇所以上存在するとき，式 (2) を k-汎化配列パターンと呼ぶ．ハイフン “ - ” は左右の文字の接続を意味するが，以後，たびたび省略されることがある．

式 (2) の最右端にある記号 cnt は，たびたび省略されることがあるが，そのパターンの支持数を意味する．パターンの支持数とは，そのパターンが出現する異なる配列データの数をさす．

また，$\Sigma_i = \{\alpha_i\}$ のとき，式 (2) を以下のように表現する．

$$<pat^k> = <\alpha_1\alpha_2\cdots\alpha_{k-1}\alpha_k> \qquad (3)$$

ただし，$a_i \in \Sigma$ とする $(1 \leq i \leq k)$.

3.4 インスタンスを導出する関数

k-汎化配列パターン $<\Sigma_1\Sigma_2\cdots\Sigma_{k-1}\Sigma_k>$ から k-インスタンスのすべて，すなわち長さ k の部分文字列の集合を導出する関数を $EVAL(<\Sigma_1\Sigma_2\cdots\Sigma_{k-1}\Sigma_k>)$

と書くことにすると，以下の関係式が成立する．

$$EVAL(<\Sigma_1\Sigma_2\cdots\Sigma_{i-1}(\Sigma_i-\{\alpha_i\})\cdots\Sigma_k>) = EVAL(<\Sigma_1\Sigma_2\cdots\Sigma_{i-1}\Sigma_i\cdots\Sigma_k>) - EVAL(<\Sigma_1\Sigma_2\cdots\Sigma_{i-1}\{\alpha_i\}\cdots\Sigma_k>) \quad (4)$$

例えば，式 (4) の左辺を $EVAL(<[AB]([CD]-D)>)$ としてみよう．これは，$EVAL(<[AB]C>) = \{<AC>, <BC>\}$ となるので，式 (4) の右辺= $EVAL(<[AB][CD]>) - EVAL([AB]D)$ に等しいことが分かる．

以後，この関数 $EVAL$ の表記を拡張し，n 個の k-汎化配列パターンからなる集合 $P^k = \{<pat_1{}^k>, <pat_2{}^k>, \cdots, <pat_n{}^k>\}$ に対して，P^k の各要素に含まれるすべての k-インスタンスを導出する関数を $EVAL(P^k)$ と表記する．したがって，$EVAL(\{<pat^k>\})$ は，1 個の k-汎化配列パターン $<pat^k>$ からすべての k-インスタンスを導出することを意味する．

二つの k-汎化配列パターンを $<pat_1{}^k>, <pat_2{}^k>$ としよう．$EVAL(\{<pat_1{}^k>\}) \supseteq EVAL(\{<pat_2{}^k>\})$ が成立するとき，$<pat_2{}^k>$ は $<pat_1{}^k>$ に含まれるという．別の言い方では，$<pat_2{}^k>$ は $<pat_1{}^k>$ に冗長であるともいう．

3.5 最汎パターン

ある k-インスタンスの集合を I^k としよう．$1 \le j \le k$ に対して，アルファベット Σ_j を以下のように定義する．

$$\Sigma_j = \{inst[j] \mid inst \in I^k\} \quad (5)$$

ただし，$inst[j]$ は，$inst$ の先頭から j 番目の文字を意味する．このとき，$<\Sigma_1\Sigma_2\cdots\Sigma_k>$ を k-インスタンス集合 I^k に対する最汎パターンと呼び，この最汎パターンを $MGP(I^k)$ と表記する．

また，ある k-汎化配列パターンの集合を P^k とするとき，$1 \le j \le k$ に対して，アルファベット Σ_j を以下のように定義する．

$$\Sigma_j = \{string[j] \mid string \in EVAL(\{<pat^k>\}), <pat^k> \in P^k\} \quad (6)$$

このとき，$<\Sigma_1\Sigma_2\cdots\Sigma_k>$ を k-汎化配列パターン集合 P^k に対する最汎パターンと呼び，この最汎パターンを $MGP(P^k)$ と表記する．

3.6 問題の定義

我々のゴールは，あいまいな問合せの結果として得られるミスマッチクラスタ MIS の最小汎化集合を効率的に計算することである．例えば，$MIS = \{<ABC>, <ABD>, <ACD>, <BCD>\}$ の最小汎化集合として，$MGS = \{<AB[CD]>, <[AB]CD>, <A[BC]D>\}$ を効率的に計算することである．

ある集合 $MGS = \{G_1, G_2, \cdots, G_m\}$ が，k-汎化配列パターン $<pat^k>$ 及び k-インスタンス $<inst^k>$ から構成されているとしよう ($1 \le m \le |MIS|$)．ただし，$EVAL(\{<pat^k>\}) \subseteq MIS$ かつ $<inst^k> \in MIS$ を満たすものとする．

この集合 MGS が以下の性質を満たすとき，MGS を MIS に対する最小汎化集合と呼ぶ．

(1) $EVAL(MGS) = MIS$ が成立する．

(2) MGS の任意の 2 要素 G_i, G_j に対して，G_i と G_j の間には冗長な関係が存在しない ($1 \le i \ne j \le m$)．

(3) MGS に含まれるどの要素 G_i も極大である ($1 \le i \le m$)．すなわち，更に汎化すると MIS に存在しないインスタンスを含んでしまうことになる．

(4) 上記の (1)〜(3) を満たす任意の MGS' に対して，$|MGS'| \le |MGS|$ が成立する．

上記 (2) では，検出が容易な冗長性に限定しているので，MGS が上記 (1)〜(3) を満たすだけでは MGS を一意に定めることができない．したがって，これらに上記 (4) を追加し，最小汎化集合が一意に定まるようにしている．例えば，$MIS = \{<ABC>, <ABD>, <ACD>, <BCD>\}$ とすると，$MGS_1 = \{<AB[CD]>, <[AB]CD>, <A[BC]D>\}$ と $MGS_2 = \{<AB[CD]>, <[AB]CD>\}$ の二つの汎化集合はどちらも (1)〜(3) を満たすが，(4) を追加することにより，MGS_1 が最小汎化集合として一意に選択される．この詳細は，**5.2** で考察する．

4. パターン切除操作

本論文では，ミスマッチクラスタ MIS に対する最小汎化集合を計算する方法として，MIS の部分集合 $SubMIS$ についてサイズの小さなものから順に $SubMIS$ の最汎パターン $MGP(SubMIS)$ を列挙するボトムアップ探索のアプローチによる方法を提案しようとしている．提案する方法では，$MGP(SubMIS)$ から汎化パターンを切り取ることによって負のインスタンス，すなわち MIS に存在しないインスタンスが含まれていないかどうか調べる操作が必要であるので，パターン切除操作が中心的な役割をもつ．本章では，その基礎となる単一パターン切除操作とそれを複数回

