負の相関ルールマイニングの効率化のための飽和アイテム集合からの極小生成子の高速抽出

Fast Extraction of Minimal Generators from Closed Itemsets to Improve Negative Association Rule Mining

谷島 健斗 [1*]　岩沼 宏治 [2]　山本 泰生 [2,3]
Kento Yajima[1]　Koji Iwanuma[2]　Yoshitaka Yamamoto[2,3]

[1] 山梨大学大学院医工農学総合教育部工学専攻コンピュータ理工学コース
[1] Computer Science and Engineering Course, Integrated Graduate School of Medicine, Engineering and Agricultural Sciences, University of Yamanashi
[2] 山梨大学大学院総合研究部
[2] Interdisciplinary Graduate School, University of Yamanashi
[3] 科学技術振興機構 さきがけ
[3] JST PRESTO

Abstract:
This paper considers fast extraction of the minimal generators from a given closed itemset. The minimal generator plays a very important role for compressing the set of valid negative association rules. We propose three kinds of algorithms to efficiently extract the minimal generators from all closed itemsets. The first is a bottom-up method, the second is a top-down one and the third is a fusion algorithm, i.e., a top-down search algorithm with a bottom-up pre-processing for pruning. We also show some experimental results for evaluating our proposed methods.

1　はじめに

本論文では, 負の相関ルールマイニングの効率的な実装のために, 極小生成子を飽和アイテム集合から高速に抽出する手法について考察する. トップダウン型とボトムアップ型の 2 種類の極小生成子抽出手法について実証的な比較研究を行う.

相関ルールとは, トランザクションデータベース中に頻繁に共起するアイテム集合の関係を記述したものである. X と Y をアイテム集合とするとき, データベース中で X が出現するトランザクションの多くに Y も出現することを $X \Rightarrow Y$ と表し, 正の相関ルール [1, 2] と呼ぶ. これに対して, 本論文では, X と Y がほとんど同時に出現しない現象を表現する $X \Rightarrow \neg Y$ や $\neg X \Rightarrow Y$ なる形の負の相関ルールを考察する. 負の相関ルールは正のルールでは表現が困難な共起関係を記述でき, 有用である [3, 4, 5, 6]. ただし, 負の相関ルールを抽出するためには, 非頻出なアイテム集合を扱う必要がある. そのため, 正の相関ルールの場合に比べて探索空間が格段に大きく, また抽出されるルールの数も非常に多くなる. 井出ら [7] は接尾辞木を用いた深さ優先型の負ルール抽出アルゴリズムを提案し, 負ルール抽出の効率化を行った. 岩沼ら [8] は極小生成子を用いて, 抽出される負ルール集合の可逆圧縮を行った. さらに, 我々は [9] において, 有効な負の相関ルール集合を圧縮した形で効率的に抽出するために, 頻出アイテム集合ではなく, 極小生成子を用いて負の相関ルールの抽出を効率的に行う計算手法を提案した. 本研究では, 先行研究 [9] で提案された負の相関ルール抽出システムを効率的に実装するために, 飽和アイテム集合から極小生成子を高速抽出する手法を提案する.

*連絡先： 山梨大学大学院工学専攻 (修士課程)
コンピュータ理工学コース
山梨県甲府市武田 4-3-11
E-mail:g17tk024@yamanashi.ac.jp

2　準備

2.1　正の相関ルール

$I = \{x_1, x_2, ..., x_n\}$ を**アイテム**の全体集合とする時, **トランザクション** t をアイテム集合 $t \subseteq I$ と定める. **トランザクションデータベース** D をトランザクションの

多重集合とする．X をアイテム集合とするとき，$X \subseteq t$ となる D 中のトランザクション t を X の**出現**と呼び，その多重集合を $D(X)$ と略記する．多重集合 X の**大きさ**を $|X|$ と表記し，$|D(X)|$ を X の D 中の**出現頻度**と呼ぶ．X の D 中の**支持度 sup**(X) を $\sup(X) = \frac{|D(X)|}{|D|}$ と定義する．**正の相関ルール** (以下，“正ルール”と略記する) を，$X \cap Y = \emptyset$ であるアイテム集合 X，Y からなる表現 $X \Rightarrow Y$ と定める．X と Y をそれぞれルールの**前件**，**後件**と呼ぶ．正ルールに対する**支持度** sup と**確信度** conf は以下のように定義される [1]．

$$\sup(X \Rightarrow Y) = \sup(X \cup Y)\ ,\ \ \mathrm{conf}(X \Rightarrow Y) = \frac{\sup(X \cup Y)}{\sup(X)}$$

最小支持度 ms と**最小確信度** mc とはユーザが支持度と確信度に関して与える閾値である．$\sup(X) \geq ms$ を満たす X を**頻出アイテム集合**と呼ぶ．また $\sup(X \Rightarrow Y) \geq ms$ と $\mathrm{conf}(X \Rightarrow Y) \geq mc$ の両方を満たす $X \Rightarrow Y$ を**妥当** (valid) な正の相関ルールと呼ぶ．

2.2 負の相関ルール

本論文では，**負の相関ルール** (*negative association rule:* 以下では “負ルール” と略記する) を考察する．X と Y を $X \cap Y = \emptyset$ なるアイテム集合とするとき，負ルールとは以下のいずれかの表現である．

