曖昧検索に基づく最小汎化パターンの抽出法

荒木康太郎† 田村 慶一†† 加藤 智之† 黒木 進†† 北上 始††

† 広島市立大学大学院情報科学研究科 〒 731–3194 広島市安佐南区大塚東三丁目 4 番 1 号
†† 広島市立大学情報科学部 〒 731–3194 広島市安佐南区大塚東三丁目 4 番 1 号
E-mail: †{kotaro,kato}@db.its.hiroshima-cu.ac.jp, ††{ktamura,kuroki,kitakami}@its.hiroshima-cu.ac.jp

あらまし 配列データベースに対して曖昧文字表現を含む頻出配列パターンを抽出する方法を提案する．この提案手法は，曖昧検索と，最小汎化を行う処理の 2 段階で構成される．曖昧検索の段階では，ディスク上のサフィックス木を用いて，ある基準となる長さ k の検索パターンに対して，最小支持数 $Mini\_sup$ を満たし，許容誤差半径 r 内にあるすべての $k-$ 部分文字列（長さ k の部分文字列）が正の部分文字列集合として取得される．最小汎化では，正の部分文字列集合から計算された負の最小汎化パターン集合を用いて，最汎パターンを精密化することにより，求めるべき正の最小汎化パターンを計算する．提案手法の有効性を確認するため，2 種類のデータセットを用いて実験を行なった．その結果，曖昧検索で得られた大量の $k-$ 部分文字列集合を少ない汎化パターンで表現できることがわかった．また，本手法によりデータセットに含まれるモチーフの一部分を表現する汎化パターンを取り出すことに成功した．

キーワード データマイニング，配列データベース，バイオインフォマティックス，正規表現

Extraction of sequential patterns with least minimum generalization based on ambiguous retrieval

Kotaro ARAKI†, Keiichi TAMURA††, Tomoyuki KATO†, Susumu KUROKI††, and Hajime KITAKAMI††

† Graduate school of Information Sciences, Hiroshima City University 3–4–1 Ozuka-Higashi, Asa-Minami-Ku, Hiroshima, 731–3194 Japan
†† Faculty of Information Sciences, Hiroshima City University 3–4–1 Ozuka-Higashi, Asa-Minami-Ku, Hiroshima, 731–3194 Japan
E-mail: †{kotaro,kato}@db.its.hiroshima-cu.ac.jp, ††{ktamura,kuroki,kitakami}@its.hiroshima-cu.ac.jp

Abstract We propose a method for extracting frequent sequence patterns including the expression of ambiguous characters from sequence databases. The proposed method is composed of two stages that are ambiguous retrieval allowing mismatch characters and least minimum generalization for the results of the retrieval. At the stage of the ambiguous retrieval, the set of positive $k-$length subsequences for a retrieval key given by user are selected from the disk based-suffix tree on the condition that the each subsequence satisfies a minimum support, $Mini\_sup$, and permissible error margin radius, r. At the stage of the least minimum generalization, the least minimum patterns are acquired by refining a most general pattern using the set of negative $k-$length patterns. The set of negative $k-$length patterns are computed by refining a most general pattern using the set of positive $k-$length patterns. We experiment by using some data sets to confirm the usefulness of the proposed method. As a result, we confirmed that a large amount of $k-$length character strings selected by the first stage is represented by few generalization patterns. Moreover, we succeeded in extracting the generalization pattern that represented a part of the motif included in the dataset by this method.

Key words Data Mining, Sequence Database, Bioinformatics, Regular Expression

1. はじめに

アミノ酸配列やテキスト情報などを含むデータベースに対する配列データマイニング [1]〜[5] では，正規表現として表現された頻出配列パターンを抽出する方法が注目されている．例えば，生物の進化の過程で保存されてきているモチーフは，蛋白

質の生物学的機能をもつと考えられており，アミノ酸文字 (アミノ酸を表す 20 種類のアルファベット文字) によって正規表現として表現されている．現在までに発見されたモチーフは PROSITE データベースに登録され管理されている．その例である，Basic-leucine zipper (bZIP) モチーフ (PS00036) は以下のように表現されている．

$< [KR] - x(1,3) - [RKSAQ] - N - \{VL\} - x - [SAQ](2) - \{L\} - [RKTAENQ] - x - R - \{S\} - [RK] >$

[RKSAQ] は，R，K，S，A，Q の中の文字であれば，どれかのアミノ酸文字が対応することを示している．{VL} は，他の曖昧文字表現であり，この中に現れない 1 文字のアミノ酸文字が対応することを示している．曖昧文字 [KR] と曖昧文字 [RKSAQ] の間の記号 x(1,3) は，ワイルドカード文字の許容数を現しており，この場合は，曖昧文字 [KR] と曖昧文字 [RKSAQ] の間に 1 文字から 3 文字までのワイルドカードが許されていることを示している．また，要素 [SAQ] の後の () 内の数字は，その要素の繰り返しを表現している．

従来の配列データマイニング研究 [1]〜[5] では，ワイルドカードを含む頻出配列パターンの抽出法が盛んに研究されてきた．しかしながら，PROSITE に見られるような曖昧文字を含む頻出配列パターンの抽出法については，まだ，十分な研究が行われていない．

このため，我々は，n 件の配列データベースから，曖昧文字表現を含む配列パターンを抽出する方法について着目している．入力データとしては，n 件からなる配列データベース，パターン長 k，最小支持数 $Mini\_sup$，許容誤差半径 r が与えられる．これらの条件を満たす曖昧な頻出配列パターンを抽出するためには，高速な曖昧パターン検索と検索によって取得された部分文字列集合から最小汎化された正規表現を導出しなければならない．

古くから，数多くの曖昧検索の研究 [6], [7] が行われてきた．しかし，これらの研究では，配列データマイニングで扱われる支持数や曖昧表現について考慮されていないため，これらを曖昧な頻出配列パターンの抽出に役立てることはできない．

近年，SPELLER [8]，WINNOWER [9]，MITRA [10] といったアルゴリズムは，この問題を解決する方向性が示されている．

SPELLER は配列データベースに対するサフィックス木という構造を内部メモリー上で保持しており，WINNOWER および MITRA はそのような構造をもたず，配列データベースを長さ k の部分文字列に分割した集合 (ブロックデータ構造と呼ばれる) を内部メモリー上に保持している．SPELLER は，検索キー集合の各要素を (中心) 部分文字列とし，サフィックス木をルートから下方にたどることにより，その基準から許容誤差半径 r 以内にある部分文字列をすべて抽出している．

