

2004 年度卒業論文

制約付き頻出パターン抽出手法  
におけるユーザビリティの向上

ユーザ入力を削減した制約付き頻出パターン抽出手法

提出日：2005 年 2 月 2 日

指導：山名早人助教授

早稲田大学理工学部情報学科

学籍番号：1G01P052-7

社本 基宏

## 概要

本論文では,制約付き頻出パターン抽出手法におけるユーザビリティの向上のための一手法を提案する.昨今,情報技術の発展に伴いデータベースの大規模化・低価格化が進み,データ量も増加している.膨大なデータを人間の目で分析し,役に立つ情報を見つけ出すことは困難となっている.データベースに蓄えられたデータから,人間の代わりに機械的に情報を抽出する手段として,データマイニングが用いられている.データマイニングの研究分野の一つとして頻出パターン抽出問題がある.頻出パターン抽出とはトランザクションデータベースから最小サポート値を満たす頻出パターンを抽出する処理である.しかし,データの規模が大きければ,抽出される頻出パターン数も膨大になる可能性があり,抽出された膨大な数の頻出パターンを人間の目で確認することは難しくなる.そこで,膨大な頻出パターンを人間が判断できるように情報を絞り込んだマイニング手法が求められており,その手法の一つとして,制約付き頻出パターン抽出手法(Frequent Itemset Mining with Constraints)がある.制約付き頻出パターン抽出手法とは,トランザクション内に出現する各アイテムが持つ数値を利用して,ユーザが指定した制約条件と最小サポート値を同時に満たした頻出パターンを抽出する方法である.しかし,制約付き頻出パターン抽出手法を実行する際は「最小サポート値」,「制約の種類」と「制約の閾値」の3つの入力が必要である.制約付き頻出パターン抽出手法を行う前に,ユーザが3つの適切な入力値を設定することは困難である.そこで,本論文では,最小サポートを満たし,制約の計算値が上位となる頻出パターンをユーザに順次返す手法を提案する.提案手法では,上位となる頻出パターンから抽出することで,閾値の入力をユーザに求めず,「最小サポート」と「制約の種類」の2つのみで制約付き頻出パターン抽出を実行できる.

# 目次

第1章	はじめに	3
第2章	関連研究	5
2.1.	相関ルール	5
2.2.	頻出パターン抽出手法	6
2.2.1.	Apriori[2]	6
2.2.2.	FP-growth[4]	8
2.2.3.	H-mine (Mem)[9]	11
2.3.	頻出パターン抽出手法の拡張[1][3]	13
2.3.1.	パターンのまとめ上げ手法[5][6]	14
2.3.2.	制約付き頻出パターン抽出手法[1][3]	14
2.3.3.	制約付き頻出パターン抽出手法の定義[1][12]	15
2.3.4.	制約付き頻出パターン抽出手法の分類[1][4]	15
	Anti-Monotone	16
	Monotone	16
	Succinct	17
	Convertible Constraint	18
	Strongly Convertible Constraint	18
2.3.5.	CFG[1]	19
2.3.6.	ExAMiner[13]	21
2.3.7.	DualMiner[14]	23
2.3.8.	FIC <sup>A</sup> , FIC <sup>M</sup> [12]	25
2.4.	まとめ	29
第3章	提案手法	30
3.1.	従来手法の問題点	30
3.2.	提案手法の概要	30
3.3.	提案手法のアルゴリズム	30
3.4.	実行例	31
3.5.	その他 ( $avg(X)$ , $sum(X)$ 以外) の関数の提案手法適用	35
第4章	評価	37
4.1.	実験環境	37
4.2.	実験に用いたデータセット	37
4.3.	提案手法の評価	37
第5章	おわりに	42
	謝辞	43
	参考文献	44
	付録	45

# 第1章 はじめに

昨今、情報技術の発展に伴いデータベースの大規模化・低価格化が進み、蓄積されているデータ量も増加している。その結果、コンピュータの記憶装置に蓄積された膨大なデータを人間の目で分析し、役に立つ情報を見つけ出すことが困難である。したがって、膨大なデータを人間の代わりに機械的に情報を抽出する手段としてデータマイニング技術が用いられている。データマイニングの研究分野の一つとして頻出パターン抽出問題がある。頻出パターン抽出とはトランザクションデータベースからユーザが指定した最小サポート値を満たす頻出パターンを抽出する問題である。頻出パターン抽出問題の基本的なアルゴリズムとして Apriori[2]が有名である。Apriori アルゴリズムは、効率的にすべての頻出パターン(相関ルール)を発見することができる。しかし、次の2つの要因によって、Apriori も大規模なデータベースの頻出パターンを抽出する際には、抽出時間が長期化する問題がある[1][3]。

1. 頻出パターンを抽出するために生成される候補アイテム集合が膨大な数になり、必要となる記憶容量や計算量が増大する。
2. 問題1の候補アイテム集合の出現回数をカウントするためにデータベースを繰り返しスキャンする必要がある、マイニング時間が増大する。

上述の2つ問題を解消するために、FP-growth[4]が提案されている。FP-growth は FP-tree という構造を作り、FP-tree を参照して頻出パターンを抽出する。FP-tree を利用することで、大きな記憶容量を必要とせず、データベースを繰り返しスキャンする必要もなく、頻出パターンを抽出することができる。

しかし、頻出パターン抽出対象のデータベースの規模が大きいと、抽出される頻出パターン数も増大してしまい、ユーザが抽出された頻出パターンを把握することが困難になる。そこで、ユーザにとって有益な頻出パターンのみを頻出パターンマイニングが求められている。1つ目の手法として、Maximal pattern や Closed pattern のみを抽出しパターン数を減らす Maximal Pattern Mining[5]、Closed Pattern Mining[6]という手法が提案されている。2つ目に、最小サポート値以外の制約を付け加えることにより、抽出アイテムセット数を減らす制約付き頻出パターン抽出手法(Frequent Itemset Mining with Constraints[1])が提案されている。頻出パターン抽出手法に制約(Constraints)を組み込んだものが、制約付き頻出パターン抽出手法である。制約付き頻出パターン抽出手法とは、トランザクション内に出現する各アイテムが持つ数値を参照して、ユーザが指定した制約を満たす頻出パターンを抽出する方法である。世の中に存在するアイテムは、何らかの数値的な情報を持っている。例えば、商品であれば価格や粗利益、人であれば、身長や年齢や所得などが考えられる。小売店で販売情報を分析するケースでは、買い物客がカゴに入れた商品の頻出パターンを抽出するだけでなく、購入合計金額が10,000円を超える頻出パターンのみを抽出することが考えられる。

制約付き頻出パターン抽出手法は、ユーザが望むマイニング結果を得る手法として有効である。しかし、目的の情報を得るために制約付き頻出パターン抽出手法を行う時、ユーザは「最小サポート」、「制約の種類」と「制約の閾値」の3つの入力を求められる。

ユーザにとって、制約付き頻出パターン抽出を行う対象のデータに対して、すべての入力値を適切に設定することは非常に困難である。そこで、本論文では、最小サポートを満たし、制約の計算値が上位となる頻出パターンをユーザに順次返す手法を提案する。上位となる頻出パターンから抽出することで、閾値の入力をユーザに求めず、「最小サポート」と「制約の種類」の2つのみで制約付き頻出パターン抽出を実行できる。さらに、提案手法を retail[7]データに対して実行し、評価を行う。

第2章では、相関ルールと頻出パターン抽出について述べた後で、制約付き頻出パターン抽出について述べる。第3章では提案手法について述べ、第4章では提案手法についての評価を述べる。第5章ではまとめを述べる。

## 第2章 関連研究

1 節では、データマイニングの基本として、相関ルールの抽出問題についての概略を述べる。2 節では、頻出パターン抽出手法と一般的な頻出パターン抽出のアルゴリズムの概略を述べる。3 節では、制約付き頻出パターン抽出手法の概略と問題となる制約を分類し、基本的な制約付き頻出パターン抽出のアルゴリズムを述べる。

### 2.1. 相関ルール

小売店の POS システムでは顧客が購入した商品の集合がデータとして保存されており、同時にどのような商品が購入されたか知ることができる。POS データから、たとえば「商品  $X$  を購入した顧客は、商品  $Y$  も高い確率で購入する」という知識が得られれば、セット販売や商品陳列の再配置などの販売戦略を取ることができる。一般的に商品  $A, B$  を  $X, Y$  を商品の集合として、

$$X \Rightarrow Y$$

と記述される事実を相関ルールと言う[8]。

$I = \{i_1, i_2, \dots, i_m\}$  をアイテムの集合とする。データベース  $D$  はトランザクションの集合であり、トランザクション  $T$  はアイテムの集合である。各トランザクションにはユニークな識別子  $TID$  (transaction id) がつけられている。トランザクション  $T$  とアイテム集合  $X \subseteq I$  に関して  $T \supseteq X$  が成り立つとき、「 $T$  は  $X$  を含む」という。相関ルールとは  $X \subseteq I, Y \subseteq I, X \cap Y = \emptyset$  であるような品位のアイテム集合  $X, Y$  を作って作られる  $X \Rightarrow Y$  という表現のことである。

データベース  $D$  中の  $X$  を含むトランザクションのうち、 $Y$  を含むトランザクション数の割合が  $c\%$  であるとき、「相関ルール  $X \Rightarrow Y$  は  $D$  において  $c\%$  の確信度で成立している」と言い  $conf(X \Rightarrow Y) = c\%$  と表記する。また  $D$  中の  $X \cup Y$  を含むトランザクションの全トランザクションに対する割合が  $s\%$  であるとき「相関ルール  $X \Rightarrow Y$  は  $D$  において  $s\%$  のサポートを持つ」といい、 $support(X \Rightarrow Y) = s\%$  と表記する。

小売店の例では、アイテムは商品で、集合  $I$  は店舗で取り扱っている商品である。トランザクション  $T$  は 1 人の顧客が買い物かごに入れた商品集合である。データベース  $D$  はすべての顧客が買い物かごの内容を記録したものである。例えば表 2-1 では 4 人の顧客の購買行動が記録されているトランザクションデータベースである。このデータベースでは各行がトランザクションを表し、すなわち一人の顧客の購買行動を表す。たとえば、 $TID$  が 100 の顧客は商品  $a, c, d$  を購入したことを意味している。 $X \Rightarrow Y$  という相関ルールの確信度  $c\%$  で、そのサポートが  $s\%$  だとすると、商品の集合  $X$  を購入した顧客のうち  $c\%$  が商品の集合  $Y$  も同時に購入していて、 $X \cup Y$  を購入していた顧客の全体に対する割合は  $s\%$  であったということである。

表 2-1 トランザクションデータベースの例

TID	アイテム
100	a c d
200	b c e
300	a b c e
400	b e

アイテム集合を適当に組み合わせることにより、非常に多くの相関ルールを作ることができる。m種類のアイテムを自由に使うことができる相関ルールは

$\sum_{k=2}^m \binom{m}{k} (2^k - 2)$  個であるから、たとえば m=10 の場合 57000 個、m=100 の場合  $5.15 \times$

$10^{47}$  以上もの相関ルールを作り出すことが混在的に可能である。それらをしらみつぶしに調べることは不可能であり、そのなかで役に立つものはほんの僅かである。そこで、相関ルールの価値を図る上で、その確からしさを表す確信度が重要になる。さらにその上で、サポートも高いことが望ましい。サポートが低い相関ルールは、わずかなデータにしか当てはまらない稀な規則だからである。しかし、事前に価値のある相関ルールを作るにはどのアイテムを組み合わせれば良いかは分からない。そこで、自動的にデータベースからサポートと確信度の高い相関ルールを効率的に、しかももれなく発見する手法が必要である。

## 2.2. 頻出パターン抽出手法

頻出パターン抽出手法は高い相関ルールを持った頻出アイテム集合を漏れなくすべて抽出する手法である。頻出パターンとは、トランザクションデータベースにユーザが与えた最小サポート値以上出現するアイテム集合である。しかし、対象とするデータベースが巨大であるために、頻出パターンを抽出するには多くの時間が掛かってしまう。したがって、抽出に必要な時間を短くするために、高速に頻出パターンを抽出する方法が提案されている。以下に、頻出パターン抽出手法の基礎になった Apriori アルゴリズム 0 を述べる。

### 2.2.1. Apriori[2]

Agrawl らによって 1994 年に提案された Apriori[2] は、効率的にすべての頻出パターンを発見することができる手法である。Apriori は“あるアイテム集合 k が頻出パターンでなかった時、k を含むすべてのアイテム集合は頻出パターンではない”という考えの下で、頻出パターンを抽出する手法である。Apriori のマイニングの頻出パターン手順を以下に述べる。

1. 要素数 1 の候補アイテム集合を作り、TDB をスキャンしてすべてのサポート数をカウントする。

2. サポート数が最小サポートを満たさない要素数 1 の候補アイテム集合は排除し , 最小サポートを満たした候補アイテム集合を結合することにより , 要素数 2 の候補アイテムセット集合を作る .
  3. 生成した要素数 2 の候補アイテムセット集合を *TDB* をスキャンすることにより , サポート数をカウントし , 最小サポートを満たす要素数 2 の候補アイテムセット集合を要素数 2 の頻出アイテムセット集合とする .
- 以降は , 候補アイテム集合が作れなくなるまで同じことを繰り返す .

### Apriori の実行例

表 2-1 に示す *TDB* を対象とし , 最小サポート値を 2 としたときを例として , 頻出パターン抽出の流れを述べる( 図 2-1 を参照) .

要素数 1 の候補アイテム集合  $C_1$  は , *TDB* に出現するアイテム集合となる . ここで ,  $C_1$  に含まれる全てのアイテムのサポート数をカウントし , 最小サポート値を満たすアイテム  $\{a, b, c, e\}$  が , 要素数 1 の頻出アイテム集合  $L_1$  となる .  $L_1$  に属する任意の 2 つのアイテムを結合することにより , 要素数 2 の候補アイテムセット集合  $C_2$  を生成する .  $C_2$  についても *TDB* をスキャンすることで , サポート数をカウントすると  $ab$  と  $ae$  は最小サポートを満たさないので要素数 2 の頻出アイテムセット集合  $L_2$  は  $L_2 = \{ac, bc, ce\}$  になる .  $L_2$  に含まれる任意のアイテムセット集合を結合することにより , に作った要素数 3 の候補アイテム集合は  $C_3$  であり , そのサポート数をカウントする . これ以上は候補アイテム集合を生成することが出来ないのでアルゴリズムは終了する .

最終的にこの例では ,  $a, b, c, e, ac, bc, be, ce, bce$  の 9 つの頻出パターンが抽出される .

