分類階層を伴う相関ルールの並列マイニングとその動的負荷分散手法

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

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とその負荷分散方式を提案してきたが，従来の方式では効果的な負荷バランスを実現することが出来なかった．
本稿では，分散メモリ型並列計算機環境上でのデータの分類階層を伴う相関ルール抽出におけ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．

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

Parallel Mining Algorithms for Generalized Association Rules with Classification Hierarchy and their Load Balancing Strategy

Takahiko SHINTANI　Masaru KITSUREGAWA

Institute of Industrial Science, University of Tokyo
7-22-1, Roppongi, Minato-ku, Tokyo 106, JAPAN.
{shintani, kitsure}@tkl.iis.u-tokyo.ac.jp

Association rule mining recently attracted strong attention. Usually, the classification hierarchy over the data items is available. Users are interested in generalized association rules that span different levels of the hierarchy. In our previous study, we proposed parallel algorithms for mining generalized association rules with classification hierarchy and their load balancing methods. However, our previous works cannot attain sufficient load distributions.
In this paper, we present new load balancing algorithms for parallel mining generalized association rules with classification hierarchy. We implemented these algorithms on a shared-nothing environment and evaluated the performance of our algorithms.

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を抽出することが可能である．しかし，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 [2, 3]。[2, 3] 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，探索候補もノード間に分割することにより，システム全体の主記憶空間を効果的に利用している．また，データの分類階層を考慮して探索候補をハッシュ分割することにより，通信量の削減を図っている．さらに，数多くのトランザクションに含まれると予想される探索候補を全ノードに複製して個々のノードで処理することにより，処理負荷の偏りを低減する手法を提案してきた．しかし，それらの手法では全ノードに複製する探索候補の粒度が粗いため，十分な処理負荷の平坦化が実現できなかった．

本稿では，分類階層を考慮した相関ルール抽出の並列処理方式について述べ，従来の手法よりもさらに細粒度の負荷分散の制御を実現する手法を提案する．また，実際の分散メモリ型並列計算機上に実装し，提案する手法の性能評価を行い，その有効性を示す．

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 $\mathcal{T}$ は図 1 に示すような木構造を有し，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

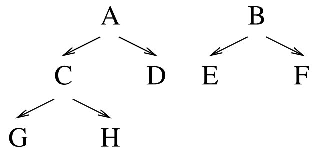

図 1: データの階層構造

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 ($ancestor$(x)) 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 $Sup(X)$ 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 のアイテムまたはそれら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．分類階層を考慮した相関ルール (一般化相関ルール) は $X \Rightarrow Y$ で表現され，$X, Y \subset \mathcal{I},\ X \cap Y = \phi$，$Y$ は X の上位アイテムを含まない．一般化相関ルールは支持度，確信度の 2 つの値により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．一般化相関ルール $X \Rightarrow Y$ の支持度 $Sup(X \Rightarrow Y)$ 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 $Sup(X \cup Y)$ により，また，確信度 $Conf(X \Rightarrow Y)$ 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 $Sup(X \cup Y)/Sup(X)$ により定義される．

一般化相関ルール抽出問題はユーザにより指定された最小支持度と最小確信度を満足する全てのルールを見出すことに相当する．その処理は，まず最小支持度を満足するアイテム集合 (ラージアイテム集合) を生成し，それらを用いて最小確信度を満足するルールを導出する．ラージアイテム集合生成処理はアイテム数，トランザクション数が多い場合に高負荷となり，この効率化が研究の中心となっている．

3 並列処理方式

はじめに，基本的な並列処理方式として分散メモリ型並列計算機環境をモデルとした並列処理方式 H-HPA(Hierarcical Hash Partitioned Apriori)[3] について述べる．図 2 にノード n での長さ k のラージアイテム集合を求める処理 (パス k) 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.
$\mathcal{D}^n$	Transactions stored in the local disk of the n-th node.
C_k^n	Sets of candidate k-itemsets whose hash value calculated with their root item is corresponding to n-th node.
L_k^n	Sets of large k-itemsets derived from C_k^n.

表 1: Notation

1. 候補アイテム集合の作成 (2) ~ (5)
長さ $(k-1)$ のラージアイテム集合から k 個のアイテムの組合せ (長さ k の候補アイテム集合) を作成し，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．残った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，自分のノードの識別子と等しい場合に主記憶上のハッシュ表に保持する．
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のアイテムを最下位ラージアイテムに置き換え，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

(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) $\{C_k^n\}$:=All the candidate k-itemsets, whose hash value calculated with its root items corresponding to the n–th node
(6) **forall** $t \in \mathcal{D}^n$ **do**
(7) t':=Replace items in t to the large item which is closest to the bottom of the hierarchy, 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) $\{L_k^n\}$:=All the candidates in C_k^n with minimum support
(19) Send L_k^n to the coordinator node
(20) /* Coordinator node make up $\mathcal{L}_k := \bigcup_n L_k^n$ and broadcast to all the nodes */
(21) Receive $\mathcal{L}_k$ from the coordinator node

