統計数理(2005)
第 53 巻 第 2 号 317–329

特集「計算推論—モデリング・数理・アルゴリズム—」
[原著論文]

# 飽和集合列挙アルゴリズムを用いた大規模データベースからのルール発見手法

宇野 毅明[1] ・有村 博紀[2]

(受付 2005 年 3 月 25 日; 改訂 2005 年 9 月 20 日)

**要　　旨**

**データベースの分析を通じた学習・知識発見は科学・産業の中で重要な位置を占める．学習・知識発見手法のひとつに，データベースから特徴的なパターン・ルールを列挙し，その中から役に立つ知識を探し出すというものがある．しかし，データベースが巨大であると，これらパターン・ルールの列挙には非常に長い時間が必要となり，また役に立たない不必要なパターンが大量に現れる．本稿では，筆者らが開発した飽和集合列挙アルゴリズム LCM を用いて効率良くルール発見を行う手法を提案する．飽和集合は意味が等しいパターンを代表するものであり，飽和集合に注目することにより，大量の不必要なパターンに関する処理を省略し，計算速度の向上と出力される解の数の減少を同時に行うことができる．また，飽和集合から高速にルールを生成し，その精度(確からしさ)を計算する手法も合わせて提案する．これにより，確からしい，またはある種の特徴的なルールのみを短時間で列挙することができるようになった．**

**キーワード：頻出集合，飽和集合，アソシエーションルール，アルゴリズム，データマイニング，機械学習．**

## 1. はじめに

**IT 技術の発達は，様々なデータを半自動的に収集することを可能にした．その結果，様々な種類の巨大なデータベースが構築されることとなった．これらのデータベースの利用目的は大きく分けて 2 つある．1 つは情報を記録し，検索することであり，もう 1 つはデータベースの分析により，新たな知識の発見やシステムの解析を行うことである．データの分析は，人類の科学的な活動の中で非常に重要な位置を占める．医学・工学・理学などの学術的な分野では，観測データからの知識発見や実験結果の検証などに，工業・経営などの産業の分野ではマーケティングや故障診断などに用いられる．**

**データベースの分析は，それ自体計算機パワーを用いる作業である．例えば，相関がありそうな要素の組を見つけるという単純な作業でも，素朴な方法で行えば，相関の度合いを調べる処理を，要素数の 2 乗回行うことになる．相関度合いの計算が 1 秒間に 1000 回できたとしても，要素数が 100 万となるような大きなデータベースでは，現実的な時間内に計算が終了しない．しかし，近年のアルゴリズムの発達と計算機パワーの増大により，より複雑な計算も高速に計算できるようになった．データマイニングでの頻出パターン発見(**Bayardo, 1998; Burdick et al., 2001;

[1] **国立情報学研究所：〒**101–8430 **東京都千代田区一ツ橋** 2–1–2; uno@nii.jp
[2] **北海道大学 情報科学研究科：〒**060–0814 **北海道札幌市北区北** 14 **条西** 9; arim@ist.hokudai.ac.jp

Han et al., 2000; Liu et al., 2003; Orlando et al., 2003; Pietracaprina and Zandolin, 2003)や，知識学習におけるルール発見(Agrawal and Srikant, 1994; Agrawal et al., 1996)はその良い例である．

頻出パターンは，データベースの中に頻出するパターン，あるいは構造のことをいう．データベースがトランザクションデータベース，つまり各項目がアイテムの多重でない集合となっており，かつパターンが要素の集合である場合，頻出パターンは頻出集合とよばれる．頻出集合はルール生成やクラスタリングなどに応用があり，またそれ自身が重要な知識を導くこともあるため，データマイニングの分野では基礎的かつ重要な問題である．そのため，多くのアルゴリズムが提案されており(Bayardo, 1998; Burdick et al., 2001; Han et al., 2000; Liu et al., 2003; Orlando et al., 2003; Pietracaprina and Zandolin, 2003)，多くの場合に実際に短時間で求解が行える．また，頻出集合を用いた多くのモデルが提案されている．しかし，重要な知識をもらさず見つけようとすると，頻出度合いの低い集合も見つける必要があり，そのために多数の解を処理しなければならない．これが頻出集合を用いたモデル化の弱点である．

頻出集合を用いて良いモデル化を行うためには，頻出集合の持つ情報を失わずに不要な頻出集合を捨て，解を絞り込む作業が必要である(Zheng et al., 2000)．そのため，例えば極大な頻出集合のみを見つける，といった手法が提案されている．飽和集合(closed itemset)は，そのような頻出集合の絞込みに用いることのできる構造である(Pasquier et al., 1999a, 1999b; Pei et al., 2000; Zaki and Hsiao, 2002; Uno et al., 2004a, 2004b)．全ての頻出集合を，「その頻出集合がどのトランザクションに含まれるか」に注目してグループ分けする．すると各グループの極大元は必ず唯一に定まる．その極大元のことを飽和集合とよぶ．飽和集合を全て列挙すれば，各グループから 1 つずつ代表を列挙したことに相当し，頻出度合い，および含まれるトランザクションの組に関しては，全ての可能性を網羅することができる．

例として，図 1 のトランザクションデータベースを見ていただきたい．このデータベースは，1 つの項目が 1 つの行に対応し，各行に書かれた数値がその項目に含まれるアイテムに対応する．