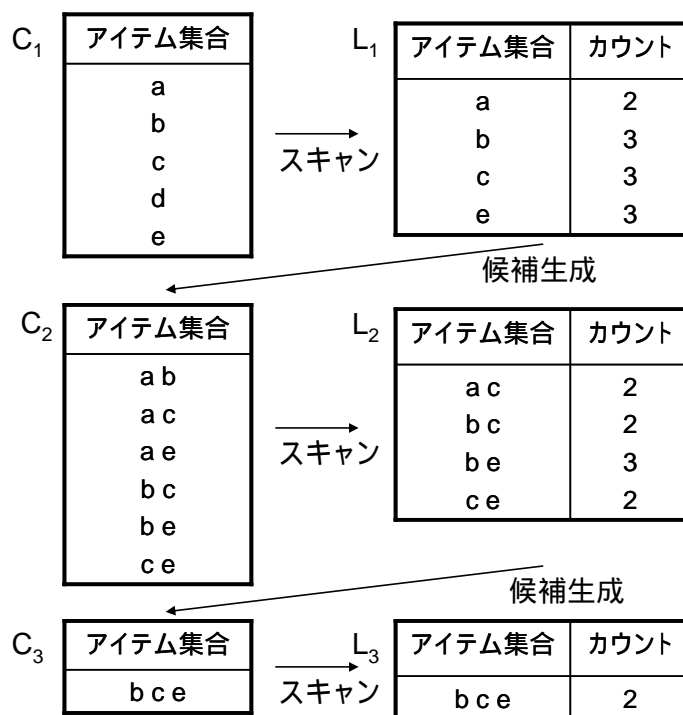


図 2-1 Apriori の実行例



## 2.2.2. FP-growth[4]

Apriori のアルゴリズムで候補アイテム集合を生成して、頻出パターン抽出をすると、候補アイテム集合のための計算コストが増大する問題が生じる。この問題を解消するため、J.Han らによって FP-tree 構造を構築して、頻出パターン抽出する FP-growth という手法が提案された[4]。FP-tree とはトランザクションデータベースを圧縮したデータ構造であり、頻出パターン抽出のために必要な情報はすべて格納される。したがって、最初にデータベースをスキャンして FP-tree を構築すれば、以降は TDB をスキャンする必要がなく、FP-tree のみを利用して頻出パターンを抽出することができる。この結果、データベースを繰り返しスキャンする必要がなく、メモリ容量の削減し、マイニングの効率を上げることができた。最初に、FP-tree の生成の手順を述べた後で、FP-tree を用いた頻出パターン抽出手法 FP-growth について述べる。

### FP-tree の構築例

圧縮したデータ構造の FP-tree は、prefix-tree 構造であり、以下の特徴を基に設計されている。

1. 頻出パターンに含まれる任意の単一アイテムは、最小サポート値を満たすアイテムである。頻出アイテムを検出するために一度は TDB をスキャンする必要がある。
2. FP-tree に各トランザクションの頻出アイテム集合を格納したら、以降は TDB を繰り返しスキャンする必要がない。
3. 複数トランザクションが同じアイテムセットを共有するならば、に共通するアイテムセットを同一 prefix に併合することができる。全トランザクションの頻出アイテムをサポート値降順にソートしておけば、2つのアイテム集合が同一かどうかを判断することは容易である。
4. 2つのトランザクションが prefix を共有していれば、ソートされた頻出アイテム順に従って、共通部分を併合することができる。頻出アイテムが減少順にソートされていれば、prefix を共有する機会が増える。

以下では、表 2-2 で示す TDB を対象として、最小サポート値を 3 とした時の例を用いて、FP-tree 構築手順を示す。最初に TDB をスキャンして、すべてのアイテムのサポート値を求める。最小サポート値を満たすアイテムのみをサポート値順にソートしたものを頻出アイテムリストとする。頻出アイテムリストは、(アイテム名: サポート値) のリストで構成される。この例では、頻出アイテムリストは、(f:4) (c:4) (a:3) (b:3) (m:3) (p:3) となる。

データベースに対しての 2 回目のスキャンをすることにより、prefix-tree 構造を構築する。まず初めに、木のルートを“null”とする。TID:100 のトランザクションを頻出アイテムリストで射影することにより、TID:100 の頻出アイテム{f,c,a,m,p}を得る。TID:100 の頻出アイテム{f,c,a,m,p}を基にして、root ノードの直下にパス (f:1) (c:1) (a:1) (m:1) (p:1) が構築される。

ノードのアイテムは最初に作った頻出アイテムリスト順に並んでいることに注意する．カウント値は 1 回ずつ出現したので，すべて 1 になっている．同様に，TID:200 のトランザクションの頻出アイテムは f c a b m であり，先頭部分 f c a は prefix-tree 構造にすでに構築されている部分構造と一致する．そこで，(f:1) (c:1) (a:1) はカウント値をインクリメントし，(f:2) (c:2) (a:2) とする．b m は (a:2) の直下に，パス (b:1) (m:1) を構築する．さらに，TID:300 は f b であり，f のみがすでに構築されている prefix-tree 構造と一致する．したがって (f:2) のカウント値をインクリメントし，(f:3) とする．さらに，(f:3) の子ノードとして (b:1) を構築する．次に，TID:400 は既存の prefix-tree 構造と一致する部分がないので，ルートから新しいパス (c:1) (b:1) (p:1) を構築する．TID:500 は既存の prefix-tree 構造の部分構造と完全に一致するので，一致するノードのカウント値をインクリメントする．

すべてのトランザクションのスキャンが終了したら，頻出アイテムリスト順のアイテムを持ったヘッダテーブルを作る．ヘッダテーブルに prefix-tree 部分のアイテム名が同じノード間にリンクを張る．さらに，他のノードにも同じアイテムがあれば，ノード間にリンクを張る．この状態を表したものが図 2-2 である．ヘッダテーブルの head of node links からリンクを辿ることで，特定のアイテムのノードをすべて探索できる．

表 2-2 トランザクションデータベース[4]

TID	Items Bought	(Ordered) Frequent Items
100	f a c d g i m p	f c a m p
200	a b c f l m o	f c a b m
300	b f h j o	f b
400	b c k s p	c b p
500	a f c e l p m n	f c a m p

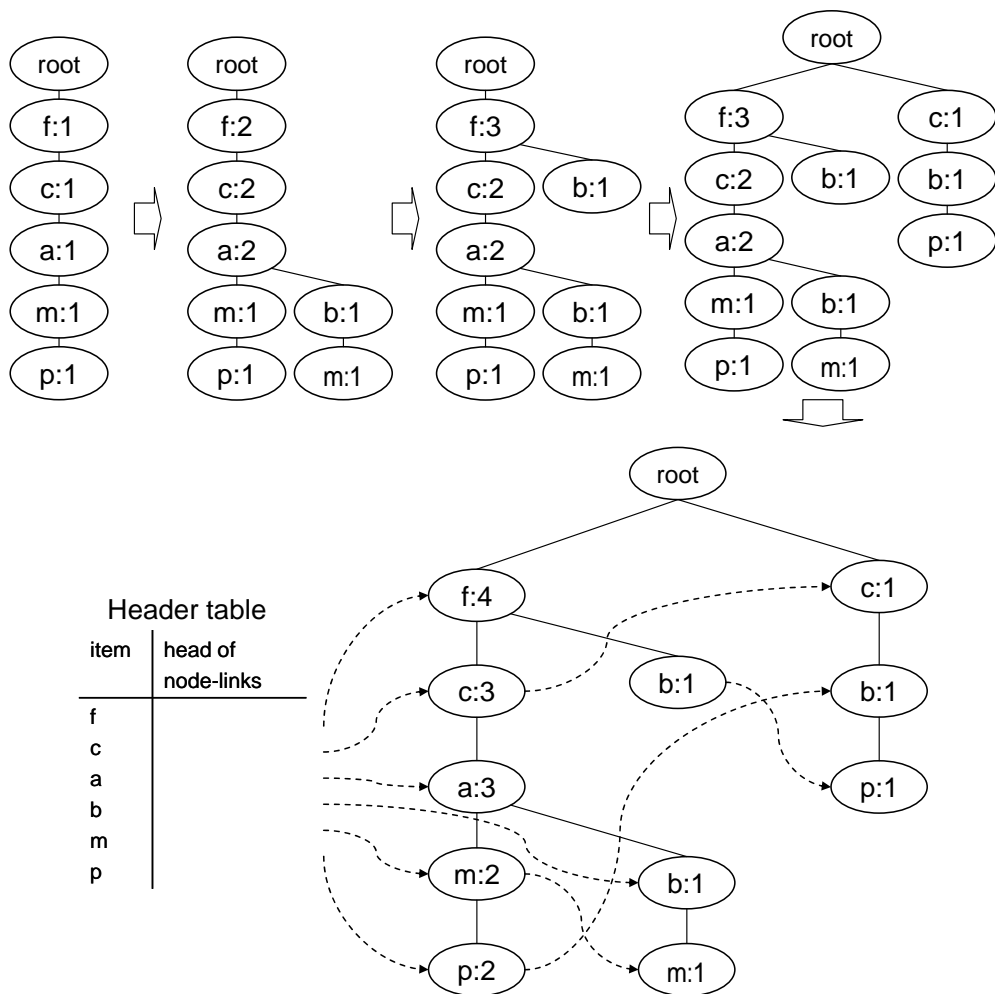


図 2-2 FP-tree[4]

### FP-growth の実行例

図 2-2 に示す FP-tree を用いて FP-growth の頻出パターン抽出の流れを説明する。ノード p に着目すると (f:4) (c:3) (a:3) (m:2) (p:2) と (c:1) (b:1) (p:1) の 2 つのパスが見つかる。ここから f c a m p はデータベースに 2 回出現し, f c a は 3 回出現し, f は 4 回出現することが分かる。しかし, p と同時に f が出現する回数は 2 回しかないことにも注目する。p を含む頻出アイテム集合の p-prefix は (f:2) (c:2) (a:2) (m:2) とカウントできる。p-prefix とは p を含むパターンで, p を除いたものである。同様に 2 つめのパス c b p はデータベースに 1 回しか出現しないので p-prefix は (c:1) (b:1) とカウントできる。p の部分集合である p-prefix のパスは { (f:2) (c:2) (a:2) (m:2), (c:1) (b:1) } となる。よって, p-prefix で最小サポート 3 を満たす頻出パターンは (c:3) のみであることがわかる。p の検索は以上であり, p を含むすべての頻出パターンが抽出された。他のアイテム f, c, a, b, m についても同様に検索することですべての頻出パターンを抽出することができる。

このように, FP-growth は FP-tree のみを用いて, すべての頻出パターンを抽出することができる。

### 2.2.3. H-mine(Mem)[9]

2.2.1 で述べた，FP-growth にも次の問題がある．

1. 対象とするデータセットが大規模になった場合，スワップを起こさずに効率よく頻出アイテムセットを抽出できない．
2. 対象とするデータセットの疎密両方に対して効率よく頻出アイテムセットを抽出できない．実社会では，特定の傾向がある密なデータと関連性がないランダムなデータが混ざっている．

この問題を解決するために，2001 年 J.Peï らによって，メインメモリにデータが収まるように分割可能な Hyper-Structure 構造が考え出された．H-mine(Mem)はメモリ上に Hyper-Structure を構成することで頻出パターンを抽出する手法である[9]．この Hyper-Structure を利用して，メインメモリ上にデータが収まるサイズに分割して頻出パターン抽出することを可能にし，大規模なデータに対応している．ここでは，Hyper-Structure と H-mine(Mem)について，例を用いて説明する．

#### Hyper-Structure の構築

表 2-3 に示す TDB を対象として，最小サポートの値は 2 とした場合の例を用いて Hyper-Structure 構造の構築方法を示す．最初は Apriori と同じ手順で，アイテムサイズが 1 の頻出アイテムを検出するために，TDB をスキャンする．そして，a:3，c:3，d:4，e:3，g:2 が抽出される．また，辞書順にアイテムを並べたものを F-list(: a-c-d-e-g) と呼ぶことにする．a-c-d-e-g の頻出パターンは次の 5 つのパターンに分類することができる．

a を含むパターン

a は含まないが c を含むパターン

a と c は含まないが，d を含むパターン

a と c と d は含まないが，e を含むパターン

g のみを含むパターン

また，射影されたすべての頻出アイテムは F-list に従ってソートされる．例えば，TransID100 の Frequent-item Projection は cdeg の順にソートする(表 2-3 の Frequent-item projection を参照)．すべての頻出パターンは「TransID」と「Hyper-link」の 2 つをエントリに格納する．

表 2-3 トランザクションデータベース[9]

Trans ID	Items	Frequent-item projection
100	c d e f g i	c d e g
200	a c d e m	a c d e
300	a b d e g k	a d e g
400	a c d h	a c d

次に、頻出アイテムのエントリ毎に「TransID」「support 値」「Hyper-link」の3つ値を持つハイパーテーブルHを作る。頻出アイテムの射影がメモリに読み込まれると、最初のアイテムと同じアイテムがキューとしてハイパーリンクによってリンクされる。そしてヘッダテーブル H のエントリはキューのヘッダとして振舞う。たとえば、ヘッダテーブルでアイテム a のエントリは a-キューのヘッドであり、トランザクション 200, 300, 400 の頻出アイテムの射影にリンクされる。これらの3つの射影は先頭に a をもった頻出アイテムである。同様にトランザクション 100 の頻出アイテムの射影は H で c を先頭に持った c-キューとしてリンクされる。d-, e-, g-キューはこれらを先頭に持つ頻出パターンの射影がないので空である。これで、Hyper-Structure の構築は終了する。この状態を図で表したものが、図 2-3 である。

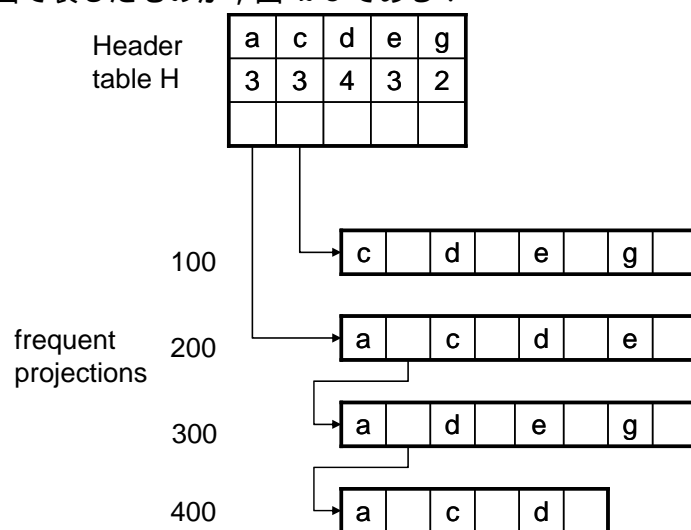


図 2-3 Hyper-Structure[9]

### H-Mine(Mem)の実行例

最初に Hyper-Structure を構築すれば、以降はデータベースを参照しなくても、Hyper-Structure のみを用いて頻出パターンを抽出できる。ヘッダテーブルの5つの頻出パターンのうち a を含むパターンについての頻出パターン抽出方法を以下に示す。

a を含む頻出アイテム射影を探す際は、Hyper-Structure のリンク構造が利用できる。a をヘッダとして保持するヘッダテーブル Ha が生成される。Ha は、a が射影されたデータベースから H と同様に各サポート値を計算する。アイテム c は TID:200 と 400 の2回出現するため、Ha における c のサポート値は2となる。d, e, g にたいしても同様にサポート値を求めると、d:3, e:2, g:1 となる。g は最小サポート値を満たさない。ここで抽出された頻出パターンは ac:2, ad:3, ae:2 の3つとなる(図 2-4 参照)。

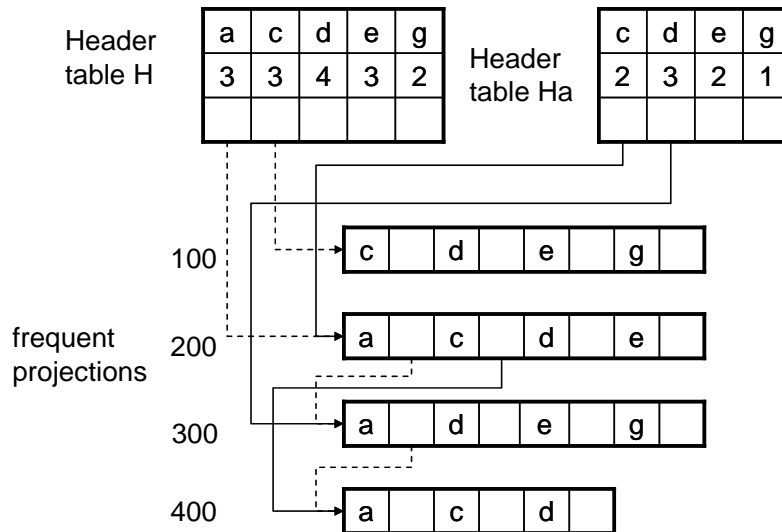


図 2-4 ヘッダテーブル Ha[9]

次は Ha にある c キューを調べると ,ac が射影されたデータベースで処理が続けられる . そして , 図 2-5 のように ac ヘッダテーブル Hac が生成される . Hac における d のサポート値は 2 ,e のサポート値は 1 となり , ここで抽出された頻出パターンは acd:2 のみとなる . これ以上は射影されたデータベースで処理が進められないので , このフェーズはこれで終了する .

