時系列パターンマイニングの並列化

新谷 隆彦　喜連川 優
東京大学 生産技術研究所
〒 106 東京都港区六本木 7–22–1
TEL:03-3402-6231 FAX:03-3423-2834
{shintani, kitsure}@tkl.iis.u-tokyo.ac.jp

あらまし

データマイニングで得られる情報のひとつとして，時系列パターンがある．データマイニングが処理対象とする履歴データは時間情報を含んでいる場合が多く，時間軸に沿った解析を行うことにより時間の持つ意味を含むルールの抽出が可能となる．時系列パターンはデータの時間情報を考慮したデータの出現パターンを示すルールである．時系列パターン抽出には数多くのデータの順列を調べ，データベースを繰り返し検索する必要があるため，その処理は高負荷となり，効率の良い処理には並列化が不可欠である．本稿では，分散メモリ型並列計算機環境上での時系列パターン抽出の並列処理方式について述べる．また，実際の分散メモリ型並列計算機上に実装し，提案する並列処理方式の有効性を明らかにする．

キーワード：　データマイニング，時系列パターン，並列処理方式

Parallelization of Sequential Pattern Mining

Takahiko SHINTANI and Masaru KITSUREGAWA
Institute of Industrial Science, University of Tokyo
7–22–1 Roppongi, Minato-ku, Tokyo 106

Abstract

One of the problem in data mining is discovery of sequential patterns in large databases, where a sequential pattern is an ordered list of sets of items. Sequential pattern mining requires examining a large number of candidate sequences and reading the database repeatedly. It is expected that parallel algorithms significantly help to improve the performance. In this paper, we present the parallel algorithms for mining sequential patterns on a shared-nothing environment. We implement these algorithms on a shared-nothing environment and describe the performance evaluation from this implementation.

Keywords :　data mining, sequential pattern, parallel algorithm

1 はじめに

大量に蓄積された履歴データを解析し，その中に埋もれた有用な情報を抽出する研究として「データマイニング (Data Mining)」が注目されている．データマイニングで得られる情報のひとつとして，時系列パターン (sequential pattern)[1, 2, 3] がある．データマイニングが処理対象とする履歴データは時間情報を含んでいる場合が多く，時間軸に沿った解析を行うことにより時間の持つ意味を含むルールの抽出が可能となる．時系列パターンは「全体の 5%の顧客は A と B を購入した次の買物で C を購入する」のようなデータの時間情報を考慮したデータの出現パターンを示すルールである．時系列パターン抽出には数多くのデータの順列を調べ，データベースを繰り返し検索する必要があるため，その処理は高負荷となり，効率の良い処理には並列化が不可欠である．

我々は現在までにデータマイニングにおける相関ルール (association rule) 抽出の分散メモリ型並列計算機環境を基本的なモデルとした並列処理方式 HPA(Hash Partitioned Apriori) を提案してきた [4]．相関ルールは「A と B を購入した顧客の 90%が同時に C も購入している」のようなデータ間の関連性を示すルールである．HPA ではデータベースをノード間に分割して割り当てることによるディスク入出力処理の並列化のみでなく，探索候補もノード間に分割して割り当てることによりシステム全体の主記憶空間を効果的に活用する．

本稿では，データマイニングにおける時系列パターン抽出の並列処理方式について述べる．また，実際の分散メモリ型並列計算機上に実装し，提案する並列処理方式の有効性を明らかにする．

2 時系列パターン

はじめに，時系列パターンを次のように定義する [1]．アイテムの集合を $\mathcal{I} = \{i_1, \ldots, i_m\}$ とし，k 個のアイテムの組合せを長さ k のアイテム集合 (k-itemset) と呼ぶ．顧客シーケンスデータベースを $\mathcal{D} = \{s_1, \ldots, s_n\}$ とする．各要素 $s_i = \{t_1, \ldots, t_m\}$ を顧客シーケンスと呼ぶ．各シーケンスはトランザクションの集合であり，時刻の順に並べられている．各トランザクション t_j はアイテム集合 ($t_j \subseteq \mathcal{I}$)，顧客 ID，トランザクション ID，時刻で構成される．時系列パターンは $\langle a_1, \ldots, a_x \rangle$ で表現され，各要素 a_y はアイテム集合である．時系列パターンは支持度として，顧客シーケンスデータベース $\mathcal{D}$ のうち，そのパターンを含む割合により評価される．また，シーケンス $s_a = \langle a_1, a_2, \ldots, a_n \rangle$ と $s_b = \langle b_1, b_2, \ldots, b_m \rangle$ について，$a_{i_1} \supseteq b_1, a_{i_2} \supseteq b_2, \ldots, a_{i_m} \supseteq b_m$ を満たす整数 $i_1 < i_2 < \ldots < i_m$ が存在するとき，s_a は s_b を含むと表現する．