- $X \Rightarrow \neg Y$　　(右否定形もしくは後件負形)
- $\neg X \Rightarrow Y$　　(左否定形もしくは前件負形)
- $\neg X \Rightarrow \neg Y$　　(両否定形)

両否定形 $\neg X \Rightarrow \neg Y$ は，一般に非常に数が多く，効率的な抽出は困難である．また，ルールとしての有用性も低いため，本論文では両否定形を扱わない．上記の $\neg X$ はアイテム集合の否定表現であり，**負アイテム集合**と呼ぶ．以下では C_X はアイテム集合 X または負アイテム集合 $\neg X$ のどちらかを表すものとする．負アイテム集合および負ルールの**支持度** sup と**確信度** conf を以下のように定める [3]．

$$\sup(\neg X) = 1 - \sup(X) \tag{1}$$

$$\sup(X \Rightarrow \neg Y) = \sup(X) - \sup(X \cup Y) \tag{2}$$

$$\sup(\neg X \Rightarrow Y) = \sup(Y) - \sup(X \cup Y) \tag{3}$$

$$\mathrm{conf}(C_X \Rightarrow C_Y) = \frac{\sup(C_X \Rightarrow C_Y)}{\sup(C_X)} \tag{4}$$

以上のとき，以下の 5 つの条件を満たすルール $C_X \Rightarrow C_Y$ を**妥当 (valid) な負ルール**と呼ぶ [3, 5, 7]．

1. $X \cap Y = \emptyset$　　(独立性)
2. $\sup(X) \geq ms$ かつ $\sup(Y) \geq ms$
   (前件と後件の頻出性)
3. $\sup(X \Rightarrow Y) < ms$　　(無矛盾性)
4. $\sup(C_X \Rightarrow C_Y) \geq ms$　　(ルールの頻出性)
5. $\mathrm{conf}(C_X \Rightarrow C_Y) \geq mc$　　(ルールの確信度)

条件 1 は前件と後件の独立性の条件である．条件 3 は，無矛盾性の条件であり，先行研究 [7] で提案されたものである．負ルール $X \Rightarrow \neg Y$ や $\neg X \Rightarrow Y$ が抽出されたとき，正ルール $X \Rightarrow Y$ が同時に抽出される可能性を排除する条件である．

2.3 極小生成子を用いた負ルール集合の圧縮

先行研究 [8, 9] の負ルール集合の圧縮表現と抽出手法について概説する．アイテム集合 X に対して，$X \subset X'$，$X \neq X'$ かつ $\sup(X)$=$\sup(X')$ を満たす X' が存在しないならば，X を**飽和アイテム集合** (*closed itemset*) と呼ぶ．また X が X' に対して，$X \subseteq X'$ かつ $\sup(X')$ = $\sup(X)$ を満たすならば，X を X' の**生成子** (*generator*) と呼ぶ．生成子は一般には複数存在するので，その中でより小さな生成子が存在しないものを**極小生成子** (*minimal generator*) [10] と呼ぶ．

頻出アイテム集合や正のルールの圧縮には飽和アイテム集合がよく用いられているが，負ルールの圧縮に用いた場合，本来抽出すべき負ルールが表現できなくなる現象が生じる [8]．先行研究 [8] では，飽和アイテム集合ではなく極小生成子を用いることで妥当な負ルールの集合を圧縮できることを示した．

例を用いて負ルールの圧縮について説明を行う．

例 1　表 1 にデータベースを示す．以下では，アイテム集合 $\{A_1, ..., A_i\}$ は簡単化のため，$A_1...A_i$ と表す．また，アイテム集合 $A_1...A_i$ の頻度が $\boldsymbol{S}$ であることを $A_1...A_i : \boldsymbol{S}$ と表している．ここでは，右否定形のみを扱っていく．ms を 0.4，mc を 0.4 としたとき，前件を A と固定すると表 1 からは，表 2 に示す 6 個の妥当なルールが抽出される．飽和アイテム集合を用いて表 2 のルール集合を圧縮していく．表 1 における頻出アイテム集合は，$A:3$, $B:6$, $C:3$, $D:3$, $AB:3$, $BC:3$, $BD:3$, $CD:3$, $BCD:3$ である．このとき，飽和アイテム集合は，$AB:3, B:6, BCD:3$ であるので，表 2 の 6 つのルールは 1 つの $AB \Rightarrow \neg BCD$ で表現することができる．しかし，圧縮後のルール $AB \Rightarrow \neg BCD$ は，2.2 で述べた負ルール条件の独立性条件を満たさないため，妥当なルールではない．そこで，極小生成子を用いて 6 個の負ルールを圧縮することを考える．表 1 における極小生成子は $A:3$, $B:6$, $C:3$, $D:3$ であるので，表 2 の 6 つのルールは $A \Rightarrow \neg C$ と $A \Rightarrow \neg D$ の 2 つのルール集合で表現することができる．この 2 つのルールはどちらも妥当な負ルールである．負ルール集合の圧縮には極小生成子を用いることが適切である．

表 1: データベース

TID	アイテム集合
1	AB
2	AB
3	BCD
4	AB
5	BCD
6	BCD

表 2: 抽出ルール例

No	抽出ルール
1	$A \Rightarrow \neg C$
2	$A \Rightarrow \neg D$
3	$A \Rightarrow \neg BC$
4	$A \Rightarrow \neg CD$
5	$A \Rightarrow \neg BD$
6	$A \Rightarrow \neg BCD$

2.4 極小生成子を用いた負ルール抽出

先行研究 [9] のルール抽出アルゴリズムは, 極小生成子から**接尾辞木** [11] を作成し, 左優先深さ優先探索を行い, 負ルール抽出を行う.