実施する多重パターン切除操作について述べる．

4.1 単一パターン切除操作

k-汎化配列パターン $<pat^k>=<\Sigma_1\Sigma_2\cdots\Sigma_i\cdots\Sigma_k>$ と k-インスタンス $<inst^k>=<\alpha_1\alpha_2\cdots\alpha_{k-1}\alpha_k>$ があるとしよう．

k-汎化配列パターン $<pat^k>$ から $<inst^k>$ をパターン切除する操作 $PCUT$ は以下のとおりである．

$$PCUT(<pat^k>,<inst^k>) = \{<\Sigma_1\Sigma_2\cdots\Sigma_{i-1}(\Sigma_i-\{\alpha_i\})\cdots\Sigma_k> \mid 1\le i\le k\} \quad (7)$$

すべての i に対して，$\alpha_i\in\Sigma_i$ であれば，$PCUT(<pat^k>,<inst^k>)$ は，最大 k 個の汎化配列パターンを生成し，$\alpha_i\notin\Sigma_i$ であれば，$PCUT(<pat^k>,<inst^k>)=\{<pat^k>\}$ となる $(1\le i\le k)$．式 (7) の右辺は，$EVAL(\{<pat^k>\})-\{<inst^k>\}$ を表現する最小汎化集合である．

例) $PCUT(<[AD][BE][CF]>,<ABF>)$ の計算結果は，$\{<D[BE][CF]>,<[AD]E[CF]>,<[AD][BE]C>\}$ となる．

式 (7) において，k-インスタンス $<inst^k>=<\alpha_1\alpha_2\cdots\alpha_k>$ を k-汎化配列パターン $<\Gamma_1\Gamma_2\cdots\Gamma_k>$ に置き換えてみよう．パターン切除操作 $PCUT$ を以下のように拡張することができる．ただし，$\Gamma_i\subseteq\Sigma_i$ とする $(1\le i\le k)$．

$$PCUT(<\Sigma_1\Sigma_2\cdots\Sigma_k>,<\Gamma_1\Gamma_2\cdots\Gamma_k>) = \{<\Sigma_1\Sigma_2\cdots\Sigma_{i-1}(\Sigma_i-\{\Gamma_i\})\cdots\Sigma_k> \mid 1\le i\le k\} \quad (8)$$

すべての i に対して，$\Gamma_i\neq\Sigma_i$ かつ $\Gamma_i\cap\Sigma_i\neq\phi$ であれば，$PCUT(<\Sigma_1\Sigma_2\cdots\Sigma_k>,<\Gamma_1\Gamma_2\cdots\Gamma_k>)$ は，k 個の汎化配列パターンを要素として返し，$\Gamma_i\cap\Sigma_i=\phi$ であれば，$PCUT(<\Sigma_1\Sigma_2\cdots\Sigma_k>,<\Gamma_1\Gamma_2\cdots\Gamma_k>)$ は $\{<\Sigma_1\Sigma_2\cdots\Sigma_k>\}$ を返す $(1\le i\le k)$．特に，すべての i に対して，$\Gamma_i=\Sigma_i$ であれば，$PCUT(<\Sigma_1\Sigma_2\cdots\Sigma_k>,<\Gamma_1\Gamma_2\cdots\Gamma_k>)$ は空集合 ϕ を返す．

4.2 多重パターン切除操作

k-汎化配列パターン $<pat^k>=<\Sigma_1\Sigma_2\cdots\Sigma_k>$ のほかに k-汎化配列パターン集合 $P^k=\{<pat_1{}^k>,<pat_2{}^k>,\cdots,<pat_n{}^k>\}$ があるとしよう．多重パターン切除操作は，k-汎化配列パターン $<pat^k>$ から k-汎化配列パターン集合 P^k を切除する操作である．多重パターン切除操作 $MCUT(<pat^k>,P^k)$ により $EVAL(\{<pat^k>\})-EVAL(P^k)$ を表現する最小汎化集合が計算される．

$P^k=\{<pat_1{}^k>,<pat_2{}^k>\}$ のとき，$EVAL(\{<pat^k>\})-EVAL(P^k)$ を表現する最小汎化集合は，以下の $MCUT$ で計算可能である．

$$MCUT(<pat^k>,\{<pat_1{}^k>,<pat_2{}^k>\}) = \{pat \mid pat\in PCUT(x,<pat_2{}^k>), x\in PCUT(<pat^k>,<pat_1{}^k>)\} \quad (9)$$

$|P^k|\ge 2$ のとき，$P^k=\{w\}\cup SubP^k, |SubP^k|\ge 1$ の関係が成立するとしよう．このとき，式 (9) を以下の再帰的な計算式に拡張すれば計算可能になる．

$$MCUT(<pat^k>,P^k) = \{pat \mid pat\in MCUT(x,SubP^k), x\in PCUT(<pat^k>,w)\}, \quad (10)$$

この多重パターン切除操作 $MCUT$ では k-汎化配列パターン集合 P^k の先頭要素から順に単一パターン切除操作 $PCUT$ を適用することにより計算が進められる．しかし，$PCUT$ はパターン切除の仕方のすべての組合せを計算するので，P^k の要素の選び方に関係ない．すなわち，P^k の要素は，どのような順番で単一パターン切除操作 $PCUT$ を適用されても $MCUT$ の計算結果は変わらない．

5. 段階的一般化法

ミスマッチクラスタ MIS の冪集合 2^{MIS} ($2^{|MIS|}$ 個の部分集合 $SubMIS$ の集まり）に含まれる要素を，ボトムアップに小さいサイズの要素から順に列挙することについて考えてみよう．本章で提案する段階的一般化法は，それらの要素（部分集合）をボトムアップに列挙しながら負のインスタンスを含まない最汎パターン $MGP(SubMIS)$ を深さ優先ですべて探索する方法を採用している．ただし，親ノードから子ノードを列挙するときは，幅優先で列挙する．

MIS の部分集合 $SubMIS$ をすべて列挙するとき，重複した列挙は回避すべきである．そのために，MIS の各要素に識別番号を振り，この識別番号を用いて辞書式順に MIS の部分集合を列挙する方法がある．図 2 は，問合せ文字列 $<DEF>$ と許容誤差 $r=2$ に対して，$MIS=\{<ABF>,<AEC>,<AEF>,<$

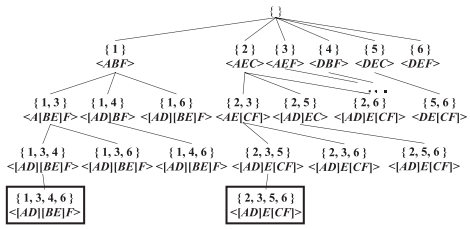


図 2　列挙木による探索処理の例
Fig. 2　An example of search using the enumeration tree.

