計算機アーキテクチャ 147−2
ハイパフォーマンス
コンピューティング 89−2
（2002．3．7）

最小サポート値を考慮した仮想的データ分割による データマイニングの高速化

篠田武志　松尾啓志

名古屋工業大学 電気情報工学科

データマイニングの重要な問題の一つに相関ルール抽出がある．データマイニングにおいては扱うデータが大規模であるため，効率の良い処理手法の研究が重要である．本論文では，ハッシュを用いてデータを仮想的に分割し，分割データの内容の偏りを利用して処理の効率化を行う手法を提案する．本手法を実装した結果，ハッシュによりデータが内容の偏りが大きくなるように分割され，処理の高速化が達成できたことを示す．また本手法を並列化し，SMP 計算機において実装を行い，性能を評価する．

High-Speed Association Rule Mining using Hash with Minimum Support

Takeshi Shinoda　Hiroshi Matsuo

Nagoya Institute of Technology

Mining association rules from a huge database is an important problem in data mining. Since databases are very huge, it is important to develop fast method for data mining. In this paper, an efficient alogorithm for mining association rules is proposed. This algorithm virtually partitions database with hash function, and then exploits the skewness of partitioned databases. As a result of implementing this algorithm, database is partitioned to have high skewness so that high-speed mining is achieved. Further, this algorithm is parallelized and implemented on SMP computer. The performance of parallelized algorithm is evaluated.

1. はじめに

データマイニングの重要な問題の一つに相関ルール抽出がある．相関ルールとは，例えばデパートの小売業データ（トランザクションデータ）において，“枕を購入した客の内の 98%は枕カバーも購入しており，枕と枕カバーの商品（アイテム）の組合わせ（アイテムセット）を購入した件数（サポート値）は買い物件数全体の 8%である” というような規則である．相関ルール抽出のためには，各アイテムセットのサポート値を数え上げることが必要となる．

データマイニングにおいては扱うデータが大規模であり，データのアイテム数も多いため，効率の良い処理手法の研究が重要である．Apriori Algorithm[1] をはじめとして，多くの手法が相関ルール抽出のために提案されている．Apriori Algorithm は，指定された最小サポート値以上のサポート値を持つアイテムセットを全て効率よく抽出するアルゴリズムである．Apriori Algorithm は他の多くの相関ルール抽出のためのアルゴリズムの基となっており，様々な拡張手法が提案されている．本手法も Apriori Algorithm を拡張した手法である．

また大規模データを扱うデータマイニングでは，並列処理が極めて有効である．そのため相関ルール抽出のための多くの並列手法が提案されている．Count Distribution[2], Non Partitioned Apriori (NPA)[3] は，トランザクションデータを各プロセッサに分割し，Apriori Algorithm を並列に処理する手法である．Data Distribution[2], Simply Partitioned Apriori (SPA)[3] は，トランザクションデータを各プロセッサに複製し，サポート値算出の対象となるアイテムセットを各プロセッサに分割して，並列処理を行う．Hash Partitioned Apriori with Extremely Large Itemset Decomposition (HPA-ELD)[3] は，SPA と同様にサポート値算出の対象となるアイテムセットを各プロセッサに分割し並列処理するが，一部の処理負荷の大きいと予想されるアイテムセットについては全プロセッサに複製して並列処理を行う手法である．

Distributed Mining of Association rules (DMA)[4], Fast Parallel Mining (FPM)[5], Adaptive Parallel

Mining (APM)[6] の手法は，トランザクションデータを分割した際の各分割データの内容の違いを利用して冗長なアイテムセットを処理から除外し，処理の効率化を行う．しかしそれらのアルゴリズムで用いられている冗長なアイテムセットを削減する手法は，内容が偏るようにデータが分割されていることを前提としており，分割されたデータの内容が互いに類似している場合には処理の高速化の効果が期待できない．Manning らは，主成分分析（Principal Component Analysis, PCA）を用いて分割データの偏りが大きくなるようにデータを分割し，DMA アルゴリズムを適用する手法を提案している[7]．しかし主成分分析は一般に計算コストがかかるため，処理全体の高速化の効果は少ないと考えられる．

