隣接グラフを用いた系列データからの高速な頻出パターン抽出方法

川端　貴幸† 　伊藤　史朗† 　横田　治夫††

† キヤノン（株）ソフトウェア応用技術開発センター　〒 146-8501 東京都大田区下丸子 3-30-2
†† 東京工業大学大学院情報理工学研究科計算工学専攻　〒 152-8552 東京都目黒区大岡山 2-12-1
E-mail: †{kawabata.takayuki,ito.fumiaki}@canon.co.jp, ††yokota@cs.titech.ac.jp

あらまし　系列データからの頻出パターン抽出は，データマイニングの領域で重要な課題である．その代表的な方法であるエピソードマイニングは，イベントの系列データから，頻出するイベントの半順序集合パターン（エピソードと呼ぶ）を発見する問題である．既存手法は，計算時間が大きい，出力されるパターンが膨大という問題点がある．本論文では，従来手法よりも適切なパターンに絞って，高速に抽出する手法を提案する．提案手法は，連続して現れやすいイベントを繋ぐことで重み付き隣接グラフを作成し，その隣接グラフに対してノイズのカットやパターンの分離などの操作を行うことでパターンを抽出する．シミュレーションデータとファイル操作履歴の実データを用いて評価実験を行い，提案手法は既存手法より適切なパターンを高速に抽出できることを確認した．

キーワード　エピソードマイニング，高速化，データマイニング

1. はじめに

昨今，ビッグデータというキーワードに象徴されるように，大規模なデータから有用な知識や情報をパターンとして抽出するデータマイニングの研究が盛んに行われている．特に，系列データは最も基本的な大規模データの一つであり，系列データマイニングの研究が近年注目を集めている．実際に，系列データマイニングは多くの応用分野で利用されており，例えば，工場の障害ログからの原因診断 [1]，Web 操作履歴からユーザの振る舞い予測 [2]，地震データ分析 [3]，ファイル操作履歴からのワークフロー抽出 [4] などが挙げられる．

一般的に，系列データマイニングは問題定義の違いから二つに大別される．一つは，Agrawal ら [5] によって提案されたシーケンシャルパターンマイニング（Sequential Pattern Mining : 以下，SPM と呼ぶ）である．SPM とは，図 1 の (a) に示すように，系列データ集合を入力とし，複数の系列に現れる頻出部分系列をパターンとして出力する問題である．出力するパターンはシーケンシャルパターンと呼ばれ，順序が一意に固定されたものである．

もう一つは，Mannila ら [6] によって提案されたエピソードマイニングである．エピソードマイニングとは，図 1 の (b) に示すように，単一の長い系列データを入力とし，その中に頻出して現れる半順序集合をパターンとして出力する問題である．出力するパターンはエピソードと呼ばれ，順序が一意に固定されず，半順序を許容した汎用性の高いものとなる．また，エピソードマイニングでは，この A や B などのデータをイベントと呼ぶ．エピソードマイニングの詳細な定義については後述する．図 2 に示すように，エピソードは 3 種類に分けて考えられる．(a) はシリアルエピソードであり，イベント間の順序は一意に固定されたものである．(b) はパラレルエピソードであり，イベント間の順序が全てないものである．(c) はジェネラルエピソードで，イベント間に半順序な関係が存在するものである．シリアルエピソードやパラレルエピソードは，ジェネラルエピソードの特別なケースと考えられる．また，シリアルエピソードはシーケンシャルパターンと等価である．

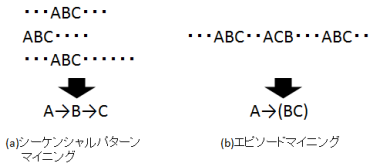


図 1　系列データマイニングの例

我々は，SPM はエピソードマイニングのサブセット問題であると考える．なぜなら、SPM の入力形式である系列データ集合は，単一の系列データを分節化したものと考えられ，シーケンシャルパターンはエピソードに包含されるパターンだからである．つまり，SPM の適用問題は，エピソードマイニングの手法で解くことが可能である．

そこで，本論文では系列データマイニングとしてより汎用的なエピソードマイニングにおける新手法を提案する．提案手法は，既存の手法と比べて高速であり，かつ，抽出したエピソードの精度が高いことが特徴である．エピソードを高速に抽出するために，提案手法では，連続して現れやすいイベントを繋ぐことで重み付き隣接グラフを作成し，その隣接グラフに対してノイズのカットやパターンの分離などの操作を行うことでエピソードを抽出する．本論文では，シミュレーションデータとファイル操作履歴などの実データを用いて評価実験を行い，提案手法が従来手法より適切なエピソードを高速に抽出できることを確認した．本論文の構成は以下の通りである．2 章で関連研究について述べる．3 章で提案手法について説明し，4 章と 5 章で評価実験について述べ，6 章でまとめる．

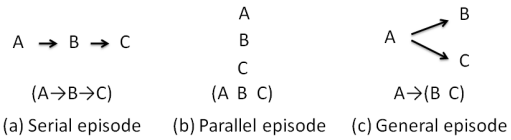


図 2　エピソードの例

2. 関 連 研 究

2.1　イベント系列の中のエピソード

ここでは，Mannila ら [6] によって提案されたエピソードマイニングの枠組みを簡単に説明する．エピソードマイニングでは，入力となるデータ D はイベント系列と呼ばれ，$D =< (E_1, t_1), (E_2, t_2), \ldots (E_n, t_n) >$ のように表される．E_i はイベントのタイプを表し，t_i はイベントの発生時刻を表す．イベント系列の中で，全ての i において $t_i < t_{i+1}$ となる．例えば，以下は 8 つのイベントを含むイベント系列である．

$$< (A,1), (B,3), (D,4), (C,6), (E,12), (A,14), (C,15), (B,17) >$$

エピソードは，イベントの半順序集合と定義され，例えば，図 2 のように表される．図 2 の (a) は 3 ノードのシリアルエピソードと呼ばれる．エピソードマイニングはイベント系列から頻出エピソードを発見する問題である．