$DBF>, <DEC>, <DEF>\}$ が返されたものとし，この MIS から最小汎化集合を抽出する例である．この例を用いると，MIS を識別番号集合 $\{1,2,3,4,5,6\}$ に置き換えて，MIS の部分集合を列挙することができる．すなわち，列挙木ノードは，MIS の部分集合 $SubMIS$ の代わりに，MIS の要素に付与された識別番号の集合で表現する．以下では，列挙木ノード（識別番号集合）に対応する MIS の部分集合 $SubMIS$ を列挙木ノードの MIS 部分集合 $SubMIS$ とも呼ぶ．また，列挙木ノードの MIS 部分集合 $SubMIS$ から計算される最汎パターン $MGP(SubMIS)$ を列挙木ノードの最汎パターンとも呼ぶ．

5.1　列挙木ノードに対する基本操作

MIS の部分集合 $SubMIS$ の列挙により列挙木が成長する．この列挙木の成長や解集合の探索のために，列挙木ノードに対する五つの基本操作が重要である．ここでは，これらの基本操作について述べる．

（操作 1）親ノードから子ノードの列挙

MIS の冪集合 2^{MIS} に包含関係 $\subseteq$ を定義した順序集合 $(2^{MIS}, \subseteq)$ を用いて，列挙木を生成すると，親ノードは子ノードに含まれるという性質を利用できる．親ノードが空集合である場合は，ルートノードを意味するので，その子ノードは一つの要素番号から構成される集合とし，すべての子ノードを幅優先で辞書式順に列挙する．親ノードが空集合でない場合は，子ノードを次のように列挙する．P' を親ノード P の兄弟ノードとするとき，$|P|+1 = |P \cup P'|$ となるような P' を選択し，辞書式順に $P \cup P'$ を子ノードとして列挙する．図 2 の例を用いて説明しよう．ここでは，MIS が番号集合 $\{1,2,3,4,5,6\}$ で表記されているとする．親ノードを $\{1\}$ とすると，その子ノードは $\{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{1,6\}$ の順に列挙される．親ノードを $\{2\}$ とするとき，その子ノードは $\{2,3\}, \{2,4\}, \{2,5\}, \{2,6\}$ の順に列挙される．親ノードを $\{6\}$ とするとき，その子ノードは存在しない．

（操作 2）最汎パターンの計算

列挙木ノードに対応する部分集合 $SubMIS$ から最汎パターン $MGP(SubMIS)$ を計算する．図 2 の例を用いて，列挙木ノード $\{1,3,4\}$ から最汎パターンを計算してみよう．$\{1,3,4\}$ に対応する部分集合 $SubMIS$ は，$\{<ABF>, <AEF>, <DBF>\}$ である．したがって，最汎パターン $MGP(SubMIS)$ は $<[AD][BE]F>$ となる．

（操作 3）枝刈り処理

列挙木を深さ方向に探索を進めると，汎化が進むので，列挙木ノードから計算される最汎パターン $MGP(SubMIS)$ に負のインスタンス（MIS に存在しないインスタンス）が含まれる確率が高くなる．したがって，$MGP(SubMIS)$ に負のインスタンスが含まれた時点で $SubMIS$ をもつ列挙木ノードをルートとする部分列挙木の枝刈りが可能である．$MGP(SubMIS)$ に負のインスタンスが存在するかどうかの判定は，$EVAL(\{MGP(SubMIS)\})$ の中に負のインスタンスが存在するかどうか調べればよい．すなわち，$EVAL(\{MGP(SubMIS)\}) - MIS$ の計算結果が空集合でないとき，その要素は MIS に存在しないインスタンスである．したがって，以下の条件式が成立するならば $EVAL(\{MGP(SubMIS)\})$ に負のインスタンスが存在すると結論できる．

$$EVAL(\{MGP(SubMIS)\}) - MIS \neq \phi \quad (11)$$

例えば，図 2 において，列挙木ノード $\{1,3,4\}$ は存在するが，$\{1,3,5\}$ は存在しない．前者の最汎パターンは $<[AD][BE]F>$ であり，後者の最汎パターンは $<[AD][BE][FC]>$ である．前者のインスタンス $EVAL(\{<[AD][BE]F>\})$ 集合は，MIS に含まれるが，後者のインスタンス $EVAL(\{<[AD][BE][FC]>\})$ 集合には MIS には存在しない負のインスタンス $<ABC>$ を含むため，式 (12) を満足してしまう．したがって，図 2 では，前者は残るが，後者は枝刈りにより除去される．

（操作 4）候補解集合の構成要素の収集

それぞれの葉ノードの先祖に該当する非葉ノードはすべて葉ノードに冗長であるので，葉ノードの MIS 部分集合（これを $Leaf$ と表記する）から計算された最汎パターン $MGP(Leaf)$ は負のインスタンスを含まない汎化配列パターンの中で最も一般化されたパターンである．したがって，$MGP(Leaf)$ は候補解集合

の一要素である．$MGP(SubMIS)$ に負のインスタンスを含まない列挙木ノードが葉ノードであるための必要十分条件は，その列挙木ノードから子ノードを幅優先ですべて列挙したときに，(1) 親ノードとその兄弟ノードをどのように組み合わせても子ノードがもはや列挙不可能か，または (2) そのどの子ノードにも負のインスタンスが含まれてしまうことである．図 2 の例において，負のインスタンスを含まない三つの列挙木ノード $\{1\},\{1,3\},\{1,3,6\}$ について考えて見よう．$\{1\}$ 及び $\{1,3\}$ については，いずれも葉ノードの条件（2）を満たさない．$\{1,3,6\}$ については，もはやその子ノードを列挙できないため，葉ノードの条件（1）を満たす．したがって，葉ノード $\{1,3,6\}$ の MIS 部分集合 $\{<ABF>,<AEF>,<DEF>\}$ から計算される最汎パターン $<[AD][BE]F>$ は候補解集合の一要素になる．

（操作 5）冗長性の除去

葉ノード間に冗長性が存在すると，最小汎化集合のサイズを大きくするので除去すべきである．したがって，列挙木探索を終了した時点で，冗長な葉ノードを候補解集合から一括除去する．図 2 の例では，$\{1,3,4,6\},\{1,3,6\},\{1,4,6\},\{1,6\}$ の順に葉ノードであるため，候補解集合に含まれるが，$\{1,3,6\},\{1,4,6\},\{1,6\}$ はどれも $\{1,3,4,6\}$ に冗長であるので，これらは候補解集合から除去される．

