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

2008－DBS－145 (15)
2008－ＦＩ－ 91 (15)
2008／6／20

極小出現区間を用いたエピソードマイニングの高速化

大谷 英行[†*] 喜田 拓也[†] 宇野 毅明[‡] 有村 博紀[†]

[†] 北海道大学大学院情報科学研究科，〒 060-0814 札幌市北区北 14 条西 9 丁目

[‡] 国立情報学研究所，〒 101-8430 東京都千代田区一ツ橋 2-1-2

Abstract: 巨大なデータ集合の中から，有用な知識や情報を発見するデータマイニングは，現代の様々な分野で重要な技術であり，特に，時系列データを対象とした時系列データマイニングが注目を集めている．本論文では，頻出時系列エピソード発見問題を研究する．これは，与えられた系列データから，最大窓幅 w の制約の元で，頻出エピソードと呼ばれる頻出する部分系列を効率的に取り出す問題である．本稿では，高速な頻出集合発見アルゴリズム LCM を元にして，エピソード発見のための深さ優先型探索アルゴリズムといくつかの高速化手法を提示し，計算機実験で評価する．技法の一部は，高速系列パターン発見エンジン LCM_seq に実装されており，本稿はこれらの技法の系列マイニングにおける有効性について考察する．

Efficent Serial Episode Mining with Minimal Occurrences

Hideyuki Ohtani[†] Takuya Kida[†] Takeaki Uno[‡] Hiroki Arimura[†]

[†]Graduate School of IST, Hokkaido University, N14 W9 Kita-ku, Sapporo 060-0814, Japan

[‡]National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

Abstract: Recently, knowledge discovery in large data increases its importance in various fields. Especially, data mining from time-series data gains much attension. This paper studies the problem of finding frequent episodes appearing in a sequence of events. We propose an efficient depth-first-search algorithm for mining frequent serial episodes in a given event sequence using the notion of right-minimal occurrences. Then, we present some techniques for speeding up the algorithm. Finally, we ran experiments on real datasets to evaluate the usefulness of the proposed methods.

1 はじめに

1.1 背景

ネットワークと計算機の性能上昇に伴うデータ量の増大を背景として，巨大なデータから，有用な知識や情報を規則やパターンとして発見するデータマイニング (data mining) の研究が行われている．とくに，データ系列は最も基本的な大規模データの一つであり，系列マイニングの研究が注目を集めている [2, 3, 5, 4, 10]. 例えば医学や生命科学分野では，膨大な記号列である DNA 配列から特徴的な系列を発見することで新しい仮説を立てたり，流通分野では，購買履歴から有効な販売戦略を発見することができる．

1.2 頻出エピソード発見問題

本稿では，**直列エピソード** (episode) と呼ばれる単純なパターンのクラスに対して，頻出パターン発見問題を考察する．これは，与えられた系列データに一定以上の頻度で出現する全ての頻出パターンを発見する問題である．ここに，入力イベント列は，与えられたアルファベット Σ の各要素（イベント）が並んだ系列 S である．頻出エピソードマイニングでは，窓幅 w と最少頻度値 σ が与えられる．窓とは，幅がちょうど w の S の連続部分列 W である．**直列エピソード**は，入力イベント列の中のある窓に出現するイベントの部分系列 $\alpha = a_1 \ldots a_m$ $(a_i \in \Sigma)$ であり，各イベントがイベント系列中の窓に，その順序で出現することを表す．エピソード α の頻度 α は，α を含む窓の総数 $fr(\alpha, S)$ として定め，エピソード α が頻出 (frequent) であるとは，$fr(\alpha, S) \geq \sigma$ をいう．

データが大きくても現実的な時間で列挙するためには，計算量の意味で高速なアルゴリズムが望ましい．一般に，長さ n の入力系列には，最悪時に指数個の頻出エピソードが含まれる．よって，ここでは出力多項式時間のアルゴリズムを考える．本稿の目標は，頻出エピソード発見問題を時間とメモリの両面で，効率よく解く高速なアルゴリズムを与えることである．

1.3 本稿の目的と内容

本稿では系列データから頻出直列パターンを列挙する手法を考察する．初めに，基本的なアルゴリズムとして，PrefixSpan [4] と同様の**深さ優先探索型**を用いたエピソードマイニングアルゴリズム DFS-MO (LCM_seq) を与える．DFS-MO は，パターンの探索にはパターン成長を用いた深さ優先探索を用い，パターンの頻度計算には，右極小出現区間のリストを漸増的に更新する．これにより，簡素かつ効率よい出現区間の更新計算が可能になる．

この深さ優先探索型のエピソード発見アルゴリズム

* 大谷 英行, e-mail: h-ohtani@ist.hokudai.ac.jp

に，頻出アイテム集合発見アルゴリズム LCM アルゴリズムで導入された次の三つの高速化手法 [7] を適用する．

- **振り分け**技法 (occurrence deliver)．パターンの成長に貢献しない文字による拡張を避ける．
- **再帰的データベース縮約**（anytime database reduction)．射影データベースの縮約を高速に行う．
- **飽和パターン発見**（closed episode discovery）．出現同値なエピソードの列挙を回避する．

これらの高速化手法の有効性を評価するために，提案した高速化手法を実装し，実データ上で計算機実験を行った．実験によって，深さ優先探索と出現区間リストを用いたアルゴリズムが効率的であることと，振り分け技法と再帰的データベース縮約技法が計算時間向上に効果的であり，より詳細な情報を持つ飽和エピソードを効率よく発見できることを示す．また，飽和エピソード発見についても有効性を示す．

