DEWS2004 I-7-04

大規模系列データのための近似系列パターン発見アルゴリズム

内田　雄三†　　浅井　達哉†　　有村　博紀†

† 九州大学大学院システム情報科学研究科・研究院,
〒 812-8581 福岡市東区箱崎 6-10-1
E-mail: †{y-uchida,t-asai,arim}@i.kyushu-u.ac.jp

あらまし　近似エピソードパターンとは，系列と最大編集距離の組であり，本稿では, 挿入, 削除, 変換の三つの編集操作を許した上で，与えられた閾値以上の頻度で系列データに出現する近似エピソードパターンを発見する近似エピソードマイニングについて考察する．我々は，パターンを拡張しながら，漸増的に生成されたパターンの出現頻度を計算し，与えられた系列データベースに頻出する近似エピソードを高速に発見するアルゴリズムを提案する．さらに，アルゴリズムの正当性と，実験による評価を与える．

キーワード　データマイニング，知識発見，テキスト DB，バイオインフォマティクス，Web とインターネット

Discovering Frequent Approximate Episodes in Large Sequence Databases

Yuzo UCHIDA†, Tastuya ASAI†, and Hiroki ARIMURA†

† Department of Informatics, Kyushu University,
6-10-1 Hakozaki, Fukuoka, 812-8581, Japan
E-mail: †{y-uchida,t-asai,arim}@i.kyushu-u.ac.jp

Abstract　In this paper, we propose a framework of approximate sequence mining, where insert, delete, and exchange edit operations are allowed. Then, we present a practically fast algorithm for finding the frequent approximate sequences extending a pattern-growth type frequent sequence mining algorithms. Furthermore, we give a modification of the basic algorithm tailored for sparse datasets.

Key words　Data Mining, Discovery of knowledge, Database of text, Bioinformatics, Web and Internet.

1. はじめに

近年，構造のない系列データからのデータマイニング（系列マイニング）が注目を集めている．通信記録からの異常検出や，遺伝子情報配列からのモチーフ発見，テキストデータからのキーワードパターン発見などは，その例である．1995 年に Mannila 等 [3], [4] は，窓付きエピソードパターンと呼ばれる系列パターンを高速に発見するアルゴリズムを提案した．しかし，遺伝子配列や，ノイズを含むテキストデータからのマイニングでは，より表現力の高いパターンが必要である．

本稿では，イベントの置換重み行列と挿入・削除・置換の三つの編集規則を用いてエピソードパターンを拡張し，近似系列パターンの族を導入する．このクラスは，窓付きエピソードパターン，部分系列パターン，部分文字列パターンの自然な一般化になっている．このクラスに対して，系列列挙技法 [1] とパターン成長技法 [5] を用いた高速なアルゴリズムを提案する．

具体的には，系列データベースからの近似系列パターン発見について考察し，全ての頻出近似エピソードを，エピソード一つあたり $O(n)$ 時間で列挙する効率よいマイニングアルゴリズム Array を与える．ここに n は，列挙されたエピソードが出現するイベント系列の長さの総和である．これは，長さ m のエピソード一つ当り $O(mn)$ 時間を要する素朴な生成検査法アルゴリズム Naive に比べて高速である．

一方，多くの実際的な系列データでは，パターンの出現が非常に疎であることが珍しくない．そこで，上のアルゴリズムをもとに，疎な系列データ上でのエピソード発見について研究した．結果として，近似エピソード P の出現位置の総数を ℓ_P としたとき，全ての頻出近似エピソードを，エピソード一つあたり $O(\ell_P)$ 時間で列挙する効率よいマイニングアルゴリズム List を与える．一般に，総出現数 ℓ_P は出現するイベント系列長の総和 n に比べて小さいと期待できる．したがって疎な系列データに対して，List は Array に比べて高速だと予想される．

最後に，Naive と Array，List は，C++で実装し，DNA 配列データと，アミノ酸配列データ，英文テキストデータを用

いて計算機実験を行った．実験では，Naive に比べて 2 倍程度高速であり，テキストデータのような疎なデータでは 10 倍近く高速化された．また，計算時間に関して挿入のみが高速であり，削除と置換の振る舞いは似ていた．

1.1 関 連 研 究

従来の手法に窓なしエピソードマイニングアルゴリズム Pei と Han 等 [5] や窓付きエピソードマイニングアルゴリズム Mannila と Toivonen 等 [3] [4] の論文がある．提案の手法のもとになっている編集距離について Wagner と Fischer [8] の論文がある．系列パターンのマイニングアルゴリズムとして Zaki [9] や Takeda ら [6] の論文があるが，本稿とは異なるアプローチである．

1.2 本稿の構成

2 節で，近似エピソードに関する基本的な定義と結果を導入する．3 節で，近似エピソード発見アルゴリズムを与える，さらに正当性と計算時間を調べる．4 節で実験結果を述べる．5 節で，まとめを述べる．

2. 近似エピソード

2.1 イベント系列

Σ を**イベント**の集合とする．イベント $e \in \Sigma$ とは，単一の数や文字または複数の数や文字列とする．例えば，ネットワークイベントや DNA 塩基等はイベントの例である．**イベント系列**とはイベントの順序列 $S = e_1, e_2, \ldots, e_n$ $(n \geq 0)$ で与える．このとき，S の k 番目のイベント e_k を $S[k]$ で表し，S の長さを $|S| = n$ とする．また，$|S| = 0$ のとき $S = \varepsilon$ を空イベントとする．イベント系列を α, β, γ とおく．α と β の連接を $\alpha\beta$ または $\alpha \cdot \beta$ と書く．系列 $S = \alpha\beta\gamma$ に対して，β を S の部分文字列という．