5.2 段階的一般化法の処理手順

ここでは，識別番号を付与したミスマッチクラスタの表現を $MIS = \{(1,<inst_1>),(2,<inst_2>),\cdots,(n,<inst_n>)\}$ とする．図 3 に，本提案手法に基づく処理手順を示す．図 3 の処理手順 (1) は，列挙木の深さ優先探索を行うための変数 $Stack$ と候補解を格納するための変数 $Candidate$ について，初期化を行う処理である．処理手順 (2) (a) は，**5.1** の操作 1 に基づいて行われる親ノードから子ノードを列挙する処理である．処理手順 (2) (b) ① は操作 2 に基づいて行われる最汎パターンを計算する処理であり，処理手順 (2) (b) ② は操作 3 に基づいて行われる枝刈り処理である．処理手順 (2) (c) は操作 4 に基づいて行われる候補解集合の構成要素を収集する処理であり，処理手順 (3) は操作 5 に基づいて行われる冗長性除去の処理である．以上の処理を経て，$Candidate$ に残された要素集合が最小汎化集合となり，それに支持数を付与して作られる $Solution$ が処理手順 (4) で出力される．

以下では，段階的一般化法の処理手順により，**3.6** で定義された最小汎化集合 MGS が得られていること，及び MGS の一意性について考察する．

(1) $Stack$:=$\{\{1\},\{2\},\cdots,\{n\}\}$; ただし，$\{i\} \in Stack$ の i はミスマッチクラスタ MIS に存在する要素の識別番号とし，$Stack$ の要素は辞書式順に取り出せるように管理する．
(2) $Candidate := \{\}$; ただし，$Candidate$ を候補解を格納する領域とする．
(3) **while**$(Stack \neq \phi)$
  for each $S \in Stack$
  (a) $Next := S$ を親ノードとして列挙された子ノード C の集合;
  (b) **for each** $C \in Next$
    ①<pat>:= $MGP(C$ に対する $SubMIS)$;
    ②**if** $EVAL($<pat>$) - MIS \neq \phi$ **then** C を捨てる;
      else $Stack := Stack \cup \{C\}$;
  (c) **if** S が葉ノードの条件を満たす
    then $Candidate := Candidate \cup MGP(S$ に対する $SubMIS)$;
(4) $Candidate$ から冗長な要素を除去した後，配列データベース DB における各要素の支持数を計算し，$Solution$ に格納する;
(5) $Solution$ を支持数付き最小汎化集合 MGS として出力する;

図 3 提案手法に基づく処理手順
Fig. 3 Processing step based on the proposed method.

最小汎化集合が **3.6** (1) の $EVAL(MGS) = MIS$ の成立については，ミスマッチクラスタ MIS のべき集合 2^{MIS} の中から負のインスタンスを含まない部分集合 $SubMIS$ のすべてを **5.1** の操作 1 と操作 3 により選び出し，それに基づいて認識されたそれぞれの葉ノードから最汎パターン $MGP(SubMIS)$ を **5.1** の操作 2 と操作 4 により計算することにより保証している．

3.6 (2) の非冗長性については，最小汎化集合 MGS の構成要素は葉ノード以外から作成しないこと（**5.1** の操作 4），及び葉ノード間の冗長性を除去すること（**5.1** の操作 5）により保証している．**3.6** で述べたが，**3.6** (1)〜(3) だけでは最小汎化集合 MGS の一意性を保証しないが，**3.6** (4) を追加することにより，一意性を保証している．

以下，**3.6** (4) の $|MGS'| \leq |MGS|$ の成立について考察する．探索は **5.1** の操作 1 と操作 3 により列挙木を段階的に成長させながら負のインスタンスを含む直前まで進められるので，最も多くの正のインスタンスを含む汎化配列パターンを見つけ出すことにより **3.6** (4) を保証している．もし，$|MGS| \leq |MGS'|$ となる MGS' が存在すれば操作 1 ですべてを列挙して

いないか，操作 3 の枝刈りが適切でないか，**3.6** (2) に対応する操作 5 で検出に手間のかかる冗長性の除去が実施されたか，であり矛盾する．以上により，最小汎化集合 MGS は非冗長であっても，MGS 内の二つの要素間には同じインスタンスが含まれる可能性がある．なお，検出に手間のかかる冗長性については，**7.** の今後の課題で触れる．

5.3 枝刈り条件式の高速計算

ここでは，まず式 (11) における $EVAL(MGP(SubMIS))$ の部分の計算量について考察し，この部分を改良する方法について述べる．ある列挙木ノードの最汎パターン $<pat^k>$ とし，そこに含まれるあいまい文字領域のあいまい文字ドメインの平均サイズを l とし，許容誤差を r としよう．$0 \leq r \leq k/2$ のとき誤差直径 $d = 2r$ を満たし，$k/2 \leq r \leq k$ のとき $d = k$ を満たすので，あいまい文字領域は汎化配列パターン上に最大 k 個存在する．

式 (11) における $EVAL(MGP(SubMIS))$ の部分の計算量について考察しよう．$MGP(SubMIS)$ の部分の計算量は，$O(k \cdot |SubMIS|)$ である．最汎パターン $MGP(SubMIS)$ 上にあいまい文字領域が最大 k 個存在する．この点を考慮すると，$EVAL(MGP(SubMIS))$ の $EVAL$ の部分は k 個のあいまい文字ドメインの直積をとることにより計算されるので，$EVAL(MGP(SubMIS))$ の計算量は $O(k \cdot |SubMIS| + O(l^k))$ となる．

列挙木ノードの探索が進行すると，$MGP(SubMIS)$ のドメインサイズ l が大きくなるので，誤差直径 d が小さくても，式 (11) における $EVAL(MGP(SubMIS))$ の計算量 $O(k \cdot |SubMIS| + (l^d))$ が大きくなる．以下では，式 (11) における $EVAL(MGP(SubMIS))$ を高速化する方法について述べる．