このデータベースにおいて，アイテム 7 は $T_1, T_2, T_4$ に含まれる．しかしこれは飽和集合ではない．$T_1, T_2, T_4$ に含まれるアイテムの集合で最も大きなものは $\{4,5,7\}$ であり，これは飽和集合である．アイテム集合 $\{7\}$ を用いて，$\{7\} \to 2$ というルールを考えよう．これは，「アイテム集合 $\{7\}$ を含むトランザクションはアイテム 2 を含む」という，ある種の性質である．このルールは，$T_1, T_2, T_4$ についての性質を述べている．つまり，左辺の $\{7\}$ というアイテム集合は，$T_1, T_2, T_4$ のみに共通に含まれるアイテム集合であれば，代わりにどれを用いても意味的に変化はない．そのため，このような形式のルールを全て生成するときには，アイテム集合が飽和集合であるもののみを考慮するだけで十分なのである．また，飽和集合を含むトランザクション数が少ない場合には，ルールの価値が低くなると考えられるため，頻出度の大きい飽和集合を用いることも重要である．

一般に，データベースがある種の偏りや構造を持つ場合，頻出飽和集合は頻出集合より数

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $T_1 = \{$ | 1 | 2 | 3 | 4 | 5 | | 7 | $\}$ |
| $T_2 = \{$ | | | | 4 | 5 | 6 | 7 | $\}$ |
| $T_3 = \{$ | | 2 | 3 | 4 | 5 | | | $\}$ |
| $T_4 = \{$ | | 2 | | 4 | 5 | | 7 | $\}$ |
| $T_5 = \{$ | 1 | 2 | 3 | | | | | $\}$ |

**図** 1.

が少ない．また，実用的にも高速な求解が可能である（Goethals, 2003）．筆者らは，2003 年に LCM（Linear time Closed itemset Miner）とよぶ頻出飽和集合列挙アルゴリズムを開発した（Uno et al., 2003, 2004a, 2004b）．このアルゴリズムは初めての多項式時間遅延アルゴリズムであり，また実用的にも高速である．アルゴリズムが多項式時間遅延であるとは，1 つの解を出力してから次の解を出力するまでの計算時間が，入力の大きさの多項式でおさえられることをいう．多項式遅延であれば，頻出飽和集合 1 つあたりの計算時間が入力の大きさの多項式時間となる．一般に列挙問題は多くの解を持つため，解の数の増大に伴う計算時間の増大，つまり大量の解を持つ問題をどの程度高速に解くことができるか，という基準が，アルゴリズムの設計上重要である．LCM は，実験的にも，入力の大きさの増大，出力する解の数の増大に伴う計算時間の増大が小さく，実用上有効である．また，いくつかのアルゴリズム的な技術を付加した実装は，データマイニングのプログラミングコンテストで優勝した実績を持つ（Goethals, 2003）．

LCM のスループット，つまり 1 秒間に見つける飽和集合の数はおおよそ 1 万個である．この値は，どのようなデータを用いても，解の個数がある程度大きければそれほど変化しない．つまり，現実的には，どのようなデータベースでもある程度の速度で飽和集合の列挙が可能なのである．だからといってルール発見が高速に行えるわけではない．例えば，あるルールを生成したときに，そのルールがどの程度の精度でデータを特徴づけているかを計算するためには，データベースをスキャンする必要があり，データベースが大きければそれだけ時間がかかる．いくら飽和集合を高速に見つけても，その後の処理に時間がかかってしまっては意味がない．

そこで本稿では，LCM を用いた効率良いルール発見の方法を提案する．アルゴリズムの基礎的な構造は LCM と同一であり，これにルールの精度を効率良く計算する手法を新たに付加した．この技術の導入により，ルールの確からしさ，特徴の評価が極めて短時間で行えるようになり，LCM の高速性を維持したまま，確からしい，あるいは特徴的なルールのみを短時間で列挙できるようになった．

まず 2 節で用語の定義を行い，続いて 3 節で LCM の概要を解説する．さらに 4 節で，前述のデータマイニングプログラミングコンテスト FIMI で使われた実データを用いて，処理速度と，どの程度の数のルールが発見されるか検証をする．なお，本稿の実装は著者の web ページで公開されている．http://research.nii.ac.jp/~uno/codes-j.html を参照されたい．

## 2. 準備

$I=\{1,2,...,n\}$ をアイテムの集合，$\mathcal{T}$ を $I$ 上のトランザクションの集合からなるデータベースとする．$||\mathcal{T}||=\sum_{T\in\mathcal{T}}|T|$，つまり $||\mathcal{T}||$ をデータベースの大きさとする．$|\mathcal{T}|$ は $\mathcal{T}$ が含むトランザクションの数であり，$||\mathcal{T}||$ とは異なることに注意されたい．$I$ の部分集合 $P$ に対して，$P$ を含む $\mathcal{T}$ のトランザクションを $P$ の出現とよぶ．$P$ の出現の集合を $\mathcal{T}(P)=\{T|P\subseteq T,T\in\mathcal{T}\}$ と表記し，$P$ の出現集合とよぶ．$P$ の出現集合の大きさを $P$ の頻度といい，$frq(P)$ と表記する．$P\subseteq Q$ であれば $\mathcal{T}(P)\supseteq\mathcal{T}(Q)$ となる．$P$ に含まれ，添え字が $i$ 以下のアイテムの集合を $P_{\leq i}$ と表記する．

アイテム集合 $P$ に対して，$P$ を真に含み，かつ $P$ と頻度が等しいアイテム集合が存在しないとき，$P$ を飽和集合とよぶ．アイテム集合 $P$ の閉包を，$P$ の全ての出現の共通部分，つまり $\bigcap_{T\in\mathcal{T}(P)}T$ で定義し，$clo(P)$ と表記する．アイテム集合 $P$ が飽和集合であるとき，またそのときに限り，$P$ の閉包は $P$ と等しい．

