CF-003

重み付き起点順序分解に基づく飽和系列マイニング
Closed Sequence Mining based on Weighted Root-Orders Decomposition

酒匂　日菜乃[1)]　　山本　泰生[1)]
Hinano Sako　Yoshitaka Yamamoto

1　はじめに

本研究ではストリームデータに対する新たなスライディングウィンドウ型の飽和系列マイニングを提案する．観測対象から生成され続けるデータをストリームデータと呼ぶ．Web ログや健康データといった様々なストリームデータにおいて，特徴のある情報を抽出する研究が行われている [7, 6].

特に，ストリームデータから特徴的な系列を求める研究は**エピソードマイニング** (*episode mining*) と呼ばれ，これまでに多くの効率的手法が提案されている．例えば文献 [3] と [4] では，それぞれ極大系列と頻出 top-k 系列を対象とする問題が扱われている．他方，これらの手法はいずれも複数回のデータ走査が必要であり，逐次処理が必要となるストリームデータには適用し難い．

一方，系列ではなく各時刻のイベントの組み合わせを対象とする**アイテム集合マイニング** (*itemset mining*) の研究では，飽和アイテム集合族を逐次的に更新するオンラインアルゴリズムが提案されている [5, 8]．飽和性は，頻度情報を復元できる非可逆圧縮表現を得る際によく利用される概念である．アイテム集合に対しては**交点計算** (*incremental intersection*)[2] と呼ばれるシンプルな更新手法を用いて飽和表現を求めることができる．一方，系列の飽和表現を逐次的に求めるエピソードマイニングの手法は知られていない．

本研究では，系列の情報を**起点順序** (*root-order*) と呼ぶアイテムの集合として表現し，飽和アイテム集合マイニングの技法を用いて，飽和系列を抽出するオンラインアルゴリズムを提案する．起点順序分解により，系列はシンプルなイベント発生順序を保持した集合として解釈される．これにより，本来複雑な計算を要する系列の飽和表現を，アイテム集合マイニングのタスクとして抽出することが可能となる．本稿では，スライディングウィンドウを伴う飽和アイテム集合マイニング法を利用するが，従来の固定長のウィンドウでは，突発的に多くのイベントが発生するバーストに対して，処理性能が安定しない問題を持つ．そこで本研究では，計算リソースに応じてウィンドウサイズを伸縮させる**動的ウィンドウ** (*elastic window*) を提案し，メモリ消費と処理時間のロバストネスの改善する．

2　関連研究

ストリームデータから重要な特徴を抽出する研究分野は数多くある．その中で本研究に関わる 2 分野について概説する．

エピソードとはストリームデータ上に出現するイベント集合の列のことを指す．列の長さをエピソード長と呼ぶ．エピソードマイニングの主要なタスクはストリームデータ中に出現する頻出エピソードを抽出することである．ただし頻出エピソードを全て求めようとすると組み合わせ爆発が発生する．そこで頻出エピソードの中でもより特徴的なものを対象とする手法が開発されている．MaxFEM[3] は極大な頻出エピソードを抽出対象とする手法である．TKE[4] は出現回数が上位 K 件のエピソードを抽出対象とする手法である．メモリ消費の効率化と処理の高速化を実現しているが，いずれの手法も入力データを複数回読み込む必要がある．頻出エピソードを対象とするオンラインアルゴリズム [1] も存在するが，膨大な量の頻出系列全体を扱う必要があり運用は限定的である．

エピソードマイニングにおいて，エピソード長が 1 であるものはアイテム集合マイニングと同じタスクとなる．このタスクはデータマイニングの基礎問題として古くから研究されている．最近では，飽和アイテム集合と呼ばれる非可逆圧縮表現を逐次的に求めるオンラインアルゴリズムが提案されている．例えば，PARASOL[8] はランドマークウィンドウ型の飽和アイテム集合マイニングであり，計算リソースをもとに飽和集合族の近似解を効率的を抽出できる点が特徴として挙げられる．CICLAD[5] はスライディングウィンドウ型の飽和アイテム集合マイニングの一つで**生成子** (*generator*) を用いて厳密に飽和アイテム集合族を求めることができる．両手法とも入力データの走査は 1 度きりでよく，on-the-fly で到着するデータをオンライン処理しながら飽和アイテム集合族を取得できる．

3　背景

3.1　用語定義

初めにエピソードマイニングで用いる用語の定義を行う [3, 4].