WINNOWER および MITRA の両者は，$k-$ 部分文字列をグラフ上のノードとみなし，完全グラフをすべて見つけ出すアプローチを採用している．この完全グラフが，抽出すべき $k-$ 部分文字列の集合である．WINNOWER は，基準となる検索キーを設定せず，集合内の $k-$ 部分文字列どうしを比較し，その誤差が誤差直径 $d = 2r$ 以内であれば，2 つの $k-$ 部分文字列（ノード）を接続する．ただし，同じ配列データ中の $k-$ 部分文字列どうしは比較を行わない．すべての組合せの比較を終了結果として得られたグラフをもとに，最小支持数を満たす完全グラフだけを見つけ出す．MITRA は，WINNOWER の処理を効率化するために，完全グラフを計算する前に，完全グラフの中心となる基準部分文字列を検索キーとして設定し，グラフ分割を行っている．このグラフ分割を用いたアルゴリズムは,MTRA-グラフと呼ばれており，基準となる検索パターンを生成し，許容誤差半径 r 以内にある $k-$ 配列パターンの集合 (どの要素も互いに誤差直径 $d = 2r$ 以下の誤差を持つ) を効率的に見つけ出している．

しかしながら，これらのどの研究においても，互いの誤差が誤差直径 $d = 2r$ 以内にある $k-$ 部分文字列の集合のすべてを求めるだけにとどまっており，PROSITE で見られるような曖昧配列パターンの正規表現を計算するまでには至っていないという問題がある．また，Sagot [8] が用いたサフィックス木は，内部メモリーに基づいたものであり，高速処理が可能だが，大きな配列データベースには向いていないという問題がある．

以上の問題を解決するために，本論文では，Cheung らが開発した DynaCluster [11] と呼ばれるアルゴリズムを用いてサフィックス木をディスク上に構築し，基準となる長さ k の検索キーに対して，最小支持数を満たしかつ許容誤差半径 r 内にある $k-$ 部分文字列の集合を高速に検索し，検索結果として得られた集合から最小汎化された曖昧パターンの正規化表現を効果的に導出する方法を提案する．

2. 用語と問題の定義

配列データベース $DB = \{t_1, t_2, \cdots, t_n\}$ において，各要素は，(sid, SEQ_{sid}) と表現される (n は要素数，sid は配列識別子)．配列データベースの配列識別子の集合は $\Omega = \{1, 2, 3, \cdots, n\}$ と表現する．各 SEQ_{sid} は，配列識別子の値として sid を持つ配列データであり，あるアルファベット Σ 上で定義される文字列である．配列 SEQ_{sid} の先頭から j 番目の文字は，$SEQ_{sid}[j]$ として表される．表 1 は，$DB = \{t_1, t_2, t_3, t_4, t_5\}, \Omega = \{1, 2, 3, 4, 5\}, t_1 = (1, FKYAKWLCDN), t_2 = (2, SFVKTAEHNQC), t_3 = (3, ALR), t_4 = (4, MSKPL), t_5 = (5, FSKFLMAWEH)$ であることを示している．表 1 の例で言うと，アルファベット文字 A は，t_1, t_2, t_3, t_5 に含まれているので，$SEQ_1[4] = SEQ_2[6] = SEQ_3[1] = SEQ_5[7] =$ “A” と表現することができる．配列データのアルファベット文字数が k 個であるなら，その配列データを k-配列データと呼び，最初の配列は 10-配列データである．

ここでは，n 本の配列に対するサフィックス木を構築するため，予め，n 本の配列 $SEQ_1, SEQ_2, \cdots SEQ_n$ をつなげ，1 本の統合配列 $S =< s_1 s_2 \cdots s_n >$ を作成する．ただし，s_i は配列 SEQ_i の末尾にその配列識別子 (記号 $\#_i$ で表現する) が付与された配列を表す．これにより，1 本の統合配列 S を対象としたサフィックス木を構築している．このとき，S_i を i 文字目から最初に配列識別子が現れるまでの間

表 1　配列データベース

sid	配列データ
1	FKYAKWLCDN
2	SFVKTAEHNQC
3	ALR
4	MSKPL
5	FSKFLMAWEH

に対応するサフィックスとする．i をサフィックス番号と呼ぶ．例えば，統合配列 $S =< AAC\#_1CAA\#_2 >$ であるとき，$S_1 =< AAC\#_1 >, S_2 =< AC\#_1 >, S_5 =< CAA\#_2 >$ となる．また，サフィックス S_k に対して長さ l の部分文字列 $S_k^l =< s_k s_{k+1} \cdots s_{k+l-1} >$ をサフィックス S_k における長さ l のプレフィックスと呼ぶ．上記の例では，$S_1 =< AAC\#_1 >$ であり，$S_1^2 =< AA >$ である．

2.1　曖昧な配列パターン

α_i をアルファベット Σ の要素（1 文字）とすると，$k-$ パターン（k 個のアルファベット文字を並べたパターン）は，以下の形式で表現する．

$$< pat^k >=< \alpha_1 - x(i_1, j_1) - \alpha_2 - x(i_2, j_2) - \cdots - x(i_{k-1}, j_{k-1}) - \alpha_k > \quad (1)$$

式 (1) 中の，x(i,j) は，ワイルドカード領域と呼ばれ，ワイルドカード数が i 個から j 個までの範囲内であることを示している．$i < j$ のとき，x(i,j) を可変長ワイルドカード領域と呼び，$i = j$ のとき，x(i,j) は x(i) と書き，x(i) は，固定長ワイルドカード領域と呼ぶ．x(0) は空文字を表現する．また，ε =j-i をワイルドカード領域 x(i,j) の誤差と呼ぶ．ある $k-$ パターン中の全てのワイルドカード領域が固定長であるとき，つまり，誤差を含まないパターンを固定長パターンと呼ぶ．少なくとも 1 つの可変長ワイルドカードを持つパターン，つまり，誤差を含むパターンを，可変長パターンと呼ぶ．式 (1) の α_i が配置されている位置に，あるアルファベット Σ_i 内に存在する任意の 1 文字の配置を許すとき，式 (1) を以下のように拡張表現する．

$$< pat^k >=< \Sigma_1 - x(i_1, j_1) - \Sigma_2 - x(i_2, j_2) - \cdots - x(i_{k-1}, j_{k-1}) - \Sigma_k > \quad (2)$$

我々は，式 (2) で表現されるパターンを曖昧な配列パターンと呼ぶ．例えば，以下の bZIP モチーフの正規表現について考えてみよう．

$$< [KR] - x(1,3) - [RKSAQ] - N - \{VL\} - x - [SAQ](2) - \{L\} - [RKTA\ \ ENQ] - x - R - \{S\} - [RK] > \quad (3)$$

