社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

頻出パターンの高速列挙

宇野　毅明†

† 国立情報学研究所, 〒 101-8430 東京都千代田区一ツ橋 2-1-2
E-mail: †uno@nii.jp

キーワード　頻出集合, 頻出度, 飽和集合, 閉包集合, 極大頻出集合, Trie, FP-tree, アプリオリ, データベース, データマイニング

Fast Enumeration of Frequent Patterns

Takeaki UNO†

† National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, JAPAN, 101-8430
E-mail: †uno@nii.jp

アイテム集合 $\mathcal{I} = \{1, ..., n\}$ の部分集合をトランザクションという．各項目がトランザクションとその項目の ID のペアからなるデータベースを**トランザクションデータベース**とよぶ．以下では簡単のため ID を除いて考え，トランザクションデータベースはトランザクションの集合であるとする．トランザクションデータベースは，同一のトランザクションを複数含んでよいとする．データベース $\mathcal{T}$ とアイテム集合 I に対して，I を含む $\mathcal{T}$ のトランザクションを I の**出現** とよび，I の出現の集合を I の**出現集合**とよぶ．出現集合の大きさを I の**頻出度**という．データベース $\mathcal{T}$ と定数 θ (**サポート**とよぶ) が与えられたとき，頻出度が θ 以上の集合を**頻出集合**とよぶ．極大な頻出集合を極大頻出集合とよぶ．また，アイテム集合 I が，I と同じ頻出度を持ち，I を含むような集合が存在しないとき，I を**飽和集合** [6] とよぶ．頻出集合は**頻出パターン**とよばれることも多く，むしろ飽和集合より一般的に用いられるが，パターンという用語は文字列など他の構造を表すこともあるため，ここでは頻出集合という用語を用いる．

与えられたデータベースとサポートに対して，その頻出集合を全て列挙する問題は，データマイニングの分野で非常に基礎的かつ重要である．アソシエーションルールや分類ルールなどの発見や，データベースの解析など，数々の問題の基礎となるからである．本発表では，今までに行われてきた研究を，アルゴリズムと実装の面からまとめ，昨年，今年と行われているプログラム実装コンテスト [4] の結果(注1)とあわせて紹介する．以下にその内容を簡単にまとめる．

計算上の困難性

頻出集合は，簡単なバックトラック法で，頻出集合 1 つあたり多項式時間で列挙できる．飽和集合も，2 部グラフの極大 2 部クリークと等価であるため，築山らのアルゴリズム [8], [9] を用いて 1 つあたり多項式時間で列挙できる．しかし，極大頻出集合を列挙する，極大頻出集合 1 つあたり多項式時間のアルゴリズムはまだ発見されていない．また，存在しないことの証明もされていない．実際の計算では，効率の良いヒューリスティックな枝狩り法が使われており，ほとんどの問題が効率良く解けている．さらに言えば，ある種のアルゴリズムが必ず指数時間費やすような問題を構築することもまた難しい．

実際の計算で，最も時間を費やす部分は頻出度の計算である．通常，データマイニングに用いるデータベースは巨大であるため，素直な方法で頻出度を計算すると，データベースを全てスキャンし，計算に数秒を要することになる．これでは，100 万を超える出力を持つ問題は解けない．そこで，入力時，および計算途中でのデータベースの縮約や，疎・密な構造に特化したアルゴリズムの開発により，高速化が行われている．

計算時間の他に，メモリ使用量も計算効率化の大きな課題である．入力が大きなことに加え，極大や飽和集合の列挙アルゴリズムには，発見した解をメモリに蓄えるアルゴリズムが多いため，解の増加に伴うメモリの使用量が増加する．また，これらの解やメモリの管理にも多大な時間を費やすことになる．

(注1)：Frequent Itemset Mining Implementations:
http://fimi.cs.helsinki.fi

列挙法

頻出集合の列挙法で最も古いものは，幅優先的な探索を行う**アプリオリ** [1] という手法である．これは，大きさ 1 の頻出集合を全て求めてメモリに蓄え，これらを組合せて大きさ 2 の頻出集合を全て求め，と 1 つずつ大きなサイズの頻出集合を求めていくものである．これに対して, [2] を初めとする**バックトラック法**(注2)は，空集合から出発して，アイテムを 1 つずつ加え，非頻出になったらバックトラック (直前に加えたアイテムを取り除くこと) し，他のアイテムを加えて探索を続ける，というバックトラック的 (深さ優先的) に頻出集合を列挙する手法である．出力の重複を避けるため，バックトラック法は「アイテム i を追加したら，それより先の探索 (再帰呼び出しで行われる探索) では i より小さいアイテムは追加しない」というルールを用いる．これは，アイテム集合を辞書順で探索していることに対応する．