時系列パターン抽出問題はユーザにより設定された最小支持度を満足する全てのパターンを見出すことに相当する．その処理を次に示す．はじめにデータベースを検索し，各アイテムの支持回数を数え上げ，最小支持度を満足するものを取り出し，長さ 1 の頻出シーケンス (large 1-sequence : L_1) とする．ここで，k 個のアイテムからなる最小支持度を満たすシーケンスを長さ k の頻出シーケンス (large k-sequence : L_k) と呼ぶ．次に L_1 から 2 つのアイテムからなるシーケンスを作成する．これらを長さ 2 の候補シーケンス (candidate 2-sequence : C_2) と呼ぶ．再びデータベースを検索して支持度を求め，頻出シーケンスを決定する．以降，長さ k の頻出シーケンスを求める処理 (パス k) は次のようになる．

1. 候補シーケンス作成:
   長さ $(k-1)$ の頻出シーケンス (L_{k-1}) から長さ k の候補シーケンス (C_k) を作成する．
2. 支持度の数え上げ:
   データベースを検索し，支持度を求める．
3. 頻出シーケンスの決定頻出シーケンスの決定:
   最小支持度を満たすものを取り出し，長さ k の頻出アイテム集合とする．

この処理により取り出された頻出シーケンスが求める時系列パターンである．

3 並列処理方式

単一プロセッサの主記憶上にすべての候補シーケンスを保持できる場合には，並列化は容易で

ある．顧客シーケンスデータベースをプロセッサ間に分割して割り当てることにより，ディスク入出力の並列化が実現できる．しかし，データマイニングが処理対象とするデータベースが大規模であるため，単一プロセッサがすべての候補シーケンスを保持できない場合が多い．また，時系列パターン抽出では探索候補としてアイテムの順列を調べなければならないため，その探索候補数は相関ルール抽出よりも多くなる．本稿では，すべての候補シーケンスを単一プロセッサの主記憶上に保持できない場合に関する考察を進める．

3.1 Non Partitioned Sequential Pattern Mining : NPSPM

NPSPM では候補シーケンスを全ノードに複製して処理する．図 1 にノード n でのパス k の処理を示す．また，用いた記号の定義を表 1 に示す．

表 1: 記号の定義

$\mathcal{L}_k$	Sets of all the frequent k-sequences
$\mathcal{C}_k$	Sets of all the candidate k-sequences
$\lvert\mathcal{C}_k\rvert$	The size of $\mathcal{C}_k$ in bytes
$\mathcal{D}^n$	Customer-sequences stored in the local disk of the n-th node
C_k^d	Sets of fragments of the candidate k-sequences. Each fragments fits in the local memory of a node.
$\lvert C_k^d\rvert$	The size of C_k^d in bytes
L_k^d	Sets frequent k-sequences derived from C_k^d

1. 候補シーケンス作成：
 $\mathcal{L}_{k-1}$から $\mathcal{C}_k$を作成し，全ノードに複製する．
2. 支持度の数え上げ：
 ローカルディスクから顧客シーケンスデータベースを読み出し，各顧客シーケンスから長さ k のシーケンスを作成することにより $\mathcal{C}_k$の支持回数を数え上げる．
3. 頻出シーケンス決定：
 全ての顧客シーケンスデータに対する処理が終了した時点で，$\mathcal{C}_k$の全ノードでの支持回数を取りまとめノードに送信する．取りまとめノードは $\mathcal{C}_k$の全ノードでの支持回数の和を求め，支持度を計算し，$\mathcal{L}_k$を決定し，全ノードに放送する．