本論文では，ハッシュを用いて分割データの内容の偏りが大きくなるようにデータを仮想的に分割し，APM で用いられる Partition Pruning を適用することにより処理の高速化を行う，Hash Partition Pruning (HPP) を提案する．ハッシュ関数は実行コストが小さく，十分にデータ内容の偏りが大きくなるようにデータを分割することが可能であるため，本手法により Partition Pruning の性能の向上を図ることができ，処理の高速化が期待できる．

本手法を実装し，ハッシュ関数を用いたデータ分割によって Partition Pruning の性能が向上し，処理の高速化が達成できたことを示す．また，本手法を並列計算環境で実装し，その性能を評価する．

本論文の構成は以下のとおりである．2 で相関ルール抽出について説明する．3 で従来手法である Apriori Algorithm，Partition Pruning の原理を説明する．4 で本論文で提案するハッシュを用いたデータ分割手法 Hash Partition Pruning について説明する．5 で本論文で提案する手法を実装した結果を示し，その性能評価を行う．6 で本論文のまとめを行う．

2. 相関ルール抽出

相関ルール抽出の目的は，データベースの中から価値のある相関ルールを効率的に発見することである．以下に相関ルール抽出について説明する．$I = \{i_1, i_2, \cdots, i_N\}$ をアイテム全体の集合とする（N はアイテム数を表す）．また D をトランザクション集合データベースとする．D 中の各トランザクション t は，$t \subseteq I$ となるアイテムの集合である．

相関ルールとは，アイテムセット $X(\subseteq I)$ と $Y(\subseteq I)$ により，$X \Rightarrow Y$ の形で記述される関係である．ここで X と Y は $X \cap Y = \emptyset$ である．$X \Rightarrow Y$ の関係は “トランザクションがアイテムセット X を含むならば，アイテムセット Y も含む” ということを表す．あるアイテムセット X について，D の内の $s\%$ のトランザクションが X を含むとき，アイテムセット X は s のサポート値を持つという．また，相関ルール $X \Rightarrow Y$ については，アイテムセット $X \cup Y$ のサポート値を相関ルール $X \Rightarrow Y$ のサポート値と定義する．また相関ルール $X \Rightarrow Y$ について，X を含むトランザクションの内の $c\%$ のトランザクションが Y も含むとき，相関ルール $X \Rightarrow Y$ は c の確信度（confidence）を持つという．

3. 従来手法

3.1 Apriori Algorithm

相関ルールのサポート値・確信度を求めることは，アイテムセットのサポート値を求めることに集約される．アイテムセットのサポート値を算出するとき，考えられる全てのアイテムセット数はアイテム数 N に対して 2^N 個という莫大な数になる．これら 2^N 個の全てのアイテムセットに対してそれぞれサポート値を求めるのは非常に計算コストがかかる．Apriori Algorithm[1] は，ユーザーに指定された最小サポート値を満たすアイテムセット（頻出アイテムセット）を全て抽出する効率の良い手法である．Apriori Algorithm は多くの頻出アイテムセット抽出手法の基となっている手法であり，本論文で提案する手法も Apriori Algorithm を改良した手法である．Apriori Algorithm は，あるアイテムセット X について，$Y \subset X$ なる Y が全て頻出アイテムセットでなければ X は頻出アイテムセットではない，という原理に基づき，サポート値を数えるアイテムセット（候補アイテムセット）を限定する．

