社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－BIO－12 (11)
2008／3／4

# 塩基およびアミノ酸配列における共変異集合を列挙する高速アルゴリズム

伊藤公人[1] 谷口剛[2] 村上悌治[1] 有村博紀[2] 原口誠[2]

北海道大学，人獣共通感染症リサーチセンター[1] 大学院情報科学研究科[2]

**概 要**

塩基やアミノ酸の多重配列アライメントから，同時に変異する傾向にある塩基または残基位置の集合を高速に列挙する手法を提案する．本研究では，共変異の検出において，1) 条件付きエントロピー $H(i|j)$ を指標として用いることが有効であること，2) 条件付きエントロピー $H(i|j)$ の指標を条件付き同時エントロピー $H(i_1 \cdots i_m|j)$ へと自然に拡張することにより，三つ以上の位置における共変異を検出可能であること，3) 条件付き同時エントロピーの単調性を利用した枝刈り規則と，位置集合と配列集合に関する閉包性を用いることにより，共変異位置集合の列挙アルゴリズムを著しく高速化することが可能であることを示す．

# Fast Algorithms for Finding Correlated Mutations from Nucleic/Amino Acid Sequences

Kimihito ITO[1], Tsuyoshi TANIGUCHI[2], Teiji MURAKAMI[1],
Hiroki ARIMURA[2], Makoto HARAGUCHI[2]

Hokkaido University Research Center for Zoonosis Control[1]
Graduate School of Information Science and Technology[2],Hokkaido University

**Abstract**

We present new algorithms to find the sets of positions involved in correlated mutations from amino/nucleic acid sequences. In this paper, we show that 1) use of the conditional entropy $H(i|j)$ is overall effective in detecting the sets of positions involved in correlated mutations, 2) a natural extension of $H(i|j)$ to the conditional entropy $H(i_1 \cdots i_m|j)$ allows us to detect the sets of more than two positions involved in correlated mutations, 3) pruning rules derived from the monotonicity and closure of positions under logical equivalence boost the running performance in terms of computational speed remarkably.

## 1 はじめに

本研究では，大量の塩基やアミノ酸の配列から，同時に変異する傾向にある塩基または残基位置の集合を検出する問題を扱う．

一般に，遺伝子やタンパク質の進化過程において，塩基やアミノ酸残基が二つ以上の異なる位置で同時に変異する場合があることが知られている [7, 1, 11, 9, 5, 17]．塩基やアミノ酸残基が二つ以上の位置で同時に変異する進化過程を，**共変異** (*correlated mutation*) と呼ぶ．

塩基やアミノ酸残基における共変異は，タンパク質やRNAの構造・機能・相互作用の保持および進化に本質的な事象であると考えられている [9, 11, 17]．そこで，逆に大量の配列データから共変異を検出し，タンパク質やRNAの構造・機能・相互作用を予測する試みが行われている．例え

ば，タンパク質内の残基のコンタクトや構造予測 [9, 2, 13, 20, 8, 10, 19]，RNA の二次構造予測 [5, 22]，タンパクタンパク間相互作用の予測 [1, 17]，ウイルスタンパク質の進化解析 [11, 4] 等，広く応用が期待されている．

共変異を検出する既存手法においては，検出能力・検出対象・計算速度の面から下記の問題が浮上する．まず，従来手法においては，配列中の位置 $i$ と位置 $j$ における共変異を検出する尺度として，アミノ酸置換行列上での相関係数 $\rho_{i_j}$ を用いる手法 [9, 17, 6, 19], 同時置換における $\chi^2$ 検定を用いる手法 [13, 16] や，相互情報量 $I(i;j)$ を用いる手法 [11, 5, 15, 12, 2, 8, 14, 14, 18] 等が提案されている．特に，概念が単純であり実装が容易であることから，相互情報量は共変異検出の尺度として広く用いられている．しかし，**共変異の前と後の配列数の比が大きい場合，相互情報量を用いる従来手法では共変異を適切に検出できない**．