[KR]，[RKSAQ]，N は，それぞれ，$\Sigma_1 = \{K, R\}$,$\Sigma_2 = \{R, K, S, A, Q\}$,$\Sigma_3 = \{N\}$ を意味する．以下では，曖昧表現に関する本質を浮き彫りにするため，ワイルドカード領域の部分の記述を省略することがある．

2.2　汎化パターン

長さ k の部分文字列上の各文字位置に配置される文字の種類を $\Sigma_1, \Sigma_2, \cdots, \Sigma_k$ とする．ただし，Σ_i は，正の $k-$ 部分文字列集合の i 番目の文字位置から見つけ出された文字の集合とする $(1 \leqq i \leqq k)$．これらから構成される汎化パターン $\Sigma_1\Sigma_2 \cdots \Sigma_k$ を正の $k-$ 部分文字列集合に対する最汎パターンと呼ぶ．検索結果として取得された $k-$ 部分文字列を正の $k-$ 部分文字列と呼び，最汎パターンを構成するために利用される $k-$ 部分文字列の中で，正の $k-$ 部分文字列ではない部分文字列を負の $k-$ 部分文字列と呼ぶ．Σ_i の要素数を $|\Sigma_i|$ で表現すると，負の $k-$ 部分文字列集合の要素数は，以下のとおりである．

$$|\Sigma_1\Sigma_2 \cdots \Sigma_k| - |\text{ 正の } k - \text{ 部分文字列集合 }| \quad (4)$$

$\Sigma = \Sigma_1 = \Sigma_2 = \cdots = \Sigma_i$ とすると，$\Sigma_1\Sigma_2 \cdots \Sigma_k = |\Sigma|^k$ となるので，負の $k-$ 部分文字列集合の要素数は，k の増加と共に指数関数的に増加する．正の $k-$ 部分文字列の集合が与えられたとき，これを被覆する汎化パターンのなかで，どんな負の $k-$ 部分文字列も被覆しない最大のパターンを正の最小汎化パターンと呼ぶ．

2.2.1　問題の定義

ここで扱う問題は，配列データベース $DB = \{t_1, t_2, \cdots t_n\}$ に対して，基準となる $k-$ 部分文字列を検索キー P として与え，入力パラメータとして与えられる最小支持数 $Mini\_sup$，許容誤差半径 r を満たす曖昧表現をもつ頻出配列パターンを抽出する．頻出パターンとは，最小支持数以上の支持数をもつパターンをさす．例えば，検索キーとして $< FSK >$ の長さ 3 のパターンが与えられ，最小支持数 3，許容誤差半径 1 であるとすると，表 1 の配列データベースからは，検索キーに対して許容誤差半径 1 以内である $< FVK >, < MSK >, < FSK >$ の 3 件の部分文字列が抽出されるので最小支持数を満たす．これらをまとめると，$< [FM]SK >: 2, < F[SV]K >: 2$ が得られる．この問題を解くために，以下の処理を行う必要がある．

(1) 基準となる $k-$ 部分文字列に対して，高速な曖昧検索を行い，誤差半径 r 内にある $k-$ 部分文字列の集合を全て求める (集合内の各要素は互いに誤差直径 d=2r 内にある)．

(2) 検索結果として得られた $k-$ 部分文字列の集合に対して，最小汎化された曖昧パターン表現を導出する．

3.　高速な曖昧パターン検索

曖昧検索を高速に行うために，ディスク上のサフィックス木を用いている．これによる曖昧検索は，ディスク上に構築したサフィックス木を深さ優先にスキャンしていくことで達成している．スキャンするサフィックス木の深さは最大で検索パターンの長さ k と同じである．以下では，使用したディスク上のサフィックス木の構築について簡単に説明する．

3.1　ディスク上のサフィックス木構築

我々はディスク上のサフィックス木の構築アルゴリズムとして DynaCluster アルゴリズム [11] を用いている． DynaCluster アルゴリズムは動的クラスタリング技術を用いている．この技術では，距離の近いノードは同ページ，もしくは隣接したページに格納され，ノード挿入時に起こる $edge - splitting$ と呼ばれる枝の分割によるディスクページアクセスの数を抑える工夫

がされている．

DynaCluster アルゴリズムは配列 1 本を対象として，サフィックス木を構築するアルゴリズムであるので，n 本の配列データベースを処理するには不向きである．この問題を解決するため，ここでは，配列データベースに含まれる n 本の配列データを 1 本につなげた統合配列 S を作成し，S に対するサフィックス木を構築する．DynaCluster では，構築されるサフィックス木の深さとノードの種類をユーザが自由に制御できるようにするために，整数を用いて，任意長の文字列を単位とする符号化を可能にしている．我々は，この符号化の単位を 1 文字に固定し，20 種類のアミノ酸文字からなる文字列の符号化を行っている．

3.1.1 サフィックス木の構築

以下では，サフィックス木のある親ノードに着目するとき，その親ノードとそれにつながる子ノードのすべてをクラスタと呼ぶ．サフィクス木を深さ優先に構築するために必要な情報は，接頭辞データベース PDB と呼ばれるテーブルに格納される．接頭辞データベース PDB はクラスタごとに作成される．ルートノードから深さ k のあるノードまでの間に出現する $k-$ 部分文字列を $< string^k >$ とすると，$k-$ 部分文字列 $< string^k >$ に対する接頭辞データベース PDB($< string^k >$) には，その次に現れる文字 $\alpha \in \Sigma$ の集合，各文字に対応する数値符号，各文字 α が存在する全ての位置リスト PList の 3 種類の情報が格納されている．また，PList には，各 $(k+1)-$ 部分文字列 $< string^k - \alpha >$ が存在する統合配列のサフィックス番号 i と S_i に含まれる配列識別子 sid の組が格納されている．

サフィックス木の構築過程において，クラスタごとに作成される PList の要素数はサフィックス木のリーフノードに近くなると急激に減少する．このことから DynaCluster では，予め最適な PList の要素数を閾値 τ として実測し，その値を用いて終端クラスタを作成している．終端クラスタでは，PDB を用いずに，未処理の部分文字列どうしを比較しながら部分木が構築される．閾値 τ を大きくすると終端クラスタに分割が発生する頻度が高まり，小さくすると終端クラスタの数が増える．DynaCluster の文献によれば，閾値 $\tau = 1024$ が性能上最適であると報告されているので，本稿でもそれを採用する．サフィックス木の構築手順は以下のとおりである．