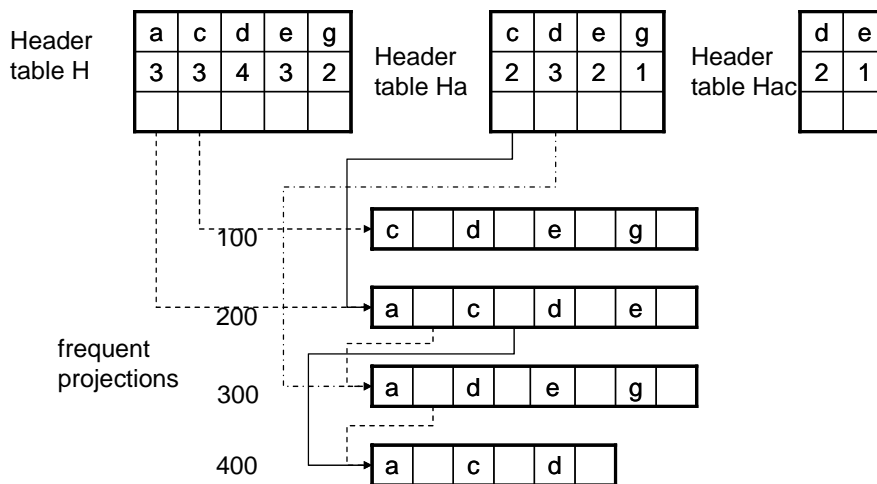


図 2-5 ヘッダテーブル Hac[9]

以上のように他の Had , Hae についても処理をすることで Ha の処理はすべて終了する . Ha の処理の終了後は , a は含まないが , c を含むパターンである Hc など Hd , He , Hg をすべて処理することで , すべての頻出パターンを抽出することができる .

## 2.3. 頻出パターン抽出手法の拡張[1][3]

頻出パターン抽出手法は最小サポート値を満たす頻出パターンを抽出することで , 価値のある情報を発見する手段を提供してきた . しかし , 頻出パターン抽出対象のデータ

ベースの規模が大きいと、抽出される頻出パターン数も増大してしまい、抽出された結果から有用な知識につながるパターンを発見することが困難になっている。そこで、ユーザにとって有益な頻出パターンのみを頻出パターン抽出し、より簡単に有益なパターンを発見しやすくする手法として、パターンのまとめ挙げ手法、制約付き頻出パターン抽出手法が提案されている。

### 2.3.1. パターンのまとめ上げ手法[5][6]

極大アイテムセット(Maximal Itemset)や飽和アイテムセット(Closed Itemset)のみを抽出しパターン数を減らす極大頻出アイテムセット抽出手法(Maximal Frequent Itemset Mining)[5]、飽和頻出アイテムセット抽出手法(Closed Frequent Itemset Mining)[6]という手法が提案されている。

ここで、アイテムセット  $X$  が頻出極大アイテムセットであるということは、「アイテムセット  $X$  のサポート値が最小サポート値以上であり、かつ  $X$  のスーパーセットである任意の  $X'$  が、最小サポート値未満のサポート値である」ことである。また、アイテムセット  $X$  が頻出飽和アイテムセットであるということは、「アイテムセット  $X$  のサポート値が最小サポート値以上であり、かつ  $X$  と同一のトランザクション上にある  $X$  の全てのスーパーセット  $X'$  が、最小サポート値未満のサポート値である」ことである。

### 2.3.2. 制約付き頻出パターン抽出手法[1][3]

2つ目に、最小サポート値以外の制約を付け加えることにより、抽出アイテムセット数を減らす制約付き頻出パターン抽出手法(Frequent Itemset Mining with Constraints[1])が提案されている。頻出パターン抽出手法に最小サポート値とは別の制約(Constraints)を組み込んだものが、制約付き頻出パターン抽出手法である。制約付き頻出パターン抽出手法は、トランザクションデータベースだけではなく、トランザクションに出現するアイテムと対応する profit という数値を定義し、profit の数値での制約を満たすアイテムセットを抽出する。従来の頻出パターン抽出はトランザクション内にアイテムの出現頻度のみが抽出条件になっていた。これに対し、制約付き頻出パターン抽出手法は、最小サポート値を満たし、かつ、トランザクション内に出現するアイテムとアイテムが持つ profit を参照して、ユーザが指定した制約を満たす頻出パターンのみを抽出する。

世の中に存在するアイテムは、何らかの数値的な情報を持っていることが多い。例えば、商品であれば価格や粗利益、人であれば、身長や年齢や所得などが考えられる。頻出パターン抽出では、商品 A と C、商品 B と C はいずれもサポート値 10% の頻出パターンとして抽出された時、両者の意味は同じである。もし、商品 A の販売価格が 1000 円、商品 B が 10,000 円、商品 C が 500 円の時、同じ頻出パターンでも、パターン{A,C} は 1,500 円、パターン{B,C} は 10,500 円となり、販売者側にとって 2 つパターンの意味は大きく異なる。小売店で販売情報を分析するケースでは、買い物客がカゴに入れた商品の頻出パターンを抽出するだけではなく、購入合計金額が 10,000 円を超える頻出パ

ターンのみを抽出することが考えられ、このケースでは商品 B と C のパターンのみが抽出される。このように、マイニングに制約を加えることによって、ユーザが興味を持っている頻出パターンを絞り込むことが可能になる。

### 2.3.3. 制約付き頻出パターン抽出手法の定義[1][12]

$I$  をすべてのアイテムとし、各アイテムはあらかじめ(価格、重さ、年齢など)特徴が定義されている。トランザクション  $T = \langle tid, I_t \rangle$  は一つのタプルで、 $tid$  はトランザクションを特定し、 $I_t \subseteq I$  である。トランザクションデータベース  $T$  はトランザクションで構成されている。アイテム集合は  $S \subseteq I$  であり、すべてのアイテムの部分集合である。 $k$ -itemset  $S$  は  $k$  個のアイテムで構成されたアイテム集合  $S$  であり、 $k$  はアイテム集合の要素数( $k = |S|$ )を表している。

あるアイテム集合  $S$  は  $S \subseteq I_t$  であり、トランザクション  $T = \langle tid, I_t \rangle$  に含まれている。

**constraint**  $C$  はアイテム  $I$  の冪集合を意味し、 $C : 2^I \rightarrow \{true, false\}$  である。 $C(S) = \{true\}$  であることを、アイテム集合  $S$  が  $C$  を満たすという。 $C$  を満たしているアイテム集合を  $SAT_C(I) = \{S \mid S \subseteq I \wedge C(S) = true\}$  と表す。

### 2.3.4. 制約付き頻出パターン抽出手法の分類[1][4]

制約付き頻出パターン抽出手法の問題を考えるにあたり、頻出パターン抽出手法に新たに付け加えられた制約を表現する関数の種類と制約の分類について述べる。

制約で利用する関数の種類の代表的なものには、集合、min、max、count、sum、range、avg、support がある。

- $\min(X)$  は、アイテム集合  $X$  に含まれているアイテムのうち、もっとも小さい profit を持つアイテムの profit を返す関数。
- $\max(X)$  は、アイテム集合  $X$  に含まれているアイテムのうち、もっとも大きい profit を持つアイテムの profit を返す関数。
- $\text{count}(X)$  は、アイテム集合  $X$  に含まれているアイテム数を返す関数。
- $\text{sum}(X)$  は、アイテム集合  $X$  に含まれているアイテムの profit の和を返す関数。
- $\text{range}(X)$  は、 $|\max(X) - \min(X)|$  を返す関数。
- $\text{avg}(X)$  は、アイテム集合  $X$  に含まれているアイテムの profit の平均、 $\frac{\text{sum}(X)}{\text{count}(X)}$

を返す関数。

- $\text{support}(X)$  は、アイテムセット  $X$  が TDB に出現する回数を返す関数。

その他、アイテム集合  $X$  に特定のアイテムまたはアイテムが含まれているかを判定する。例えば、 $S \subseteq V$  の制約は、候補アイテム集合  $S$  が必ずアイテム集合  $V$  に含まれている必要がある。

$\text{sum}(X)$  関数を用いて、アイテム集合  $S$ 、制約  $C$ 、制約の閾値  $v$  を付けたものが

$$C(S.\text{profit}) \theta v \quad \theta \in \{<, \leq, \geq, >\}$$



である． $S.profit$  はアイテム集合  $S$  の各アイテムの  $profit$  である．例えば， $sum(S.profit) \geq v$  であれば， $S$  の各アイテムの  $profit$  の和が閾値  $v$  以上であることを表す．

以上の制約関数は，制約付きの頻出パターン抽出時の振る舞いによって Anti-Monotone，Monotone，Succinct，Convertible Monotone と分類することが可能である．この分類について述べる．

## Anti-Monotone

Anti-Monotone の制約は，“あるアイテム集合  $S$  が制約を満たさなければ， $S$  を含む集合はすべて制約を満たさない”ことを言う．つまり，Apriori もこの部類であり，あるアイテム集合  $S$  が制約を満たさなければ，以降は  $S$  を含む候補アイテム集合を探索する必要がない．

例(トランザクションデータベースは表 2-4，アイテムの  $profit$  は表 2-5，最小サポート値：1)

制約： $range(S.profit) \leq 15$  は Anti-monotone であり，アイテム集合  $ab$  は制約を満たさないので， $ab$  を含むすべての集合は制約を満たすことはない．つまり， $ab$  を含んでいる  $abc$  や  $abdf$  も制約を満たさない．

表 2-4 トランザクションデータベース

TID	Item in transaction
10	a b c d f
20	b c d f g h
30	a c d e f
40	c e f g

表 2-5 アイテムの  $profit$

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

## Monotone

Monotone は“あるアイテム集合  $S$  が制約を満たすならば， $S$  を含むすべてのアイテム集合が制約を満たす”ことを言う．つまり，Monotone の制約を満たすアイテム集合  $S$  が見つければ， $S$  をサブセットとして含む  $S$  のスーパーセットは必ず制約を満たす．

例(トランザクションデータベースは表 2-4 , アイテムの profit は表 2-5 , 最小サポート値 : 1)

制約 :  $range(S.profit) \geq 15$  は monotone であり , アイテム集合 ab は制約を満たすので , ab を含むすべてのアイテム集合は制約を満たす . つまり , ab を含む abc や abdf は必ず制約を満たす .

## Succinct

Succinct は”アイテム選択に依存しており , トランザクションデータベースを見ただけでは , あるアイテム集合 S が制約を満たしているか判断できない”ことを言う .

例(トランザクションデータベースは表 2-4 , アイテムの profit は表 2-5 , 最小サポート値 : 1)

制約 :  $\min(S.profit) < 5$  は Succinct であり , a は制約を満たさないからと言って , a を含むすべてのアイテムが制約を満たすか満たさないかどうかは , 現時点では判断できない . 仮に , b を加えた ab の時は制約を満たすが , d を加えた ad の時は制約を満たさない .

制約関数が Anti-Monotone , Monotone , Succinct に分類されるかどうかは , 表 2-6 を参照 .

表 2-6 Anti-Monotone, Monotone, Succinct[3]

制約	Anti-Monotone	Monotone	Succinct
$v \leq S$	no	yes	yes
$S \leq V$	no	yes	yes
$S \geq V$	yes	no	yes
$\min(S) \leq v$	no	yes	yes
$\min(S) \geq v$	yes	no	yes
$\max(S) \leq v$	yes	no	yes
$\max(S) \geq v$	no	yes	yes
$\text{count}(S) \leq v$	yes	no	weakly <sup>1</sup>
$\text{count}(S) \geq v$	no	yes	weakly
$\text{sum}(S) \leq v(a \in S, a \geq 0)$	yes	no	no
$\text{sum}(S) \geq v(a \in S, a \geq 0)$	no	yes	no
$\text{range}(S) \leq v$	yes	no	no
$\text{range}(S) \geq v$	no	yes	no
$\text{avg}(S) \leq v, \{=, \geq, \leq\}$	convertible	convertible	no
$\text{support}(S)$	yes	no	no
$\text{support}(S)$	no	yes	no

<sup>1</sup> weakly とは , 追加するアイテム集合の要素数に制約が依存するので , yes/no が判断できないことを言う .

## Convertible Constraint

Anti-Monotone , Monotone , Succinct と 3 つに分類したが , 他にアイテムの profit 順序に依存して Anti-Monotone や Monotone に分類されるものがある .

例(トランザクションデータベースは表 2-4 , アイテムの profit は表 2-5 , 最小サポート値 : 1)

制約 :  $avg(S.Pforit) \leq 25$  とし , アイテムを profit 減少順に並べる  $\langle a,f,g,d,b,h,c,e \rangle$  . この時 , アイテム集合  $afb$  は制約を満たさず ,  $afbh$  も制約を満たさないことが成り立つので , Anti-Monotone になる . このような Anti-Monotone を Convertible Anti-Monotone と言う .

トランザクションに出現するアイテムを profit 降順に並べる . このアイテム列を  $R$  として ,

**Convertible Anti-Monotone** は , " $R$  の先頭のアイテムから順番に候補アイテム集合を生成する時 , あるアイテム集合  $S$  が制約を満たさなければ ,  $S$  を含むすべてのアイテム集合は制約を満たさない . "ことを言う .

例 アイテムを降順にした時の  $avg(S) \leq v$

**Convertible Monotone** は , " $R$  の先頭のアイテムから順番に候補アイテム集合を生成する時 , あるアイテム集合  $S$  が制約を満たすならば ,  $S$  を含むすべてのアイテム集合は制約を満たす . "ことを言う .

例 アイテムを昇順にした時の  $avg(S) \geq v$

## Strongly Convertible Constraint

アイテム減少順  $R$  で候補アイテム集合を作る時 ,  $C : avg(S) \geq 10$  は Convertible Anti-Monotone である .