S を長さ n のイベント系列とする．S 上の**区間**とは，組 $I = [i : j] (0 \leqq i \leqq j \leqq n)$ である．このとき，区間 I の**幅**を $|I| = j - i + 1$ とする．イベント系列 S と S 上の区間 $[i, j]$ に対して，部分系列 $S[i, j] = S[i], S[i+1], \ldots, S[j]$ と定義する．但し，$i > j$ のとき $S[i : j] = \varepsilon$ とする．

2.2 編 集 距 離

Σ をイベント集合とする．**編集操作**を，イベントの順序組 $D = (a, b) \in (\Sigma \cup \{\varepsilon\})^2$ とし，$a \to b$ で表す．$e, e' \in \Sigma$ を任意のイベントとする．ある編集操作 $a \to b$ が $\varepsilon \to e$ の形であるとき**挿入 (insert)** といい，$e \to \varepsilon$ であるとき**削除 (delete)**，そして $e \to e'$ であるとき**置換 (exchange)** という．イベントの**置換行列 (距離行列)** とは，実数の関数 $\gamma : (\Sigma \cup \{\varepsilon\})^2 \to \mathbf{R}$ とする．$\gamma(a, b)$ を $a \to b$ の**コスト**という．置換行列 γ は，次の条件が成立すると仮定する．

- (反射性) 任意の $a \in \Sigma \cup \{\varepsilon\}$ について，$\gamma(a, a) = 0$.
- (対称性) 任意の $e_1, e_2 \in \Sigma$ について，
  $\gamma(e_1, e_2) = \gamma(e_2, e_1)$.
- (三角不等式) 任意の $a, b, c \in \Sigma \cup \{\varepsilon\}$ について，
  $\gamma(a, b) + \gamma(b, c) \geqq \gamma(a, c)$.

$\mathcal{D} \subseteqq (\Sigma \cup \{\varepsilon\})^2$ を編集操作の集合とする．**編集系列**を $\mathbf{D} = D_1, \ldots, D_m (m \geq 0) \in \mathcal{D}^*$ とする．γ 上の編集系列 $\mathbf{D}$ のコスト $\gamma(\mathbf{D})$ とは，各編集操作 D_i のコスト $\gamma(D_i)$ の総和 $\gamma(\mathbf{D}) = \sum_{i=1}^{m} \gamma(D_i)$ である．

α, β をイベント系列とする．編集操作 $D = a \to b$ によって α が β に編集されるとは，$\alpha = \alpha_1 a \alpha_2, \beta = \alpha_1 b \alpha_2$ となる $\alpha_1, \alpha_2 \in \Sigma^*$ が存在することである．このとき，D による α から β への編集を $\alpha \overset{1}{\underset{D}{\to}} \beta$ と表す．編集系列 $\mathbf{D} = D_1, \ldots, D_m (m \geq 0)$ によって α が β に編集されるとは，$\alpha \overset{*}{\underset{\mathbf{D}}{\to}} \beta$ と表し，次のように定義する．

$$\alpha \overset{*}{\underset{\mathbf{D}}{\to}} \beta \Longleftrightarrow \alpha = \alpha_0 \overset{1}{\underset{D_1}{\to}} \alpha_1 \overset{1}{\underset{D_2}{\to}} \alpha_2 \overset{1}{\underset{D_3}{\to}} \cdots \overset{1}{\underset{D_m}{\to}} \alpha_m = \beta$$

このとき，γ 上の α から β への**編集距離** $ED(\alpha, \beta, \gamma)$ とは，α から β を生成する $\mathcal{D}$ 上の全ての編集系列 $\mathbf{D}$ のコスト $\gamma(\mathbf{D})$ の最小値のことである．

$$ED(\alpha, \beta, \gamma) = min\{\gamma(\mathbf{D}) \mid \alpha \overset{*}{\underset{\mathbf{D}}{\to}} \beta \text{なる編集系列 } \mathbf{D} \in \mathcal{D}^*\}$$

［例 1］ イベント系列 $S = abab$ と $T = acba$ の編集距離は 2 である．まず，S の 4 文字目に $(b \to \varepsilon)$ を，次に 1 文字目に (ε, c) を順に適用することで，2 回の編集で T が得られる．一方，S からただ 1 回の編集では T は得られないことは明らか．

2.3 近似エピソード

近似エピソード P とは，組 $P = (\alpha, k)$ である．ここに，α は長さ m の編集系列であり，$k \geqq 0$ は非負の整数で，編集距離の閾値とする．エピソード $P = (\alpha, k)$ に対して，末尾にイベント $e \in \Sigma$ を付加して得られるエピソードを $P \cdot e = ((\alpha \cdot e),\ k)$ と書く．近似エピソードのクラスを $\mathcal{A}P = \Sigma^* \times \mathbf{N}$ と表わす．

S をイベント系列とする．γ の下で近似エピソード $P = (\alpha, k)$ が S に**出現する**とは，S のある部分文字列 β に対して，$ED(\alpha, \beta, \gamma) \leqq k$ が成立することを言う．このとき，$P \sqsubseteqq_{\mathrm{ap}} S$ と書く．このとき，$\beta = S[i, j]$ となる区間 $I = [i, j]$ を P の**出現区間**と呼ぶ．

