データベースシステム 109－23
（1996. 7. 25）

分類階層を考慮した相関関係の並列データマイニング

新谷 隆彦　喜連川 優

東京大学生産技術研究所

E-mail: {shintani, kitsure}@tkl.iis.u-tokyo.ac.jp

概要

データマイニングで得られる情報の代表的なものにデータ間の相関関係があり、これを抽出する効率の良い処理方式に関する研究が行われてきた。実際のデータはその特徴により分類階層化されている場合が多いが従来の研究ではこれを考慮しないものが主であった。このデータの分類階層を考慮することにより更に有用な情報抽出が期待される。

本論文では、分散メモリ型並列計算機環境上におけるデータの分類階層を考慮した相関関係抽出の並列データマイニング処理方式を報告する。データの分類階層を考慮した相関関係抽出に従来の分類階層を考慮しない並列処理方式を基にした方式を提案する。また、実際に分散メモリ型並列計算機上に実装し、それぞれの並列処理方式の性能評価を行い、良い台数効果を実現できること、データの偏りの影響を軽減することが可能であることを示す。

Parallel Mining Association Rules with Classification Hierarchy

Takahiko Shintani　Masaru Kitsuregawa

Institute of Industrial Science, University of Tokyo

Abstract

One of the most important problems in database mining is the discovery of association rules with in large databases. In most cases, classification hierarchies over the data items are available. Users are interested in generating association rules that span different levels of the classification hierarchy. In this paper, we consider the parallel algorithms for mining association rules with classification hierarchy on a shared-nothing environment. We present three parallel algorithms based on the parallel algorithms for single concept level association rules. We implemented these algorithms on a shared-nothing parallel computer. Performance evaluation show that the best algorithm, HPA-ELD, attains good linearity on speedup and is effective for skew handling.

1　はじめに

近年のプロセッサの高性能化、二次記憶装置の大容量化等の計算機技術の進展により、莫大な履歴データが蓄積され、それらを解析することが可能となった。このような大量に蓄積された履歴データを解析することにより、その中に埋もれた法則や関係など有用な情報を抽出し、データの効果的な活用を図る研究として、「データマイニング (Database Mining)」が着目されている。

データマイニングで得られる情報の代表的なものにデータ間の相関関係 (association rule) がある。この例として、「パンとバターを購入した客のうち90% が牛乳も購入している。」というルールが挙げられる。相関関係を抽出する場合、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、効率の良い並列処理方式の研究が進められてきた [1, 2, 3]。また、実際のデータはその特徴により分類階層化されている場合が多く、このデータの分類階層を考慮することにより更に詳細な情報が抽出することが可能となる。この場合には分類階層を考慮しない場合と比較してより多くのアイテムの組合せを考慮する必要が

あり、処理負荷が増大するため、効率の良い処理方式の研究が進められている [4, 5]。しかし、それらの方式は逐次処理である。実用的な時間での処理を実現するには並列化が不可欠である。

本報告では、分散メモリ型並列計算機環境におけるデータの分類階層を考慮した相関関係抽出の並列データマイニング処理方式について述べる。[2] で提案した単一階層の相関関係抽出の並列処理方式を基にした、データの分類階層を考慮した相関関係抽出の並列処理方式を示す。また、実際に分散メモリ型並列計算機上に実装し、それぞれの並列処理方式の性能評価を行い、提案した並列処理方式の有効性を明らかにする。

以下、2 章でデータの分類階層を考慮した相関関係の定義を述べる。3 章では提案する 3 つの並列処理方式 (NPA, HPA, HPA-ELD) を説明する。4 章では 3 章で示した並列処理方式を IBM 社製分散メモリ型並列計算機 SP-2 上に実装し、性能を評価する。最後に 5 章でまとめを行う。

2 分類階層を考慮した相関関係

データの分類階層構造を $\mathcal{T}$ とし、図 1に示す木構造とする。$\mathcal{T}$ の要素をアイテムと呼び、集合 $\mathcal{I}$ とする。