2.2　ジェネラルエピソードの抽出

これまでに提案されたエピソードマイニングの手法は，そのほとんどが抽出対象をシリアルエピソード，または，パラレルエピソードに限定した手法 [6] [7] [8] [9] [10] である．近年ようやく，制約はあるものの，ジェネラルエピソードを抽出する手法 [11] [12] が提案された．これらの手法は，Apriori アルゴリズム [5] をベースとしており，最少ノードのエピソードから始まり段階的にノードを増やしながら頻出エピソードを発見する幅優先探索である．そのアルゴリズムの概略を Algorithm 1 で示す．Apriori アルゴリズムを単純に適用した場合の大きな問題点は，計算量の多さと，解である抽出した頻出エピソードの多さである．これらの問題点は，組合せ爆発によって起こる．例えば，イベント系列の中に 9 ノードのシリアルエピソードが含まれる場合，抽出されるエピソードは組合せで得られるサブエピソードを含んで少なくとも一億を超える．9 ノードのシリアルエピソードを 1 つ含むだけでこれだけの組合せ爆発が起こるが，実際には，複数のエピソードが含まれるのが普通であり，さらには，ジェネラルエピソードの組合せ爆発は，シリアルエピソードの比ではない．そこで，早い段階で候補を削減する枝刈りの方法が提案されている．Tatti ら [11] は，パターンマイニングにおいて一般的である飽和パターンをジェネラルエピソードマイニングに適用した方法を提案している．飽和パターンとは，同一の頻度の親パターンが存在しないパターンである．また，Laxman ら [12] は，Bidirectional evidence という指標を与え，有益でないエピソードを除去する方法を提案している．Bidirectional evidence は，直観的には，エピソードが持つ順不同なイベントペアが，観測されたイベント系列で実際には順序がありそうなときに低い値となる．例えば，A → (B C) というエピソードがあった時に，イベント系列中で B,C が 8 回，C,B が 2 回観測された場合，実は B → C という順序関係が強いと考えられるため，A → (B C) というエピソードは頻出から除かれる．

Algorithm 1 幅優先探索の概略

Input: 系列イベント s, 最少出現回数 σ, 最大エピソード長 ρ
Output: 頻出エピソード集合

```
N ⇐ 1;
ε ⇐ 1 ノードの頻出エピソード集合
C ⇐ ε から N + 1 ノードの候補エピソード集合を生成する
N ⇐ N + 1
while C ≠ 0 do
    P ⇐ s をスキャンして C から頻出エピソードを抽出する
    add P to ε
    C ⇐ ε から N + 1 ノードの候補エピソード集合を生成する
    N ⇐ N + 1
end while
return ε
```

既存手法は，有益でないと考えられるエピソードを早い段階で枝刈りすることで，高速化と解の数を減らすことに成功しているが，我々はまだ現実問題に適用するには不十分であると考えている．まず第一に，データ規模に対して計算コストが指数関数的に増大するため，大規模データに対してはやはり現実時間での計算が難しい．我々の実験では，10 万を超える系列データに対して 10 時間以上の計算時間を要した．現実時間で解を求める為には，最少出現回数やエピソードの最大長などのパラメータを厳しい方向に設定する必要があるが，これは，低頻度のエピソードや長いエピソードの抽出を困難にする．そして第二の問題として，この最少出現回数やエピソードの最大長などのパラメータを決める難しさがある．有益なエピソードを漏れなく (網羅性)，かつ，妥当な個数（正確性）抽出することが本来望ましいが，これらはパラメータに対してトレードオフの関係にある．つまり，最少出現回数を小さく設定すれば，漏れは少なくなるが，ゴミが爆発的に増え，有益なエピソードが埋もれてしまう．そこで，妥協点を探すことになるが，あくまでそれは妥協点であり，本来望んでいた結果とは異なる．本論文では，これらの問題を解決し，有益なエピソードを漏れとノイズを少なくして高速に抽出する手法を提案する．

3. 提 案 手 法

3.1　基本的アイデア

我々は，エピソードは連続して現れやすいイベントの集合であるという仮定のもとに本手法を提案する．この仮定は、エピソードに則って生成されるイベントは、部分的には連続していることが多いことを意味する．具体例で説明すると、系列イベント中にエピソード A → B → C が複数回出現するとき，1.A,B,C / 2.A,B,*,C / 3.A,*,B,C / 4.A,*,B,*,C(* は 1 つ以上のノイズイベント）の 4 つの出現パターンが考えられるが、部分的には連続しているというのは 1 〜3 を指し、これらを合わせた出現回数はそれなりに多いはずである．そこで，局所的な隣接関係（部分的な連続）を観察することで全体を復元するアプローチが有効と考える．このような仮定は世の中の多くのデータに当てはまると考えられるが，特に，人が関わって生成されるデータに対して良く当てはまると考えている．例えば，オフィスでは，ある作業（エピソード）に取り組んでいるとき

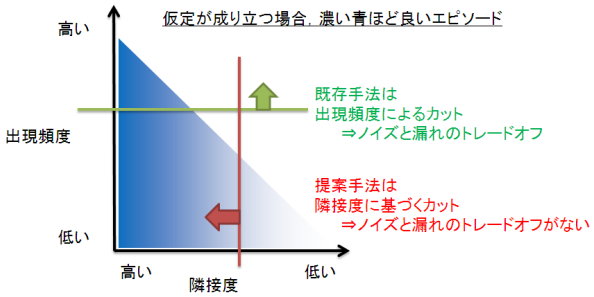


図 3　既存手法との違い

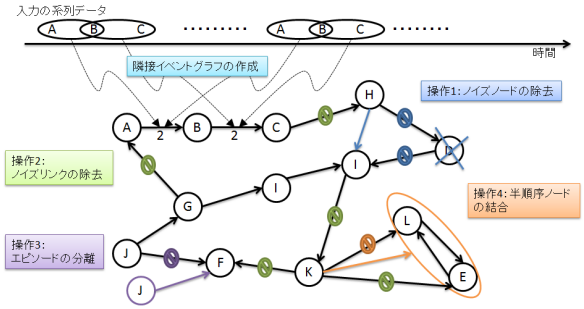


図 4　提案手法の概要

は，その作業に関するファイル操作（イベント）が連続して行われやすい．このような仮定に基づき，連続して現れやすいイベントを中心に，ノイズを除去しながらグラフとして繋いでいくことでエピソードを抽出する方法を提案する．