2.4 データマイニング問題

Σ 上の系列データベースは，集合 $\mathcal{S} = \{S_1, \ldots, S_\ell\} (\ell \geqq 0)$ である．ここに，各 $S_i \in \Sigma^*$ は Σ 上のイベント系列である．$\mathcal{S}$ における近似エピソード $P = (\alpha, k)$ の**頻度**とは，P が出現する $\mathcal{S}$ 中の系列数

$$freq(P, \mathcal{S}, \gamma) = |\{S \in \mathcal{S} \mid P \sqsubseteqq_{ap} S\}|$$

である．**最小頻度**とよぶ正整数 σ を仮定する．近似エピソード P は，頻度が σ 以上のとき，頻出であるという．われわれのデータマイニング問題は，次の頻出近似エピソード発見問題である．

頻出近似エピソード発見問題

入力: イベント集合 Σ 上の系列データベース $\mathcal{S}$，および
置換行列 $\gamma : (\Sigma \cup \{\varepsilon\})^2 \to \mathbf{R}$，非負の整数 $\sigma, k \geqq 0$.

出力: $\mathcal{S}$ における頻度が σ 以上であるような最大編集距離 k の全ての近似エピソード P.

2.5 近似文字列照合の動的計画法アルゴリズム

近似エピソードとイベント系列の照合には，よくしられた動

Algorithm NAIVE
入力: イベント集合 Σ, Σ 上の系列データベース $\mathcal{S} = \{S_1, \ldots, S_\ell\}$, 最小サポート値 $\sigma \geqq 0$, 編集距離の閾値 $k \geqq 0$.
出力: $\mathcal{S}$ に頻度 σ 以上で出現するすべての近似エピソード.
1 $S_i \in \mathcal{S}$ に対して, $Occ(\varepsilon, \mathcal{S}) = \{1, \ldots, |S_i|\}$ とする.
2 EXPAND-NAIVE$(\varepsilon, Occ(\varepsilon, \mathcal{S}))$

図 1 素朴なアルゴリズム

的計画法による近似パターン照合アルゴリズムを用いることができる [6], [8]. このアルゴリズムは, 次のように定義される 2 次元の表 $\mathcal{B} : (m+1) \times (n+1) \to \mathbf{N}$ を構築する [8].

[定義 2] $\mathcal{B}(i,j) = min\{ED(\alpha[1:i], S[\ell:j], \gamma) \mid 1 \leqq \ell \leqq j+1\}$ $(0 \leqq i \leqq m, 0 \leqq j \leqq n)$

このとき, $\mathcal{B}(m,j) \leqq k$ なる $0 \leqq j \leqq |S|$ が求める解となる. ただし, $m = |\alpha|$ である. この表において, 行 $\mathcal{C}[i,*]$ を高さ i の行という. 表 $\mathcal{B}$ の動的計画法による効率よい計算には次の二つの補題を用いる.

[補題 3] 任意の $i, j (1 \leqq i \leqq m, 0 \leqq j \leqq n)$ について $\mathcal{B}(0,j) = 0$, $\mathcal{B}(i,0) = \sum_{i=1}^{m} \gamma(\alpha[i] \to \varepsilon)$ が成立する.

[補題 4] 任意の $i, j (1 \leqq i \leqq m, 1 \leqq j \leqq n)$ について, 次式が成立する. $\mathcal{B}(i,j) = min\{$(i) $\mathcal{B}(i-1,j-1) + \gamma(\alpha[i] \to S[j])$, (ii) $\mathcal{B}(i-1,j) + \gamma(\alpha[i] \to \varepsilon)$, (iii) $\mathcal{B}(i,j-1) + \gamma(\varepsilon \to S[j])\}$

[補題 5] 上の動的計画法に基づくアルゴリズムは, 近似パターン照合問題を $O(mn)$ 時間と $O(mn)$ 領域で解く.

頻出近似エピソードの計算のための一つのアプローチとして, はじめに何らかの方法で候補となるエピソードを列挙しておき, 次に系列データベースの各系列に対して, 上の動的計画法に基づく近似パターン照合により出現数を計算する. この方法は, パターン一つ当り $O(mn)$ 時間と $O(mn)$ 領域で出現数を計算する. Takeda 等 [6] は, このアプローチをビット並列技法を用いて改良し, パターン一つ当り $O(mn/\log m)$ 時間と領域のアルゴリズムを与えている.

3. 近似エピソードマイニングアルゴリズム

頻出近似エピソード問題を解く三つのアルゴリズムを示す. 一つは, 列挙された近似エピソードを動的計画法から素朴なアルゴリズム, 次に素朴なアルゴリズムよりも効率よく解く配列型アルゴリズムを与える. そして, 疎なデータを配列型アルゴリズムより効率よく解くリスト型アルゴリズムを与える.

この章において, 以降では, 入力として与えられる系列データベース $\mathcal{S}$ と最大編集距離 $k \geqq 0$, 最小頻度 $\sigma \leqq 0$ を固定する. 列挙された近似エピソード $P = (\alpha, k)$ とイベント系列 $S \in \mathcal{S}$ のそれぞれの長さを m と n と書く. また, 近似エピソード $P = (\alpha, k)$ を単にエピソード α と書く.

3.1 素朴なアルゴリズム

頻出近似エピソード発見問題を素直に解く素朴なアルゴリズムを与える. 図 1 に素朴なアルゴリズムを示す.