また，共変異においては，三つ以上の塩基または残基位置がクラスターを形成するように同時に変異する場合が知られている．相関係数 $\rho_{i_j}$ や相互情報量 $I(i;j)$ を用いた従来手法は，位置の組 $(i,j)$ に対し共変異の検出を行う．三つ以上の位置において共変異する場合は，重み付き完全グラフにおけるクラスター解析を行う必要があり [18]，位置のクラスターを共変異と見なす尺度の妥当性を評価することが困難となる．一般に，与えられた塩基またはアミノ酸配列の長さが $m$ である場合，共変異する位置の組の検出は，$m(m-1)/2$ 通りの全ての組み合わせに関してスコア関数の計算を行えばよい．しかし，二つ以上の位置の共変異を全て考慮する場合，素朴な手法では $2^m$ 回の計算が必要となり**膨大な計算コストが必要となる**．

本研究では，塩基やアミノ酸の多重配列アライメントから，同時に変異する傾向にある塩基または残基位置の集合 (共変異位置集合) を高速に列挙する手法を提案する．具体的には，共変異位置集合の検出において，1) **条件付きエントロピーを指標として用いることが有効である**こと，2) 条件付きエントロピーの指標を条件付き同時エントロピーへと自然に拡張することにより，**三つ以上の位置における共変異をも検出可能である**こと，3) 条件付き同時エントロピーの単調性を利用した枝刈り規則と，位置集合と配列集合に関する閉包性を用いることにより，**共変異位置集合の列挙アルゴリズムを著しく高速化することが可能である**ことを示す．

## 2 準備

$\Sigma$ を有限個の文字の集合とする．例えば，DNA を対象とする場合は $\Sigma = \{\mathrm{A, C, G, T}\}$ であり，RNA を対象とする場合は $\Sigma = \{\mathrm{A, C, G, U}\}$, アミノ酸配列を対象とする場合は，$\Sigma = \{\mathrm{A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V}\}$ である．

$\Sigma$ 上の長さ $m$ の文字列全体の集合を $\Sigma^m$ で表し，文字列全体の集合 $\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cdots$ を $\Sigma^*$ で表す．任意の文字列 $s \in \Sigma^*$ に対し，$|s|$ で $s$ の長さを表し，$s[i]$ で，$s$ の $i$ 番目の文字を表す．正整数 $n$ に対し，$[n]$ で $\{1, \cdots, n\}$ を表す．

正整数を $m$ とおく．入力文字列集合は，集合 $S = \{s_1, \ldots, s_n\}$ である．ここに，各 $1 \leq j \leq n$ に対し，$s_j = s_j[1] \cdots s_j[m]$ は長さ $m$ の $\Sigma$ 上の文字列である．以後，特に注意しない場合は，$m$ と $n, S$ を固定する．

入力文字列集合 $S \subseteq \Sigma^m$ の任意の位置を $i \in [m]$ とおく．このとき，$S$ における $i$ 番目の位置における文字 $a \in \Sigma$ の (経験的) 確率は，

$$P_i(a) = \frac{1}{n} |\{1 \leq j \leq n \,:\, s_j[i] = a\}|.$$

となる．また，$S$ の $k$ 個の位置を $i_1 \cdots i_k \subseteq [m]$ とおく．このとき，$k$ 個の位置 $i_1 \cdots i_k$ における文字がそれぞれ $a_1 \cdots a_k$ である同時確率は，

$$P_{i_1 \cdots i_k}(a_1 \cdots a_k) = \frac{1}{n} |\{1 \leq j \leq n \,:\, s_j[i_1] \cdots s_j[i_k] = a_1 \cdots a_k\}|.$$

となる．次に，$S$ における $j$ 番目の文字 $c$ が与えられたとき，位置 $i_1 \cdots i_k$ における文字がそれぞれ $a_1 \cdots a_k$ となる条件付き同時確率を，

$$P_{i_1 \cdots i_k | j}(a_1 \cdots a_k | c) = \frac{P_{i_1 \cdots i_k j}(a_1 \cdots a_k c)}{P_j(c)}$$

とする．次節で，入力文字列集合 $S$ の位置 $i_1 \cdots i_k \subseteq [m]$ の共変異を検出するためのスコア関数を定義する．

## 3 スコア関数

入力文字列集合 $S \in \Sigma^m$ の $k$ 個の位置を $i_1 \cdots i_k \subseteq [m]$ とおく．このとき，位置 $i_1 \cdots i_k$ における**同時エントロピー** (*joint entropy*) は，

$$H(i_1 \cdots i_k) = - \sum_{a_1 \cdots a_k \in \Sigma^k} P_{i_1 \cdots i_k}(a_1 \cdots a_k) \log P_{i_1 \cdots i_k}(a_1 \cdots a_k)$$