3.2　既存手法との違い

提案手法と既存手法の直感的な違いについて図 3 を用いて説明する．図 3 は系列データに含まれるエピソードの分布を表しており，縦軸はエピソードの出現頻度，横軸はエピソードの隣接度を表す。エピソードの隣接度とは，先ほどの具体例で説明すると全て隣接している”1.A,B,C”ばかりが出現するエピソードは高いとなり，その逆に全て隣接していない”4.A,*,B*,C”ばかりが出現するエピソードは低いこととする。エピソード内のイベントは連続して現れやすいという仮定が成り立つ場合、濃い青ほど良いエピソードであると言える。そのとき、既存手法は最少出現頻度によるカットを行うために，必ずノイズと漏れのトレードオフに陥る。それに対して，提案手法では直観的には隣接度に基づくカットを行っているので、ノイズと漏れのトレードオフを回避できる。また、提案手法は隣接度が低いならエピソードから漏れる可能性を許容することで高速化されている．

3.3　提案手法の概要

提案手法の概要を図 4 を用いて説明する．まず，入力の系列データから隣接イベントグラフを作成する．隣接イベントグラフは，イベントをノード，隣接するイベント間に，出現回数に応じた重みの有向リンクを張ったグラフである．次に，作成した隣接イベントグラフに対して図 4 に示す 4 つの操作を行い，エピソードを抽出する．

それぞれの操作の目的について簡単に説明する．操作 1 のノイズノードの除去は，有益なエピソードの一部と成りえなさそうなイベントを除去することで，有益なエピソード内のイベント間の結びつきを強化する．操作 2 のノイズリンクの除去は，イベント間の結びつきが弱い（連続して現れにくい）リンクを除去することで，異なるエピソードに分割する．操作 3 のエピソードの分離は，複数のエピソードに含まれるイベントを推測して分離することでエピソードを分離する．操作 4 の半順序ノードの結合は，エピソードに含まれる半順序ノード群の入出力リンクを補正する．

3.4　提案手法の詳細

ここではまず隣接イベントグラフの作成方法について述べ，そこから有益なエピソードを抽出するために行う各操作について目的と方法を具体的に説明する．

3.4.1　隣接イベントグラフの作成

隣接イベントグラフは重み付き有向グラフ $G=(V,L,g)$ として表す．ここで，$V=\{v_1,v_2,...,v_N\}$ はノード集合を表し，$E \subset V \times V$ はノード集合 V 上の重み付き有向リンク集合を表し，$g: V \rightarrow \Sigma$ はノードからイベントタイプへの写像である．Σ はイベントタイプの集合である．入力の系列イベント $D=<(E_1,t_1),(E_2,t_2),...(E_n,t_n)>$ が与えられたとき，隣接イベントグラフ G は次のように作成する．なお，リンクの重みをリンク強度とも呼ぶ．

隣接イベントグラフの作成は，系列イベントの連続する 2 つのイベント $(E_i,t_i),(E_{i+1},t_{i+1})$ の全ペアに対して以下の処理を行う．

（1）　リンク強度 w を以下の式で計算する．

$$w=2(1-(\frac{1}{1+e^{-a((t_{i+1}-t_i)/\delta}})) \qquad (1)$$

この式は，0〜1 の値を取り，2 つのイベントの発生時刻が近いほど大きな値となる．また，δ は変曲点を表すパラメータであり，a はゲインと呼ばれ変化の大きさを制御する．δ と a を任意に設定することができるが，δ は系列イベントの連続する 2 つの全イベント間の発生時刻の差の平均とし，a は 1 とするのが妥当である．図 5 に δ を 100，a を 1 とした式 1 のグラフをのせる．

（2）　$g(v_s)=E_i$，$g(v_t)=E_{i+1}$ なるノード v_s，v_t 間に重み w の有向リンクを張る．もし，すでに有向リンクが存在する場合は重み w を既存の重みに加える．

以上の処理により，系列イベントから隣接イベントグラフを作成する．

3.4.2　ノイズノードの除去

ノイズノードの除去を行うことで，エピソード内の本来のイベント（ノード）間の結びつきを強化する．ノイズノードとは有益なエピソードの一部と成りえないノードである．それは，特定のノードとの強い結びつきを持たないノードであると考える．図 6 の例を用いて説明する．なお，簡単のためリンク強度は式 1 によらず 1.0 とした．イベント X は 3 回出現したうち，3 回とも前後に出現したイベントが異なるので，特定のイベントとの強い結びつきがないためノイズと考えられる．逆に，イベント Y は 3 回出現したうち，後に出現したイベントが 3 回とも同じイベントであるため有益なエピソードの一部と考えられる．このような考えに基づきノード v_s のノイズスコア $H(v_s)$ を以下の式で定義する．

$$H(v_s) = Min(-\frac{\sum_{j \in Inlink_s} p_j log_2 p_j}{log_2 \frac{1}{N_s}}, -\frac{\sum_{j \in Outlink_s} q_j log_2 q_j}{log_2 \frac{1}{N_s}}) \quad (2)$$

ここで，N_s はノード v_s に写像されるイベントの出現回数を表し，$Inlink_s$，$Outlink_s$ はそれぞれノード v_s に接続する Inlink の集合，Outlink の集合を表す．また，p_j は $link_j$ のリンク強度を $Inlink_s$ に含まれる全 $link$ のリンク強度の和で割ったものであり，q_j は $link_j$ のリンク強度を $Outlink_s$ に含まれる全 $link$ のリンク強度で割ったものである．このノイズスコアは 0〜1 の値をとり，値が高いほどノイズと考えられる．また，Inlink，Outlink ごとにスコアを計算しその小さい方をノイズスコアとしている．これは，エピソードの両端に当たるノードのノイズスコアが高くなるのを防ぐためである．図 6 の例では，それぞれのノイズスコアはノード X は 1 となり，ノード Y は 0 となる．

