飽和集合上の極小生成子抽出アルゴリズム：支持度計算なし上昇型手法を中心とした考察

Extraction Algorithms of Minimal Generators from Closed Itemsets : Consideration Bottom-up Methods without Support Calculation

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

[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

Abstract: A fast extraction of the minimal generator plays a very important role for compressing the huge set of valid negative association rules. We propose a novel bottom-up algorithm that can efficiently extract all minimal generators from closed itemsets without support calculations. We also show some experimental results for evaluating our proposed methods.

1　はじめに

負の相関ルールマイニングの効率的な実装のためには, 極小生成子を高速に生成することが必要である. 本論文では, 支持度計算を行わず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. 既に提案した上昇型と下降型, 支持度計算なし下降型と本論文で新たに提案する支持度計算なし上昇型の 4 種類の極小生成子抽出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, $X \Rightarrow Y$ のように表し, 正の相関ルール [1] 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相関ルールでは表現が困難な共起関係を記述でき, 潜在的な関係を表現できるため極めて有用である [2, 3, 5]. 例えば, $e_1, e_2, e_3, \ldots, e_n, e_c$ を現在考慮しているすべてのアイテムとするとき, $n-1$ 個の正の相関ルール $\{e_2\} \Rightarrow \{e_c\}$, $\{e_3\} \Rightarrow \{e_c\}$, ... , $\{e_n\} \Rightarrow \{e_c\}$ 全体で表す性質は, ただ一つの負の相関ルール $\neg\{e_1\} \Rightarrow \{e_c\}$ でコンパクトに表現することが可能になる. 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. そのため岩沼ら [6] は, 極小生成子を用いた負ルール集合の可逆圧縮手法を提案した. さらに我々[9] は, 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を圧縮したままの形で抽出を行う効率的な計算手法を提案した. また [10] では, 極小生成子を飽和アイテム集合から効率的に抽出する下降型と上昇型の探索アルゴリズムを考察した. それらの計算上のボトルネックは, アイテム集合の支持度計算にあることが判明している. [11] では, 支持度計算を行わずに極小生成子を高速抽出する下降型の探索アルゴリズムを考察した. 支持度計算なし下降型手法では, 探索時にアイテム集合を全て展開するため, メモリを膨大に使用する. そこで, 本研究では, 支持度計算を行わずに極小生成子を高速抽出する上昇型手法を提案する.

本論文の構成は以下の通りである. 2 章は準備である. 3 章は本論文で新しく提案する支持度計算なし上昇型アルゴリズムを説明する. 4 章で実験結果と考察を示す. 5 章はまとめである.

*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 正負の相関ルール

$I = \{x_1, x_2, ..., x_n\}$ 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 α の**大きさ**を $|\alpha|$ と表記するとき，$|D(X)|$ を X の D 中の**支持度**と呼ぶ．X の D 中の**相対支持度 sup**(X) を $\sup(X) = \frac{|D(X)|}{|D|}$ と定義する．**最小支持度** ms と**最小確信度** mc とはユーザが相対支持度と確信度に関して与える閾値 (0 から 1 の間の実数値) である．$\sup(X) \geq ms$ を満たす X を**頻出アイテム集合**と呼ぶ．**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．

本論文では，**負の相関ルール** (以下，“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)

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．$\neg X$ は，X の各アイテムが同時に出現することはほとんどないことを示すものと考える．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**相対支持度** sup と**確信度** conf を以下のように定める [2].

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

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

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

$$\sup(\neg X \Rightarrow \neg Y) = 1 - \sup(X) - \sup(Y) + \sup(X \cup Y) \tag{4}$$

$$\mathrm{conf}(C_X \Rightarrow C_Y) = \frac{\sup(C_X \Rightarrow C_Y)}{\sup(C_X)} \tag{5}$$

このとき，以下の 4 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 [2, 3].

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. $\sup(C_X \Rightarrow C_Y) \geq ms$　(ルールの頻出性)
4. $\mathrm{conf}(C_X \Rightarrow C_Y) \geq mc$　(ルールの確信度)

2.2 極小生成子を用いた負ルール集合の圧縮と抽出

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*)[4] と呼ぶ．X に対して，$X \subseteq Y$ なる飽和アイテム集合 Y を X の**閉包** (*closure*) と呼び，$\mathrm{clo}(X)$ と表記する．飽和アイテム集合の最も重要な性質の一つとして，以下がある．

定理 1 (飽和集合の積演算の閉包性 [4]) X と Y が飽和アイテム集合ならば，その積 $X \cap Y$ も飽和アイテム集合．

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*) [4] と呼ぶ．極小生成子の重要な性質の一つとして，以下がある．

定理 2 (極小生成子の下方閉包性 [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とする．このとき，MG の任意の真部分集合 MG' は，D 上のあるアイテム集合 $B(\neq A)$ の極小生成子となる．

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 [6]．そこで，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の可逆圧縮ができることが分かっている [6].