図 2: Algorithm H-HPA

するノードを求める．各ノードに対応する最下位アイテムを送信する．
同時に他ノードから送信されたアイテム集合から長さ k のアイテム集合を作成し，対応する候補アイテム集合とそ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)
全トランザクションの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，他ノードへ放送する．

H-HPA では分類階層を考慮し，階層の上下関係で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ノード間にハッシュ分割する．つまり，候補アイテム集合を木の組合せでまとめ，その木の組合せ単位でノード間に割り当てる．

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，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低減させる手法を示す．本手法では支持度の高いアイテムからなる候補アイテム集合を全ノードに複製し，ノード毎に支持回数を数え上げ，各パスの最後に全ノードでの支持回数の総和を求め，ラージアイテム集合を決定する．他の候補アイテム集合は H-HPA と同様に処理する．

4.1 H-HPA with Tree Grain Duplicate (H-HPA-TGD)

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，木の組合せの粒度での候補アイテム集合の複製を行う．ノード n 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，記号の定義を表 2 に示す．

C_k^n	Sets of candidate k-itemsets whose hash value calculated with their root item is corresponding to n-th node.
L_k^n	Sets of large k-itemsets derived from C_k^n
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: Notation

(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 on 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) Delete the candidates from $\mathcal{C}_k$.
(5.6) $\{C_k^n\}$:= All the candidate k–itemsets, whose hash value calculated with its root items corresponding to the n–th node

(7.0) t':=Replace items in t to the large item which is closest to the bottom of the hierarchy, if there are small items in t
(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 $\mathcal{L}_k$:=$L_k^D + \bigcup_n L_k^n$ and broadcast to all the nodes */

図 3: Algorithm H-HPA-TGD

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．H-HPA 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の和でソートする．各ノードの主記憶空間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 n にのみ割り当てられた候補アイテム集合を C_k^n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^n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^nに関しては H-HPA と同様．

H-HPA-TGD では木の組合せの粒度で候補アイテム集合を全ノードに複製するため，十分な主記憶空間の空きが無い場合，H-HPA と等しくなる．

4.2 H-HPA with Path Grain Duplicate (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．ここで，最下位アイテムとそのすべての上位アイテムの組合せをパス (Path) と呼ぶ．H-HPA-PGD は H-HPA-TGD よりも最粒度の候補アイテム集合の複製を行うため，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 n 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．

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]
H-HPA 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 n にのみ割り当てられた候補アイテム集合を C_k^n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

(5.0) 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 $\mathcal{C}_k$.
(5.5) $\{C_k^n\}$:= All the candidate k–itemsets, whose hash value calculated with its root items corresponding to the n–th node

(7.0) t':=Replace items in t to the large item which is closest to the bottom of the hierarchy, if there are small items in t
(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 $\mathcal{L}_k$:=$L_k^D+\bigcup_n L_k^n$ and broadcast to all the nodes */

図 4: Algorithm H-HPA-PGD

4.3 H-HPA with Fine Grain Duplicate (H-HPA-FGD)

H-HPA-PGD は H-HPA-TGD よりも細粒度の負荷制御が可能であるが，複製する候補アイテム集合を最下位アイテムで判断するため，最下位アイテムの支持度は低いが，その上位の中間アイテムの支持度が高いものが存在する場合，十分な負荷制御が実現できない．

H-HPA-FGD では支持度の高い中間アイテムからなる候補アイテム集合とその上位候補アイテム集合を全ノードに複製する．つまり，H-HPA-PGD のパスから部分的に候補アイテム集合を選び出すことになるため，不必要な候補アイテム集合の複製を回避し，更に細粒度の負荷制御が可能となる．ノード n でのパス k の処理は図 2 の H-HPA に図 5 に示す変更を加える．

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]
H-HPA 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．全ての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を作成し，各ノードの主記憶に空きがなくなるまで複製する．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 n にのみ割り当てられた候補アイテム集合を C_k^n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

(5.0) Count the number of candidates allocated for each node.
(5.1) Sort the large items based on their frequency of appearance.
(5.2) Generate k-itemsets from the large items. in order of the frequency of appearance.
(5.3) $\{C_k^D\}$:=All the 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 $\mathcal{C}_k$.
(5.5) $\{C_k^n\}$:= All the candidate k-itemsets, whose hash value calculated with its root items corresponding to the n–th node

(7.0) t':=Replace items in t to the large item which is closest to the bottom of the hierarchy, if there are small items in t
(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 $\mathcal{L}_k$:=$L_k^D+\bigcup_n L_k^n$ and broadcast to all the nodes */