1.4 本稿の結果

振り分け技法と再帰的データベース縮約技法は，元々，頻出アイテム集合発見において導入されたが，系列マイニングにおける有効性は確かめられていなかった，今回の実験で，これらの手法が系列マイニングにおいても有効であることがわかった．また，従来，飽和直列エピソード（窓付きの飽和系列パターン）については，出力多項式時間の意味で，効率よい頻出パターン発見アルゴリズムは知られていなかった．本稿の結果により，LCM と同様の逆探索 [9] を用いて効率よいパターン発見が可能になることがわかった．

1.5 本稿の構成

本稿の構成は以下のとおりである．第 2 章では，基本的な定義と概念を与え，第 3 章では，深さ優先探索アルゴリズム DFS-MO を与える．第 3 章と 4 章，5 章では，振り分け技法と，再帰的データベース縮約，飽和エピソードによる高速化について述べる．第 6 章では，計算機実験を行い，第 7 章で結論をまとめる．

本稿で考察した深さ優先探索型の直列エピソード発見アルゴリズムは，他の機能とあわせ，公開プログラム LCM_seq として実装・公開されている [8]．今回は，振り分けと飽和パターンについて考察する．

1.6 関連研究

関連研究として，Mannila ら [5] は幅優先探索を用いた頻出エピソード発見アルゴリズムを与えている．頻出アイテム集合系列のマイニングについて，Pei, Han ら [4] は，(記号列に制限したとき）窓なしエピソード（部分系列パターン）に対して，深さ優先探索と極小出現を用いたアルゴリズム PrefixSpan を与えている．Wang と Han [10] は，これを系列の飽和パターンにした BIDE を提案している．内田 [6] は，窓なしエピソードと窓付きエピソードに対して，深さ優先探索と極小出現リストを用いた頻出エピソード発見アルゴリズムを与えている．また，近似照合を用いたパターンも考察している．有村と宇野 [3] は，本稿の窓なしの飽和直列エピソードで連続定数文字列を許した飽和正則パターン（VLDC モチーフ）の発見を考察している．

また，頻出アイテム集合発見の高速化の手法として，LCM アルゴリズムにおいて，宇野ら [7, 9] は振り分けアルゴリズムを与えている。本稿ではこらの技術の系列マイニングえの適用を考察している．

2 準備

本章では，本稿で用いるイベント列，直列エピソードなどの基本的概念を定義し，頻出エピソード発見問題について述べる．

2.1 イベント列

本稿では，各単位時刻に一つのイベントが生起する場合を考えて，次のように定義する．一般の場合については [5] を参照されたい

与えられたアルファベット Σ の各要素 $e \in \Sigma$ を**イベント**(event) と呼ぶ．**入力イベント列**は，イベントの系列 $S = S[1]\cdots S[n]$ $(n \geq 0, S[i] \in \Sigma)$ である．S に対して，正整数 $i < j$ に対して，$S[i]$ で位置 i の文字を表し，$S[i..j] = S[i]S[i+1]\cdots S[j]$ で位置 i から j までの連続部分列を表す．

このとき，**窓の幅**とは任意の正整数 $1 \leq win \leq n$ である．**窓**(window) とは，幅がちょうど w の S の連続部分列 $W = S[i..i + win - 1]$ をいう．S に対して，$\mathcal{W}(S, win) = \{ W_i = S[i..i + win - 1] : i = 1, \ldots, n - win + 1 \}$ を S 上の幅 win の窓の全体とする．窓の総数は $n - win + 1$ である．

S 上の**区間**とは，位置の組 $[i, j] \in \mathbf{N}^2$ $(i \leq j)$ であり，S 上の位置集合 $\{i, i+1, \ldots, j\} \subset \{1, \ldots, n\}$ を表す．二つの区間 $I = [i, j], I' = [i', j']$ に対して，$i' \leq i$ かつ $j \leq j'$ が成立するならば，I が I' に**区間として含まれる**といい，$I \subseteq I'$ と表す．もし $I \subseteq I'$ かつ $I' \not\subseteq I$ ならば，$I \subset I'$ と書く．

2.2 直列エピソードと頻出エピソード発見問題

イベント列中で，ある順番で現れる一連のイベントの並びを，直列エピソードと呼ぶ．ただし，直列エピソードを構成するイベント間に，他のイベントが発生してもよい．

形式的には，**直列エピソード** *(serial episode)*（または単に**エピソード**(episode)）は，入力イベント列のある窓に出現するようなイベントの部分系列 $\alpha = a_1 \ldots a_m$ である．ここで，$m \geq 0, 1 \leq i \leq m, a_i \in \Sigma$ であり，$m = |\alpha|$ をエピソードの**サイズ**という．$\mathcal{C}$ を，全ての直列エピソードのクラスとする．

ある窓 $W_i = W[i..i + w - 1]$ に対して，エピソード α が**窓 W_i に出現する**とは，ある正整数列 $i \leq \phi(1) \leq \cdots \leq \phi(k) \leq i+w-1$ が存在して，任意の $k = 1, \ldots, m$

に対して，$a_k = S[\phi(k)]$ が成立することを言い，$\alpha \sqsubseteq W_i$ と表す．

イベント列 S と窓幅 win が与えられた時，S 中に発生する直列エピソード α の**出現回数** *(occurrence times)* を，次のように定義する．

$$fr(\alpha, S, win) = |\{ W \in \mathcal{W}(S, win) : \alpha \sqsubseteq W \}|$$