与えられた定数 $\theta$ に対して，$|\mathcal{T}(P)|\geq\theta$ を満たす $P$ は頻出集合とよばれる．$\theta$ をサポート値とよぶ．飽和集合であり，かつ頻出集合であるアイテム集合は，頻出飽和集合とよばれる．頻出集合 $P$ が他の頻出集合に含まれないならば，$P$ を極大頻出集合とよぶ．

アイテム集合 $P$ とアイテム $i$ の組 $(P,i)$ を，アソシエーションルール，あるいは単にルールとよぶ．特に $P$ が頻出飽和集合であるとき，$(P,i)$ を頻出飽和ルールとよぶ．アイテム集合 $P$ に対して，$frq(\{P\})/|\mathcal{T}|$ を $P$ の出現割合とよび，$ratio(P)$ と表記する．さらに，$P$ の独立出現割合 を $\Pi_{e\in P} ratio(\{e\})$ で定義し，$Iratio(P)$ と表記する．また，ルール $(P,i)$ に対して，$frq(P\cup\{i\})/frq(P)$ を誘導出現割合とよび，$ratio(P|i)$ と表記する．集合 $P$ の出現割合は，データベースのトランザクションのうち，$P$ を含むものの割合を示す．また，独立出現割合は，$P$ に含まれる各アイテムがトランザクションに含まれる，という事象が互いに独立であると仮定したときに，$P$ がある 1 つのトランザクションに含まれる確率である．

## 3. 飽和集合列挙

与えられたデータベース $\mathcal{T}$ とサポート値 $\theta$ に対して，$\mathcal{T}$ の頻出飽和集合を全てもらさず，かつ重複なく列挙する問題を頻出飽和集合列挙問題とよぶ．この節では，この問題を解く出力数多項式時間アルゴリズム LCM (Linear time Closed itemset Miner)の解説を行う．LCM は，prefix 保存飽和拡張という手法を用いて深さ優先的に飽和集合を生成する．また，効率良く prefix 保存飽和拡張を行い，かつ頻出度を計算するために，出現配布という手法を使う．以下の小節でこれらの技術の解説を行う．

### 3.1 prefix 保存飽和拡張

既存の頻出飽和集合の列挙アルゴリズムは，頻出集合の列挙を基礎とするものが多い(Pasquier et al., 1999a, 1999b; Pei et al., 2000; Zaki and Hsiao, 2002)．つまり，頻出集合を列挙し，その中で飽和集合となっているもののみを出力する，というものである．この方法は，頻出集合数と頻出飽和集合数に大きな違いがない場合は有効であるが，両者の差が大きいと，飽和集合とならない頻出集合を大量に生成することになり，効率が悪くなる．そのためにいくつかのヒューリスティックな枝刈り法が提案されており，計算速度の向上に貢献している．しかし，枝刈りは完全ではないため，計算量は頻出飽和集合数に対して線形とはならない．出力される解の数が増加すると，計算時間が線形より真に大きな割合で増加する可能性がある．

飽和集合は極大 2 部クリークと等価であることが知られている(Boros et al., 2002)．トランザクションデータベース $\mathcal{T}$ に対して，頂点がアイテムとトランザクションに対応し，トランザクションがアイテムを含むときに両者に対応する頂点間に枝を張った 2 部グラフの極大 2 部クリークと飽和集合が 1 対 1 に対応する．極大 2 部クリークに対する多項式時間列挙アルゴリズムはいくつか提案されているので(Tsukiyama et al., 1977; Uno et al., 2003; Makino and Uno, 2004)，それらを改良すれば，飽和集合の列挙アルゴリズムが得られる．

LCM は，これら極大クリーク列挙アルゴリズムから派生した prefix 保存飽和拡張という手法を用いている．飽和集合 $P$ に対して，そのコアアイテム $core\_i(P)$ を $clo(P_{\leq i})=P$ が成り立つような添え字最小のアイテムとする．$core\_i(P)$ は必ず $P$ に含まれる．このとき，$P'$ が $P$ の prefix 保存飽和拡張であるとは，あるアイテム $e>core\_i(P)$ に対して $P'=clo(P\cup\{e\}), P'_{\leq e-1}=P_{\leq e-1}$ が成り立つことをいう．$P_0$ を，$\mathcal{T}(P_0)=\mathcal{T}$ となる飽和集合で定義する．$P_0$ はすべてのトランザクションに含まれる集合の中で極大なものである．$P_0$ は空集合となり得ることに注意されたい．$P_0$ でない任意の飽和集合は，自身より頻出度が真に大きい飽和集合の prefix 保存飽和拡張となっている．さらに，そのような飽和集合は唯一である(Uno et al., 2004a, 2004b)．そのため，$P_0$ から出発し，再帰的に prefix 保存飽和拡張を生成することにより，全ての頻出飽和集合を深さ優先のやり方で発見できる．

prefix 保存飽和拡張を用いた飽和集合列挙法の計算量は，頻出飽和集合数に対して線形である．そのため，計算時間は入力した問題が持つ解の数の増加に対して，線形より大きく増加す

ることはないということが理論的に保証される．prefix 保存飽和拡張の詳細については Uno et al.（2004a, 2004b）を参照されたい．

### 3.2 頻出度計算

一般に，頻出集合列挙アルゴリズムにおいて，最も計算コストがかかる部分は新たに生成したアイテム集合 $P$ の頻出度，つまり $|\mathcal{T}(P)|$ の計算である．アイテム集合 $P$ に対して $|\mathcal{T}(P)|$ の計算を単純に行うと，基本的にデータベースを全てスキャンする必要がある．そのため，頻出集合列挙アルゴリズムの多くは，$P$ の部分集合の出現から，$\mathcal{T}(P)$ を効率良く計算することで，高速な頻出度計算を実現している．例えば，ダウンプロジェクト（Bayardo, 1998; Han et al., 2000; Liu et al., 2003; Pietracaprina and Zandolin, 2003）とよばれる手法は，$\mathcal{T}(P \cup \{i\}) = \mathcal{T}(P) \cap \mathcal{T}(\{i\})$ であることを利用して，$\mathcal{T}(P \cup \{i\})$ を求める際に，$\mathcal{T}(P)$ と $\mathcal{T}(\{i\})$ の共通部分を計算することにより高速化を行っている．ダウンプロジェクトは頻出集合列挙においては一般的な手法であり，多くのアルゴリズムと実装に用いられている．