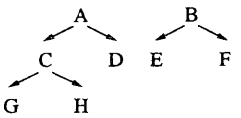

図 1: データの階層構造

$\mathcal{T}$ の辺はアイテム間の階層を示し、アイテム A から C への辺がある場合、A を C の上位アイテム ($ancestor(C)$) と呼ぶ。トランザクションデータベース $\mathcal{D}$ はトランザクション T の集合とし、$T \subseteq \mathcal{I}$ である。また、アイテムの組合せ (アイテム集合) X がトランザクション T 内のアイテムまたは T 内のアイテムの上位アイテムで構成される場合、T は X を含むと表現する。相関関係は $X \Rightarrow Y$ で表現され、$X, Y \subset \mathcal{I}$, $X \cap Y = \emptyset$ であり、Y は X の上位アイテムを含まない。相関関係は、サポート値 (support)、コンフィデンス値 (confidence) の 2 つの値を伴う。アイテム集合 X のサポート値 $sup(X)$ は $\mathcal{D}$ のうち X を含むトランザクションの割合を表す。相関関係 $X \Rightarrow Y$ のサポート値 $sup(X \Rightarrow Y)$ は X と Y を共に含むトランザクションの割合 ($sup(X \cup Y)$) となる。また、相関関係 $X \Rightarrow Y$ のコンフィデンス値は $\mathcal{D}$ の中で X を含むトランザクションのうち X と Y を共に含むトランザクションの割合を表し、$sup(X \cup Y)/sup(X)$ で定義される。ここで、Y が X の上位アイテムである相関関係 $X \Rightarrow ancestor(X)$ は $sup(X \cup ancestor(X))$ と $sup(X)$ が同値であるため、コンフィデンス値 ($sup(X \cup ancestor(X))/sup(X)$) は常に 100% となり、冗長なルールである。したがって、Y に X の上位アイテムを含む相関関係は考慮する必要がない。

相関関係抽出はユーザが定義した最小サポート値と最小コンフィデンス値を満足するすべての相関関係を見つけ出すことになる。その処理は、次の 2 段階で行われる。

1. 最小サポート値を満足するアイテム集合をすべて見つけ出す。これらのアイテム集合を頻出アイテム集合 (large itemset) と呼ぶ。
2. 第 1 段階で求めた頻出アイテム集合を用いて最小コンフィデンス値を満足する相関関係を導き出す。

相関関係を抽出する 2 段階の処理のうち、第 2 段階の相関関係を導出する処理では第 1 段階で求めたすべての頻出アイテム集合に対して順次調べ上げる処理である。しかし、第 1 段階のすべての頻出アイテム集合を見つけ出す処理ではデータベースを繰り返し検索する処理を必要とするため、アイテムの数やトランザクション量が多い場合に負荷の高い処理となる。データの分類階層を考慮する場合は考慮しないときと比較してより多くのアイテムの組合せを調べる必要があるため、効率の良い処理方式の研究が進められている [4]。ここでは、基本逐次処理方式として [4] で提案された Basic アルゴリズムと Cumulate アルゴリズムを示す。後に説明する並列処理方式は Cumulate アルゴリズムを基にしている。

Basic アルゴリズムは [6] で提案された Apriori アルゴリズムとほぼ同様である。はじめに、データベースの各トランザクションにそれに含まれるアイテムのすべての上位アイテムを付加し、データベースを再編成しておく。以降の処理ではこの再編成したデータベースを用いる。頻出アイテム集合を求める処理は、まず、データベースを検索し、それぞれのアイテムのトランザクションに含まれる回数 (生起回数) を数え上げ、最小サポート値を満足するアイテムを取り出して長さ 1 の頻出アイテム集合 (large 1-itemset) とする。ここで、アイテム集合に含まれるアイテム数を長さと表現する。次に、長さ 1 の頻出アイテム集合から 2 つのアイテムの組合せを作成する。これらを長さ 2 の候補アイテム集合 (candidate 2-itemset) と呼ぶ。再びデータベースを検索して生起回数を数え上げることによりサポート値を求め、長さ 2 の頻出アイテム集合を決定する。以降、長さ $k(\geq 2)$ の頻出アイテム集合を求める処理は次のようになる。

1. 長さ $(k-1)$ の頻出アイテム集合から、長さ k の候補アイテム集合を作成する。

2. トランザクションデータベースを検索し、サポート値を求める。

3. 最小サポート値を満足するものを取り出し、長さ k の頻出アイテム集合とする。

上記処理を長さ k のパスと呼ぶ。この処理は頻出アイテム集合が空になった時点で終了する。

しかし、相関関係のうち分類階層の上下関係であるアイテム (アイテム x とその上位アイテム $ancestor(x)$) を共に含むルールは冗長であるため、そのようなアイテムの組合せを含む頻出アイテム集合を求める必要がない。そこで [4] では Cumulate アルゴリズムとして、以下のような効率化を行っている。