定理 1[9] D をデータベースとする. MG が D 上のあるアイテム集合 A の極小生成子であるならば, MG の任意の真部分集合 $MG'(\subset MG)$ は, $B \neq A$ なる, あるアイテム集合 B の D 上の極小生成子となる.

定理 1 より, 極小生成子の集合から接尾辞木が必ず作成できることに注意していただきたい. 以下に, 先行研究 [9] における, 妥当な負ルール抽出アルゴリズムの大枠を示す. 擬似コード中の Check_Rule(X, Y) で $X \Rightarrow \neg Y$ と $\neg Y \Rightarrow X$ が妥当なルールであるかチェックしている. また, 同時に接尾辞木の性質などを用いて適宜枝刈りを行い, 探索の効率化を行っている. ルール抽出規則 Check_Rule(X, Y) での主な工夫点は, 文献 [9] を参照いただきたい.

Algorithm 1 負の相関ルールの大枠

```
    トランザクションデータベースから極小生成子（MGS）
    の集合を抽出し, N を抽出した極小生成子の総数とする;
1:  MGS の集合から接尾辞木を生成;
2:  接尾辞木上で左優先深さ優先探索で極小生成子を MGS_1,
    ... ,MGS_N の順に並べる;
3:  for i = 1 to N do
4:      X = MGS_i
5:      for j = 1 to N do
6:          Y = MGS_j
7:          Check_Rule(X, Y)
8:      end for
9:  end for
```

3 飽和集合からの極小生成子の抽出

本章では, 飽和アイテム集合から極小生成子を効率的に抽出する手法について考察する. ボトムアップ型とトップダウン型の 2 種類の抽出手法について研究を進める. 以下は定義より, 容易に示せる.

命題 1[12] ある飽和アイテム集合 X に対して, Y がその極小生成子である場合, Y を真に含む X の部分集合 X' は X の極小生成子になり得ない.

命題 2[12] 飽和アイテム集合 X の極小生成子 Y の真の部分集合 Y' は X の極小生成子となり得ない.

ボトムアップ型抽出は, 命題 1 に基づいた枝刈りを行う. 極小生成子のアイテム数が小さい場合には, ボトムアップ型手法は極小生成子を高速に抽出することができると期待される. トップダウン型は, 飽和アイテム集合の要素を 1 つづつ順に削除しながら部分集合を探索, 検査するため, 命題 2 に基づいた枝刈りを行う. 飽和アイテム集合と極小生成子の大きさがあまり変わらないとき, トップダウン型手法は極小生成子を高速に抽出することができると期待される. 以下で, それぞれの手法を詳しく説明していく.

3.1 ボトムアップ型アルゴリズム

極小生成子を飽和アイテム集合からボトムアップ型で抽出する手法について考える. ボトムアップ型では, 飽和アイテム集合 X の部分集合 X' で $|X'| = 1$ となるものから検査を行う. このとき, 以下の補題を利用して枝刈りを行う.

補題 1 飽和アイテム集合 X に対して, $X' \subset X$ かつ $|X'| = 1$ なる X' が X の極小生成子であるならば, X の他の極小生成子は全て差集合 $X - X'$ に含まれる.

(証明) 命題 1 より容易に証明できるので省略. ■

補題 1 を, 以下の例で説明する.

例 2 表 3 のデータベース上の飽和アイテム集合の 1 つに $ABCD : 2$ がある. $ABCD$ の極小生成子は $C : 2$, $AD : 2$, $BD : 2$ である. $ABCD$ と C の差分を取ると ABD となり, 残る 2 つの極小生成子 AD, BD は ABD の部分集合となっている.

表 3: データベース

TID	アイテム集合
1	ABCD
2	ABCD
3	AB
4	A
5	B
6	D

しかし, X の極小生成子 X' の大きさが $|X'| \geq 2$ の場合は, 補題 1 の性質は成り立たないことに注意して

ほしい．$ABCD$ と AD の差分を取ると BC となり，極小生成子 BD が含まれない．

例 2 より，$|Y| \geq 2$ の極小生成子では，差分を用いた枝刈りを行うことができないことが分かる．

図 1 に表 3 の飽和アイテム集合 $ABCD$ に対する束構造，図 2 にボトムアップ型による探索空間を示す．図 1 と 2 における青背景は飽和アイテム集合を示し，赤背景は極小生成子を示している．図 2 において，C が極小生成子であることから，補題 1 によって，$ABCD$ の部分集合で C が含まれる集合の枝刈りを行える．$k \geq 2$ の部分集合の探索は C を含まない集合のみ行っていることが分かる．ボトムアップ型アルゴリズムを **Algorithm 2** に示す．