$k \geq 2$
$\mathcal{C}_k :=$ The candidates of size k generated from $\mathcal{L}_{k-1}$
$\{C_k^d\} :=$ Partition $\mathcal{C}_k$ into fragments each of in a node's local memory($d = 1, \ldots, \lceil|\mathcal{C}_k|/M\rceil$)
for ($d=1$; $d \leq \lceil|\mathcal{C}_k|/M\rceil$; $d{+}{+}$) **do**
  forall $s \in \mathcal{D}^n$ **do**
    Increment the support_count of all candidates in C_k^d that are contained in s
  end
  Send the support_count of C_k^d to the coordinator node
  /* Coordinator node determine L_k^d which satisfy user-specified minimum support in C_k^d and broadcast L_1^d to all the nodes */
  Receive L_k^d from the coordinator node
end
$\mathcal{L}_k := \bigcup_d L_k^d$

図 1: NPSPM アルゴリズム

全ての候補シーケンスを単一プロセッサの主記憶に保持できる場合は，NPSPM のように処理することにより容易に並列化が可能である．しかし，データマイニングは大規模なデータベースを処理対象としているため，多くの場合にこの仮定は成り立たない．パス k において作成される全ての候補シーケンスを単一プロセッサの主記憶上に保持出来ない場合，NPSPM では候補シーケンスを主記憶に保持できる大きさに分割して主記憶に挿入し，その分割単位毎に繰り返しデータベースを検索して支持度を求める．

3.2 Hash Partitioned Sequential Pattern Mining : HPSPM

HPSPM はハッシュ関数を用いて候補シーケンスをノード間に分割して処理する．図 2 にノード p でのパス k の処理を示す．また，用いた記号の定義を表 2 に示す．

1. 候補シーケンス作成：
 $\mathcal{L}_{k-1}$から $\mathcal{C}_k$を作成する．これにハッシュ関数を適用し，対応するノードの識別子を求め，自分のノードの識別子と等しい候補シーケンス (C_k^p) を主記憶上に保持する．
2. 支持度の数え上げ：
 ローカルディスクから顧客シーケンスデータを読み出す．各顧客シーケンスデータか

表 2: 記号の定義

$\mathcal{L}_k$	Sets of all the large k-sequences.
$\mathcal{C}_k$	Sets of all the candidate k-sequences.
$\mathcal{D}^n$	Customer-sequences stored in the local disk of the n-th node
C_k^n	Sets of the candidate k-sequences assigned the n-th node
L_k^n	Sets of large k-sequences derived from C_k^n

$k \geq 2$
$\mathcal{C}_k$:=The candidates of size k generated from $\mathcal{L}_{k-1}$
$\{C_k^n\}$:= All the candidate k-itemsets, whose hashed value corresponding to the n-th node
forall $s \in \mathcal{D}^n$ **do**
  s' := Select all the items that contained in $\mathcal{C}_k$ from the customer-sequence s
  forall k-sequence $x \in s'$ **do**
    Determine the destination node ID by applying the same hash function which is used in candidate sequences partitioning, and send that k-sequences to it. If it is its own ID, increment the support-count for the sequence.
    Receive k-sequences from the other nodes and increment the support-count for that sequence
  end
end
$\{L_k^n\}$:= All the candidates in C_k^n with minimum support
Send L_k^n to the coordinator node
$/*$Coordinator node make up $\mathcal{L}_k := \bigcup_n L_k^n$ and broadcast to all the nodes $*/$
Receive $\mathcal{L}_k$ from the coordinator node

図 2: HPSPM アルゴリズム

らアイテム数 k のシーケンスを作成し，"1" と同一のハッシュ関数を適用し，ハッシュ値に対応するノードを求め，そのノードに作成したシーケンスを送信する．
同時に，受信したアイテム数 k のシーケンスから C_k^pの支持度を数え上げる．

3. 頻出シーケンス決定:
全ての顧客シーケンスデータに対する処理が終了した時点で，ノード毎に C_k^pから最小支持度を満たすものを取り出し，アイテム数 k の頻出シーケンス L_k^pを決定し，取りまとめノードに送信する．取りまとめノードは各ノードから送信された L_k^pを集め，全頻出シーケンス ($\mathcal{L}_k$) を求め，全ノードに放送する．