すなわち $fr(\alpha, S, win)$ は，イベント列の中で，ある直列エピソードが出現する窓の回数である．

頻度の最小しきい値(minimum freqency threshold) は任意の正整数 σ である．もし $fr(\alpha, S, win) \geq \sigma$ ならば，α は**頻出**(frequent) という．ここでの問題は，与えられたエピソードのクラス $\mathcal{C}$ から，全ての頻出である直列エピソードを見つけることである．S と win, σ によって得られる，頻出である直列エピソードの集合を，**頻出エピソード集合** *(frequent episode set)* と呼び，$\mathcal{F}(S, win, \sigma)$ と書く．

定義 1 頻出エピソード発見問題 *(Frequent Episode Enumeration)* とは，入力としてイベント列 S と頻出しきい値 σ が与えられた時に，S 上の全ての頻出である直列エピソードを出力する問題である．ただし，各頻出直列エピソードは，重複なく出力されるものとする．

例: 入力イベント系列 $S = ABABBCAD$ と，窓幅 $win = 4$，頻度しきい値 $\sigma = 2$ に対して，$\alpha = ABB$ は，2 個の極小出現区間 $[1,4], [3,5]$ をもち，窓 $[1,4], [2,5], [3,6]$ に出現し，窓 $[4,7], [5,8]$ には出現しない．よって窓頻度 $fr = 3$ の頻出エピソードである．また，その部分系列 $\varepsilon, A, AB, B, BB$ は頻出である．

3 右極小出現区間を用いた頻出直列エピソード発見アルゴリズム

本章では，直列エピソードのクラスに対して，頻出エピソード発見問題を解くアルゴリズム DFS-MO を与える．これは，エピソードの生成に**深さ優先探索** *(DFS, Depth-First Search)* を用いて，頻度の計算に**右極小出現区間** *(MO, Minimal Occurrence)* を用いたアルゴリズムである．

3.1 アルゴリズムの概要とパターン成長

図 1 に，基本となる深さ優先探索型の頻出エピソード発見アルゴリズム DFS-MO を示す．このアルゴリズムは，入力イベント系列 S と窓幅 win，最少頻度値 σ を受け取り，再帰手続きを Expand-MO を用いて，次のように全ての頻出エピソードの空間を深さ優先探索する．

初めに，最小のエピソードである空系列 ε から出発する．次に，再帰手続き Expand-MO が頻出直列エピソード α（ここでは**親エピソード**と呼ぶ）に対して呼び出されたとする．このとき，アルゴリズムは，α の末尾に頻出な文字 $c \in \Sigma$ を加えて拡大し，子エピソード $\beta = \alpha \cdot c$ をつくる．

Algorithm DFS-MO(Σ, S, win, σ)
入力: イベント集合 Σ，イベント列 $S = S[1] \cdots S[n] \in \Sigma^*$，窓幅 win，頻出しきい値 σ.
出力: S に頻度 σ 以上で出現する，窓幅 w をもつ全てのエピソード.

1: $MO(\varepsilon) = \{ [i, i] : 1 \leq i \leq n \}$;
2: Expand-MO$(\varepsilon, MO(\varepsilon), S)$;

Algorithm Expand-MO$(\alpha, MO(\alpha), S)$
入力: エピソード α, 極小出現集合 $MO(\alpha)$, イベント列 S.
出力: S に頻度 σ 以上で出現する，エピソードサイズが窓幅以内の全てのエピソード.

1: **if** WinCount$(MO(\alpha \cdot e), S) < \sigma$ **then** return;
2: Output the new episode $\alpha \cdot e$;
3: **for all** $e \in \Sigma$ **do**
4: $\quad MO(\alpha \cdot e) = Update(MO(\alpha), e)$;
5: $\quad$ Expand-MO$(\alpha, MO(\alpha \cdot e), S)$;
6: **end for**

図 1: 右極小出現リストを用いた頻出エピソード発見アルゴリズム

さらに，親エピソードの出現リストを更新して，子エピソードの出現リスト $MO(\beta, S)$ を漸増的に計算する．このために，後の節で説明する副手続き Update と WinCount を用いて，高速に出現リスト計算を行う．

以下では，$n = |S|$ で入力イベント系列のサイズを表し，win で窓幅を，σ で頻度のしきい値を，$k \leq |\Sigma|$ で S 中の頻出イベント数を表す．また，特に混乱を招かないならば，列挙されるパターン α に対して，$m = |MO(\alpha, S)| \leq fr(\alpha, S, win) \leq |S|$ で列挙されるパターン α の極小出現数を表す．

3.2 右極小出現リスト

用語を定義する．以後，長さ n の入力イベント系列を固定し，$S = S[1] \cdots S[n] \in \Sigma^*$ とおく．

直列エピソード α が，S 上のある窓 $W = S[s, e]$ に対して，$\alpha \sqsubseteq W[s, e]$ を満たすとき，α は区間 $[s, e]$ に出現するという．このとき，$I = [s, e]$ を α の**出現区間**という．Mannila ら [5] は，極小出現区間を定義した．

定義 2 ([5]) 直列エピソード α の**極小出現区間** *(minimal occurence)* とは，次を満たす区間 $I = [s, e]$ をいう．