5.1 の操作 1 からも明らかなように，子ノードの最汎パターン $<pat^k>$ は式 (11) の $MGP(SubMIS)$ に相当するが，これは，親ノードの最汎パターン $<pat_1{}^k>$ や親の兄弟ノードの最汎パターン $<pat_2{}^k>$ に少しの違いしかないので，$<pat^k>$ から $\{<pat_1{}^k>, <pat_2{}^k>\}$ をパターン切除すれば，$<pat^k>$ のインスタンス集合を直接調べる，すなわち負のインスタンスの存在を調べるよりもこのパターン切除の結果得られるパターン集合のインスタンス集合を調べる方が効率的である．これにより，式 (11) を次式のように改良できる．

$$EVAL(MCUT(<pat^k>, \{<pat_1{}^k>, <pat_2{}^k>\})) - MIS \neq \phi \quad (12)$$

以下では，$EVAL(MCUT(<pat^k>, \{<pat_1{}^k>, <pat_2{}^k>\}))$ の計算量について考察する．$<pat^k>$ は $<pat_1{}^k>$ や $<pat_2{}^k>$ とはわずかな違いしかないので，$<pat^k>$ は，$<pat_1{}^k>$ や $<pat_2{}^k>$ と比較し，最大 k 個のあいまい文字領域が増加すると考えることができる．

このとき，式 (9) を用いて式 (12) の $MCUT$ の計算量を求めると，$O(k \cdot l) + O(k \cdot k \cdot l)$ となる．第 1 項目は $<pat^k>$ から $<pat_1{}^k>$ をパターン切除するための計算量であり，このパターン切除により最大 $k-1$ 個のあいまい文字領域をもつ k 個の配列パターンが得られる．第 2 項目は最大 k 個の配列パターンから $<pat_2{}^k>$ を切除するための計算量である．$O(k \cdot l) < O(k^2 \cdot l)$ が成立するので，$MCUT$ の計算量は，$O(k^2 \cdot l)$ となる．$MCUT$ により最大 $k-2$ 個のあいまい文字領域をもつ最大 k^2 個の配列パターンが得られる．しかし，後述するが，$MCUT$ の計算量は無視できる大きさであることが分かる．

次に，$EVAL(MCUT(<pat^k>, \{<pat_1{}^k>, <pat_2{}^k>\}))$ の計算量を求める．$<pat^k>$ を $MGP(SubMIS)$ により生成するための計算量は，$O(k \cdot |SubMIS|)$ であった．$MCUT(<pat^k>, \{<pat_1{}^k>, <pat_2{}^k>\})$ は計算量 $O(k^2 \cdot l)$ をもち，最大 $k-2$ 個のあいまい文字領域をもつ最大 k^2 個の配列パターンを生成するので $EVAL(MCUT(<pat^k>, \{<pat_1{}^k>, <pat_2{}^k>\}))$ の計算量は，$O(k \cdot |SubMIS|) + O(k^2 \cdot l) + O(k^2 \cdot l^{k-2})$ となる．

以上により，式 (11) の計算量 $O(k \cdot |SubMIS|) + O(l^k)$ では $O(l^k)$ が大きな項となり，式 (12) の計算量 $O(k \cdot |SubMIS|) + O(k^2 \cdot l) + O(k^2 \cdot l^{k-2})$ では $O(k^2 \cdot l^{k-2})$ が大きな項となる．探索が進むにつれ，あいまい文字ドメインのサイズ l がパターン長 k よりも大きくなることを考慮すると，$O(l^k) > O(k^2 \cdot l^{k-2})$ であることが分かる．よって，式 (12) を利用する方がより効率的であるといえる．

6. 実験評価

表 1 に実験評価に用いた 10 種類のデータセットの特徴を示す．この 10 種類のデータセットは PROSITE [1] から取得したものである．最初に，10 種類のデータセットのそれぞれに対して，あいまいな問合せ処理を行う．次に，従来手法の反復精密化法，提案手法の段

表 1　データセット詳細
Table 1　Characteristics of the datasets.

データセット名	登録番号	データ件数	総長 (bytes)
Kringle	PS00021	90	59123
Homeobox	PS00027	1272	448596
ZincFinger	PS00028	1839	1146506
Kunitz	PS00280	224	58220
PTS_EIIA_1	PS00371	78	41476
PTS_EIIA_2	PS00372	50	15470
HTH_ASNC_1	PS00519	37	5766
Leucine	PS00717	28	13481
Adrenodxin	PS00814	35	5316
HTH_DEOR_1	PS00894	80	6339

階的一般化法の 2 種類の手法を用いて，あいまいな問合せ処理の結果として得られたミスマッチクラスタの最小汎化集合を抽出する．利用した計算機環境は，Intel Core i7-920（2.66 GHz），メモリ：3.0 GByte，HDD：1 TByte，OS：Fedora Core 10 である．

あいまいな問合せにおける検索文字は各データセットに含まれるモチーフの一部分を用いている．この検索文字列に可変長ワイルドカード領域が含まれる場合は，それを固定長ワイルドカード領域が含まれる部分文字列の集合に表現し，その集合内の要素を論理和で結合した条件で問合せを行っている．

また，あいまいな問合せにおける許容誤差半径は，各データセットのモチーフに含まれるあいまい文字の数，モチーフに含まれるあいまい文字の数+1，モチーフに含まれるあいまい文字の数 −1 の 3 種類を用いて実験を行った．ただし，モチーフに含まれるあいまい文字の数が 1 のデータセットについては，許容誤差半径を 1，2，3，*Homeobox* については問題の規模が大きいため，許容誤差半径を 6，7，8 と変化させ実験を行った．

6.1 実験結果

各データセットについて許容誤差半径ごとに，あいまいな問合せ処理で得られた類似部分文字列の数，汎化処理で得られた最小汎化集合の要素数，汎化処理に要した計算時間と性能比を表 2 に示す．計算時間は計 3 回の測定における平均値を示している．表 2 中のハイフン “-” は汎化処理を行うことができなかったことを示す．ここで，汎化処理を行うことができないとは，汎化処理を行うことでメインメモリが不足し，OS のスラッシングが始まることで実用的な時間で解くことが不可能であることを示す．なお，汎化処理を行うことができなかったすべての測定については，24 時間実行しても処理が終わらないことを確認している．また，二つの手法の出力結果が等しく，汎化処理で得られた最小汎化集合はあいまいパターン検索によって得られたミスマッチクラスタ内のすべての要素を含んでいることをプログラムで確認している．

表 2 に示すように，提案手法である段階的一般化法は 30 個の検索結果について，26 個の検索結果の汎化処理を行うことができたが，従来手法の反復精密化法は 7 個しか汎化処理を行うことができなかった．*Kringle*，*Homeobox*，*PTS_EIIA_1*，*PTS_EIIA_2*，*HTH_ASNC_1* と *HTH_DEOR_1* は，反復精密化法では，すべての検索結果，すなわちあいまいな問合せ処理の結果に対して汎化処理を行うことができなかった．段階的一般化法はミスマッチクラスタを構成する類似部分文字列の数が大規模なときに汎化処理を行うことができていないが，反復精密化法ではミスマッチクラスタを構成する類似部分文字列の数がそれほど多くなくても汎化処理を行うことができていない．この点については **6.2** で考察を行う．