4 性能評価

提案した手法を IBM 社製分散メモリ型並列計算機 SP-2 上に実装し，性能測定を行った．今回の測定では 16 台のノード (RS/6000) が HPS(High Performance Switch) と呼ばれる高速ネットワークを介して接続された構成を使用した．また，各ノードにはローカルディスクが接続されている．評価には [1] で述べられた手順に従い，小売業における購買トランザクションを模倣して作成したデータセットを用いた．データセット作成に用いたパラメータを表 3 に示す．また，データセットはノード間でほぼ均等となるように分割して各ノードのローカルディスクに割り当てた．

4.1 処理時間

図 3，4 に最小支持度を変化させた場合のパス 2 での処理時間を示す． ここでは，作成される

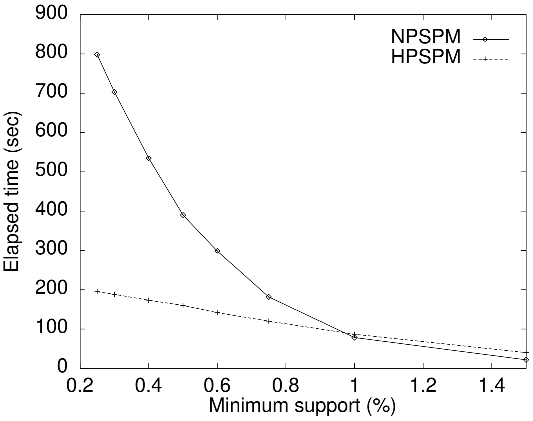

図 3: パス 2 での処理時間 (C10T5)

候補シーケンス数の多いパス 2 のみ測定した．

結果から，NPSPM では最小支持度が小さくなるに従い，処理時間が急増することがわかる．最小支持度が小さい場合，作成される候補シーケンス数が増大し，全ての候補シーケンスを単一ノードの主記憶が保持できなくなる．この場合，NPSPM は候補シーケンスを単一ノードの主記憶に入り切る大きさに分割し，その分割単位毎にデータベースを検索して処理するため，処理時間が長くなる．HPSPM は最小支持度が小さい時に NPSPM よりも処理時間が短く，最小支持度が大きい場合も NPSPM と比較的近い処理時間で実

表 3: データセットのパラメータ

Parameter	C10T5	C20T2.5
Number of customer-sequences (= size of Database)	384,000	384,000
Average number of transactions par customer-sequence	10	20
Average number of items par transaction	5	2.5
Average length of maximal potentially frequent sequences	4	4
Average size of Itemsets in maximal potentially frequent sequences	1.25	1.25
Number of maximal potentially frequent sequences	5,000	5,000
Number of maximal potentially frequent itemsets	25,000	25,000
Number of items	10,000	10,000

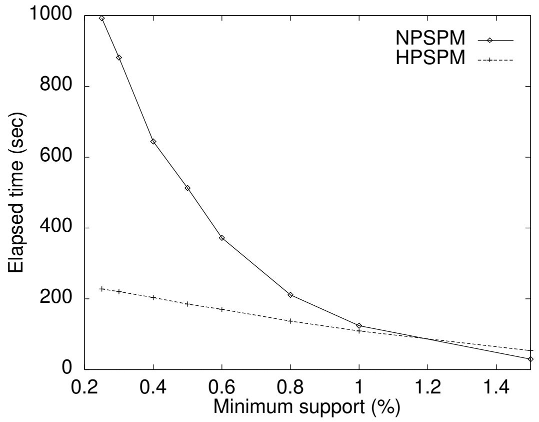

図 4: パス 2 での処理時間 (C20T2.5)

行できていることがわかる．HPSPM は候補シーケンスをノード間に分割して処理するため，候補シーケンス数の多い最小支持度が小さい場合にその効果を発揮することができる．

4.2 台数効果

図5 ,6にノード台数を変化させた場合の結果を示す．ここでは ,データセット C10T5 , C20T2.5 に対してノード台数を 4 , 6 , 8 , 12 , 16 台と変化させ，最小支持度をそれぞれ 0.75% , 0.8%とした．また ,グラフは 4 台のノードによる処理時間で正規化してあり，理想線を ideal として示してある．