定理 2 より，極小生成子の集合から接尾 (もしくは接頭) 辞木が必ず作れることを保証できる [9]．そこで，先行研究 [9] のアルゴリズムでは，極小生成子から**接尾辞木** [5] を作成し，左優先深さ優先探索を行いながら，妥当な負ルールを全て抽出する．先行研究 [9] の負ルール抽出アルゴリズムの大枠を **Algorithm 1** に示す．擬似コード中の $\mathrm{Check\_Rule}(X, Y)$ 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．詳細は，文献 [9] を参照いただきたい．

2.3 飽和集合からの極小生成子の抽出

以上から，負ルールの効率的な抽出には，極小生成子の高速な生成が非常に重要となる．極小生成子を用いて負ルール集合を圧縮した場合，その復元には対応する飽和アイテム集合 (即ち，閉包) が必要になる．その

Algorithm 1 極小生成子上の負の相関ルール抽出法

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;
MGS の集合から接尾辞木を生成;
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;
for $i = 1$ to N **do**
  $X \leftarrow MGS_i$
  for $j = 1$ to N **do**
    $Y \leftarrow MGS_j$
    Check_Rule(X, Y)
  end for
end for

ため，極小生成子は，あとの復元を考えて飽和アイテム集合から生成を行う．その計算は，図 2 に示すような，飽和アイテム集合の部分集合からなる束空間の探索によって行う．先行研究 [10] では，上昇型と下降型の 2 つの極小生成子抽出アルゴリズムを提案した．

命題 1 [8] 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 [8] 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．

上昇型手法では，命題 1 に基づいた探索空間の枝刈りを行う．極小生成子の大きさが小さい場合には，上昇型手法を用いて極小生成子を高速に抽出することができる．下降型手法では，飽和アイテム集合の要素を 1 つずつ順に削除しながら部分集合を探索するため，命題 2 に基づいた探索空間の枝刈りを行う．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，下降型手法を用いて極小生成子を高速に抽出することができる．図 1 に表 1 から抽出される飽和アイテム集合と対応する極小生成子を示す．図 2 に図 1 の飽和アイテム集合 $ABCD$ に対する束構造を示す．各頂点内の $X : k$ はアイテム集合 X の支持度が k であることを表している．図 3 に上昇型による探索空間，図 4 に下降型による探索空間を示す．図 2, 3 と 4 における網掛け背景は飽和アイテム集合を示し，破線は極小生成子を示している．抽出法の擬似コード等の詳細は，文献 [10] を参照いただきたい．

これらの上昇型，下降型手法では，各部分集合が (一番上の) 飽和アイテム集合の生成子であるか否かの判定を支持度を毎回計算して行っている．特に下降型においては，検査する部分集合が大きい場合が多いため，この支持度計算は大変重く，計算上のボトルネックになっている．そこで，[11] では，支持度計算を全く行わずに生成子か否か判定する下降型手法を提案した．以下の定理 3 が探索の基本となる．

定理 3 (生成子判定原理 [11]) 飽和アイテム集合 X とその任意の部分集合 Y に関して以下が同値である．

(1) $\sup(X) = \sup(Y)$，即ち Y は X の生成子である

(2) $Y \subseteq Y' \subsetneqq X$ なる飽和アイテム集合 Y' が存在しない

支持度計算なし下降型手法では，飽和アイテム集合をその大きさで昇順にソートし，小さい順に極小生成子を探索する．これにより，要素数 k の飽和アイテム集合 X を探索する際に，X の真部分集合となる飽和アイテム集合 Y の極小生成子の探索が必ず終了していることが保証される．この場合，定理 3 を用いて，支持度計算を行わずに，X の部分集合 Z が生成子であるか否か，即ち，X と Z の支持度が同じであるか否かの判定ができることになる．擬似コード等の詳細は，文献 [11] を参照いただきたい．この支持度なし下降型では，極小生成子を抽出する際に，頻出アイテム集合を全て探索するため，メモリを膨大に使用する．そこで，本論文では，下降型に比べてメモリの使用量を抑えることのできる上昇型の支持度計算なし手法を提案する．