Kunitz については，許容誤差半径が 1 のときは差がほとんどなく，許容誤差半径が 2 のとき 86 倍の高速化ができている．また，*Leucine* についても，許容誤差半径が 5 のときは差がほとんどなく，許容誤差半径が 6 のとき 20 倍の高速化ができている．*Adrenodxin* については，許容誤差半径が 3 と 4 では差がほとんどなく，5 のときには反復精密化法は汎化処理を行うことができなかった．計算時間として数値として比較ができなかった他の 19 個の検索結果についても，段階的一般化法は反復精密化法と比較して実用的な時間で汎化処理を終えることができるということで，高速化ができているといえる．

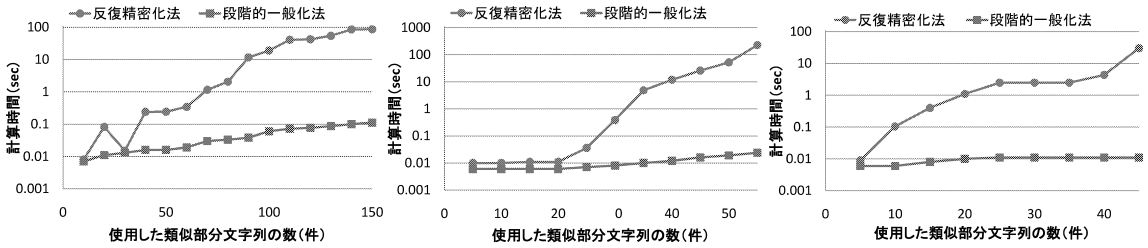
次に，反復精密化法と段階的一般化法の計算時間を数値として比較するために，人工的なミスマッチクラスタを使用した測定結果を示す．測定では，*Homeobox* の許容誤差半径 6，*Kunitz* の許容誤差半径 3，*Adrenodxin* の許容誤差半径 5 について，検索結果で得られた類似部分文字列を先頭から一部を取り出して汎化処理を行った．測定結果を図 4 に示す．

図 4 は，縦軸が計算時間で横軸が使用した類似部分文字列の件数である．グラフから，段階的一般化法は件数とともに徐々に計算時間が上昇しているが，反復精密化法については急激に計算時間が増えていることがいえる．各データセットともに件数が最も多いときに 1000 倍以上の性能差が出てきている．

Zinc Finger については，段階的一般化法は反

表 2　段階的一般化法と反復精密化法との性能比較
Table 2　Performance comparisons of the step-wise generalization method and the iterative refinement method.

データセット名	問合せ文字列	許容誤差半径	類似部分文字列の数（件）	最小汎化集合の要素数（件）	計算時間 反復精密化法（sec）: A	計算時間 段階的一般化法（sec）: B	性能比（A/B）
Kringle	<*YCRNx*(7, 8)*WC*>	3	586	69	-	0.073	-
		4	3577	918	-	19.875	-
		5	26704	-	-	-	-
Homeobox	<*LAx*(2)*LxLx*(4)*LRLWFxNx*(5)*R*>	6	1312	156	-	1.574	-
		7	2385	717	-	68.642	-
		8	11568	-	-	-	-
Zinc Finger	<*Cx*(2, 3)*Cx*(3)*Lx*(8)*Hx*(3, 5)*H*>	1	14513	42	0.076	5.363	0.014
		2	75026	-	-	-	-
		3	288587	-	-	-	-
Kunitz	<*YCRNx*(7, 8)*WC*>	1	287	4	0.003	0.004	0.75
		2	324	29	3.096	0.036	86
		3	1299	489	-	11.781	-
PTS_EIIA_1	<*Gx*(2)*LLLHLGLxTL*>	5	116	32	-	0.044	-
		6	447	127	-	0.815	-
		7	2258	692	-	33.751	-
PTS_EIIA_2	<*Dx*(6)*LGx*(2)*LALPHG*>	5	98	51	-	0.08	-
		6	469	295	-	3.793	-
		7	2229	1514	-	198.379	-
HTH_ASNC_1	< *Gx*(2)*DLGx*(2)*LGLSx*(2)*TLRKDL*>	9	463	219	-	13.526	-
		10	1262	670	-	157.335	-
		11	3019	1674	-	1148.196	-
Leucine	<*PLxLx*(2)*LAx*(2)*Lx*(2)*HxSTLSR*>	5	28	11	0.008	0.006	1.333
		6	31	15	0.28	0.014	20
		7	61	32	-	0.09	-
Adrenodxin	<*Cx*(2)*SxSCSTCH*>	3	35	7	0.003	0.004	0.75
		4	35	7	0.003	0.004	0.75
		5	58	28	-	0.02	-
HTH_DEOR_1	< *Rx*(3)*Lx*(3)*Lx*(16, 17)*Sx*(2)*TLRKDL*>	6	263	97	-	0.308	-
		7	1101	424	-	8.641	-
		8	5944	2018	-	284.116	-



(a) Homeobox（許容誤差半径 6）　(b) Kunitz（許容誤差半径 3）　(c) Adrenodxin（許容誤差半径 5）

図 4　類似部分文字列の数と計算時間の関係
Fig. 4　Correlation of the number of similar subsequences and processing time.

復精密化法と比較して計算時間が大きくなった．*Zinc Finger* については，二つの可変長ワイルドカード領域をそれぞれ複数の固定長ワイルドカード領域に置き換えることにより，9 通りの問合せ文字列を用いて実験を行ったが，それぞれの実行時間にばらつきが見られた．この結果については，**6.2** で考察を

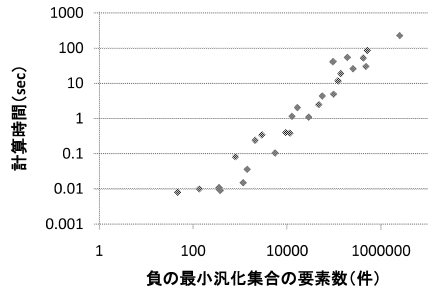


図 5　負の最小汎化集合の要素数と計算時間の関係（反復精密化法）
Fig. 5　Correlation of the number of elements of negative minimum generalized set and processing time. (The iterative refinement method)

行う．

6.2 考　　察

表 2 に示すように，段階的一般化法は反復精密化法と比較して多くの検索結果で汎化処理を行うことができた．また，図 4 に示すように段階的一般化法は反復精密化法と比較して高速化に成功しているといえる．ここでは，反復精密化法が段階的一般化法と比べて計算時間が必要となる理由と，逆に，*Zinc Finger* のように反復精密化法の方が高速になる場合についての考察を行う．