LCM も他のアルゴリズムと同様，各反復で現在注目している頻出集合 $P$ に各アイテム $i$ を追加し，その頻出度を計算している．$P \cup \{i\}$ が頻出であれば，さらに，その閉包が prefix 保存飽和拡張であるかどうかを調べる．この際，各 $i$ について $\mathcal{T}(P \cup \{i\})$ を計算する．他のアルゴリズムはこれらの作業にダウンプロジェクトを用いているが，LCM は出現配布法とよばれる手法を用いている（Uno et al., 2003, 2004a, 2004b を参照）．

出現配布法の仕組みは以下のとおりである．まず，追加する各アイテム $e \in H$ に対して，空のバケツを用意する．次に，$\mathcal{T}(P)$ の各出現 $T$ に対して，$T$ が含むアイテム $f$ のバケツに $T$ を入れる，という作業を各 $f \in T$ について行う．この操作の終了後，アイテム $e$ のバケツは $\mathcal{T}(P \cup \{e\})$ と等しくなる．この操作にかかる計算時間は $O(\sum_{T \in \mathcal{T}(P)} |T|)$ となる．これは，ダウンプロジェクトを用いた場合の計算時間 $O(||T||)$ より短い．

### 3.3 計算時間

LCM の各反復では，各アイテム $e$ に対して，$P \cup \{e\}$ の頻出度計算を行い，さらに閉包の計算を行う．頻出度計算にかかる時間は，前節の議論より $O(\sum_{T \in \mathcal{T}(P)} |T|)$ である．また，閉包の計算には，各アイテム $e$ に対して $O(\sum_{T \in \mathcal{T}(P \cup \{e\})} |T|)$ 時間かかる．つまり，各反復での，再帰呼び出し内での計算を除いた計算時間は，$O(\sum_e \sum_{T \in \mathcal{T}(P \cup \{e\})} |T|) = \mathrm{O}(|I| \times ||\mathcal{T}||)$ 時間である．LCM の反復数は，頻出飽和集合数と等しいため，LCM の計算時間は，飽和集合 1 つあたり $O(|I| \times ||\mathcal{T}||)$ である．実際の計算では，多くの $e$ が prefix 保存飽和拡張を生成するため，1 反復の計算時間は平均して $O(\sum_{T \in \mathcal{T}(P)} |T|)$ 程度である．また，頻出度の小さい飽和集合の数は，頻出度の大きな飽和集合の数に比べて大きいことが多く，そのため多くの反復では $\sum_{T \in \mathcal{T}(P)} |T|$ は小さい．これが LCM が実用上も高速である 1 つの根拠である．

## 4. 効率的なルール発見

本節では，前節で解説した LCM を用いて，特徴的な頻出飽和集合と頻出飽和ルールの発見を高速に行う手法を提案する．特徴的であるとは，飽和集合や飽和ルールが，その独立出現割合に比べてある程度大きい，あるいは小さい出現割合を持つことをいう．飽和集合やルールが特徴的である場合，それらのパターンに含まれるアイテムは依存関係を持ち，またルールは何らかの意味を持つと考えられる．ここでは具体的に，与えられた定数 $\epsilon < 1$ に対して，特徴的な飽和集合 $P$ として以下のような条件を満たすものを考える．

条件 1: $Iratio(P) \leq \epsilon(ratio(P))$ か $1 - Iratio(P) \leq \epsilon(1 - ratio(P))$ が成り立つ
条件 2: $ratio(P) \leq \epsilon(Iratio(P))$ か $1 - ratio(P) \leq \epsilon(1 - Iratio(P))$ が成り立つ

この 2 つの条件は，$P$ に含まれるアイテムの集合が依存関係を持つことを表している．$\epsilon$ が小さいほど，強い依存関係を表す．

同様に，特徴的な頻出飽和ルール $(P,i)$ として，以下の条件を考える．

条件 3: $ratio(P|i) \geq 1-\epsilon$
条件 4: $ratio(P|i) \leq \epsilon$
条件 5: $\epsilon(ratio(P|i)) \geq ratio(\{i\})$　あるいは $1-ratio(P|i) \leq \epsilon(1-ratio(\{i\}))$
条件 6: $ratio(P|i) \leq \epsilon(ratio(\{i\}))$　あるいは $\epsilon(1-ratio(P|i)) \geq (1-ratio(\{i\}))$

これらの条件は，ルールの確からしさの尺度となっている．条件 3，4 は「$P$ ならば $i$ である」，あるいは「$P$ ならば $i$ でない」というルールが確からしいことを表す条件である．条件 5，6 は，$P$ という事象と $i$ という事象が依存関係を持つことを表す条件である．これらの条件も，$\epsilon$ が小さいほど強い条件となる．