Algorithm 2 ボトムアップ型アルゴリズム

Require: 3 項組 $< i, X_i, f_i >$ の族 $\mathcal{CS} = \{< 1, X_1, f_1 >, \ldots, < n, X_n, f_n >\}$．但し，各 i は識別番号，X_i は飽和アイテム集合，f_i は X_i の出現頻度である．
Ensure: $\mathcal{CS}$ の各飽和集合（識別番号 i）に対する極小生成子 M_j との頻度の 3 項組 $< M_j, i, f_i >$ の集合 $\mathcal{MG}$

```
MG ← ∅                                  ▷ MG は，極小生成子の族を格納する
TCS ← ∅                                 ▷ 飽和アイテム集合の差分を保持する
for each <i, X_i, f_i> ∈ CS do
    Y'' ← X_i
    for each Y's.t. Y' ⊆ X_i かつ |Y'| = 1 do
        if f_i = sup(Y') then
            MG ← MG ∪ {<Y', i, f_i>}
            Y'' ← Y'' − Y'
        end if
    end for
    TCS ← TCS ∪ {<i, Y'', f_i>}
end for
k ← 2
while 1 do                              ▷ break するまで繰り返す
    flag ← 0                            ▷ フラグの初期化
    for each <i, Z_i, f_i> ∈ TCS do
        for each Y's.t.Y' ⊆ Z_i かつ |Y'| = k do
            if Check_MG(Y', k)= true then
                if f_i = sup(Y') then   ▷ Y' が生成子である
                    MG ← MG ∪ {<Y', i, f_i>}
                    flag ← 1            ▷ フラグの設定
                end if
            else
                do_noting
            end if
        end for
    end for
    if flag = 0 then                    ▷ 要素数 k の極小生成子が存在しない
        break                           ▷ while ループから抜ける
    else
        k++
    end if
end while
return (MG)
```

Algorithm 2 では，まず要素数 $k = 1$ の部分集合を生成し，生成した部分集合に対して極小生成子であるか検査する．部分集合が生成子であれば極小生成子として追加し，飽和アイテム集合から差分をとる．次に，k を 1 つづつ増やし極小生成子を抽出し，$|Y| = k$ となる極小生成子が 1 つも存在しなくなるまで検査を行う．

定理 1 より，要素数 k の極小生成子が存在しない場合，k より大きい極小生成子が存在しないことが保証される．このことから，擬似コード中の $flag$ を用いて，**while** 文を制御する．$flag$ は要素数 k の極小生成子が存在するか判定するためのフラグであり，1 ならば要素数 k の極小生成子が 1 つ以上存在することを示し，0 ならば要素数 k の極小生成子が存在しないことを示す．また，定理 1 より，アイテム集合 Y' が極小生成子になる場合，Y' の部分集合は全て極小生成子になることが保証される．このことから，擬似コード中の関数 Check_MG によって，極小生成子になり得ないアイテム集合の頻度計算を回避している．関数 Check_MG は，アイテム集合 Y' の要素数 $k-1$ の部分集合がすでに他の飽和アイテム集合の極小生成子として登録されているか判定する関数である．Y' の要素数 $k-1$ の部分集合がすべて他のアイテム集合の極小生成子であるときアイテム集合 Y' が飽和アイテム集合 X_i の極小生成子になり得る $(true)$ となり，それ以外のときは，Y' は X_i の極小生成子にはならない $(false)$ となる．

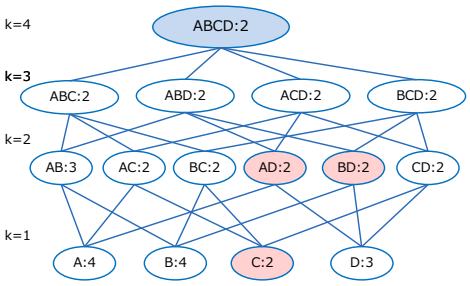

図 1: 束構造

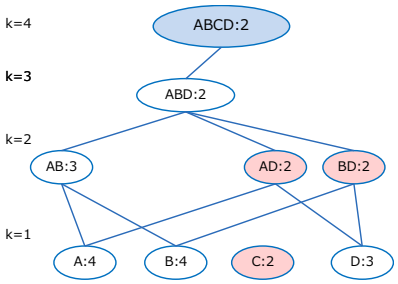

図 2: ボトムアップ型における探索空間

ボトムアップ型探索で極小生成子の抽出を行うため，$|Y| = k$ の極小生成子を探索する時，すでに $|Y| < k$ の極小生成子についてはすべて抽出されている．極小生成子のアイテム数が小さい場合には，ボトムアップ型で効率よく探索することができると推測される．

3.2 トップダウン型アルゴリズム

トップダウン型では，飽和アイテム集合 X の要素数を小さくしながら，極小生成子の検査を行うアルゴリズ

ムである．要素数を減らしながら部分集合の探索を行うため，命題 2 による枝刈りを自然に行うことができる．図 3 にトップダウン型による探索空間を示す．トップダウン型アルゴリズムを **Algorithm 3** に示す．