である．特に，$k = 1$ である場合，$H(i)$ は位置 $i$ における Shannon エントロピーである．また，位置 $j$ が与えられた時の，$i_1 \cdots i_k$ の**条件付き同時エントロピー** (*conditional joint entropy*) は，$j$ が $c \in \Sigma$ である時の $i_1 \cdots i_k$ における同時エントロピーの $c \in \Sigma$ に関する平均値である．

$$\begin{aligned} H(i_1, \ldots, i_k | i_0) &= \sum_{c \in \Sigma} P_{i_0}(c) \left[ - \sum_{a_1 \cdots a_k \in \Sigma^k} P_{i_1, \ldots, i_k | i_0}(a_1, \ldots, a_k | c) \log P_{i_1, \ldots, i_k | i_0}(a_1, \ldots, a_k | c) \right] \\ &= - \sum_{a_1 \cdots a_k c \in \Sigma^{k+1}} P_{i_1, \ldots, i_k, i_0}(a_1, \ldots, a_k, c) \left[ \log P_{i_1, \ldots, i_k, i_0}(a_1, \ldots, a_k, c) - \log P_{i_0}(c) \right] \\ &= H(i_1, \ldots, i_k, i_0) - H(i_0) \geq 0 \end{aligned}$$

入力文字列集合 $S$ の位置の組を $(i, j)$ とおく．このとき，$S$ の $i, j$ における**相互情報量** (*mutual information*) は，

$$\begin{aligned} I(i; j) &= H(i) - H(i|j) = H(j) - H(j|i) \\ &= H(i) + H(j) - H(ij) \geq 0 \end{aligned}$$

であり，$I(i; j) = I(j; i)$ を満たす．

### 3.1 主条件付き同時エントロピー

条件付き同時エントロピーにおいては，一般に $H(i|j) \neq H(j|i)$ である．本研究では，ある閾値 $\epsilon$ を考え，$H(i|j) \leq \epsilon$ かつ $H(j|i) \leq \epsilon$ となる組 $(i, j)$ を見つける問題を扱うため，**主**条件付き同時エントロピーを下記のように定義する．

**定義 1** 入力文字列集合 $S$ の任意の $k$ 個の位置を $i_1 \cdots i_k \subseteq [m]$ とおく．このとき，位置 $i_1 \cdots i_k$ における**主条件付き同時エントロピー** (*major conditional joint entropy*) を，全ての可能な条件付きエントロピーの最大値と定義する．つまり，

$$\begin{aligned} MCJE(i_1, \ldots, i_k) &= \max_{1 \leq j \leq k} \left[ H(i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_k | i_j) \right] \\ &= \max_{1 \leq j \leq k} \left[ H(i_1, \ldots, i_k) - H(i_j) \right] \\ &= H(i_1, \ldots, i_k) - \min_{1 \leq j \leq k} H(i_j) \end{aligned}$$