- トランザクションに上位アイテムを付加するとき、どの候補アイテム集合にも含まれていないアイテムを除去する。それぞれのアイテムについては下位階層になるに従い、トランザクションに含まれる回数 (生起回数) が少なくなる。したがって、どの候補アイテム集合にも含まれないアイテムはその上位アイテムと置き換えることで処理することができる。
- アイテム集合のうち、分類階層の上下関係であるアイテムの組合せを含むものを除去する。

3 並列処理方式

本節では Cumulate アルゴリズムを基にした、分類階層を考慮した相関関係抽出の並列処理方式 (NPA, HPA, HPA-ELD) について述べる。

単一ノードの主記憶上にすべての候補アイテム集合を保持できる場合、並列化は容易であるが、データの分類階層を考慮した場合には更に多くの候補アイテム集合について調べなければならないため、多くの場合にこの仮定は成立しない。本稿では、すべての候補アイテム集合が単一ノードの主記憶上に入り切らない場合について説明する。また簡単のため、候補アイテム集合はすべてのノードの主記憶の総量より小さいとする。主記憶の総量よりも大きい場合の拡張は容易に行うことが可能である。

3.1 Non Partitioned Apriori : NPA

NPA では候補アイテム集合を全ノードに複製してサポート値を調べる。図 2にノード p での長さ k の頻出アイテム集合を求める処理 (長さ k のパス) を示す。また、用いた記号の定義を表 1に示す。

各ノードは以下の処理を行う。

1. 長さ $(k-1)$ の頻出アイテム集合を用いて、長さ k の候補アイテム集合を作成する。ここで、分類階層の上下関係となるアイテムの組合せを含むものを削除し、残った候補アイテム集合を主記憶上のハッシュ表に挿入する。

$\mathcal{L}_k$	Set of all the large k-itemsets.
$\mathcal{C}_k$	Set of all the candidate k-itemsets.
$\|\mathcal{C}_k\|$	The size of $\mathcal{C}_k$ in bytes.
M	The size of main memory in bytes.
$\mathcal{D}^p$	Transactions stored in the local disk of the p-th processor
C_k^d	Sets of fragment of candidate k-itemsets. Each fragment fits in the local memory of a processor. ($d = 1, \ldots, \lceil \|\mathcal{C}_k\|/M \rceil$, $\mathcal{C}_k = \bigcup_{d=1}^{\lceil \|\mathcal{C}_k\|/M \rceil} C_k^d$)
$\|C_k^d\|$	The size of C_k^d in bytes.
L_k^d	Sets of large k-itemsets derived from C_k^d.

表 1: 記号の定義

2. ローカルディスクからトランザクションデータベースを読み出す。各トランザクションに上位アイテムを付加する。ここで、どの候補アイテム集合にも含まれていないアイテムを除去する。k 個のアイテムの組合せを作成し、ハッシュ表を検索する。ここで、アイテムとその上位アイテムを共に含む組合せについては調べない。対応する候補アイテム集合が存在する場合、その生起回数を 1 増加する。

3. すべてのトランザクション対する処理が終了した時点で、それぞれの候補アイテム集合の全ノードでの生起回数の総和を求め、頻出アイテム集合を決定する。

すべての候補アイテム集合を単一ノードの主記憶内に保持できない場合、候補アイテム集合を分割して主記憶内のハッシュ表に挿入し、繰り返しデータベースを検索してサポート値を求める。NPA は単純であるが、候補アイテム集合の数が多い場合、データベースを検索する回数が多くなる。

3.2 Hash Partitioned Apriori : HPA

HPA では候補アイテム集合をハッシュ関数を用いてノード間に分割する。図 3にノード p での長さ k のパスを示す。また、用いた記号の定義を表 2に示す。

1. 長さ $(k-1)$ の頻出アイテム集合を用いて、長さ k の候補アイテム集合を作成する。ここで、分類階層の上下関係となるアイテムの組合せを含むものを削除し、残った候補アイテム集合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、自分の識別子と等しい場合に、主記憶上のハッシュ表に挿入する。

$k \geq 2$

C_k:= The candidates of size k generated from L_{k-1}

Delete the candidates that contains an items and its ancestor.