例(トランザクションデータベースは表 2-4 , アイテムの profit は表 2-5 , 最小サポート値 : 1)

アイテム集合  $db$  は制約を満たさないので ,  $dbh$  のように  $db$  を含むすべてのアイテム集合は制約を満たさない .

そして , アイテム昇順  $R^{-1}$  で候補アイテム集合を作る時 ,  $C : avg(S) \geq 10$  は Convertible Monotone である .

例(トランザクションデータベースは表 2-4 , アイテムの profit は表 2-5 , 最小サポート値 : 1)

アイテム集合  $d$  は制約を満たすので ,  $df$  や  $dfa$  のように  $d$  を含むアイテム集合は制約を満たす .

このように , 候補アイテム集合の順序によって Convertible Anti-monotone が Convertible Monotone が変化することを Strongly Convertible という . 制約が Convertible Anti-Monotone , Convertible Monotone , Strongly Convertible に分類したものを , 表 2-7 に示す . median という制約関数は , 候補アイテム集合のアイテムをアイテムの profit 順にソートし , アイテム数の中位にある数値が制約を満たしているかを判断する関数である . sum はアイテムの profit に負数を含んでいるアイテムが存在するかどうかで分類が異なる . これは , アイテムの profit に負数を含んでいなければ ,

候補アイテム集合のアイテム数が増えれば、必ず  $\text{sum}$  の値は 0 以上増えることが期待できるが、負数を含んでいる場合は減少の可能性もある。よって、 $\text{sum}(S) \leq v$  のケースでは、アイテムの  $\text{profit}$  に負数があれば Convertible Anti-Monotone となるが、負数があると Convertible Monotone となる。また、これまでに紹介した示した制約の種類を分類図で表したものを図 2-6 に示す。

表 2-7 Convertible Constraint[3]

制約	Convertible Anti-Monotone	Convertible Monotone	Strongly Convertible
$\text{avg}(S) \leq v$	yes	yes	yes
$\text{median}(S) \leq v$	yes	yes	yes
$\text{sum}(S) \leq v$ (非負の $\text{profit}$ )	yes	no	no
$\text{sum}(S) \leq v$ (負の $\text{profit}$ を含む)	no	yes	no
$\text{sum}(S) \geq v$ (非負の $\text{profit}$ )	no	yes	no
$\text{sum}(S) \geq v$ (負の $\text{profit}$ を含む)	yes	no	no

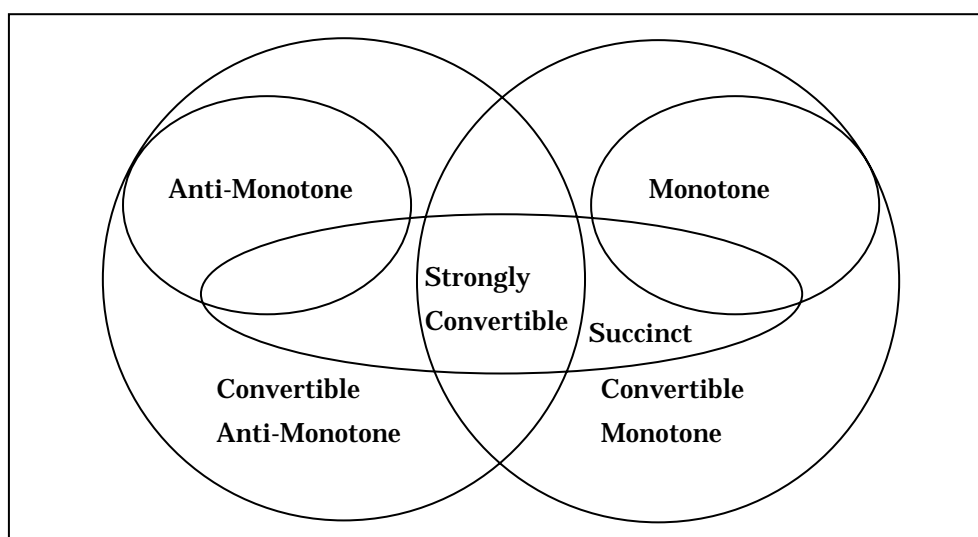


図 2-6 制約の分類図[12]

### 2.3.5. CFG[1]

CFG は、2000 年 J.Peï らによって提案された[1]。CGG は、アプリアリをベースとして anti-monotone・Convertible Anti-monotone の制約付き頻出パターン抽出手法[1]である。アイテムの持つ  $\text{profit}$  はすべて非負であることが前提条件となる。トランザクションデータベースを Constraint  $C$  を満たすアイテムで射影し、条件付きトランザクションデータベースを構築することによって、制約付き頻出パターンを抽出する。ここで、 $\text{TDB}|_f$  は  $f$  を含むトランザクションの射影から  $f$  を取り除いたデータベースである。 $\text{TDB}|_{\neg f}$  は  $f$  を含まず、 $d$  を含むトランザクションの射影から  $d$  を取り除いたデータベー

スである．すべての Constraint を満たす頻出アイテムに対して，データベースを射影することで，抽出対象を絞り込んでいる．図 2-7 に，トランザクションデータベースの射影方法を示す．

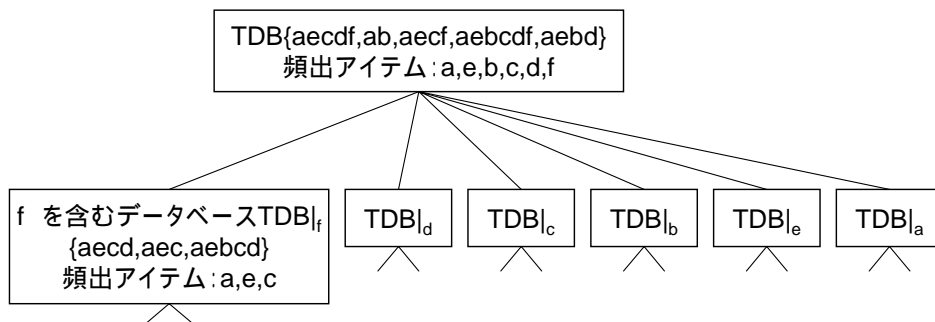


図 2-7 CFG[1]

制約付き頻出パターン抽出対象のトランザクションデータベースは表 2-8 ,アイテムの Profit は表 2-9 を利用して，最小サポート値を 3，制約  $C: \text{sum}(S.\text{profit}) \leq 180$  とした場合の CFG の動作例を示す．CFG では以下の 3 つを基に，トランザクションを絞り込む．最初にアイテム毎の頻出回数を数えて，最小サポートを満たしていないアイテムはトランザクションから削除する．この例では  $a: 5, b: 4, c: 3, d: 3, e: 3, f: 3$  とすべて最小サポートを満たしている．この状態で以下の 3 つの理論を用いて，マイニング対象を絞り込んでマイニングを行う．

1. 単一アイテムで制約を満たさないものはパターン抽出のための探索対象から除く  
 例) d 単体の場合  $d.\text{profit}=200 > 180$  となり，d を含んだ候補アイテム集合は必ず制約を満たさないので，d は候補から取り除くことができる．
2. もしアイテム集合 が制約を満たさなければ， を部分集合として含む候補アイテム集合を生成する必要はない．  
 例) アイテム集合  $ab.\text{profit}=200 > 180$  となり，ab を含んだ候補アイテム集合は必ず制約を満たさないので，ab を含んだ候補アイテム集合は生成する必要がない．
3. もしアイテム集合 が制約を満たせば， の部分集合が制約を満たすことを確認するする必要はない．  
 例) アイテム集合  $acef=160 < 200$  となり，acdf は制約を満たすので，acdf の部分集合が制約を満たすことは明らかである．

また，2.3.4 のアイテム順 R を利用することにより，Convertible Anti-monotone にも対応できる．

## CFG のアルゴリズム

表 2-8 トランザクションデータベース[1]

TID	Items in transaction
100	a c d e f
200	a b
300	a c e f
400	a b c d e f
500	a b d e

表 2-9 アイテムの profit[1]

Item	a	b	c	d	e	f
Profit	50	150	10	200	20	80

### 2.3.6. ExAMiner[13]

ExAMiner は、2003 年に F.Bonchi らによって提案された[13]。ExAMiner は、基本的には Apriori アルゴリズムに類似したものである。ExAMiner はアプリアリのアルゴリズムと制約(monotone, anti-monotone)を組み合わせ、検索対象のデータサイズを減らしているところが特徴である。アイテム集合数を増やす前に極力 TDB をフィルター(-reduction,  $\mu$ -reduction)にかけて、データを振り落としてデータサイズを小さくしている。

#### -reduction

Apriori アルゴリズムを用いて、最小サポートに満たさない候補アイテム集合を抽出対象のアイテムセット集合から取り除くことによって、計算量を削減する手法を -reduction という。

#### $\mu$ -reduction

-reduciton を実行することにより、制約を満たさないトランザクションを抽出対象のトランザクション集合から取り除くことによって、計算量を削減する手法を  $\mu$ -reduction という。

表 2-10 トランザクションデータベース[13]

TID	Itemset	Total
1	g,h,i	21
2	a,d,i,k	60
3	a,c,g,h,j	34
4	i,l,j,k	47
5	f,h,k	33
6	c,e,j,k	46
7	a,c,g,l,j,k	61
8	c,e,g,i,j	44
9	f,g,i,j	31
10	c,f,g,i,j	39
11	g,c,e,g,i	59
12	a,d,g,k	56
13	e,g,i	31
14	a,b,i,l,j	59

表 2-11 アイテムの profit[13]

Item	a	b	c	d	e	f	g	h	i	j	k	l
profit	10	20	8	22	15	15	6	5	10	5	18	14

### アルゴリズムの説明

表 2-10 と表 2-11 のようにアイテムとデータベースが与えられたとする。Total は各トランザクションのアイテムの profit の合計である。

最小サポート値：3 制約： $sum(S.profit) \geq 30$

- I. 制約から TID = 1 の Total が 30 未満で、TID=1 が削除される。
- II. 残りの TDB から要素数 1 の候補アイテムをカウントすると、b,d,h が非頻出アイテムと分かり、3 つのアイテムは取り除かれる(  $\mu$ -reduction)。
- III. ここで TID=3 に注目すると、h が取り除かれたことにより、Total = 29 < 30 となる。よって、TID=3 も削除される。同様の理由で TID=5 も削除される( $\mu$ -reduction)。
- IV. TID=5 が削除されたことにより、f は最小サポートを満たさないので、f が取り除かれる(  $\mu$ -reduction)。
- V. f が取り除かれたことにより、TID=9、10 の Total も 30 未満となってしまうので、TID=9、10 も削除される( $\mu$ -reduction)。

以上の様に繰り返し、削除されずに残った要素数 1 のアイテムは  $L_1 = \{a, c, e, g, i, j, k, l\}$  である。V まで終了した段階を表 2-12 に示す。

表 2-12 V まで終了した時点のトランザクション

TID	Itemset	Total
2	a,i,k	38
4	i,l,j,k	47
6	c,e,j,k	46
7	a,c,g,l,j,k	61
8	c,e,g,i,j	44
11	c,e,g,i	39
12	a,g,k	34
13	e,g,i	31
14	a,i,l,j	39

以降は ~ の繰り返し .

- VI. 表 2-12 から要素数 2 の頻出アイテム集合を作ると ,  $L_2 = \{ak, , ce, cg, cj, eg, ei, gi, ij, jk, , jl\}$  .
- VII.  $L_2$  から a と l は非頻出アイテムセットで , a と l は取り除かれる (  $\gamma$ -reduction ) . 同時に TID=2 , 12 , 14 の Total が 30 未満となるので , TID=2 , 12 , 14 は削除される (  $\mu$ -reduction ) . の削除によって ,  $L_2 = ce, cg, cj, eg, ei, gi, ij, jk$  となる . ,
- VIII. TID=14 が削除されたことから TID=4,6,7 も制約を満たさなくなるので削除される . 表 2-13 の状態になり , c と j は頻出アイテムセットでなくなり , 取り除かれる (  $\gamma$ -reduction ) .

表 2-13 まで終了した時点のトランザクション

TID	Itemset	Total
8	c,e,g,i,j	44
11	c,e,g,i	39
13	e,g,i	31

- IX. 最終的に頻出アイテムセットは egi になる . (表 2-14 参照)
- 以上のように  $\gamma$ -reduction と  $\mu$ -reduction を使い , マイニング対象を絞り込んで , 頻出パターンを抽出する .

表 2-14 まで終了した時点でのトランザクション

TID	Itemset	Total
8	e,g,i	31
11	e,g,i	31
13	e,g,i	31

### 2.3.7. DualMiner[14]

DualMiner は , 2003 年に C.Bucila らによって提案された[14] . ExMiner は 1 つの Anti-Monotone 又は Monotone の制約付き頻出パターン抽出ができたが , DualMiner



では2つの制約付き頻出パターン抽出ができる手法である。たとえば, ExMiner では,  $\max(X.price) > \$100$  のような制約がつけられるが, DualMiner では  $\min(X.price) \leq \$200 \wedge \max(S.price) \leq \$400$  のように2つの制約付き頻出パターン抽出が行える。

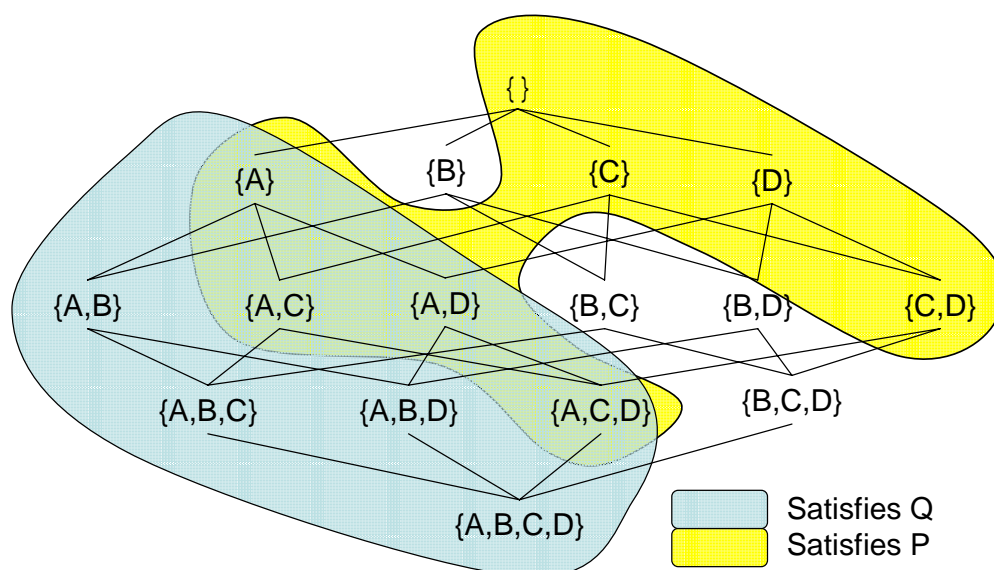


図 2-8 DualMiner[14]

DualMiner では, 計算木と呼ばれる独自の木構造を用いて候補アイテム集合を絞り込む。計算木について例を用いて説明する。あるトランザクション A(26)B(26)C(1)D(1)E(100) があり, 括弧内の数値は profit とする。制約は  $\text{sum}(S.\text{profit}) > 50$  とする。このトランザクションを図にしたものが図 2-9 であり, すべてのノード( )は  $(X,Y,Z)$  と3つの要素で構成されている。