1. α が区間 $W = S[s, e]$ に出現している．
2. $[s', e'] \subset [s, e]$ を満たすような，α の出現区間 $[s', e']$ が存在しない．

これに対して，右極小出現区間を提案する．

Algorithm Update($MO(\alpha), c, S, win$)
入力: 右極小出現リスト $MO(\alpha$，追加するイベント $c \in \Sigma$, イベント列 S，窓幅 win.
出力: 新しいエピソード $\alpha \cdot c$ の右極小出現リスト $MO(\alpha \cdot c, S)$.

```
1: MO(α·c, S) = ∅;
2: for each [s, e] ∈ MO(α) sequentially do
3:   for (j = e + 1, ..., s + win − 1) do
4:     if (S[j] = c) then
5:       MO(α·c) ∪ {[s, j]};
6:       break the inner for-loop;
7:     end if
8: return MO(α·c);
```

図 2: 右極小出現リストの更新アルゴリズム

定義 3 直列エピソード α の**右極小出現区間** *(right-minimal occurence)* とは次を満たす区間 $I = [s, e]$ をいう.

1. α が区間 $[s, e]$ に出現している.
2. $s' = s$ かつ $e' < e$ を満たすような，α の出現区間 $[s', e']$ が存在しない.

補題 1 任意の α の極小出現区間 $[s, e]$ は，右極小出現区間である．一般に，この逆は成立しない.

補題 2 任意の α に対して，$|MO(\alpha, S)| \leq n = |S|$.

PrefixSpan [4] 等の系列発見アルゴリズムでは，出現位置の表現として右端位置 e を用いる．ここでは，直列エピソードの窓制約を満たすために，左端位置 s をあわせた区間 $[s, e]$ として表現している.

補題 3 任意のエピソード α と窓 $W_i = S[i..i+win-1]$ に対して，(i)α が W_i に出現することと，(ii) α のある右極小出現区間 $[s, e]$ に対して $[s, e] \subseteq [i..i + win - 1]$ が成立することは同値である.

上の補題より，窓に基づく頻度の計算には，幅 win 以下の極小出現を保持すれば十分である．窓幅 win とイベント列 S に対して，エピソード α の**右極小出現リスト**を，$MO(\alpha, S) = \{\ [s, e] \mid\ [s, e]$ は α の右極小出現区間かつ $e - s + 1 \leq win\ \}$ と定義する.

アルゴリズム中では，右極小出現リストの要素 $[s, e] \in MO(\alpha, S)$ は，その開始時刻 s の昇順にソートされて，順アクセス可能な形で格納されると仮定する.

3.2.1 右極小出現リストの更新

図 1 の再帰手続き Expand-MO において，末尾に頻出イベント c を付け足して得られたエピソード $\alpha \cdot c$ に対して，その右極小出現リスト $MO(\alpha \cdot c, S)$ を更新する副手続き Update について以下説明する．図 2 にアルゴリズムを示す.

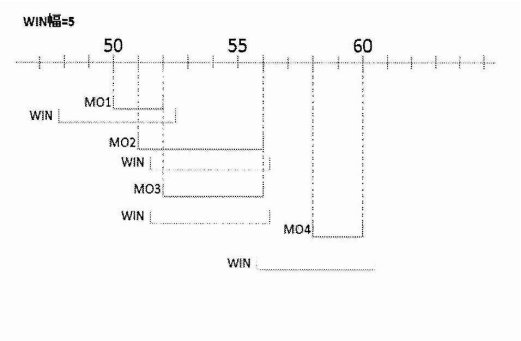

図 3: 頻出区間

アルゴリズム Update の基本的な考え方は，以下のとおりである．右極小出現区間の定義から，ある 2 つの連続した右極小出現区間 $MO1 = (s, e), MO2 = (s', e')$ について考えられる順序は,

1. $s < s'$ and $s' < e$ and $e \leq e'$
2. $s < s'$ and $e \leq s'$ and $e < e'$

である．図 3 で，新しいイベントは，それぞれ位置 52, 56, 56, 60 に存在しており，極小区間はそれぞれそこまで伸ばした状態である.

窓幅 win に対して，あるエピソード α の頻出右極小区間リスト $MO(\alpha) = \{\ [s_i, e_i] : 1 \leq i \leq m\ \}$ について，それぞれの出現区間が (s_i, e_i) で表され，新しく末尾につけるイベントを c とする．この時，新しいエピソード $\alpha \cdot c$ を作るには，$e_i + 1$ から $s_i + win - 1$ までで c の捜索を開始する．もしあればその c の位置を pos に記憶し，$MO(\alpha \cdot e)$ に (s_i, pos) を記録する．以上から，図 3 の $MO2$ は窓幅より長くなり，新しい頻出区間にはならないことが分かる.

補題 4 図 2 のアルゴリズム Update の時間計算量は $O(|MO(\alpha, S)| \cdot win)$ である.

3.2.2 右極小出現リストからの窓頻度の計算

図 1 の再帰手続き Expand-MO において，更新した右極小出現リストから，エピソード出現窓数を求める副手続き WinCount 関数について説明する．図 4 にアルゴリズムを示す.

基本的な考えとして，それぞれの右極小出現区間 $[s, e]$ の最後の位置 e に窓の末尾を合わせ，窓の先頭 $w.s$ が $w.s \leq s$ を満たす限り窓を右側に動かす．窓を動かしている間は，窓の中にエピソードが入っているので，動かした分だけ出現窓数 $count$ をインクリメントする.