表 1: データベース

TID	アイテム集合
1	ABCD
2	ABCD
3	BCD
4	DEF
5	DEF
6	C
7	E

飽和集合	-	極小生成子
C	-	C
D	-	D
E	-	E
BCD	-	B, CD
DEF	-	F, DE
ABCD	-	A

図 1: 飽和集合と対応する極小生成子

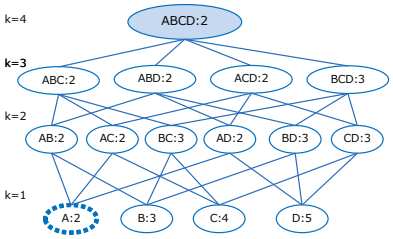

図 2: 束構造

3 支持度計算なし上昇型極小生成子抽出アルゴリズム

本章では，飽和アイテム集合から支持度計算を行わずに極小生成子を効率的に抽出する上昇型手法について考察する．

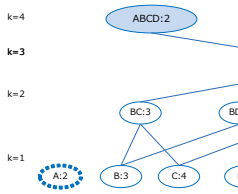

図 3: 上昇型の探索空間

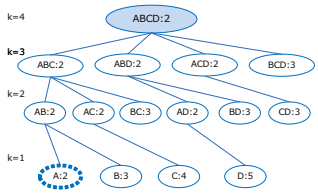

図 4: 下降型の探索空間

飽和アイテム集合をその大きさで昇順にソートし, 順に要素数 k の部分集合を探索することで, 飽和アイテム集合 X の要素数 k の部分集合探索時には, $X' \subset X$ なる飽和アイテム集合 X' の要素数 k の部分集合の探索はすでに終了していることが保証される. この場合, 以下の補題 1 と補題 2 を用いて, 支持度計算を行わずに, X の部分集合 Z が生成子であるか否か, 即ち, X と Z の支持度が同じであるか否かの判定ができることになる.

補題 1 飽和アイテム集合 X の部分集合 Y の大きさを $|Y| = k$ とする. このとき, 提案アルゴリズムの大きさ k の部分集合の探索 (**Algorithm 2** の 21 行目〜42 行目) において, Y の探索 (処理) が初めてであれば, Y は X の生成子である.

証明： Y が X の生成子ではないと仮定し, 背理法で証明する. このとき, 定理 3 より, 飽和アイテム集合 X の部分集合 Y の探索が初めてでないなら, Y は X の生成子にならず, 他の飽和アイテム集合 $X'(X' \subset X)$ の生成子になる. $Y \subseteq Y' \subsetneq X$ なる飽和アイテム集合 Y' が存在する. $|Y'| < |X|$ であるので, 提案アルゴリズムの構成から, X の前に Y' の大きさ k の部分集合が必ず探索される. よって, X の大きさ k の部分集合の探索処理の時点では, 必ず Y は探索処理されている. これは前提に矛盾する. よって, Y は X の生成子である. ■

補題 2 飽和アイテム集合 X の部分集合 Y が探索済みであるならば, Y は X の生成子ではない.

証明： Y が探索済みであるならば, 提案アルゴリズムの構成より, $Y \subseteq Z$ かつ $|Z| \leq |X|$ なる飽和アイテム集合 Z が存在する. 定理 1 より $Z \cap X$ は飽和アイテム集合であり, $Y \subseteq \{Z \cap X\} \subseteq X$ である. よって, Y は X の生成子ではない. ■

このことから, アイテム集合 Y がすでにハッシュ表に登録されていれば, X よりも小さい飽和アイテム集合 X' の部分集合になっていることが確認できるので, Y は X の生成子とはならないことが保証できる. 以上から, ハッシュ表の探索のみで生成子判定を行える.

表 2 に表 1 のデータベースから抽出される飽和アイテム集合と支持度を示し, 図 5 から図 7 に支持度なし上昇型手法における要素数 k の探索空間を示す. また, 図 8 に要素数 2 の極小生成子探索時の飽和アイテム集合 $ABCD$ の要素数 2 の部分集合を探索する前のハッシュ表で示し, 図 9 に要素数 2 の極小生成子探索時の飽和アイテム集合集合 $ABCD$ を探索した後のハッシュ表を示す. 支持度計算なし上昇型極小生成子抽出アルゴリズムを **Algorithm 2** に示す.