固定した最大編集距離を $k \geqq 0$ とする. このアルゴリズムは, 空エピソードからはじめて, エピソードの末尾にイベントを一つずつ追加しながら, すべての近似エピソードを列挙する. すなわち, 次の系列列挙木を深さ優先順に巡回する:

```
//A subroutine for NAIVE
Proc EXPAND-NAIVE(α, Occ(α, S))
入力: イベント系列 α, 出現リスト集合 Occ(α, S)).
1  If freq(α, S, γ) < σ then return.
2  Else Output α;
3  Foreach e ∈ Σ do:
4    Foreach S ∈ S do:
5      Occ(αe, S_i) =MATCH(α · e)
6    EXPAND-NAIVE(α · e, Occ(α · e, S))
```

図 2 素朴なアルゴリズム：エピソードの拡張手続き

```
//A subroutine for NAIVE
Proc MATCH(α, S)
入力: エピソード α.
出力: 末尾出現集合 Occ(α, S).
1  Foreach j = 0, ..., |S| do:
2    B[0, j] := 0.
3  Foreach i = 1, ..., |α| do:
4    B[i, 0] := B[i − 1, 0] + γ(α[i] → ε).
5    Foreach j = 1, ..., |S| do:
6      REP = B[i − 1, j − 1] + γ(α[i] → S[j]).
7      INS = B[i − 1, j] + γ(α[i] → ε).
8      DEL = B[i, j − 1] + γ(ε → S[j]).
9      ED = min{REP, INS, DEL}.
10 Foreach j = 0, ..., |S| do:
11   if B[|α|, j] ≦ k then do:
12     j を Occ(α, S) に加える.
13 Return Occ(α, S);
```

図 3 素朴なアルゴリズム：近似照合手続き

系列列挙木とは, 次の条件を満たす根付木 $\mathcal{T}$ である.

- $\mathcal{T}$ の節点は頻出エピソードである.
- $\mathcal{T}$ の根は, 空エピソード (ε, k) である.
- エピソード $P = (\alpha e, k)$, $\alpha \in \mathcal{A}P, e \in \Sigma$ が $\mathcal{T}$ の節点ならば, $Q = (\alpha, k)$ も $\mathcal{T}$ の節点である.

この系列列挙木を用いると, すべての系列を一つ当り $O(1)$ 時間で, 辞書順に重複なく列挙できることがわかる. さらに, 次の頻度の単調性から, 近似エピソードにおいてもアイテム集合と同様の頻度の下限による枝刈りができる.

[補題 6] **(単調性)** エピソード P がエピソード Q の先祖ならば, $freq(P, \mathcal{S}, \gamma) \geqq freq(Q, \mathcal{S}, \gamma)$ が成立する.

頻出近似エピソード発見問題を素直に解く素朴なアルゴリズムを与える. 図 1 に素朴なアルゴリズムを示す. 素朴なアルゴリズムは, 前の章で述べた系列列挙木を用いて, 空エピソード ε から拡張イベント e を一つずつ末尾拡張しながらすべてのエピソードを辞書順に重複なく列挙する.

系列列挙木からエピソード α が列挙されたら, そのエピソード α が各イベント系列 $S \in \mathcal{S}$ に対して, 近似照合を行い, 出現頻度を計算すればよい. 区間 $[i,j]$ が, S における P の出現区間ならば, 位置 j を P の**末尾出現** (tail occurrence) という. このアルゴリズムでは, $Occ(P, \mathcal{S})$ ですべての系列に関する末尾出現集合を表す. このとき, イベント系列 $S \in \mathcal{S}$ に対する出

現頻度の計算には，S における P の近似照合に関する末尾出現集合

$$Occ(P,S) = \{\ 1 \leqq j \leqq |S|\,|\,[i,j] \text{ は } P \text{ の } S \text{ 上の近似出現 }\ \}$$

を求めればよい．

これは，パターン P とテキスト S に対して，先の動的計画法に基づく近似照合手法で構築する表 $\mathcal{B}[1..m, 1..n]$ を考え，その最下段の行 $\mathcal{B}[m] \stackrel{def}{=} (\mathcal{B}[m,1], \ldots, \mathcal{B}[m,n])$ から容易に求められる．

[補題 7]　任意の $j = 1, \ldots, n$ に対して，$j \in Occ(P,S)$ が成立することと，$\mathcal{B}[m,j] > 0$ が成立することは同値である．

以上のことから，図 2 の素朴なアルゴリズムの拡張手続きと，図 3 の素朴なアルゴリズムの近似照合手続きが得られる．

このアルゴリズムは，長さ m のエピソードと長さ n のイベント系列につき表全体の構築に $O(mn)$ 時間がかかる．つまり，系列データベース $\mathcal{S}$ におけるあるエピソード α の近似照合の時間計算量は $O(MN)$ である．ここで，M はエピソードの長さで，N は系列データベースの大きさ $N = ||\mathcal{S}||$ である．

3.2　配列型アルゴリズム

この節では，素朴なアルゴリズムの冗長な部分を除くことで効率的に末尾出現集合を得る配列型アルゴリズムについて示す．

図 4 に配列型アルゴリズムの擬似コードを示し，図 5 に配列型アルゴリズムの拡張手続きの擬似コードを示す．素朴なアルゴリズムの問題点として，長さ m のエピソード $\alpha \cdot e$ と長さ n のイベント系列 S に対して，動的計画法の表を構築するのに $O(mn)$ 時間かかる．しかし，系列列挙木を用いたアルゴリズムでは，エピソード $\alpha \cdot e$ の親エピソード α から新しいイベント e を末尾拡張して計算される．このとき，親エピソード α の動的計画法の表と子エピソード $\alpha \cdot e$ の動的計画法の表は，$\mathcal{B}[1..m-1, 1..n]$ が共通する．

