マルチ最小サポートを用いて継続時間と時間間隔を考慮した時系列パターンマイニングアルゴリズム

史　旭† 新谷 隆彦† 大森 匡† 藤田 秀之†

† 電気通信大学大学院情報システム研究科 〒 182–8585 東京都調布市調布ヶ丘 1-5-1
E-mail: †shixu@hol.is.uec.ac.jp, ††{shintani,omori,fujita}@is.uec.ac.jp

あらまし 時系列パターンマイニングはイベントの発生順序を考慮したパターンを抽出する重要な技術である．しかし，従来の手法は，イベントが時間的にどれだけ継続したか，また二つのイベント間にどれだけ間隔があるかを考慮しないため，意思決定を十分に支援できない場合がある．本研究では，イベントそれぞれの継続時間と時間間隔を階層に分割し，マルチレベルで継続時間と時間間隔を考慮した時系列パターンを抽出することを目的とする．さらに階層のレベルに従って異なる最小サポートとするマルチ最小サポートを用いて時系列パターンを抽出するアルゴリズムを提案する．

キーワード 時系列パターンマイニング，継続時間，時間間隔，マルチ最小サポート

XU SHI†, Takahiko SHINTANI†, Tadashi OHMORI†, and Hideyuki FUJITA†

† The University of Electro-Communications, Graduate School of Information Systems
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182–8585 Jappan
E-mail: †shixu@hol.is.uec.ac.jp, ††{shintani,omori,fujita}@is.uec.ac.jp

1. はじめに

蓄積された膨大なデータから潜在的に価値のある情報を見つけ出すデータマイニング技術の 1 つとして，時系列パターンマイニングの研究が進められてきた [1,2]．時系列パターンマイニングによってイベントの発生順序を示すパターンが抽出されるが，イベントの継続時間とイベント間の時間間隔は考慮されて来なかった．そのため，抽出されたパターン中のイベントがどれだけ継続したか，また二つのイベント間にどれだけ時間間隔があるかを知ることはできない．この問題を解決するために，イベントの継続時間とイベント間の時間間隔を考慮した時系列パターンの抽出手法が必要となる．

時系列パターンを抽出するには継続時間と時間間隔を離散化してアイテムにしなければならない．継続時間はイベント毎に，時間間隔はイベント間毎に分布や範囲が異なるため，時間を適切に分割することが困難である．そこで，本研究ではそれぞれイベントの継続時間とイベント間の時間間隔を階層に分割し，マルチレベルで継続時間と時間間隔を考慮した時系列パターンを抽出する手法を検討する．

また，従来の時系列パターンマイニングは閾値として単一の最小サポートを設定していた．階層で上位レベルのパターンの出現頻度は下位レベルのパターンの出現頻度を合わせた値となるため，上位レベルのパターンの出現頻度が高くなる．最小サポートを高く設定した場合，下位レベルのパターンが抽出されない場合がある．下位レベルのパターンを抽出するためには最小サポートを低く設定する必要があるが，この場合には大量のパターンが抽出されてしまう．そのため，階層を持つデータに対してはレベル毎に異なる最小サポートを設定するマルチ最小サポートが利用される場合がある [3]．

本研究ではイベントの継続時間と時間間隔のレベルに従って異なる最小サポートとするマルチ最小サポートにより継続時間と時間間隔を考慮した時系列パターンを抽出する．

2. 関連研究

時系列パターンマイニングに関する研究はこれまでに数多く行われてきた．イベントの発生順序のみを考慮する手法として，幅優先探索でパターン候補作成型の手法である GSP[1] と深さ優先探索でパターン成長型の手法である PrefixSpan[2] が提案されている．time window と呼ばれる所定の時間窓の間に発生する時系列パターンであるエピソードを抽出する手法も提案されている [4]．エピソードでは時間間隔が大きなパターンの抽出を回避できるが，様々な時間間隔を持つパターンを抽出することはできない．イベント間の時間間隔情報を扱った時系列パターンを抽出する研究がある [5,6]．I-PrefixSpan[5] は時間間

隔を平均的にいくつかの時間帯に分割し，単一レベルで時間間隔を考慮した時系列パターンを抽出する．また，I-PrefixSpan を拡張した MULTI-PrefixSpan[6] は時間間隔を階層に分割することで時間間隔が異なるレベルに属するパターンも抽出することができる．しかし，これら手法ではイベントの継続時間を考慮することができない．また，階層を考慮したイベントの組合せパターンを階層レベル毎に異なる最小サポートであるマルチ最小サポートにより抽出するアルゴリズムである MMS-Cumulate[7] が提案されているが，この手法では時系列パターンを抽出することができない．

3. 問 題 定 義

3.1 シーケンスデータ

本研究では開始時刻と終了時刻を持つイベントからなるデータを処理対象とする．イベント名 ($event$)，開始時刻 (t^{start})，終了時刻 (t^{end}) の組をイベントセット，イベントの開始時刻の順でソートしたイベントセットをシーケンスと呼び，$\langle(event_1, t_1^{start}, t_1^{end})(event_2, t_2^{start}, t_2^{end})\cdots(event_n, t_n^{start}, t_n^{end})\rangle$ と表現する．ここで，$(event_i, t_i^{start}, t_i^{end})$ は i 番目に開始されるイベントを示す．イベント i の継続時間は $t_i^{end} - t_i^{start}$，イベント i と $j(i<j)$ 間の時間間隔は $t_j^{start} - t_i^{end}$ となる．時間間隔がマイナスの場合はイベント j がイベント i の終了する前に開始したこと，ゼロの場合はイベント j がイベント i の終了した直後に開始したことを意味する．ID 毎に並べたシーケンスの集合をシーケンスデータと呼び，表 1 に例を示す．

表 1　シーケンスデータ

ID	シーケンス
1	$\langle(C,1,5)(A,6,7)(D,7,20)(B,15,25)(E,25,32)(B,35,40)\rangle$
2	$\langle(A,2,6)(F,4,7)(B,8,9)(A,9,17)(C,22,26)(B,26,30)\rangle$
...	...