反復精密化法はトップダウンアプローチであるために，汎化処理の過程でミスマッチクラスタの最汎パターンからミスマッチクラスタを構成するすべてのインスタンスをパターン切除して求められる負の最小汎化集合を抽出する必要がある．抽出されれる負の最小汎化集合の要素数が増えていくと，負の最小汎化集合を抽出するための計算時間が急激に大きくなることが分かっている．

このことを示すために，図 5 に汎化処理の過程で抽出された負の最小汎化集合の要素数と計算時間の関係を示す．図 5 に示すグラフは図 4 のすべての測定結果について，抽出された負の最小汎化集合の要素数と計算時間を散布図にして示したグラフである．図 5 のグラフに示すように抽出される負の最小汎化集合の要素数が増えるごとに計算時間が急激に増加していることが分かる．

負の最小汎化集合の要素数は，ミスマッチクラスタを構成する類似部分文字列の数ではなく，類似部分文字列同士の類似性に関係する．そのために，類似部分文字列の数が少ないときにおいても抽出されるミスマッチクラスタの類似性が低い場合，負の最小汎化集合の要素数が爆発的に増え，反復精密化法では汎化処理を行うことができなかった．

Zinc Finger では，反復精密化法の方が高速であった．これは，ミスマッチクラスタを構成する類似部分文字列同士が非常に似通ったものであったことが原因であると考える．類似した類似部分文字列が多いために，抽出される負の負の最小汎化集合の要素数が抑えられため，計算時間が大きくならなかった．実際，問合せ文字列として $<Cx(2)Cx(3)Lx(8)Hx(4)H>$ を用いたとき，検索結果として返されたミスマッチクラスタ内には，他の場合に比べてかなり類似した類似部分文字列が多い．また，反復精密化法において抽出される負の最小汎化集合の要素数は 80 であり，図 5 のグラフから抽出される負の最小汎化集合の要素数が少ない場合は高速に問題を解くことができていることと結果が一致する．

一方，段階的一般化法はボトムアップアプローチであるためミスマッチクラスタを構成する類似部分文字列の数に計算時間が依存する．加えて，ミスマッチクラスタを構成する類似部分文字列同士の類似性が高まると部分列挙木の枝刈りができず，探索ノードが増える．*Zinc Finger* で許容誤差半径が 1 の場合，ミスマッチクラスタを構成する類似部分文字列同士が非常に似通ったものであったことと，その数が多いために，反復精密化法と比較して計算時間が大きくなったといえる．

7. む　す　び

本論文では，あいまいな問合せ処理の結果として返されるミスマッチクラスタの最小汎化集合の計算時間を短縮するため，段階的一般化法を提案した．提案手法の有効性を確かめるため，10 種類のデータセットを用いて実験を行った．許容誤差数を 3 種類変化させたあいまいな問合せ処理で得られた検索結果 30 個について，段階的一般化法は 26 個，反復精密化法は 7 個と，段階的一般化法は反復精密化法と比べてより多くの検索結果の汎化処理を行うことができた．

以下，今後の課題について述べる．

（1）　最小の大きさの MGS の探索

最小の大きさの MGS を探すという新たな問題に変更することが理想的である．この新たな問題の定義は，**3.6** の条件 (4) の関係式を $|MGS| \leq |MGS'|$ に変更すること，あるいは条件 (4) を定義から外す代わりに条件 (2) において，検出に手間のかかる冗長性を

導入することによって得られる．したがって，この新たな問題を解くには，例えば，**3.4** の後半で定義した冗長性の概念を次のように拡張する必要がある．二つの k-汎化配列パターン集合 S_1 及び S_2 に対して，$EVAL(S_1) \supseteq EVAL(S_2)$ が成立するとき，S_2 は S_1 に冗長であると定義する．そして，この定義に基づいた冗長性の除去を行うには，図 3 の (4) の候補集合 $Candidate$ を MGS' とおき，**5.1** の操作 5 を以下の処理に変更する必要がある．まず，MGS' を用いてべき集合 $P(MGS')$ を作る．このとき，$m = |MGS'|$ とすると，$P(MGS')$ の要素数は 2^m となる．次に，$P(MGS') - S_x$ に冗長な集合 $S_x \in P(MGS')$ のすべてを MGS' から取り除くことによって MGS を得る．ただし，集合 S_x は空集合ではないとする．以上により，この新たな問題を解くには，$O(m2^m)$ の計算量が必要になる．しかし，変更前の **5.1** の操作 5 では MGS' に存在する任意の 2 要素 $< {pat_1}^k >, < {pat_2}^k >$ を選択し，冗長な要素を除去していたため，$O(m^2)$ の計算量で済んでいた．$O(m^2)$ の計算量は，$O(m2^m)$ よりもはるかに少ない．したがって，この新たな問題を効率的に解くために，$O(m2^m)$ の計算量をどのように減らすかという課題が残されている．

（2） 可変長ワイルドカード領域の抽出

可変長ワイルドカード領域とあいまい文字領域を同時に求める頻出パターンを抽出するために，Modified PrefixSpan 法を実行した後に，段階的一般化法を実行するのは適切ではない．段階的一般化法ではミスマッチクラスタに含まれる部分文字列は頻出であるか否かにかかわらずすべてを汎化の対象にしている．PrefixSpan や Modified PrefixSpan は，頻繁に出現する部分文字列パターンを抽出する方法であるので，支持数の小さなモチーフ配列は見落としてしまう．また，最小支持数を下げ，小さな支持数をもつモチーフ配列を抽出しようとすると，PrefixSpan や Modified PrefixSpan では，メモリ領域不足となり，事実上解けないことが既に報告されている [23]．したがって，可変長ワイルドカード領域とあいまい文字領域を同時に求める頻出パターンを抽出するには，少なくとも Modified PrefixSpan 法と段階的一般化法とのそれぞれの仕組みを考慮した新アルゴリズムの研究が必要である．

（3） ドメイン分割法の併用

本論文では，汎化の処理スピードに着目したため，アミノ酸の特徴を考慮したドメイン分割法の併用 [18] については，段階的一般化法にも反復精密化法にも組み込まれていない．今後は，どちらの手法にもドメイン分割法を併用した場合の計算性能を調べる研究が残されている．

（4） その他

3.1 のあいまいな問合せ処理では，ハミング距離に基づいて検索文字列 K と類似する部分文字列 K' を検索し，ミスマッチクラスタを作成したため，どの部分配列も同じ長さであった．ハミング距離の代わりに編集距離 [11] を利用すると，ミスマッチクラスタの中には，K に比べ長いものや短いものが含まれるので，本提案方式を利用するには，ミスマッチクラスタの全要素に対してマルチプルアラインメントが必要になる．このマルチプルアラインメントが全体性能にどのような影響があるのかについての検討が残されている．