次に，このノイズスコアを用いたノイズノードの除去方法について説明する．まず，隣接イベントグラフの全ノードについてノイズスコアを計算し，あらかじめ設定した閾値 (H_{th}) を超えたノードをノイズノードとして記録しておく．ノイズノードが 1 つ以上発見された場合には，再度，系列イベントを読み込み隣接イベントグラフを作成し直す．その際に，ノイズノードとして記録されているイベントについては，読み込みをスキップする．以上を，ノイズノードが新しく発見されなくなるまで繰り返し，ノイズノードの除去を完了する．この繰り返し回数の最大値はノイズイベントの最大連続長であるが，実際には連続したノイズイベントが 1 回の処理で 1 個づつしか除去されないことは稀有であり，期待値は最大値よりもずっと小さい．経験的にほとんどの場合においてこの繰り返し処理は数回のうちに収束し，計算時間に大きな影響は与えない．

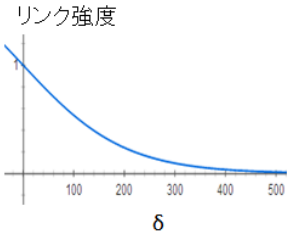


図 5　イベント時間間隔とリンク強度

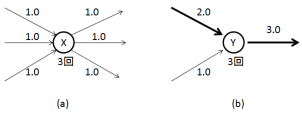


図 6　ノイズノードの除去

3.4.3　ノイズリンクの除去

イベント間の結びつきが弱い（連続して現れにくい）リンクを除去することで，エピソードに分割する．ノイズリンクの判断基準として 2 つの指標を用いる．1 つ目の指標はリンク強度の絶対値であり，2 つめの指標は，イベントの出現回数におけるリンク強度の割合である．それぞれの指標における閾値を $L1_{th}$，$L2_{th}$ とする．ここで，イベントの出現回数とは，リンクに接続する 2 つのイベントの出現回数の小さい方とする．それぞれについて，図 7 を用いて説明する．例えば $L1_{th}$=4 としたとき，リンク強度の絶対値が 4 未満のリンクをノイズリンクとし，図 7(b) がノイズリンクとなる．また，$L2_{th}$=0.5 としたとき，イベントの出現回数におけるリンク強度の割合が 0.5 未満のリンクをノイズリンクとし，図 7(a) と (c) がノイズリンクとなる．最終的にはこれらを組み合わせ，$L1_{th}$=2，$L2_{th}$=0.2 としたとき、図 7(c) をノイズリンクとして除去する．

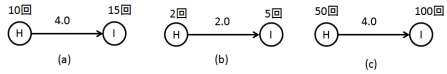


図 7　ノイズリンクの除去

このように 2 つの指標を組み合わせることで，低頻度だが連続して現れやすいリンク（図 7(b)）を残し，高頻度だが連続して現れやすいとは言えないリンク（図 7(c)）を除くことができる．従来手法のパラメータである最少出現回数は，本手法における $L1_{th}$ にあたり，頻度の絶対値しか考えないために網羅性と正確性のトレードオフに陥る．また，従来手法のもう一つの重要なパラメータであるエピソードの最大長も同様の性質を持つが，本手法では連続して現れやすいかどうかで自動的にエピソードを分割するため必要がない．

3.4.4　パターンの分離

複数のエピソードに含まれるイベントを推測して分離することでエピソードを分離する．本手法は，系列イベントの前後関係だけを用いて隣接イベントグラフを作成するため，複数のエピソードに含まれるイベントを介して，それらのエピソードが 1 つのグラフにまとまる問題がある．例えば，系列イベントの中に次の 3 つのエピソード，J→K→Z，O→Z→P，Z→S→T が含まれていたとする．すると，図 8(a) に示す隣接イベントグラフが生成され，3 つのエピソードが混合された 1 つのエピソードが抽出されてしまう．そこで，複数のエピソードに含まれるイベントを推測して分離することでエピソードを分離する．その方法について図 9 の例を併用して説明する．

まず，事前に隣接イベントグラフの各ノード v_s に対して，隣接ノードペア v_t-v_u の共起頻度 $fq(v_t, v_u)$ を記録しておく．隣接ノードペアの共起頻度とは，イベント系列の中で v_t,v_s,v_u，または，v_u,v_s,v_t の順に現れた回数である．図 9(b) はノード Z に関連付けられた隣接ノードペアの共起頻度情報である．例えば，一行目は隣接ノードペア O-P の共起頻度が 9 であることを示している．このノードに関連付けられた隣接ノードペアの共起頻度情報は，隣接イベントグラフを作成する際に容易に作ることができる．

隣接イベントグラフの各ノード $\forall v \in V$ に対して以下の処理を行う．

（1）　v_s の隣接ノードを抽出し，これを隣接ノード集合 NV_s とする．ノード Z の隣接ノード集合は図 9(a) に示すように，K，O，P，S である．

（2）　隣接ノード集合 NV_s の全ペア $v_t, v_u \in NV_s$ に対して，共起性による類似度を計算する．類似度 $sim(v_t, v_u)$ は，隣接ノードペアの共起頻度情報を用いて以下の式で計算する．

$$sim(v_t, v_u) = \frac{fq(v_t, v_u)}{Min(lw(s,t), lw(s,u))} \quad (3)$$

ここで，$lw(i,j)$ はノード v_i と v_j 間の双方向のリンク強度の和である．例えば，隣接ノードペア O，P の類似度は，$9 \,/\, Min\,(10.0, 9.0) = 1.0$ となる．