定義 1　$I = \{a_1, a_2, \ldots, a_n\}$ をイベント[1)]の全体集合とする時，I の有限な部分集合を**イベント集合**と呼ぶ．時刻 i までに到達するイベント集合の列 $T_i = \langle ev_1, ev_2, \ldots, ev_i \rangle$ を時刻 i におけるストリームデータと呼ぶ．

定義 2　2 つのイベント集合の列 $\alpha = \langle s_1, s_2, \ldots s_n \rangle$, $\beta = \langle t_1, t_2, \ldots, t_m \rangle$ を考える．任意の $k\,(1 \leq k \leq n)$ に対し，i_k $(1 \leq i_1 < i_2 < \cdots < i_k \leq m)$ が存在し，$s_k \subseteq t_{i_k}$ を満たすとき，α を β の部分イベント集合列と呼び，$\alpha \sqsubseteq \beta$ と表す．β における t_{i_1} から t_{i_n} のイベント列を α のオカレンスと呼び，$[t_{i_1}, t_{i_n}]$ と表す．$i_n - i_1$ をオカレンス長と呼ぶ．また s_1, t_1 を先頭イベント集合，s_n, t_m を末尾イベント集合と呼ぶ．

定義 3　T_i の部分イベント集合列かつ先頭イベント集合が空ではないイベント集合の列を**エピソード**と呼ぶ．エピソード ep について，オカレンス長が閾値 s 以下であるようなオカレンスの集合を $OccSet(ep)$ と表す．以後，閾値 s を**スコープ**と呼ぶ．時刻 i における ep の先頭頻度を以下の式で表す．

1)　静岡大学大学院総合科学技術研究科

1)　アイテム集合マイニングの文脈ではアイテムに相当する．

$sup(ep, T_i) = |\{t_k \mid [t_k, t_l] \in OccSet(ep), l \leq i\}|$ また閾値 σ に対して，$sup(ep, T_i) \geq \sigma$ を満たすような ep を頻出エピソードと呼ぶ．

以後，曖昧性のない限り，イベントを a，イベント集合 $\{a_{k_1}, a_{k_2}, \cdots, a_{k_l}\}$ を $a_{k_1}a_{k_2}\cdots a_{k_l}$，エピソードを ep と表記する．

例 1 $T_5 = \langle ab, \phi, ac, b, cd\rangle$ というストリームデータが与えられたとする．この時エピソード $\langle b, c\rangle$ のオカレンスは $[1,3],[1,5],[4,5]$ であり，$s = 3$ の時，$OccSet(\langle b, c\rangle) = \{[1,3],[4,5]\}$ である．また，エピソード $\langle b, c\rangle$ の先頭頻度は 2 となる．

エピソード長が 1 である時のエピソードマイニングを特にアイテム集合マイニングと呼ぶ．すなわち，エピソード ep はアイテム集合に相当する．この時，以下のことがいえる．

定義 4 時刻 i において $\nexists a \in I\ s.t.\ sup(ep \cup \{a\}, T_i) = sup(ep, T_i)$ を満たすような ep を飽和アイテム集合と呼び，以降 CI と略す．また T_i 上の全ての飽和アイテム集合の族を $C(T_i)$ と表す．

定義 5 頻度が σ 以上の CI を頻出飽和アイテム集合 (以降 FCI と略す) と呼ぶ．

例 2 例 1 と同様のストリームデータが与えられたとする．この時，$C(T_5) = \{ab, ac, b, cd, a, c\}$ であり，閾値 $\sigma = 2$ とした時の FCI は a と c である．

定義 6 $C(T_i)$ に含まれる CI x, y について，$x \supset y$ かつ $sup(y, T_i) = sup(x, T_i) + 1$ を満たす時，x を y の**生成子**と呼ぶ．

命題 1 $C(T_i)$ 上の CI x, y について，$x \supseteq y$ ならば $sup(x, T_i) \leq sup(y, T_i)$ である．

命題 2 $C(T_i)$ 上の CI x, y について，$x \cap y$ もまた $C(T_i)$ に含まれる．

3.2 交点計算

本研究ではスライディングウィンドウ型の飽和アイテム集合マイニングを用いて飽和エピソードを抽出する．スライディングウィンドウ型とは，時間と共にウィンドウをスライドさせる方式のことである．スライディングウィンドウ型の頻出飽和アイテム集合マイニングのタスクはウィンドウ上の全ての頻出飽和アイテム集合を求めることである．

定義 7 時刻 m におけるウィンドウを W_m とする．ウィンドウサイズを w とする時，以下の式を満たす．

$$W_m = \begin{cases} (ev_1, \ldots, ev_m) & (m \leq w) \\ (ev_{m-k+1}, \ldots, ev_m) & (m > w) \end{cases}$$

