2004 年度卒業論文

アイテム間の距離を考慮した Sequential Pattern Mining の提案

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：1G01P043-6

小松　俊介

概要

近年の技術の進歩によって記憶装置の大容量化が進み、膨大な量のデータを人間が直接扱うことが困難になってきている。そこで、膨大なデータの中から有用な情報を取り出す技術としてデータマイニング技術が注目されている。データマイニングの中で重要とされる問題が、頻出パターンと呼ばれる大量のデータの中から頻繁に出現するアイテムの組合せを抽出する（頻出パターン抽出 : Frequent Pattern Mining）問題である。Frequent Pattern Mining はユーザが与えた最小サポート値よりも高い頻度で出現するアイテムの組合せを抽出するが、アイテム同士の順序は考えられていない。しかし、アイテム間の順序が重要となる事例が世の中には数多くある。そこで、考え出されたのが Sequential Pattern Mining である。Sequential Pattern Mining ではアイテム間の順序を考慮してマイニングを行う。アイテム間の順序を考慮することによってより現実に即したマイニングが可能となった。しかし、Sequential Pattern Mining ではアイテム間の距離（時間）については考慮していないため、抽出されたアイテムセット中の任意の２つのアイテム間に，どれだけの距離があるのかを区別することが出来ない。しかし、距離を区別することができれば、意味の違う行動をそれぞれ抽出することができる。

本論文では Sequential Pattern Mining におけるアイテム間の順序に加えて、アイテム間の距離（時間）を考慮してマイニングを行う方法を提案する。提案手法ではアイテム間の順序だけで無く、距離（時間）を考慮してマイニングを行うことにより、直後に起こるトランザクションとしばらくして起こるトランザクションとの区別を可能にすることができた。

目次

第１章　はじめに

近年の技術の進歩により、記憶装置の容量が飛躍的に増大した。それによって、様々な場面で多くの情報を蓄積することが可能になった。しかし、蓄積される情報量が増大すればするほど、人間が直接データを扱うことが困難になる。情報を大量に集めることができても有効に活用できなければ無意味となってしまう。そこで、大量のデータの中から有用な情報を取り出す技術としてデータマイニング技術が注目されている。

データマイニングにおける重要な技術の一つとして頻出パターン抽出が挙げられる。頻出パターン抽出（Frequent Pattern Mining）とは、データベースの中からユーザが与えた最小サポート値以上の頻度で出現するパターン（アイテムの組合せ）を見つける技術である。Frequent Pattern Mining では速度の向上のためのアルゴリズムや最小サポートにこだわらないアルゴリズムが数多く提案された。

しかし、現実の世界では、顧客の購買行動、自然災害（地震など）、Web ページのクリックの流れ、株取引などアイテムの出現順序が重要となってくる場面が多く存在するが、Frequent Pattern Mining では順序を考慮せずにマイニングを行っている。そこで、より現実に即したマイニングを行うために、Sequential Pattern Mining が１９９５年に Agrawal と Srikant によって提案された[1]。Sequential Pattern Mining ではアイテム間の順序を考えてマイニングを行うことにより、上で述べたような事例において Frequent Pattern Mining より有用な情報を得ることが可能になった。Sequential Pattern Mining のアルゴリズムは現在までにいくつか提案されているが、有名なものは GSP[2]、PrefixSpan[5]、SPADE[3]、SPAM[4]が挙げられる。

しかし、Sequential Pattern Mining ではアイテム間の順序は考慮するが、アイテム間の距離（時間）は考慮されていない。そのため、抽出されたアイテムセット中の任意の２つのアイテム間に，どれだけの時間間隔があるのかを区別することが出来ない．例えば、商品 A を購入したあと、１日後に商品 B を購入する人と１年後に商品 B を購入する人がいるとする。この場合、従来の Sequential Pattern Mining ではこの２人の行動は同じものとして同じパターンで扱われてしまう。しかし、２人の行動の持つ意味はそれぞれ違っている。そこで、本論文では、従来の Sequential Pattern Mining において、アイテム間の距離が考慮されていない問題を解決することを目的として、従来の Sequential Pattern Mining では別々のトランザクションとして扱われていたものを、ユーザが定義した一定の距離（時間）内に起こったトランザクションを同時に起こったものとみなしてマイニングを行う手法を提案する。

本論文では以下の形をとる。第２章で Sequential Pattern Mining について述べる。第３章で関連研究として、既存の Sequential Pattern Mining アルゴリズムについて述べる。第４章でアイテム間の距離（時間）を考慮した Sequential Pattern Mining の提案をする。第５章で提案手法の評価と考察をし、最後に第６章でまとめを述べる。

第２章　Sequential Pattern Mining とは

本章では、Sequential Pattern Mining とは何かについて説明する。具体的には、まず知識抽出におけるデータマイニングについて述べる。次に、そして最後に、Sequential Pattern Mining の問題定義について述べる。

２.１　データマイニングとは

２.１.１　KDD プロセス[7]

大量のデータから有用な情報を取り出すプロセスである、KDD(Knowledge Discovery and DataMining)プロセスは、図２.１のような５段階で構成されている。以下に、図 2.1 における 5 段階のプロセスについて説明する。

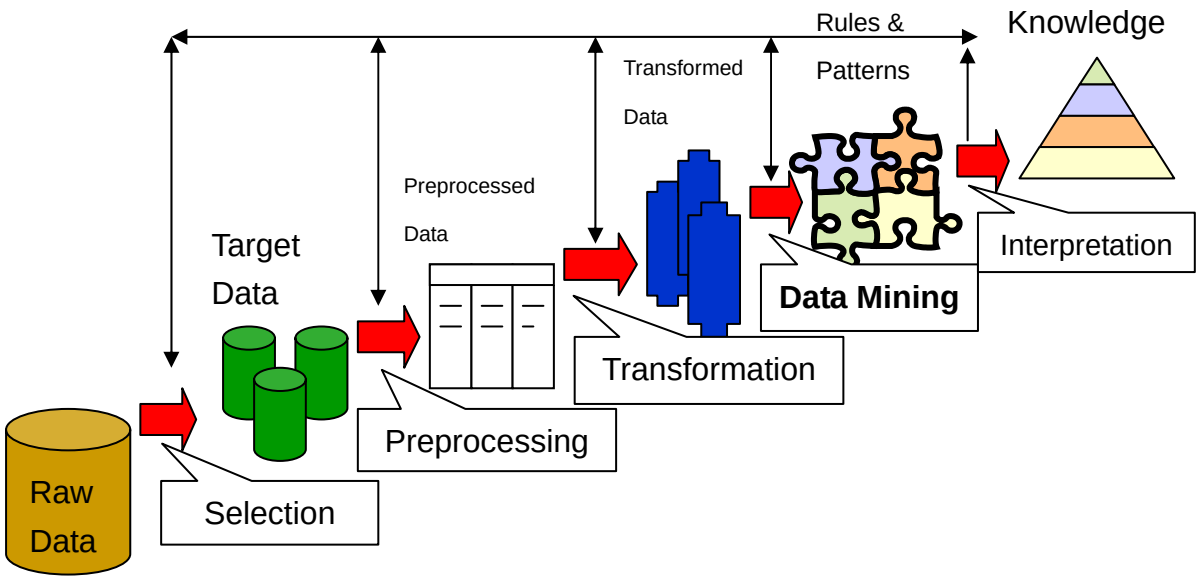


図２.１　KDD プロセス

（１）Selection（データ選択）

大規模なデータベースの中から、どのデータに対して、知識発見をするかを選択することを Selection という。

（２）Preprocessing（前処理）

データの中の欠損値に対して、ダミーのデータを与えたり、欠損値が存在するレコードは消去するなど、マイニングの前にデータを完全な状態にしておくことを Preprocessing という。

（３）Transformation(データ変換)

マイニングを実行する前にデータベースフォーマットからマイニングしやすいフォーマッ

トに変換することを Transformation という。

（4）Mining（データマイニング）

完全性の保たれているデータに対してルールやパターンを抽出することを Mining という。データマイニング手法には、相関ルール抽出、クラスタリング、決定木などがある。

（5）Interpretation/Evaluation（解釈・分析）

マイニングによって得られたルールやパターンを解釈、分析することを Interpretation/Evaluation という。

2.1.2 相関ルール抽出問題

KDD プロセスにおける Mining において、相関ルール抽出問題というのは重要な役割を担っている。相関ルール抽出問題とは何かを説明する前に、相関ルール抽出における定義、相関ルールの例を示す。

定義

$I=\{i_1,i_2,\cdots,i_n\}$をアイテム集合とする。データベース $D=\{t_1,t_2,\cdots,t_n\}$をトランザクションの集合とする。任意のトランザクション t はアイテムの集合で構成されている$(t_i \subseteq I(1 \leqq i \leqq n))$。また、各トランザクションにはユニークな識別子 TID(transaction id)が付けられているとする。