Apriori Algorithm の処理手順を以下に示す．Apriori Algorithm ではまず始めに 1 回目のデータベースのスキャン（パス 1）において，全てのアイテムについてそれぞれサポート値を数え，アイテム数 1 の頻出アイテムセットを抽出する．次に，アイテム数 1 の頻出アイテムセットを組み合わせることによりアイテム数 2 の候補アイテムセットを生成する（以下この操作を apriori-gen と記述する）．そしてパス 2 において，アイテム数 2 の候補アイテムセットのサポート値を数えるために再びトランザクションデータをスキャンする．以降同様に，k 回目のデータベースのスキャン（パス k）では，アイテム数 k の候補アイテムセットについてトランザクションデータをスキャンすることにより各アイテムセットのサポート値を算出し，アイテム数 k の頻出アイテムセットを抽出する．そしてそれら頻出アイテムセットから関数 apriori-gen を用いてアイテム数 $k+1$ の候補アイテムセットを生成し，次のパスに移行する．このようにして，新たな候補アイテムセットが生成されなくなるまで処理を繰り返す

ことにより，データ中の全ての頻出アイテムセットを抽出する．

頻出アイテムセットではないようなアイテムセットに対するサポート値の数え上げは冗長であり，Apriori Algorithm はそのような余分なサポート値を数えるアイテムセットを 2^N 個から大きく減らすことができる．しかし Apriori Algorithm を用いた場合でも，特にアイテム数 1 の頻出アイテムセットから生成されるアイテム数 2 の候補アイテムセットは膨大な数になる．そしてそれら膨大数の候補アイテムセットの中に多く含まれる冗長な候補アイテムセットのサポート値の数え上げは，頻出アイテムセット抽出において多くの計算時間を占める．頻出アイテムセット抽出においては，冗長な候補アイテムセットの生成を抑制し冗長な計算を削減することが，処理の高速化において重要となる．

3.2 Partition Pruning

Adaptive Parallel Mining (APM)[6] で用いられる Partition Pruning は，トランザクションデータを仮想的に分割し，冗長な候補アイテムセットを削減する．ここではその原理について説明する．データベースを D とし，D を n 個の分割データ D_p $(1 \leq p \leq n)$ に分割する．あるアイテムセット W について，データ全体 D での W のサポート値をグローバルサポート値と定義し，$sup(W)$ で表す．またアイテムセット W の，分割データ D_p でのサポート値をローカルサポート値と定義し，$sup_p(W)$ で表す．

X をアイテム数 k の候補アイテムセットとする．もし $Y \subset X$ ならば，$sup_p(Y) \geq sup_p(X)$ $(1 \leq p \leq n)$ である．よって，X の D_p におけるローカルサポート値 $sup_p(X)$ は $\min\{sup_p(Y) | Y \subset X, \text{ and } |Y| = k-1\}$ を上界とする（ここで $|Y|$ はアイテムセット Y のアイテム数を表す）．従って，アイテム数 k である候補アイテムセット X のグローバルサポート値の上界は次の式 (1) のようになる．

$$sup(X) \leq \sum_{p=1}^{n} \left(\min_{Y \subset X,\ |Y|=k-1} \left\{ sup_p(Y) \right\} \right) \quad (1)$$

式 (1) 右辺が最小サポート値より小さいならば，X のサポート値の数え上げは不要であり省くことができる．

Partition Pruning の例を表 1 に示す．最小サポート値は 10 とする．表 1 最下行より {A,B,C,D} は全てアイテム数 1 の頻出アイテムセットであることがわかる．よって，関数 apriori-gen から生成されるアイテム数 2 の候補アイテムセット数は ${}_4C_2 = 6$ 個である．しかし式 (1) より，例えばアイテムセット {BC} のサポート値は 1+3=4 以下であることがわかり，アイテムセット {BC} は Partiton Pruning によって候補から除外される．6 個の候補の中で，{AB},{CD} 以外のアイテムセットは Partition Pruning によって全て候補から除外される．

表 1 Partition Pruning の例
Table 1 Example of Partition Pruning

アイテムセット	A	B	C	D
分割データ 1 でのサポート値	13	12	1	2
分割データ 2 でのサポート値	3	3	12	14
グローバルサポート値	16	15	13	16