定義 8 [2, 5] ある飽和アイテム集合の族に新たなイベント集合を追加されたとき，追加後の飽和アイテム集合の族を求めることを**増分交点計算**と呼ぶ．時刻 m において，飽和アイテム集合の族 T が与えられた時，T の時刻 m における増分交点計算の出力 $INC(T, m)$ は以下の式で表される．

$$INC(T, m) = T \cup \{\alpha \mid \beta \in T \cup \{I\}, \alpha = \beta \cap ev_m\}$$

例 3 $T_1 = \langle abd\rangle$ が与えられた時に $ev_2 = acd$ が追加される時を考える．この時 $C(T_1) = \{abd\}$ である．$INC(C(T_1), 2) = C(T_1) \cup \{\alpha \mid \beta \in C(T_1) \cup \{I\}, \alpha = \beta \cap ev_2\}$．ここで $abd \cap acd = ad, abcd \cap acd = acd$ であるので $INC(C(T_1), 2) = \{abd, ad, acd\}$ となる．

定義 9 [5] ある飽和アイテム集合の族からイベント集合を削除する時，削除後の飽和アイテム集合の族を求めることを**差分交点計算**と呼ぶ．時刻 m における飽和アイテム集合の族 T およびウィンドウサイズ k のウィンドウ W_m が与えられた時，T の差分交点計算の出力 $DEC(T, m)$ は以下の式で表される．

$$DEC(T, m) = T - \{\alpha \in T \mid \exists \beta \in T \cup \{I\},\ \alpha = \beta \cap ev_{m-w+1},\ sup(\beta, W_m) = sup(\alpha, W_m) - 1\}$$

例 4 $T_2 = \langle abd, acd\rangle$ が与えられた時に $ev_1 = abd$ が削除される時を考える．$C(T_2) = \{abd, acd, ad\}$ である．$sup(abd, W_2) = 1, sup(acd, W_2) = 1, sup(ad, W_2) = 1$ である．$DEC(C(T_2), 2) = C(T_2) - \{\alpha \in C(T_2) \mid \exists \beta \in C(T_2) \cup \{I\},\ \alpha = \beta \cap ev_1, sup(\beta, W_2) = sup(\alpha, W_2) - 1\}$ より，$C(T_2)$ から削除されるイベント集合を考える．$\alpha = abd$ の時，条件を満たすイベント集合 $\beta = I = abcd$ が存在する．$\alpha = acd$ の時，条件を満たすイベント集合 β は存在しない．$\alpha = ad$ の時，条件を満たすイベント集合 $\beta = acd$ が存在する．よって $DEC(C(T_2), 1) = \{acd\}$ となる．

定理 1 [5] ウィンドウサイズ w，時刻 $m + 1$ における飽和アイテム集合の族は以下のように書ける．

$$C(W_{m+1}) = \begin{cases} INC(C(W_m), m+1) & (m \leq w) \\ DEC(INC(W_m, m+1), m) & (m > w) \end{cases} \quad (1)$$

この漸化式を用いて飽和アイテム集合の族を求めることを**交点計算**と呼ぶ．

4 飽和エピソード抽出のための交点計算の課題

飽和アイテム集合マイニングでは，飽和アイテム集合の族の各要素に対して，追加/削除するイベント集合との共通部分を求めるだけで新しい飽和アイテム集合の族を求めることができる (式 (1))．命題 2 に示す通り，2 つのイベント集合の共通部分は一意に決まり，それが飽和アイテム集合となる．一方，これをエピソードに拡張しようとするとうまく当てはまらないことに気づく．例えば $\alpha = \langle a, b, c, d\rangle$ と $\beta = \langle a, c, b, d\rangle$ を考える．このとき α と β に含まれる極大なエピソード，すなわちアイテム集合の共通部分に相当するような系列を考えると，

$\langle a,b,d\rangle, \langle a,c,d\rangle$ の 2 つ存在しており，1 つに定まらないことがわかる．これは含意関係 $\sqsubseteq$ を持つ系列集合が束でないことに起因する．興味深いことに，束を形成する任意の離散構造データは式 (1) に基づく交点計算を適用できるが，その逆は成り立たない．よって，飽和アイテム集合マイニングにおける交点計算を飽和エピソード抽出に直接適用することは原理的に困難である．そこで本研究では，エピソードを構成する各イベントに対して先頭イベント集合からの距離情報を付加した順序情報に焦点を当てることで，エピソードをあるシンプルなイベント集合とみなす変換手法を提案する．これにより，飽和エピソードの抽出タスクを飽和アイテム集合マイニングの問題に帰着させることが可能となる．