相関ルールとは、$X \subseteq I$、$Y \subseteq I$、$X \cap Y = \phi$ であるような任意のアイテム集合 X、Y を使って作られる $X \Rightarrow Y$ という表現のことである。

相関ルールは２つのパラメータ、確信度 conf(X⇒Y)と、サポート sup(X⇒Y) と呼ばれる２つのパラメータを持つ。確信度は、データベース D 中の X を含むトランザクションにおける、Y を含むトランザクションの割合である。サポートはデータベース D における、$X \cup Y$ を含む全トランザクションに対する割合である。

相関ルールの例

表２.１に、スーパーマーケットの購買データの例を示す。以下に、相関ルールについて、表２.１を元に説明していく。

表２.１　スーパーマーケットの購買データ D

TID	アイテム
0001	A,C,D
0002	B,C,E
0003	A,B,C,E
0004	B,E

表２.１のデータベースにおいて、各行がトランザクションを表す。つまり、表２.１において、TID(TransactionID)0001 番の顧客は商品 A,C,D を購入し、TID0002 番の顧客は商品 B,C,E

を購入したことになる。X⇒Y という相関ルールの確信度が c%で、そのサポートが s%だとする。この場合、商品集合 X を購入したトランザクションのうち c%が商品の集合 Y も同時に購入していて、X∪Y を購入していた顧客のトランザクション全体に対する割合は s%だったということである。

相関ルール抽出問題

相関ルール抽出問題では、データベース D、確信度の最小値 *min_conf*、そしてサポートの最小値 *min_sup* が与えられて、これらを満足する相関ルールを抽出する。相関ルール抽出は以下の２段階の処理に分けられる。

① ユーザが与えた *min_sup* を満たすアイテム集合（頻出アイテム集合という）を全て抽出する。

② 得られた頻出アイテム集合の中から *min_conf* を満たす相関ルールを得る。

②は①で求めた頻出アイテム集合を用いて相関ルールを得る処理であり、負荷が小さいので比較的早く算出することができる。しかし、①は大規模なデータベースに対し繰り返しスキャンを行い、それぞれのアイテム集合のサポートを計算するため、負荷が大きくなり算出に時間がかかる。つまり、相関ルール抽出アルゴリズムというのは、①の処理をいかに効率よく計算するかが主要な課題となっており、相関ルール抽出問題は頻出パターン抽出（Freqent Pattern Mining）問題に置き換えることができる。

２.１.３ Frequent Pattern Mining から Sequential Pattern Mining へ

Frequent Pattern Mining が対象とするデータにおいて、アイテム間の順序という概念は存在していない。例えば、表２.１の購入データにおいて、TID0001 の顧客は ACD を購入しているが、ACD は同時に購入されている。つまり、ACD でも CDA でも構わないのである。同様に、抽出されるパターンもアイテム間の順序は考えられていない。

しかし、現実の世界ではアイテム間の順序が多くの場面で重要となってくる。そこで、Frequent Pattern Mining においてアイテム間の順序が考慮されていないという問題を解決するために、Sequential Pattern Minig が考案された。Sequential Pattern Mining は、顧客購買行動、医療、自然災害（地震など)、科学実験、株取引、電話をかけるパターン、Web ページ閲覧の流れ、DNA 配列、遺伝子構造などの分野に応用することができる。

２.２ Sequential Pattern Mining

２.２.１ シーケンス

Sequential Pattern Mining においてのパターンは、シーケンスである。ひとつのシーケンスは、”<”で開始し、”>”で終了する。シーケンスでは、”< >”の間にアイテムを時系列順に配置する。なお、同時期に発生したアイテムは、”( )”でくくることによって表現される。シーケンスの例を以下に示す。

シーケンス：＜(ef) (ab) (df) c b＞

□を１つのトランザクションとして考える。( )の中は同時に行われているので順序は問わない。しかし、□同士は時系列で並んでいるので入れ替えはできない。

サブシーケンス：＜a (bc) d c＞は＜*a* (a*bc*) (ac) *d* (*c*f)＞のサブシーケンスである。

シーケンス S が S'のサブシーケンスとは、S のアイテムが S'内において S と同じ順番で存在しているということを意味する。上の例では、＜a (bc) d c＞は＜a (abc) (ac) d (cf)＞内において、斜体字で示すように、a、(bc)、d、c の順番で存在しているので＜a (bc) d c＞は＜a (abc) (ac) d (cf)＞のサブシーケンスである。

２.２.２　問題定義

表２.２のようなデータベース *D* が与えられている。それぞれのトランザクションには顧客 ID、トランザクションタイム、アイテム集合が含まれている。

シーケンスはアイテム集合の整列されたリストである。アイテム集合は空ではなく通常、完全に連続したものであると考える。アイテム集合ｉは（$i_1 i_2 \cdots i_m$）と表される。
ここでi_j（1≦j≦m）は１つのアイテムである。さらにシーケンスｓは＜$s_1 s_2 \cdots s_n$＞と表される。ここで、s_k(1≦k≦n)は１つのアイテム集合である。

表２.２　元のデータベース

Transaction Time	Customer ID	Items Bought
June 10 '93	2	10,20
June 12 '93	5	90
June 15 '93	2	30
June 20 '93	2	40,60,70
June 25 '93	4	30
June 25 '93	3	30,50,70
June 25 '93	1	30
June 30 '93	1	90
June 30 '93	4	40,70
July 25 '93	4	90

２.２.３　マイニングの例

表２.３の顧客の購買データベースの例を用いて、Sequential Pattern Mining について説明する。この例において、min_sup=2 とする。

２つのシーケンス＜(30)(90)＞と＜(30)(4070)(90)＞は、アイテム３０を買い、その後にアイテム９０を買っている。つまり＜(30)(90)＞は最小サポート値を満たしている。

同様に＜(10 20)(30)(40 60 70)＞と＜(30)(40 70)(90)＞は、アイテム３０を買った後、アイ

テム４０と７０を同時に購入しているので、＜(30)(40 70)＞は最小サポート値を満たしている。マイニングの結果を表２.４に示す。

表２.３　Costomer-Sequence バージョンのデータベース

Costomer ID	Customer Sequence
1	＜(30)(90)＞
2	＜(10 20)(30)(40 60 70)＞
3	＜(30 50 70)＞
4	＜(30)(40 70)(90)＞
5	＜(90)＞

表２.４　マイニングの結果

Sequential Patterns with support >40%
＜(30)(90)＞
＜(30)(40 70)＞

第３章　関連研究

本章では、既存の Sequential Pattern Mining のアルゴリズムを示す。まず、Sequential Pattern Mining の歴史について説明する。次に、Sequential Pattern Mining の代表的なアルゴリズムである GSP[3]、SPADE[4]、SPAM[5]、PrefixSpan[6]について説明する。最後に４つのアルゴリズムをそれぞれ比較する。

３.１　Sequential Pattern Mining の歴史

Sequential Pattern Mining はアイテム間の順序を考慮したデータマイニングで、1995 年に Agrawal と Srikant によって初めて提案された[2]。同時に、Sequential Pattern Mining のアルゴリズムとして AprioriAll という Apriori（Apriori については次節で説明する）的なアルゴリズムが提案された。さらに 96 年に Apriori ベースのアルゴリズムである GSP（**G**eneralized **S**equential **P**attern mining）[3]が Agrawal と Srikant によって提案された。GSP は Apriori の「**あるシーケンス S のサブシーケンス S'が非頻出ならばそのシーケンスは非頻出である**」という考えのもとに候補シーケンスを生成し、頻出シーケンスを抽出する手法である。しかし、GSP では候補シーケンスが膨大な数になってしまうという欠点がある。

GSP の候補シーケンスが膨大な数になるという欠点を解決するために、2000 年に Zaki が SPADE(**S**equential **PA**ttern **D**iscovery using **E**quivalence classes)[4]を提案した。SPADE は Lattice という概念を用い候補シーケンスをグループに分割する。また、ID-List というデータ構造を用いることによってサポート値をカウントする際のコストを削減することができる。

さらに、SPADE を改良したアルゴリズムとして、2001 年に Ayres らが、SPAM(**S**equential **PA**ttern **M**ining)[5]を提案した。SPAM では、ID-List をビットマップを用いて表現することによりさらなる高速化を実現した。

一方で、候補シーケンスを生成せずにデータベースを射影することによって頻出シーケンスを抽出する手法が 2001 年に Pei らによって提案された PrefixSpan[6]である。[8]によれば、PrefixSpan が 2004 年時点では最も高速なアルゴリズムである。

３.２　GSP[3]

GSP(**G**eneralized **S**equentilal **P**attern mining)は 1996 年に Agrawal,Srkant によって提案されたアルゴリズムである。GSP は Sequential Pattern Mining において最初に提案された有名なアルゴリズムである。GSP は Frequent Pattern Mining のアルゴリズムである