3.2 継続時間と時間間隔の階層

時系列パターンを抽出するには継続時間と時間間隔を離散化してアイテムにしなければならない．しかし時間を細かい分割する場合，一定数以上なパターンを抽出することが困難である．時間を粗い分割する場合，詳細な継続時間情報と時間間隔情報を取れなくなってしまう．そこで，本研究ではそれぞれイベントの継続時間とイベント間の時間間隔を階層に分割し，さまざまな範囲で分割した離散値として扱う．これによりマルチレベルで継続時間と時間間隔を考慮した時系列パターンを抽出する．

イベント毎に継続時間の分布が異なるため，継続時間はイベント毎に深さ D_l，それぞれの親ノートが D_k 個の子ノートを持つ階層に分割する．分割には K-means クラスタリング [8] を用い，イベントを含む全てのシーケンスにおいて出現する継続時間の値を対象に，範囲分割する．生成された各ノードを 1 つのクラスタとして，そのノードにおける最小値と最大値間の区間を表す．図 1 に D_k を 2，D_l を 3 とした場合のイベント A の階層の例を示す．

イベントが属す継続時間クラスタを示すために，それぞれイベントの継続時間階層によって，そのイベントが属す全ての継続時間クラスタの集合をイベントセットにおけるイベント名と差し替え，$([event_i cluster], t_i^{start}, t_i^{end})$ の継続クラスタセット

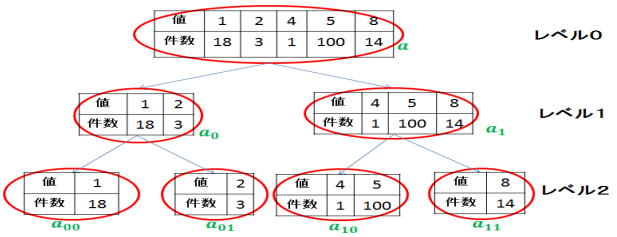


図 1　継続時間の階層

を生成する．イベントセット毎にイベントが属する継続時間のクラスタ集合を追加したシーケンスを継続クラスタシーケンスと呼ぶ．そして，継続クラスタシーケンスの集合を継続クラスタデータと呼び，第 4 節で提案する手法は本データを利用する．

時間間隔の階層は与えられた最小と最大の時間間隔からなる時間帯を平均的に深さ I_l，それぞれの親ノートが I_k 個の子ノートを持つ階層に分割する．生成された各のノードを 1 つの時間帯として，いつからいつまでの時間単位を表す．図 2 には最小時間間隔を 0，最大時間間隔を 27，I_k を 2，I_l を 3 とした場合の時間間隔階層の例を示す．

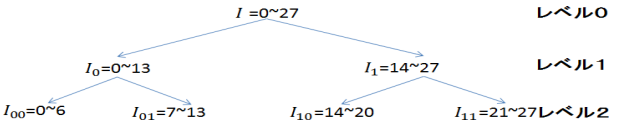


図 2　時間間隔の階層

3.3 継続時間と時間間隔を考慮した時系列パターン

以上継続時間と時間間隔の階層に基づいて，継続時間と時間間隔を考慮したマルチレベルの時系列パターン (以降パターンと呼ぶ) を定義する．

[定義 1]　継続時間アイテム ($Ditem$)：継続時間の階層における個々のクラスタを継続時間アイテムとする．

[定義 2]　時間間隔アイテム ($Iitem$)：時間間隔の階層における個々の時間帯を時間間隔アイテムとする．

[定義 3]　パターン：継続時間アイテムを先頭と末尾として，継続時間アイテムと時間間隔アイテムを交互に現れる開始時刻の順に並べたリストをパターンと呼び，$\langle Ditem_1 Iitem_{(1,2)} Ditem_2 Iitem_{(2,3)} \cdots Iitem_{(n-1,n)} Ditem_n\rangle$ と表現する．ここで，$Ditem_i$ は i 番目に開始される継続時間アイテム，$Ditem_j(j=i+1)$ は $Ditem_i$ の次に開始される継続時間アイテムを示す．$Iitem_{(i,j)}$ は $Ditem_i$ と $Ditem_j$ 間の時間間隔が属する時間間隔アイテムを表す．

[定義 4]　パターンの長さ：パターンを構成する継続時間アイテムの数である．

[定義 5]　シーケンスがパターンを含む：継続クラスタシーケンスの中に，パターンを構成する全ての継続時間アイテムが存在する．それらの継続時間アイテムの出現順序がパターンにおける継続時間アイテムの出現順序と一致し，また継続時間アイテム間の個々時間間隔がパターンに対応する時間間隔アイテムに属す場合，シーケンスがパターンを含むと言う．

[定義 6]　パターンの出現頻度：継続クラスタデータにパターンを含む異なるシーケンスの件数をパターンの出現頻度と呼ぶ．

［定義 7］ パターンのサポート：パターンの出現頻度に対する継続クラスタデータ中の全てのシーケンス件数の割合である．

［定義 8］ 頻出パターン：パターンのサポートは与えられたサポートの最小値 (最小サポートと呼ぶ) を満たすパターンを頻出パターンとする．

本研究では 2 つの継続時間アイテム間の時間間隔が最小時間間隔以上，最大時間間隔未満となる頻出パターンのみを抽出する．また，イベントが 1 つのシーケンスに重複して出現する場合，そのイベントからなる異なる継続時間アイテムと時間間隔アイテムによって生成された全てのパターンを抽出する．図 3 には時間間隔を図 2 に示す例の階層に分割する状態で，ある継続クラスタシーケンスにおいて重複して出現する 2 つのイベント A，B からなる考慮するパターンの例を挙げる．

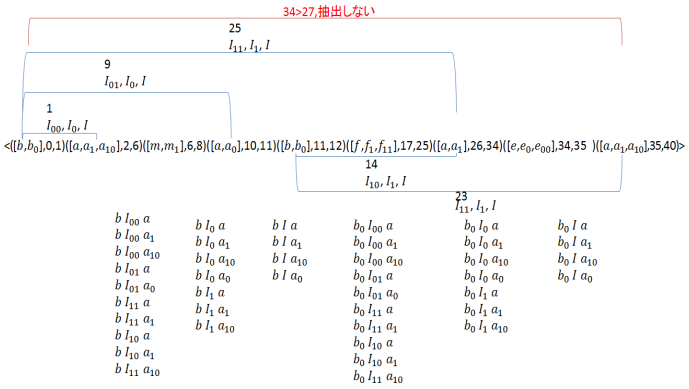


図 3　考慮するパターン

3.4　パターンのレベルとマルチ最小サポート

パターンは継続時間アイテムと時間間隔アイテムから構成され，継続時間アイテムと時間間隔アイテムはそれぞれ階層を持つことから，パターンに対しても継続時間の階層と時間間隔の階層を両方考慮してパターンの階層を定義する必要がある．

［定義 9］ 長さ 2 のパターンのレベル ($Plevel_2$)：$Plevel_2 = level_{Ditem_1} + level_{Iitem_{(1,2)}} + level_{Ditme_2}$ で計算する．ここで，$level_{Ditem}$ は継続時間アイテムのレベル，$level_{Iitem}$ は時間間隔アイテムのレベルを表す．