5 提案手法

はじめに，飽和エピソードマイニングのタスクを次の特殊の含意関係をもとに再定義する．

定義 10 2 つのイベント集合の列 α と β が与えられた時，$\alpha \sqsubseteq \beta$ かつ α の先頭イベント集合 s_1 が β の先頭イベント集合 t_1 に含まれる時，α **は** β **に先頭含意される**と言い，$\alpha \preceq \beta$ と書く．

命題 3 s をスコープ，T_i を時刻 i におけるストリームデータ，W_m を時刻 $m\,(1 \le m \le i)$ におけるウィンドウ，ep をエピソードとする．ただし，W_m のウィンドウサイズは s とする．この時，以下の式が成り立つ:

$$sup(ep, T_i) = |\{W_m \mid ep \preceq W_m, 1 \le m \le i\}| \tag{2}$$

定義 11 時刻 i において次式を満たすような ep を**飽和エピソード**と呼ぶ:

$$\nexists ep'\ s.t.\ ep \prec ep', sup(ep, T_i) = sup(ep', T_i)$$

T_i 上の全飽和エピソードの集合を $CP(T_i)$ と表す．

提案手法の直感は，式 (2) にある．すなわち，各ウィンドウ W_m やエピソード ep がもし集合として表現されていれば，頻度の定義はアイテム集合マイニングのものと一致し，式 (1) の交点計算を用いて $CP(T_i)$ を漸近的に求めることができるようになる．

5.1 起点順序分解

起点順序分解 (*weighted root-orders decomposition*) と呼ぶ，任意のイベント集合の列を集合に変換する手法を述べる．

定義 12 イベント集合の列 $\alpha = \langle ev_1, ev_2, \cdots ev_n\rangle$ を考える．このとき α **に関する起点順序分解**とは以下の式で表される 3 つ組の集合 $RD(\alpha)$ を指すものとする．

$$RD(\alpha) = \{(a_p, a_q, d) \mid a_p \in ev_1, a_q \in ev_k, \\ d = k-1, 1 \le k \le n\}$$

例 5 イベント集合の列 $\alpha = \langle ab, c, d\rangle$ が与えられたとする．このとき α に関して起点順序分解すると $RD(\alpha) = \{(a,a,0),(a,b,0),(b,a,0),(b,b,0),(a,c,1),(b,c,1),(a,d,2),(b,d,2)\}$ となる．

ある 2 つのイベント集合の列と起点順序分解について，以下のことがいえる．

命題 4 2 つのイベント集合の列 α, β について，$RD(\alpha) \subseteq RD(\beta)$ ならば $\alpha \sqsubseteq \beta$ である．その逆は成り立たない．

定義 12 に沿ってイベント集合の列を起点順序分解すると先頭イベント集合及びイベント集合の列の大きさに従って起点順序分解後のイベント集合の大きさが増大する．そこで起点順序分解を再定義する．

定義 13 イベント集合の列 $\alpha = \langle ev_1, ev_2, \cdots ev_n\rangle$ を考える．このとき起点順序分解後のイベント集合 $RD'(\alpha)$ は以下の式で表される．

$RD'(\alpha) = \{(a : d) \mid a \in ev_k, 1 \le k \le n, d = k-1\}$

例 6 例 5 と同様のイベント集合の列が与えられたとする．このとき α に関して定義 13 をもとに起点順序分解すると $RD'(\alpha) = \{(a:0),(b:0),(c:1),(d:2)\}$ となる．

付録 B で示した命題 5 及び命題 4 より再定義した起点順序分解についても命題 4 がいえる．

本研究では，先頭含意の代わりに起点順序集合の包含関係を用いて，飽和エピソードを抽出する．

定義 14 s をスコープ，T_i を時刻 i におけるストリームデータ，W_m を時刻 $m\,(1 \le m \le i)$ におけるウィンドウ，ep をエピソードとする．ただし，W_m のウィンドウサイズは w とする．この時，**起点順序に基づく** ep **の先頭頻度** $sup_o(ep, T_i)$ を次式のように定める．

$$sup_o(ep, T_i) = |\{W_m \mid RD'(ep) \subseteq RD'(W_m), 1 \le m \le i\}| \tag{3}$$

また時刻 i において次式を満たすような ep を**起点順序に基づく飽和エピソード**と呼ぶ:

$$\nexists ep'\ s.t\ RD'(ep) \subset RD'(ep'), sup_o(ep, T_i) = sup_o(ep', T_i)$$

T_i 上の起点順序に基づく全飽和エピソードの集合を $CP_o(T_i)$ と表す．

従来の飽和エピソードマイニングでは定義 11 の $CP(T_i)$ を抽出対象としてきた．一方，本研究が対象とするのは定義 14 の $CP_o(T_i)$ である．$CP_o(T_i)$ は交点計算に基づく飽和アイテム集合マイニングにより求めることができる．