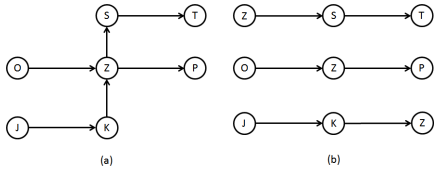


図 8　3つのエピソードの混合

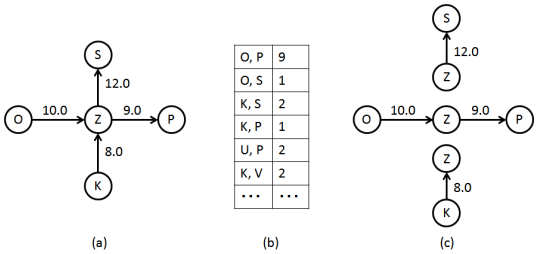


図 9　エピソードの分離

（3） 隣接ノードペアの類似度を用いて，隣接ノード集合に含まれるノードをクラスタリングする．クラスタリングの手法としては，クラスタ数を予め定める必要のない単連結法による階層型クラスタリングを用い，クラスタにカットする閾値を C_{th} とする．例えば，図 9(a)，(b) の例では，隣接イベント集合K，O，P，Sは（O，P），(K)，(S）の３つにクラスタリングされる．

（4） クラスタリングした隣接ノード集合を用いて，v_s を分離する．クラスタリングされたそれぞれのノードクラスターが独立したエピソードと考えられるため，クラスターの数だけ v_s を複製し，各ノードクラスター内に閉じてリンクを繋ぎかえるノードZの分離を行った後の状態を図 9(c) に示す．

以上の処理を隣接イベントグラフの各ノードに対して行うことで，最終的に図 8(b) に示すように本来の３つのエピソードに分離することができる．

3.4.5　半順序ノードの結合

エピソードに含まれる半順序ノード群の入出力リンクを補正する．この操作を行う理由について説明する．シリアルエピソード A → B → C → D が系列イベント中に n 回出現するとき，B の後に C が出現する期待値は n 回である．次に，ジェネラルエピソード（A B) → (C D) が系列イベント中に n 回出現する場合の B の後に C が出現する期待値を考える．(A B) → (C D) のインスタンス (実現パターン）は，1.A,B,C,D，2.A,B,D,C，3.B,A,C,D，4.B,A,D,C の４つであり，その中で B の後に C が出現しているのはインスタンス 1 だけなので，期待値は出現回数 n の 1/4 の n/4 回となる．この期待値は隣接イベントグラフにおけるリンク強度の期待値にあたる．つまり，半順序ノード群への入出力リンクの強度は確率的に元々低い値を取りやすい．よって，例えジェネラルエピソード（A B) → (C D) が 8 回出現してもノード B からノード C へのリンク強度の期待値は 8/4=2 であり，ノイズリンクとして除去される可能性が高くなる．このような問題に対応するために，エピソードに含まれる半順序ノード群の入出力リンクを補正する．その方法について説明する．

隣接イベントグラフの全ノードペア $v_s, v_t \in V$ に対して以下の処理を行う．

（1） ノード v_s, v_t について半順序関係の判定をする．ノード v_s, v_t の双方向のリンク強度の和を，それぞれのノードの出現回数の小さい方で割った値が閾値 P_{th} 以上のものを半順序関係とする．

（2） もし，半順序関係なら以降の処理を行う．

（3） ノード v_s, v_t を結合して，ノード v_{st} を生成する．

（4） ノード v_s, v_t の隣接ノード集合 NV_{st} とし，NV_{st} に含まれる各ノード $v_u \in NV_{st}$ とノード v_{st} 間に，ノード v_s と v_u，v_t と v_u 間のリンク強度を足し合わせたリンクを生成する．

（5） ノード v_s, v_t を除去する．

上記処理を，新しく半順序ノードの結合が起こらなくなるまで繰り返す．この繰り返し処理の回数は，系列イベントに含まれるエピソードの半順序集合の最大サイズに比例する．

3.4.6　エピソードの抽出

系列イベントから作成した隣接イベントグラフに対して，上述の操作１～４を行うことで最終的に生成された隣接イベントグラフの部分グラフがエピソードとして抽出される．

4.　シミュレーションデータによる評価実験

提案手法の有効性を確認するためにシミュレーションデータを用いて次の３つの実験を行った．

（1） 提案手法における各パラメータの影響度

（2） 提案手法における各操作の効果

（3） 提案手法と既存手法の比較評価

既存手法 [11] に対して、著者により公開されている C++のプログラム（closedepisodeminer(注1)：以下，CEM とする）を用いて比較した．比較実験では以下の項目について確認した．

- 高速性：エピソード抽出の計算時間の早さ
- 正確性：無駄なエピソードを出力していないか
- 網羅性：漏れなくエピソードを抽出できているか

まず、初めにシミュレーションデータの生成方法について説明し、次に、評価尺度である正確性と網羅性について説明する．その後、各実験の結果について説明を行う．

4.1　シミュレーションデータ

シミュレーションデータは，確率生成モデルとして次の 2 段階の方法で作成する．まず初めに、生起確率付きの正解エピソード集合を作成する．それから，生起確率付き正解エピソード集合からエピソードをサンプリングし、さらにそのエピソードからイベントをサンプリングすることでイベント列を生成する．

4.1.1　生起確率付き正解エピソード集合

エピソードとその生起確率をパラメータに従って確率的に生成する．パラメータは以下の通りである．なお，実験で用いた値を括弧書きで記す．

- エピソード数：N_{epi}(=100)
- エピソードの平均長：L_{ave}(=6)
- エピソードの最大長：L_{max}(=12)
- イベントの種類数：N_{eve}(=2000)

エピソードの長さ（イベントの個数）は，パラメータ L_{ave} の幾何分布に従って生成する．ただし，エピソードの最大長 L_{max}

(注1)：http://adrem.ua.ac.be/mining-closed-strict-episodes

を上限とする．エピソードのイベント列は，事前にイベントの種類数 N_{eve} に応じて作成したイベント集合からランダムに取得する．それぞれのイベント間の順序関係もランダムに決定する．以上，エピソードの作成をエピソード数 N_{epi} 繰り返し，最後に各エピソードに対してランダムに生起確率を割り振ることで生起確率付き正解エピソード集合を作成する．