(1) 配列データベース DB に含まれる n 本の配列データを 1 本につなげ，統合配列 S を作成する．次に，統合配列 S に対するサフィックス S_i を全て作成する $(1 \leqq i \leqq |S|)$．これらのサフィックスの集合を $T = \{S_1, S_2, \cdots, S_{|S|}\}$ とする．$k := 0$ とする．

(2) ルートクラスタを作成するために，T に含まれる各サフィックス S_i における長さ 1 のプレフィックス (先頭 1 文字) S^1_{i+k} を子ノードとし，それらの親ノードに相当するルートノードを作成する $(1 \leqq i \leqq |S|)$．このとき，サフィックス木を深さ方向に成長させるために必要な PDB を作成する．また，あるプレフィックス S^1_i であるノードに対して，PList に含まれる配列識別子の種類の数がそのノードの支持数となる．さらに，$k := k + 1$ とする．

(3) PDB に含まれる各アルファベット文字に対して以下の処理を行う．

if PList に含まれる要素数 > 閾値 τ then do;
PList に含まれる各サフィックス番号 i に対して，S^1_{i+k} を子ノードとする非終端クラスタとその PDB を作成する．また，あるプレフィックス S^1_{i+k} であるノードに対して，PList に含まれる配列識別子の種類の数がそのノードの支持数となる．
$k := k + 1$ とし，再帰的に (3) の処理を行う．
end
else 終端クラスタを作成する；

(4) 処理を終了する．

3.1.2 構　築　例

サンプルデータとして配列データベース $t_1 =< 1, AAC >, t_2 =< 2, CAA >$ を用いてサフィックス木構築を説明する．この例では閾値 $\tau = 2$ とする．まず，前節の (1) の処理によって，統合配列 $S =< AAC\#_1CAA\#_2 >$ が得られる．また，統合配列 S に対するサフィックス S_i を全て作成する $(1 \leqq i \leqq 8)$．次に前節の (2) の処理によって，ルートクラスタを作成する．

図 1a に示されるように，ルートノードを生成し，ナンバーを 0 とする．次に T に含まれる各サフィックスのプレフィックス S^1_i を子ノードとして作成する．最初に，サフィックス $S_1 =< AAC\#_1 >$ を扱う．プレフィックスは $S^1_1 =< A >$ であるので，ルートノードの子として二重丸のノードを生成し，ラベルを $< A >$ とする．同時に，サフィックス S_1 のサフィックス番号とその配列識別子の組 (1,1) が PDB のアルファベット文字 $< A >$ に対応する PList に挿入される (図 1b)．2 番目に，サフィックス $S_2 =< AC\#_1 >$ を扱う．プレフィックス $S^1_2 =< A >$ であり，$< A >$ はすでにルートクラスタに現れているので，ノードを生成する必要がない．よってサフィックス番号と配列識別子の組 (2,1) を PDB のアルファベット文字 $< A >$ に対応する PList に追加するだけである．3 番目に，サフィックス $S_3 =< C\#_1 >$ を扱う．プレフィックスは $S^1_3 =< C >$ であるので，ルートノードの 2 番目の子として二重丸のノードを生成し，ラベルを $< C >$ とする．同時に，サフィックス S_3 のサフィックス番号とその配列識別子の組 (3,1) を PDB のアルファベット文字 $< C >$ に対応する PList に追加する．4 番目に，サフィックス $S_4 =< \#_1 >$ を扱うが終端記号のみの部分文字列であるので，無視する．続いて，サフィックス S_5 として $< CAA\#_2 >$ を扱う．プレフィックス $S^1_5 =< C >$ であり，$< C >$ はルートクラスタに現れているので，ノードを生成する必要がない．よって動作としてはサフィックス番号と配列識別子の組 (5,2) を PDB のアルファベット文字 $< C >$ に対応する PList に追加するだけである．以下すべてのサフィックスに対して同様の処理を行うと，ルートクラスタとそのクラスタにおける PDB は図 1a, 図 1b に示しているようになる．また $< A >$, $< C >$ のノードの支持数は PDB の PList より，共に 2 である．ここで，次のクラスタ生成のため，$k = k + 1$ とする．

次に，(3) の処理を適用することで，深さ方向にクラスタを生成していく．アルファベット文字 $< A >$ に対する PList に含まれる要素数 $4 > \tau$ であることから，ノード $< A >$ を親ノード

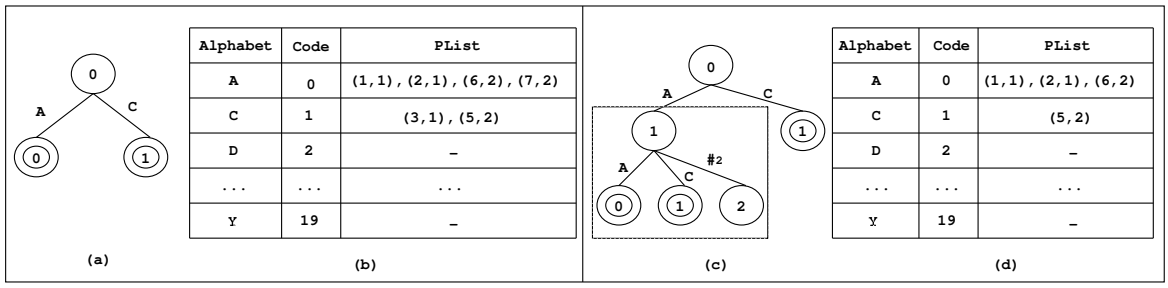

図 1　$S =< AAC\#_1CAA\#_2 >$ に対する最初の 2 つのクラスタ生成

とした非終端クラスタを作成する．アルファベット文字 $< C >$ に対する PList に含まれる要素数 $2 \leqq \tau$ であることからノード $< C >$ を親ノードとした終端クラスタを作成する．図 1c のクラスタ (点で囲まれた四角形) は，図 1b のアルファベット文字 $< A >$ に対して処理を行なったものである．以下はアルファベット文字 $< A >$ に対する非終端クラスタの生成を示す.