これら 5 つの条件は，アイテム集合とその頻度からのみ得られる条件としては自然であろう．飽和集合が特徴的であるかどうかの検証，つまり条件 1，2 の判定には，その飽和集合の大きさに比例する時間しかかからず，効率良く計算できる．しかし，特徴的なルールの場合，条件 3，4，5，6 を検証するためには，各反復で保持している現在注目している頻出集合 $P$ に対して，各 $(P,i)$ の誘導出現割合を計算する必要がある．つまり，各 $i$ に対して $frq(P \cup \{i\})$ の値を計算する必要がある．一般に，この作業には多大な時間がかかる．例えば，多くの頻出集合・頻出飽和集合列挙アルゴリズムで用いられている手法を用いた場合，各アイテム $i$ について $\mathcal{T}(P)$ と $\mathcal{T}(\{i\})$ の共通部分を計算する必要があり，これには少なくとも $O(\sum_{i \in I} |\mathcal{T}(\{i\})|) = O(||\mathcal{T}||)$ 時間かかり，つまり本質的にデータベースをすべてスキャンする必要があるのである．対して LCM は，各反復で出現配布を行い，各 $i$ に対して $\mathcal{T}(P \cup \{i\})$ を求めている．そのため，各 $i$ に対して，$frq(P \cup \{i\})$ を定数時間で求められる．よって，計算量オーダーの増加なしにこれら飽和ルールの条件検証ができることとなる．

## 5. 計算実験

本研究では，前節で提案した特徴的な頻出飽和集合，および特徴的なルールを発見するアルゴリズムの実装を行った．この節では，多種の実データやベンチマーク問題に対する計算実験の結果を提示し，主に以下の点について観察を行った．

- データベースの密度と計算時間の増加
- 飽和集合発見アルゴリズムとの比較
- 実データはどの程度の特徴的なルールを含むか

まず，基本となる LCM アルゴリズムであるが，Uno et al. (2003)にて提案された，もっとも基礎的なものを用いた．Uno et al. (2004a)や Uno et al. (2004b)で提案されたアルゴリズムのほうがより高速であるが，誘導出現割合と相性が悪い技術(反復的データベース縮約)が使われているため，使用しなかった．プログラムは標準的な C で書かれ，gcc でコンパイルされた．計算機環境は，Pentium4 3.2GHz，メモリ 2GB を積んだ PC であり，OS は Linux である．ただし，今回の実験では最大でも 500MB のメモリしか使用しなかったことを注意しておく．

実験に用いたデータセットは，頻出集合列挙のレポジトリである FIMI レポジトリ(Goethals, 2003）のデータセットの中から表 1 の数点を選んだ．

BMS-WebView-1，retails，BMS-pos，kosarak は，アイテム集合の大きさが 500 から 1 万と大きいが，1 つのトランザクションが含むアイテムの数が平均で 10 程度，あるいはそれ以下

表 1.

| データセット名 (種別) | トランザクション数 | アイテム数 | データベースサイズ | 平均トランザクションサイズ |
|---|---|---|---|---|
| BMS-WebView-1(商業 web ページの閲覧記録データ) | 59602 | 497 | 149639 | 2.5 |
| retails (バスケットデータ) | 88162 | 16470 | 908576 | 10.3 |
| BMS-pos (POS データ) | 517255 | 1657 | 3367020 | 6.5 |
| kosarak (ハンガリーのニュースサイトでのクリックストリームデータ) | 990002 | 41271 | 8019015 | 8.1 |
| connect (機械学習用ベンチマークデータ) | 67557 | 130 | 2904951 | 43 |
| pumsb (機械学習用ベンチマークデータ) | 49046 | 7117 | 3629404 | 74 |
| accidents (交通事故のデータ) | 340183 | 469 | 11500870 | 33.8 |

表 2.

| | |
|---|---|
| LCM(time) | LCM の計算時間 |
| rule(time) | ルール発見アルゴリズムの計算時間 |
| 90% | $\epsilon = 0.1$ に対して条件 3 を満たすルールの数, つまり $ratio(P\|i) \geq 0.9$ であるルールの数 |
| 90% & minimal | $\epsilon = 0.1$ に対して条件 3 を満たすルール $(P, i)$ の中で, $P$ が極小である ($P$ の任意の真部分集合が導出するルールは条件 3 を満たさない) ものの数 |
| 90% & 10times & minimal | $\epsilon = 0.1$ に対して, 条件 3 と条件 5 を満たすルール $(P, i)$ の中で, $P$ が極小なものの数 |
| really frequent closed itemset | 条件 1 の前半を $\epsilon = 0.1$ に対して満たす頻出飽和集合数 |
| frequent closed itemset | 頻出飽和集合数 |

**と，疎なデータセットになっている．**connect，pumsb **では，一部のアイテムが半分以上のトランザクションに含まれ，密な部分を形作っている．そのため，小さいサポート値に対しては莫大な数の頻出集合を持つが，各トランザクションが規則的に生成されているため，ある種の構造を持ち，頻出飽和集合の数はそれほど大きくない．**accidents **も同様に密な部分構造を持つが，実データであるために規則的な構造を持たず，頻出飽和集合の数と頻出集合の数に大きな開きはない．**

**各データセットに対し，いくつかのサポート値で実験を行った．実験結果を，図** 2 **から図** 8 **のグラフにまとめた．縦軸は問題の持つ解の数と計算時間(秒)，横軸はサポート値である．それぞれの項目は表** 2 **のようになっている．**

**個数が** 0 **である場合，その項目は表示されていない．例えば，**accidents，pumsb，connect **においては，特徴的な頻出飽和集合数，**really frequent closed itemset **の数は** 0 **である．ルール発見の計算時間については，パラメータや条件の変化に関わらず，データセットとサポート値が同じであれば，すべて等しいことを注意しておく．また，特徴的な頻出飽和集合発見にかかる計算時間も，**LCM **の計算時間とほぼ等しかったため，グラフ上への表記は省略した．**

BMS-WebView-1 **および** BMS-pos **では，任意のサポート値に対して，頻出飽和集合数と** really frequent closed itemset **はほぼ等しく，グラフ中では重なって表示されている．また，**pumsb，connect **および** accidents **では，任意のサポート値に関して，**really frequent closed itemset **は**