従って，継続時間と時間間隔をそれぞれ 3 レベルの階層に分割する場合，長さ 2 のパターンは上位レベル 0 から，下位レベル 6 までの 7 レベルの階層に分けられる．ここで，イベントの継続時間を $D_k = 2$，$D_l = 3$，時間間隔を $I_k = 2$，$I_l = 3$ の階層に分割し，生成されるイベント A の継続時間アイテム a，a_0，a_{00} とイベント B の継続時間アイテム b，b_0，b_{00}，また時間間隔アイテム I，I_1，I_{11} からなる 7 つのレベルに属する長さ 2 のパターンの例を表 2 に挙げる．

表 2　長さ 2 のパターンのレベル

レベル	長さ 2 のパターン
0	$\langle aIb \rangle$
1	$\langle a_0Ib \rangle, \langle aI_1b \rangle, \langle aIb_0 \rangle$
2	$\langle a_{00}Ib \rangle, \langle a_0I_1b \rangle, \langle a_0Ib_0 \rangle, \langle aI_1b_0 \rangle, \langle aIb_{00} \rangle, \langle aI_{11}b \rangle$
3	$\langle a_{00}I_1b \rangle, \langle a_{00}Ib_0 \rangle, \langle a_0I_{11}b \rangle, \langle a_0I_1b_0 \rangle, \langle a_0Ib_{00} \rangle, \langle aI_{11}b_0 \rangle, \langle aI_1b_{00} \rangle$
4	$\langle a_{00}I_{11}b \rangle, \langle a_{00}I_1b_0 \rangle, \langle a_{00}Ib_{00} \rangle, \langle a_0I_{11}b_0 \rangle, \langle a_0I_1b_{00} \rangle, \langle aI_{11}b_{00} \rangle$
5	$\langle a_{00}I_{11}b_0 \rangle, \langle a_{00}I_1b_{00} \rangle, \langle a_0I_{11}b_{00} \rangle$
6	$\langle a_{00}I_{11}b_{00} \rangle$

［定義 10］ 長さ 3 以上のパターンのレベル：長さ 3 以上のパターンの場合，パターンに現れる連続する $Ditem$ と $Iitem$ と $Ditem$ からなる長さ 2 のパターンのうち，最下位のレベルを持つ長さ 2 のパターンのレベルをこのパターンのレベルとする．

階層で上位レベルのパターンの出現頻度は下位レベルのパターンの出現頻度を合わせた値となるため，上位レベルのパターンの出現頻度が下位レベルのパターンの出現頻度より高くなる傾向がある．そのため，最小サポートについて，上位レベルのパターンに相対的に高い最小サポート，下位レベルのパターンに相対的に低い最小サポートによりパターンの頻出を判断する必要がある．本研究では，パターンのレベルに従い，各レベルで異なる最小サポートを設定し，複数の最小サポート (マルチ最小サポートと呼ぶ) を用いて頻出パターンを抽出する．

3.5　パターンの包含関係

同じ出現回数を持つ同じ長さの 2 つのパターン例えば，$\langle aIb \rangle$ と $\langle aI_1b \rangle$ がある場合，I_1 が I の子ノードとして，I より細かい時間間隔の時間帯を表すため，$\langle aI_1b \rangle$ のほうが $\langle aIb \rangle$ より明確に a が終了した後に b の開始時刻を示すことができる．また，$\langle aIb_1 \rangle$ と $\langle aIb_{11} \rangle$ がある場合，b_{11} が b_1 の子ノードとして，b_1 より細かい継続時間の範囲を表すため，$\langle aIb_{11} \rangle$ のほうが $\langle aIb_1 \rangle$ より明確にイベント B の継続時間を示す．この場合，$\langle aI_1b \rangle$ と $\langle aIb \rangle$，また $\langle aIb_1 \rangle$ と $\langle aIb_{11} \rangle$ を包含関係を満たすパターンと定義し，包含関係を満たす 2 つのパターンのうちより明確な情報を提供する $\langle aI_1b \rangle$ と $\langle aIb_{11} \rangle$ のほうがより良い意思決定のサポートができるパターンと考えられる．

［定義 11］ パターン m，n が以下 4 つの条件に従う場合，m と n が包含関係を満たすとする：

（1） m と n は，同じ長さ l のパターンである．

（2） m と n は，1 番目の継続時間アイテムから，$(l-1)$ 番目の継続時間アイテムまでの間の全ての継続時間アイテムと時間間隔アイテムが一致する．

（3） m と n は，l 番目で同じ継続時間アイテムを持ち，かつ $(l-1)$ 番目の継続時間アイテムと l 番目の継続時間アイテム間の時間間隔アイテムが時間間隔階層の上下関係を満たす．
或は m と n は，$(l-1)$ 番目の継続時間アイテムと l 番目の継続時間アイテム間の時間間隔アイテムが同じで，かつ l 番目の継続時間アイテムが継続時間階層の上下関係を満たす．

（4） m と n は，同じ出現回数である．

本研究では継続時間と時間間隔を階層に分割し，マルチレベルでパターンを抽出するため，大量のパターンが抽出されてしまう．従って，パターンが包含関係を満たす場合，継続時間と時間間隔情報をより明確に示すパターンのみを抽出することで，より実用性の高いパターンを得られることが期待できる．さらに，パターン数の爆発を抑え，抽出計算コストの削減にも役に立つと考えられる．

4. 提案手法

本研究では継続時間を持つイベントを階層に分割した継続時間アイテムと時間間隔を階層に分割した時間間隔アイテムを処理する必要がある．GSP のような候補作成型の手法では大量

の候補パターンが作成され，処理効率が悪くなることが予想される．そこで，本研究では GSP より数多くのパターンを効率よく抽出できるパターン成長型の手法である PrefixSpan を参考としたアルゴリズムを提案する．

4.1 PrefixSpan

まずはじめに PrefixSpan について述べる．PrefixSpan はイベントの出現順序のみを考慮した時系列パターンを効率よく抽出するアルゴリズムであり，主に以下 3 つのステップからなる．