PList に含まれる各サフィックス番号 i に対してプレフィックス S^1_{i+k} を子ノードとする非終端クラスタとその PDB を作成する．まず，S_1 に対して，プレフィックスは $S^1_{1+1} =< A >$ である．$< A >$ はクラスタに現れていないので，ナンバー 0 として二重丸のノードを生成する．同時に，サフィックス番号とその配列識別子の組 (1,1) を PDB のアルファベット文字 $< A >$ に対応する PList に追加する．2 番目に，S_2 に対して，プレフィックス $S^1_{2+1} =< C >$ はクラスタに現れていないので，ナンバー 1 として二重丸のノードを生成する．同時に，サフィックス番号とその配列識別子の組 (2,1) を PDB のアルファベット文字 $< C >$ に対応する PList に追加する．3 番目に，S_6 に対して，プレフィックス $S^1_{6+1} =< A >$ はすでにクラスタに現れているので，ノードを生成する必要がない．よって，サフィックス番号と配列識別子の組 (6,2) を PDB のアルファベット文字 $< A >$ に対応する PList に追加する．4 番目に，S_7 に対して $S^1_{7+1} =< \#_2 >$ を処理する．これは終端記号であるので，ラベルを配列識別子として葉ノードを生成する．PDB への追加はしない．ここで，親クラスタにおける PDB のアルファベット文字 $< A >$ に対する PList に含まれる全てのサフィックスの処理を行ったのでこのクラスタの処理は終了する．ここで，$< A >$,$< C >$ のノードの支持数は PDB の PList よりそれぞれ，2,1 となる．以下 $k = k + 1$ とし，再帰的に同様の処理を行う.

図 2 は結果として構築されたサフィックス木を表す．3 つの終端クラスタを点線の枠で示している．左と右のクラスタは新しいクラスタとして生成された．真ん中のクラスタでは，扱われるサフィックスが 1 つしかないので，新しく生成する必要がない．よって図 1c の二重丸のノードが取って代わる.

4. 最小汎化された正規表現の導出

4.1 残差汎化パターン集合

長さが k の部分文字列 $< \alpha_1\alpha_2\cdots\alpha_k >$ をある汎化パターン $< \Sigma_1\Sigma_2\cdots\Sigma_k >$ のインスタンス（$k-$ 部分文字列）とするとき，$< \Sigma_1\Sigma_2\cdots\Sigma_k >$ は以下の 2 要素の汎化パターンとして分

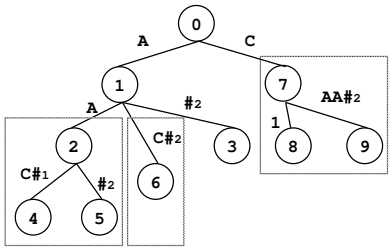

図 2　$S =< AAC\#_1CAA\#_2 >$ に対するサフィックス木

解可能である $(1 \leqq i \leqq k)$.

$$\{< \Sigma_1\cdots\Sigma_{i-1}(\Sigma_i - \{\alpha_i\})\Sigma_{i+1}\cdots\Sigma_k >, \\ < \Sigma_1\cdots\Sigma_{i-1}\alpha_i\ \Sigma_{i+1}\cdots\Sigma_k >\}$$

以後，これを以下のように表現する.

$$< \Sigma_1\Sigma_2\cdots\Sigma_k >=< \Sigma_1\cdots\Sigma_{i-1}(\Sigma_i - \{\alpha_i\})\Sigma_{i+1}\cdots \\ \Sigma_k > + < \Sigma_1\cdots\Sigma_{i-1}\alpha_i\Sigma_{i+1}\cdots\Sigma_k > \quad (5)$$

式 (5) の右辺第二項を左辺に移項すると，以下のようになる.

$$< \Sigma_1\Sigma_2\cdots\Sigma_{k-1}\Sigma_k > - < \Sigma_1\Sigma_2\cdots \\ \Sigma_{i-1}\alpha_i\Sigma_{i+1}\cdots\Sigma_{k-1}\Sigma_k >= \\ < \Sigma_1\Sigma_2\cdots(\Sigma_{i-1} - \{\alpha_i\})\Sigma_{i+1}\ \cdots\Sigma_{k-1}\Sigma_k > \quad (6)$$

式 (5) の分解の組み合わせは，k 通り存在する．式 (5) の右辺第一項を k 通り集めたものを $< \Sigma_1\Sigma_2\cdots\Sigma_k >$ に対する $< \alpha_1\alpha_2\cdots\alpha_k >$ の除去により得られる残差汎化パターン集合 $RPS(< \Sigma_1\Sigma_2\cdots\Sigma_k >, < \alpha_1\alpha_2\cdots\alpha_k >)$ と呼ぶ.

$$RPS(< \Sigma_1\Sigma_2\cdots\Sigma_k >, < \alpha_1\alpha_2\cdots\alpha_k >) = \\ < (\Sigma_1 - \{\alpha_1\})\Sigma_2\cdots\Sigma_{k-1}\Sigma_k > + \cdots + \\ < \Sigma_1\Sigma_2\cdots\Sigma_{i-1}(\Sigma_i - \{\alpha_i\})\Sigma_{i+1}\cdots\Sigma_{k-1}\ \Sigma_k > \\ + \cdots + < \Sigma_1\Sigma_2\cdots\Sigma_{k-1}(\Sigma_k - \{\alpha_k\}) > \quad (7)$$

ただし，右辺の中で冗長なパターンは除去する．$< \Sigma_1\Sigma_2\cdots\Sigma_{i-1}\alpha_i\Sigma_{i+1}\cdots\Sigma_{k-1}\Sigma_k >$ から残差汎化パターン集合 RPS を削除すると，$< \alpha_1\alpha_2\cdots\alpha_{k-1}\alpha_k >$ だけが残るので，残差汎化パターン集合は，$< \Sigma_1\Sigma_2\cdots\Sigma_{k-1}\Sigma_k > - < \alpha_1\alpha_2\cdots\alpha_{k-1}\alpha_k >$ のインスタンス集合を最小被覆する汎化パターンを表現している．なぜなら，$< \Sigma_1\Sigma_2\cdots\Sigma_{i-1}\alpha_i\Sigma_{i+1}\cdots\Sigma_{k-1}\Sigma_k >$ を式 (7) の第一項

$< (\Sigma_1 - \{\alpha_1\})\Sigma_2 \cdots \Sigma_{k-1}\Sigma_k >$ から削除すると，$< \alpha_1\Sigma_2 \cdots \Sigma_{i-1}\alpha_i\Sigma_{i+1} \cdots \Sigma_k >$ となり，これをさらに，式 (7) の第二項 $< \Sigma_1\Sigma_2(\Sigma_2 - \{\alpha_2\})\Sigma_3 \cdots \Sigma_k >$ から削除すると，$< \alpha_1\alpha_2\Sigma_3 \cdots \Sigma_{i-1}\alpha_i\Sigma_{i+1} \cdots \Sigma_k >$ となる．これを繰り返すと，最後に $< \alpha_1\alpha_2 \cdots \alpha_{k-1}\Sigma_k >$ から式 (7) の第 k 項 $< \Sigma_1\Sigma_2 \cdots \Sigma_{k-1}(\Sigma_k - \{\alpha_k\}) >$ を削除することになる．その結果として $< \alpha_1\alpha_2 \cdots \alpha_{k-1}\alpha_k >$ が得られる．