4.1.2 生起確率付き正解エピソード集合からサンプリング

生起確率に従い，正解エピソード集合からエピソードをサンプリングし，エピソードからパラメータに従いイベントを生成する．その際に，ランダムにノイズイベントも生成する．この処理を生成したイベントの発生時刻が指定した時間 T_{seq} になるまで繰り返す．パラメータは以下の通りである．なお，実験で用いた値を括弧書きで記す．

- イベントの平均間隔時間：GT_{eve}（=1.0 / (60 * 5)）
- ノイズの平均間隔時間：GT_{noise}（=1.0 / (60 * 7)）
- エピソード間の平均間隔時間：GT_{epi}（=1.0 / (60 * 20)）
- エピソード内のノイズ生起確率：NG_{in}（=NG_{out}/5）
- エピソード外のノイズ生起確率：NG_{out}(=0.7)
- ノイズイベントの種類数：N_{noise}(=100)
- 経過時間 (日)：T_{seq}(=30)

まず、エピソードを生成するかノイズを生成するかをエピソード外のノイズ生起確率：NG_{out} に従って選択する．エピソードの生成は，生起確率付き正解エピソード集合から生起確率に従いサンプリングする．次にエピソード内のイベントを生成するかノイズを生成するかをエピソード内のノイズ生起確率：NG_{in} に従って選択し，これを，エピソード内のすべてのイベントを生成するまで繰り返す．イベントやノイズの発生時刻は，直前のイベントとの時間間隔をそれぞれパラメータ GT_{eve}，GT_{noise} の幾何分布に従って取得することで生成する．また，エピソードの開始時にはパラメーター GT_{epi} の幾何分布に従って取得した間隔時間を加える．

4.2 評価尺度

まずエピソード間のマッチ度を次のように定義する．もし，エピソード間のイベント列が一致しないならばマッチ度を 0.0 とする．イベント列が一致するならば，イベント間の順序関係の一致割合 (0.0〜1.0) をマッチ度とする．例えば，A → B → C と (A B) → C は，イベントペアは A-B，B-C，A-C の 3 つであり，そのうち B-C と A-C の順序関係が一致するため，マッチ度は 2/3 となる．このように完全一致ではなくマッチ度を導入する理由は，正解エピソードから発生し得るイベント列の全パターンが必ずしもシミュレーションデータ中に含まれるわけではなく、正解エピソードを完全に復元することは理論的に困難なためである．

エピソードマイニングの評価尺度として正確性と網羅性を定義する．正確性は，無駄なエピソードを出力していないかを測るもので，正解エピソード集合と出力されたエピソード集合間のマッチ度の合計を出力されたエピソード数で割ったものとする．網羅性は、漏れなく有益なエピソードを抽出できているかを測るもので，正解エピソード集合と出力されたエピソード集合間のマッチ度の合計を正解エピソード数で割ったものとする．また，高速性としてエピソード抽出の計算時間とする．今回 3 つの評価尺度を用いて実験を行った．

4.3 実験結果

4.3.1 提案手法における各パラメータの影響度

提案手法におけるパラメータは，リンク強度を決める δ，ノイズリンクの閾値である $L1_{th}$・$L2_{th}$，ノイズノードの閾値である H_{th}，エピソードの分離を制御する C_{th}，半順序ノードの結合を制御する P_{th} の 6 つである．これらのパラメータについての影響度を調べるために，シミュレーションデータ生成時のパラメータを変化させ（N_{epi}=50〜200，NG_{out}=0.3〜0.9，T_{seq}=30〜90），様々な入力データについて実験を行った．入力データの違い（エピソードの平均出現回数の多少，ノイズの多少）により精度にはばらつきがあったが，パラメータの影響のばらつきは少なかったため，入力データの違いによる考察は割愛する．

δ は前述したように自動設定可能であり，入力の系列イベントのイベント間の平均間隔時間とすればよい．それを示す実験結果を図 10 載せる．横軸はパラメータ δ を表し，縦軸は正確率と網羅率の調和平均（以降，F 値と呼ぶ.）を表す．このパラメータはイベント間の間隔時間とリンク強度の関係を制御し，大きくするほどイベント間の間隔時間によらずリンク強度は固定値（1.0) を取るようになる．図 10 が示すように，δ を固定値で与えた場合，60 から段階的に大きくしていくことで F 値は高くなり，3600 と 7200 をピークにそれ以降は F 値が下がった．このとき，自動設定での実験では，δ はおよそ 5000 であり，自動設定した結果は手動設定した結果の最大値とほぼ一致した．

ノイズリンクの閾値である $L1_{th}$・$L2_{th}$ について評価した結果を図 12 に載せる．図 12 から見て分かるようにどちらの閾値も比較的小さく設定する方が正確率，網羅率共に結果が良い．このことは前述したように 2 つの指標を組み合わせることで正確率と網羅率のトレードオフに陥ることを緩和していることを示している．

残り 3 つのパラメータの実験結果を，それぞれ図 11，13，14 に載せる．各パラメータはその特性は似ており，効果を無効にした状態（図 11 は左端，13 は左端，14 は右端）よりは効果を有効にした方が精度が良く、ただし効かせ過ぎる（図 11 は右端，13 は右端，14 は左端）と無効時より精度が悪くなる．

以上の実験より，提案手法における各パラメータは適切な固定値を設定できると考えている．以降の実験では，各パラメータをそれぞれ $L1_{th}$=2.0，$L2_{th}$=0.1，H_{th}=0.9，C_{th}=0.2，P_{th}=0.25 とした．

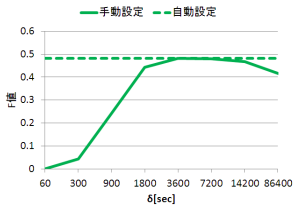


図 10　リンク強度の評価

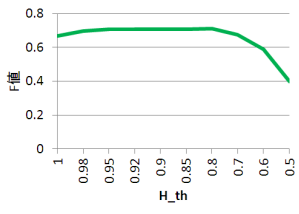
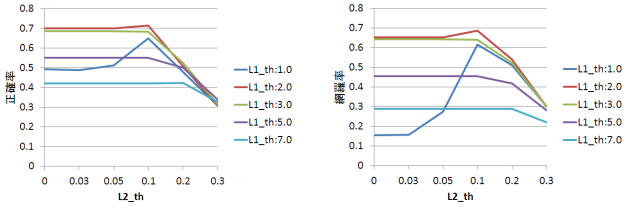