図 1 は，入力のストリームデータ T_i からどのような手順で起点順序分解を行うかを示している． 本研究では，スライディングウィンドウ型の飽和アイテム集合マイニング法を用いる．はじめにスコープサイズ毎にスライドさせながらイベント集合の系列を抽出する．次に抽出した系列を起点順序集合に分解し，スライディングウィンドウに登録する．最後に式 (1) の漸化式に基づき交点計算を行う．

5.2 アルゴリズムの概要

以上の手法をもとに，本研究で利用するアルゴリズムを示す．

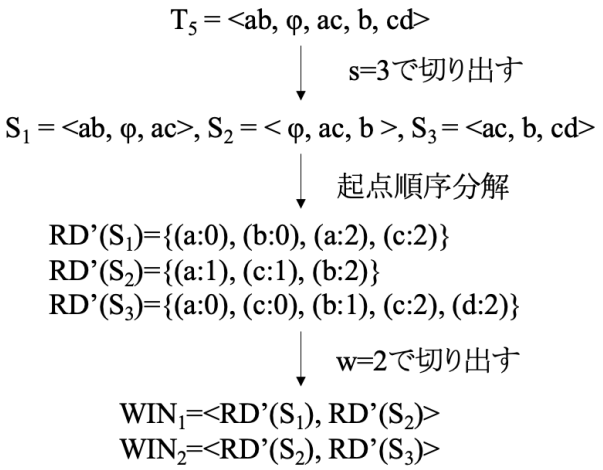

図 1　起点順序分解の流れ

Algorithm 1 提案手法

Input: $T_i = \langle ev_1, \ldots, ev_i \rangle$: ストリームデータ,
w : ウィンドウサイズ,
s : スコープサイズ,
k : 最大飽和エピソード数
Output: *CPTable*: 飽和エピソーとその先頭頻度が保存されたテーブル

1: **for** $m = 1, 2, \ldots i$ **do**
2: 時刻 m の系列 (スコープサイズ s) S_m を抽出する
3: $WIN_{m+1} := WIN_m \cup \{RS'(S_m)\}$ (ウィンドウ更新)
4: $CPTable := C(WIN_{m+1})$ (式 (1) に従う)
5: $WIN_{m+1} := WIN_{m+1} - \{RS'(S_{m-w+1})\}$
6: **end for**

1 行目でストリームデータからスコープサイズに応じて系列を抜き出す．2 行目で系列を起点順序分解し，ウィンドウを更新する．その後ウィンドウに対して交点計算を行う．

なお，本研究では CICLAD[5] アルゴリズムを一部修正することで提案手法での先頭イベントの一致条件を再現する．本誌の関係上，修正箇所のアルゴリズムのみを記載する．

Algorithm 2 ExpandPath*[5]

1: $n \leftarrow loop\_succ(c.last, item)$
2: **if** a が先頭イベント *or* $c.last \neq rootnood$ **then**
3: **if** $n = null$ **then**
4: $n \leftarrow create\_succ(a)$
5: **end if**
6: $n \leftarrow create\_succ(item)$
7: $c.last.cpt - -$;
8: $n.cpt + +$;
9: $c.last \leftarrow n$
10: $UpdateGen(entry)$
11: **end if**

5.3　動的ウィンドウの概要

スライディングウィンドウ型の飽和アイテム集合マイニングでは，ウィンドウサイズを固定した上で飽和アイテム集合 (CI) を更新することが一般的である．しかしイベント集合の大きさが大きくなると CI 数および各タイムスタンプあたりにかかる交点計算の時間が増加する．そこで本研究では，安定した計算時間と効率的なメモリ利用を実現するために動的ウィンドウを導入する．図 2 に概要を示す．

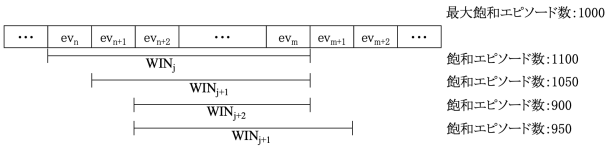

図 2　動的ウィンドウの概要

保存する飽和エピソード数の上限 (最大飽和エピソード数) を決め，上限を越した際は差分交点計算のみを実施し，ウィンドウサイズを小さくする．下回った場合はユーザが設定したのウィンドウの長さに戻るまでは増分交点計算をし，戻れば定理 1 に基づいて飽和エピソードマイニングを行う．