Algorithm 2 支持度計算なし上昇型極小生成子抽出アルゴリズム

Input: 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.
Output: $\mathcal{CS}$ 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: MG ← ∅                          ▷ MG は, 極小生成子の族を格納する
2: HashTable← ∅                    ▷ 探索した頂点を格納したハッシュ表
3: HashTable_MG← ∅
4: sortCS()                        ▷ CS をアイテム数で昇順にソートする
5: initHashTable()                 ▷ ハッシュ表の初期化
6: initHashTable_MG()              ▷ ハッシュ表の初期化
7: for each < i, X_i, f_i >∈ CS do
8:     Z ← X_i
9:     for each Y s.t. Y ⊆ X_i かつ |Y| = 1 do
10:        key ←calcHashkey(Y)     ▷ Y のハッシュ値の計算
11:        if HashTable[key]= NULL then
12:            HashTable[key]← Y
13:            key_mg ←calcHashkey_mg(Y)
14:            HashTable_MG[key_mg][0]← Y
15:            HashTable_MG[key_mg][1]← X_i
16:            MG ← MG ∪ {<Y, i, f_i>}
17:        end if
18:    end for
19: end for
20: k ← 2
21: while 1 do                     ▷ break するまで繰り返す
22:    flag ← 0   ▷ 要素数 k の極小生成子が存在するか判定するフラグの初期化
23:    for each < i, X_i, f_i >∈ TCS do
24:        for each Y s.t. Y ⊆ X_i かつ |Y| = k do
25:            key ←calcHashkey(Y)
26:            if HashTable[key]= NULL then
27:                HashTable[key]← Y
28:                if Check_MG(Y, k, X_i)= true then
29:                    13〜16 行目と同処理を実行
30:                    flag ← 1                ▷ フラグの設定
31:                else
32:                    do_nothing
33:                end if
34:            end if
35:        end for
36:    end for
37:    if flag = 0 then   ▷ 要素数 k の極小生成子が存在しない
38:        break                   ▷ while ループから抜ける
39:    else
40:        k + +
41:    end if
42: end while
43: return (MG)
```

Algorithm 2 では, まず事前に LCM[7] 等を用いて