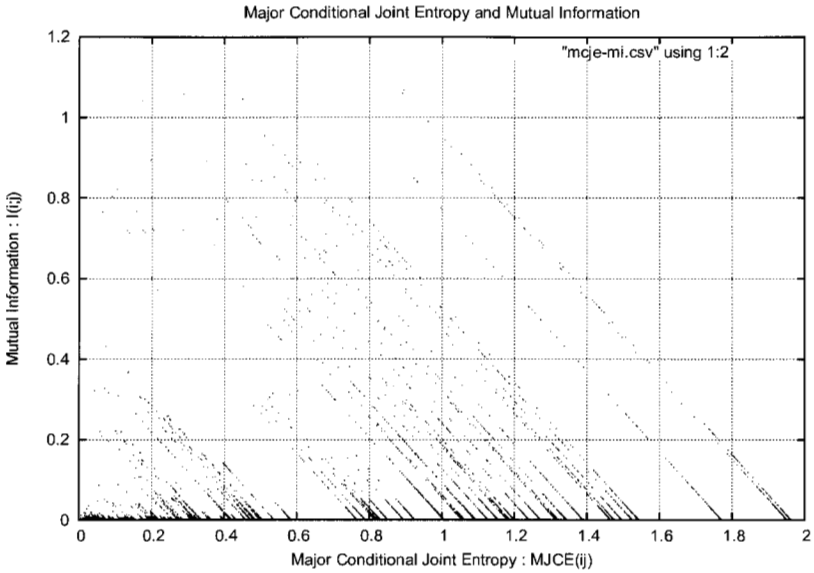


図 1: 主条件付き同時エントロピーと相互情報量

### 3.2 主条件付き同時エントロピーの有用性

入力文字列集合から位置の組 $(i,j)$ が共変異しているかどうかを判定する問題を考える．位置の組 $(i,j)$ の文字が同時に変化する傾向にあるとき，相互情報量 $I(i;j)$ の値は大きくなり，主条件付き同時エントロピー $MCJE(ij)$ の値は小さくなると考えられる．しかし，相互情報量 $I(i;j)$ の値が比較的小さな値をとる場合においても，位置の組 $(i,j)$ が同時に変化する場合も考えられる．つまり，$(i,j)$ の文字が常に同時に変化する場合であっても，$I(i;j)$ の値が大きい値をとるとは限らない．$I(i;j) \leq H(i)$ かつ $I(i;j) \leq H(j)$ が成り立つため，$H(i)$ または $H(j)$ が小さいときには $I(i;j)$ は小さな値をとるためである．このため，大きな相互情報量を持つ組 $(i,j)$ の検索による共変異の検出手法には，その感度に問題がある．

図 1 に主条件付き同時エントロピーと相互情報量の関係を示す．図 1 のグラフは，後述する H3N2 亜型インフルエンザウイルスの HA タンパク質のアミノ酸配列から作成した．相互情報量 $I(i,j)$ では，小さい値 (悪いスコア) を示しているが，主条件付きエントロピー $MCJE(ij)$ も小さい値 (良いスコア) を持つ組 $(i,j)$ が数多く存在することが判る．このような組 $(i,j)$ は，相互情報量を用いた手法では検出されない．

逆に，相互情報量が大きな値 (良いスコア) を持つが，主条件付きエントロピーも大きな値 (悪いスコア) を持つような組 $(i,j)$ も存在する．主条件付きエントロピーが大きな値をとることは，$i$ または $j$ のどちらか一方の位置のみが変異することを意味し，相互情報量を用いた検出手法は精度の点でも問題がある．上記の議論から，共変異の検出には相互情報量 $I(i;j)$ よりも，主条件付きエントロピー $MCJE(i,j)$ をスコア関数として用いる方が，感度および精度の高い検出を行えると考えられる．

## 4 主条件付き同時エントロピーを用いた共変異の検出

主条件付き同時エントロピーを用いた共変異検出の問題は下記のとおりである．

**定義 2** 入力文字列集合を $S \subseteq \Sigma^m$，閾値を $\epsilon$ とおく．スコア関数 $MCJE$ を用いた共変異位置の検出問題とは，$MCJE(i_1 \cdots i_k) \leq \epsilon$ を満たす全ての $i_1 \cdots i_k \subseteq [m]$ を列挙することである．

**Algorithm** FINDCRM($\epsilon$)
1: 入力文字列集合を $S \subseteq \Sigma^m$ とする;
2: $C := [m]$;
3: **while** $C \neq \phi$ **do**
4: $C$ 中の位置を $i$ とおく;
5: $C := C - \{i\}$;
6: **call** EXPAND($\{i\}, C, \epsilon$);
7: **end while**
**Procedure** EXPAND($I, C, \epsilon$)
1: $C_{next} := C$;
2: **for all** $i \in C$ **do**
3: $C_{next} := C_{next} - \{i\}$;
4: **if** $MCJE(I \cup \{i\}) \leq \epsilon$ **then**
5: **print** $I \cup \{i\}$;
6: **end if**
7: **call** EXPAND($I \cup \{i\}, C_{next}, \epsilon$);
8: **end for**