6　実験

6.1　使用するデータセット

実験に使うデータセットについて説明する．[8] および [9] から 2 つのデータセットを利用する．各データセットの詳細を表 1 に示す．なお表中の event はイベントの種類数を，stream はストリームデータの長さを，avg はイベント集合長の平均を，feature は各データの特徴を表す．

dataset	event	stream	avg	feature
eq	1228	16764	2.5	イベント集合長が突発的に増大
hadoop	134	199000	1	シングルストリーム

表 1　データセットの詳細

6.2　実験 1：起点順序分解の性能評価

ウィンドウサイズおよびスコープサイズの違いにより実行時間および抽出される飽和エピソード数がどのように変わるか検証した．なお Algorithm1 において，最大飽和エピソード数 k を十分に大きな数字を設定し，ウィンドウサイズが変わらないようにした．結果を以下に示す．

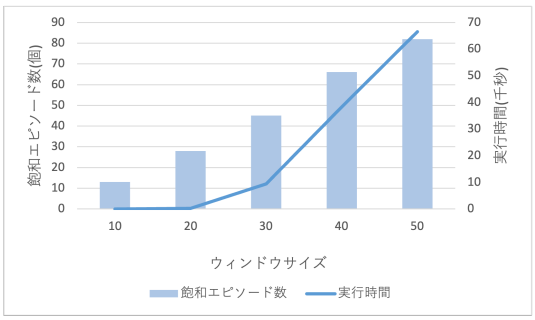

図 3　ウィンドウサイズを変化させたときの実行結果 (eq データ，スコープサイズ 2 固定)

結果より，ウィンドウサイズ及びスコープサイズが大きくなるに従って実行時間および飽和エピソード数が増加

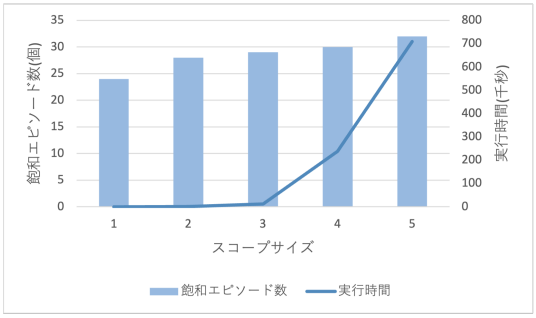

図 4　スコープサイズを変化させたときの実行結果 (eq データ，ウィンドウサイズ 20 固定)

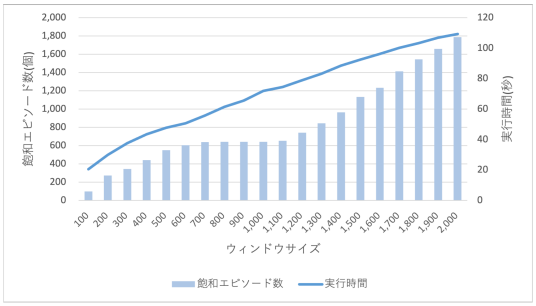

図 5　ウィンドウサイズを変化させたときの実行結果 (hadoop データ，スコープサイズ 5 固定)

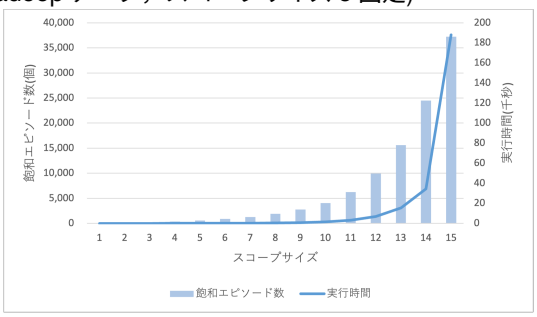

図 6　スコープサイズを変化させたときの実行結果 (hadoop データ，ウィンドウサイズ 500 固定)

することがわかる．また両データともスコープサイズの変化が実行時間に大きく影響することがわかる．

6.3　実験 2：動的ウィンドウの性能評価

実験 1 より，ウィンドウサイズに応じて飽和エピソード数及び実行時間が増加することがわかった．そこで eq データに対して，動的ウィンドウの有無が実行時間にどのような影響を与えるか検証した．実験では最大エピソード数 k を 20 万とし実験を行った．結果を図 7 に示す．w はウィンドウサイズ，s はスコープサイズを表している．なお，$w = 50, s = 3$ は 2 週間ほど実行しているが計算が完了しなかったため記載していない．

実験結果より，条件 $w = 50, s = 2$ では実行時間が 3 分の 1 以下に減少していることがわかる．また条件 $w = 50, s = 3$ では実行時間が少なくとも 10 分の 1 以下に減少していることがわかった．このことから動的ウィンドウが実行時間に大きく影響を与えることがわかった．