表 2: 飽和集合のデータベース

飽和 ID	飽和集合	支持度
1	C	4
2	D	5
3	E	3
4	BCD	3
5	DEF	2
6	ABCD	2

抽出した飽和アイテム集合を大きさで昇順ソートする (表 2). ソート後に, 全ての飽和アイテム集合の要素数 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を検査する (図 5). 次に, k を 1 つず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 (図 6, 7). この例では, ハッシュ表に登録されている要素数 2 のアイテム集合は飽和アイテム集合 $ABCD$ の生成子にはならない. 飽和アイテム集合 $ABCD$ の要素数 2 の生成子は AB, AC, AD となる (図 7). 飽和アイテム集合 $ABCD$ の要素数 2 の部分集合探索後のハッシュ表は, 図 9 のようになる.

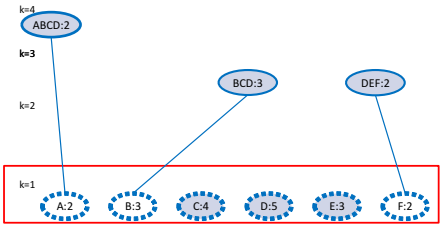

図 5: $k = 1$ の探索空間

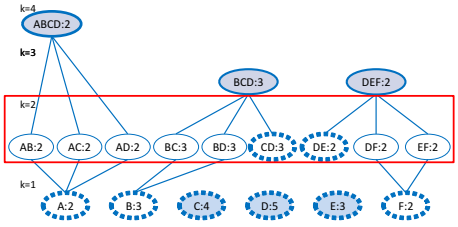

図 6: $k = 2$ の探索空間

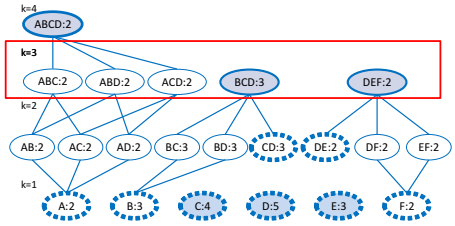

図 7: $k = 3$ の探索空間

定理 2 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 (37〜40 行目). $flag$ 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 1 つも存在しないことを示す.

以下の命題 3, 補題 3 を用いることで, Y よりも要素数が 1 つ小さい部分集合を見ることで極小生成子の判定を行うことができる.

命題 3 Z をアイテム集合とする. このとき, $|Y| = |Z| - 1$ なる Z の部分集合 Y で極小生成子出ないものがあるならば, Z は極小生成子にならない.

証明： 定理 2 より明らか. ■

補題 3 飽和アイテム集合 X の生成子 Y の部分集合 Y' が全て X とは別の飽和アイテム集合の極小生成子なら, Y は X の極小生成子になる.

証明： 全ての Y' が X とは別の飽和アイテム集合の極小生成子ならば, 支持度の逆単調性より, 必ず $\sup(Y) < \sup(Y')$ となる. よって定義より, Y が極小生成子となることは明らかである. ■

このことから, 擬似コード中の関数 Check_MG によって, Y が飽和アイテム集合 X の極小生成子か判定を行う. 関数 Check_MG は, アイテム集合 Y の要素数が 1 つ小さい部分集合が全て別の飽和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 の極小生成子になる ($true$), それ以外のときは, Y は飽和アイテム集合 X_i の極小生成子にはならない ($false$) となる (28 行目).

関数 Check_MG でアイテム集合 Y が X_i の極小生成子になり得るか高速に判定を行うために極小生成子を HashTable_MG に登録する (14, 15 行目). 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, アイテム集 Y が飽和アイテム集

	アイテム集合	大きさ
1	EF	2
2		
3	BC	2
4		
5	DF	2
6		
7	BD	2
8		
9	CD	2
10		
11	DE	2
12		
13		
14		
15		

	アイテム集合	大きさ
1	EF	2
2		
3	BC	2
4	AC	2
5	DF	2
6		