図 2: 素朴なアルゴリズム

**Algorithm** FINDCRMP($\epsilon$)
1: 入力文字列集合を $S \subseteq \Sigma^m$ とする;
2: $C := [m]$;
3: **while** $C \neq \phi$ **do**
4: $C$ 中の位置を $i$ とおく;
5: $C := C - \{i\}$;
6: **call** EXPANDP($\{i\}, C, \epsilon$);
7: **end while**
**Procedure** EXPANDP($I, C, \epsilon$)
1: $C_{next} := C$;
2: **for all** $i \in C$ **do**
3: $C_{next} := C_{next} - \{i\}$;
4: **if** $MCJE(I \cup \{i\}) \leq \epsilon$ **then**
5: **print** $I \cup \{i\}$;
6: **call** EXPANDP($I \cup \{i\}, C_{next}, \epsilon$);
7: **end if**
8: **end for**

図 3: 枝刈りを考慮したアルゴリズム

深さ優先探索により集合を列挙する手法を用いた素朴なアルゴリズムを考える．図 2 に入力文字列集合 $S \subseteq \Sigma^m$ から共変異位置の $MCJE(i_1 \cdots i_k) \leq \epsilon$ を満たす全ての $i_1 \cdots i_k \subseteq [m]$ を列挙するアルゴリズムを示す．$n$ を $|S|$ とすると，$MCJE$ の計算は $O(n)$ であり，$m$ 個の位置の部分集合 $2^m - n - 1$ 個に関して $MCJE$ を計算するため，FindCRM の計算量は $O(n2^m)$ である．

### 4.1 枝刈りによる高速化

深さ優先探索により集合を列挙する手法においては，集合の包含関係に対し，単調なスコアを定義できれば，効率よいアルゴリズムを設計可能であることが知られている [23, 21].

**補題 1 (単調性)** 位置の集合 $I, J \subseteq [m]$ をそれぞれ二つ以上の位置から成る集合とする．このとき，$I \subseteq J$ ならば，$MCJE(I) \leq MCJE(J)$ である．

**系 1 (枝刈り規則)** 位置の集合 $I$ を二つ以上の位置から成る集合とする．このとき，$MCJE(I) > \epsilon$ であるならば，任意の位置 $i$ に対して，$MCJE(I \cup \{i\}) > \epsilon$ である．

図 3 に，FindCRM に枝刈りを採用したアルゴリズム FindCRMP を示す．FindCRMP は解の個数 $M$ に比例した時間で動作する出力線形時間アルゴリズムである．

**定理 1** 入力文字列集合 $S \subseteq \Sigma^m$ に対し，FindCRMP は，$MCJE(i_1 \cdots i_k) \leq \epsilon$ を満たす全ての位置集合 $i_1 \cdots i_k \subseteq [m]$ を $O(nmM) = O(n2^m)$ 時間で列挙する．ここに，$M = O(2^m)$ は解となる位置集合 $I$ の個数である．

**証明:** 任意の位置集合 $I$ と任意の位置 $i$ に対し，$MCJE(I \cup \{i\}) \leq \epsilon$ ならば $MCJE(I) \leq \epsilon$ である．これより，図 3 の EXPANDP は，位置集合の末尾に新しい位置を追加しながら，全部で $M$ 個ある解となる位置集合を, 1 個あたり $O(mn)$ 時間で重複なく列挙することが示せる． ■

解となる位置集合の個数 $M = O(2^m)$ を減らすヒューリスティクスとして，新たな閾値 $\tau \geq \epsilon$ を導入し，検索対象の位置を $\{ i ; H(i) > \tau\} \subseteq [m]$ に限定することが有効であると考えられる．

**Algorithm** FINDCRMPC($\epsilon$)
1: 入力文字列集合を $S \subseteq \Sigma^m$ とする;
2: $C :=$ SQUEEZE(S);
3: **while** $C \neq \phi$ **do**
4: $C$ 中の位置を $i$ とおく;
5: $C := C - \{i\}$;
6: **call** EXPANDPC($\{i\}, C, \epsilon$);
7: **end while**