バックトラック法の長所は，メモリ使用量が少ないことである．入力の線形サイズしかメモリを使用しない．対してアプリオリは，解を全て保存するためメモリ使用量が莫大になる．アプリオリの長所は，頻出度計算を行う際に，非頻出となる集合をある程度枝狩りできることである．大きさ $k+1$ の集合 I の部分集合で頻出でないものが存在すると，その時点で I は非頻出であることが確認できる．アプリオリは既発見解を全てメモリに蓄えているため，この方法で頻出かどうかをある程度チェックすることができ，頻出度計算を一部省略することができる．しかし実際の計算では，省略できる計算は全体の 1/4 以下であることが多く [12]，効果はあまり大きくない．近年の頻出集合列挙アルゴリズムの多くがバックトラック法を用いているようである．

飽和集合の列挙

飽和集合の列挙は多項式時間で行えるが，そのままでは非常に遅いアルゴリズムとなるため，実際には頻出集合列挙と非飽和集合のヒューリスティックな枝狩を組合せたアルゴリズムが用いられることが多い [7], [13]．この方法は，メモリを多く用いること，解の数に対して計算時間が長くなることがあることが欠点である．対して，宇野・有村ら [10]〜[12] は *ppc-extension* を提案している．これは，既発見解を用いることなく，重複なく多項式時間で各飽和集合を生成できる．ppc-extension を用いたアルゴリズムは，入力サイズに多項式のメモリしか使用しない上に，高速である．

頻出度計算

アイテム集合 I の出現集合を計算しようとすると，最悪の場合データベースを全てスキャンする必要がある．しかし，I の出現集合が得られていれば，I に アイテム e を加えた集合の出現集合は，I の出現集合と $\{e\}$ の出現集合の共通部分で与えられる．このようにして出現集合の計算を行い，計算の高速化を行う方法を *Down project* という．多くのアルゴリズムがこの手法を用いて頻出度計算を行っている．対して，宇野・有村ら [10], [11] は**出現配布** という手法を提案している．この手法は，I の出現集合と I に追加しようとしているアイテムの集合に限定したデータベースを作り，それの転置を計算することで各アイテムを追加したときの出現集合を全て計算するという手法である．出現配布は出現集合の大きさの合計に対して線形の時間で実行できるため，特に疎なデータベースに対して効率的であり，また，密なデータベースでも Down project と同程度の効率である．

データ構造

頻出集合列挙の入力はしばしば大きく，また不必要な部分が多くある．そのため，データを保持するためのデータ構造も高速化には重要である．基本的な操作として，データベースから非頻出なアイテムを取り除き，同一なトランザクションを 1 つにまとめる，**データベース縮約**という操作がある．ほぼ全てのアルゴリズムが，入力データに対してこの操作を行っている．

MAFIA [3] は，データの保持にビットマップを用いている．トランザクションとアイテムの組に対して 1 ビットしか要しないため，密なデータを効率良くメモリに格納でき，また，32 ビット単位で計算ができること，および CPU キャッシュのヒット率が高くなることにより計算効率も良くなる．しかし，データベースの縮約などとは相性が悪い．

Trie (FP-tree [5]) は，prefix tree を用いて入力データを保持する手法である．各トランザクションのアイテムを頻出度の高い順でソートすると，prefix を共有するものが多くなり，圧縮効率が良くなる．近年の極大・飽和・頻出集合列挙アルゴリズムの非常に多くが Trie を用いている．しかし，実際の圧縮率は，単純なデータベース縮約 (同一なトランザクションを縮約する操作のみを行うもの) と比べて要素数が半分になる程度であり，2 分木を構築するためは 1 つの要素につき 3 つのポインタが必要なことを考えると，実際の効率は良くないようである [12].