Algorithm 3 トップダウン型アルゴリズム

Require: 3 項組 $< i, X_i, f_i >$ の族 $\mathcal{CS} = \{< 1, X_1, f_1 >, \ldots, < n, X_n, f_n >\}$．但し，各 i は識別番号，X_i は飽和アイテム集合，f_i は X_i の出現頻度である．
Ensure: $\mathcal{CS}$ の各飽和集合（識別番号 i）に対する極小生成子 M_j との頻度の 3 項組 $< M_j, i, f_i >$ の集合 $\mathcal{MG}$

```
MG ← ∅                                  ▷ MG は，極小生成子の族を格納
S ← ∅                                   ▷ S は，スタック構造で生成子の候補を格納
Y ← ∅                                   ▷ Y は，S から取り出すアイテム集合を格納
HashTable ← ∅                           ▷ 訪問した頂点を保持するハッシュ表
for each < i, X_i, f_i > ∈ CS do
    X_i を S に追加
    while S ≠ ∅ do
        Flag ← true                     ▷ 極小生成子フラグの初期化
        S から取り出し Y に登録する
        if Y の要素数が 1 より大きい then
            for each Y' s.t. Y' ⊆ X_i かつ |Y'| = |Y| − 1 do
                key ← calcHashkey(Y')
                if HashTable[key]= NULL then
                    HashTable[key][0] ← Y'
                                        ▷ 探索したアイテム集合を格納
                end if
                if HashTable[key][1] ≠ i then
                                        ▷ Y' を i 番目で未探索
                    HashTable[key][1] ← i
                                        ▷ 探索時の飽和集合の識別番号を保持
                    HashTable[key][2] ← 0   ▷ ラベルの初期化
                    if f_i = sup(Y') then   ▷ Y' が生成子
                        HashTable[key][2] ← 1 ▷ ラベルの設定
                        Flag ← false
                                        ▷ 極小生成子フラグの設定
                        if |Y'| ≠ 1 then
                            Y' を S に格納する
                        else if |Y'| = 1 then
                            MG ← MG ∪ {<Y', i, f_i>}
                        end if
                    end if
                else                    ▷ Y' をすでに i 番目で探索済
                    if HashTable[key][2] = 1 then
                                        ▷ Y' が生成子
                        Flag ← false
                                        ▷ 極小生成子フラグの設定
                    end if
                end if
            end for
        else
            do_nothing
        end if
        if Flag = ture then             ▷ Y が極小生成子である
            MG ← MG ∪ {<Y, i, f_i>}
        end if
    end while
end for
return (MG)
```

Algorithm 3 では，まず飽和アイテム集合 X_i をスタックに格納する．スタックからアイテム集合 Y を取り出し，深さ優先探索を行う．アイテム集合 Y の部分集合 Y' がすべて生成子でなければ，アイテム集合 Y を X_i の極小生成子として登録する．

擬似コード中の $Flag$ はアイテム集合 Y が極小生成子か判定するためのフラグであり，$true$ ならば Y が X_i の極小生成子であることを示し，$false$ ならば Y が X_i の極小生成子ではないことを示す．また，擬似コード中の HashTable は探索を行ったアイテム集合を格納するハッシュ表であり，部分集合の頻度再計算を抑制するために使用する．以下に HashTable の詳細を示す．

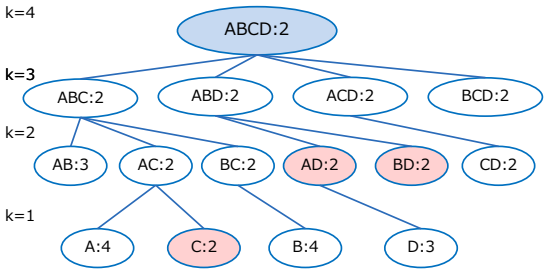

図 3: トップダウン型における探索空間

- HashTable[key][0] にアイテム集合を格納．
- HashTable[key][1] に訪問時の飽和アイテム集合の識別番号を格納．
- HashTable[key][2] に Y' が X_i の生成子か否かを判定するラベルを格納．

生成子判定ラベルを用いることで，HashTable[key][1]= i のときの頻度計算を 1 回だけに抑えている．通常，生成子判定ラベルを用いるためには，飽和アイテム集合ごとにハッシュ表を初期化する必要がある．しかし，初期化は計算時間がかなりかかるので，ハッシュ表に訪問時の飽和アイテム集合の識別番号を同時に格納することで，ハッシュ表の初期化を 1 回だけに抑えている．即ち，図 3 において，図 1 の AB の頻度計算は，Y が ABC の時にのみ行われていることが分かる．

小さい極小生成子の探索においては，ボトムアップ型より劣ると予想されるが，極小生成子が大きい場合，即ち，飽和アイテム集合よりわずかに小さい場合には，トップダウン型が効率的であると考える．実証実験にて効果の検証を行う．