**Procedure** EXPANDPC($I, C, \epsilon$)
1: $C_{next} := C$;
2: **for all** $i \in C$ **do**
3: $C_{next} := C_{next} - \{i\}$;
4: **if** $MCJE(I \cup \{i\}) \leq \epsilon$ **then**
5: $I \cup \{i\}$ を $j_1 \cdots j_k$ とする;
6: **print** $[j_1]_{\equiv_S} \cdots [j_k]_{\equiv_S}$;
7: **call** EXPANDPC($I \cup \{i\}, C_{next}, \epsilon$);
8: **end if**
9: **end for**

**Procedure** SQUEEZE($S$)
1: $C := [m]$;
2: **while** $C \neq \phi$ **do**
3: $C$ 中の位置を $i$ とおく;
4: $CL =$ CLOSURE($i, C$);
5: **if** $|CL| > 1$ **then**
6: **print** $CL$;
7: **end if**
8: $[i]_{\equiv_S}$ をハッシュに記憶する;
9: $C := C - CL$;
10: **end while**
11: **return** $C$

**Procedure** CLOSURE($i, C$)
1: $CL$:={i};
2: **for all** $j \in C$ **do**
3: **if** $MCJE(ij) = 0$ **then**
4: $CL$:=$CL \cup \{j\}$;
5: **end if**
6: **end for**
7: **return** $CL$

図 4: 枝刈りと閉包を考慮したアルゴリズム

### 4.2 位置集合の閉包性を利用した高速化

**定義 3** 入力文字列集合を $S \subseteq \Sigma^m$ とする．関係 $\equiv_S \subseteq [m] \times [m]$ を，

$$「i \equiv_S j \Longleftrightarrow H(i|j) = H(j|i) = 0」$$

によって定義される関係とする．

関係 $\equiv_S$ は同値関係であり，$[m]$ は $S \subseteq \Sigma^m$ によって同値類分割可能である．同値関係 $\equiv_S$ による同値類で代表元 $i$ を持つ同値類を $[i]_{\equiv_S}$ で表す．

**補題 2 (閉包性)**
位置の組を $(i, j)$ とおく．任意の $I \subseteq [i]_{\equiv_S}$ に対して，$MCJE(ij) = MCJE(Ij)$ である．

**定理 2** 入力文字列集合 $S \subseteq \Sigma^m$ に対し，FindCRMPCは，$MCJE(i_1 \cdots i_k) \leq \epsilon$ を満たす全ての位置集合 $i_1 \cdots i_k \subseteq [m]$ を $O(nmM) = O(n2^m)$ 時間で列挙する．ここに，$M = O(2^m)$ は解となる位置集合 $I$ の個数である．

### 4.3 ベンチマークテスト

Ruby 言語を用いて FindCRM, FindCRMP, FindCRMPC アルゴリズムの実装を行った．ベンチマークには，NCBI の Infuluenza Viruse Resource[3] から，H3N2 亜型のインフルエンザウイルスの HA アミノ酸配列を 2837 本取得し，ランダムサンプリングによって長さと個数が異なる入力文字列集合を多数作成して利用した．表 1 に $n = 294$ 本のサンプル配列に対し，HA1 領域の全長 $m = 328$ を対象として，$\epsilon$ を変化させた結果を示す．FindCRMPC 以外は列挙が終了せず，60 分で計算を打ち切った．$\epsilon$ を 0.0 と設定しても FindCRM および FindCRMP による列挙が完了しない理由は，アミノ酸残基が全く変化しない位置が 144 箇所あり，これらのどの部分集合 $I$ に対しても $MCJE(I) = 0$ となり，計 $2^{144}$ 回の探索が行われるためである．一方，FindCRMPC は当該箇所をあらかじめ $O(m^2)$ 時間で同値類としてまとめ，以後，代表元のみを用いて計算を行う．

| $\epsilon$ | 0.00 | 00.1 | 00.2 | 00.3 | 00.4 | 00.5 |
|---|---|---|---|---|---|---|
| FindCRM | >3600 | >3600 | >3600 | >3600 | >3600 | >3600 |
| FindCRMP | >3600 | >3600 | >3600 | >3600 | >3600 | >3600 |
| FindCRMPC | 0.3856 | 21.514 | 21.562 | 21.727 | 45.102 | 45.034 |
| num | 8 | 8 | 8 | 8 | 1059 | 1059 |