4. Hash Partition Pruning

表 1 からわかるように，Partition Pruning は各頻出アイテムセットがそれぞれ特定の分割データ上に偏って存在するときに，効果的である．逆に，分割されたデータの内容が類似しているときは Partition Pruning は全く意味を成さない．Manning らは主成分分析 (Principal Component Analysis, PCA) を用いて分割データの偏りが大きくなるようにデータ分割する手法を提案している [7]．しかし主成分分析は一般に計算コストがかかるため，処理全体の高速化の効果は少ないと考えられる．本論文は，より計算コストの小さいハッシュを用いてデータを仮想的に分割することにより分割データの内容に偏りを持たせ，全体の処理を高速化する手法 Hash Partition Pruning (HPP) を提案する．

4.1 ハッシュによる仮想的データ分割

ハッシュの適用によるトランザクションデータの仮想的分割は，パス 1 のトランザクションデータスキャン時に以下のように行う．まず，候補アイテムセットのサポート値のカウンターをデータ分割数 $n(\leq N)$ 個用意する．そしてトランザクションデータを先頭からスキャンする．この時に，各トランザクションの内のアイテムの一つにハッシュを適用し，データ分割先のインデックス p $(1 \leq p \leq n)$ を算出する．本手法では，ハッシュ関数として演算量が少ないモジュロ演算を採用した☆．このようにして，どのサポート値カウンターに対してサポート値の演算を行うかを決定する．そしてカウンターの各トランザクション中に存在する候補アイテムセットのサポート値を 1 増加させ，サポート値を数え上げる．

このように，トランザクションの内のアイテムの一つにハッシュ関数を適用して各トランザクションの分割先を決定することにより，偏りのある仮想的データ分割が成される．

4.2 ハッシュキーアイテムの選択

本手法は，各トランザクション中のアイテムの一つにハッシュ関数を適用し，各トランザクションの分割

☆ 提案手法で用いるハッシュ関数は，一様な分布を有する関数であれば問題はない

先を決定する．しかし選択されたアイテムが頻出アイテムでない場合，そのアイテムにハッシュ関数を適用することによる分割はあまり意味を成さないと考えられる．そこで，最初のパス 1 において各候補アイテムの分割サポート値を数えるときに，各候補アイテムのグローバルサポート値を同時に数え，そのグローバルサポート値を元にどのアイテムにハッシュ関数を適用するかを決定する手法 HPPms（HPP with minimum support）を提案する．HPPms のアルゴリズムを図 1 に示す．HPPms では，パス 1 のトランザクションスキャン時のグローバルサポート値がその時点での最小サポート値を満たすアイテムの内で最初のアイテムにハッシュ関数を適用し，各トランザクションの分割先を決定する．グローバルサポート値を用いて統計的にハッシュキーとするアイテムを選択することにより，頻出アイテムセットでないアイテムを基にデータ分割を行うことをある程度避けられ，より効果的なハッシュ関数によるデータ分割を行うことができると考えられる．

図 2 に HPPms によるハッシュ分割の例を示す．トランザクション t={ABE} に対して，まず t に含まれる候補アイテム A,B,E のグローバルサポート値を 1 増加させる．次にグローバルサポート値が最小サポート率を満たすアイテムを一つ選ぶ．例えば図 2 において，t が 100 番目のトランザクションで，最小サポート値 s が 10%であったとする．この場合 A,B,E の中でグローバルサポート値が 10 以上であるアイテム B,E の内，最初のアイテムである B が選ばれる．そして選ばれたアイテム B に対してハッシュ関数を適用する．選ばれたアイテムのインデックス番号（B の場合は 2）にデータ分割数（例では 2）のモジュロ演算を適用することによりハッシュ値を算出し，トランザクションの分割先を決定する（モジュロ演算の結果が 0 の場合は最後尾の分割先に割り振る）．最後に分割先の A,B,E のサポート値を 1 増加させる．