$\{C_k^d\}$:= Partition C_k into fragments each of which fits in a processor's local memory (d=1, ..., $\lceil |C_k|/M \rceil$)

for (d=1; $d \leq \lceil |C_k|/M \rceil$; d++) **do**

  forall $t \in \mathcal{D}^p$ **do**

    $t\prime$:=Add all ancestors of item $x(\in t)$ that are present in the candidates in C_k

    Increment the support_count of all candidates in C_k^d that are contained in $t\prime$

  end

  Send the support_count of C_k^d for to the coordinator

  /* Coordinator determine L_k^d which satisfy user-specified minimum support in C_k^d and broadcast L_1^d to all processors */

  Receive L_k^d from the coordinator

end

$\mathcal{L}_k := \bigcup_d L_k^d$

図 2: NPA algorithm

$\mathcal{L}_k$	Set of all the large k-itemsets.
$\mathcal{C}_k$	Set of all the candidate k-itemsets.
$\mathcal{D}^p$	Transactions stored in the local disk of the p-th processor.
C_k^p	Sets of candidate k-itemsets assigned the p-th processor. (N means the number of processors) ($\mathcal{C}_k = \bigcup_{p=1}^{N} C_k^p$)
L_k^p	Sets of large k-itemsets derived from C_k^p

表 2: 記号の定義

2. ローカルディスクからトランザクションデータベースを読み出す。各トランザクションに上位アイテムを付加する。どの候補アイテム集合にも含まれなていないアイテムを削除したものから k 個のアイテムの組合せを作成し、"1" と同一のハッシュ関数を適用する。ここで、アイテムとその上位アイテムを共に含む組合せについては調べない。ハッシュ値に対応するノードの識別子を求め、そのノードにアイテムの組合せを送信する。他のノードから送信されたメッセージに対して、ハッシュ表を検索し、対応する候補アイテム集合の生起回数を 1 増加させる。

3. すべてのトランザクションに対する処理が終了した時点で、ノード毎に頻出アイテム集合を決定し、他のすべてのノードに放送する。これにより、すべてのノードが長さ k の頻出アイテム集合を持つことになる。ここで、頻出アイテム集合の数は、候補アイテム集合と比較して非常に少ないため、この様な処理が可能となる。

$k \geq 2$

C_k:=The candidates of size k generated from L_{k-1}

Delete the candidates that contains an items and its ancestor.

$\{C_k^p\}$:=All the candidate k-itemsets, whose hashed value corresponding to the p-th processor

forall $t \in \mathcal{D}^p$ **do**

  $t\prime$:=Add all ancestors of $x(\in t)$ that are present in the candidates in C_k

  forall k-itemset $x \in t\prime$ **do**

    Determine the destination processor id by applying the same hash function which is used in item partitioning, and send that k-itemset to it. If it is its own id, increment the support_count for the itemset.

    Receive k-itemset from the other processors and increment the support_count for that itemset

  end

end

$\{L_k^p\}$:=All the candidates in C_k^p with minimum support

Send L_k^p to the coordinator

/* Coordinator make up $\mathcal{L}_k := \bigcup_p L_k^p$ and broadcast to all the processors */

Receive $\mathcal{L}_k$ from the coordinator

図 3: HPA algorithm

3.3 HPA with Extremely Large Itemset Decomposition : HPA-ELD

トランザクションデータに偏りがある場合、つまり、極端に多くのトランザクションに含まれるアイテム集合が存在する場合、そのアイテム集合を割り当てられたノードに他のノードからのメッセージが集中するため、処理の偏りが生じる。データの分類階層を考慮した場合、階層の上位になるに従い、トランザクションに含まれる回数が増大するため、データの偏りが大きくなる。HPA-ELD では、この影響を減少する方法をとる。

HPA-ELD では生起回数が極端に大きいと予想される候補アイテム集合をすべてのノードに複製し、ノード毎に生起回数を調べる。すべてのトランザクションデータに対する処理が終了した時点で、全ノードでの

生起回数の総和を求め、サポート値を求める。長さ $(k-1)$ の頻出アイテム集合から長さ k の候補アイテム集合を作成するとき、元にした頻出アイテム集合のサポート値の和が閾値以上の場合、NPA と同様にすべてのノードに複製して割り当てる。他の候補アイテム集合は HPA と同様に、ハッシュ関数を適用して分割する。