- Xの要素は候補アイテム集合に含まれるものであり, IN( )とする。
- Yの要素は子ノードに含まれるアイテム集合であり, CHILD( )とする。
- ZにもYにも含まれないアイテム集合であり, OUT( )とする。

ノード を初期値とし, ノード では IN(E)とする。E.profit=100 なので, Eを含むすべて候補アイテム集合が制約を満たすことを明らかに, CHILD(ABCD)OUT( ), CHILD(BCD)OUT(A), CHILD(CD)OUT(AB), CHILD(D)OUT(ABC), CHILD( )OUT(ABCD)となり, Eを含む全パターンが制約を満たす。ノード では, OUT(E)とした CHILD(ABCD)について考える。ABCDの中のAをOUTにした場合, BCD.profit=28 となり制約を満たすことはない, IN(A)の候補アイテム集合を考える。次に IN(AB)を考えると, AB.profit=52 となって, 制約を満たすので, CHILD(CD)OUT(E)となる。ノード は ABの時点で制約を満たしている, IN(ABC)が制約を満たしていることは明らかであり, CHILD(D)OUT(E)でノード は終了する。ノード はDをINに加え, IN(ABCD)とし, CHILD( )となるのでこれ以上は下にノードは伸びない。

以上のように候補アイテム集合を構成して, 頻出パターン抽出を行う手法である。

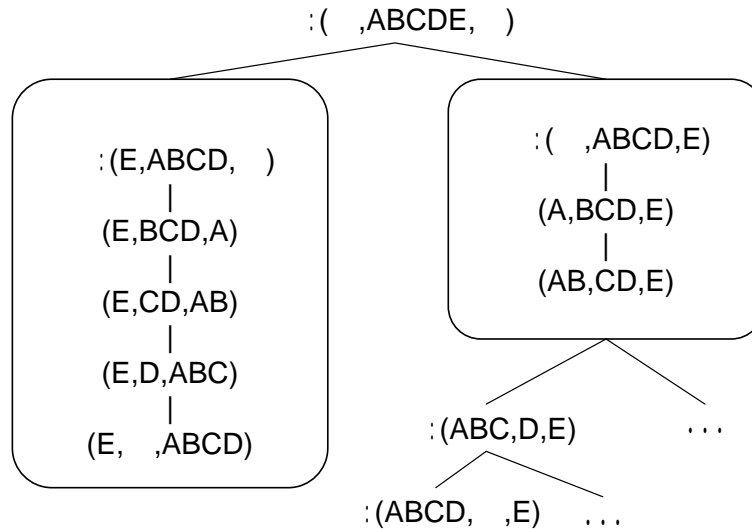


図 2-9 計算木の例[14]

### 2.3.8. $FIC^A$ , $FIC^M$ [12]

$FIC$  は、2001 年に J.Peï らによって提案された制約付き頻出パターン抽出手法[12]であり、このアルゴリズムは Convertible Constraint の制約付き頻出パターン抽出手法が実行できる。最初に Convertible Constraint が Apriori アルゴリズムに導入できないことを説明する。

Apriori の理論は「あるアイテム集合  $S$  がマイニングの条件を満たさなければ、 $S$  を含むアイテム集合はすべて候補アイテムから排除することができる」である。しかし、Convertible Constraint は Apriori のように候補アイテムを排除することができない。

たとえば、制約： $avg(S) \geq 25$ ，最小サポート値：2 とし、トランザクションデータベースとアイテムの profit は表 2-15 と表 2-16 を利用すると、 $fg$  は制約： $avg(S) \geq 25$  を満たしているが、 $fg$  のサブセットである  $g$  は制約を満たしていない。また、 $fg$  のスーパーセットである  $dfg$  も制約を満たしていないというケースがある。Apriori アルゴリズム”あるアイテム集合  $k$  が頻出パターンでなかった時、 $k$  を含むすべてのアイテム集合は頻出パターンではない”が成り立たず、Apriori はこのケースのような Convertible Constraint の候補アイテムを絞り込むことができない。

$FIC^A$  は Convertible Anti-Monotone の制約付き頻出パターン抽出の手法であり、 $FIC^M$  は Convertible Monotone の制約付き頻出パターン抽出の手法である。次に、各手法のアルゴリズムを述べ、 $FIC^A$  の実行例を述べる。

#### $FIC^A$

入力：トランザクションデータベース  $TDB$ ，最小サポート：，Convertible Anti-Monotone constraint  $C$ ，アイテム  $I$  を減少順にソートした  $R$

出力：  $C$  を満たす頻出パターン

メソッド：  $\text{fic}_A(\_, \text{TDB})$

関数：  $\text{fic}_A(\_, \text{TDB} | \_)$

引数：  $\_$  はアイテム集合であり，  $\text{TDB} | \_$  は  $\_$  を部分集合として含むトランザクションで構成されたデータベースである．

1.  $\text{TDB} | \_$  をスキャンして，  $\text{TDB} | \_$  の頻出アイテムを見つける．  
 $\forall a \in I_\alpha, C(\alpha \cup \{a\}) = \text{true}$  である  $\text{TDB} | \_$  内の頻出パターンを  $I$  とする
2. if  $I = \_$  return, else  $C$  を満たす頻出アイテムセットとして  $\alpha \cup \{a\}$  を出力する
3. if  $f$  が前置関数で  $C$  が  $f(S) \vee$  の形であれば ( $\_ \{ \_, \_ \}$ ),  $C$  を満たしている頻出アイテムセットではない  $b$  は  $I$  から取り除くことで最適化
4. もう一度  $\text{TDB} | \_$  をスキャンして，  $a \in I$  ,  $\{a\}$ -projected database  $\text{TDB} | \_{\{a\}}$  を生成
5.  $I$  内のそれぞれのアイテム  $a$  に， call  $\text{fic}_A(\_ \{a\}, \text{TDB} | \_{\{a\}})$

## FIC<sup>M</sup>

入力：トランザクションデータベース  $\text{TDB}$ ，最小サポート：  $\_$ ，制約：Convertible Monotone constraint  $\_$ ，アイテム  $I$  を減少順にソートした  $R$

出力：  $C$  を満たす頻出パターン

Method: Call  $\text{fic}_m(\_, \text{TDB}, 1)$

function:  $\text{fic}_m(\_, \text{TDB} | \_, \text{check\_flag})^2$

1.  $\text{TDB} | \_$  を一度スキャンする，  $\text{TDB} | \_$  の頻出アイテムを見つける．もし，  $\text{check\_flag}$  が 1 であれば，  $I^+$  を  $\text{TDB} | \_$  内の頻出アイテムセットで  $a \in I^+, C(\_ \{a\}) = \text{true}$  として  $I$  を  $\text{TDB} | \_$  内の頻出アイテムセットで  $b \in I, C(\_ \{b\}) = \text{false}$  とする．もし，  $\text{check\_flag}$  が 0 だったら，  $I^+$  は  $\text{TDB} | \_$  内で頻出アイテムセットであるが，  $I$  は  $\_$  である．
2.  $a \in I^+$ ，制約を満たす頻出アイテムセットとして  $\_ \{a\}$  を出力
3. もう一度  $\text{TDB} | \_$  をスキャン，  $a \in I^+ \setminus I$  ,  $\{a\}$ -projected database  $\text{TDB} | \_{\{a\}}$  を生成
4.  $I^+$  内のそれぞれのアイテム  $a$  に， call  $\text{fic}_m(\_ a[a], \text{TDB} | \_{\{a\}}, 0)$  .  $I$  内のそれぞれのアイテム  $a$  に， call  $\text{fic}_m(\_ a[a], \text{TDB} | \_{\{a\}}, 1)$  .

<sup>2</sup>  $\_$  は prefix としてのアイテムセットで，  $T | \_$  は  $\_$ -projected database である．  $\text{check\_flag}$  は制約をチェックするためのもの

表 2-15 トランザクションデータベース[12]

TID	Items in transaction
10	a b c d f
20	b c d f g h
30	a c d e f
40	c e f g

表 2-16 Item of Value[12]

Item	a	b	c	d	e	f	g	h
Value	40	0	-20	10	-30	30	20	-10

FIC<sup>A</sup> 実行例

トランザクション は表 2-15, アイテムの profit は表 2-16 を用い, 最小サポート値は 2, 制約  $avg(S) \geq 25$  とする.

最初に T を一度スキャンして, すべてのアイテムの出現数を数える. この時点で, h は一度しか出現しないので, この時点で振り落とされ, 頻出 1 アイテムセットは a, f, g, d, b, c, e となる(減少順に並べられる). 頻出 1 アイテムセットで C を満たしているものを考えると, a と f と言うことが分かる( $g(=20)$ 以降は 25 未満のため不適). よって, C を満たしている頻出アイテムセットは 2 つのサブセットに分けられる.

先頭に a を持っているアイテムセット

先頭に f を持ち, a を持っていないアイテムセット

1. a は制約を満たす頻出アイテムセットである. と同時に, 先頭に a を持った頻出アイテムセットは a を含むトランザクションのサブセットであることが分かる. これを *a-projected database* と呼ぶ. a は a-projected database 中のすべてのトランザクションに現れるので, 省略される. a-projected database は 2 つのトランザクションを含んでいる: bcdf, cdef. アイテム b と e はこの projected database では非頻出であり, ab や ae が頻出になることはない. よって ab や ae は取り除かれる. a-projected database の頻出アイテムセットは R 順に f, d, c である. ac は制約を満たしていないから, ac-projected database を作る必要はない.
2. それぞれ先頭から af と ad を持った a-projected database をマイニングするために, 2 つの projected database を構築する必要がある, ここをさらにマイニングする. この過程は a-projected database のマイニングと同様に行う. af-projected database は d と c の 2 つの頻出アイテムを含んでおり, afd だけが制約を満たしている. さらに afdc は制約を満たしていないので, この枝の過程は完成する. afc は制約により排除できるから, afc-projected database を構築する必要はない. ad-projected database は頻出アイテムとして c を含んでいるが, adc は制約を満たしていない. それゆえ, 制約を満たし, a を先頭に含む頻出アイテムセットは a, af, afd, ad である.

3. 同様に f-projected database は f を含んだトランザクションのサブセットであり, a と f は取り除かれている。これは bcd, bcdg, cde と ceg の 4 つ存在する。この projected database 内の頻出アイテムは R 順に g, d, b, c, e である。fg と fd だけは制約を満たしているので, fg と fd を持った頻出アイテムセットを検索するだけでよい。fg-projected database は制約を満たして fg を先頭に持った頻出アイテムセットを含んでいない。b は R 順で d のすぐ後に出てくるアイテムであり, fdb が制約を満たさないので, fd を先頭に持つアイテムセットはすべて制約を満たさない。それゆえ, f を先頭に持ち, 制約満たす頻出アイテムセットは f と fg の 2 つである。

よって, この例で制約を満たす頻出アイテムセットは a, f, af, ad, afd, fg の 6 つである。とマイニングできる。以上の過程を図として表現したものが, 図 2-10 FICA の実行例 図 2-10 である。

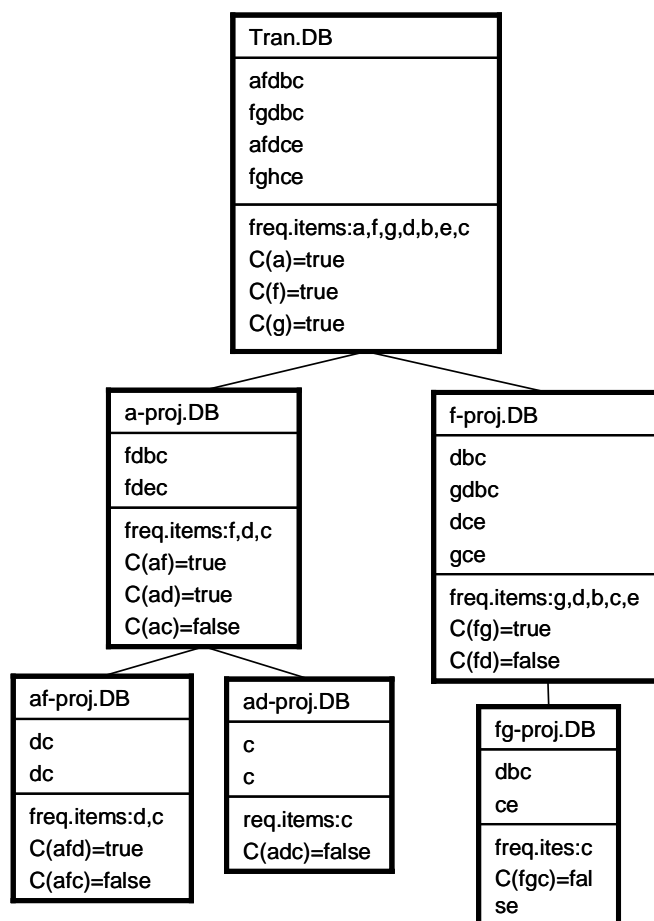


図 2-10 FICA の実行例[12]

## 2.4. まとめ

制約付き頻出パターン抽出のアルゴリズムをまとめたものが表 2-17 である。

表 2-17 制約付き頻出パターン抽出のアルゴリズム

アルゴリズム	提案年	提案者	制約	アイテムの profit	概要
CFG[1]	2000	J. Pei J. Han	Anti-monotone (Convertible Anti-monotone)	非負数	制約にも Apriori アルゴリズムを適用 データ構造を射影
ExMiner[13]	2003	F.Bonchi A.Mazzanti	Anti-monotone Monotone	非負数	2 つの枝狩りを実装し, Anti-Monotone , Monotone の 2 つに対応
DualMiner[14]	2003	C.Bucila J.Gehrke D.Kifer	Anti-monotone Monotone	非負数	2 つの制約付き頻出パターン抽出が可能 独自の計算木を利用
FIC <sup>A</sup> [11][12]	2001	J.Pe J.Han L.V.S.Lakshmanan	Convertible Anti-monotone	負数を含む	Strongly Convertible の制約に対応 データ構造を射影
FIC <sup>M</sup> [11][12]			Convertible Monotone	負数を含む	

## 第3章 提案手法

### 3.1. 従来手法の問題点

第2章で説明した制約付き頻出パターン抽出手法を実行する場合、ユーザには「最小サポート」と「制約の種類」と「制約の閾値」の3つの入力を求める必要がある。頻出パターン抽出手法は最小サポートの一つのみの入力でもよかったが、制約付き頻出パターン抽出手法は新たに制約の種類と制約の閾値の2つの入力が必要になっている。ユーザによる入力値が増えたことにより、ユーザは必要としている情報を絞り込んでマイニングを実行することが可能になった。一方で、ユーザは3つの入力値を入力する。頻出パターン抽出数は、最小サポートや制約の設定によって大きく依存する。例えば、制約の閾値が低すぎてユーザが把握できないほどの膨大な頻出パターンが抽出されたり、逆に制約の閾値が高すぎて十分な数の頻出パターンが抽出できなかったりケースが考えられる。このようなケースでは、ユーザは新たに入力値を設定し直して、新たにマイニングを実行しなくてはならない。しかし、初めてマイニングを実行させるデータベースのデータの濃度、偏りは分からないので、適切な入力値をユーザが設定することは非常に困難である。この問題を解決するために入力値を削減する手法を本章で提案する。