表 1: 各プログラムの計算時間 (sec) および検出された共変異の個数

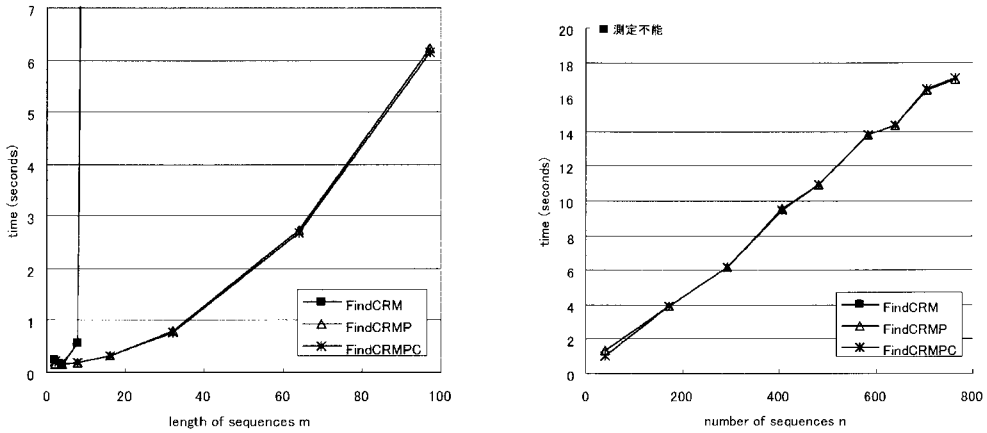


図 5: 文字列の長さ $m$，文字列の数 $n$ における実行速度の比較

次に，4.1 節で述べた $\tau$ によるヒューリスティクスを用いて，解析対象の位置を限定し，FindCRM, FindCRMP, FindCRMPC のベンチマークテストを行った．パラメータ $\epsilon$ と $\tau$ は共に 0.1 とした．図 5 に文字列の長さ $m$，文字列の数 $n$ における実行速度の変化を示す．図 5 左においては，文字列数 $n$ を 294 に固定し，文字列の長さ $m$ を変化させて実験を行った．FindCRM と FindCRMP, FindCRMPC の結果を比べると枝刈りの著しい効果が判る．図 5 右においては，$m$ を 328 に固定し，$n$ を変化させて実験を行った．FindCRMP, FindCRMPC の実行時間は，文字列数 $n$ 対して線形であることがわかる．FindCRM はの実行時間は前頁と同様に計測不可能であった．

## 5 おわりに

本研究では，塩基やアミノ酸の配列から，同時に変異する傾向にある塩基または残基位置の集合を列挙する問題を議論した．共変異の検出において，1) 条件付きエントロピーを指標として用いることが有効であること，2) 条件付きエントロピーの指標を条件付き同時エントロピーへと自然に拡張することにより，三つ以上の位置における共変異を検出可能であること，3) 条件付き同時エントロピーの単調性を利用した枝刈り規則と，位置集合と配列集合に関する閉包性を用いることにより，共変異位置集合の列挙アルゴリズムを著しく高速化することが可能であることを示した．

## 謝辞

本研究は，文部科学省「新興・再興感染症拠点形成プログラム」および科学研究費・基盤研究 (B)「インフルエンザウイルスの抗原変異予測のためのパターン発見手法に関する研究」(研究代表者・伊藤公人，課題番号：19300041)，特別推進研究「知識基盤形成のための大規模半構造データからの超高速パターン発見」(研究代表者：有村博紀　課題番号：17002008) の支援を受けて実施した．

# 参考文献


[1] D. Altschuh, A. M. Lesk, A. C. Bloomer, and A. Klug. Correlation of coordinated amino-acid substitutions with function in viruses related to tobacco mosaic-virus. *Journal of Molecular Biology*, 193(4):693–707, 1987.

[2] W. R. Atchley, K. R. Wollenberg, W. M. Fitch, W. Terhalle, and A. W. Dress. Correlations among amino acid sites in bhlh protein domains: An information theoretic analysis. *Molecular Biology and Evolution*, 17(1):164–178, 2000.

[3] Y. M. Bao, P. Bolotov, D. Dernovoy, B. Kiryutin, L. Zaslavsky, T. Tatusova, J. Ostell, and D. Lipman. The influenza virus resource at the national center for biotechnology information. *Journal of Virology*, 82(2):596–601, 2008.

