分類階層を考慮した相関ルール抽出の並列処理方式における負荷制御手法

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

あらまし

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ルールがあり、その抽出処理方式に関する研究が行われてきた。従来は単に個々のデータ間のみを考慮したものが中心であったが、実際のデータはその特徴により分類階層化されており、これを考慮することにより更に有用な情報の抽出が可能となる。我々はデータの分類階層を考慮した相関ルール抽出の並列処理方式を提案してきたが、従来の方式では負荷分散の制御を行っていないため、処理負荷の偏りが大きかった。

本稿では、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ルール抽出において、従来の並列処理方式の問題点である処理負荷の偏りを低減させる手法を提案する。また、実際に分散メモリ型並列計算機上に実装し、提案する手法の性能評価を行う。

キーワード :　データマイニング、相関ルール、分類階層、並列アルゴリズム、負荷分散

Load Balancing for Parallel Mining Association Rules with Classification Hierarchy

Takahiko SHINTANI　Masaru KITSUREGAWA

Institute of Industrial Science, University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo 106, JAPAN.

Abstract

One of the most important problem in database mining is discovery of association rules in large database. In most cases, the classification hierarchy over the data is available. Users are interested in generating association rules that span different levels of the classification hierarchy. We have proposed parallel algorithms for association rules with classification hierarchy, named H-HPA. In H-HPA, it is generally difficult to achive a flat workload distribution.

In this paper, we present load balancing algorithms for parallel mining association rules with classification hierarchy. We implemented these algorithms on a shared-nothing environment and analyzed the performance of our algorithms.

Key Words :　Data mining, Association rule, Classification hierarchy, Parallel algorithm, Load balancing

1　はじめに

近年、大量に蓄積された履歴データを解析し、その中に埋もれた法則や関係など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代表的なものに相関ルール (association rule) があり、我々はその抽出処理性能向上を目的とした並列処理方式の研究を進めてきた [1]。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、これ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であり、その抽出処理方式に関する研究が進められてきた [2, 3]。しかし、それらの方式は逐次処理である。分類階層を考慮する場合にはより多くのアイテムの組合せを調べる必要があるため、処理負荷が増大し、実用的な時間での処理には並列化が不可欠である。我々は分散メモリ型並列計算機環境を基本的なモデルとした分類階層を考慮した相関ルール抽出の並列処理方式を提案してきた [4, 5]。[4, 5] で提案した並列処理方式では、データベースの分割によるディスク入出力処理の並列化だけでなく、候補アイテム集合もノード間に分割して割り当てることにより、システム全体の記憶空間を効果的に利用する。また、分類階層を考慮して候補アイテム集合を分割することにより、ノード間の通信量を削減している。しかし、それらの方式ではノード間の負荷バランスの制御を行っていないため、処理負荷の偏りが大きかった。

本報告では、分類階層を考慮した相関ルール抽出の並列処理方式における負荷制御手法について述べる。本手法では著しく多くのトランザクションに含まれるアイテムで構成される候補アイテム集合を見つけ出し、それらを全てのノードに複製して、ノード毎に個々に処理することにより、ノード間の負荷分散の制御を行う。また、本手法を実際の分散メモリ型並列計算機上に実装し、性能評価を行い、その有効性を明確化する。

2　分類階層を考慮した相関ルール

データの分類階層構造を $\mathcal{T}$ とする。図 1 に示されるように $\mathcal{T}$ は木構造を有し、その要素をアイテム $\mathcal{I}$ と呼ぶ。 $\mathcal{T}$ の辺はアイテム間の階層を示し、アイテム x から y への矢がある場合、x を y の上位アイテム ($ancestor(\mathrm{y})$) と呼ぶ。トランザクションデータベースを $\mathcal{D} = \{t_1, t_2, \ldots, t_n\}(t_i \subseteq \mathcal{I})$ とし、各要素 t_iをトランザクションと呼ぶ。長さ k のアイテム集合 (itemset) とは k 個のアイテムの組合せを指す。アイテム集合 X の支持度 $support(X)$ は $\mathcal{D}$ 全体に対して X を含むトランザクションの割合を表す。また、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。相関ルールは $X \Rightarrow Y$ で表現される。ここで、$X, Y \subset \mathcal{I},\ X \cap Y = \emptyset$、又、$Y$ は X の上位アイテムを含まない。相関ルールは支持度 (support)、確信度 (confidence) の 2 つのパラメータを有し、これらの値により相関ルールの有意性を示す。ここで、Y が X の上位アイテムである相関ルール $X \Rightarrow ancestor(X)$ の確信度は常に 100%であり、冗長なルールとなる。相関ルール $X \Rightarrow Y$ の支持度 $support(X \Rightarrow Y)$ は $\mathcal{D}$ 全体に対し X と Y を共に含むトランザクションの割合 $support(X \cup Y)$ により、又、確信度 $confidence(X \Rightarrow Y)$ 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、すなわち $support(X \cup Y)/support(X)$ により定義される。