### 3.2. 提案手法の概要

3.1に挙げた問題点を解決するため、従来の条件付き頻出パターン抽出手法で必要となっていた3つの入力値、「最小サポート」「制約の種類」「制約の閾値」を、「最小サポート」と「制約の種類」の2つに削減した制約付き頻出パターン抽出手法を提案する。提案手法では「制約の閾値」の入力を必要としないので、ユーザの負担が軽減される。

提案手法は、最小サポートを満たす候補アイテム集合を抽出しつつ、内部で自動的に制約の閾値を設定している。閾値を徐々に下げて(上げて)行くことで、閾値を満たす制約値を持つ頻出パターンを順次抽出する手法である。内部で使用する制約の閾値は、ユーザが気にする必要はない。本論文では、制約のなかで扱いが難しい Convertible Anti-Monotone ( $avg(X), sum(X)$ ) の制約付き頻出パターン抽出についての提案手法の導入を行った。

### 3.3. 提案手法のアルゴリズム

制約の閾値入力を必要としない Convertible Anti-Monotone の制約付き頻出パターン抽出のアルゴリズムを以下に述べる。

入力値：最小サポート，制約の種類(平均値[avg]合計[sum]の昇順・降順)

出力値：制約を満たす頻出パターン，制約を満たす頻出パターンの閾値

1. 制約の種類として「降順」が指定された場合は、アイテムをアイテムの profit を基に降順にソートする。制約の種類として「昇順」が指定された場合は、アイテムをアイテムの profit を元に昇順にソートする。
2. データベースから最小サポートを満たさない非頻出アイテムを削除し、各トランザクションのアイテムを 1 と同様にソートする。ここで、頻出アイテムの種類数を  $m$  とする。
3.  $i=1$  とする。
4. ソートしたアイテム列の中で  $i$  番目に並べられているアイテムの profit を  $\alpha$  とする。
5. 閾値を  $\alpha$  とした状態で、制約付き頻出パターン抽出を実行し、最小サポート値を満たし、かつ、制約関数の値が  $\alpha$  以上のパターンを抽出する。
6.  $i < m$  を満たす場合は、 $i$  をインクリメントして、4 に戻る。 $i = m$  を満たす場合は、終了する。

4, 5 の過程では一度マイニングをしたデータベースに対して、反復してスキャンをする必要がないように、独自のデータ構造を利用する。このデータ構造の説明を含めて実行例を次節で述べる。

### 3.4. 実行例

入力値：最小サポート値：2，制約の種類：平均値( $avg(S.profit)$ )の降順

出力値：制約を満たす頻出パターン，制約の閾値

最小サポートを”2”，制約の種類は”平均(avg)の降順”とする。表 3-19 に示す Profit が与えられたアイテム集合に対し、表 3-18 に示すようなデータベースを対象として実際にパターンを抽出する過程を示す。3.3 で示したアルゴリズムの手順 5 で行う制約付き頻出パターン抽出は、FICA アルゴリズム[4]を利用している。

トランザクションデータベースは表 3-18，アイテムの profit には表 3-19 を利用する。トランザクションのアイテムをソートしたものが表右側の Sorted Item であり，アイテムの Profit をソートしたものが下の段にあたる。次に初期設定として，ルートを作る。ルートは TID に 100 から 500 まですべてのトランザクションとリンクし，各トランザクションの先頭アイテムを指す(図 3-11 を参照)。



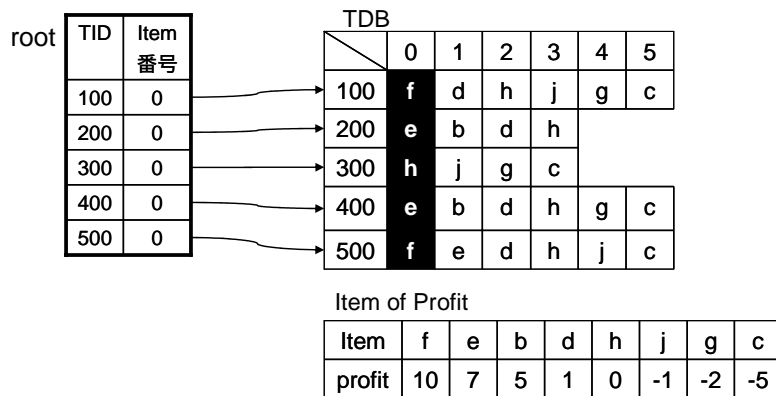


図 3-11 実行例の初期設定

1. 表 3-19 よりアイテムの中で最大の profit であるアイテム”f”に注目し，内部で利用する制約の閾値を， $avg \geq 10$ と設定する．ルートに対してアイテム”f”の頻出回数をカウントすると  $TID=100,500$  に2回出現し，最小サポートを満たすので，アイテム”f”に関する射影を構築する．これを f-proj.とする．f-proj.は TID の 100 と 500 を持ち，リンク先として，アイテム”f”の次に出現するアイテムを指すため 1 を保持する(図 3-12 を参照)．他に  $avg \geq 10$ を満たす頻出パターンがないので探索を終了する．ここで抽出された頻出パターンは{f}のみである．

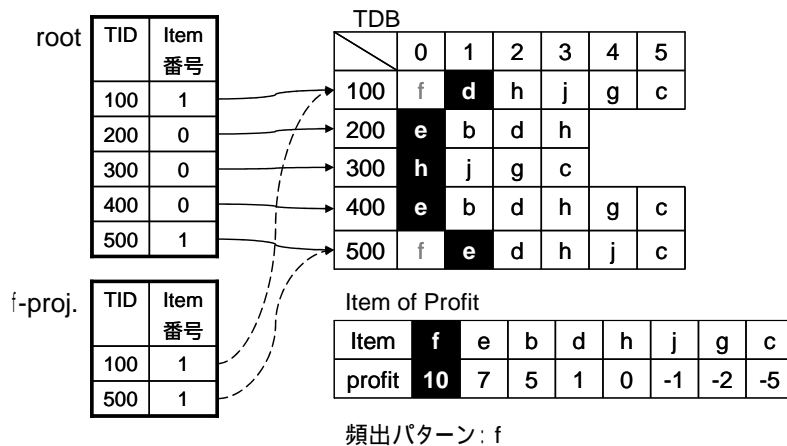


図 3-12 実行例 f-proj.終了時点

2. 2 番目に profit が大きいアイテム”e”に注目し， $avg \geq 7$ と設定し直す．f-proj.で  $avg \geq 7$ を満たす候補アイテムはアイテム”e”と”b”のみである．次にアイテムの profit が大きいアイテム d では  $avg \geq 7$ を満たさない．まず，f-proj.についてアイテム”e”の頻出回数をカウントすると， $TID=500$  に1回出現するのみである．これはパターン{f,e}が最小サポートを満たさないことを意味する．次にアイテムの profit 大きいアイテム”2”について計算するが，最小サポートを満たさないので探索終了．ルートに対してアイテム”e”の頻出回数をカウントすると  $TID=200,400,500$  に3回出現し，最小サポートを満たすので，アイテム”5”に関する射影を構築する．これを e-proj.とする(図 3-13 参照)．よって，ここで抽出された頻出パターンは{e}のみである．

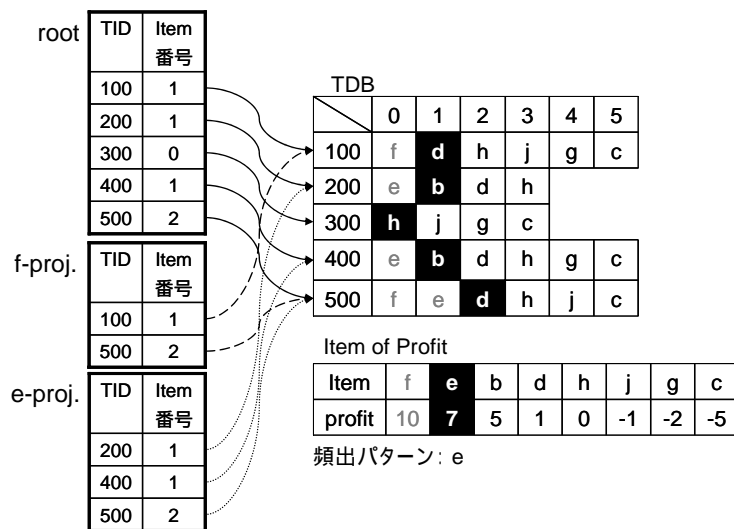


図 3-13 実行例 e-proj.終了時点

3. 3 番目に profit が大きいアイテム”b”に注目し， $avg \geq 5$ と設定する．e-proj.で  $avg \geq 5$ を満たす候補アイテムはアイテム”b”のみである．そこで，アイテム”b”の頻出回数をカウントすると，TID=200 と 400 の 2 回出現するので，eb-proj.を構築する．f-proj.で  $avg \geq 5$ を満たす候補アイテムはアイテム”d”と”h”であり，同様に頻出回数をカウントするとアイテム”d”と”h”は最小サポート値を満たす．f-proj.では， $\{f,d\}$  $\{f,h\}$ の頻出パターンが検出されたので，fd-proj.と fh-proj.が構築する．最後にルートに対してアイテム”b”の頻出回数をカウントすると，最小サポート値を満たすので，b-proj.を構築する(図 3-14 参照)．よって，ここで抽出された頻出パターンは $\{e,b\}$ ， $\{f,d\}$ ， $\{b\}$ ， $\{f,h\}$ である．

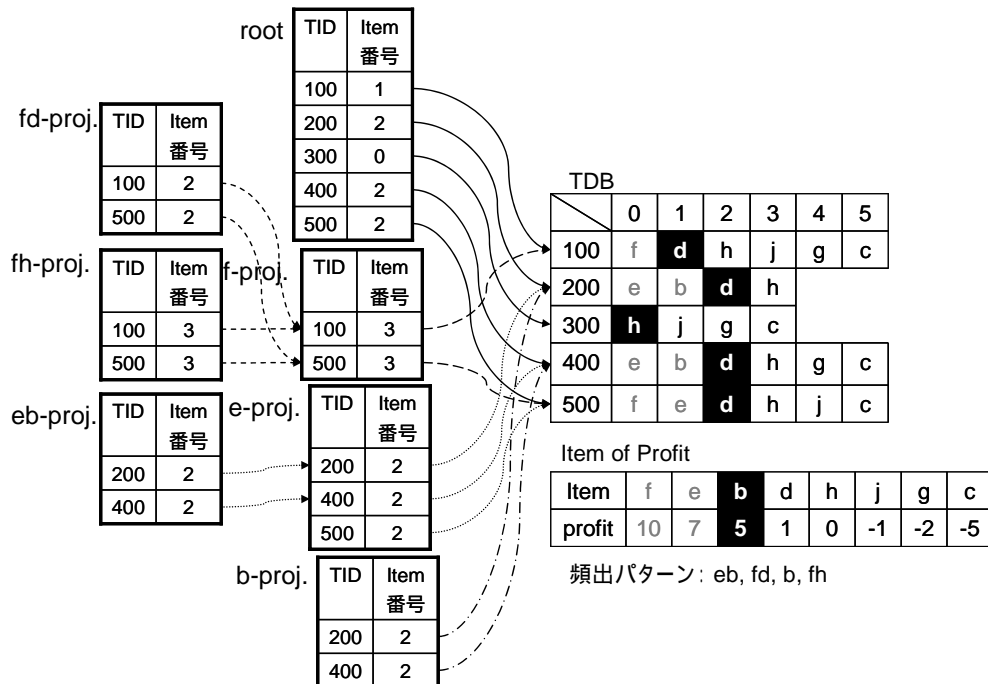


図 3-14 実行例 b-proj.終了時点

以上を繰り返して，頻出パターンを検索する．h-proj.まで終了した時点では，図 3-15

のようになっている。TID=200 に注目すると、すべてのアイテムの探索が終了していることが分かる。よって、TID=200 を含む頻出パターンはこれ以降、抽出される可能性はなくなる。TID=200 が排除されることにより、例えば、eb-proj.から先の proj.は TID=400 しかトランザクションが存在せず、トランザクションが一つでは最小サポート値の”2”を満たさない。このようにデータ構造のトランザクション数が最小サポート値を満たさなくなった場合は、探索の対象から外れる。

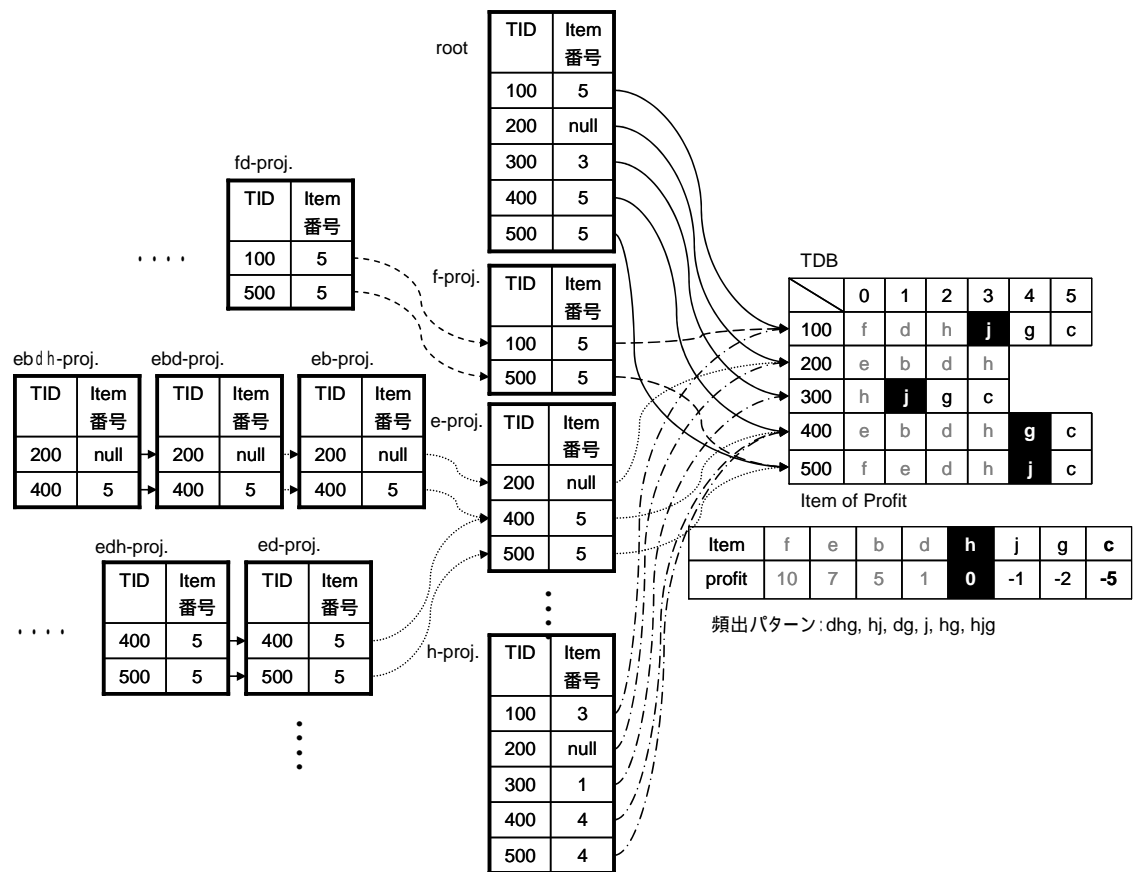


図 3-15 実行例 h-proj.終了時点

最終的に、c-proj.の構築をすると、すべての頻出パターンを抽出することができる。c-proj.構築後では、TDB をすべてスキャンし終えているので、データ構造の Item 番号はすべて null になる(図 3-16 を参照)。提案手法の実行例の出力例を付録 2 に載せてある。