ただし，このままでは図 5 のように一つの窓の中に複数個のエピソードが入っている場合に，その窓の中で複数回 $count$ が増やされてされてしまう．よってこれを

Algorithm WinCount($MO(\alpha, S), win$)
入力: 右極小出現リスト $MO(\alpha, S)$, 窓幅 win.
出力: 出現窓数 $count = fr(\alpha, S, win)$.

1: $count = 0$;
2: **foreach** $mo \in MO(\alpha, S)$ sequentially **do**
3: $w.s = mo.e - win + 1$;
4: $w.s = \max\{w.s,\ last\_mo.s + 1\}$
5: **while** $(w.s \leq mo.s)$ **do**
6: $count = count + 1$;
7: $w.s = w.s + 1$;
8: **end while**
9: $last\_mo = mo$;
10: **return** $count$;

図 4: 出現窓数の計算アルゴリズム

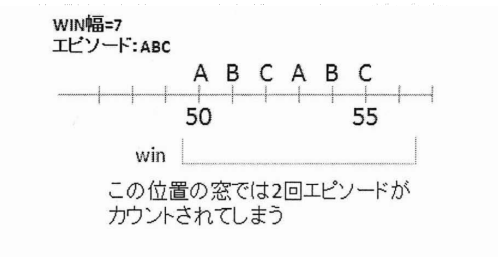

図 5: 一つの窓内に 2 つ右極小出現区間がある場合

回避するために，次のようにする．まず，$MO(\alpha, S) = \{mo_1, \ldots, mo_m\}$ とおく．窓 w を動かしていく場合に，現在の極小区間 mo_i に対して，窓 w が (i) $mo_i \subseteq w$, かつ (ii) $mo_{i-1}.s < w.s$ を満たす場合のみ，$count$ をインクリメントするようにする．ここに，$w.s$ は窓や区間 $w = [s, e]$ の先頭位置 s を表す

窓の先頭位置が戻らないことから，次の補題が示せる．

補題 5 図 4 のアルゴリズム WinCount の時間計算量は $O(\min\{m \cdot win,\ n\}) = O(n)$ である．ここに，$m = |MO(\alpha, S)|$ は α の極小出現数である．

すなわち，窓頻度の計算は，入力長の線形時間を越えず，さらにパターンの成長が進み，極小出現数が小さくなるとそれに応じて短い時間で実行される．これは，win が可変のとき，素朴な更新アルゴリズムの計算時間 $O(m \cdot win) \neq O(n)$ に比べて，効率的である．

以上をまとめると，アルゴリズムの計算時間について次を得る．

補題 6 図 1 のアルゴリズム DFS-MO は，入力イベント列 S と，窓幅 win に対して，すべての頻出エピソードを一つあたり $O(km \cdot win)$ で計算する．ここに，$k \leq |\Sigma|$ は頻出イベント数, $m = |MO(\alpha, S)| \leq n$ である．

4 振り分けアルゴリズム

本章では，3 章で示した DFS-MO をより高速化する手法として，振り分けアルゴリズムを示す．

従来の方法 [4] では，頻出直列エピソード α の末尾に頻出な文字 $e \in \Sigma$ を加えて，拡大エピソード $\alpha \cdot e_i$ をつくる．さらに，頻度計算を行った後，それが頻出ならば探索を継続する．

これに対して，振り分けアルゴリズムは，親エピソードに追加可能な文字 c と対応する右極小出現リスト $MO_c = MO(\alpha \cdot c, S)$ の組を格納したハッシュ表 $Dict = \{\langle c, MO_c\rangle : c \in \Sigma, MO_c = MO(\alpha \cdot c, S), |MO_c| \geq \sigma\}$ を一度の走査で計算する．計算は，再帰手続き Extend-MO の中で，以下のように行う．

Algorithm OccDeliver($\alpha, MO(\alpha, S), win, \sigma$)

1. 親エピソード α に対して，その右極小出現リスト内の各出現区間 $mo \in MO(\alpha, S)$ に対して，末尾 $mo.e$ から窓幅 win で許される時点まで，右に入力系列を走査し，各位置の文字 c についてハッシュ表のエントリ $Dict[c]$ を引き，文字 c の右極小出現リスト $MO(\alpha \cdot c, S)$ にその位置を記録する．ただし，同じ出現区間では c の位置は一つしか記録できない．

2. 上の 1 で親の極小出現リストを走査し終わったら，ハッシュ表に挿入された各キー $c \in Dict.key$ に対して，作成した $MO(\alpha \cdot c, S) = Dict[c]$ を取り出す．もし c に対するエントリが頻出，すなわち $|Dict[c]| \geq \sigma$ ならば，新しい直列エピソードは頻出なので，再帰的に手続き Extend-MO を呼び出す．

3. もし頻出でなければ次のキー c について 2 を繰り返す．すべてのキー c について行ったら，末尾を削った直列エピソードに戻る．

振り分けアルゴリズムのメリットは，ある深さの階層において，末尾に付ける文字が k 個あった際に，1 文字多い直列エピソードを作るにあたって，通常 k 回右極小出現リストを走査する所を，1 度の走査で行える点にある．頻出な文字の個数を $k \leq |\Sigma|$ とすると，補題 6 より，先の DFS-MO の時間計算量は $O(k \cdot |MO(\alpha, S)| \cdot win)$ である．これに対して，振り分け技法により，次のように k 倍の高速化を得る．