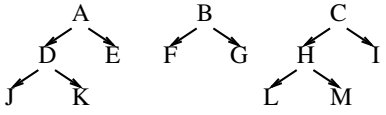

図 1: データの階層構造

相関ルールの抽出問題はユーザにより指定された最小支持度 (minimum support) と最小確信度 (minimum confidence) を満足する全てのルールを見出すことに相当する。

相関ルールは次の 2 つのステップで抽出される。

1. 最小支持度を満たすアイテム集合を全て見出す。これらのアイテム集合をラージアイテム集合と呼ぶ。
2. 第 1 ステップで求めたラージアイテム集合から、最小確信度を満たす相関ルールを導出する。

相関ルール抽出処理のうち、第 2 ステップは第 1 ステップで求めたラージアイテム集合に対する処理であるため、その負荷は軽い。しかし、第 1 ステップはトランザクションデータベースを繰り返し検索し、数多くのアイテム集合を処理するため、処理時間の大半を占めることになる。相関ルール抽出アルゴリズムに関する研究は第 1 ステップの効率化が中心となっている。

ここで、基本逐次処理方式として [2] で提案された Cumulate アルゴリズムを示す。後に説明する並列処理方式はこれを元にしている。はじめにデータベー

スを検索し、各アイテムの支持度を求め、最小支持度を満たすアイテムを取り出す。これらを長さ 1 のラージアイテム集合 (L_1) と呼ぶ。次に、L_1を用いて 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 (C_2) と呼ぶ。ここで、C_2のうち分類階層の上下関係にあるアイテムを含むものを除去する。分類階層の上下関係にあるアイテムを共に含む相関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ラージアイテム集合を求める必要はない。更に、C_2のどの候補アイテム集合にも含まれていないアイテムを $\mathcal{T}$ から除去しておく。その後、データベースを検索し、C_2の支持度を求め、最小支持度を満たすものを取り出し、長さ 2 のラージアイテム集合 (L_2) とする。以降、長さ k のラージアイテム集合 (L_k) を求める処理は次のようになる。

1. L_{k-1}から C_kを作成する。C_kのどの候補アイテム集合にも含まれないアイテムを $\mathcal{T}$ から除去する。
2. データベースを検索し、C_kの支持度を求める。
3. C_kのうち最小支持度を満たすものを取り出し、L_kとする。

上記の L_kを求める処理をパス k と呼ぶ。この処理は新たなラージアイテム集合が空となるまで続けられる。

3 並列処理方式

本節では [4, 5] で提案した分散メモリ型並列計算機環境を基本的なモデルとした、データの分類階層を考慮した相関ルール抽出の並列処理方式 (NPA、HPA、H-HPA) について述べる。

単一ノードの主記憶上に全ての候補アイテム集合が保持できる場合、並列化は容易であるが、分類階層を考慮する場合には更に多くの候補アイテム集合を調べる必要があるため、多くの場合にこの仮定は成立しない。本報告では、全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い場合について考察を進める。また、以下に述べる全ての手法においてトランザクションデータベースは分割され、各ノードのローカルディスクに割当てられている。

3.1 None Partitioned Apriori : NPA

NPA では候補アイテム集合を全てのノードに複製して処理する。以下に各ノードのパス k の処理を示す。

1. L_{k-1}を用いて C_kを作成し、分類階層の上下関係にあ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。ここで、C_kのどの候補アイテム集合にも含まれないアイテムを $\mathcal{T}$ から削除する。
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し、C_kの支持回数を数え上げる。
3. 全てのトランザクション対する処理が終了した時点で、各々の候補アイテム集合の全ノードでの支持回数の総和から支持度を求め、最小支持度を満たすものを L_kとする。

全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支持度を求める。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。これにより、システム全体の記憶空間を効果的に活用できる。以下に各ノードのパス k での処理を示す。