7	BD	2
8		
9	CD	2
10		
11	DE	2
12		
13	AB	2
14		
15	AD	2

図 8: 探索前のハッシュ表　図 9: 探索後のハッシュ表

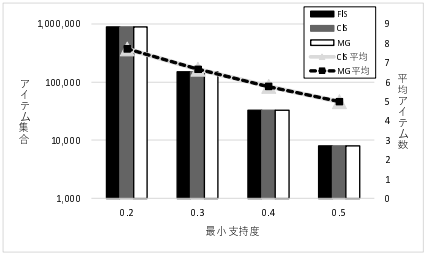

図 10: データの概要 (accidents)

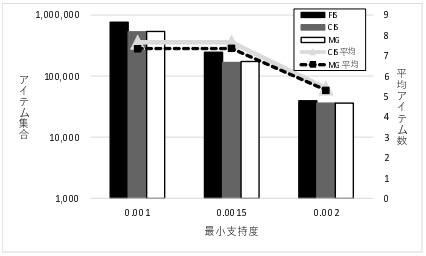

図 11: データの概要 (kosarak)

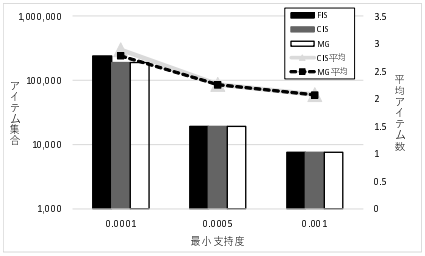

図 12: データの概要 (retail)

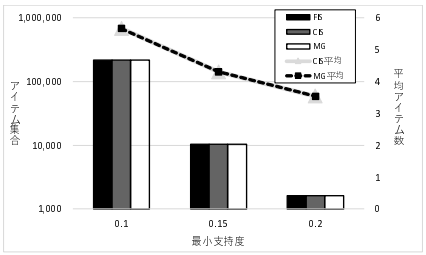

図 13: データの概要 (WebDocs)

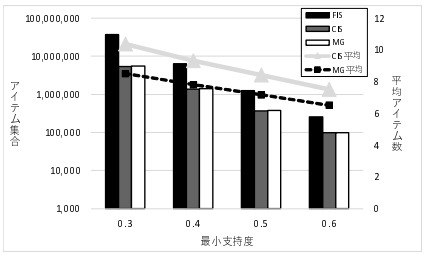

図 14: データの概要 (chess)

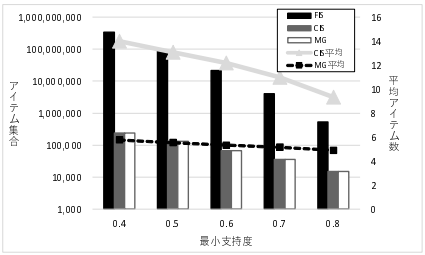

図 15: データの概要 (connect)

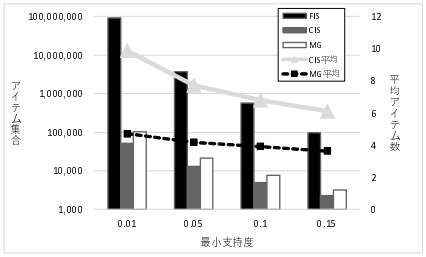

図 16: データの概要 (mushroom)

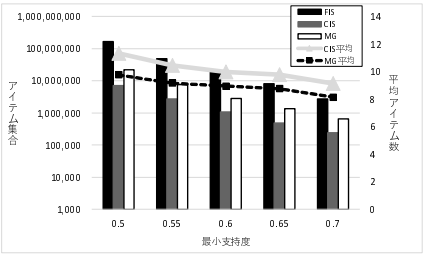

図 17: データの概要 (pumsb)

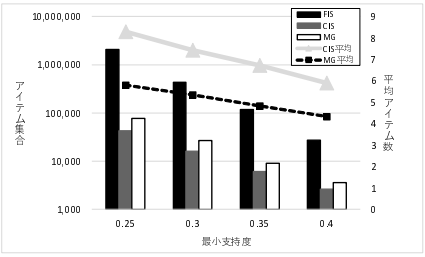

図 18: データの概要 (pumsb_star)

合 X_i の生成子か判定するために利用する. アイテム集合 Y の探索は $k = |Y|$ の時のみしか行わない. そこで, メモリの有効活用のためにハッシュ表に登録されているアイテム集合 Z が $k \neq |Z|$ ならば上書きする.

4 実験と考察

実験には, Frequent Itemset Mining Dataset Repository [12] 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 3 と図 10〜18 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. 表 3 の accidents, kosarak, retail, WebDocs は疎なデータである. それに対し, chess, connect, mushroom, pumsb, pumsb_star は密なデータである. 図 10〜18 に最小支持度を変化させた時の抽出される飽和アイテム集合数 (CIS), 頻出アイテム集合数 (FIS), 極小生成子 (MG) と飽和アイテム集合と極小生成子の平均アイテム数を示す. 図 10〜13 より, 疎なデータは飽和アイテム集合や極小生成子を用いた頻出アイテム集合の圧縮ができないことがわかる. また, 図 14〜18 より, 密なデータは飽和アイテム集合や極小生成子を用いた頻出アイテム集合の圧縮が有効なことがわかる.

図 19〜27 に上昇型, 下降型, 支持度なし上昇型, 支持度なし下降型で飽和アイテム集合から極小生成子を抽出した際の抽出時間を示したものである. 実験では, 探索頂点の最大数を 2^{27} 個とし, ハッシュ表のサイズは 2^{28} とした.

connect の最小支持度を 0.4 としたとき, 下降型と支持度計算なし下降型は, 探索頂点数が最大 (2^{27} 個) に達したため, 結果を得ることができていない (図 24). pumsb の最小支持度を 0.5 としたとき, 支持度計算なし上昇型以外の結果を得ることができない (図 26). 上昇型は 6 時間でタイムアウトしたためであり, 下降型はメモリが不足したためである. また, 支持度計算なし

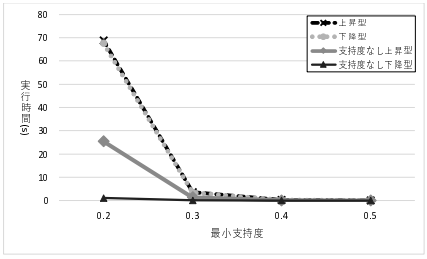