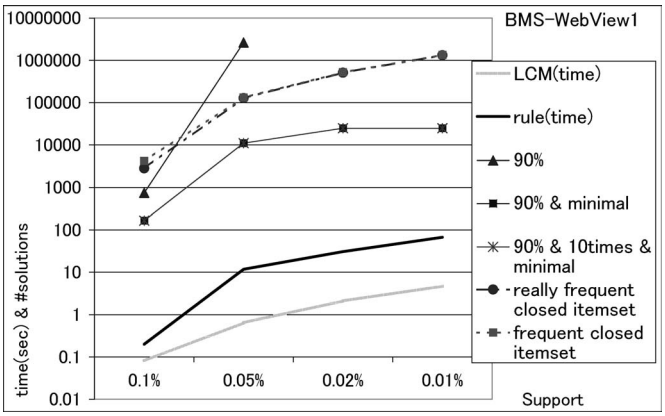
BMS-WebView1
10000000
1000000
100000
10000
1000
100
10
1
0.1
0.01
time(sec) & #solutions
0.1%
0.05%
0.02%
0.01%
Support
LCM(time)
rule(time)
90%
90% & minimal
90% & 10times & minimal
really frequent closed itemset
frequent closed itemset

**図** 2.

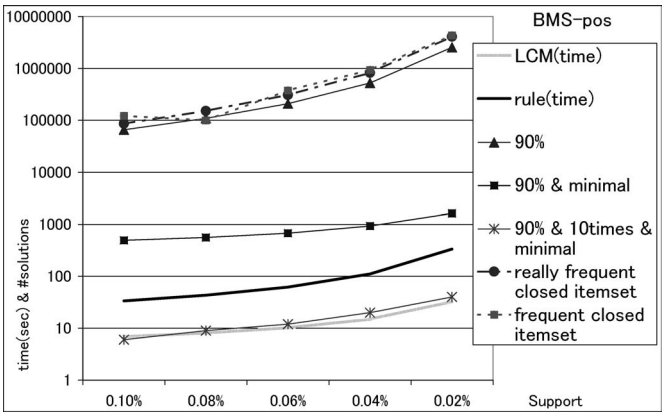
BMS-pos
10000000
1000000
100000
10000
1000
100
10
1
time(sec) & #solutions
0.10%
0.08%
0.06%
0.04%
0.02%
Support
LCM(time)
rule(time)
90%
90% & minimal
90% & 10times & minimal
really frequent closed itemset
frequent closed itemset

**図** 3.

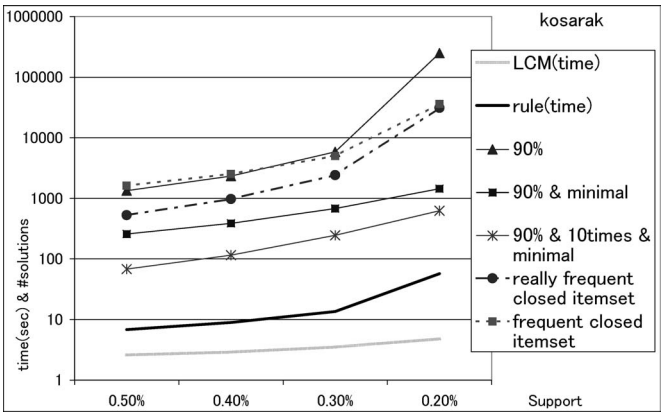
kosarak
1000000
100000
10000
1000
100
10
1
time(sec) & #solutions
0.50%
0.40%
0.30%
0.20%
Support
LCM(time)
rule(time)
90%
90% & minimal
90% & 10times & minimal
really frequent closed itemset
frequent closed itemset

**図** 4.

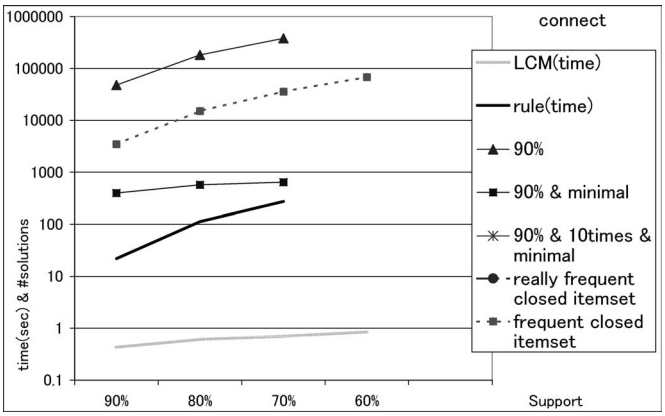
connect
1000000
100000
10000
1000
100
10
1
0.1
time(sec) & #solutions
90%
80%
70%
60%
Support
LCM(time)
rule(time)
90%
90% & minimal
90% & 10times & minimal
really frequent closed itemset
frequent closed itemset

**図** 5.

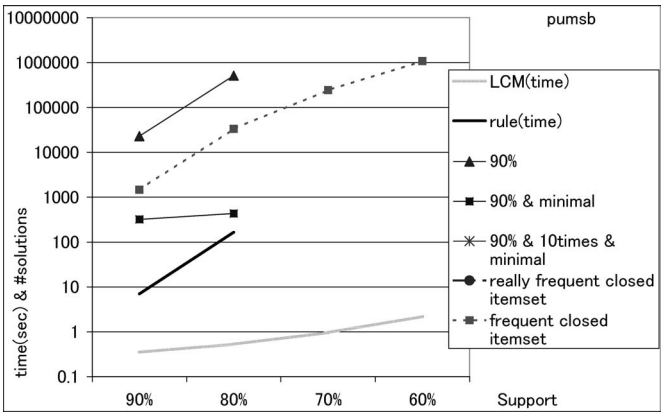
pumsb
10000000
1000000
100000
10000
1000
100
10
1
0.1
time(sec) & #solutions
90%
80%
70%
60%
Support
LCM(time)
rule(time)
90%
90% & minimal
90% & 10times & minimal
really frequent closed itemset
frequent closed itemset