1. NPA と同様にして作成した C_kの各候補アイテム集合に対して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。更に、NPA と同様にして不要なアイテムを $\mathcal{T}$ から削除する。
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し、各トランザクションにそのトランザクションが含むアイテムの上位アイテムのうち $\mathcal{T}$ に存在するものを付加する。これらから 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し、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。ここで、アイテムとその上位アイテムを共に含む組合せは調べない。
   他のノードから送信されたアイテムの組合せに対応する候補アイテム集合の支持回数を 1 増加させる。
3. 全てのトランザクションに対する処理が終了した時点で、ノード毎にラージアイテム集合を決定し、他のノードに放送する。

HPA では、候補アイテム集合に単純にハッシュ関数を用いて分割するため、トランザクションデータに上位アイテムを付加したものから作成したアイテムの組合せを送信する必要があり、大量の通信が引き起こされる。

3.3 Hierarchical HPA : H-HPA

H-HPA ではデータの分類階層を考慮し、上下関係にある候補アイテム集合が同一のノードに割り当てられるように候補アイテム集合を分割する。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 1 に示す。

$\mathcal{L}_k$	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 node.
C_k^p	Sets of candidate k-itemsets whose hash value calculated with thier root item is corresponding to p-th node.
L_k^p	Sets of large k-itemsets derived from C_k^p.

表 1: 記号の定義

(1) $k \geq 2$
(2) $\mathcal{C}_k$:=The candidates of size k generated from L_{k-1}
(3) **if**($k = 2$)**then**
(4) Delete the candidates that contains an item and its ancestor
(5) $\{C_k^p\}$:=All the candidate k-itemsets, whose hash value calculated with its root items corresponding to the p–th node
(6) **forall** $t \in \mathcal{D}^p$ **do**
(7) t':=Replace items in t with the lowest large item within its ancestors, if there are small items in t
(8) **foreach** n–th node **do**
(9) t'' := Select all items whose root items are allocated to n–th node
(10) **if** (n = own node ID) **then**
(11) Generate k–itemset from t'', and increment the support_count for the itemset and all its ancestor candidates
(12) **else**
(13) Send t'' to n–th node
(14) Receive items from the other nodes
(15) Generate k-itemset from receive items, and increment the support_count for the itemset and all its ancestor candidates
(16) **end**
(17) **end**
(18) $\{L_k^p\}$:=All the candidates in C_k^p with minimum support
(19) Send L_k^p to the coordinator node
(20) /* Coordinator node make up $\mathcal{L}_k := \bigcup_p L_k^p$ and broadcast to all the nodes */
(21) Receive $\mathcal{L}_k$ from the coordinator node

図 2: H-HPA アルゴリズム

1. 候補アイテム集合の作成 (2) ~ (5)
   NPA と同様にして作成した C_kの各候補アイテム集合に対して、アイテムをその最上位アイテムに置き換えたものにハッシュ関数を適用し、ハッシュ値に対応するノードの識別子を求め、この識別子が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。他の候補アイテム集合はその最上位候補アイテム集合のハッシュ値に対応するノードに割り当てる。更に、NPA と同様に不要なアイテムを $\mathcal{T}$ から削除する。
2. 支持回数の数え上げ (6) ~ (17)
   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し、各トランザクションに含まれるアイテムを最下位のラージアイテムに置き換える。それらから k 個のアイテムの組合せを作成し、“1” と同様に最上位アイテムに置き換えたもの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る。各ノードに対応する最下位アイテムを送信する。
   他のノードから送信されたアイテム集合から k 個のアイテムの組合せを作成し、対応する頻出アイテム集合とその全ての上位候補アイテム集合の支持回数を 1 増加させる。
3. ラージアイテム集合の決定 (18) ~ (21)
   HPA と同様

H-HPA では分類階層の上下関係にある候補アイテム集合を同一のノードに割り当てるため、最下位アイテムのみの通信で処理が可能である。これにより、HPA と比較して通信負荷を削減することが可能となる。

4 負荷制御手法

H-HPA は木の組合せ単位で候補アイテム集合をノード間に割り当てることになるため、均等な負荷バランスを実現することが困難である。また、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、そのアイテム集合が割り当てられたノードに負荷が集中し、処理の偏りが生じる。

本節では、ノード間の処理負荷の偏りを軽減させる 2 つの手法を述べる。本手法では、支持回数が大きいアイテムで構成される候補アイテム集合を全てのノードに複製し、NPA と同様に処理する。他の候補アイテム集合は H-HPA と同様に分類階層を考慮してハッシュ分割して処理する。