この共通する部分を子エピソードで作ることは冗長である．そこで，親エピソードの表の最下列 $\mathcal{B}[m-1]$ をそれぞれのエピソードで情報としてもつことで，子エピソードは表の最下列 $\mathcal{B}[m]$ だけを $O(n)$ 時間で計算可能である．したがって，列挙アルゴリズムに関する結果と合わせて，次の定理を得る．

[定理 8]　$\mathcal{S}$ を系列データベースとし，$k \geqq 0$ を最大編集距離，$\sigma \leqq 0$ を最小頻度とする．このとき，図のアルゴリズム Array は，$\mathcal{S}$ に出現するすべての頻出近似エピソードを，一つ当り $O(n)$ 時間で列挙する．ここに，$n = ||\mathcal{S}||$ は $\mathcal{S}$ の系列の総サイズである．

これは，長さ m のパターン一つ当り $O(mn)$ 時間を要する素朴なアルゴリズムに比べて高速である．

Algorithm Array

入力: イベント集合 Σ，Σ 上の系列データベース $\mathcal{S} = \{S_1, \ldots, S_\ell\}$，最小サポート値 $\sigma \geqq 0$，編集距離の閾値 $k \geqq 0$.

出力: $\mathcal{S}$ に頻度 σ 以上で出現するすべての近似エピソード.

1　$S_i \in \mathcal{S}$ に対して，$Botom(\varepsilon, S_i)[0..|S_i|] = [0..0]$ とする.

2　Expand-Array$(\varepsilon, Botom(\varepsilon, \mathcal{S}))$

図 4　配列型アルゴリズム

//A subroutine for Array

Proc Expand-Array$(\alpha, Botom(\alpha, \mathcal{S}))$

入力: イベント系列 α，出現リスト集合 $Botom(\alpha, \mathcal{S})$).

1　If $freq(\alpha, \mathcal{S}, \gamma) < \sigma$ then **return**.

2　Else Output α;

3　Foreach $e \in \Sigma$ do:

4　　Foreach $S \in \mathcal{S}$ do:

5　　　$Botom(\alpha e, S_i)$ =Update-Array$(e, Botom(\alpha, S_i))$

6　　Expand-Array$(\alpha \cdot e, Botom(\alpha \cdot e, \mathcal{S}))$

図 5　配列型アルゴリズム：エピソードの拡張手続き

//A subroutine for Array

Proc Update-Array(α, S)

入力: 長さ m のエピソード α，出現配列 $Botom(\alpha, S) = \mathcal{B}[m]$.

出力: 出現配列 $Botom(\alpha \cdot e, S) = \mathcal{B}[m+1]$.

1　$\mathcal{B}[m+1, 0] := \mathcal{B}[m-1, 0] + \gamma(\alpha[m] \to \varepsilon)$.

2　Foreach $j = 1, \ldots, |S|$ do:

3　　$REP = \mathcal{B}[m, j-1] + \gamma(\alpha[m] \to S[j])$.

4　　$INS = \mathcal{B}[m, j] + \gamma(\alpha[m] \to \varepsilon)$.

5　　$DEL = \mathcal{B}[m+1, j-1] + \gamma(\varepsilon \to S[j])$.

6　　$ED = min\{REP, INS, DEL\}$.

7　**Return** $Botom(\alpha \cdot e, S)$;

図 6　配列型アルゴリズム：更新手続き

このアルゴリズムでは，各エピソード α で動的計画法の表の最下列を情報を $Botom(\alpha, S) = \mathcal{B}[m]$ で表し，**出現配列**と呼ぶ．$Botom(\alpha, S)$ の j 番目の値 $\mathcal{B}[m,j]$ は，$Botom(\alpha, S)[j]$ で表す．図 6 は配列型アルゴリズムの更新手続きの擬似コードである．

このアルゴリズムは，長さ m のエピソードと長さ n のイベント系列につき表の更新に $O(n)$ 時間がかかる．つまり，系列データベース $\mathcal{S}$ におけるあるエピソード α の近似照合の時間計算量は $O(N)$ である．ここで，N は系列データベースの大きさ $N = ||\mathcal{S}||$ である．

3.3　リスト版アルゴリズム

本節では，配列型アルゴリズムの編集距離配列の値が閾値以下のものだけをリストでもって更新するリスト型アルゴリズムを提案する．

前節の配列アルゴリズムでは，長さ n のイベント系列 $S \in \mathcal{S}$ に対して，長さ m の親エピソード α と子エピソード $\alpha \cdot e$ の動的計画法の表 $\mathcal{B}[1..m+1, 1..n]$ の差分である出現配列 $\mathcal{B}[m+1]$ だけを求めることができたが，補題 4 から次の補題が与えられる．

[補題 9]　親エピソードの出現配列 $\mathcal{B}[m]$ の内 $\mathcal{B}[m,j] > k$ となる (m,j) からは，どのような編集操作を加えても，任意の j' $(j \leqq j' \leqq n)$ について $\mathcal{B}[m+1, j'] \leqq k$ となる j' は存在しない．