図 11　ノイズノードの評価

4.3.2 提案手法における各操作の効果

我々の提案手法の特徴は，入力の系列イベントから隣接イベントグラフを作成し，そこにノイズノードの除去，ノイズリン



[1] 正確率　　[2] 網羅率

図 12　ノイズリンクの評価

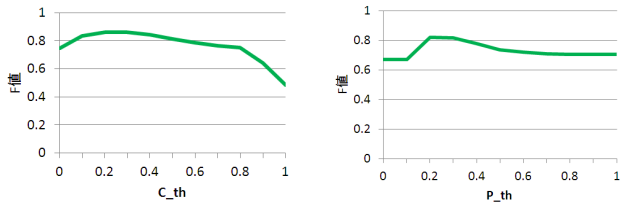


図 13　エピソード分離の評価　　図 14　半順序ノード結合の評価

クの除去，エピソードの分離，半順序ノードの結合という操作を行いエピソードを抽出することである．各操作の効果を調べるために，前実験と同様に様々な入力データについて実験を行ったが，結果のばらつきは少なかったため入力データの違いによる考察は割愛する．各操作を行う場合と行わない場合とでの評価結果を図 15 に載せる．横軸の項目は 4 つの操作の ON(1)/OFF(0) の組合せを表しており，上から半順序ノードの結合，エピソードの分離，ノイズノードの除去，ノイズリンクの除去（割合との複合）である．ノイズリンクの除去に関しては，リンク強度の絶対値は標準操作とし，そこにノードの出現回数におけるリンク強度の割合を組み合わせたものとする．縦軸は F 値である．

図 15 に示すように，各操作を全て行わない場合（一番左）に比べて，全て行う場合（一番右）の方が f 値は 2 倍以上に高くなった．また，各操作は組合せによっては何もしない時より F 値が下がっているケースも見られ，これら操作は相互補完的に働くと考えられる．また，各操作にはそれぞれ目的があり，そのような目的に合う特徴の入力データで実験した場合には F 値の差はより顕著に表れる．

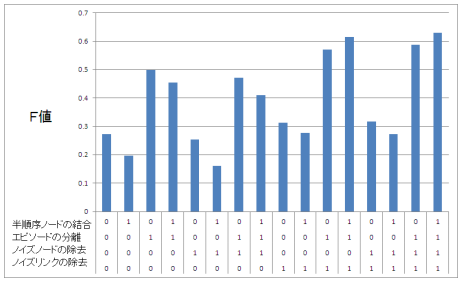


図 15　各操作の評価

4.3.3　提案手法と既存手法の比較評価

提案手法と既存手法の比較実験の結果を示す．前述したように既存手法としては著者により C++のプログラムが公開されている CEM [11] と比較した．我々の提案手法は Java で実装をした．また，実験に用いた PC は 2.26GHz の Xeon4 コアでメモリは 8GB である．

まず高速性についての実験結果を図 16 に示す．横軸は入力の系列イベントの全長を表し，縦軸は計算時間 (sec) を表している．提案手法は CEM と比較して系列イベントの長さに対し計算時間の増加が少ない．また，CEM はメモリを多く使用するため，スケールアウトさせていくと swap を起こすため急激に計算時間の増大が起こる．提案手法はメモリ使用量も少なく入力データサイズをスケールアウトしても高速に動作する．

高速性についての実験結果をもう一つ図 17 に示す．この実験は入力の系列イベントに含まれる正解エピソードの長さを変化させたときの計算時間を比較したものである．横軸は入力の系列イベントに含まれる正解エピソードの長さを表し，縦軸は計算時間 (sec) を対数で表している．図 17 から見て分かる通り，CEM は正解エピソードの長さに対して，指数関数的に計算時間が増加し、ある点からはメモリ不足で計算が不可能になった．これは，CEM が頻出候補となるエピソードを作成してから，各候補の出現回数を数える方法に起因しており，この候補数が正解エピソードの長さに対して指数関数的に増加しやすいためである．提案手法は，正解エピソードの長さに対して線形にしか計算時間は増えず高速に動作する．

次に，正確性・網羅性についての実験結果を，図 18 に示す．横軸は入力の系列イベントの全長を表し，縦軸は正確率と網羅率を表している．図 18 から見て分かるように，正確性については提案手法は CEM より大きく勝る．これは，CEM が正解エピソードのサブエピソードなど，冗長なエピソードを多く出力しているためである．今回の実験では平均数千のエピソードを出力していた．この中から実際に有益なエピソードを人手で見つけ出すのは困難である．

網羅性については，CEM の方が優れている．これは，正確性の結果も同様だが CEM と提案手法とのアプローチの違いによる．CEM では，条件を満たすエピソードをサブエピソードも含めて漏れなく列挙するのに対して，提案手法は，連続して現れやすいエピソードのみをピンポイントで出力する．そのために，一つでもノイズイベントが混ざると不正解とする今回の評価尺度では提案手法は網羅率を落としやすい．多少のノイズイベントは許容する評価尺度を用いることで網羅率の差は少し減少すると思われる．

また、入力の系列イベント長に対して、提案手法は正確率も網羅率も向上したが、CEM は網羅率は向上し正確率は低下するというトレードオフが発生した．この解釈は、図 3 を用いて説明すると、CEM は入力の系列イベント長が増加すると，出現頻度によるカットの閾値が相対的に低い方に移動することを意味する．その結果，網羅率は向上し、正確率は低下する．それに対して提案手法は、入力の系列イベント長が増加すると、隣接度によるカットの確率的な確度が上がるため正確率も網羅率も向上する．

5.　ファイル操作レコメンドへの適用

我々のグループでは，ファイル操作履歴から典型的なパターンをワークフローとして抽出し，抽出したワークフローを用いてユーザにファイル操作をレコメンドする手法を研究してき