Apriori[1]の考え方をベースにしている。本節では、まず GSP のベースとなっている概念である Apriori について説明する。次に、GSP における重要な動作である候補シーケンス生成について説明した後、GSP のアルゴリズムの流れを説明する。そして、最後に GSP の欠点を説明する。

３.２.１　Apriori の理論の適用

Apriori は 1994 年に Agrawal らによって提案された Frequent Pattern Mining のアルゴリズムである。Apriori は「**長さ k の頻出でないパターンを含む長さ k+1 のパターンは頻出でない**」という理論の元で、ボトムアップに頻出パターンを抽出するアルゴリズムである。

Sequential Pattern Mining においても Apriori と同様の理論が適用でき、「あるシーケンス S が頻出シーケンスならば S のスーパーシーケンスはすべて頻出ではない」。例えば、シーケンス＜hb＞が頻出でなければ、＜hb＞のスーパーシーケンス（内部に＜hb＞を含むシーケンス）である、＜hab＞と＜(ah)b＞も頻出でない。この理論を元にして、GSP はマイニングを行っている。

３.２.２ 候補シーケンス生成

k 個のアイテムを持つ（長さが k である）シーケンスを k-sequence と呼ぶ。L_k をすべての頻出 k-sequence の集合とし、C_k を候補 k-sequence の集合とする。

候補シーケンス生成とは、与えられた頻出(k-1)-sequence の集合 L_{k-1} が与えられたとして、すべての候補 k-sequence の集合のスーパーセットを生成することを指す。そこで、まず連続サブシーケンスの定義を述べる。

シーケンス s=＜s_1 s_2...s_n＞と、サブシーケンス c が与えられているとする。以下のいずれかの条件が成り立つとき、c は s の連続サブシーケンスである。

１、s_1 または s_n からアイテムを削除することで c が s から導出される。

２、少なくとも２つのアイテムを持つ要素 s_i からアイテムを削除することで、c から s が導出される。

３、c が c'の連続サブシーケンスであり、c'が s の連続サブシーケンスである。

例えば、シーケンス s ＜(1,2) (3,4) (5) (6)＞について考える。ここで、＜(2) (3,4) (5)＞＜(1,2) (3) (5) (6)＞ ＜(3) (5)＞は s の連続サブシーケンスである。
しかし、＜(1,2) (3,4)(6)＞＜(1) (5) (6)＞は s の連続サブシーケンスではない。

候補生成は Join Phase と Prune Phase の２つのフェーズに分かれている。

(１) Join Phase

L_{k-1} に含まれるシーケンス s_a と s_b を結合することによって候補シーケンスを生成する。ここで、$s_a =< sa_1, sa_2, \cdots, sa_m >, s_b =< sb_1, sb_2, \cdots, sb_n >$ とする。$sa_2 = sb_1, sa_3 = sb_2, \cdots, sa_m = sb_{n-1}$ が成立する場合、候補シーケンスは s_a と s_b を結合し、$< sa_1, sa_2, \cdots, sa_m, sb_n >$ となる．追加されるアイテムは、それが s_b において分離した要素

（最後のトランザクションが(ab)のときに b だけ取り出すと(_b)になる。これを分離した要素という）は、分離した要素にする。それ以外のときは s_a の最後の要素の一部とする。L_1 と L_1 の結合するときは、s_b のアイテムをアイテム集合の一部として、また分離した要素としてとの両方で追加する必要がある。なぜなら＜(x) (y)＞と＜(x,y)＞は最初のアイテムを削除することで同一のシーケンス＜(y)＞を生成するためである。。

(2) Prune Phase

サポート値が最小サポートより小さい連続(k-1)-subsequence を持つ候補シーケンスを削除する。

以下で長さ３の頻出シーケンスが与えられたときに、長さ４の候補シーケンスを生成する例を説明する。

表３.１　候補生成の例

Frequent 3-Sequences	Candidate 4-Sequence	
	After join	After pruning
＜(1,2) (3)＞ ＜(1,2) (4)＞ ＜(1) (3,4)＞ ＜(1,3) (5)＞ ＜(2) (3,4)＞ ＜(2) (3) (5)＞	＜(1,2) (3,4)＞ ＜(1,2)(3)(5)＞	＜(1,2) (3,4)＞

Join Phase では、＜(1,2) (3)＞と＜(2) (3,4)＞の結合によって＜(1,2) (3,4)＞が生成される。同様に＜(1,2) (3)＞と＜(2) (3) (5)＞の結合によって＜(1,2) (3) (5)＞が生成される。

次に、Prune Phase では＜(1,2) (3) (5)＞がサブシーケンスの一つである＜(1) (3) (5)＞が頻出でないので、削除される。よって長さ４の候補シーケンスは＜(1,2) (3,4)＞となる。

３.２.２　GSP アルゴリズムの流れ

GSP アルゴリズムの流れを以下に示す。

１、長さ k の頻出シーケンスから長さ k+1 の候補シーケンスを生成する
２、候補シーケンスのサポート値をカウントし、最小サポート値以上のシーケンスのみが長さ k+1 の頻出シーケンスとして抽出される。
３、候補シーケンスを生成できなくなる、または頻出シーケンスが抽出できなくなるまで１，２の動作を繰り返す

表３.２のシーケンスデータベースが与えられたとき、上の各ステップでどのような動きになるのか説明する。なお、min_sup=2 とする。

表３.２　シーケンスデータベース

SID	Sequence
10	＜(bd) c b (ac)＞
20	＜(bf) (ce) b (fg)＞
30	＜(ah) (bf) a b f＞
40	＜(be) (ce) d＞
50	＜a (bd) b c b (ade)＞

表３.３　長さ１の頻出シーケンス

Cand	Sup
＜a＞	3
＜b＞	5
＜c＞	4
＜d＞	3
＜e＞	3
＜f＞	2
~~＜g＞~~	1
~~＜h＞~~	1

表３.４　長さ２の候補シーケンス

	＜a＞	＜b＞	＜c＞	＜d＞	＜e＞	＜f＞
＜a＞	＜aa＞	＜ab＞	＜ac＞	＜ad＞	＜ae＞	＜af＞
＜b＞	＜ba＞	＜bb＞	＜bc＞	＜bd＞	＜be＞	＜bf＞
＜c＞	＜ca＞	＜cb＞	＜cc＞	＜cd＞	＜ce＞	＜cf＞
＜d＞	＜da＞	＜db＞	＜dc＞	＜dd＞	＜de＞	＜df＞
＜e＞	＜ea＞	＜eb＞	＜ec＞	＜ed＞	＜ee＞	＜ef＞
＜f＞	＜fa＞	＜fb＞	＜fc＞	＜fd＞	＜fe＞	＜ff＞
	＜a＞	＜b＞	＜c＞	＜d＞	＜e＞	＜f＞
＜a＞	＜(aa)＞	＜(ab)＞	＜(ac)＞	＜(ad)＞	＜(ae)＞	＜(af)＞
＜b＞		＜(bb)＞	＜(bc)＞	＜(bd)＞	＜(be)＞	＜(bf)＞
＜c＞			＜(cc)＞	＜(cd)＞	＜(ce)＞	＜(cf)＞
＜d＞				＜(dd)＞	＜(de)＞	＜(df)＞
＜e＞					＜(ee)＞	＜(ef)＞
＜f＞						＜(ff)＞

表３.５　長さ２の頻出シーケンス

	＜a＞	＜b＞	＜c＞	＜d＞	＜e＞	＜f＞
＜a＞	＜aa＞:2	＜ab＞:2	＜ac＞:2	~~＜ad＞:1~~	~~＜ae＞:1~~	~~＜af＞:1~~
＜b＞	＜ba＞:2	＜bb＞:4	＜bc＞:4	~~＜bd＞:1~~	＜be＞:3	＜bf＞:2
＜c＞	＜ca＞:2	＜cb＞:2	~~＜cc＞:1~~	＜cd＞:2	~~＜ce＞:1~~	~~＜cf＞:1~~
＜d＞	＜da＞:2	＜db＞:2	＜dc＞:2	~~＜dd＞:1~~	~~＜de＞:1~~	~~＜df＞:0~~
＜e＞	~~＜ea＞:0~~	~~＜eb＞:1~~	~~＜ec＞:1~~	~~＜ed＞:1~~	~~＜ee＞:1~~	~~＜ef＞:1~~
＜f＞	~~＜fa＞:1~~	＜fb＞:2	~~＜fc＞:1~~	~~＜fd＞:0~~	~~＜fe＞:1~~	＜ff＞:2
	＜a＞	＜b＞	＜c＞	＜d＞	＜e＞	＜f＞
＜a＞	~~＜(aa)＞:0~~	~~＜(ab)＞:0~~	~~＜(ac)＞:1~~	~~＜(ad)＞:1~~	~~＜(ae)＞:1~~	~~＜(af)＞:0~~
＜b＞		~~＜(bb)＞:0~~	~~＜(bc)＞:0~~	＜(bd)＞:2	~~＜(be)＞:1~~	＜(bf)＞:2
＜c＞			~~＜(cc)＞:0~~	~~＜(cd)＞:0~~	＜(ce)＞:2	~~＜(cf)＞:0~~
＜d＞				~~＜(dd)＞:0~~	~~＜(de)＞:1~~	~~＜(df)＞:0~~
＜e＞					~~＜(ee)＞:0~~	~~＜(ef)＞:0~~
＜f＞						~~＜(ff)＞:0~~