- ステップ 1： シーケンスデータをスキャンして，長さ 1 の頻出パターン α を抽出する．
- ステップ 2： それぞれ長さ 1 の頻出パターン α に対し，PrefixSpan$(\alpha, l, DB\mid_\alpha)$ を呼び出す．ここで，α を Prefix，l を α の長さ，$DB\mid_\alpha$ を α の射影データベースとする．
- ステップ 3： PrefixSpan$(\alpha, l, DB\mid_\alpha)$ を再帰で実行する．(1)$DB\mid_\alpha$ をスキャンし，α に加えることが可能な頻出イベント β を見つけ出す．(2) 各頻出イベント β について，β を α に連接し，α を拡張したパターン α' を生成して，長さ $(l+1)$ の頻出パターンとして出力する．(3) 各 α' について，α' の射影データベース $DB\mid_{\alpha'}$ を生成し，PrefixSpan$(\alpha', l+1, DB\mid_{\alpha'})$ を呼び出す．

4.2 DI-PrefixSPM

継続時間と時間間隔を考慮した時系列パターンを抽出するために，PrefixSpan を拡張したナイーブなアルゴリズム Duration Interval-Prefix Sequential Pattern Mining(以降 DI-PrefixSPM と呼ぶ) を提案する．

4.2.1 DI-PrefixSPM のアルゴリズム

本手法では，それぞれのイベントの継続時間階層を記録した継続クラスタデータ，時間間隔階層，パターンレベルごとのマルチ最小サポートを入力する．主に 2 つのステップからなる．

- ステップ 1：PrefixSpan と同じ手順で最下位パターンレベルの最小サポートを満たすパターンを抽出する．
- ステップ 2：抽出されたそれぞれのパターンについて，パターンのレベルに対応する最小サポートを用いて頻出パターンを選択する．

また，PrefixSpan はイベントの発生順序のみを考慮するアルゴリズムであり，時間間隔を考慮しないため，1 件のシーケンスにイベントが重複して出現しても順序が変わらない場合には，Prefix ごとに 1 件のシーケンスに対して 1 回の射影を行うことで新たなパターンを生成するためのシーケンスデータが得られる．しかし，時間間隔を考慮する場合，重複して出現するイベントから生成された異なるパターンを抽出するため，Prefix ごとに 1 件のシーケンスに対してイベントの重複する回数の射影を行う必要がある．本研究ではそれぞれのシーケンスに対して，複数回の射影を行うことによって生成された射影データベースをマルチ射影データベースと呼ぶ．

4.2.2 DI-PrefixSPM の問題点

DI-PrefixSPM は PrefixSpan と同様に Prefirx を伸ばすことと射影データベースの生成によって頻出パターンを抽出する．しかし，長さ毎にマルチ最小サポートを用いてパターンを枝刈りすることができない．それは DI-PrefixSPM が長さ l の Prefix から長さ $(l+1)$ のパターンを探索するため，長さ l のパターンのレベルで頻出とならないパターンから，長さ $(l+1)$ のパターンを抽出することはできない．しかし，長さ l のパターンを 1 つ伸ばした長さ $(l+1)$ のパターンを作成するときにパターンのレベルが下がる場合がある．下位レベルに対応する最小サポートのほうが小さな値となるため，長さ $(l+1)$ のパターンが下位レベルに対応する最小サポートでは頻出パターンとなる可能性があるが，Prefix とする長さ l のパターンが枝刈りされてしまう場合，そのパターンを抽出できない．従って，DI-PrefixSPM では，最下位パターンレベルの最小サポートを満たすパターンを全て抽出した後に，マルチ最小サポートを用いて頻出パターンを選択する手法とした．しかし，DI-PrefixSPM では上位レベルの非頻出のパターンも最下位レベルの最小サポートを満たす場合には数えされてしまうため，冗長なパターンを大量に探索することになり，処理効率が落ちてしまう問題がある．

4.3 DI-SufPrefixSPM

DI-PrefixSPM の問題点を解決し，冗長なパターンの計算を回避しながら，マルチ最小サポートを満たす全ての継続時間と時間間隔を考慮した時系列パターンを抽出するアルゴリズム Duration Interval-Suffix Prefix Sequential Pattern Mining(以降 DI-SufPrefixSPM と呼ぶ) を提案する．

DI-PrefixSPM においてマルチ最小サポートを用いて頻出パターンを抽出しようとしたとき，全ての頻出パターンを抽出できない場合がある．あるパターンを伸ばしたときにそのレベルよりも下位レベルのパターンとなる場合があることが原因となることに注目する．DI-SufPrefixSPM では，パターンに現れる連続する $Ditem$ と $Iitem$ と $Ditem$ からなる長さ 2 のパターンのうち，最下位レベルを持つ長さ 2 のパターンをベースパターンと定義する．ベースパターンのレベルに対応する最小サポートを用いてパターンを前後に伸ばすことによって，頻出パターンを抽出する．

4.3.1 DI-SufPrefixSPM のアルゴリズム

本手法では，PrefixSpan と同様に頻出パターンである Prefix を伸ばすことと Prefix に対する射影を行うことを繰り返すことによって頻出パターンを抽出するだけでなく，頻出パターンを Suffix(接尾) にして，Suffix Projection という射影方法によって生成される Suffix 射影データベースを用いることで頻出パターンを抽出する必要がある．Suffix Projection とは，射影元のシーケンスから射影対象の Suffix より前に存在する継続クラスタセットからなるシーケンスのみを抽出する射影である．そして，与えられた継続クラスタデータベースに対し，Suffix に対する射影を行った結果のデータベースを Suffix 射影データベースと言う．Suffix 射影データベースにはその Suffix を接尾に持つ頻出パターンを見つけ出すために必要な継続クラスタデータが全て含まれることになる．Suffix を伸ばすことと Suffix に対する射影を行うことを繰り返すことによって頻出パターンを抽出する．また，本研究では Prefix 毎にマルチ射影データベースを生成すると同様に Suffix 毎に 1 件の継続クラスタシーケンスに対して継続時間アイテムの重複する回数の射影を行い，Suffix 毎にマルチ射影データベースの生成が必要である．

それぞれのイベントの継続時間階層を記録した継続クラスタデータ，時間間隔階層，パターンレベルごとのマルチ最小サポートを入力する．主なステップを以下に示す．

- ステップ 1：入力された継続クラスタデータベースにおいて，最下位パターンレベルの最小サポートを満たす長さ 1 のパターンを抽出する．
- ステップ 2：長さ 1 のパターンを Suffix(接尾) として，Suffix より前方に出現する継続時間アイテムを探索し，マルチ最小サポートを用いて長さ 2 の頻出パターンを抽出し，ベースパターンとする．
- ステップ 3：それぞれのベースパターンを Suffix として Suffix 射影を行い，ベースパターンのレベルに対応する最小サポートを用いて，前方に出現する 条件 1 を満たす継続時間アイテムを加え，再帰で頻出パターンである Suffix を成長させる．