[例] 長さ 3 の文字列 $< ABC >$ は，汎化パターン $< [AD][BE][CF] >$ のインスタンスである．$< [AD][BE][CF] >$ に対する $< ABC >$ の除去により得られる残差汎化パターン集合 RPS は以下のとおり．

$$RPS(< [AD][BE][CF] >, < ABC >) =< D[BE][CF] > + < [AD]E[CF] > + < [AD][BE]F >$$

式 (5),(6),(7) をさらに一般化してみよう．ある汎化パターン $< \Gamma_1\Gamma_2 \cdots \Gamma_k >$ に対して，$\Delta_i = \Gamma_i \cap \Sigma_i \neq \phi$ が成立するとき，他の汎化パターン $< \Sigma_1\Sigma_2 \cdots \Sigma_k >$ を以下のように分解可能である $(1 \leqq i \leqq k)$.

$$< \Sigma_1\Sigma_2 \cdots \Sigma_{k-1}\Sigma_k >= < \Sigma_1\Sigma_2 \cdots \Sigma_{i-1}(\Sigma_i - \Gamma_i)\Sigma_{i+1} \cdots \Sigma_{k-1}\Sigma_k > + < \Sigma_1\Sigma_2 \cdots \Sigma_{i-1}\Delta_i\Sigma_{i+1} \cdots \Sigma_{k-1}\Sigma_k > \quad (8)$$

式 (8) の右辺第二項を左辺に移項すると，以下のようになる．

$$< \Sigma_1\Sigma_2 \cdots \Sigma_{k-1}\Sigma_k > - < \Sigma_1\Sigma_2 \cdots \Sigma_{i-1}\Delta_i\Sigma_{i+1} \cdots \Sigma_{k-1}\Sigma_k >=< \Sigma_1\Sigma_2 \cdots \Sigma_{i-1}(\Sigma_i - \Gamma_i)\Sigma_{i+1} \cdots \Sigma_{k-1}\Sigma_k > \quad (9)$$

また，$< \Sigma_1\Sigma_2 \cdots \Sigma_k >$ に対する $< \Gamma_1\Gamma_2 \cdots \Gamma_k >$ の除去により得られる残差汎化パターン集合 $RPS(< \Sigma_1\Sigma_2 \cdots \Sigma_k >, < \Gamma_1\Gamma_2 \cdots \Gamma_k >)$ は以下のとおりである．

$$RPS(< \Sigma_1\Sigma_2 \cdots \Sigma_k >, < \Gamma_1\Gamma_2 \cdots \Gamma_k >) = < (\Sigma_1 - \Gamma_1)\Sigma_2 \cdots \Sigma_k > + \cdots + < \Sigma_1 \cdots \Sigma_{i-1}(\Sigma_i - \Gamma_i)\Sigma_{i+1} \cdots \Sigma_k > + \cdots + < \Sigma_1\Sigma_2 \cdots \Sigma_{k-1}(\Sigma_k - \Gamma_k) > \quad (10)$$

ただし，右辺の中で冗長なパターンは除去する．

4.2 最小汎化パターンの探索

ある $k-$ 部分文字列の集合を被覆する汎化パターンの中で最小の汎化パターンを最小汎化パターンと呼ぶ．正（あるいは負）の $k-$ 部分文字列の集合から得られる最小汎化パターンをそれぞれ正（あるいは負）の最小汎化パターンと呼ぶ．

(1) k-部分文字列の集合 PS から $< \Sigma_1, \Sigma_2, \cdots, \Sigma_k >$ それぞれを見つけ出し，最汎パターン $< \Sigma_1\Sigma_2 \cdots \Sigma_k >$ を構成する．

(2) 式 (7) を用いて，$< \Sigma_1\Sigma_2 \cdots \Sigma_k >$ に対する PS の除去により得られる残差汎化パターン集合 RPS を計算する． ただし，冗長なパターンが出現した場合は，ただちに取り除く．

(3) 上記の残差汎化パターン RPS から負の最小汎化パターンの集合を選び出す．それぞれの最小汎化パターンは，PS のどんな部分集合（ただし，空集合は除く）も被覆しないパターンである．

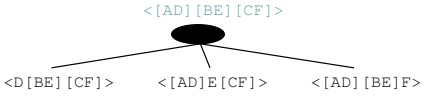

図 3 $< ABC >$ の除去結果

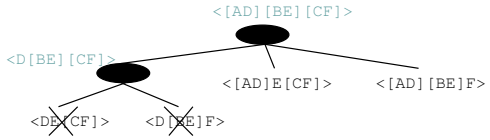

図 4 $< ABC >$ と $< DBC >$ の除去結果

(4) 式 (10) を用いて，$< \Sigma_1\Sigma_2 \cdots \Sigma_k >$ に対する負の最小汎化パターン集合の除去により得られる残差汎化パターン集合 RPS' を計算する．ただし，冗長なパターンが出現した場合は，ただちに取り除く．

(5) 上記の残差汎化パターン集合 RPS' から正の最小汎化パターンの集合を選び出す．それぞれの最小汎化パターンは，$NG = \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_k -$ PS のどんな部分集合（ただし，空集合は除く）も被覆しないが，PS のある部分集合を被覆する汎化パターンである．

[例] 最汎パターンを $< [AD][BE][CF] >$，正の 3-部分文字列集合 PS を $\{< ABC >, < DBC >\}$ とするとき，負の最小汎化パターンを見つけてみよう．

(1) $< [AD][BE][CF] >$ に対する $< ABC > \in NG$ の除去により得られる残差汎化パターン集合は，$RPS(< [AD][BE][CF] >, < ABC >) =< D[BE][CF] > + < [AD]E[CF] > + < [AD][BE]F >$ である．

(2) $< D[BE][CF] >$ に対する $< DBC > \in NG$ の除去により得られる残差汎化パターン集合は，$RPS(< D[BE][CF] >, < DBC >) =< DE[CF] > + < D[BE]F >$ である．