4.1 H-HPA with Tree Grain Dupulicate : H-HPA-TGD

H-HPA-TGD では木の頂点のアイテムの支持度が高い木の組合せを見つけ出し、それらを全てのノー

ドに複製することにより、木の組合せ単位での複製を行う。ノード p でのパス k の処理は図 2 の H-HPA に図 3 に示す変更を加えたものとなる。また、記号の意味を表 2 に示す。

C_k^p	Sets of candidate k-itemsets whose hash value calculated with their root item is corresponding to p-th node.
L_k^p	Sets of large k-itemsets derived from C_k^p
C_k^D	Sets of candidate k-itemsets which are copied all the nodes.
L_k^D	Sets of large k-itemsets derived from C_k^D

表 2: 記号の意味

(5.0) Count the number of descendant candidate for each root k-itemset and the number of candidates allocated for each node by generating the k-itemsets using $\mathcal{L}_k$.
(5.1) Generate k-itemsets from root items.
(5.2) Sort the root itemsets based in their frequency of appearance.
(5.3) $\{C_k^D\}$:=All the root k-itemsets whose frequency is high so as to use the memory space fully.
(5.4) $\{C_k^D\}$:=All the candidate k-itemsets whose root itemset is contained in C_k^D.
(5.5) $\{C_k^p\}$:= All the candidate k–itemsets, whose hash value calculated with its root items corresponding to the p–th node

(7.1) Increment the support_count of all candidates in C_k^D that are contained in t

(20.0) Send the support_count of C_k^D to the coordinator
(20.1) /* Coordinator determine L_k^D which satisfy user-specified minimum support in C_k^D */
(20.2) /* Coordinator node make up $\mathcal{L}_k := L_k^D + \bigcup_p L_k^p$ and broadcast to all the nodes */

図 3: H-HPA-TGD アルゴリズム

1. 候補アイテム集合の作成 [図 2(5) を図 3(5.0) ~ (5.5) に変更]
   最上位アイテムを取り出し、k 個のアイテムの組合せ (最上位候補アイテム集合) を作成する。ここでは、同一のアイテムからなる組合せも作成しておく。NPA と同様にして $\mathcal{C}_k$ を作成し、各ノードに割り当てられる候補アイテム数と各最上位候補アイテム集合に属する候補アイテム集合数を数え上げる。最上位候補アイテム集合をそれを構成するアイテムの支持度がの和でソートする。各ノードの主記憶に空きがなくなるまで、最上位候補アイテム集合とその下位候補アイテム集合を複製する。全ノードの複製された候補アイテム集合を C_k^D とする。残りの候補アイテム集合を H-HPA と同様にしてハッシュ分割する。ノード p にのみ割り当てられた候補アイテム集合を C_k^p とする。
2. 支持回数の数え上げ [図 2(7) の次に図 3(7.1) を付加]
   ローカルディスクから読み出した各トランザクションに対して、C_k^D の支持回数を数え上げる。C_k^p に対しては H-HPA と同様。
3. ラージアイテム集合の決定 [図 2(20) を図 3(20.0) ~ (20.2) に変更]
   全てのトランザクションに対する処理が終了した時点で、C_k^D の各々の候補アイテム集合の全ノードでの支持回数の総和から支持度を求め、最小支持度を満たすものを L_k^D とする。L_k^p に関しては H-HPA と同様。

H-HPA-TGD では木の組合せ単位で候補アイテム集合を全ノードに複製するため、複製するために十分な記憶空間の空きが無い場合、H-HPA と等しくなる。また、複製された塊には支持度の小さいアイテムからなる候補アイテム集合が含まれる場合もあり、十分な効果が得られない。

4.2 H-HPA with Path Grain Dupulicate : H-HPA-PGD

H-HPA-PGD では、支持度の高い最下位アイテムで構成される候補アイテム集合とその全ての上位アイテムを全ノードに複製して処理する。これにより、効果的な負荷制御が実現できる。また、記憶空間に空きが少ない場合にも効果が期待できる。ノード p でのパス k の処理は図 2 の H-HPA に図 4 に示す変更を加える。

(5.0) Count the number of candidates allocated for each node.
(5.1) Sort the lowest large items based on their frequency of appearance.
(5.2) Generate k-itemsets from the lowest items.
(5.3) $\{C_k^D\}$:=All the lowest candidate k-itemsets whose frequency is high and their ancestor candidates so as to use the memory space fully.
(5.4) Delete the candidates in C_k^D from $\mathcal{C}_k$.

(7.1) Increment the support_count of all candidates in C_k^D that are contained in t.