7　まとめ

本研究では起点順序を導入することで飽和アイテム集合マイニングでイベント集合の列を扱えるように拡張した．そして動的ウィンドウを導入することで実行時間の

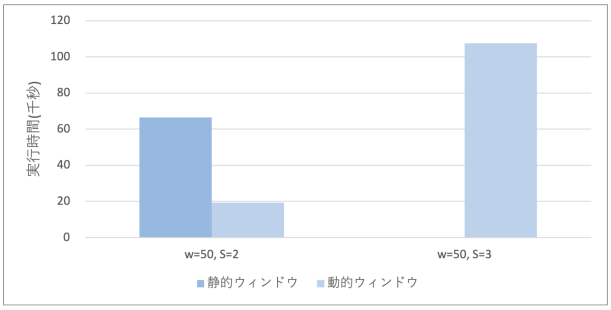

図 7　動的ウィンドウによる実行時間の違い

短縮を実現した．今後の課題として提案手法で抽出された飽和エピソードの妥当性の評価が挙げられる．既存のエピソードマイニングの手法で抽出されるエピソードを提案手法がどれだけカバーできているのかを検証したい．またプログラムの高速化が挙げられる．6.3 節で示したように，eq データにおいてウィンドウサイズ 50，スコープサイズ 3，動的ウィンドウ未使用の条件下では 2 週間ほどかかっている．原因としては CICLAD アルゴリズムで使用する転置リストの探索効率が悪いことが挙げられる．実際，転置リストの探索の時間は総実行時間の 3 割ほどあることがわかっている．探索効率を向上することでより大きなウィンドウサイズ・スコープサイズを扱えるようにしたいと考えている．そして最終的には提案手法により抽出した飽和エピソードを用いて系列予測を行なっていきたい．

参考文献

[1] X. Ao, P. Luo, C. Li, F. Zhuang and Q. He: Online frequent episode mining, *ICDM'15*, 2015
[2] C. Borgelt, X. Yang, R. N. Cadenas, P.C. Saez and A. P. Montano: Finding closed itemsets by intersecting transactions, *EDBT'11*, 2011
[3] P. Fournier-Viger, M. S. Nawaz, Y. He, Y. Wu, F. Nouioua and U. Yun MaxFEM: Mining maximal frequent episodes in complex event sequences, *MIWAI 2022*, 2022
[4] P. Fournier-Viger, Y. Yang, P. Yang, J. C.-W. Lin and U. Yun: TKE: Mining top-K frequent episodes, *IEA/AIE 2020*, 2020
[5] T. Martin, G. Francoeur and P. Valtchev: CICLAD: A fast and memory-efficient closed itemset miner for streams, *KDD'20*, 2020
[6] R. Rawassizadeh, E. Momeni, C. Dobbins and J. Gharibshah: Scalable daily human behavioral pattern mining from multivariate temporal data, *IEEE Transactions on Knowledge and Data Engineering*, 2016
[7] M.-Y. Su: Applying episode mining and pruning to identify malicious online attacks, *Computers & Electrical Engineering 59*, 2017
[8] Y. Yamamoto, Y. Tabei and K. Iwanuma: PARASOL: a hybrid approximation approach for scalable frequent itemset mining in streaming data, *Journal of Intelligent Information Systems*, 2020
[9] Yahoo! Hadoop grid logs (from `https://webscope.sandbox.yahoo.com/`)
[10] 安井 壱陽, 新谷 隆彦, 大森 匡, 藤田 秀之: 行動時間帯に偏りのある長時間エピソード抽出における探索候補枝刈り手法, *DEIM2022*, 2022

A　命題 4 の証明

証明　$\alpha = \langle A_1, A_2, \dots A_n \rangle, \beta = \langle B_1, B_2, \dots B_m \rangle$ とする．このとき仮定より任意の $(a_p, a_q, d) \in RD(\alpha)$ について $(a_p, a_q, d) \in RD(\beta)$ である．ここで定義 12 より，任意の $a_q \in A_k (1 \leq k \leq n)$ について $a_q \in B_k$ となる．よって任意の $A_k \in \alpha (1 \leq k \leq n)$ について $A_k \subseteq B_k$ である．これは定義 2 を満たす．よって命題は示された．　□

なお，命題 4 の逆は正しくない．反例として $\alpha = \langle a, b \rangle, \beta = \langle a, c, b \rangle$ を考える．このとき定義 2 より $\alpha \sqsubseteq \beta$ である．一方で $RD(\alpha) = \{(a, a, 0), (a, b, 1)\}$, $RD(\beta) = \{(a, a, 0), (a, c, 1), (a, b, 2)\}$ であるので，$RD(\alpha) \nsubseteq RD(\beta)$ となる．