例の場合，HPP によりトランザクションデータは分割データ 1 に割り振られたトランザクションはアイテム A,C,E のどれか一つを必ず持ち，分割データ 2 に割り振られたトランザクションはアイテム B,D を持つように分割される．その結果，アイテム C を持つトランザクションの多くが分割データ 1 に分割され，アイテム B を持つトランザクションは分割データ 2 に割り振られる．しかしトランザクション {ABD} に対して，サポート値の小さいアイテム A にハッシュを適用し分割データ 1 にトランザクションを割り振ることは，この場合偏りが大きくなるように分割することを妨げる．HPPms は A,B,D のグローバルサポート値を考慮してアイテムを選ぶことにより，サポート値

```
forall candidate c
    global_support[c]=0;
for(p = 1; p ≤ n; p++)
    forall candidate c
        support_counter[p][c]=0;
forall transaction t ∈ D
    forall candidate c ∈ t
        global_support[c]++;
    forall item i ∈ t
        if(global_support[i] ≥ order(t) × s)
            {p=hash(i); break;}
    forall candidate c ∈ t
        support_counter[p][c]++;
```

図 1 ハッシュによる仮想的データ分割アルゴリズム
Fig. 1 Algorithm of virtual partition by hash

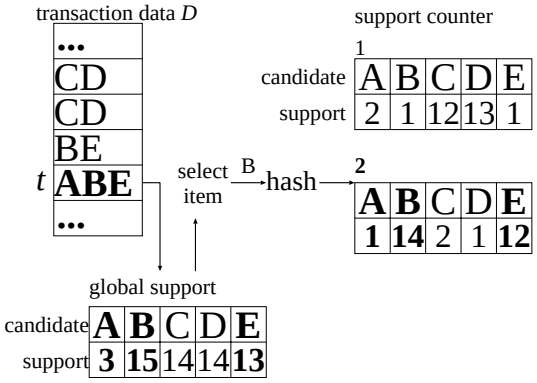

図 2 ハッシュによる仮想的データ分割
Fig. 2 Virtual partition by hash

が小さいアイテムにより分割先を決定することを回避する．

このようにして仮想的にデータを分割し，分割データごとの候補アイテムセットのサポート値を求める．偏りの大きい分割データに対して Partition Pruning を適用することにより，冗長な候補アイテムセット生成を抑制することができ，全体の処理の効率化が行われる．

5. 性能評価

提案した手法を単一計算機（Athlon 1.2GHz, 512MB）及び並列計算機（富士通 汎用計算サーバ GP7000Fmodel900）上で実装し，性能評価を行った．論文 [1] で提案された方法に従う人工データ生成プログラム [8] から生成したデータを用いた．データ生成に用いたパラメータは，トランザクション数 1,000,000，データのサイズは約 44MB である．最小サポート値を 0.75%として実験を行った．

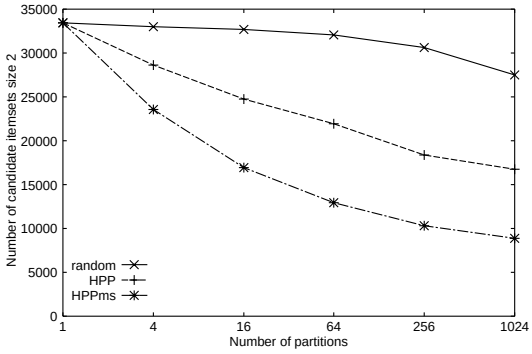

図 3 データ分割数とアイテム数 2 の候補アイテムセット数
Fig. 3 The number of data partitions vs the number of candidate itemsets in pass 2

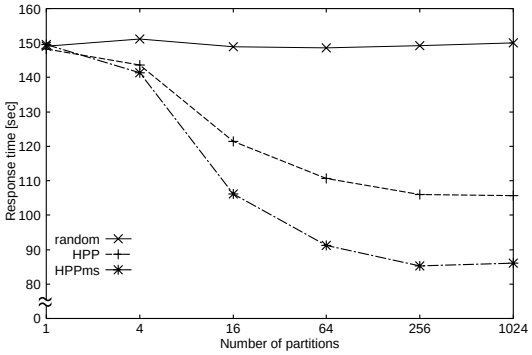

図 4 データ分割数と総実行時間
Fig. 4 The number of data partitions vs the response time [sec]