定理 1 図 1 のアルゴリズム DFS-MO は，入力イベント列 S と，窓幅 win に対して，すべての頻出エピソード α を一つあたり $O(m \cdot win)$ で計算する．ここに，$m = |MO(\alpha, S)| \leq fr(\alpha, S, win) \leq |S|$ である．

これは，異なるイベント数が多いときに有用である．一方で，DFS-MO では e_i を発見した時点でその出現区間を抜けられるが，振り分けアルゴリズムではそれができない．しかし，振り分けアルゴリズムでも $\mathbf{e}$ を保持しておき，$\mathbf{e}$ の中身すべてが出現した時に次の出現区間に飛ぶことで，その時間を短縮できる．

5 頻出飽和直列エピソード探索

本章では頻出かつ飽和である直列エピソードを右極小区間を用いて求める方法について述べる．

右極小区間出現を用いた飽和エピソード(closed episode)を次のように定義する．エピソードαとβがS上で**右極小出現に関して等価**であるとは，$MO(\alpha, S) = MO(\beta, S)$をいう．明らかに，任意の窓幅$win$に対して，等価なエピソード同士は等しい窓頻度をもつ．エピソードαがS上で**飽和エピソード**であるとは，αが真の部分系列であり，S上で右極小出現に関して等価なより長いエピソードβをもたないことと定義する．このとき，頻出飽和エピソードの族に対して，次の性質を示すことができる．

補題 7 (飽和エピソードの逆探索性) 任意の空でない頻出な飽和エピソード$\beta = \alpha \cdot c$に対して，その末尾の文字$c \in \Sigma$を取り除いて得られるエピソードαは頻出な飽和エピソードである．

上記の性質の証明は文献 [3, 10] と同様のアイディアに基づく．これより，任意の頻出飽和エピソードはあるサイズの一つ小さな頻出飽和エピソードから，末尾に文字を追加して生成されることがわかる．

したがって，飽和エピソードαの探索における分枝限定として，図 1 の再帰手続き Expand-MO の 1 行目で，飽和性の検査を行い，条件が成立しないならば，子孫の探索を打ち切ることで，健全かつ完全に頻出飽和パターンの列挙ができる．飽和性の検査は高々$O(m \cdot win)$時間で可能である．以上の詳細は別稿としたい．

6 実験と考察

本章では，これまでの章で提案したアルゴリズムを実装し，実験による性能評価を行った．

6.1 データ

実験データとして，DNA アルファベット$\Sigma = \{a, t, c, g\}$上の配列データとして，GenBank から取得した DNA 配列データ `dna` と，Calgary Corpus からの取得した DNA 配列データ `ecoli` を用いた．英文テキストデータとして，情報科学関係の英文国際会議論文 1 の PDF から，本文を取得して連結したデータ `paper` を用いた．入力データ `data` のサイズを変化させる際は，先頭からN行をとり出した部分データ `data`N を用いた．詳細について，図 6 に示す．

入力ファイル名	行数	バイト数		入力ファイル名	行数	バイト数
paper1000	1000	49388		paper9000	9000	434092
paper2000	2000	92925		paper10000	10000	482426
paper3000	3000	143582		paper11000	11000	529906
paper4000	4000	194832		paper12000	12000	577833
paper5000	5000	246068		paper13000	13000	626807
paper6000	6000	292970		paper14000	14000	676669
paper7000	7000	342574		paper15000	15000	726683
paper8000	8000	389949		e_coli1000	1000	80000

図 6: 各テキストデータの詳細

6.2 実験環境

アルゴリズムは，全て C++言語で実装し，GNU g++でコンパイルした．全ての実験は，ノート PC (Genuine Intel(R) CPU U2400 1.06Ghz，RAM 1014MB，OS WindowsVista，Cygwin) 上で行った．実験 1 と実験 2 では，3 章で示した右極小出現区間を用いて深さ優先探索を行うアルゴリズムの有効性を示すために，提案の DFS-MO を含み，以下の 3 つのアルゴリズムを実装し，比較した．

- BFS-Scan: 幅優先探索を用い，入力全体を走査し，各窓内で出現の有無を調べて頻度計算を行う．
- BFS-MO: 右極小出現区間を用いた幅優先探索型．
- DFS-MO: 本稿で提案した右極小出現区間を用いた深さ優先探索型．
- DFS-MO-OD: DFS-MO に振り分け技法を実装したもの．
- DFS-CLO: DFS-MO に飽和エピソード発見（非飽和エピソードの枝刈り）を実装したもの．

6.3 実験 1：規模耐性

図 7 に，`paper` データ上で，入力サイズを$N = 1000 \sim 15000$(行) まで変えて，DFS-MO の計算時間（左縦軸）と頻出エピソード数（右縦軸）を示した．窓幅$win = 6$，しきい値σは入力サイズによらず固定の$\sigma = 5000$(約 XX%) とした．このとき窓幅 6，最小しきい値 5000 に固定した．結果として，相対頻度は，10.1% ～ 0.69% まで変化した．