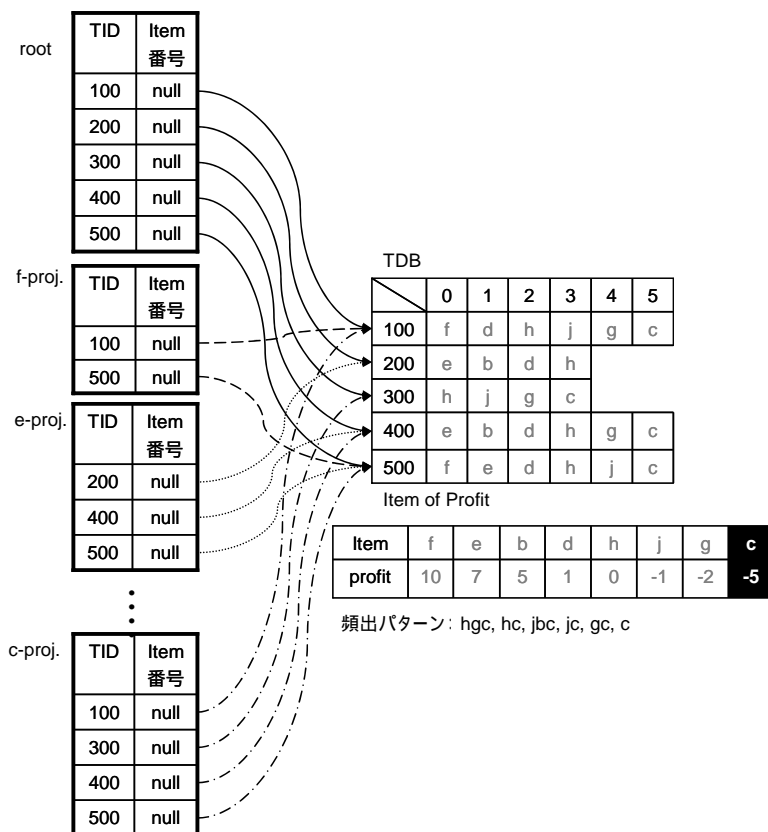


図 3-16 実行例 c-proj.終了時点

表 3-18 トランザクションデータベース

TID	Item in transaction	Sorted Item
100	a f g j c h d	f d h j g c
200	d e h b	e b d h
300	j i c h g	h j g c
400	g e d b h c	e b d h g c
500	f h j c d e	f e d h j c

表 3-19 アイテムの Profit

Item	a	b	c	d	e	f	g	h	i	j
Profit	-10	5	-5	1	7	10	-2	0	3	-1
profit 降順にソート										
Item	f	e	b	i	d	h	j	g	c	a
Profit	10	7	5	3	1	0	-1	-2	-5	-10

### 3.5. その他 ( $avg(X)$ , $sum(X)$ 以外) の関数の提案手法適用

提案手法は Convertible Constraint を対象にしているが, アイテム列をソートする

ことで、制約付き頻出パターン抽出を行っている。そのため、 $\min(X) \cdot \max(X) \cdot \text{count}(X) \cdot \text{range}(X)$  の関数に対しても提案手法を適用することはできる。しかし、提案手法では、候補アイテム集合を生成する際に制約を満たしているか毎回確認しているが、 $\min(X) \cdot \max(X) \cdot \text{count}(X) \cdot \text{range}(X)$  にはその必要がない。必要性のない処理を含み、処理速度が低下するので、処理工程を省く必要がある。

例えば、 $\min(X)$  に関しては、アイテム列昇順  $R^1$  を基に prefix で候補アイテム集合を生成すれば、最小の  $S.\text{profit}$  を持つ頻出パターン順に抽出することが可能である。実行例を次に示す。

### min(X)の実行例

入力値：最小サポート：2，制約の種類： $\min(X)$  昇順

出力値：制約を満たす頻出パターン，制約を満たす頻出パターンの閾値

トランザクションデータベースには表 3-20，アイテムの profit には表 3-21 を参照。

1. 各トランザクション，アイテム列は昇順にソートする(表 3-20 表 3-21 参照)。
2. 頻出アイテムで最小の profit を持つアイテム"c"に注目し，閾値を 5 に設定する。
3. アイテム"c"を含むすべての頻出パターンを抽出する。
4. 次に，アイテム"g"に注目し，閾値を-2 に設定する。アイテム"c"は3の段階ですべて探索が終了しているので，ここでは"c"を含まないが，"g"を含むすべての頻出パターンを抽出する。

この過程を繰り返して，閾値を上げつつ制約付き頻出パターンを抽出する。

提案手法では，制約を満たす候補アイテム集合を生成して，頻出パターンを探索しているが， $\min(X)$  の実行例では削除されている。この  $\min(X)$  の場合は Monotone であり，以降は制約を満たすか判断する必要はないからである。

表 3-20 トランザクションデータベース

TID	Item in transaction	Sorted Item
100	a f g j c h d	c g j h d f
200	d e h b	h d b e
300	j i c h g	c g j h
400	g e d b h c	c g h d b e
500	f h j c d e	c j h d e f

表 3-21 アイテムの profit

Item	a	b	c	d	e	f	g	h	i	j
Profit	-10	5	-5	1	7	10	-2	0	3	-1
profit 昇順にソート										
Item	a	c	g	j	h	d	i	b	e	f
Profit	-10	-5	-2	-1	0	1	3	5	7	10

## 第4章 評価

### 4.1. 実験環境

実験環境は表 4-22 とおりである .

表 4-22 実験環境

PC	Panasonic Let's Note CF-T1
CPU	Pentium 866MHz
メモリ	504MB
コンパイラ	Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
OS	Windows XP Professional SP2

### 4.2. 実験に用いたデータセット

実験には Tom Brijs が提供している "Retail Market Basket Data Set"[7]を利用させてもらっている .このデータセットは 1999 年 12 月中旬から 2000 年 1 月中旬 ,2000 年 6 月 , 2000 年 8 月末から 11 月末までの計 5 か月分のドイツのスーパーマーケットの販売データである . 総アイテム数は 16469 , トランザクション数は 88162 である . 各トランザクションは一人の顧客が購入した買い物商品がアイテム番号 1 から 16469 の数値で記述されている . 各アイテム番号の商品名 , 各商品の価格などは提供されていない . 制約付き頻出パターン抽出手法を実行するには各アイテムの Profit を用意する必要があるので , Profit は -5000 から 5000 の範囲でランダムに設定した(同じ Profit 値を持つアイテムも存在する) . データセット中の各アイテムの出現数は付録に掲載してある .

### 4.3. 提案手法の評価

提案手法の平均(Avg)を実装し , 次の条件で提案手法の評価を行った( 図 4-17 , 図 4-18 , 図 4-19 , 図 4-20 , 図 4-21 参照) . データセットとして Retail Market Basket Data Set , 制約は Convertible Anti-Monotone の制約である平均値  $avg(S.profit) \leq \theta$  とする . 最小サポートは 0.2% , 0.4% , 0.6% , 0.8% , 1.0% の 5 段階とする . 各評価とも頻出パターンを抽出し , 制約である平均値を基にソートする従来手法と , 制約を満たす上位頻出パターンから順次返す提案手法の 2 つを比べている . 従来手法は , H-mine をベースにしており , 提案手法は , H-mine と FICA をベースにして , 提案手法のデータ構造を利用している . また , すべてのソーティングはクイックソートを利用している .

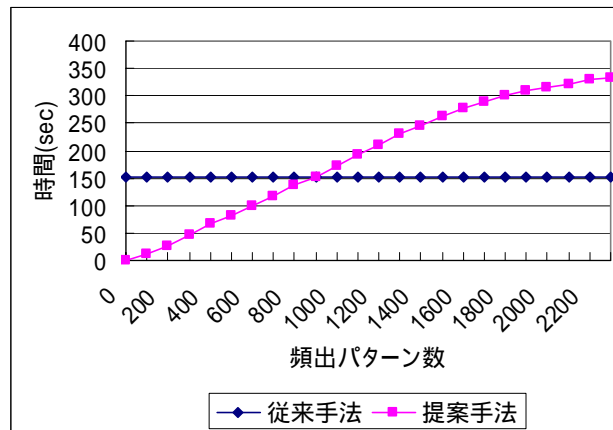


図 4-17 最小サポート 0.2%の実行時間比較(頻出パターン数：2249)

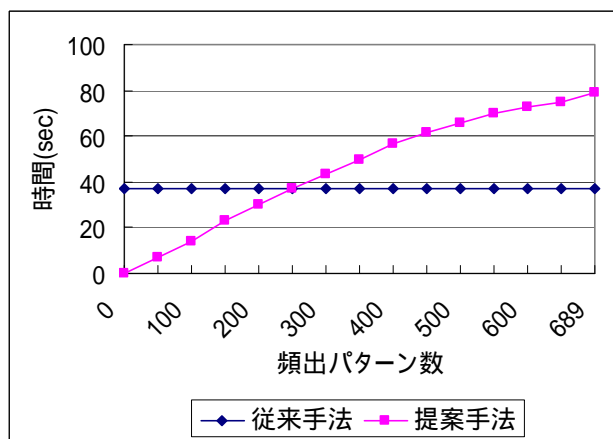


図 4-18 最小サポート 0.4%の実行時間比較(頻出パターン数：689)

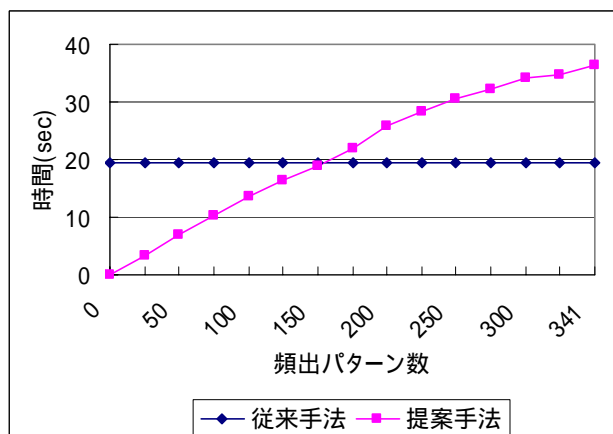


図 4-19 最小サポート 0.6%の実行時間比較(頻出パターン数：341)

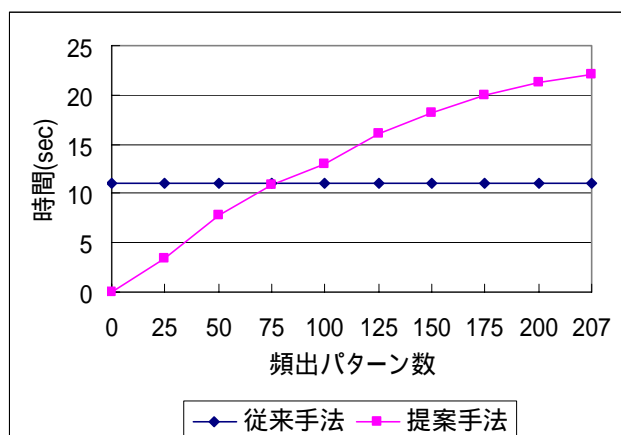


図 4-20 最小サポート 0.8% の実行時間比較(頻出パターン数 : 207)

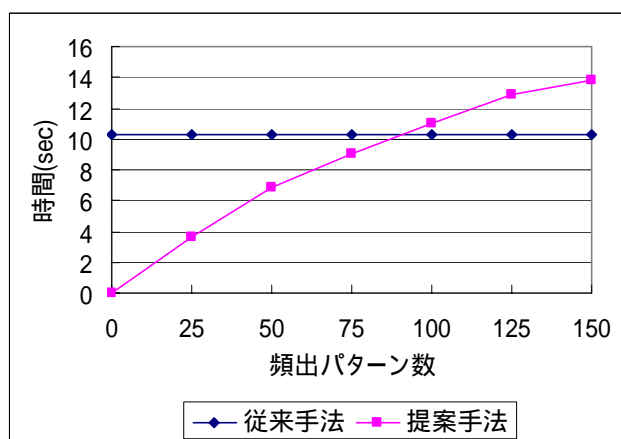


図 4-21 最小サポート 1.0% の実行時間比較(頻出パターン数 : 150)

表 4-23 提案手法と従来手法の対比

最小サポート(%)	0.2	0.4	0.6	0.8	1.0
提案手法の前処理時間(sec)	2.70	1.46	1.04	0.80	0.75
提案手法の抽出時間(sec)	332	77.2	35.5	21.2	13.1
提案手法の合計実行時間(sec)	334	78.7	36.5	22.0	13.8
従来手法の前処理時間(sec)	2.57	2.89	3.2	2.61	2.56
従来手法の抽出時間(sec)	139	41.6	15.0	10.4	7.53
従来手法のソート時間(sec)	0.25	0.021	0.01	0.01	0.00
従来手法の合計実行時間(sec)	141	44.5	18.2	13.0	10.0
提案手法 — 従来手法	2.4	1.9	2.4	2.0	1.7

表 4-23 から提案手法は従来手法に比べて、すべての頻出パターンを抽出するには 2 倍前後の抽出時間が掛かる。提案手法では、閾値の変更後の制約付き頻出パターン抽出



をするために、データ構造を用意する必要がある。このデータ構造の構築や呼び出しのためのオーバーヘッドが掛かるので、従来手法と比べると多くの時間を必要とする。閾値の変更とデータ構造の関係を表したものが、図 4-22 である。これは、最小サポート値が 0.2%の時、閾値の変更(閾値を下げることに伴うデータ構造数の増減を表している。図 4-22 から、前半は閾値を変更する毎にデータ構造数が増加するが、後半は逆にデータ構造数が減っていることが分かる。これは、提案手法の実行例で示した図 3-15 のように、すべての抽出を終えたデータ構造が探索の対象から外れるためである。同様に、図 4-17 から図 4-21 の実験結果でも、後半部分は抽出時間が短縮されている。