LCM2 [12] は，反復型データベース縮約を用いている．これは，出現配布で作成する限定したデータベースに，データベース縮約を行い，再帰にしたがって反復的にデータベースを小さくする手法である．(FP-growth [5] も FP-tree を用いて同じような操作を行っている) 列挙アルゴリズムは，深いレベルの反復の数が浅いレベルに比べて指数的に大きくなるため，非常に効率が良い．

極大性の判定

極大頻出集合・飽和集合の列挙には，新たに見つけた頻出集合が極大であるか，あるいは飽和集合であるかの判定が不可欠である．アプリオリ型のアルゴリズムは，既発見解の中で極大となっているもの (飽和集合の場合は同一の頻出度を持つものの中で極大となっているもの) を保持することで，極大性の判定を行っている．宇野・清見・有村 [12] のアルゴリズムは，極大・飽和の判定用に，新たなデータベース縮約を行うことで，解を

(注2)：部分グラフ族の列挙などで用いられる列挙アルゴリズムの基本的な手法である

メモリに保持することなく，効率的に極大性・飽和集合の判定を行っている．

FIMI: 実装コンテスト

昨年・今年と，頻出集合・頻出飽和集合・極大頻出集合の 3 種類の列挙アルゴリズムの実装コンテストが行われた．様々なアイディアを用いた多くのアルゴリズムが投稿された．多くのアルゴリズムは，

・頻出集合列挙にはバックトラック法
・飽和・極大頻出集合列挙にはアプリオリ型アルゴリズム
・データ構造はビットマップか FP-tree
・頻出度計算は DownProject

を用いていた．対して，宇野・有村らの LCM [11], [12] は，

・飽和・極大頻出集合列挙には,ppc-extension とメモリを使わない枝狩りを用いたバックトラック法
・データ構造は配列と反復型データベース縮約
・頻出度計算は出現配布

を用いた．昨年は，FP-tree を用いた基本アルゴリズムである FP-growth が優勝し，今年は LCM が優勝した．半分程度の問題例で LCM が最速であったが，疎なデータでの高速化など，今後の課題は多い．

文　献

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. I. Verkamo, “Fast Discovery of Association Rules,” *In Advances in Knowledge Discovery and Data Mining,* MIT Press, pp. 307–328, 1996.

[2] R. J. Bayardo Jr., “Efficiently Mining Long Patterns from Databases”, In *Proc. SIGMOD’98*, pp. 85–93, 1998.

[3] D. Burdick, M. Calimlim, J. Gehrke, “MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases,” In *Proc. ICDE 2001*, pp. 443-452, 2001.

[4] B. Goethals, *the FIMI Homepage*, `http://fimi.cs.helsinki.fi/` , 2003.

[5] J. Han, J. Pei, Y. Yin, “Mining Frequent Patterns without Candidate Generation,” *SIGMOD Conference 2000*, pp. 1-12, 2000

[6] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, *Efficient Mining of Association Rules Using Closed Itemset Lattices*, *Inform. Syst.*, 24(1), 25–46, 1999.

[7] J. Pei, J. Han, R. Mao, “CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets,” *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery 2000*, pp. 21-30, 2000.

[8] S. Tsukiyama, M. Ide, H. Ariyoshi and I. Shirakawa, “A New Algorithm for Generating All the Maximum Independent Sets,” *SIAM Journal on Computing,* Vol. 6, pp. 505–517, 1977.

[9] 宇野 毅明, “大規模グラフに対する高速クリーク列挙アルゴリズム,” 電気通信学会コンピュテーション研究会, 京都大学, 2003.

[10] T. Uno, T. Asai, Y. Uchida, H. Arimura, “An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases,” Discovery Science 2004, Lecture Notes in Artificial Inteligence 3245, pp. 16–31, 2004.

[11] T. Uno, T. Asai, Y. Uchida, H. Arimura, “LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets,” In *Proc. IEEE ICDM’03 Workshop FIMI’03*, 2003. (Available as CEUR Workshop Proc. series, Vol. 90, http://ceur-ws.org/vol-90)

[12] T. Uno, M. Kiyomi, H. Arimura, “LCM ver.2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets,” In *Proc. IEEE ICDM’04 Workshop FIMI’04*, 2004. (Available as CEUR Workshop Proc. series, http://ceur-ws.org/)

[13] M. J. Zaki, C. Hsiao, “CHARM: An Efficient Algorithm for Closed Itemset Mining,” *2nd SIAM International Conference on Data Mining (SDM’02)*, pp. 457-473, 2002.