4 性能評価

前節で述べた並列アルゴリズムを 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。本性能測定では 16 台のノード (RS/6000) が HPS (ハイパフォーマンス・スイッチ) 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 2GB のローカルディスクが接続されている。

3 つの並列処理方式の性能を評価には、[4] で述べられた手順を基に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。各パラメータを表 3に示す。

Parameter	Value
Number of transactions	1600000
Average size of the transactions	12
Average size of the maximal potentially large itemsets	4
Number of maximal potentially large itemsets	10000
Number if items	100000
Number of roots	250
Number of levels	4-5
Fanout	5

表 3: データセットのパラメータ

4.1 並列処理方式の処理時間

図 4に各並列処理方式の最小サポート値を変化させた場合の長さ 2 のパスの処理時間の変化を示す。以下の実験では、最も候補アイテム集合数の多い長さ 2 のパスについてのみ測定した。また、トランザクションデータファイルはノード間でほぼ均等に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。

NPA では最小サポート値が小さくなるに従い、処理時間が急増する。最小サポート値が小さくなると候補アイテム集合数が増大し、すべての候補アイテム集合を単一ノードの主記憶上に保持できなくなる。この場合、NPA は候補アイテム集合を分割し、トランザクションデータベースを繰り返し検索して処理するため、多くのディスク入出力処理が必要となり、処理時間が非常に長くなる。また、HPA と HPA-ELD では候補

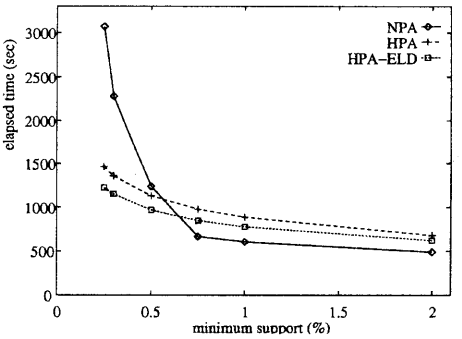

図 4: 並列処理方式の処理時間

アイテム集合を分割することによりシステム全体の主記憶を効果的に活用できるため、NPA と比較してディスク入出力コストが少ない。しかし、アイテムの組合せの通信を必要とするため、候補アイテム集合数が少ない場合、NPA よりも処理性能が悪くなる。HPA-ELD は一部の候補アイテム集合を全ノードに複製して NPA と同様に処理するため、これらに対応するアイテムの組合せを送信する必要がない。

4.2 HPA と HPA-ELD の比較

図 5に HPA と HPA-ELD の長さ 2 のパスにおける各ノードの候補アイテム集合のハッシュ表の検索回数を示す。ここでは、ノード数を 16 台、最小サポート値を 0.5% とした。

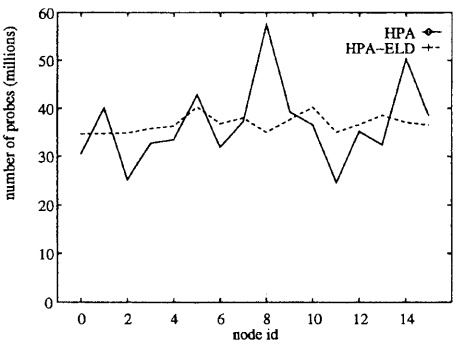

図 5: ノード毎の検索回数

HPA ではノード間の検索回数の顕著な偏りが見られる。HPA では候補アイテム集合を 1 つのノードに割り当て、他のノードから送信されるアイテムの組合せからその候補アイテム集合のサポート値を調べるため、多くのトランザクションに含まれる候補アイテム集合が割り当てられたノードにメッセージが集中する。特にデータの分類階層を考慮する場合、階層が上位になるに従い、トランザクションに含まれる回数が増大するため、ノード間の負荷の偏りが生じやすくなる。

一方、HPA-ELD ではノード間の検索回数の大きな差が見られない。図4に見られる処理時間の差はこのノード間の負荷の偏りによるものである。以上から、HPA-ELD の一部の候補アイテム集合をすべてのノードに複製して処理することにより、ノード間の処理負荷の偏りを回避することが実現できることがわかる。