表３.６　長さ３の候補シーケンス

candidate 3-sequence						
＜aaa＞	＜aca＞	＜bbc＞	＜bfb＞	＜cda＞	＜dbc＞	＜ffb＞
＜aab＞	＜acb＞	＜bbe＞	＜bff＞	＜cdb＞	＜dca＞	＜fff＞
＜aac＞	＜baa＞	＜bbf＞	＜caa＞	＜daa＞	＜dcb＞	＜(bd)a＞
＜aba＞	＜bab＞	＜b(bf)＞	＜cab＞	＜dab＞	＜fbb＞	＜(bd)b＞
＜abb＞	＜bac＞	＜bca＞	＜cba＞	＜dac＞	＜fbf＞	＜(bd)c＞
＜abc＞	＜bba＞	＜bcb＞	＜cbb＞	＜dba＞	＜f(bd)＞	＜(bf)b＞
＜abe＞	＜bbb＞	＜b(ce)＞	＜c(bd)＞	＜dbb＞	＜f(bf)＞	＜(bf)f＞

表３.７　長さ３の頻出シーケンス

frequent 3-sequence						
~~＜aaa＞:0~~	~~＜aca＞:1~~	＜bbc＞:2	~~＜bfb＞:0~~	~~＜cda＞:0~~	＜dbc＞:2	~~＜ffb＞:0~~
~~＜aab＞:1~~	~~＜acb＞:1~~	~~＜bbe＞:1~~	~~＜bff＞:0~~	~~＜cdb＞:0~~	＜dca＞:2	~~＜fff＞:0~~
~~＜aac＞:0~~	~~＜baa＞:0~~	＜bbf＞:2	~~＜caa＞:0~~	~~＜daa＞:0~~	＜dcb＞:2	＜(bd)a＞:2
＜aba＞:2	~~＜bab＞:1~~	~~＜b(bf)＞:0~~	~~＜cab＞:0~~	~~＜dab＞:0~~	~~＜fbb＞:0~~	＜(bd)b＞:2
＜abb＞:2	~~＜bac＞:0~~	＜bca＞:2	＜cba＞:2	~~＜dac＞:0~~	＜fbf＞:2	＜(bd)c＞:2
~~＜abc＞:1~~	＜bba＞:2	＜bcb＞:2	~~＜cbb＞:0~~	＜dba＞:2	~~＜f(bd)＞:0~~	＜(bf)b＞:2
~~＜abe＞:1~~	~~＜bbb＞:1~~	＜b(ce)＞:2	~~＜c(bd)＞:0~~	~~＜dbb＞:1~~	~~＜f(bf)＞:0~~	＜(bf)f＞:2

表３.８　長さ４の候補シーケンス

candidate 4-sequence		
＜abba＞	＜dbca＞	＜(bd)bc＞
＜bbca＞	＜dcba＞	＜(bd)ca＞
＜bcba＞	＜(bd)ba＞	＜(bd)cb＞

表３.９　長さ４の頻出シーケンス

frequent 4-sequence		
~~＜abba＞:1~~	~~＜dbca＞:1~~	＜(bd)bc＞:2
~~＜bbca＞:1~~	＜dcba＞:2	＜(bd)ca＞:2
＜bcba＞:2	＜(bd)ba＞:2	＜(bd)cb＞:2

表３.１０　長さ５の候補シーケンス

candidate 5-sequence
＜(bd)cba＞

表３.１１　長さ５の頻出シーケンス

frequent 5-sequence
＜(bd)cba＞:2

まず表３.２のデータベースをスキャンし、長さ１の頻出シーケンスを抽出する。抽出した結果が表３.３である。次に抽出した長さ１の頻出シーケンスを元に、表３.４のような長さ２の候補シーケンスを生成する。そして、長さ２の候補シーケンスのサポート値をカウントし、長さ２の頻出シーケンスを抽出する（表３.５）。同様に長さ２の頻出シーケンスを元に、長さ３の候補シーケンスを生成（表３.６）し、頻出シーケンスを抽出する（表３.７）。続いて長さ３の頻出シーケンスを元に、長さ４の候補シーケンスを生成し（表３.８）、長さ４の頻出シーケンスを抽出する（表３.９）。引き続き長さ５の候補シーケンスを生成し（表３.１０）、長さ５の頻出シーケンスが抽出される（表３.１１）。長さ５の頻出シーケンスから長さ６の候補シーケンスは生成できないので、ここでアルゴリズムは終了する。

３.２.３　GSP の欠点

GSP の欠点は以下の２つである。

・生成される候補シーケンスが莫大な数になる

GSP における候補シーケンスはシーケンスの中で起こり得るすべてのアイテムの組み合わせを含んでいる。そのため、候補シーケンスが莫大な数になってしまう可能性がある。例えば、長さ 1 の頻出アイテム(＜a_1＞,＜a_2＞…＜a_{1000}＞)が 1000 個あるとき、長さ 2 の候補シーケンス(＜a_1a_1＞,＜a_1a_2＞…＜a_1a_{1000}＞,＜a_2a_1＞…＜$a_{1000}a_{1000}$＞と＜(a_1a_2)＞＜(a_1a_3)＞…＜$(a_{999}a_{1000})$＞)は 1000×1000＋1000×999÷２＝1499500 個生成されることになる。

また、抽出するシーケンスの長さが長くなればなるほど、必要となる候補シーケンスが指数的に増大する。例えば、min_sup=1（すべてのパターンが頻出のとき）が与えられたとき、長さ 100 のシーケンスをマイニングした場合、長さ 1 の候補シーケンスは 100 個、長さ 2 の候補シーケンスは 100×100＋100×99÷２＝14950 個、同様に長さ３の候補シーケンスは 161700 個……、合計では 2^{100}－１個となり、記憶容量が多く必要となってしまう。

・データベースのスキャン回数が多くなる

GSP ではそれぞれの長さごとに候補シーケンスを生成するためにデータベースをスキャンしなければならない。そのため、データベースのスキャン回数が多くなってしまう。例えば、＜(abc) (abc) (abc) (abc) (abc)＞というパターンを見つけるためには 15 回もデータベースをスキャンしなければならないので、記憶容量が多く必要になってしまう。

３.３　SPADE[5]

SPADE(**S**equential **PA**ttern **D**iscovery using **E**quivalence classes)[5]は 2000 年に Zaki に提案されたアルゴリズムである。SPADE は *lattice*（格子）という概念を用いて、候補シーケンスをアイテムごとにグループに分割し、それぞれのグループがメインメモリに完全に格納されることによって高速化を図っている。また、*ID-List* というデータフォーマットを用いることによってサポート値をカウントするコストを削減している。本節ではまず、*ID-List* と *Lattice* という概念について説明した後、具体的な SPADE アルゴリズムの流れを説明する。

３.３.１　*ID-List*

ID-List とは SPADE で用いられているデータフォーマットである。*ID-List* はシーケンスごとに分けられており、それぞれが SID と EID という２つの要素を持っている。SID はシーケンシャル ID であり、EID はトランザクションが行われた時間を格納するイベント ID である。表３.１２のデータベースをもとに作成した長さ１の頻出アイテムの ID-List が表３.１３である。例えば、アイテム D に関するトランザクションが行われたのは SID１で時間１０と２５と SID4 で時間１０であるということを表している。

表３.１２　元のデータベース

Sequential ID	Time	Items
1	10	C D
1	15	A B C
1	20	A B F
1	25	A C D F
2	15	A B F
2	20	E
3	10	A B F
4	10	D G H
4	20	B F
4	25	A G H

表３.１３　表３.１を元に作成した長さ１の頻出シーケンスの ID-List

＜A＞		＜B＞		＜D＞		＜F＞	
SID	EID	SID	EID	SID	EID	SID	EID
1	15	1	15	1	10	1	20
1	20	1	20	1	25	1	25
1	25	2	15	4	10	2	15
2	15	3	10			3	10
3	10	4	20			4	20
4	25						

３.３.２　Lattice

SPADE では Lattice という概念に基づいてシーケンスを列挙している。Lattice とは先ほど述べたように格子という意味で、候補シーケンスをアイテムごとにグループに分割し、それぞれのグループをメインメモリに完全に格納することにより高速化を実現している。表３.１４のデータベースを例に Latice の説明を行う。