– **条件 1** その継続時間アイテムは Suffix の 1 番目における継続時間アイテムと構成する長さ 2 のパターンがベースパターンのレベルより上位になること．

- ステップ 4：それぞれのベースパターンを Prefix として Prefix 射影を行い，ベースパターンのレベルに対応する最小サポートを用いて，後方に出現する 条件 2 を満たす継続時間アイテムを加え，再帰で頻出パターンである Prefix を成長させる．

– **条件 2** その継続時間アイテムは Prefix の一番最後における継続時間アイテムと構成する長さ 2 のパターンがベースパターンのレベルより上位，または同じレベルになること．

- ステップ 5：ステップ 3 で抽出されたそれぞれの頻出パターンを Prefix として Prefix 射影を行い，そのパターンにおけるベースパターンのレベルに対応する最小サポートを用いて，後方に出現する 条件 2 を満たす継続時間アイテムを加え，再帰で頻出パターンである Prefix を成長させる．

以上 5 つのステップにおいて，ステップ 3，4，5 は再帰で呼び出され，別々に長さ l から長 $(l+1)$ の頻出パターンを抽出するが，条件 1 と条件 2 の制限によって各ステップで抽出される頻出パターンが全て異なるため，重複して探索することなく全ての頻出パターンを抽出できる．

アルゴリズムを Algorithm1 に示す．サブルーチンとなる主な処理手順である Suffix を伸ばす処理 (SuffixDISPM) と Prefix を伸ばす処理 (PrefixDISPM) はそれぞれ Algorithm2 と Algorithm3 に示す．

4.3.2 DI-SufPrefixSPM の問題点

DI-SufPrefixSPM はマルチ最小サポートを用いた長さごとの枝刈りによって，頻出とならないパターンの計算を回避することで良い処理効率を実現する．しかし，DI-PrefixSPM で後ろ方向に頻出パターンを伸ばすのみであったことに対して，DI-SufPrefixSPM では後ろ方向だけでなく，前方向にも頻出パターンを伸ばす必要とある．従って，マルチ最小サポートによって枝刈りされるパターンが多い場合は DI-SufPrefixSPM の処理効率が良いが，そうではない場合は DI-SufPrefixSPM の処理性能が低下する場合があると考えられる．

Algorithm 1 DI-SufPrefixSPM($DB, Ihierarchy, minsup[pl]$)

Input: DB：継続クラスタデータベース
$Ihierarchy$：時間間隔階層
$minsup[pl]$：パターンレベルに対応する最小サポート
Output: 継続時間と時間間隔を考慮した頻出時系列パターン
Subroutine: GeneratePrefixMDB($\alpha', l', PrefixMDB\ |_\alpha$)
GenerateSuffixMDB($\alpha', l', SuffixMDB\ |_\alpha$)
PrefixDISPM($\alpha, l, baselevel, PrefixMDB\ |_\alpha$)
SuffixDISPM($\alpha, l, baselevel, SuffixMDB\ |_\alpha$)
Parameters: α, α'：継続時間と時間間隔を考慮した時系列パターン
l :α の長さ
l' :α' の長さ
$PrefixMDB\ |_\alpha$:α のマルチ Prefix 射影データベース
$SuffixMDB\ |_\alpha$:α のマルチ Suffix 射影データベース
L：継続時間と時間間隔を考慮した頻出時系列パターンの集合
L_l：長さ l の継続時間と時間間隔を考慮した頻出時系列パターンの集合
$baselevel$：ベースパターンのレベル
H_2：包含関係を満たした長さ 2 の上位レベルパターンの集合
Sup：サポート
$Ditem$：継続時間アイテム
$Iitem$：時間間隔アイテム

1: L^i =SuffixDISPM($null, 0, minsup$[最下位レベル], DB)
2: **for** α in L_2 **do**
3: $PrefixMDB\ |_\alpha$=GeneratePrefixMDB($\alpha, 2, DB$)
4: L^{ii} =PrefixDISPM($\alpha, 2, baselevel, PrefixMDB\ |_\alpha$)
5: **end for**
6: **for** α in L^i **do**
7: $PrefixMDB\ |_\alpha$=GeneratePrefixMDB(α, l, DB)
8: L^{iii} =PrefixDISPM($\alpha, l, baselevel, PrefixMDB\ |_\alpha$)
9: **end for**
10: $L = L^i + L^{ii} + L^{iii}$
11: L を出力

5. 評価実験

DI-PrefixSPM と DI-SufPrefixSPM の比較実験を通して，マルチ最小サポートにおける最下位レベルの最小サポートを変化させた場合の処理性能，シーケンス長またはシーケンス数を増加させた場合の処理性能，そして，マルチ最小サポートによる枝刈りの効果の 3 つの観点から 2 つのアルゴリズムの性能を評価する．また，包含関係を満たすパターンの抽出の回避によって，計算コストがどれだけ減少したかも考察する．

2 つのアルゴリズムを Java 言語で実装し，人工的に作成したデータを用いて，継続時間と時間間隔を最大 3 レベルの階層に分割した条件下で，評価実験を行った．実験で使うマシンは Intel(R) Xeon(R) 3.07GHz の CPU と 12GB のメモリを用いる．Java version は 1.7 であり，OS は ubuntu 12.04 である．

5.1 実験データ

人工生成データは [9] で紹介する synthetic data generation algorithm においてイベントに継続時間と時間間隔を追加し，Potentially large シーケンスにおけるイベントの継続時間クラスタが異なる分布になるように作成した．表 3 にデータ生成で設定する各パラメータを示す．そして，各パラメータを変更した 5 つの実験データを作成し，表 4 に示す．ここで，N_e=1000，L_p=5，N_p=1000，$Dmax$=27，I=0〜27 は一定値とした．

表 3 パラメータの説明

パラメータ	説明
N_e	イベント数
N_s	シーケンス数
L_s	シーケンス平均長
N_p	Maximal Potentially large シーケンス数
L_p	Maximal Potentially large シーケンスの平均長さ
$Dmax$	継続時間の最大値
I	時間間隔の区間

表 4 実験データ

ID	N_s	L_S
1	100,000	15
$l0$	100,000	10
$l2$	100,000	20
$n0$	50,000	15
$n2$	150,000	15

Algorithm 2 SuffixDISPM($\alpha, l, baselevel, SuffixMDB\ |_\alpha$)