図 5: Algorithm H-HPA-FGD

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

本稿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，各方式の性能測定を行った．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．性能評価には，[4] 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．データセット作成に用いたパラメータとその意味を表 3 に示す．

Parameter	R30F5	R30F3	R30F10
Number of transactions	3200000		
Average size of the transactions	10		
Average size of the maximal potentially large itemsets	5		
Number of maximal potentially large itemsets	10000		
Number of items	30000		
Number of roots	30		
Number of levels	5–6	6–7	3–4
Fanout	5	3	10

表 3: データセットのパラメータ

5.1 処理時間

図 6 に各方式の最小支持度変化させた場合のパス 2 での処理時間を示す．ここでは 16 ノードのシステムを利用した．また，本稿で述べるすべての実験で候補アイテム集合数の最も多いパス 2 についてのみ測定を行い，トランザクションデータベースは各ノードのローカルディスクに均等に分割して割り当てた．

最小支持度の値が小さいとき，候補アイテム集合数が多くなり，各ノードの主記憶の大部分を占めることとなる．つまり，最小支持度が小さくなるに従い，候補アイテム集合を複製する主記憶空間が小さくなる．H-HPA-TGD では，木単位で候補アイテム集合を複製するため，最小支持度が小さい場合には候補アイテム集合を複製できなくなり，H-HPA と等しくなる．H-HPA-PGD と H-HPA-FGD は H-HPA-TGD よりも細粒度な単位で候補アイテム集合を複製するため，主記憶空間の効率の良い利用が可能であり，負荷の偏りを効果的に低減できる．特に，H-HPA-FGD はすべての最小支持度において最も良い性能を実現している．H-HPA-FGD は最も細粒度な候補アイテム集合の複製を行うため，最も効率良く主記憶空間を利用できる．

5.2 負荷バランスの比較

本節では，提案する方式の負荷バランスについて述べる．図 7 に各ノードでの候補アイテム集合のハッシュ表のパス 2 での検索回数を示す．ここで，最小支持度は 0.3%とした．結果から，H-HPA における各ノードの検索回数の偏りは大きいことがわかる．H-HPA-TGD，H-HPA-PGD，H-HPA-FGD では支持度の高いアイテムで構成される候補アイテム集合を全ノードに複製して処理するため，それらの候補アイテム集合に対するトランザクションデータの通信が不必要となる．H-HPA-TGD では候補アイテム集合の複製の粒度が大きいため，最小支持度が小さいときに効果を発揮できなくなる．H-HPA-PGD ではパス単位で候補アイテム集合を複製するため，H-HPA-TGD よりも良い負荷バランスが得られる．H-HPA-FGD は H-HPA-PGD よりもさらに細粒度な候補アイテム集合の複製を行い，不必要な候補アイテム集合の複製を回避し，最も効率良く主記憶空間を利用するため，効果的にデータの偏りの影響を低減することが可能となる．

5.3 台数効果

図 8 にノード数を変化させた場合の結果を示す．ここではノード数を 4，6，8，12，16 と変化させ，グラフは 4 ノードでの処理時間で正規化してある．また，データセットは R20F5 を利用し，最小支持度を 0.5%，0.3%とした，

H-HPA-FGD，H-HPA-PGD は H-HPA よりも良い台数効果が得られている．H-HPA は候補アイテム集合の複製を行わないため，データの偏りの影響により，良い台数効果が得られない．H-HPA-TGD，H-HPA-PGD，H-HPA-FGD では支持度の高いアイテムで構成される候補アイテム集合を見つけ出し，それらを全ノードに複製して処理する．候