**図** 6.

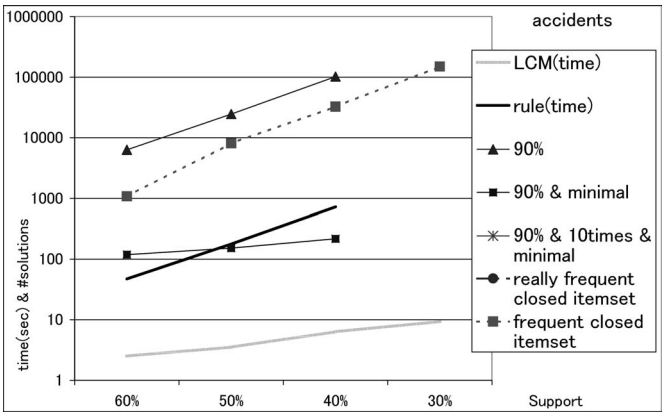
accidents
1000000
100000
10000
1000
100
10
1
time(sec) & #solutions
60%
50%
40%
30%
Support
LCM(time)
rule(time)
90%
90% & minimal
90% & 10times & minimal
really frequent closed itemset
frequent closed itemset

**図** 7.

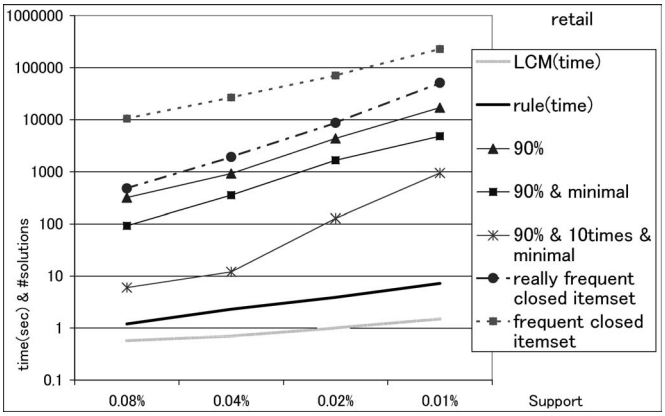


**図** 8.

0 **であった．**BMS-WebView-1 **では，任意のサポート値に対して** 90% & minimal **と** 90% & 10 times & minimal **の値が等しく，グラフ上では重なって見える．**

**ルール発見に要した計算時間は，疎なデータセットに対しては，任意のサポート値で** LCM **本体の計算時間とさほど変わりがない．計算時間の増加は高々**2∼5 **倍である．しかし，密なデータベースに対しては，計算時間が大きく増大している．これは，**LCM **が用いているデータベース縮約の技術をルール発見の計算では用いていないためである．データベース縮約は特に密なデータベースに対する計算時間の短縮に大きく貢献することが確認されている（**Uno et al., 2004a, 2004b**）．**

**条件** 1 **を満たす頻出飽和集合数と頻出飽和集合数は，**BMS-WebView-1 **および** BMS-pos **ではほぼ等しかったが，他のデータセットでは大きな違いがあった．これは，多くのデータセットにおいて，ルールや知識の候補数を意味的な損失なしに減少させる可能性があることを示している．**

**また，特徴的なルールの発見においても，**90%，90% & minimal，90% & 10 times & minimal **の値に大きな違いがあることが多く，自然な条件をいくつか用いてルールを絞り込むことにより，意味的な損失を抑えて効果的に候補数を減少できることを示している．実際にデータベースの解析にルール発見を用いる場合，この性質は大きな利点となるだろう．**

## 6. まとめ

**本稿では，頻出飽和集合列挙アルゴリズム** LCM **を用いて，データベースから特徴ある飽和集合と特徴的なルールを列挙する手法を提案した．計算実験の結果，データベースが疎であれば，単純な飽和集合の列挙に比べて，計算時間の増大がそれほど大きくないことが示された．また，特徴的なルールは，飽和集合に比べてそれほど数が多くなく，付加的な条件を加えることで，効率良く絞込みが行えることを示した．**

**本稿の計算実験で用いた実装は，密なデータベースを効率良く扱う「データベース縮約」を用いていないため，密なデータベースにおいては計算効率が落ちている．**LCM ver 2 **（**Uno et al., 2004a, 2004b**）で提案された「極大性を判定できるデータベース縮約」は，ルールの計算と同時に用いることができるため，これを改良すれば，密なデータベースを効率良く扱える可能性がある．これは今後の課題としたい．しかし，現実に現れる巨大なデータベースの多くが疎であ**

るため，密なデータベースへの対応にどの程度の意味があるかには，疑問が残るだろう．なお，本稿の実装は，著者の web ページで公開してある．http://research.nii.ac.jp/~uno/codes-j.html を参照されたい．

## 謝　辞

この研究に多大なるご助力をいただいた国立情報学研究所佐藤健先生に感謝する．また，丁寧に査読をしていただいた査読者に感謝する．また，この研究は，文部科学省・学術振興会の科学研究補助，および国立情報学研究所共同研究費の補助を受けた．

## 参 考 文 献

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases, *Proceedings of VLDB'94,* 487–499, Morgan Kaufmann, San Fransisco, California.