このことから，親エピソード α の出現配列 $Botom(\alpha, S)$ の全ての末尾出現から，子エピソード $\alpha \cdot e$ の出現配列 $\mathcal{B}[m+1]$ の全ての末尾出現が求めることができる．ここで，出現配列 $Botom[\alpha, S]$ の全ての末尾出現 j をもつために，各末尾出現 j について，組 $\{j, \mathcal{B}[m,j]\}$ を値とする配列 $Tail(\alpha, S)$ を**末尾出**

Algorithm LIST
入力: イベント集合 Σ, Σ 上の系列データベース $\mathcal{S} = \{S_1, \ldots, S_\ell\}$, 最小サポート値 $\sigma \geqq 0$, 編集距離の閾値 $k \geqq 0$.
出力: $\mathcal{S}$ における頻度が σ 以上となる全ての近似エピソード $P = (\alpha, k)$.
1 各イベント系列 $S_i \in \mathcal{S}$ について, $Tail(\varepsilon, S_i) := \langle 0,0\rangle, \ldots, \langle |S_i|, 0\rangle$;
2 EXPAND-LIST$(\varepsilon, Tail(\varepsilon, \mathcal{S}))$;

図 7 リスト版アルゴリズム

//A subroutine for LIST
Proc EXPAND-LIST$(\alpha, Tail(\alpha, \mathcal{S}))$
入力: イベント系列 α, 末尾出現リスト $Tail(\alpha, \mathcal{S})$.
1 If $freq(\alpha, \mathcal{S}, \gamma) < \sigma$ then **return**.
2 Else Output α;
3 Foreach $e \in \Sigma$ do:
4 Foreach $S \in \mathcal{S}$ do:
5 $Tail(\alpha \cdot e, S_i)$ =UPDATE-LIST$(e, Tail(\alpha, S_i))$
6 EXPAND-LIST$(\alpha \cdot e, Tail(\alpha \cdot e, \mathcal{S}))$

図 8 リスト版アルゴリズム：エピソード拡張手続き

現リストといい，次のように定義する．

[定義 10] $Tail(\alpha, S) = \langle x[1], y[1]\rangle, \ldots, \langle x[t], y[t]\rangle$ を末尾出現リストとする．任意の異なる添え字 i, j について，$i < j$ ならば $x[i] < x[j]$ が成立する．

親エピソード α の末尾出現リスト $Tail(\alpha, S)$ から子エピソード $\alpha \cdot e$ の末尾出現を求める方法は，出現配列を求める方法と基本的には変わらない．補題 9 から親エピソードの末尾出現以外からの編集操作はしない．このことから，子エピソードの末尾出現となる編集操作は，親エピソードの末尾出現からの削除と置換，子エピソードの末尾出現からの挿入の場合だけである．以上の考察から次の補題が与えられる．

[定義 11] $last[i]$ を $(m, x[i]-1)$ から $(m+1, x[i])$ への置換コストと $(m+1, x[i]-1)$ から $(m+1, x[i])$ への挿入コストの最小値とする．

[定義 12] 閾値 k より大きい編集コストおよび編集距離を ∞ とする．

[補題 13] α を長さ m の親エピソード，e を拡張イベント，$\alpha \cdot e$ を子エピソードとする．$Tail(\alpha, S)$ の任意の末尾出現 $x[i](1 \leqq i < t)$ について，$\mathcal{B}[m+1, x[i]] = min\{\mathcal{B}[m, x[i]] + \gamma(e \to \varepsilon), last[i]\}$．$(m+1, pos)$ への編集距離が k より大きくなる $pos(x[i] \leqq pos < x[i+1])$ が存在するとき，$pos \leqq j < x[i+1]$ である全ての位置 j について $\mathcal{B}[m+1, j] = \infty$ となり，また $last[i+1] = \infty$ となる．

以上から，リスト型アルゴリズムが得られる．図は図**??**はリスト型アルゴリズムの更新手続きの擬似コードである．

[定理 14] $\mathcal{S}$ を系列データベースとし，$k \geqq 0$ を最大編集距離，$\sigma \leqq 0$ を最小頻度とする．このとき，図のアルゴリズム LIST は，$\mathcal{S}$ に出現するすべての頻出近似エピソード P を，一つ当り $O(\ell_P)$ 時間で列挙する．ここに，ℓ_P は $\mathcal{S}$ における P の末尾出現の総数である．

疎なデータでは，$\ell_P \ll n$ と考えられるので LIST は ARRAY より，疎なデータに適している．

//A subroutine for LIST
Proc UPDATE-LIST$(e, Occ(\alpha, S))$
入力: イベント e, 長さ t の末尾出現リスト $Tail(\alpha, S)$.
1 $Tail' := \emptyset$.
2 $last := \infty$.
3 **For** $cur = 1, \ldots, t$ **do**:
4 $pos := x[cur]$.
5 **If** $cur = t$ then $next := |S| + 1$.
6 **Else do** $next := x[cur+1]$.
7 $DEL := y[cur] + \gamma(e, \varepsilon)$.
8 $last := min\{DEL, last\}$.
9 $Tail(\alpha \cdot e)$ に $\langle pos, last\rangle$ を加える.
10 $pos := pos + 1$.
11 $REP := y[cur] + \gamma(e, S[pos])$.
12 $INS := last + \gamma(\varepsilon, S[pos])$.
13 $last := min\{REP, INS\}$.
14 **while** $pos < next$ **do**.
15 **if** $last \leqq k$ **do**:
16 $Tail(\alpha \cdot e)$ に $\langle pos, last\rangle$ を加える.
17 **else do** $k := \infty$, **break**.
18 $last := last + \gamma(\varepsilon, S[pos])$.
19 $pos := pos + 1$.
20 **Return** $Tail(\alpha \cdot e, S)$.