一般に Apriori Algorithm を基にした手法による相関ルール抽出においては，アイテム数 2 の候補アイテムセットの数が特に膨大となる．その数はアイテム数 2 の頻出アイテムセット数よりも極めて多く，パス 2 においては多数の冗長な候補アイテムセットが存在する．一方アイテム数 3 以上の候補アイテムセット数は，アイテム数 2 の候補アイテムセット数に比べれば極めて少数である．アイテム数 3 以上の場合，候補アイテムセット数は同じアイテム数の頻出アイテムセット数と比べてそれほど多くなく，冗長な候補アイテムセットはそれほど多くない．そのためパス 2 以降において，本手法のように候補アイテムセット削減を試みても，そもそも削減する候補アイテムセットがそれほど多くないため，逆に性能低下を引き起こしうる．よって今回の実験では，パス 2 以降の仮想的データ分割は行わなかった．

5.1 データ分割法による性能比較

以下の仮想的データ分割方法を用いて Partition Pruning を実装し，性能比較を行った．

(1) ランダム分割
(2) 各トランザクション中の，先頭位置のアイテムをハッシュキーとした仮想分割（HPP）
(3) 各トランザクション中の，トランザクションスキャン時点において最小サポート率を満たすアイテムの内で最初のアイテムをハッシュキーとした仮想分割（HPPms）

アイテム数は 1,000 に設定した．データを分割しないとき，つまり分割数が 1 のときは，処理は Apriori Algorithm と同じである．上の各仮想的データ分割方法について，データ分割数を横軸，それに対して生成されるアイテム数 2 の候補アイテムセット数を縦軸に取ったグラフを図 3 に示す．又，上の各分割方法について，データ分割数を横軸，総実行時間を縦軸に取ったグラフを図 4 に示す．

図 3 より，ハッシュを用いてデータを分割することによって分割データに内容の偏りが生じ，より効果的に Partition Pruning が行われ，冗長な候補アイテムセットが削減できたことがわかる．それにより，図 4 に示されるように処理全体の高速化が行われたことが確認できる．

5.2 並列計算機環境における実装

相関ルール抽出処理は並列性が高く，実際の適用は並列計算機環境下において行われる場合が多い．そこで次に提案手法を富士通 汎用計算サーバ GP7000Fmodel900 上で，本手法及び FPM[5] を実装し，性能評価を行った．

FPM[5] と同様に，トランザクションデータを先頭から均等に複数プロセッサに分割した．この時点ではハッシュ関数によるデータ分割は行わない．そして各プロセッサにおいて，各プロセッサに割り振られたデータに対してハッシュを用いて仮想的データ分割を行い，頻出アイテムセット抽出を行った．そして分割 support 値を統合し，頻出アイテムセット抽出と次のパスの候補アイテムセット生成を行った．

本手法において，各プロセッサでの仮想的データ分割数は 64 に固定した（よって処理全体でのデータ分割数は 64× プロセッサ数となる）．データのアイテム数は 1,000 に設定した．プロセッサ台数に対する速度向上率を図 5 に示す．またプロセッサ台数を横軸，総実行時間を縦軸に取ったグラフを図 6 に示す． 図 5 においては，本手法の速度向上率が悪くなっている．これは，冗長な候補アイテムセットを削減するために，式 1 を用いて表 1 の逐次計算処理を行い，また同時に本手法によって，並列に処理を行う候補アイテムセットのサポート値の算出処理時間が短縮されたことに起因する．しかし図 6 に示す総実行時間からも並列効果は十分に確認でき，冗長な候補アイテムセット削減のための逐次処理の追加にも関わらず，本手法は FPM