Subroutine: GenerateSuffixMDB($\alpha', l', SuffixMDB\ |_\alpha$)
Method:

```
if l = 0 then
    L_1 = {α | sup_α ≧ minsup[最下位レベル]}
    for α in L_1 do
        SuffixMDB |_α=GenerateSuffixMDB(α, 1, DB)
        SuffixDISPM(α, 1, null, SuffixMDB |_α) を呼び出す
    end for
end if
if l = 1 then
    初期化 hashmapE
    for list M in SuffixMDB |_α do
        初期化 listP
        for 射影シーケンス in M do
            t^start を得る
            for Ditem in 射影シーケンス do
                interval = t^start − t^end, Ihierarchy によって {Iitems} を生成
                for Iitem in {Iitems} do
                    if !P.contains((DitemIitem)) then
                        P, E.key ← (DitemIitem), E.key.value + +
                    end if
                end for
            end for
        end for
    end for
    for (DitemIitem) in E do
        α' = (DitemIitem) + α
        if !包含関係&&Sup_α' ≧ minsup[α'のラベル] then
            baselevel = α'のラベル, l' = 2, L_2 ← α'
        end if
        if 包含関係 then
            H_2 ← α'
        end if
    end for
    for α' in L_2 do
        SuffixMDB |_α'=GenerateSuffixMDB(α', 2, SuffixMDB |_α)
        SuffixDISPM(α', 2, baselevel, SuffixMDB |_α') を呼び出す
    end for
end if
if l > 0 then
    初期化 hashmapE
    for list M in SuffixMDB |_α do
        初期化 listP
        for 射影シーケンス in M do
            t^start を得る
            for Ditem in 射影シーケンス do
                interval = t^start − t^end, Ihierarchy によって {Iitems} を生成
                for Iitem in {Iitems} do
                    β = (DitemIitem) + αの一番目の Ditem
                    if β のレベルが baselevel より上位&&!β ∈ H_2
                    &&!P.contains((DitemIitem)) then
                        P, E.key ← (DitemIitem) , E.key.value + +
                    end if
                end for
            end for
        end for
    end for
    for (DitemIitem) in E do
        α' = (DitemIitem) + α
        if !包含関係&&Sup_α' ≧ minsup[baselevel] then
            l' = l + 1, L_l' ← α'
        end if
    end for
    for α' in L_l' do
        SuffixMDB |_α'=GenerateSuffixMDB(α', l', SuffixMDB |_α)
        SuffixDISPM(α', l', baselevel, SuffixMDB |_α') を呼び出す
    end for
end if
```

Algorithm 3 PrefixDISPM($\alpha, l, baselevel, PrefixMDB\ |_\alpha$)

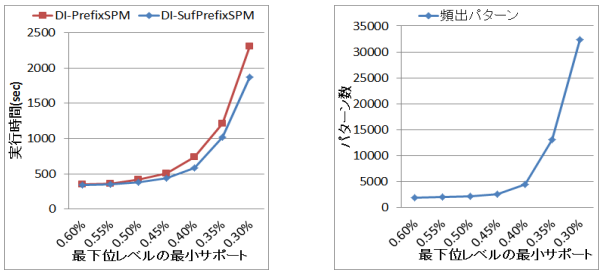
Subroutine: GeneratePrefixMDB($\alpha', l', PrefixMDB\ |_\alpha$)
Method:

```
初期化 hashmapE
for list M in PrefixMDB |_α do
    初期化 listP
    for 射影シーケンス in M do
        t^end を得る
        for Ditem in 射影シーケンス do
            interval = t^start − t^end
            Ihierarchy によって, {Iitems} を生成
            for Iitem in {Iitems} do
                β = αの一番最後の Ditem + (IitemDitem)
                if β のレベルが baselevel より上位, または baselevel と同じ
                &&!P.contains((IitemDitem)) then
                    P, E.key ← (IitemDitem), E.key.value + +
                end if
            end for
        end for
    end for
end for
for γ in L^i do
    if γの Prefix = α then
        E.key ← (IitemDitem) = γ − γの Prefix, E.key.value = Sup_γ
        L^iから削除
    end if
end for
for (IitemDitem) in E do
    α' = α + (IitemDitem)
    if !包含関係&&Sup_α' ≧ minsup[baselevel] then
        l' = l + 1
        L_l' ← α'
    end if
end for
for α' in L_l' do
    PrefixMDB |_α'=GeneratePrefixMDB(α', l', PrefixMDB |_α)
    PrefixDISPM(α', l', baselevel, PrefixMDB |_α') を呼び出す
end for
```

5.2 最小サポートの変化と処理性能

データ 1 を用いてパターンレベルの最小サポート間の差を 0.05%に設定し，最下位レベルの最小サポートを 0.6%から 0.3%まで変化させた場合の 2 つアルゴリズムの実行時間，処理したパターンの数，抽出する頻出パターンの数を考察する．

図 4(a) に 2 つのアルゴリズムが各最下位レベルの最小サポートでの実行時間を表す．また図 4(b) には各最下位レベルの最小サポートで，2 つのアルゴリズムによって抽出された頻出パターンの数を表す．図 4 によると，最下位レベルの最小サポートが小さくなるに従って，DI- PrefixSPM と DI-SufPrefixSPM の実行時間が指数的に増加する．これは最小サポートの設定が低いほど，頻出となるパターンが増加することが原因である．また，最下位レベルの最小サポートが小さくなるに従って，DI-PrefixSPM と DI-SufPrefixSPM の処理時間の差が増大していることが分かった．

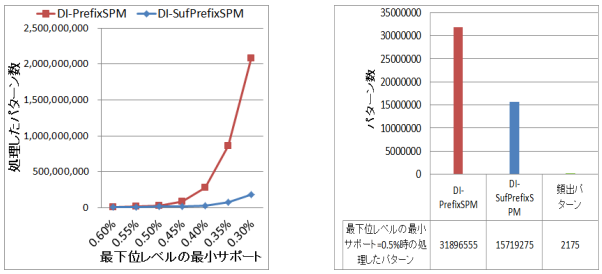


(a) 実行時間　　(b) 頻出パターン

図 4　データ 1 の実行結果

図 5(a) に最下位レベルの最小サポート毎に 2 つのアルゴリズムが処理したパターンの数を表す．図 5(b) には最下位レベルの最小サポートが 0.5%の場合，処理したパターンの数とマルチ最小サポートにより頻出と判断されたパターンの数を表す．図 5(a) から，最下位レベルの最小サポートが小さくなるに従って，DI-PrefixSPM と DI-SufPrefixSPM が実行中に処理したパターンの数の差が増大していることが分かった．これは 2 つのアルゴリズムの処理時間の差が増大する原因となる．図 5(b) によって，最下位レベルの最小サポートが 0.5%の場合，DI-PrefixSPM により処理したパターンの数が DI-SufPrefixSPM