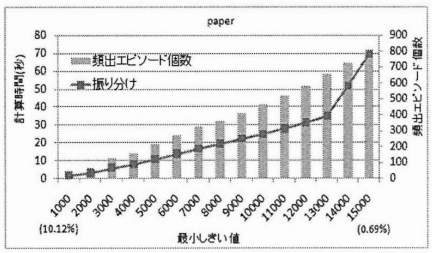

図 7: 入力データサイズに対する計算時間の変化

元データが大きくなるにつれて，発見される頻出エピソード個数は多くなり，また計算時間もかかることが分かる．

6.4 実験 2：極小出現リストの効果

図 8 に，`dna` データ上で，三つのアルゴリズム DFS-MO，BFS-MO，BFS-Scan について，入力サイズを$N = 1000 \sim 30000$(文字) まで変えて，計算時間を比較した結果を示す．窓幅$win = 10$，しきい値σを入力サイズnの 5%とした．

素朴な手法 BFS-Scan に比べて，DFS-MO と BFS-MO は右極小出現区間を用いることで，走査を節約し，計

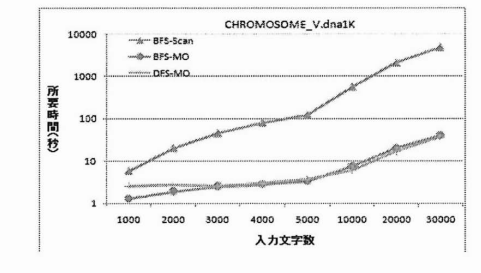

図 8: 入力サイズに対する計算時間の比較

算時間を短縮したことが分かる．速度に関しては，深さ優先と幅優先の間で差は見られない．

6.5 実験 3：深さ優先と幅優先の効果

図 9 に，上記の実験 1 と同じ設定で，入力サイズを $N = 1000$ と 10000(文字) で，DFS-MO，BFS-MO，BFS-Scan のメモリ使用量を比較した結果を示す．

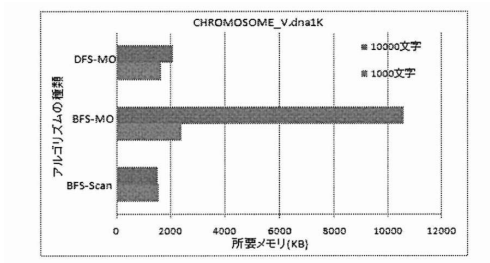

図 9: 入力サイズに対するメモリ使用量の比較

実験 1 で共に高速だった DFS-MO と BFS-MO において，深さ優先型の DFS-MO が，幅優先型の BFS-MO に比べて圧倒的にメモリ効率が良い．

6.6 実験 4：振り分け技法の効果

図 10 に，paper データ上で，窓幅 win を変化させて，DFS-MO と DFS-MO-OD の計算時間（左縦軸）と頻出エピソード数（右縦軸）に示した．データサイズは 10000 行で，しきい値を $\sigma = 5000$（相対頻度は約 1.0%）に固定した．

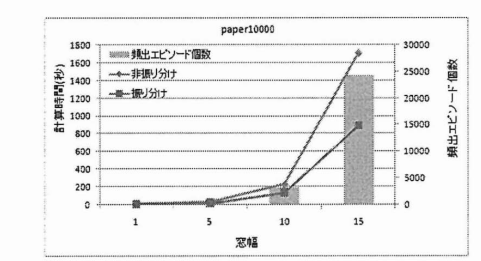

図 10: 窓幅の大きさを変えた場合の振り分けアルゴリズムによる計算時間の比較

窓幅が変化して，解の数が変化する場合でも，振り分け実装の DFS-MO-OD が安定して 2 倍程度高速に計算することがわかる．

6.7 実験 5：振り分け技法の文字種サイズによる影響

図 11 に，paper データ上で最小しきい値 $\sigma = 50 \sim 1000$（相対頻度 0.1 ～ 2.0%）を変化させて，DFS-MO と DFS-MO-OD の計算時間（左縦軸）と頻出エピソード数（右縦軸）を示した．入力文字列の性質による影響を調べるために，

図 12 に，データ ecoli 上でも同じパラメータで実験を行い，最小しきい値 $\sigma = 50 \sim 1000$（相対頻度 0.06 ～ 1.25%）を変化させて，計算時間（左縦軸）と頻出エピソード数（右縦軸）を示した．データサイズは 1000 行で，窓幅を $win = 6$ に固定した．

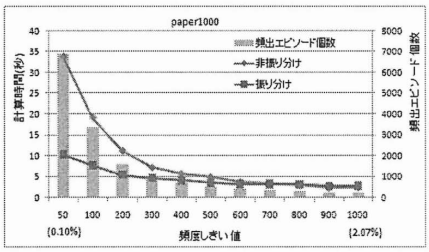

図 11: 最小しきい値に対する振り分け手法の計算時間の比較 (paper データ)

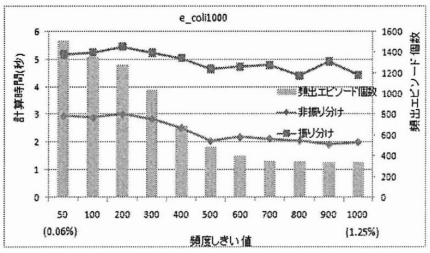

図 12: 最小しきい値に対する振り分け手法の計算時間の比較 (ecoli データ)

文字数が大きな paper データ上では，しきい値 σ が変化して，解の数が変化する場合でも，振り分け実装の DFS-MO-OD が安定して 2 倍程度高速だった．反対に，文字数が小さな ecoli データ上では，振り分け実装をしない DFS-MO の方が高速だった．これにより，振り分けアルゴリズムは文字の種類が多いデータに用いた方がよいことが分かる．