3.3 前処理ありトップダウン型アルゴリズム

前処理ありトップダウン型アルゴリズムでは，補題 1 の枝刈りを前処理として行うものである．はじめに，飽和アイテム集合 X の $|X'| = 1$ となる部分集合 X' が極小生成子となるか検査を行う．極小生成子が存在した場合，飽和アイテム集合 X から極小生成子 X' の差分を取る．この処理によって，X' を含む部分集合の探索を抑制する．その後は，トップダウン型アルゴリズム同様に，飽和アイテム集合 X の要素数を小さくしながら，極小生成子の検査を行う．ボトムアップ型の長所を取り入れた手法であり，効率的に探索を行えると考える．

実証実験にて効果の検証を行う．図 4 に前処理ありトップダウン型による探索空間を示す．前処理ありトップダウン型アルゴリズムを **Algorithm** 4 に示す．

Algorithm 4 前処理ありトップダウン型アルゴリズム

Require: 3 項組 $< i, X_i, f_i >$ の族 $\mathcal{CS} = \{< 1, X_1, f_1 >, \ldots, < n, X_n, f_n >\}$．但し，各 i は識別番号，X_i は飽和アイテム集合，f_i は X_i の出現頻度である．
Ensure: $\mathcal{CS}$ の各飽和集合（識別番号 i）に対する極小生成子 M_j との頻度の 3 項組 $< M_j, i, f_i >$ の集合 $\mathcal{MG}$
==============================

▷ ————————
1: **Algorithm 3** の 1 ∼ 4 行と同処理を実行
▷ ————————
2: **for each** $< i, X_i, f_i > \in \mathcal{CS}$ **do**
3: $Flag \leftarrow true$ ▷ 極小生成子フラグの初期化
4: $Y'' \leftarrow X_i$
▷ ————ここから前処理————
5: **for each** Y' s.t. $Y' \subseteq X_i$ かつ $|Y'| = 1$ **do**
6: **if** $f_i = \sup(Y')$ **then** ▷ Y' が極小生成子であるとき
7: $\mathcal{MG} \leftarrow \mathcal{MG} \cup \{<Y', i, f_i>\}$
8: $Flag \leftarrow false$ ▷ 極小生成子フラグの設定
9: $Y'' \leftarrow Y'' - Y'$
▷ 極小生成子の候補集合から Y' を除く
10: **end if**
11: **end for**
▷ ————ここまで前処理————
12: **if** $Flag = false$ **then** ▷ 要素数 1 の極小生成子が存在
13: **if** $f_i \neq \sup(Y'')$ **then** ▷ Y'' が生成子でない
14: **break**
15: **end if**
16: **end if**
17: Y'' を S に追加
18: **while** $S \neq \emptyset$ **do**
▷ ————————
19: **Algorithm 3** の 8 ∼ 9 行と同処理を実行
▷ ————————
20: **if** Y の要素数が 2 より大きい **then**
21: **for each** Y' s.t. $Y' \subseteq X_i$ かつ $|Y'| = |Y| - 1$ **do**
▷ ————————
22: **Algorithm 3** の 12 ∼ 15 行と同処理を実行
▷ ————————
23: **if** HashTable$[key][1] \neq i$ **then**
24: HashTable$[key][1] \leftarrow i$
▷ 探索時の飽和集合の識別番号を保持
25: HashTable$[key][2] \leftarrow 0$ ▷ ラベルの初期化
26: **if** $f_i = \sup(Y')$ **then** ▷ Y' が生成子
27: HashTable$[key][2] \leftarrow 1$ ▷ ラベルの設定
28: $Flag \leftarrow false$
▷ 極小生成子フラグの設定
29: **if** $|Y'| \neq 2$ **then**
30: Y' を S に格納する
31: **else if** $|Y'| = 2$ **then**
32: $\mathcal{MG} \leftarrow \mathcal{MG} \cup \{<Y', i, f_i>\}$
33: **end if**
34: **end if**
35: **else** ▷ Y' をすでに i 番目で探索済
▷ ————————
36: **Algorithm 3** の 29 ∼ 31 行と同処理を実行
▷ ————————
37: **end if**
38: **end for**
39: **else**
40: **do_nothing**
41: **end if**
▷ ————————
42: **Algorithm 3** の 37 ∼ 39 行と同処理を実行
▷ ————————
43: **end while**
44: **end for**
45: **return** $(\mathcal{MG})$

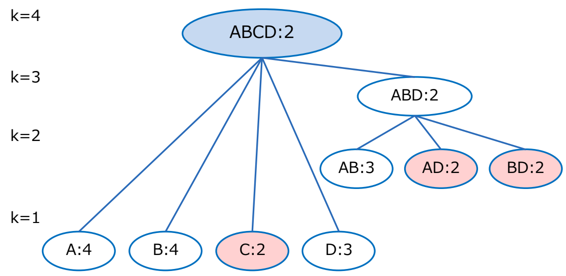

図 4: 前処理ありトップダウン型における探索空間

Algorithm 4 では，前処理で，要素数 1 の部分集合の処理が完了しているため，17 行目以降のトップダウン型探索では，スタックには $|Y| > 2$ となる部分集合 Y を格納する．

4　実験と考察

実験には，Frequent Itemset Mining Dataset Repository [13] から 4 種のデータセットを使用した．各データセットの詳細を表 4 と表 5 に示す．#(item) はデータセットに含まれるアイテムの種類数，#(trans) はデータセット中のトランザクションの総数，ave(item) は 1 トランザクション中に出現するアイテムの平均数である．#(FIS) は頻出アイテム集合の総数，#(CIS) は頻出飽和アイテム集合の総数，#(MG) は極小生成子の総数である．