図 19: 実験結果 (accidents)

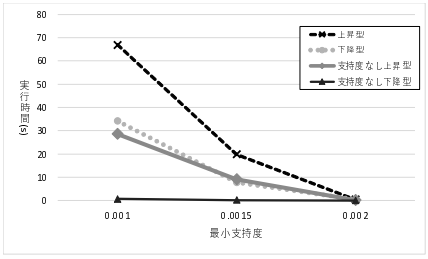

図 20: 実験結果 (kosarak)

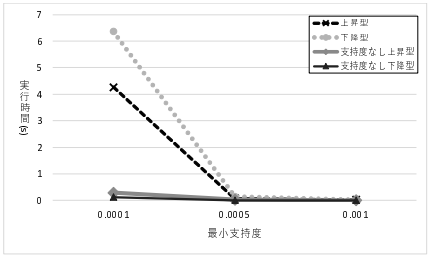

図 21: 実験結果 (retail)

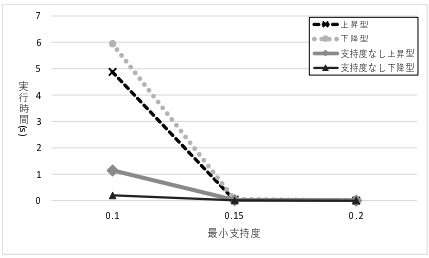

図 22: 実験結果 (WebDocs)

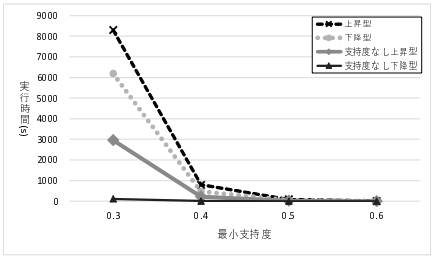

図 23: 実験結果 (chess)

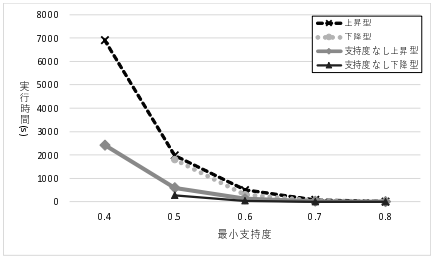

図 24: 実験結果 (connect)

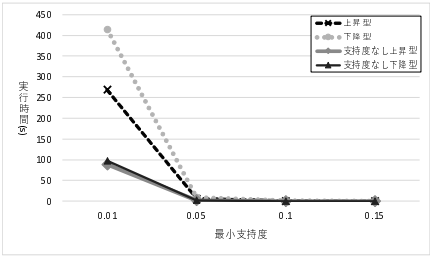

図 25: 実験結果 (mushroom)

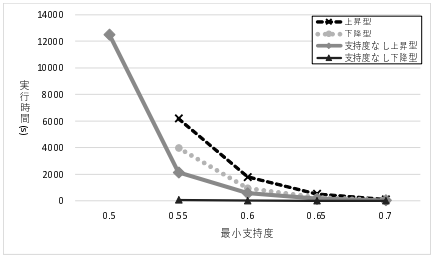

図 26: 実験結果 (pumsb)

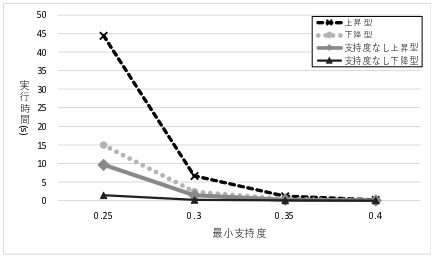

図 27: 実験結果 (pumsb_star)

表 3: 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)
accidents	468	340,183	33.8
kosarak	41,270	990,002	8.1
retail	16,470	88,162	10.3
WebDocs	5,267,656	1,692,082	177.2
chess	75	3,196	37
connect	129	67,557	43
mushroom	119	8,124	23
pumsb	2,113	49,046	74
pumsb_star	2,088	49,046	50.5

下降型は探索頂点数が最大に達したためである. 図 19 〜27 より, 全ての場合で, 支持度計算なし手法が支持度計算を行う手法より高速であることがわかる. これは, 支持度計算なし手法では, 部分集合が生成子であるか否かの判定をハッシュ表の探索のみで行えるためである. また, 支持度計算なし手法を比較した際, ほぼすべての場合で, 支持度計算なし下降型が高速であるが, connect と pumsb では, 閾値を低くした際, 支持度計算なし上昇型しか結果を得られていないことがわかる. これは, 上昇型では, 定理 2 より要素数 k の極小生成子が存在しない場合, 要素数 k 以上の探索を行わないためである.

データセット connect, msuhroom は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く (図 15, 16), 支持度なし上昇型が有利であるが, この connect, mushroom に対しても抽出時間に大きな差がないことが分かった. そこで, 図 28 と 29 に各データの生成子判定に用いるハッシュ表の探索回数とハッシュ登録時の衝突割合を示す. connect に着目すると, 支持度計算なし上昇型手法は, 支持度計算なし下降型手法に比べて, ハッシュ表の探索回数が多いことがわかる. これは, 上昇型では, 部分集合 Y を含むすべての飽和アイテム集合の時に探索を行うためである. また, 最小支持度が

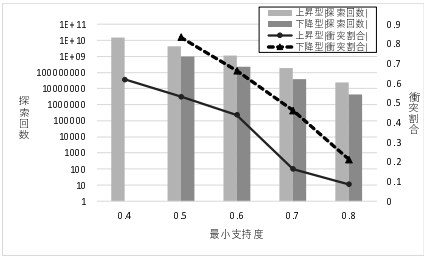

図 28: connect の詳細

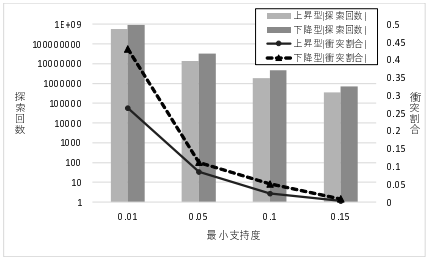

図 29: mushroom の詳細

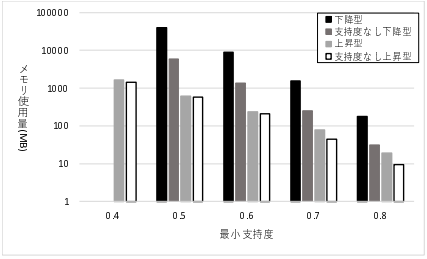

図 30: メモリ使用量 (connect)