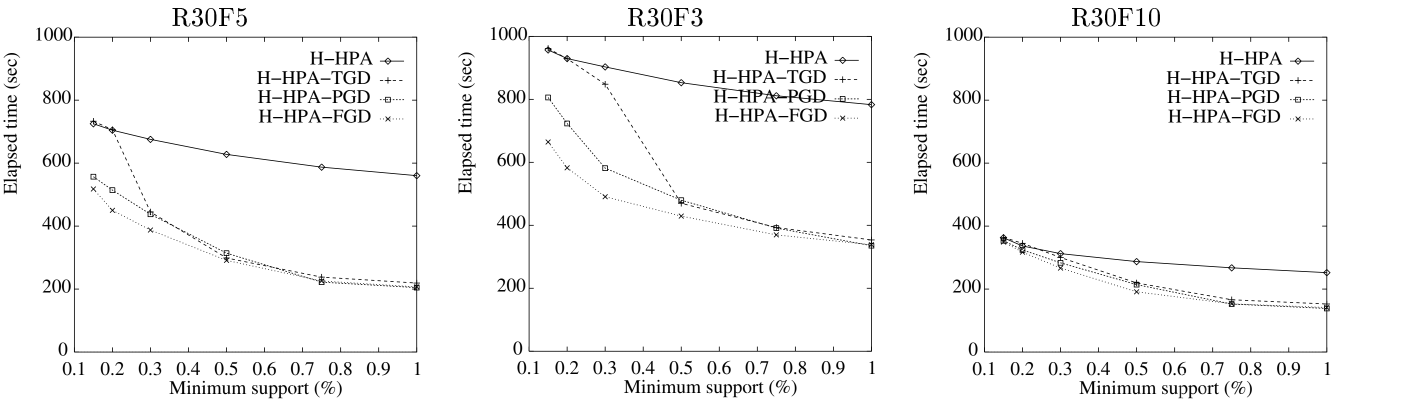

図 6: 各方式の処理時間

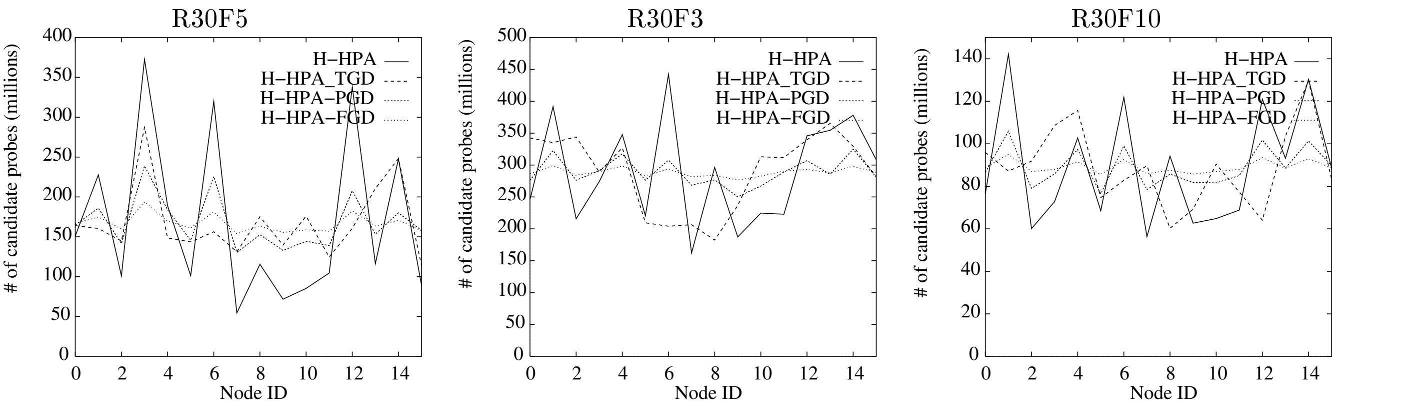

図 7: ハッシュ表の検索回数

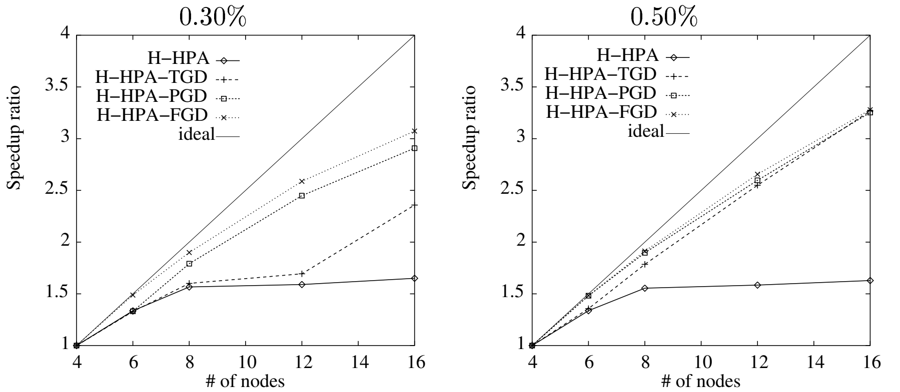

図 8: 台数効果

補アイテム集合の複製に利用できる主記憶空間が小さい場合，H-HPA-TGD では候補アイテム集合を複製できず，効果を発揮できない．一方，H-HPA-FGD は最も細粒度な候補アイテム集合の複製を行うため，効果的な性能を実現できている．

6　まとめ

本稿では分散メモリ型並列計算機環境における分類階層を伴う相関ルール抽出の並列処理方式とその負荷分散手法を提案し，実際の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．提案する手法での細粒度の候補アイテム集合の複製，主記憶空間の有効利用により，データの偏りの影響を低減し，効果的な性能を実現できることを示した．

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] 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.

[3] 新谷隆彦, 喜連川優. 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. 情報処理学会データベースシステム研究会 (DE97–44). 信学技法 Vol.97 No.161, pp. 103–108, July 1997.

[4] 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.