図 9 リスト版アルゴリズム：出現更新手続き

4. 実　　験

この章では，提案した頻出近似エピソード問題を解くアルゴリズムについて実験による性能評価を行った．近似エピソードについて，実験 A1 から実験 A3 を行った．近似エピソードの実験は，素朴なアルゴリズム NAIVE と配列型アルゴリズム ARRAY，リスト型アルゴリズム LIST の三つの性能を比較した．

これらのアルゴリズムは，C++言語で実装し，GNU g++でコンパイルした．すべての実験は，PC (Pentium4 3GHz, RAM 2GB, OS WindowsXP) 上で行った．

4.1 デ　ー　タ

実験に使用したデータとして，アミノ酸データ $AMINO495$, 英字テキストデータ TEXT100 を用いた．表 4.1 に各データの名称，データサイズ，異なりイベント数，系列数，系列一つあたりの平均長を示す．

表 1 実験に用いたデータ.

名称	サイズ	$\|\Sigma\|$	系列数	系列平均長	入手場所
AMINO495	11706(byte)	20	495	23.65	GenBank
TEXT100	77012(byte)	77	100	770.12	UCI KDD Archive

データ AMINO495 も GenBank から入手したバクテリアのアミノ酸配列を切り出したものである．英文テキストデータとして，次を用いた．データ TEXT100 は，情報検索用テキストデータ reuters21578 の記事本体から SGML タグを除き，先頭から 100 文を切り出したものである．

AMINO495 のイベント集合 Σ は，$\Sigma_{AMINO} = \{A, C, D, \ldots, W, Y\}$ の 20 文字である．TEXT100 では，各

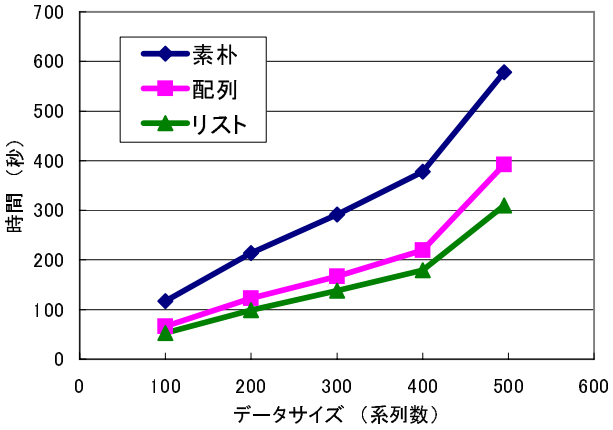

図 10　実験 A1:データサイズによる計算時間 (AMINO495, 最小頻度 70%, k=2).

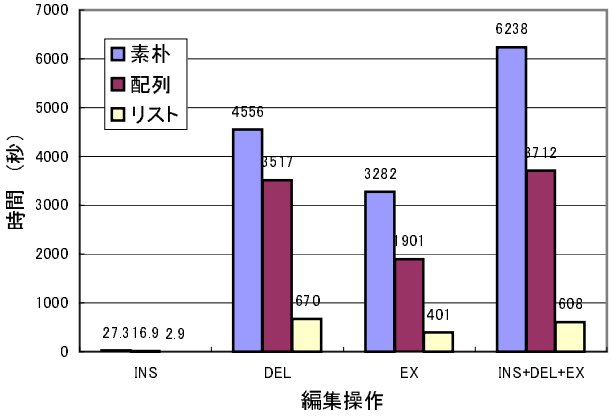

図 11　実験 A1:編集操作と計算時間 (TEXT100, 最小頻度 80%, 系列数 100, k=1).

英字一つを一つのイベントとした.

4.2　頻出近似エピソード発見アルゴリズムによる実験

まず，アルゴリズム毎のよる計算時間の比較をして，本論分で提案する手法の規模耐性を調べた.

実験 A1. 図 10 は，データ AMINO495 の系列数を 100 から 495 まで変化させたときの計算時間を調べた．ここで，簡単のためそれぞれの編集操作のスコアは全て 1 とした．また，最大編集距離を $k = 1$ までとした.

結果は，四つの編集操作の組み合わせ全てで Array は Naive より，List は Array より高速だった．例えば，INS+DEL+EX では，Array は Naive の 1.6 倍高速であり，特に，データ TEXT100 では疎なデータであるため List が有効で List は Naive の 10 倍高速であった.

実験 A2. 図 11 は，三つの編集操作に対し挿入 (INS), 削除 (DEL), 置換 (EX) 単独のときと，全ての編集操作 (INS+DEL+EX) についてデータ TEXT100 上で計算時間を調べた．ここで，簡単のためそれぞれの編集操作のスコアは全て 1 とした．また，最大編集距離を $k = 1$ までとした.