表 4: 実験に使用したデータセットの概要

データセット	#(item)	#(trans)	ave(item)
retail	16,470	88,162	10.3
mushroom	119	8,124	23
connect	129	67,557	43
pumsb	7,117	49,046	74

最小支持度 ms を変化させたときの飽和アイテム集合 (CIS) と抽出した極小生成子（MG）の最大アイテム数，最小アイテム数，平均アイテム数を表 6 に示す．表 7 は，ボトムアップ型，トップダウン型，前処理ありトップダウン型アルゴリズムで飽和アイテム集合から極小生成子を抽出した際の計算時間を示したものである．手法 1 はボトムアップ型，手法 2 はトップダウン型，手法 3 は前処理ありトップダウン型である．

表 7 より，全ての場合で前処理ありトップダウン型手法 (手法 3) は一番多くの計算時間がかかっていることがわかる．これは，要素数 1 の極小生成子の全てを先に探索するオーバーヘッドが大きいためと考えられる．

表 5: 抽出数の概要

データセット	ms	#(FIS)	#(CIS)	#(MG)
retail	0.0005	19,242	19,144	19,144
	0.001	7,589	7,572	7,572
mushroom	0.1	574,431	4,885	7,615
	0.2	53,663	1,197	1,607
connect	0.6	21,250,671	68,343	68,343
	0.7	4,129,839	35,875	35,875
	0.8	533,975	15,107	15,107
	0.9	27,127	3,486	3,486
pumsb	0.7	2,698,265	241,196	658,379
	0.75	672,390	101,047	248,298
	0.8	142,156	33,295	67,835
	0.85	20,527	8,509	13,789

疎なデータセットである retail に関しては, ボトムアップ型手法 (手法 1) とトップダウン型手法 (手法 2) で極小生成子の抽出時間に殆ど差がない. 密なデータである mushroom, connect に関しては, ボトムアップ型手法の方が高速に抽出することができている. それは, 表 6 から分かる通り, 飽和アイテム集合と極小生成子の平均アイテム数の差が大きい, 即ち, 極小生成子が小さいためと考えられる. また, pumsb に関しては, トップダウン型手法の方が高速である. それは, 表 6 から分かる通り, 飽和アイテム集合と極小生成子の平均アイテム数に差が殆どない, 即ち, 極小生成子が大きいためと考えられる.

次に, 抽出計算における頻度計算の時間を表 8 に示す. 表 8 から, トップダウン型手法では頻度計算に多くの時間がかかり, ボトルネックになっていることが分かる. 実装システムでは, 頻度計算の高速化のために, 垂直フォーマット形式 [14] を用いているが, トップダウン型手法では, 大きなアイテム集合から探索を行うため, どうしても頻度計算に膨大な時間が必要になっていると思われる. よって, 頻度計算を行わない手法を開発できれば, さらなる高速化が期待できる.

本論文では, 頻度計算を行わない極小生成子抽出手法を実現するためのアイデアの提案のみを行う. トランザクションデータベース D 中の飽和アイテム集合の集合を CS と表記し, CS_X を D 中の飽和アイテム集合であり, 飽和アイテム集合 X を含まない集合の集合と定義する. このとき, 以下の補題が成り立つ.

補題 2 D をデータベース, X を D 上の飽和アイテム集合とする. このとき, アイテム集合 Y が D 上の X の極小生成子であることと, 以下の 3 つの条件を満たすことは同値である.

1. $Y \subseteq X$
2. 任意の飽和アイテム集合 $Z \in CS_X$ に対し, $Y \not\subseteq Z$ である.
3. 上の 1 と 2 を満たすアイテム集合 Y' の中で, Y は極小である.

(証明) 紙面の都合上省略する. ■

なお, 上の条件 3 は「Y の任意の真部分集合は条件 2 を満たさない. 」と等価である. この補題を用いることで, 頻度計算を行わないで極小生成子の抽出が行えるが, その効率的な計算手法については, 今後検討が必要である.

表 6: 飽和集合と極小生成子のサイズの概要

データセット	ms		最大	最小	平均
retail	0.0005	CIS	6	1	2.2599
		MG	6	1	2.2532
	0.001	CIS	5	1	2.0704
		MG	5	1	2.0681
mushroom	0.1	CIS	16	1	6.8061
		MG	7	1	3.9241
	0.2	CIS	15	1	5.7285
		MG	7	1	3.4692
connect	0.6	CIS	20	1	12.1706
		MG	8	1	5.3307
	0.7	CIS	18	1	11.0001
		MG	7	1	5.1645
	0.8	CIS	15	1	9.3439
		MG	7	1	4.9153
	0.9	CIS	12	1	7.0195
		MG	7	1	4.4432
pumsb	0.7	CIS	18	1	9.1607
		MG	14	1	8.1535
	0.75	CIS	15	1	8.1280
		MG	12	1	7.2420
	0.8	CIS	14	1	6.9548
		MG	11	1	6.2035
	0.85	CIS	10	1	5.7281
		MG	9	1	5.1920

5 まとめ

本研究では, 極小生成子の飽和アイテム集合からの高速な抽出手法を検討した. ボトムアップ型とトップダウン型の 2 種類の極小生成子抽出手法の比較を行い, 高速な抽出手法を提案した. 実験結果より, おおむねボトムアップ型手法が優れていることが確認できた.