表３.１４　シーケンスデータベース(min_sup=2)

Sequence ID	Sequence
1	＜(C D) (A B C) (A B F) (A C D F)＞
2	＜(A B F) E＞
3	＜(A B F)＞
4	＜(D G H) (B F) (A G H)＞

ここで、最小サポートを満たす長さ１のアイテム集合を F_1={A,B,D,F}とする。この 4 つのアイテムを元に最小サポートに関係なく、長さが２、３、４…のシーケンスを作り、それを Lattice で表現したものが図３.１である。どのようにして長さ２，３，４…のシーケンスを生成するかは後ほど説明する。図３.１の格子の底は“{}”になっているが、格子は無限なので上限は存在しない。

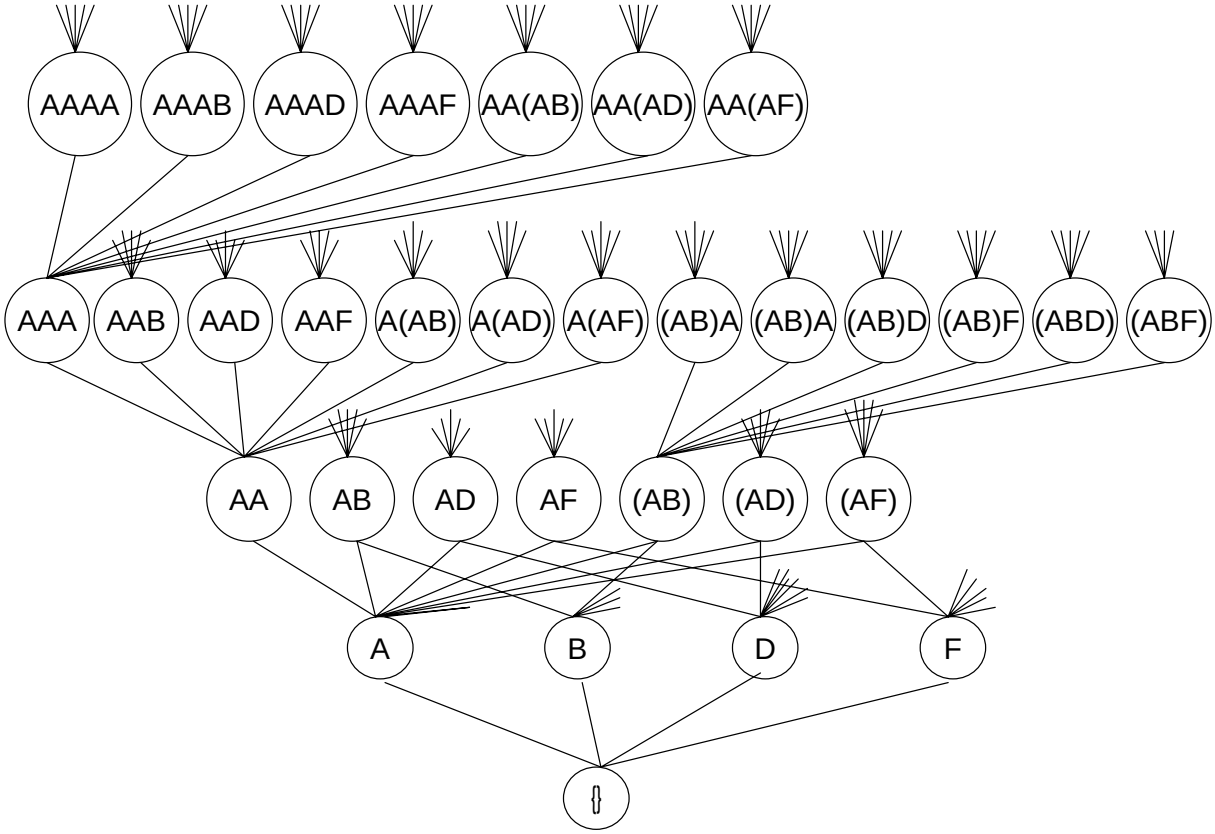


図３.１　長さ１の頻出アイテムをもとにシーケンスを格子状に表現（[5]より引用）

また､表３.１４の例において最小サポート考慮して頻出シーケンスのみで作成したLatticeが図３.２である。この図を元に、Lattice を用いた候補シーケンスのグループ分けをした探索について説明する。

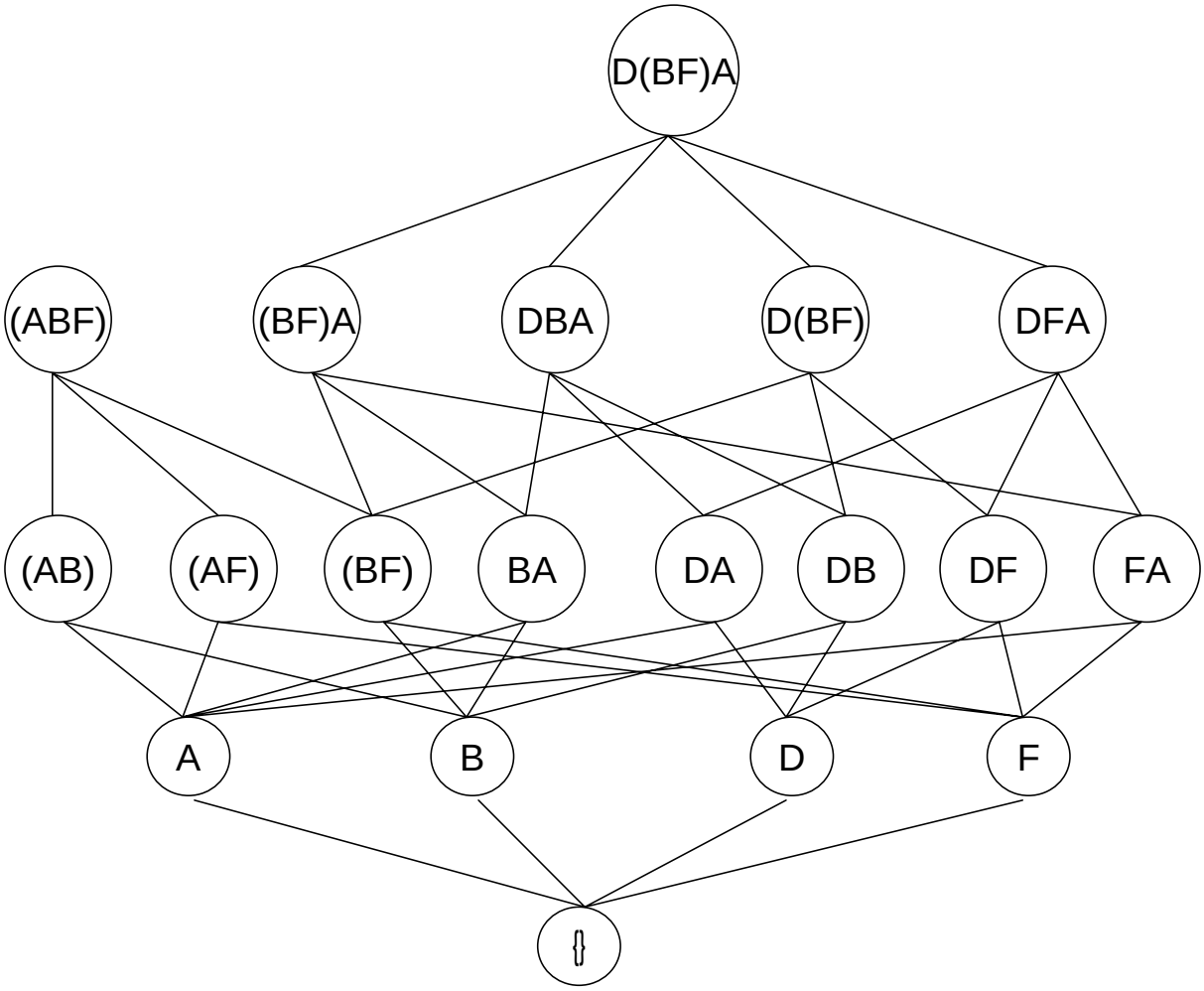


図３.２　極大の頻出シーケンス＜(ABF)＞と＜D (BF) A＞によって作成された格子([5]より引用)

SPADE では深さ優先探索でマイニングを行う。図３.３では探索の順序を 4 つに分割している。＜A＞から始まる頻出シーケンス（class[A]）の抽出、＜B＞から始まる頻出シーケンス（class[B]）の抽出、＜C＞から始まる頻出シーケンス（class[C]）の抽出、＜D＞から始まる頻出シーケンス（class[D]）の抽出の順でマイニングを行う。同様に、図３.4 は＜D＞から始まるシーケンスの抽出の順序をさらに＜DA＞で始まるもの（class[DA]）、＜DB＞で始まるもの（class[DB]）、＜DF＞で始まるもの（class[DF]）の３つに分割している。そして、図３.５では、＜D＞から始まる候補シーケンスの流れについてを表している。＜D＞とその他の長さ 1 の頻出アイテム＜A＞、＜B＞、＜F＞とを結合することによって長さ 2 の候補シーケンス＜DA＞、＜DB＞、＜DF＞が得られる。同様に探索していくと、＜DBA＞、＜D(BF)＞、＜D(BF)A＞、＜DFA＞の順に抽出される。