しかしながら，これらの 2 つの汎化パターンは，どちらも (1) で得られた汎化パターンに含まれるので，除去することができる．以上により，残された汎化パターンは，$PS = \{< ABC >, < DBC >\}$ に対する負の最小汎化パターン集合 $\{< [AD]E[CF] >, < [AD][BE]F >\}$ である．図 3 と図 4 は，この例で探索される最小汎化パターンの探索過程を表現した図である．

[例] 最汎パターンを $< [AD][BE][CF] >, PS = \{< ABF >, < AEC >, < AEF >, < DBF >, < DEC >, < DEF >\}$ とするとき，負の最小汎化パターンを見つけた後に，正の最小汎化パターンを見つけてみよう．

(1) 最汎パターン $< [AD][BE][CF] >$ に対する $< ABF > \in PS$ の除去により得られる残差汎化パターン集合は，$RPS(< [AD][BE][CF] >, < ABF >) =< D[BE][CF] > + < [AD]E[CF] > + < [AD][BE]C >$ である．

(2) $< [AD]E[CF] >$ と $< [AD][BE]C >$ の各々に対する $< AEC > \in PS$ の除去により得られる残差汎化パターン集合は，$RPS(< [AD]E[CF] >, < AEC >) =<$

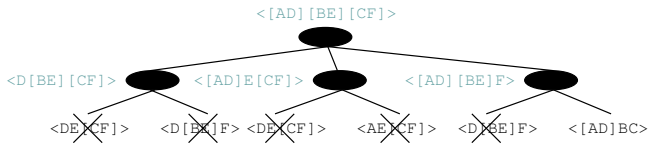

図 5 $\{< ABC >, < DBC >, < AEF >, < DBF >, < DEC >, < DEF >\}$ の除去結果

$DE[CF] > + < [AD]EF >$, $RPS(< [AD][BE]C >, < AEC >) =< D[BE]C > + < [AD]BC >$ である．しかしながら，$< DE[CF] >$ と $< D[BE]C >$ は，(1) で得られた $< D[BE][CF] >$ に含まれるので，削除する．

(3) $< [AD]EF >$ に対する $< AEF > \in PS$ の除去により得られる残差汎化パターン集合は空集合である．

(4) $< D[BE][CF] >$ に対する $< DEF > \in PS$ の除去により得られる残差汎化パターン集合は，$RPS(< D[BE][CF] >, < DBF >) =< DE[CF] > + < D[BE]C >$ である．

(5) $< DE[CF] >$ と $< D[BE]C >$ の各々に対する $< DEC > \in PS$ の除去により得られる残差汎化パターン集合は空集合である．

(6) 残った汎化パターンには $< DEF > \in PS$ が含まれないので，残差汎化パターン集合の計算は不要である．以上により，$< [AD]BC >$ が負の最小汎化パターンとなる．これを最汎パターン $< [AD][BE][CF] >$ から除去すると，正の最小汎化パターンが得られる．

(7) 最汎パターン $< [AD][BE][CF] >$ に対する $< [AD]BC >$ の除去により得られる残差汎化パターン集合は，$RPS(< [AD][BE][CF] >, < [AD]BC >) =< [AD]E[CF] > + < [AD][BE]F >$ である．

従って，$< [AD]E[CF] >, < [AD][BE]F >$ が $PS = \{< ABF >, < AEC >, < AEF >, < DBF >, < DEC >, < DEF >\}$ に対する正の最小汎化パターンとなる．図 5 は，この例で探索される最小汎化パターンの探索過程を表現した図である．

5. 実　　験

ここでは，2 種類のデータセット，Zinc Finger, Cytochrome C に対して，サフィックス木を用いて曖昧パターン検索を行い，出力として得られた $k-$ 部分文字列集合に対して最小汎化を行う．これにより，提案手法の有効性を確認する．このために利用した計算機環境は，Intel Pentium(R) 4-3.2GHz，メモリー：2GB，SWAP メモリー：2GB，HDD：320GB，OS：CentOS v4.4 である．2 種類のデータセットの詳細は以下の表 2 の通りである．

表 2　データセット詳細

データセット	データ件数	総長 (byte)	最大長 (byte)	最小長 (byte)
Zinc Finger	467	245595	4036	34
Cytochrome C	29	4235	631	116

表 3　Zinc Finger を用いた曖昧検索結果

検索条件		曖昧検索結果		
許容誤差半径	最小支持数	出力文字列数 (個)	支持数	検索時間 (sec)
1	58	2280	454	11617

表 4　Zinc Finger を用いた汎化処理結果

汎化処理結果	
パターン数	処理時間 (sec)
37	2.155

5.1 Zinc Finger データセット

Zinc Finger を用いて曖昧検索を行い，得られた部分文字列集合に対して最小汎化を行う実験について示す．Zinc Finger モチーフは

$$< C - x(2,4) - C - x(3) - [LIMFYWC] - x(8) - H - x(3,5) - H >$$

の形式で知られている．曖昧検索における検索条件は以下の表に示す．また検索キーはモチーフの一部分である $< C - x(2,4) - C - x(3) - L - x(8) - H - x(3,5) - H >$ を与えた．この実験では，可変長ワイルドカード領域部分を全て網羅するため，検索キーのワイルドカード領域数を変えていき，計 9 回の実験を行った．表 3，表 4 は 9 回の実験をまとめた結果である．

曖昧検索によって得られた部分文字列集合数は 2280 個であり，この曖昧検索結果から計算される最汎パターンは

$< [CDEFHIKLMNPQRSTVY] - x(2,4) - [ACGNVWY] - x(3) - [ACDFGHILMNPQRSTVWY] - x(8) - [ACHKRTY] - x(3,5) - [ACDEGHIKLMNPQRSTV] >$

であった．この最汎パターンは，2294082 種類の部分文字列を 1 つにまとめたものである．最小汎化パターンを抽出するために，先ずこの最汎パターンから曖昧検索によって得られた 2280 の全ての部分文字列を除去することで 96 の負の汎化パターン集合を抽出した．次に負の汎化パターン集合を最汎パターンから削除した．その結果，37 個の正の部分文字列集合に対する汎化パターンが抽出できた．この汎化処理を行うことで，曖昧検索によって得られた大量の部分文字列集合を少数のパターンでの表現に成功したといえる．曖昧検索によって得られた部分文字列集合を利用する際，少数の汎化パターンを参照するだけで全体像を容易に把握できる．　実際に抽出された汎化パターンの中には，$< C - x(3) - C - x(3) - [FLY] - x(8) - H - x(3) - H >, < C - x(4) - C - x(3) - [LIVMFYW] - x(8) - H - x(5) - H >$ などのパターンが含まれていた．これらはモチーフの一部分を表現したパターンであることがわかる．単に，部分文字列集合を見るより，汎化処理によって得られた，よりモチーフに近い汎化パターンを抽出することでデータセットに含まれているモチーフの発見に役立つ可能性が高まる．