0.5 のとき, 上昇型, 下降型共にハッシュ表の充填率が 0.5 以下であるが, ハッシュ表の衝突割合が 0.5 を超えている. このことより, 衝突の回数を減らすことが今後の課題である. mushroom に着目すると, 支持度なし上昇型手法は, 支持度計算なし下降型手法に比べて, ハッシュ表の探索回数が少ないことがわかる. しかし, 支持度計算なし上昇型では極小生成子か判定する際に極小生成子が登録されているハッシュ表をさらに探索する必要がある. そのため, 抽出時間に大きな差がないと考えられる.

また, データセット connect のメモリ使用量を図 30 に示す. 支持度計算なし手法が支持度計算を行うものよりメモリの使用量が少ないことがわかる. これは, 支持度計算なし手法では, 部分集合が生成子であるか否かの判定をハッシュ表のみで行えるためである. また, 支持度計算なし手法を比較した際, 支持度計算なし上昇型がメモリの使用量が少ない. これは, 支持度計算なし上昇型では, 定理 2 より, 要素数 k の極小生成子が存在しない場合, 要素数 k 以上の探索を行わないためである.

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支持度計算なし上昇型抽出手法を検討した. 支持度計算なし上昇型手法と既存の 3 つの手法と比較を行い, 実験結果より, おおむね支持度計算なし下降型手法が高速であることと支持度計算なし上昇型手法がメモリ使用量が少ないことが確認できた.

支持度なし上昇型では, ハッシュ表の探索回数が支持度なし下降型と比較して多いため, ハッシュの衝突割合を低くすることが高速化のために必要となる. 今後の課題として, 衝突回数が少なくなるハッシュ関数の検討がある.

謝辞

本研究は一部，ISPS 科学研究費補助金（No.16K00298）の援助を受けている.

参考文献

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

[2] 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)

[3] 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)

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

[5] 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)

[6] 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)

[7] 宇野 毅明, 有村 博紀: LCM 公開プログラム, <http://research.nii.ac.jp/uno/ dodes-j.htm>

[8] 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)

[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] 谷島健斗, 岩沼宏治, 山本泰生: 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, 人工知能学会第 112 回知識ベースシステム研究会, SIG-KBS, B5-02, pp. 17-24, (2017)

[11] 谷島健斗, 岩沼宏治, 山本泰生: 飽和集合上の極小生成子の支持度計算を行わない高速抽出 -負の相関ルール抽出の効率化にむけて-, 人工知能学会第 106 回人工知能基本問題研究会, SIG-FPAI, B5-03, pp. 110-115, (2018)

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