(20.0) Send the support_count of C_k^D to the coordinator
(20.1) /* Coordinator determine L_k^D which satisfy user-specified minimum support in C_k^D */
(20.2) /* Coordinator node make up $\mathcal{L}_k := L_k^D + \bigcup_p L_k^p$ and broadcast to all the nodes */

図 4: H-HPA-PGD アルゴリズム

1. 候補アイテム集合の作成 [図 2(5) を図 4(5.0) ~ (5.4) に変更]
   NPA と同様にして $\mathcal{C}_k$ を作成し、各ノードに割り当てられる候補アイテム数を数え上げる。最下

位アイテムを支持度の順にソートし、上から順に k 個のアイテムの組合せ (最下位候補アイテム集合) を作成し、各ノードの主記憶に空きがなくなるまで、最下位候補アイテム集合とその全上位候補アイテム集合を複製する。全ノードの複製された候補アイテム集合を C_k^D とする。残りの候補アイテム集合を H-HPA と同様にしてハッシュ分割する。ノード p にのみ割り当てられた候補アイテム集合を C_k^p とする。

2. 支持回数の数え上げ [図 2(7) の次に図 4(7.1) を付加]
   H-HPA-TGD と同様
3. ラージアイテム集合の決定 [図 2(20) を図 4(20.0) ~ (20.2) に変更]
   H-HPA-TGD と同様

5 性能評価

3 節、4 節で述べた並列処理方式を 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、各ノードにはローカルディスクが接続されている。評価には、[2] で述べられた手順を元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	3200000
Average size of the transactions	10
Number if items	30000
Number of roots	30
Number of levels	5-6
Fanout	5

表 3: データセットのパラメータ

図 5 に処理時間を示す。ここで、HPA の処理時間は示さなかった。表 4 に示すように、HPA では大量の通信が必要であり常に H-HPA よりも処理時間が長くなる。

ノード数	平均受信量 (MB)	
	HPA	H-HPA
8	7566.8	346.4
12	5532.7	272.5
16	4068.1	243.2

表 4: HPA と H-HPA の通信量

H-HPA-TGD は最小支持度が小さいとき、処理時間が長くなる。最初支持度が小さいとき、候補アイテム集合数が増大し、主記憶の空きが少なくなる。H-HPA-TGD では木の組合せ単位で候補アイテム集合の複製を行うため、このような場合に効果が得られない。H-HPA-PGD は小さな単位で候補アイテム集合の複製を行うため、最小支持度が小さい場合にも効果を発揮できる。

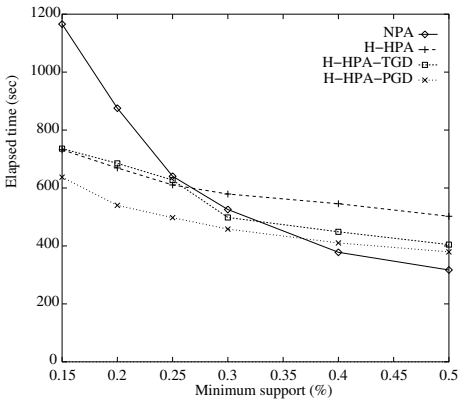

図 5: 処理時間

6 まとめ

本報告では分類階層を考慮した相関ルール抽出の並列処理方式の負荷制御手法を提案し、実際の並列計算機上への実装を行った。提案する手法により ノード間の負荷の偏りの影響を軽減させることが可能であることを示した。今回の測定では実際の 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を模倣して作成したデータセットを用いたが、候補アイテム集合数や負荷の偏りは対象とするデータに依存しているため、今後は様々なデータセットに対する詳細な性能評価を行う必要がある。

参考文献

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

[2] R.Srikant and R.Agrawal. Mining generalized association rules. In *Proc. of 20th Int. Conf. on Very Large Data Bases*, pp. 407–419, September 1995.

[3] J.Han and Y.Fu. Discovery of multiple-level association rules from large databases. In *Proc. of 20th Int. Conf. on Very Large Data Bases*, pp. 420–431, September 1995.

[4] 新谷隆彦, 喜連川優. 分類階層を考慮した相関関係の並列データマイニング. 情報処理学会データベースシステム研究会データベースワークショップ (DE96–33). 信学技法 Vol.96 No.176, pp. 55–60, July 1996.

[5] 新谷隆彦, 喜連川優. 並列相関関係抽出処理における通信負荷削減方式. 第 54 回全国大会 2R–2, pp. 249–250. 情報処理学会, March 1997.