4.3 台数効果

図6にノード数を変化させた場合の結果を示す。ここでは最小サポート値を 0.5% とし、すべてのノードのトランザクション量を一定とした。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る。

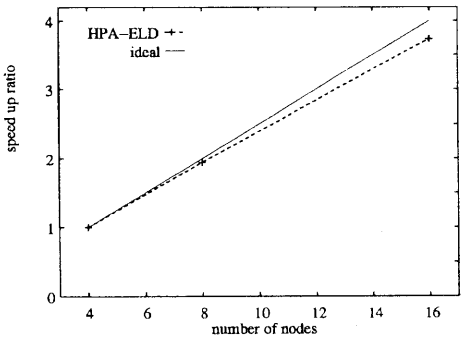

図 6: 台数効果

図6から HPA-ELD は良い台数効果を実現していることがわかる。HPA では候補アイテム集合のサポート値を調べる処理においてアイテムの組合せを通信する必要があり、図5に見られる処理負荷の偏りが生じる場合がある。HPA-ELD では一部の候補アイテム集合をすべてのノードに複製し、NPA と同様に個々のノードで生起回数を数え上げるため、トランザクションから作成されたアイテムの組合せのうち、それらに対応するものを通信しない。すべてのノードに複製して処理される候補アイテム集合は、多くのトランザクションに含まれるものを選択しているため、これらが引き起こすデータ通信量は多くなる。HPA-ELD ではこれらの通信を行わない。

5 まとめ

本論文では分散メモリ型並列計算機環境におけるデータの分類階層を考慮した相関関係抽出の並列処理方式を提案し、実際に分散メモリ型並列計算機上に実装することにより、提案した並列処理方式の性能評価を行った。相関関係の抽出ではトランザクションデータベースを繰り返し検索する必要があり、データの分類階層を考慮した場合には考慮しない場合と比較してより多くのアイテムの組合せを考慮する必要があり、単一ノード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、候補アイテム集合数が少ない場合にはトランザクションデータをノード間に分割して割り当て、ディスク入出力処理を並列化することで実現できる。しかし、データの分類階層を考慮する場合は非常に多くの候補アイテム集合について調べる必要があるため、単一ノードがすべての候補アイテム集合を保持できない場合が生じる。NPA のように候補アイテム集合をすべてのノードに複製して処理する場合、トランザクションデータベースを検索する回数が増大し、処理効率が悪くなる。効率の良い処理には HPA、HPA-ELD のように候補アイテム集合を複数のノード間に分割する必要がある。

提案した並列処理方式を実際に分散メモリ型並列計算機上に実装し、性能評価を行い、良い性能向上性が得られることを示した。本論文では IBM 社製分散メモリ型並列計算機 SP-2 の 16 台のノードにそれぞれがディスクを持つシステムを利用した。候補アイテム集合数、データベースの検索回数は処理対象とするデータに依存しており、実際の POS データではなく、小売業における購買トランザクションデータを模倣して作成したデータセットを用いた。これらのデータセットはデータ間の偏りを含んでおり、HPA-ELD ではこのようにデータの偏りを含むデータに対する処理に有効であることを示した。

参考文献

[1] J.S.Park, M.S.Chen, and P.S.Yu: "Efficient Parallel Data Mining for Association Rules", In *Proc. of the 4th CIKM Conference*, pp.31-36, Nov 1995.

[2] 新谷, 喜連川: "データマイニングの並列化に関する一考察", 電子情報通信学会コンピュータシステム研究会 (CPSY95-88), 信学技報 Vol.95 No.47, pp.57-62, Dec 1995.

[3] 新谷, 喜連川: "データマイニングにおける相関関係抽出の並列処理方式の実装とその評価", 並列処理シンポジウム *JSPP'96*, 並列処理シンポジウム論文集 Vol.96No.3, pp.97-104, June 1996.

[4] R.Srikant, R.Agrawal: "Mining Generalized Association Rules", In *Proc. of the 21th VLDB Conference*, pp.407-419, Sept 1995.

[5] J.Han, Y.Fu: "Discovery of Multi-Level Association Rules from Large Databases", In *Proc. of the 21th VLDB Conference*, pp420-431, Sept 1995.

[6] R.Agrawal, and R.Srikant: "Fast Algorithms for Mining Association Rules", In *Proc. of the 20th VLDB Conference*, pp.487-499, Sept 1994.