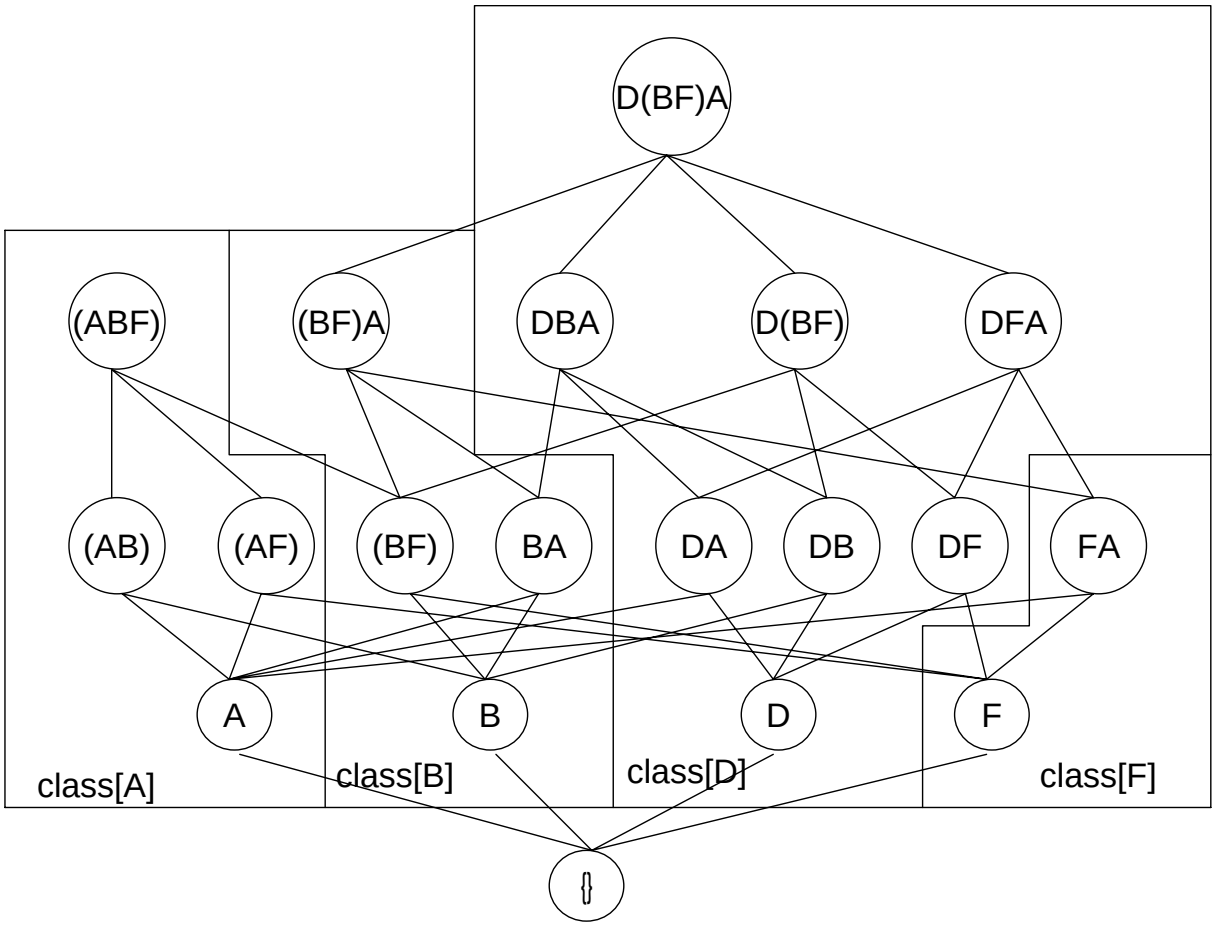


図３.３　Lattice によるグループ分け

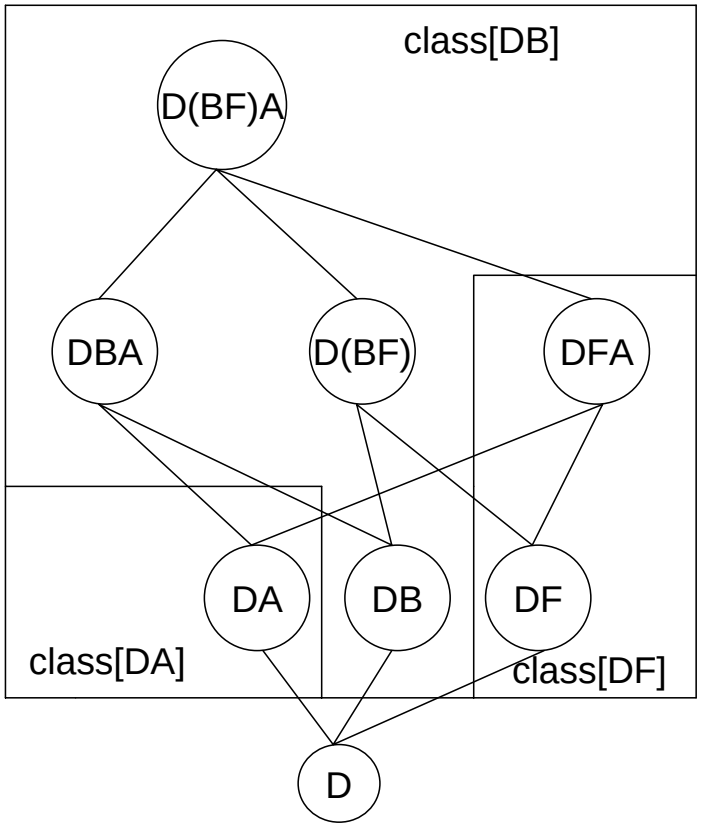


図３.４　クラス[D]におけるグループ分け

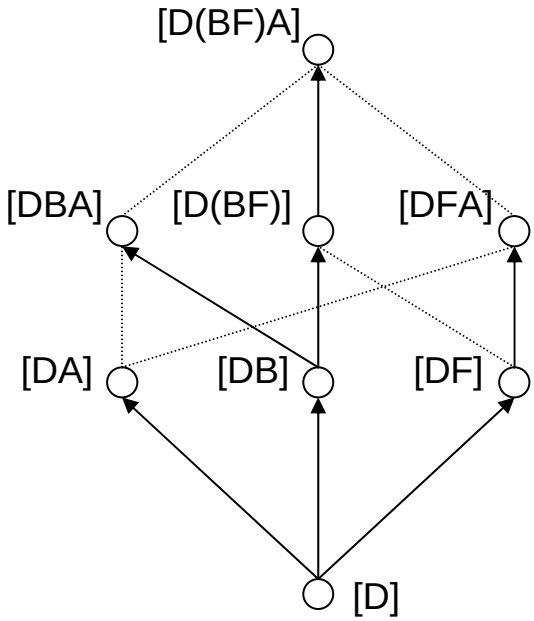


図３.５　＜D＞から始まる頻出シーケンスの抽出における探索の流れ

３.３.３　SPADE アルゴリズムの流れ

SPADE では表３.１３のような ID-List のデータベースを用いる。SPADE の主要な流れは以下のとおりである。

１、長さ１の頻出アイテムを抽出する
２、長さ２の頻出シーケンスを抽出する
３、深さ優先探索で長さ k（k ≧２）の頻出シーケンス同士を結合し、長さ k+1 の頻出シーケンスを抽出する

ID-List の結合

図３.６のシーケンス＜PA＞と＜PF＞の結合の例について考える。＜PA＞と＜PF＞との結合からは、＜PAF＞、＜PFA＞、＜P(AF)＞の 3 つのシーケンスを生成することができる。まずは、＜PAF＞を生成する場合の SID が１のトランザクションについて見てみると、PA では時間（EID）２０，３０，４０に起こっていて、PF は時間(EID)７０，８０に起こっている。ここで、＜PAF＞とは P、A、F の順でトランザクションが起こっているので、PF より前の時間(EID)に、PA が起こっていれば、結合が可能となる。よって、＜PAF＞の結合は、図３.６のようになる。また、＜P (AF)＞を生成する場合は、＜PA＞と＜PF＞が同じ時間(EID)のものを探すことになる。よって、＜PA＞、＜PF＞に共通に存在している SID が 8 で時間(EID)が３０，５０，８０のトランザクションが＜P (AF)＞の ID-List となる。