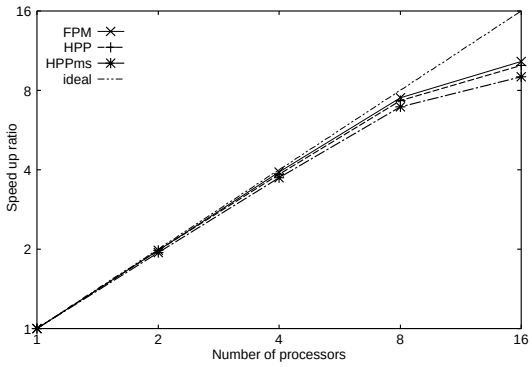

図 5 速度向上率
Fig. 5 Speed up ratio

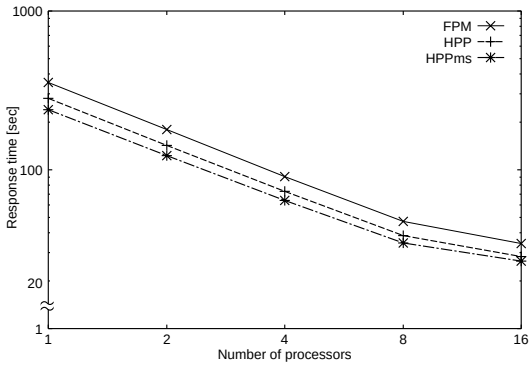

図 6 プロセッサ台数と総実行時間
Fig. 6 The number of processors vs The response time [sec]

よりも十分に高速に処理が可能であることを確認した.

6. ま と め

本論文では，相関ルール抽出において，ハッシュを用いてトランザクションデータを仮想的に分割し Partition Pruning を適用することにより，処理の効率化を図る手法を提案した．本手法では，各トランザクション中のアイテムの一つをハッシュキーとしてハッシュ値を算出し，トランザクションデータの分割先を決定することにより，内容の偏りが大きくなるようにデータ分割を行った．さらに本手法は，ハッシュを適用しデータを分割するする際に，最小サポート値を考慮しハッシュキーとなるアイテムを選択することによって更に有効なデータ分割を行い，冗長な候補アイテムセットの削減を図った．本手法を実装し性能評価を行った結果，ハッシュ関数を用いてデータを分割し Partition Pruning を適用することにより，冗長な候補アイテムセットが大幅に削減できることを示した．また本手法を並列計算機上で実装し，その性能を評価した．本手法と FPM を実装比較した結果，本手法を用いることで並列計算環境においても高速に処理できることを確認した.

今後は，本手法におけるより有効なハッシュ関数の検討や，並列処理時の効率化手法の検討などを行う予定である.

参 考 文 献

1) Rakesh Agrawal, Ramakrishnan Srikant, “Fast Algorithms for Mining Association Rules” Proc. of the 20th Int’l Conference of Very Large Databases, Santiago, Chile, Sept. 1994.
2) Rakesh Agrawal, John C. Shafer, “Parallel Mining of Association Rules” IEEE Transactions on Knowledge and Data Engineering, Vol. 8, No. 6, December 1996. Expanded version available as IBM Research Report RJ 10004, January 1996.
3) Takahiko SHINTANI, Masaru KITSUREGAWA, “Hash Based Parallel Algorithms for Mining Association Rules” Proceedings of IEEE Fourth International Conference on Parallel and Distributed Information Systems, pp.19–30, 1996.12
4) David W. Cheung, Vincent T. Ng, Ada W. Fu, Yongjian Fu, “Efficient Mining of Association Rules in Distributed Databases” Special Issue in Data Mining, IEEE Transaction on Knowledge and Data Engineering, IEEE Computer Society, V8, N6, December 1996, 911–922.
5) David W. Cheung, Yongqiao Xiao, “Effect of Data Skewness in Parallel Mining of Associations” Pacific-Asia Conference on Knowledge Discovery and Data Mining, 1998.
6) David W. Cheung, Kan Hu, Shaowei Xia, “An Adaptive Algorithm for Mining Association Rules on Shared-memory Multi-processors Parallel Machines” Distributed and Parallel Databases, Kluwer Academic Publishers, 9, 99–132, March 2001.
7) Anna M. Manning, John A. Keane, “Inducing load balancing and efficient data distribution prior to association rule discovery in a parallel environment” Euro-Par 99: Proc. 5th Int’l European Conference, pp1460–1463, Toulouse, France, Augut/Septermber 1999.
8) IBM Quest Data Mining Project. www.almaden.ibm.com/cs/quest/syndata.html