6.8 実験 6：飽和エピソード発見の効果

図 13 から図 15 に，paper データ上で，調節するパラメータを変化させて，アルゴリズム DFS-MO-OD と DFS-CLO に対して，計算時間（左縦軸）と枝刈りされ

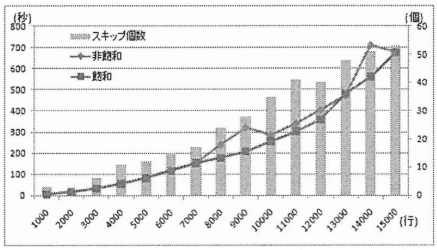

図 13: 入力サイズに対する計算時間の比較

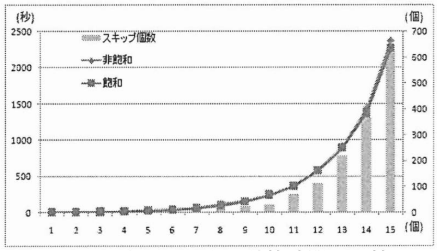

図 14: 窓幅に対する計算時間の比較

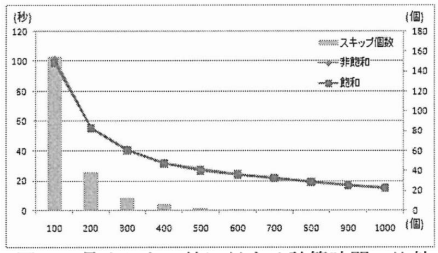

図 15: 最小しきい値に対する計算時間の比較

た非飽和エピソード数（右縦軸）を示した．図 13 では，窓幅を $win = 6$(文字) に，しきい値を $\sigma = 400$ に固定した上で，入力データサイズ $n = 1000 \sim 15000$(行)（相対頻度 $0.8 \sim 0.05\%$）を変化させた．図 14 では，窓幅を $win = 1 \sim 15$(文字) まで変化させた．データサイズは 3000 行に，最小しきい値を $\sigma = 400$（相対頻度 0.27%）に固定した．図 15 では，データサイズは 3000 行に，窓幅を $win = 6$(文字) に固定し，最小しきい値を $\sigma = 100 \sim 1000$(個)（相対頻度 $0.06 \sim 0.6\%$）まで変化させた．

図 13 から図 15 の実験結果では，枝刈りされる非飽和エピソードの総数は，多くて全頻出エピソード数の 1 割程度であり，それによって得られる高速化の効果はそれほど大きくなかった．特に，図 14 と図 15 では曲線が重なっている．しかし，オーバーヘッドはそれほど大きくなく，飽和アルゴリズムを用いても，ほぼ同等の時間でより詳細な頻出エピソード情報が得られることが分かった．

7 まとめ

本稿では，テキストデータから窓を用いて頻出な直列エピソードを発見するアルゴリズムを提案し，高速化技法を与えた．実験では，振り分け技法が小さな最小しきい値に有効であることを示し，飽和アルゴリズムが詳細な情報を持つ飽和エピソードのみを計算することを観察した．

公開プログラム LCM_seq として，本稿で考察した振り分け法と再帰的データベース縮約を含む，窓付きおよび窓なし直列エピソードに対する発見アルゴリズムが実装・公開されている [8]．今回は時間の都合で，[7] のアイテム集合発見のための高速化技法のうち，振り分け法と飽和パターン法のみについて調べた．今後の課題として，今回調べられなかった再帰的データベース縮約に関するより詳細な実験的評価や，音楽データ等の時系列データへの応用が挙げられる．

謝辞：本研究に関して，湊真一先生，Zeugmann 先生，伊藤公人先生，山本章博先生，平田耕一先生，情報知識ネットワーク研究室の学生諸君との討論と示唆に感謝いたします．

参考文献

[1] A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] H. Arimura, T. Uno, An Efficient Polynomial Space and Polynomial Delay Algorithm for Enumeration of Maximal Motifs in a Sequence, *J. Comb. Optim.*, Vol.13, 243-262, 2006.

[3] Hiroki Arimura and Takeaki Uno, Mining Maximal Flexible Patterns in a Sequence, Proc. LLLL2007, LNAI4914, Springer, 2008.

[4] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, M. Hsu, PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth, Proc. IEEE ICDE 2001, 215–224, 2001.

[5] Heikki Mannila, Hannu Toivonen, A. Inkeri Verkamo, Discovery of Frequent Episodes in Event Sequence, *Data Mining and Knowledge Discovery*, Vol. 1, 259–289, 1997.

[6] 内田雄三，窓ありエピソードと近似エピソードに対する高速な頻出系列パターン発見アルゴリズム，九州大学大学院システム情報科学研究科，修士論文，2004.

[7] 宇野毅明，有村博紀，データインテンシブコンピューティング その 2 — 頻出アイテム集合発見アルゴリズム —，人工知能学会誌，17 巻 2 号，2007.

[8] 宇野 毅明のホームページ，国立情報学研究所，`http://research.nii.ac.jp/ uno/index-j.html`

[9] T. Uno, T. Asai, Y. Uchida, H. Arimura, An efficient algorithm for enumerating closed patterns in transaction databases, In *DS'04*, LNAI 3245, 16-30, 2004.

[10] J. Wang, J. Han, BIDE: Efficient Mining of Frequent Closed Sequences, Proc. IEEE ICDE 2004, 79–90, 2004.