表 7: 抽出時間

データセット	ms	手法 1 [s]	手法 2 [s]	手法 3 [s]
retail	0.0005	0.03	0.04	0.09
	0.001	0.01	0.02	0.01
mushroom	0.1	0.37	2.22	2.37
	0.2	0.03	0.14	0.14
connect	0.6	264.86	1,204.01	1,283.02
	0.7	40.82	110.52	129.53
	0.8	4.73	8.35	11.20
	0.9	0.14	0.20	0.31
pumsb	0.7	4,194.23	1,032.14	1,580.21
	0.75	96.08	35.76	114.53
	0.8	2.86	2.78	9.90
	0.85	0.18	0.20	0.57

表 8: 抽出時間に占める頻度計算時間

データセット	ms	手法 1		手法 2	手法 3
connect	0.6	264.86		1,204.01	1,283.02
		頻度計算	4.93	869.15	952.67
	0.7	40.82		110.52	129.53
		頻度計算	1.11	91.06	110.54
pumsb	0.7	4,194,23		1,032.14	1,580.21
		頻度計算	51.37	484.62	1,023.28
	0.75	96.08		35.76	114.53
		頻度計算	8.75	27.33	106.31

極小生成子抽出において, 頻度計算がボトルネックになっていることから, 今後の課題として, 支持度計算を用いない極小生成子抽出手法の開発がある.

謝辞

本研究で使用したプログラムは, 本研究室 OB 黒岩健歩氏が作成したプログラムを基盤として開発したものである [15]. 本研究は一部，ISPS 科学研究費補助金（No.16K00298）および JST さきがけの援助を受けている.

参考文献

[1] R. Agrawal and R. Srikant: Fast Algorithm for Mining Association Rules. *Proc. VLDB*, pp.487-499, (1994)

[2] J. Han, J. Pei and Y. Yin: Frequent Patterns without Candidate Generation. *Proc. ACM-SIGMOD' 00*, pp.1-12, (2000)

[3] X. Wu, C. Zhang, and S. Zhang: Efficient Mining of Both Positive and Negative Association Rules. *ACM Trans. on Information Systems*, Vol.22(3), pp.381-405, (2004)

[4] L. Zhou and S. Yau: Efficient Association Rule Mining among Both Frequent and Infrequent Items. *Computers and Mathematics with Applications*, Vol.54, pp.737-749, (2007)

[5] H. Wang, X. Zhang and G. Chen: Mining a Complete Set of Both Positive and Negative Association Rules from Large Databases. *Proc. the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining (PAKDD'08)*, pp.777-784, (2008)

[6] C. C. Aggarwal and P. S. Yu: A New Framework for Itemset Generation. *Proc. ACM SIGACT-SIGMOD-AIGART Symp. on Principles of Database Systems*, pp.18-24, (1998)

[7] 井出典子, 岩沼宏治, 山本泰生: 負の相関ルールを抽出する高速トップダウン型アルゴリズム, 人工知能学会論文誌, 29 巻 4 号, pp. 406-415, (2014)

[8] 岩沼宏治, 佐生隼一, 黒岩健歩, 山本泰生: 負の相関ルール集合の極小生成子に基づく圧縮表現, 情報処理学会論文誌, 57 巻 8 号, pp. 1845-1849, (2016)

[9] 谷島健斗, 岩沼宏治, 黒岩健歩, 佐生隼一, 山本泰生: 極小生成子を用いた負ルール抽出計算の効率化, 第 31 回人工知能学会全国大会, 4A1-3in1 , (2017)

[10] M. Zaki: Mining Non-Redundant Association Rules. *Data Mining and Knowledge Discovery*, Vol. 9, pp. 223-248, (2004)

[11] 亀谷由隆, 佐藤泰介: 最小サポート上昇法に基づく上位 k 関連パターンの発見, 人工知能学会データ指向構成マイニングとシミュレーション研究会, SIG-DOCMAS, B101-4, pp.(2-24)-(2-32), (2011)

[12] 佐生 隼一, 岩沼 宏治, 山本 泰生, 黒岩 健歩: 負の相関ルールマイニング効率化のための極小生成子の抽出計算, 人工知能学会第 103 回人工知能基本問題研究会, SIG-FPAI, B5-03, pp. 61-66, (2017)

[13] Frequent Itemset Mining Dataset Repository, <http://fimi.ua.ac.be/>(2017-2-28)

[14] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li: New Algorithms for Fast Discovery of Association Rules, *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining* (KDD'97) , pp. 283-296, (1997)

[15] 黒岩 健歩, 岩沼宏治, 山本泰生: 負相関ルールマイニングの高速化と関連性尺度の導入, 人工知能学会第 97 回人工知能基本問題研究会, SIG-FPAI, B4-04, pp. 7-12, (2015)