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H. and Verkamo, A. I. (1996). Fast discovery of association rules, *Advances in Knowledge Discovery and Data Mining*, 307–328, AAAI Press/MIT Press, Menlo Park, California.

Bayardo, R. J., Jr. (1998). Efficiently mining long patterns from databases, *Proceedings of SIGMOD'98*, 85–93, ACM Press.

Boros, E., Gurvich, V., Khachiyan, L. and Makino, K. (2002). On the complexity of generating maximal frequent and minimal infrequent sets, *Proceedings of STACS 2002*, 133–141, Springer, Berlin.

Burdick, D., Calimlim M. and Gehrke, J. (2001). MAFIA: A maximal frequent itemset algorithm for transactional databases, *Proceedings of International Conference on Data Engineering 2001*, 443–452, IEEE Computer Society.

Goethals, B. (2003). The FIMI'03 Homepage, http://fimi.cs.helsinki.fi/

Grahne, G. and Zhu, J. (2003). Efficiently using prefix-trees in mining frequent itemsets, *Proceeding of IEEE ICDM'03 Workshop FIMI'03* (Available as CEUR Workshop Proceedings Series, Vol. 90, http://ceur-ws.org/vol-90).

Han, J., Pei, J. and Yin, Y. (2000). Mining frequent patterns without candidate generation, *SIGMOD Conference 2000*, 1–12, ACM.

Kohavi, R., Brodley, C. E., Frasca, B., Mason, L. and Zheng, Z. (2000). KDD-Cup 2000 Organizers' report: Peeling the onion, *SIGKDD Explorations*, **2**, 86–98.

Liu, G., Lu, H., Yu, J. X., Wei, W. and Xiao, X. (2003). AFOPT: An efficient implementation of pattern growth approach, *Proceeding of IEEE ICDM'03 Workshop FIMI'03* (Available as CEUR Workshop Proceedings Series, Vol. 90, http://ceur-ws.org/vol-90).

Makino, K. and Uno, T. (2004). New algorithms for enumerating all maximal cliques, *Lecture Notes in Computer Science*, **3111**, 260–272.

Orlando, S., Lucchese, C., Palmerini, P., Perego, R. and Silvestri, F. (2003). kDCI: A multi-strategy algorithm for mining frequent sets, *Proceeding of IEEE ICDM'03 Workshop FIMI'03* (Available as CEUR Workshop Proceedings Series, Vol. 90, http://ceur-ws.org/vol-90).

Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L. (1999a). Efficient mining of association rules using closed itemset lattices, *Information Systems*, **24**, 25–46.

Pasquier, N., Bastide, Y., Taouil, R. and Lakhal, L. (1999b). Discovering frequent closed itemsets for association rules, *Proceedings of ICDT'99*, 398–416.

Pei, J., Han, J. and Mao, R. (2000). CLOSET: An efficient algorithm for mining frequent closed item-

sets, *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery 2000*, 21–30.

Pietracaprina, A. and Zandolin, D. (2003). Mining frequent itemsets using patricia tries, *Proceeding of IEEE ICDM'03 Workshop FIMI'03* (Available as CEUR Workshop Proceedings Series, Vol. 90, http://ceur-ws.org/vol-90).

Tsukiyama, S., Ide, M., Ariyoshi, H. and Shirakawa, I. (1977). A new algorithm for generating all the maximum independent sets, *SIAM Journal on Computing,* **6**, 505–517.

**宇野毅明**(2003). **大規模グラフに対する高速クリーク列挙アルゴリズム，電気情報通信学会コンピュテーション研究会報告集.**

Uno, T., Asai, T., Uchida, Y. and Arimura, H. (2003). LCM: An efficient algorithm for enumerating frequent closed item sets, *Proceeding of IEEE ICDM'03 Workshop FIMI'03* (Available as CEUR Workshop Proceedings Series, Vol. 90, http://ceur-ws.org/vol-90).

Uno, T., Asai, T., Uchida, Y. and Arimura, H. (2004a). An efficient algorithm for enumerating closed patterns in transaction databases, *Lecture Notes in Artificial Intelligence*, **3245**, 16–31.

Uno, T., Kiyomi, M. and Arimura, H. (2004b). LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets, *Proceeding of IEEE ICDM'04 Workshop FIMI'04* (Available as CEUR Workshop Proceedings Series, Vol. 126, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-126/).

Zaki, M. J. and Hsiao, C. (2002). CHARM: An efficient algorithm for closed itemset mining, *Proceedings of the 2nd SIAM International Conference on Data Mining* (*SDM'02*), 457–473.

Zheng, Z., Kohavi, R. and Mason, L. (2000). Real world performance of association rule algorithms, *KDD 2001*, 401–406.

Proceedings of the Institute of Statistical Mathematics Vol. 53, No. 2, 317–329 (2005) 329

# An Efficient Method for Finding Association Rules from Large Scale Databases Based on Closed Pattern Enumeration

Takeaki Uno[1] and Hiroki Arimura[2]

[1]National Institute of Informatics
[2]Graduate School, Information Science and Technology, Hokkaido University

Learning and knowledge discovery by database analysis have important roles in science, industry, and economy. Finding distinctive patterns and association rules for discovering valuable knowledge is a common approach to such learning and knowledge discovery. However, it takes long time if the database size is huge, and we also have to deal with a huge number of unnecessary candidate patterns. In this paper, we propose a method for efficiently finding such patterns and rules by using closed pattern mining algorithm LCM that we proposed previously. Closed patterns are representatives of patterns having the same denotations. Hence, we can reduce the number of candidate patterns, and the computation time. We also propose an algorithm for evaluating rules in a short time. We can thus enumerate all association rules of high confidences or having interesting properties.

Key words: Frequent pattern, closed pattern, association rule, algorithm, data mining, machine learning.