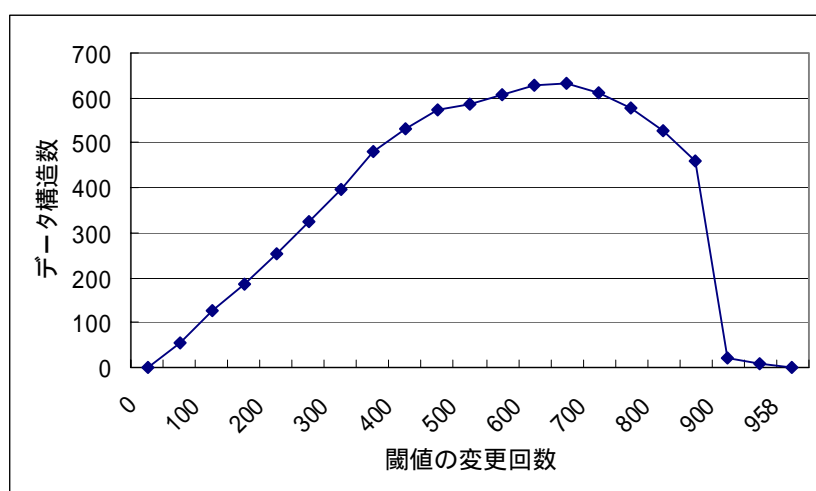


図 4-22 閾値の変更とデータ構造の関係

実験結果から、提案手法では全頻出パターンの 4 割前後の頻出パターンに関して、従来手法より短い時間で頻出パターンをユーザに返すことが可能になっている(表 4-24 参照)。従来手法では制約条件を満たす上位 100 パターンのみを抽出したい場合、すべての頻出パターンを抽出してソートするか、制約の閾値を設定し直して頻出パターンを抽出してきた。提案手法では、最小サポートの閾値が高く頻出パターン数が少ないケースでは、従来手法に比べて上位 100 パターンを抽出する時間が多く掛かっている。しかし、最小サポートが低いケースでは、提案手法の方が早い段階で頻出パターンを抽出できている。また、従来手法では最小サポートの設定によって抽出時間が大きく変化するが、提案手法では、上位 n パターンを最小サポート値に関係なく抽出が可能ということである。

**表 4-24 提案手法の評価**

最小サポート(%)	0.2	0.4	0.6	0.8	1.0
頻出パターン数	2249	689	341	207	150
提案手法が早く見つける頻出パターン数の上限	886	253	157	78	86
— (%)	39.3	36.7	46.0	37.6	57.3
上位 100 パターン 抽出時間(sec)	12.5	13.3	13.8	13.1	11.5

## 第5章 おわりに

本論文では、制約付き頻出パターン抽出手法におけるユーザビリティ向上のため、制約の閾値を動的に設定することで、入力値を削減する手法を提案した。提案手法を基に、Convertible Anti-Monotone の制約付き頻出パターン抽出手法を実装し、データセット Retail Market Basket Data Set に対して評価を行った。その結果、制約の閾値なしで制約付き頻出パターン抽出が行えるようになり、ユーザがより簡単に制約付き頻出パターン抽出手法を利用することが可能となった。従来手法に比べて提案手法では、制約値上位の頻出パターンの約 4 割弱を早く抽出することができ、ユーザは制約付き頻出パターンを短時間で把握できることが可能となった。また、上位パターンを順次抽出する手法を採用しているため、上位  $n$  パターンを最小サポート値に関係なく一定の時間で抽出が可能であることが確認できた。

また、今後の課題として、提案手法の高速化がある。本研究では、全頻出パターンの 4 割弱を従来手法より短時間で抽出することが可能である。マイニング中のデータ構築のオーバーヘッドを削減し、従来手法と提案手法の全頻出パターン抽出時間の差を縮小することで、より短時間でユーザに頻出パターンを返すことが望まれる。

2 つ目に最小サポートと制約の閾値との関係性を考えた制約付き頻出パターン抽出が問題となる。本論文では、制約の閾値の入力を必要としなかったが、最小サポートの入力は必要としている。最小サポートも 1 つの制約であり、閾値である。そこで、2 つ以上の制約に対して自動的に閾値を設定して、制約付き頻出パターン抽出手法が考えられる。2 つの閾値を自動化することにより、TDB の特徴に合わせた制約付き頻出パターン抽出が可能になる。

## 謝辞

本論文の作成にあたり，数々のご指導をいただいた山名早人助教授，また，丁寧な助言を与えてくださった平手勇宇さん齊田直幸さんに感謝致します．

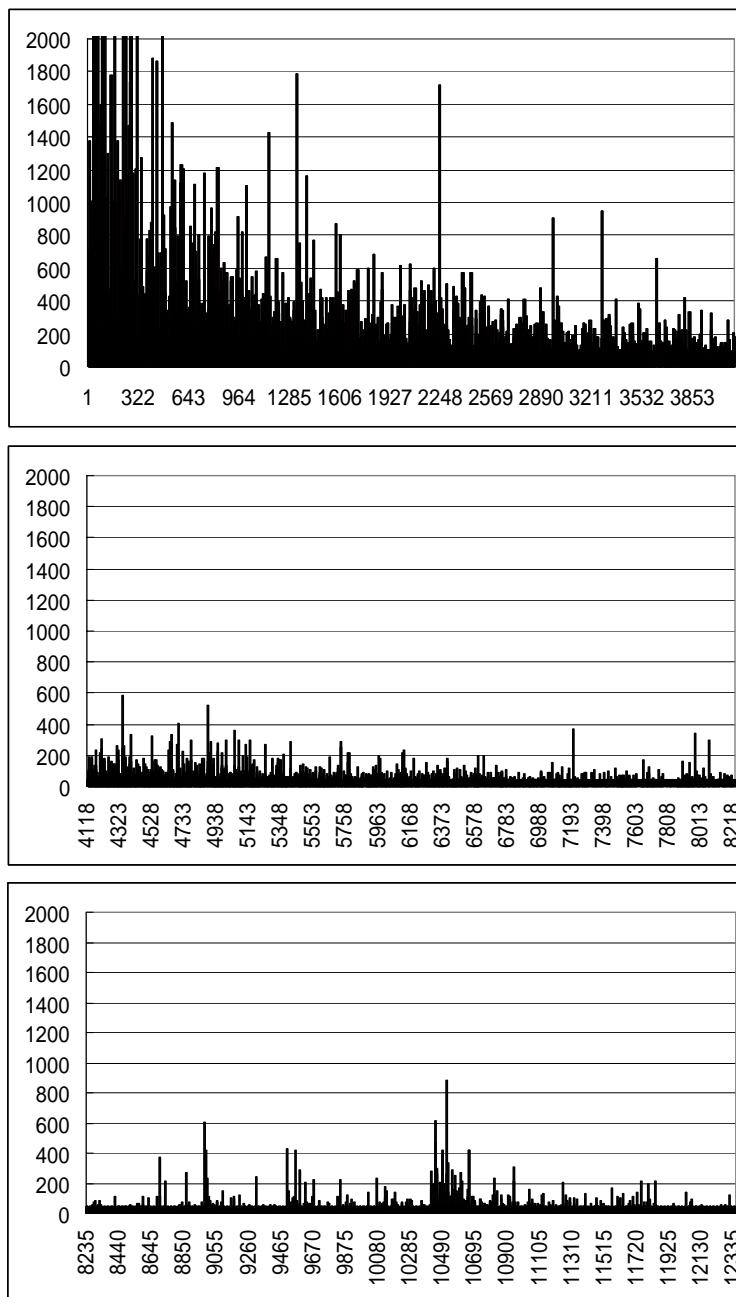
## 参考文献

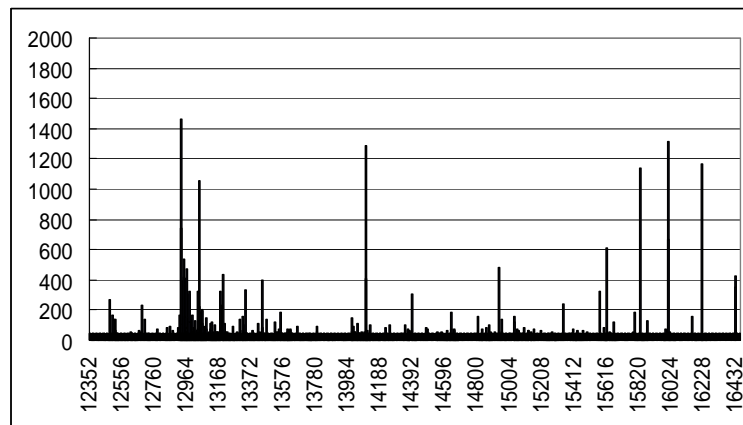
- [1] J. Pei and J. Han. "Can we push more constraints into frequent pattern mining?" In Proceedings 2000 International Conference Knowledge Discovery and Data Mining (KDD'00), pp.350–354, Aug. 2000.
- [2] R.Agrawal and R.Srikant. "Fast algorithms for mining association rules", In VLDB'94, pp.487-499, 1994.
- [3] J.Pei , "Frequent-pattern Mining" <http://www.cs.buffalo.edu/faculty/jianpei/>
- [4] J.Han, J.Pei and P.S.Yu, "Mining frequent Patterns without Candidate Generation," In Proceedings of the ACM SIGMOD Conference on Management of Data pp.1-12, 2000.
- [5] Roberto J. and Bayard Jr., "Efficiently mining long patterns from databases", In Proceedings of the ACM SIGMOD Conference on Management of Data, pp.85-93, 1998.
- [6] J.Pei, J.Han and R.Mao, "CLOSET: An efficient algorithm for mining frequent closed itemsets", In DMKD'00, 2000.
- [7] T.Brijs, "retail, Frequent Itemset Mining Dataset Repository", "<http://fimi.cs.helsinki.fi/>"
- [8] 福田剛志,森本康彦,徳山豪,"データマイニング",共立出版,6月10日2002.
- [9] J.Pei, J.Han, H.Lu, S.Nishio, S.Tang and D.Yang, "H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases", In Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM'01), pp.441-448,2001.
- [10] Roberto J. and Bayard Jr., "Efficiently mining long patterns from databases", In Proceedings of the ACM SIGMOD Conference on Management of Data, pp.85-93, 1998.
- [11] J.Pei, J.Han and L.V.S.Lakshmanan, "Pushing Convertible Constraints in Frequent Itemset Mining", (NCE/IRIS-3), 2004.
- [12] J.Pei, J.Han and L.V.S.Lakshmanan, "Mining frequent itemsets with convertible constraints", In Proceedings 2001 International Conference Data Engineering (ICDE'01), pp.433–332, April 2001.
- [13] F.Bonchi and A.Mazzanti, "ExAMiner:Optimized Level-wise Frequent Pattern Mining with Monotone Constraints", In Proceedings of ICDM'03, 2003.
- [14] C.Bucila, J.Gehrke, D.Kifer and W.White, "DualMiner: A dual-pruning algorithm for itemsets with constraints", Data Mining and Knowledge Discovery, 7(4):pp.241-272, 2003.

## 付録

### 1. Retail Market Basket Data Set の出現回数

(X 軸：商品アイテム番号，Y 軸：TDB 出現回数)





## 2. 提案手法の実行例

提案手法の実行例を示す．提案手法では，アイテムが英字であるが，実行例では下図のように数字で置き換えてある．

a	b	c	d	e	f	g	h	i	j
1	2	3	4	5	6	7	8	9	10

閾値を変化させる毎に，閾値とその閾値で頻出された頻出パターンと抽出に掛かった時間を表示している．カッコ内の数値は，制約関数の数値である．実行例では，制約関数に平均値を用いている．

### 出力結果

Sort start

Sort end

-mining start-

6 (10)

頻出パターン数 1 avg 10 time : 0

5 (7)

頻出パターン数 2 avg 7 time : 0

2 5 (6)

4 6 (5.5)

2 (5)

8 6 (5)

頻出パターン数 6 avg 5 time : 0.01

10 6 (4.5)

4 2 5 (4.33333)

4 5 (4)

8 2 5 (4)

8 4 6 (3.66667)

8 5 (3.5)

10 4 6 (3.33333)

4 8 2 5 (3.25)

4 2 (3)

10 8 6 (3)

4 8 5 (2.66667)

8 2 (2.5)

3 6 (2.5)

8 10 4 6 (2.5)

3 4 6 (2)

4 8 2 (2)

3 8 6 (1.66667)

8 3 4 6 (1.5)

10 3 6 (1.33333)

10 3 4 6 (1.25)

4 (1)

8 10 3 4 6 (1)

10 3 8 6 (1)

4 3 5 (1)

3 5 (1)

頻出パターン数 31 avg 1 time : 0.03

3 4 8 5 (0.75)

3 8 5 (0.666667)

8 4 (0.5)

8 (0)

10 4 (0)

8 10 4 (0)

頻出パターン数 37 avg 0 time : 0.04

7 8 4 (-0.333333)

10 8 (-0.5)

7 4 (-0.5)

10 (-1)

7 8 (-1)

10 7 8 (-1)

頻出パターン数 43 avg -1 time : 0.04

3 8 10 4 (-1.25)

3 8 4 (-1.33333)

7 10 (-1.5)

3 7 8 4 (-1.5)

3 10 4 (-1.66667)

3 10 7 8 (-2)

7 (-2)



3 7 4 (-2)

3 4 (-2)

3 10 8 (-2)

頻出パターン数 53 avg -2 time : 0.06

3 7 8 (-2.33333)

3 8 (-2.5)

3 7 10 (-2.66667)

3 10 (-3)

3 7 (-3.5)

3 (-5)

頻出パターン数 59 avg -5 time : 0.07

time : 0.070000

sort time : 0.000000

Mining time : 0.070000

### 3. 提案手法の評価結果(抽出時間(sec))

		最小サポート									
		0.10%	0.20%	0.30%	0.40%	0.50%	0.60%	0.70%	0.80%	0.90%	1.00%
頻出パターン	0	0	0	0	0	0	0	0	0	0	0
	25	6.56	4.99	4.35	3.55	3.80	3.46	3.30	3.47	3.48	3.67
	50	9.06	6.94	6.70	6.87	6.86	6.85	7.41	7.84	7.48	6.97
	75	11.9	9.21	10.5	10.1	10.3	10.4	10.2	10.8	10.1	9.00
	100	13.9	12.7	14.2	14.1	14.2	13.6	13.3	13.0	13.3	11.0
	125	15.8	15.8	17.6	19.6	16.9	16.4	16.2	16.1	15.0	13.8
	150	18.5	19.9	21.3	23.1	20.8	18.9	19.2	18.2	17.0	
	174									18.0	
	175	22.1	23.4	25.4	27.1	24.6	22.0	21.9	20.0		
	200	25.1	27.1	30.1	29.9	27.9	25.7	23.7	21.2		
	207								22.0		
	250	33.2	37.8	38.4	37.3	34.2	30.5	26.3			
	264							28.0			
	300	39.4	47.1	46.6	43.7	40.5	34.1				
	341						36.5				
	350	47.7	56.5	54.0	50.0	45.0					
	400	53.7	65.8	62.2	56.5	47.8					
	450	61.7	73.9	70.0	61.8	50.8					
	461					52.0					
	500	68.8	81.9	78.0	66.0						
	550	79.0	90.0	85.7	70.0						

	600	89.3	99.1	92.4	72.7
	650	99.6	108	100	75.1
	689				78.7
	700	111	118	107	
	750	123	126	115	
	800	135	136	122	
	850	147	145	128	
	900	160	153	133	
	950	174	164	139	
	1000	185	172	143	
	1100	210	192	151	
	1200	240	211	156	
	1300	263	230	164	
	1305			164	
	1400	290	246		
	1500	314	262		
	1600	344	276		
	1700	375	289		
	1800	410	300		
	1900	450	309		
	2000	481	315		
	2100	510	321		
	2200	542	330		
	2249		334		
	2300	571			
	2400	600			
	2500	633		4400	1312
	2600	665		4600	1367
	2700	693		4800	1422
	2800	718		5000	1468
	2900	743		5200	1501
	3000	781		5400	1521
	3200	849		5600	1535
	3400	915		5800	1549
	3600	1015		6000	1561
	3800	1092		6200	1572
	4000	1182		6400	1587
	4200	1259		6485	1597