[4] Xiangjun Du, Zhuo Wang, Aiping Wu, Lin Song, Yang Cao, Haiying Hang, and Taijiao Jiang. Networks of genomic co-occurrence capture characteristics of human influenza a (h3n2) evolution. *Genome Res.*, 18(1):178–187, 2008.

[5] S. R. Eddy and R. Durbin. Rna sequence-analysis using covariance-models. *Nucleic Acids Research*, 22(11):2079–2088, 1994.

[6] M. A. Fares and S. A. A. Travers. A novel method for detecting intramolecular coevolution: Adding a further dimension to selective constraints analyses. *Genetics*, 173(1):9–23, 2006.

[7] W. M. Fitch and E. Markowitz. An improved method for determining codon variability in a gene and its application to the rate of fixation of mutations in evolution. *Biochem Genet*, 4(5):579–93, 1970.

[8] G. B. Gloor, L. C. Martin, L. M. Wahl, and S. D. Dunn. Mutual information in protein multiple sequence alignments reveals two classes of coevolving positions. *Biochemistry*, 44(19):7156–7165, 2005.

[9] U. Gobel, C. Sander, R. Schneider, and A. Valencia. Correlated mutations and residue contacts in proteins. *Proteins-Structure Function and Genetics*, 18(4):309–317, 1994.

[10] M. Kolli, S. Lastere, and C. A. Schiffer. Co-evolution of nelfinavir-resistant hiv-1 protease and the p1-p6 substrate. *Virology*, 347(2):405–409, 2006.

[11] B. T. M. Korber, R. M. Farber, D. H. Wolpert, and A. S. Lapedes. Covariation of mutations in the v3 loop of human-immunodeficiency-virus type-1 envelope protein - an information-theoretic analysis. *Proceedings of the National Academy of Sciences of the United States of America*, 90(15):7176–7180, 1993.

[12] A. S. Lapedes, B. G. Giraud, L. C. Liu, and G. D. Stormo. Correlated mutations in protein sequences: phylogenetic and structural effects. *AMS/SIAM Conference on Statistics and Molecular Biology, Seattle*, 1997.

[13] S. M. Larson, A. A. Di Nardo, and A. R. Davidson. Analysis of covariation in an sh3 domain sequence alignment: Applications in tertiary contact prediction and the design of compensating hydrophobic core substitutions. *Journal of Molecular Biology*, 303(3):433–446, 2000.

[14] L. C. Martin, G. B. Gloor, S. D. Dunn, and L. M. Wahl. Using information theory to search for co-evolving residues in proteins. *Bioinformatics*, 21(22):4116–4124, 2005.

[15] T. Nagano, M. Suwa, and K. Asai. Correlated mutation analysis of c2h2 zinc finger domains. *Geneme Informatics*, 14:565–566, 2003.

[16] O. Noivirt, M. Eisenstein, and A. Horovitz. Detection and reduction of evolutionary noise in correlated mutation analysis. *Protein Engineering Design and Selection*, 18(5):247–253, 2005.

[17] F. Pazos, M. HelmerCitterich, G. Ausiello, and A. Valencia. Correlated mutations contain information about protein-protein interaction. *Journal of Molecular Biology*, 271(4):511–523, 1997.

[18] L. Pritchard, P. Bladon, J. M. O. Mitchell, and M. J. Dufton. Evaluation of a novel method for the identification of coevolving protein residues. *Protein Engineering*, 14(8):549–555, 2001.

[19] A. Rzhetsky. Detecting coevolution in and among protein domains. *PLoS Comput Biol*, 3(11):e211, 2007.

[20] I. N. Shindyalov, N. A. Kolchanov, and C. Sander. Can 3-dimensional contacts in protein structures be predicted by analysis of correlated mutations. *Protein Engineering*, 7(3):349–358, 1994.

[21] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. *Discovery Science, Proceedings*, 3245:16–31, 2004.

[22] C. H. Yeang, J. F. J. Darot, H. F. Noller, and D. Haussler. Detecting the coevolution of biosequences - an example of rna interaction prediction. *Molecular Biology and Evolution*, 24(9):2119–2131, 2007.

[23] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000.