謝辞 本研究の一部は，日本学術振興会，科学研究費補助金（基盤研究 (C)，課題番号：20500137）の支援により行われた．

文　献

[1] http://kr.expasy.org/prosite/
[2] http://www.sanger.ac.uk/software/pfam/
[3] E. Sonnhammer, S. Eddy, and R. Durbin, “Pfam: A comprehensive database of protein domain families based on seed alignments,” Proteins, Structure function and genetics, vol.28, pp.405–420, 1997.
[4] 室田誠逸，佐藤靖史，澁谷正史，戸井雅和，“血管新生研究の新しい展開，” 月刊治療学，vol.34, no.4, pp.77–89, April 2000.
[5] F. Castellino and J. Beals, “The genetic relationships between the kringle domains of human plasminogen, prothrombin, tissue plasminogen activator, urokinase, and coagulation factor xii,” J. Molecular Evolution, vol.26, pp.358–369, 1987.
[6] 高橋　敬，“凝固・線溶系のドメイン進化，” ゲノムから見た生物の多様性と進化，五條堀孝（編），pp.103–114, シュプリンガー・フェアラーク東京，2003.
[7] K. Ikeo, K. Takahashi, and T. Gojobori, “Tdifferent evolutionary histories of kringle and protease domains in serine proteases: A typical example of domain evolution,” J. Molecular Evolution, vol.40, pp.331–336, 1995.
[8] Z. Galil and K. Park, “An improved algorithm for approximate string matching,” SIAM J. Comput., vol.19, no.6, pp.989–999, 1990.
[9] G. Myers, “A fast bit-vector algorithm for approximate string matching based on dynamic programming,” J. ACM, vol.46, no.3, pp.395–415, 1999.
[10] J.Y.S. Park, W.W. Chu, and C. Hsu, “Efficient searches for similar subsequences of different lengths in sequence databases,” Proc. 16th International

Conference on Data Engineering (ICDE2000), pp.23–32, IEEE Computer Society, 2000.
[11] F. Shi and C. Mefford, “A new indexing method for approximate search in text databases,” Proc. 2005 The Fifth International Conference on Computer and Information Technology (CIT’05), pp.70–76, IEEE Computer Society, 2005.
[12] C.L.L. Jin, N. Koudas, and A.K.H. Tung, “Indexing mixed types for approximate retrieval,” Proc. 31st VLDB Conference, pp.793–804, 2005.
[13] M.-F. Sagot, “Spelling approximate repeated or common motifs using a suffix tree,” LATIN, pp.374–390, 1998.
[14] L. Marsan and M.-F. Sagot, “Extracting structured motifs using a suffix tree - algorithms and application to promoter consensus identification,” Proc. Fourth Annual International Conference on Computational Molecular Biology (RECOMB2000), pp.210–219, 2000.
[15] P.A. Pevzner and S.-H. Sze, “Combinatorial approaches to finding subtle signals in dna sequences,” ISMB, pp.269–278, 2000.
[16] E. Eskin and P.A. Pevzner, “Finding composite regulatory patterns in dna sequences,” Proc. 10th International Conference on Intelligent Systems for Molecular Biology, pp.354–363, 2002.
[17] K. Araki, K. Tamura, T. Kato, Y. Mori, and H. Kitakami, “Extraction of ambiguous sequential patterns with least minimum generalization from mismatch clusters,” The Third International Conference on Signal-Image Technology and Internet-Based Systems, pp.32–39, IEEE Computer Society Press, 2007.
[18] 荒木康太郎，田村慶一，加藤智之，北上　始，日本データベース学会論文誌 (DBSJ Letters), vol.6, no.3, pp.5–8, 2007.
[19] H. Kimura, H. Kitakami, K. Araki, and K. Tamura, “A stepwise generalization mthod for extracting minimum generalized set from mismatch cluster,” Proc. 2008 International Conference on Bioinformatics and Computational Biology (BIOCOMP’08), vol.II, pp.998–1004, 2008.
[20] E.Y. Shapiro, “Inductive inference of theories from facts,” Computational Logic — Essays in Honor of Alan Robinson, pp.199–254, 1991.
[21] S. Muggleton, “Inverse entailment and progol,” New Generation Comput., vol.13, no.3 & 4, pp.245–286, 1995.
[22] U.M. Fayyad, G. Diatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, AAAI Press/The MIT Press, 1996.
[23] 加藤智之, 北上　始, 森　康真, 田村慶一, 黒木　進, “極小かつ非冗長な可変長ワイルドカード領域を持つ頻出パターンの抽出,” 信学論（D），vol.J90-D, no.2, pp.281–291, Feb. 2007.
[24] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, “Mining sequential patterns by pattern-growth: The prefixspan approach,” IEEE Trans. Knowl. Data Eng., vol.16, no.11, pp.1424–1440, 2004.

（平成 21 年 6 月 5 日受付，10 月 5 日再受付）

田村　慶一

1998 九大・工・情報工学卒．2000 同大大学院システム情報科学研究科知能システム学専攻修士課程了．2003 同大学院システム情報科学府知能システム学専攻博士後期課程単位取得の上，満期退学．博士（情報科学）．2002 年広島市立大学情報科学部助手，現在，広島市立大学大学院情報科学研究科講師．並列処理，データ工学に関する研究に従事．情報処理学会，IEEE CS 各会員．

木村　浩明

2008 広島市立大・情報科学・知能情報システム卒．現在，(株）コア中四国カンパニーに所属．

荒木康太郎

2006 広島市立大・情報科学・知能情報システム卒．2008 同大大学院情報科学研究科知能情報システム工学専攻博士前期課程了．現在，(株）日立製作所に所属．情報処理学会，日本データベース学会各会員．

北上　　始　（正員）

広島市立大学大学院情報科学研究科教授．1976 東北大学大学院工学研究科博士前期課程了．同年富士通（株）入社．以後，富士通研究所，新世代コンピュータ技術開発機構，国立遺伝学研究所客員助教授を経て，1994 広島市立大学情報科学部教授，2007 広島市立大学大学院情報科学研究科教授，現在に至る．データマイニング，生命情報学，知識ベースなどの教育研究に従事．博士（工学）．情報処理学会 25 周年記念論文，日本工学教育協会論文論説賞，本会 DASFAA2010 組織委員，日本データベース学会論文誌編集委員，情報処理学会（TOM）論文誌編集委員，情報処理学会一般情報処理教育委員会委員，人工知能学会，IEEE，ACM 各会員．