の 2 倍以上となる．頻出パターンとして抽出されるのは処理したパターンよりかなり少ない数の 2175 件である．つまり，マルチ最小サポートによって，処理中で長さ毎に枝刈りできるパターンがたくさん存在し，DI-PrefixSPM がステップ 1 の処理でそれらのパターンを全て計算してしまうため，実行時間が DI-SufPrefixSPM より遅くなった．



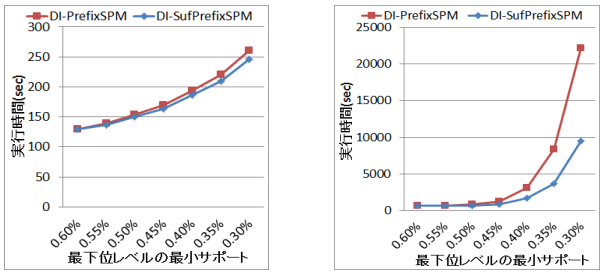
(a) 処理パターン　　(b) 最下位最小サポート=0.5%

図 5　データ 1 の実行中の処理パターン

5.3　シーケンス長，シーケンス数の変化と処理性能

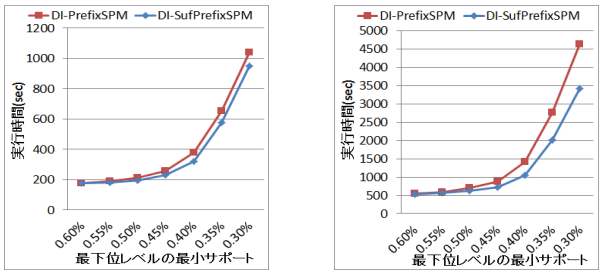
表 4 の 5 つのデータを用いて，パターンレベルの最小サポート間の差を 0.05%に設定し，最下位レベルの最小サポートを 0.6%から 0.3%まで変化させた場合のシーケンスの長さとシーケンスの数の増加に対する DI-PrefixSPM と DI-SufPrefixSPM の実行時間の変化を考察する．

図 6 には同じシーケンス数を持ち，シーケンス平均長が 10 と 20 のデータ $l0$，$l2$ の実行時間を示す．図 7 には同じシーケンス平均長であり，シーケンスの数が 50000 と 150000 のデータ $n0$，$n2$ の実行時間を示す．



(a) データ $l0$ の実行時間　　(b) データ $l2$ の実行時間

図 6　シーケンス長と実行時間



(a) データ $n0$ の実行時間　　(b) データ $n2$ の実行時間

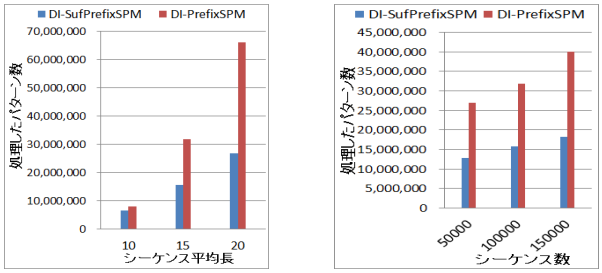
図 7　シーケンス数と実行時間

図 4(a) と図 6 によって，同じシーケンス数でシーケンス平均長を 5 ずつ増加させた場合，シーケンス長の増加に従って，DI-PrefixSPM と DI-SufPrefixSPM の実行時間の差が増加するすることが分かった．つまり，シーケンスの長さが長いほど，DI- SufPrefixSPM は DI-PrefixSPM より処理効率がよくなる．これはシーケンスの長が長いほど，1 件のシーケンスが新たなパターンを含む可能性が高くなり，それら枝刈りの対象になるパターンが DI-PrefixSPM のステップ 1 によってたくさん計算され，処理効率が落ちてしまう．

図 8(a) にはシーケンス数が 100000，最下位レベルの最小サポートが 0.5%の場合，シーケンス平均長は 10，15，20 で DI-PrefixSPM と DI-SufPrefixSPM が実行中に処理したパターンの数を示す．図 8(a) によると，シーケンス長の増加に従って，処理したパターンの数について DI-PrefixSPM のほうが DI-SufPrefixSPM より大幅に増加していることが分かった．

そして図 4(a) と図 7 から，同じシーケンス平均長で，シーケンスの数を 50000 ずつ増加させた場合，シーケンス数の増加によって，DI-PrefixSPM と DI-SufPrefixSPM の実行時間の差が増加することが分かった．つまり，シーケンスの数が多いほど，DI- SufPrefixSPM は DI-PrefixSPM より処理効率がよくなる．これはシーケンス数の増加と共に，処理途中で生成されるパターンが多くなり，それらのパターンがマルチ最小サポートにより頻出にならないが，DI-PrefixSPM のステップ 1 ではそれらパターンに対する無駄な計算をたくさん行った．

図 8(b) にはシーケンス平均長が 15，最下位レベルの最小サポートが 0.5%の場合，シーケンス数は 50000，100000，150000 で DI-PrefixSPM と DI-SufPrefixSPM が実行中に処理したパターンの数を示す．図 8(b) によると，シーケンス数の増加に従って，DI-PrefixSPM と DI-SufPrefixSPM により処理したパターンの数の差が拡大していることが分かった．



(a) シーケンス長と処理パターン　　(b) シーケンス数と処理パターン

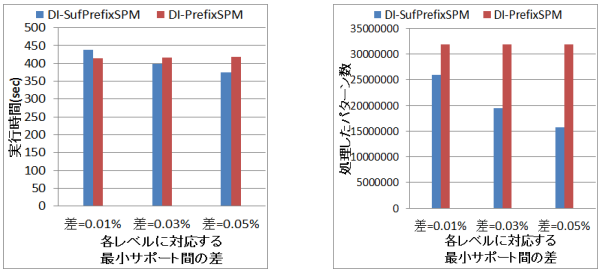
図 8　シーケンス長，シーケンス数と処理パターン

5.4　枝刈りと処理性能

4.3.2 節で述べた DI-SufPrefixSPM の問題点より，マルチ最小サポートの設定は DI-SufPrefixSPM の処理効率に影響する．各パターンレベルに対応する最小サポート間の差が大きければ大きいほど枝刈りのパターンが多くなり，DI-SufPrefixSPM の処理効率がよくなる．最小サポート間の差が小さければ小さいほど，枝刈りのパターンが少なくなり，DI-SufPrefixSPM の処理効率が落ちる．