結果は，四つの編集操作の組み合わせ全てで Array は Naive より，List は Array より高速だった．例えば，INS+DEL+EX では，Array は Naive の 1.6 倍高速であり，特に，データ

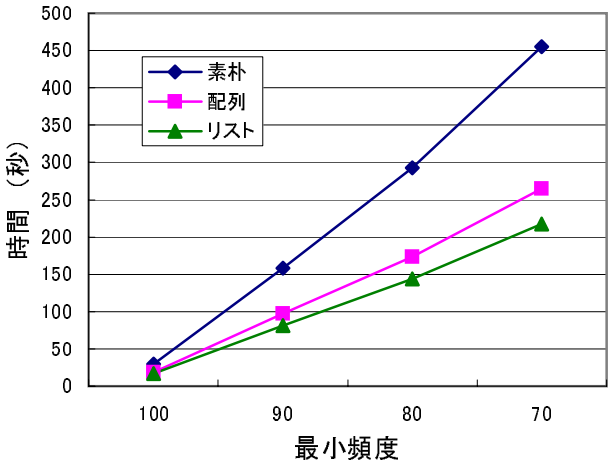

図 12　実験 A2:最小頻度による計算時間 (AMINO495, 系列数 495, 編集操作 INS+DEL+EX, k=2).

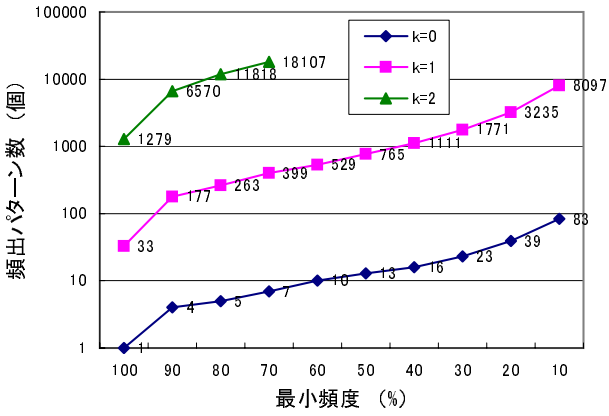

図 13　実験 A3:最小頻度による頻出パターン数 (AMINO495, 系列数 495, 編集操作 INS+DEL+EX).

TEXT100 では疎なデータであるため List が有効で List は Naive の 10 倍高速であった.

実験 A3. 次に，最小頻度に対する計算時間を調べた．データ AMINO495 に対するグラフを図 12 に示す．このグラフから，ほとんど Naive，Array，List の順に速くなっていることがわかる．このデータ AMINO495 では，List は Naive の 2 倍程度高速である.

実験 A4. 図 13 に，実験 A2 と同じデータ AMINO495 で頻出パターンの振る舞いを調べた結果を示す．図から，編集距離が $k = 0, 1, 2$ と大きくなるにつれて，頻出パターンが指数的に増加しており，小さな k のときには見つからない頻出パターンが大きな k の値では見つかることがわかる.

5.　おわりに

本稿では，系列データベースからの系列パターン発見について考察した．はじめに，近似エピソードに対して，系列列挙技法とパターン成長技法のアイデアを用いて，全ての頻出窓ありエピソードを効率よく発見する二つのアルゴリズム Array, List を与えた．この二つのアルゴリズムは共に素朴なアルゴリズム Naive より高速である．特に，疎なデータに対して，List は Array より高速であることが実験で確かめられた．今後の

課題として, [7] の技法による高速化が挙げられる.

本論文では，与えられた置換，挿入，削除の三つの編集操作のみを扱っている．しかし，実際の制約を取り入れた自然なモデルを用いることも重要である．生物情報学分野では，ここで扱った編集距離モデルの拡張として，ギャップの開始と終了に大きなペナルティをつけ，ギャップの拡張には小さなペナルティをつけるアフィンモデルが用いられることが多い [2]．本稿のアルゴリズムは，そのままではアフィンモデルには適用できない．しかし，現在導入している組 (出現位置，編集距離) に加えて，状態 $s \in (open, continue, close)$ を末尾出現リストにもたせることで本稿の枠組みをアフィンモデルに拡張できると思われる．拡張の詳細は今後の課題とする．

謝　　辞

国立情報学研究所の宇野毅明先生には，データマイニングに関する議論の機会をいただきました．ここに感謝致します．また，査読員の方には論文を改善するためのコメントと編集距離モデルの拡張について貴重な示唆をいただきました．ここに謝意を表します.

文　　献

[1] R.J.Bayardo Jr.: “Efficiently Mining Long Patterns from Databases,” In *Proc. SIGMOD*, 85-93, 1998.

[2] M.Bilenko, R.J.Mooney, “Adaptive Duplicate Detection Using Learnable String Similarity Measures,” In *Proc. SIGKDD'03*, ACM, 39-48, 2003.

[3] H.Mannila, H.Toivonen, “Discovering Generalized Episodes using Minimal Occurrences,” In *Proc. KDD*, 146-151, 1996.

[4] H.Mannila, H.Toivonen, A.I.Verkamo, “Discovering Frequent Episodes in Sequences,” In *Proc. KDD*, 210-215, 1995.

[5] J.Pei, J.Han, B.Mortazavi-Asl, H.Piano, “PrefixSpan: Mining Sequential Patterns by Prefix-Projected Pattern Growth,” In *Proc. ICDE*, 215-224, 2001.

[6] M.Takeda, S.Inenaga, H.Bannai, A.Shinohara, and S.Arikawa, “Discovering Most Classificatory Patterns for Very Expressive Pattern Classes,” Proc. *DS 2003*, 486-493, 2003.

[7] T.Uno, T.Asai, Y.Uchida, and H.Arimura, “LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets,” In *Proc. FIMI'03*, 2003.

[8] R.A.Wagner, M.J.Fischer, “The String-to-String Correction Probrem,” In *J. ACM*, 21(1), 168-173, 1974.

[9] M.J.Zaki, “SPADE: An Efficient Algorithm for Mining Sequences,” In *J. ML*, 42(1), 31-60, 2000.