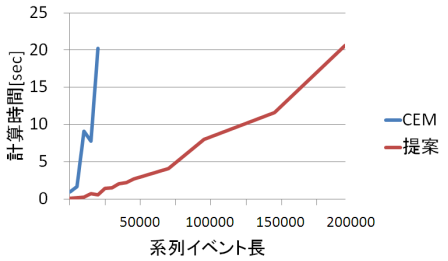


図 16　計算時間の比較 (系列イベント長)

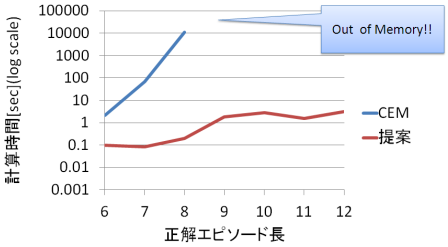


図 17　計算時間の比較 (正解エピソード長)

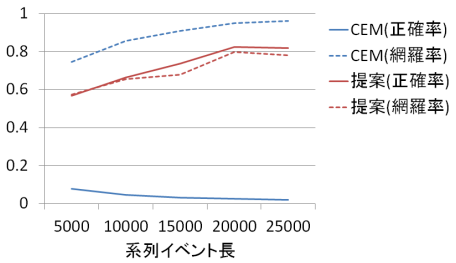


図 18　正確性・網羅性の比較

表 1　提案手法と抽象タスク法の比較

	抽象タスク法	提案手法
全テストケース数	162	
平均適合率	0.33	0.42
平均再現率	0.25	0.34
平均 F 値	0.26	0.36

た [4](以下，抽象タスク法と呼ぶ)．ここでは，本提案手法によりファイル操作履歴から抽出したエピソードをワークフローとして用いた場合のレコメンド精度の評価について述べる．評価結果を表 1 に載せる．評価には抽象タスク法で提案しているファイルの抽象化を行った後のファイル操作履歴をどちらも入力として用いた．表 1 から分かるように，本提案手法を用いて抽出したワークフローを用いた方が高い F 値を示した．これは抽象タスク法は，タスクやワークフローの分割にパラメータで固定したイベントの時間間隔を用いている点が挙げられる．つまり，抽象タスク法では作業が切り替わる際に一定時間以上の間隔が開くことを前提としているが，これが必ずしも成り立つわけではないので，成り立たない場合に精度の低下を招く．本提案手法は，同じ作業内でのイベントの繋がりやすさを前提としており，これは広く成り立つため今回のような結果になったと考える．

6. ま と め

系列データからの代表的な頻出パターン抽出方法であるエピソードマイニングにおいて，既存手法よりも適切なパターンに絞って，高速に抽出する手法を提案した．提案手法は，パターン内のイベントは連続して現れやすいと仮定して効率化を図っている．その手法は，連続して現れやすいイベントを繋ぐことで重み付き隣接グラフを作成し，その隣接グラフに対してリンクのカットやノードの分離，結合などの操作を行うことでパターンを抽出する．シミュレーションデータとファイル操作履歴などの実データを用いて評価実験を行い，提案手法は既存手法より適切なパターンを高速に抽出できることを確認した．特に大規模なデータに対して提案手法は有効性が顕著である．本手法を用いることで，大規模な様々な現実問題に対して高速に有益なパターンの抽出を行うことが期待できる．今後の課題として，実際の現実問題に対して本手法を適用したときの評価と，さらなる性能改善が挙げられる．特に，網羅性について既存手法に迫るよう改善していきたい．

文　献

[1] Unnikrishnan KP, Shadid BQ, Sastry PS, and Laxman S. Root cause diagnostics using temporal datamining. *U.S.Patent no. 7509234*, 2009.

[2] Srivatsan Laxman, Vikram Tankasali, and Ryen W. White. Stream prediction using a generative model based on frequent episodes in event sequences. In Ying Li, Bing Liu 0001, and Sunita Sarawagi, editors, *KDD*, pp. 453–461. ACM, 2008.

[3] Nicolas Meger and Christophe Rigotti. Constraint-based mining of episode rules and optimal window sizes. Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD' 04, pp. 313–324. Springer, 2004.

[4] 宋強, 川端貴幸, 伊藤史朗, 渡辺陽介, 横田治夫. ファイルレコメンデーションのためのファイル利用履歴に基づくタスク間ワークフロー抽出手法. 第四回データ工学と情報マネジメントに関するフォーラム (DEIM2012), 2012.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *ICDE*, pp. 3–14, 1995.

[6] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, Vol. 1, pp. 259–289, 1997.

[7] Gemma Casas-Garriga. Discovering unbounded episodes in sequential data. In Nada Lavrac, Dragan Gamberger, Hendrik Blockeel, and Ljupco Todorovski, editors, *PKDD*, Vol. 2838 of *Lecture Notes in Computer Science*, pp. 83–94. Springer, 2003.

[8] Srivatsan Laxman, P.S. Sastry, and K.P. Unnikrishnan. Discovering frequent episodes and learning hidden markov models: A formal connection. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 11, pp. 1505–1517, 2005.

[9] Srivatsan Laxman, P. S. Sastry, and K. P. Unnikrishnan. A fast algorithm for finding frequent episodes in event streams. In *KDD*, pp. 410–419, 2007.

[10] 大谷英行, 喜田拓也, 宇野毅明, 有村博紀. 極小出現区間を用いたエピソードマイニングの高速化. 情報処理学会研究報告. データベース・システム研究会報告 2008(56), 113-120, 2008-06-12, 2008.

[11] Nikolaj Tatti and Boris Cule. Mining closed strict episodes. *Data Min. Knowl. Discov.*, Vol. 25, No. 1, pp. 34–66, 2012.

[12] Avinash Achar, Srivatsan Laxman, Raajay Viswanathan, and P. S. Sastry. Discovering injective episodes with general partial orders. *Data Min. Knowl. Discov.*, Vol. 25, No. 1, pp. 67–108, 2012.