B　新旧起点順序分解の可逆性

以下の命題を示す．

命題 5　2 つのイベント集合の列 α, β について，$RD(\alpha) \subseteq RD(\beta) \Leftrightarrow RD'(\alpha) \subseteq RD'(\beta)$ である．

証明　$\alpha = \langle A_1, A_2, \dots A_n \rangle, \beta = \langle B_1, B_2, \dots B_m \rangle$ とする．まず右辺が左辺の必要条件であることを証明する．$RD(\alpha) \subseteq RD(\beta)$ より任意の $(a_p, a_q, d) \in RD(\alpha)$ について $(a_p, a_q, d) \in RD(\beta)$ を満たす．すなわち任意の $a_q \in A_k (1 \leq k \leq n)$ について $a_q \in B_k$ を満たす．

ここで命題 5 の右辺について考えると，任意の $(a : d) \in A_g (1 \leq g \leq n)$ について $(a : d) \in B_g$ がいえる．よって $RD'(\alpha) \subseteq RD'(\beta)$ である．以上から右辺が左辺の必要条件であることを証明された．

次に左辺が右辺の必要条件であることを証明する．$RD'(\alpha) \subseteq RD'(\beta)$ より任意の $(a : d) \in RD'(\alpha)$ について $(a : d) \in RD'(\beta)$ を満たす．すなわち任意の $a \in A_g (1 \leq g \leq n)$ について $a \in B_g$ を満たす．

ここで命題 5 の左辺について考えると，任意の $a_p \in A_1$ について $a_p \in B_1$ であり，任意の $a_q \in A_k (1 \leq k \leq n)$ について $a_q \in B_k$ がいえる．よって $RD(\alpha) \subseteq RD(\beta)$ である．以上から左辺が右辺の必要条件であることを証明された

以上から命題 5 は示された．　□

C　飽和エピソードマイニングにおける先頭イベント集合の共通イベントの必要性

初めに以下の命題 6 を示す．

命題 6　先頭イベント集合が空ではないイベント集合の列 $\alpha = \langle A_1, A_2, \dots A_n \rangle$ について，$A_1 \neq \phi$ であるとき，$RD(\alpha)$ と $RD'(\alpha)$ は可逆である．

証明　まず $RD(\alpha)$ から $RD'(\alpha)$ を求めることを考える．$A_1 \neq \phi$ および定義 12 より，$a_p \in A_1, a_q \in A_k (a \leq k \leq n), d = k - 1$ を満たす要素が存在する．これを ele とする．ここで定義 13 より，ele は $(a_q : d)$ と同値であることがわかる．よって $RD(\alpha)$ を $RD'(\alpha)$ に変換することは可能である．

次に $RD'(\alpha)$ から $RD(\alpha)$ を求めることを考える．定義 13 より，$a'_q \in A_k (1 \leq k \leq n), d = k - 1$ を満たす要素が存在する．これを ele' とする．ここで $A_1 \neq \phi$ より，$a'_q \in A_1$ となるような $RD'(\alpha)$ の要素は $d = 0$ のものである．これを ele'_h とする．すると ele'_h および ele' を用いることで (a_p, a_q, d) を求めることが可能である．よって $RD'(\alpha)$ を $RD(\alpha)$ に変換することは可能である．

以上から命題 6 は示された．　□

次に以下の命題 7 を示す．

命題 7　先頭イベント集合が空ではない 2 つのイベント集合の列 α, β について，$RD(\alpha) \cap RD(\beta) \Leftrightarrow RD'(\alpha) \cap RD'(\beta), \exists a : 0 \in RD'(\alpha) \cap RD'(\beta)$

証明　$A_k \cap B_k = C_k$ とする．まず右辺が左辺の必要条件であることを証明する．定義より $RD(\alpha) \cap RD(\beta) = \{(a_p, a_q, d) \mid a_p \in C_1, a_q \in C_k (1 \leq k \leq min(n, m)), d = k - 1\}$, $RD'(\alpha) \cap RD'(\beta) = \{(a : d) \mid a \in C_g (1 \leq g \leq min(n, m)), d = g - 1\}$ である．ここで $\exists (a : 0) \in RD'(\alpha) \cap RD'(\beta)$ について考える．定義 13 より，$\exists a$ について $a \in A_1$ かつ $a \in B_1$，つまり $a \in C_1$ である．このことから命題 7 と命題 6 は同値である．

以上から命題 7 は示された．　□