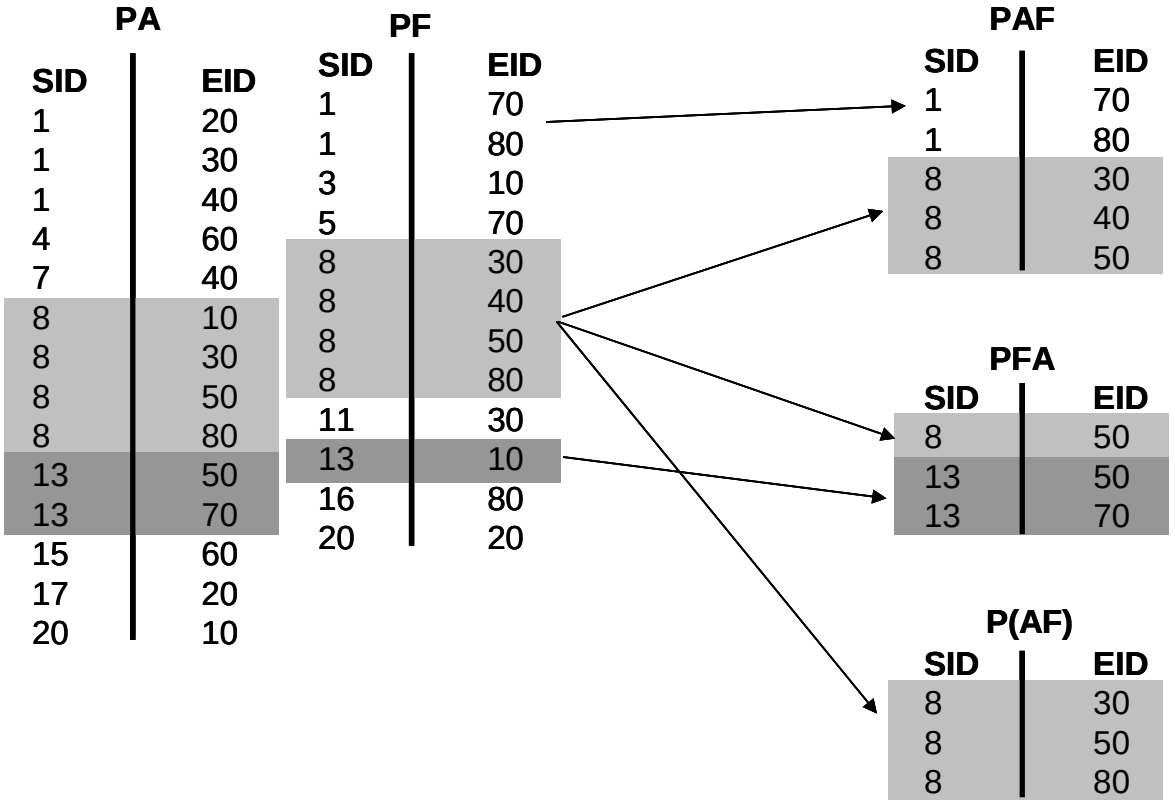


図３.６　ID-List の結合

表３.５のマイニングにおける、ID-List の結合を図３.７に示す。

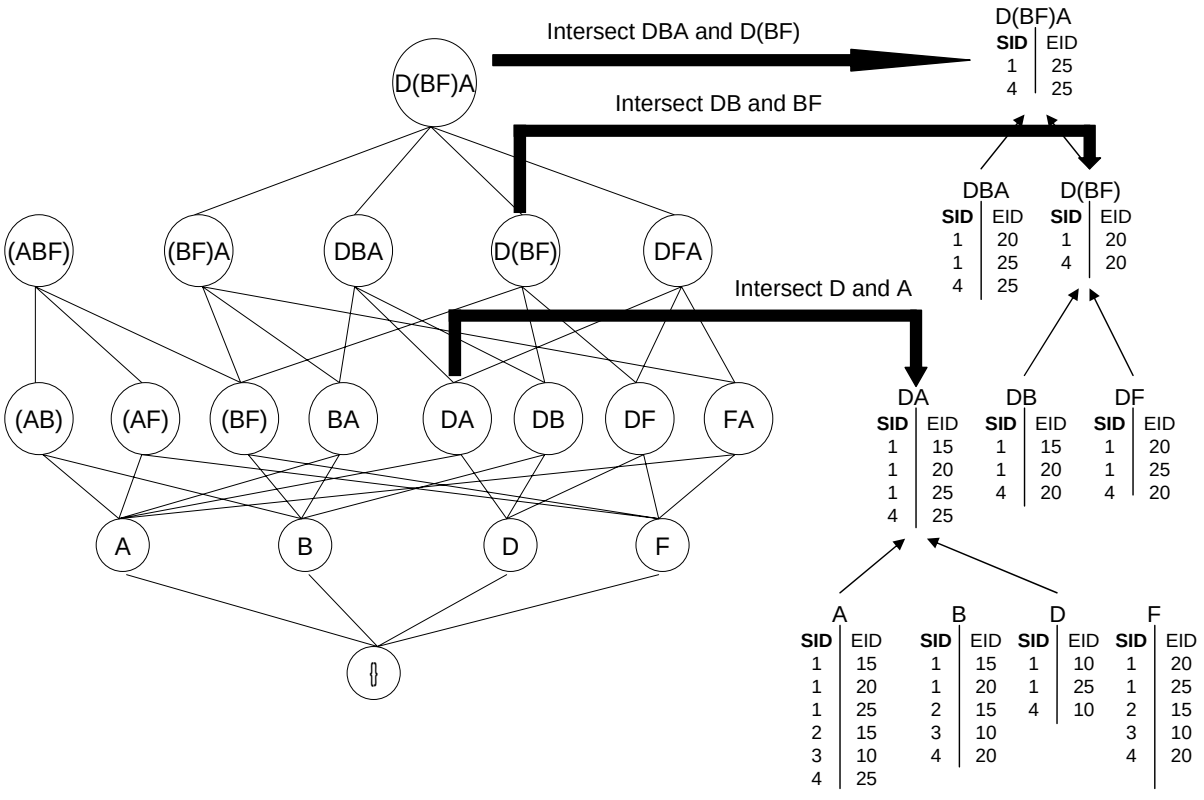


図３.７　表３.５のマイニングの流れにおける ID-List の結合

＜A＞と＜D＞を結合することによって＜DA＞（図中では DA）を生成している。同様に、＜DB＞（図中では DB）と＜DF＞（図中では DF）の結合から＜D (BF)＞（図中では D(BF)）、＜DBA＞（図中では DBA）と＜D (BF)＞（図中では D(BF)）の結合から＜D (BF) A＞（図中では D(BF)A）が生成される。

３.４　SPAM[6]

SPADE では、頻出シーケンスを結合するところが主要なコストになっている。そのため、何度も結合を繰り返すと、候補シーケンスが膨大な数になってしまい、マイニングの時間が長くかかってしまう。以上の問題を解決するために Ayres らによって 2001 年に提案された手法が SPAM(**S**equemtial **PA**ttern **M**ining)[6]である。

SPAM は SPADE と同様に Lattice の考え方を用いている。また、アルゴリズムの流れも SPADE と基本的には同じで、長さｋ－１の頻出シーケンスを深さ優先探索で結合して、長さｋの頻出パターンを抽出していく。SPADE との大きな違いは ID-List を vartical なビットマップで表現するところにある。以下で、ビットマップを用いたデータ構造について説明する。

３.４.１　ビットマップを用いたデータ構造

表３.１５のデータベースを例に説明する。表３.１５のデータベースをビットマップで表現したものが表３.１６である。

表３.１５　元のデータベース

CID	EID	Item
1	1	a,b,d
1	3	b,c,d
1	6	b,c,d
2	2	b
2	4	a,b,c
3	5	a,b
3	7	b,c,d

表３.１６　表３.１５のビットマップ表現

CID	EID	a	b	c	d
1	1	1	1	0	1
1	3	0	1	1	1
1	6	0	1	1	1
-	-	0	0	0	0
2	2	0	1	0	0
2	4	1	1	1	0
-	-	0	0	0	0
-	-	0	0	0	0
3	5	1	1	0	0
3	7	0	1	1	1
-	-	0	0	0	0
-	-	0	0	0	0

ビットマップ表現では、トランザクション内にアイテムが存在すれば１で、存在しなければ０で表現している。例えば、表３.の CID が１、EID が３のトランザクションでは、b,c,d が行われているので、a のところは０、b,c,d のところは１になっている。

３.４.２　ビットマップ表現におけるシーケンスの結合

SPAM におけるシーケンスの結合方法は２種類存在する。一つ目は別々の時間に起こったトランザクションの結合である。例えば、＜a＞と＜b＞（a と b が別の時間に起こっている）から＜ab＞を生成するといった結合を指す。二つ目は同じ時間に起こったトランザクションの結合である。例えば、＜ab＞と＜d＞（b と d が同じ時間に起こっている）から＜a(bd)＞を生成するといった結合を指す。この２種類のシーケンスの結合方法についてそれぞれ以下で説明する。