5.2 Cytochrome C データセット

Cytochrome C を用いて曖昧検索を行い，得られた部分文字列集合に対して最小汎化を行う実験について示す．Cytochrome C モチーフは

$< C - x(2) - [STAQ] - x - [STAMV]$
$-C - [STA] - T - C\ -[HR] >$

の形式で知られている．検索キーとして $< C - x(2) - A - x - A - C - A - T - C - R >$ を与えた．曖昧検索結果を表 5，汎化処理結果を表 6 に示す．

表 5　Cytochrome C を用いた曖昧検索結果

検索条件		曖昧検索結果		
許容誤差半径	最小支持数	出力文字列数 (個)	支持数	検索時間 (sec)
4	8	15	29	0.117

表 6　Cytochrome C を用いた汎化処理結果

汎化処理結果	
パターン数	処理時間 (sec)
7	0.132

曖昧検索によって得られた部分文字列集合数は 15 個であり，この曖昧検索結果から計算される最汎パターンは

$< C - x(2) - [STAQ] - x - [SAMV] - C - [STA] - T - C - [HR] >$

であった．この最汎パターンは，96 種類の部分文字列を 1 つにまとめたものである．最小汎化パターンを抽出するために，先ずこの最汎パターンから曖昧検索によって得られた 15 の全ての部分文字列を除去することで 25 の負の汎化パターン集合を抽出した．次に負の汎化パターン集合を最汎パターンから削除した．その結果，7 個の正の部分文字列集合に対する汎化パターンが抽出できた．今回の実験でも，汎化処理によって曖昧検索で得られた部分文字列集合を少数のパターンで表現することに成功した．抽出された汎化パターンは，$< C - x(2) - S - x - A - C - [AST] - T - C - H >$,$< C - x(2) - S - x - [AV] - C - T - T - C - H >$,$< C - x(2) - S - x - [AS] - C - A - T - C - H >$,$< C - x(2) - [AS] - x - A - C - [AS] - T - C - H >$,$< C - x(2) - [AST] - x - A - C - S - T - C - H >$,$< C - x(2) - Q - x - M - C - A - T - C - H >$,$< C - x(2) - S - x - V - C - A - T - C - R >$ の 7 通りであり，全てがモチーフの形式に含まれている．モチーフのパターンと同じパターンが抽出されていないことから，使用したデータセットが完全にモチーフのパターンを含んでいないことがわかる．アミノ酸配列データベースでは，多くの場合，モチーフの形式は知られているが，モチーフの形式で表現される全てのパターンは含んでいないといわれている．そこで，汎化処理を行うことで，データセットに含まれるモチーフの正確なパターンを知ることができるといえる．

6. ま と め

本研究では，ディスク上サフィックス木により曖昧検索を行い，誤差半径 r 内にある $k-$ 部分文字列集合を全て求め，汎化処理によって曖昧な表現を含む配列パターンの正規表現化を行なった．提案手法の有効性を確めるため，2 種類のデータセットを用いて実験を行った．そして，曖昧検索によって得られた部分文字列集合を用いて最汎パターンを算出し，そこから正の部分文字列集合を除去し，負の汎化パターン集合を求め，最汎パターンから負の汎化パターン集合を除去することで正の汎化パターン集合を求めた．Zinc Finger データセットでの実験では，曖昧検索によって得られた 2280 個の部分文字列を 37 個のパターンで表現することに成功した．また，単に曖昧検索によって得られたパターンよりも，よりモチーフに近い表現のパターン (モチーフの一部分) を発見することができた．Cytochrome C データセットを使用した実験では，得られた 15 個の部分文字列を 7 個の汎化パターンで表現することに成功した．また，使用したデータセットの中に含まれるモチーフの正確なパターンを抽出することができた．

今後の課題としては，アミノ酸の特徴を考慮したパターン抽出，また，PROSITE にあるモチーフの形式，例えば $< [KR] - x(1,3) - [RKSAQ] - N - \{VL\} - x - [SAQ](2) - \{L\} - [RKTAENQ] - x - R - \{S\} - [RK] >$ のパターンに含まれる可変長ワイルドカード領域や { } を含む正規表現化の研究が重要であると考えられる．

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)(一般)，課題番号：17500097)，および文部科学省・科学研究費補助金 (課題番号：16700114) の支援により行われた．

文　　献

[1] Inge Jonassen, F. Collins, and Desmond G. Higgins. Finding flexible patterns in unaligned protein sequences. *Protein Science*, pp. 1587–1595, 1995.

[2] Laurent Marsan and Marie-France Sagot. Extracting structured motifs using a suffix tree - algorithms and application to promoter consensus identification. In *RECOMB*, pp. 210–219, 2000.

[3] Tomoyuki Kato, Hajime Kitakami, Makoto Takaki, Keiichi Tamura, Yasuma Mori, and Susumu Kuroki. Extraction for frequent sequential patterns with minimum varaible-wildcard regions. In *PDPTA*, pp. 825–831, 2006.

[4] Hiroki Arimura and Takeaki Uno. A polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence. In *ISAAC*, pp. 724–737, 2005.

[5] 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配列パターンの抽出. 電子情報通信学会論文誌 D，データ工学特集号, Vol. J90-D, No. 2, 2007.

[6] Zvi Galil and Kunsoo Park. An improved algorithm for approximate string matching. *SIAM J. Comput.*, Vol. 19, No. 6, pp. 989–999, 1990.

[7] Sanghyun Park, Wesley W. Chu, Jeehee Yoon, and Chihcheng Hsu. Efficient searches for similar subsequences of different lengths in sequence databases. In *ICDE*, pp. 23–32, 2000.

[8] Marie-France Sagot. Spelling approximate repeated or common motifs using a suffix tree. In *LATIN*, pp. 374–390, 1998.

[9] Pavel A. Pevzner and Sing-Hoi Sze. Combinatorial approaches to finding subtle signals in dna sequences. In *ISMB*, pp. 269–278, 2000.

[10] Eleazar Eskin and Pavel A. Pevzner. Finding composite regulatory patterns in dna sequences. In *ISMB*, pp. 354–363, 2002.

[11] Ching-Fung Cheung, Jeffrey Xu Yu, and Hongjun Lu. Constructing suffix tree for gigabyte sequences with megabyte memory. *IEEE Trans. Knowl. Data Eng.*, Vol. 17, No. 1, pp. 90–105, 2005.