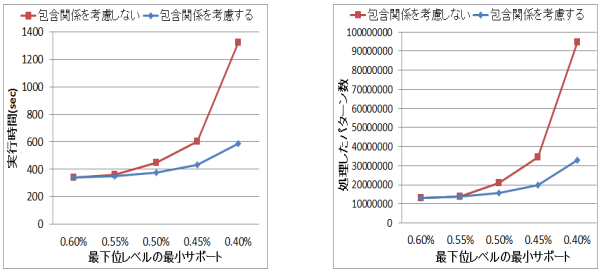
ここでデータ 1 を対象に，最下位レベルの最小サポートが 0.5%の場合，各パターンレベルに対応する最小サポート間の差を 0.01%，0.03%，0.05%に設定した場合の DI-Suf PrefixSPM と DI-PrefixSPM の実行時間と処理したパターンの数を比較する．結果を図 9 に示す．

図 9 によると，各パターンレベルに対応する最小サポート間



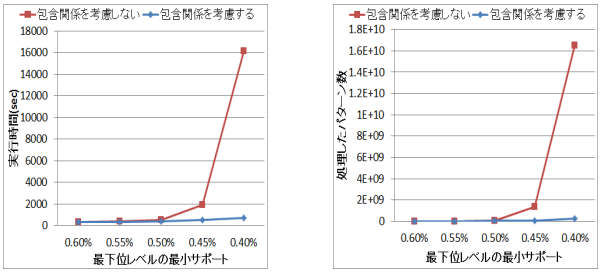
(a) 実行時間比較　(b) 処理パターン比較

図 9　枝刈りと処理性能



(a) 実行時間　(b) 処理パターン

図 10　包含関係と DI-SufPrefixSPM の処理性能



(a) 実行時間　(b) 処理パターン

図 11　包含関係と DI-PrefixSPM の処理性能

の差が高くなるに従って，マルチ最小サポートにより枝刈り対象になるパターンの数が増加し，DI- SufPrefixSPM により処理したパターンの数が減少することで，DI-SufPrefixSPM の実行時間が減少している．DI-PrefixSPM がパターン探索処理のステップ 1 では最下位レベルの最小サポートによりパターンを抽出することで，最下位レベルの最小サポートが同じ場合，各パターンレベルに対応する最小サポート間の差の変化による処理したパターン数の変化がないため，DI-PrefixSPM の実行時間がほとんど変わらない．また，各パターンレベルに対応する最小サポート間の差が 0.01%のときに，マルチ最小サポートの枝刈り効果が弱いため，DI-SufPrefixSPM と DI-PrefixSPM が実行中に処理したパターンの数の差が小さく，DI-SufPrefixSPM は DI-PrefixSPM より実行時間が多くかかることになった．一方，各パターンレベルに対応する最小サポート間の差が 0.03%と 0.05%の場合，マルチ最小サポートによる枝刈りによって，DI-SufPrefixSPM と DI-PrefixSPM が実行中に処理したパターンの数の差が拡大し，DI-SufPrefixSPM の実行時間は DI-PrefixSPM より早くなった．

5.5　パターン包含関係の影響

パターン包含関係の影響について，包含関係を満たすパターンを考慮することなく，全て抽出する場合と包含関係を満たすパターンの抽出を回避する場合の処理性能の変化を考察する．

データ 1 を使って，各パターンレベルの最小サポート間の差を 0.05%に設定し，最下位レベルの最小サポートを 0.6%から 0.4%まで変化させ，DI-SufPrefixSPM の処理において，包含関係を考慮しない場合の実行時間と処理したパターンの数を図 10 に示す．また，DI-PrefixSPM の処理において，包含関係を考慮しない場合の実行時間と処理したパターンの数を図 11 に示す．図 10 と図 11 から，最下位レベルの最小サポートが小さくなるに従って，DI-SufPrefixSPM と DI-PrefixSPM において，包含関係を考慮しない場合処理したパターンの数が包含関係を考慮した場合処理したパターンの数より大幅に増加し，実行時間に大きな影響を与えたことが分かった．

6.　おわりに

本研究では，イベントの継続時間と時間間隔情報を考慮した時系列パターンを抽出するアルゴリズムを検討した．継続時間と時間間隔を離散化してアイテムにするときに，時間を適切に分割することが困難であることを考慮し，継続時間と時間間隔をそれぞれ階層に分割した．また，階層を持つデータに対してはレベル毎に異なる最小サポートを設定するマルチ最小サポートを用いて，継続時間と時間間隔を考慮した頻出時系列パターンを抽出する 2 つのアルゴリズム DI-PrefixSPM と DI-SufPrefixSPM を提案した．実験を通して，マルチ最小サポートの設定によるが，枝刈りを可能とした DI- SufPrefixSPM は DI-PrefixSPM よりも優れていること，また，包含関係を満たすパターンを考慮することで DI-PrefixSPM と DI-SufPrefixSPM の処理効率が高まることを示した．

文　献

[1] R.Srikant, R.Agrawal, “ Mining Sequential Patterns: Generalization and Performance Improvements ”, Extending Database Technology, pp.3-17, 1996.

[2] J.Han, J.Pei, B. Mortazavi-Asl., Q. Chen, U. Dayal, and M.-C. Hsu, “ FreeSpan: frequent pattern-projected sequential pattern mining ”, ACM SIGKDD Conference on Knowledge Discovery and Data, pp.355-359, 2000.

[3] J.Han, M.Kamber, “ Data mining: concepts and techniques ”, Second Eedition, Morgan Kaufmann, 2006.

[4] H.Mannila, H.Toivonen, A.I.Verkamo, “ Discovery of frequent episodes in event sequences ”, Data Mining and Knowledge Discovery, pp.259-289, 1997.

[5] Y.L.Chen, M.C.Chiang, and M.T. Ko.,“ Discovering Time-interval Sequential Patterns in Sequence Databases ”, Expert Systems with Applications, pp.343—354, 2003.

[6] Y.H.Hu, F.Wu, and C.I. Yang,“ Mining Multi-level Time-interval Sequential Patterns in Sequence Databases ”, Software Engineering and Data Mining, pp.23-25, 2010.

[7] M.C.Tseng, W.Y.Lin, “ Efficient mining of generalized association rules with non-uniform minimum support ”, Data and Knowledge Engineering, pp.41-64, 2007.

[8] MacQueen, J.B., “ Some methods for classification and analysis of multivariate observations ”, Proceedings of the Symposium on Mathematics and Probability, 5th, Berkely, pp.281—297, 1967.

[9] R.Agrawal and R.Srikant, “ Mining sequential patterns ”, Research Report RJ 9910, IBM AI-maden Research Center, San Jose, California, 1994.