結果から，NPSPM，HPSPM により良い台数効果が得られることが分かる．NPSPM では候補シーケンスの支持回数の数え上げ処理において顧客シーケンスデータの通信が不必要なため，HP-

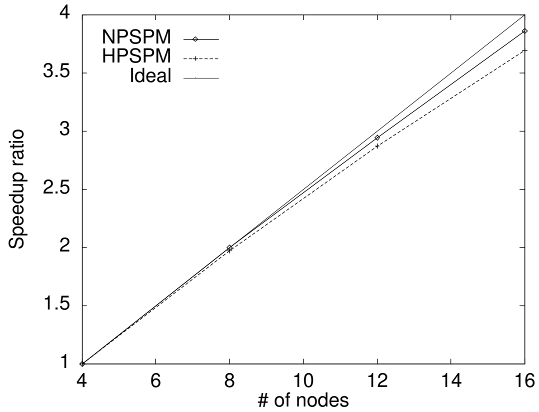

図 5: 台数効果 (C10T5)

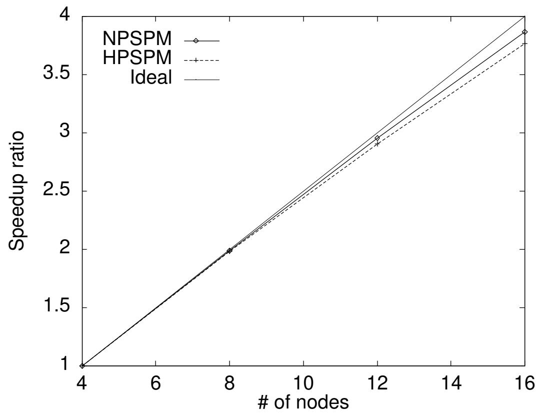

図 6: 台数効果 (C20T2.5)

SPMよりも良い台数効果が得られている．しかし，最小支持度が小さい場合には一つのパスでデータベースを繰り返し検索することがあるため，その処理性能が悪くなる．HPSPMはNPSPMと比較して少し劣るが，比較的良い台数効果を達成することが出来ていると言える．

5 まとめ

本稿では分散メモリ型並列計算機環境における時系列パターン抽出の並列処理方式を提案した．また，実際の並列計算機上に実装し，提案した並列処理方式の有効性を示した．時系列パターン抽出では，数多くの探索候補について調べ，顧客シーケンスデータベースを繰り返し検索する必要があるため，単一プロセッサによる処理では実用的な時間での処理は困難である．並列化を行う場合，候補シーケンス数が少ない場合には顧客シーケンスデータベースをノード間に分割して割り当て，ディスク入出力処理の並列化することにより効率の良い処理が可能となる．しかし，データマイニングでは大規模なデータベースを処理対象とするため，候補シーケンス数が莫大となり，単一ノードの主記憶から溢れる場合がある．その場合，候補シーケンスを全てのノードに複製して処理するNPSPMでは付加的なディスク入出力処理が必要となり，処理効率が悪くなる．効率の良い処理にはHPSPMのようにノード間に顧客シーケンスデータベースを分割すると同時に，候補シーケンスも分割して割り当てる必要がある．提案した並列処理方式を実際の並列計算機上に実装し，性能評価を行い，HPSPMにより効率の良い処理が実現できることを示した．

参考文献

[1] R.Agrawal and R.Srikant. Mining sequential pattern. In *Proceedings of the 11th International Conference on Data Engineering*, pp. 3–14, March 1995.

[2] R.Srikant and R.Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th International Conference on Extending Database Technology*, December 1996.

[3] S.-J.Yen and A.L.P.Chen. An efficient approach to discovering knowledge from large databases. In *Proceedings of the 4th International Conference on Parallel and Distributed Information Systems*, pp. 8–18, December 1996.

[4] T.Shintani and M.Kitsuregawa. Hash based parallel algorithms for mining association rules. In *Proceedings of the 4th International Conference on Parallel and Distributed Information Systems*, pp. 19–30, December 1996.