別々の時間に起こったトランザクション同士の結合

表３.１６のビットマップの＜a＞と＜b＞から＜ab＞を生成する例を考える。結合の流れを表したものが図３.８である。

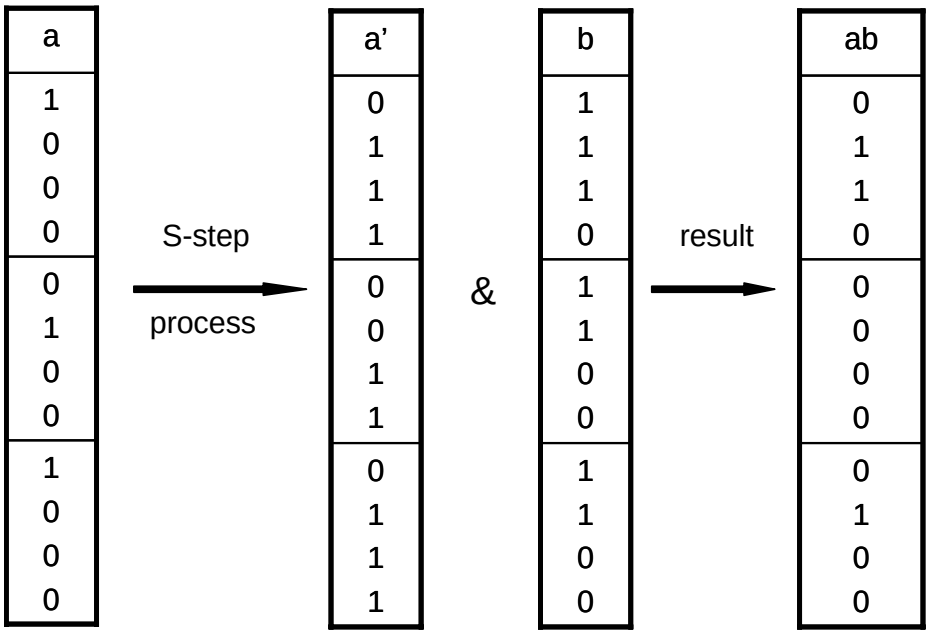


図３.８　({a})と({b})を結合して({a},{b})を生成する

結合の流れを説明する。まず、＜a＞について、それぞれのセクションではじめて１が出てくるまで０にし、それより後を１に置き換える（図３.８における S-step process で、a から a'への変換)。例えば、a の一番上のセクションを見てみると、上から１，０，０，０となっている。１は最初に出現しているので一番上だけ０にし、後はすべて１にする。同様に a の２つ目のセクションを見てみると、０，１，０，０となっており、２番目に１が出現している。よって変換後は、０，０，１，１となる。そして変換したビットマップ（図中では a と b との論理積が＜ab＞を表している（図３.８中では ab が結合した結果を表している)。図３.では CID が１と３のシーケンスで＜ab＞が存在していることを表している。つまりサポート値が２であることが分かる。

同時に起こったトランザクションの結合

図３.８で結合したシーケンス＜ab＞と＜d＞を結合して＜a(bd)＞を生成する例を考える。結合の流れを図３.９に示す。

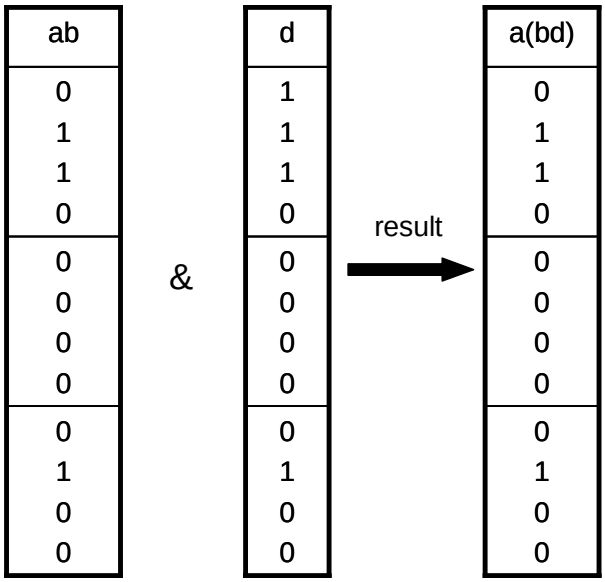


図３.９　＜ab＞と＜d＞を結合して＜a(bd)＞を生成する

同時に起こったトランザクションの結合はそれぞれのビットマップの論理積をとることで生成することができる。図３.９の例では、＜a (bd)＞（図中では a(bd)）のサポート値が２であることが分かる。

３.５　PrefixSpan[6]

PrefixSpan は 2001 年に Pei らによって提案された射影ベースのアルゴリズムである。射影ベースのアルゴリズムとは、候補シーケンスを生成せずに、データベースを射影することによって頻出シーケンスを抽出するアルゴリズムである。すなわち、PrefixSpan は候補シーケンスを生成せずに、Prefix projection という特殊な射影方法と射影によって生成される Prefix データベースを用いることで、マイニングの高速化を実現している。本節では、まず Prefix projection と Prefix データベースについての説明をした後、具体的なアルゴリズムの流れを説明する。また、PrefixSpan を最適化したものとして bi-level projection と、Pseudo-Projection の２つを紹介する。

３.５.１　Prefix projection と Prefix データベース

PrefixSpan では Prefix Projection と呼ばれる射影を行っている。Prefix Projection とは、射影元のシーケンスから射影対象のシーケンスより後ろに存在するアイテムからなるシーケンスのみを抽出する射影である。例えばシーケンス＜a (abc) (ac) d(cf)＞の射影について考えてみる。ここで、＜a＞、＜aa＞、＜ab＞について射影を行った結果が表３.１７である。

表３.１７　Prefix projection

Prefix	射影後のシーケンス
＜a＞	＜(abc) (ac) d (cf)＞
＜aa＞	＜(_bc) (ac) d (cf)＞
＜ab＞	＜(_c) (ac) d (cf)＞

＜a＞について射影を行ったときは先頭の a を省いたシーケンスが射影後のシーケンスとなる。同様に、＜aa＞についても最初の ab が除かれている。＜ab＞についての射影では、シーケンス内で最初に出てくる ab 以降が射影後のシーケンスとなる。そして、与えられたデータベースに対し、射影を行った結果のデータベースを Prefix データベースという。

３.５.２　PrefixSpan アルゴリズムの流れ

PrefixSpan アルゴリズムの流れを以下に示す。

１、長さ１の頻出シーケンスを抽出する
２、深さ優先探索で射影を行い、マイニングを行う。

表３.１８のデータベースが与えられたときの PrefixSpan の動きを以下に示す。なお、min_sup=2 とする。

表３.１８　シーケンスデータベース

SID	Sequence
10	＜a (abc) (ac) d (cf)＞
20	＜(ad) c (bc) (ae)＞
30	＜(ef) (ab) (df) c b＞
40	＜e g (af) c b c＞

１、長さ１の頻出シーケンスを抽出する

データベースをスキャンし、長さ１の頻出アイテムを抽出する。表３.１８の例では長さ１の頻出シーケンスは＜a＞:4、＜b＞:4、＜c＞:4、＜d＞:3、＜e＞:3、＜f＞:3 の６つとなる。ここで、各シーケンスの後ろの数字はサポート値である。

２、深さ優先探索で射影を行い、マイニングを行う（図３.９参照）

例ではまず、＜a＞について射影を行い、射影されたデータベースを構築し、a を先頭とする長さ２の頻出シーケンスを抽出する。ここでは＜aa＞:2、＜ab＞:4、＜(ab)＞:2、＜ac＞:4、＜ad＞:2、＜af＞:2 の６つが抽出される。PrefixSpan は深さ優先探索でマイニングを行うので、次は＜aa＞について射影を行う。射影によって構築されたデータベースが表３.１９である。ここでそれぞれのサポート値をカウントすると、＜aaa＞:1、＜a(ab)＞:1、＜a(ac)＞:1、＜aac＞:1、＜aad＞:1、＜aae＞:1、＜aaf＞:1 となり最小サポート以上のシーケンスは存在しないので、＜aa＞より深く射影を行うことは不可能である。よって次は、＜ab＞について

射影を行う。以降も同様に深さ優先で射影したデータベースにおいて、最小サポート以上のアイテムが存在しなくなるまでマイニングを行っていく。具体的には表３.２０のようになる。表３.の Frequent Patterns の＜a＞、＜aa＞、…、＜af＞,＜b＞、…＜ec＞の順に射影を行う。また、表３.１８のマイニングの全体像を図３.１０に示す。

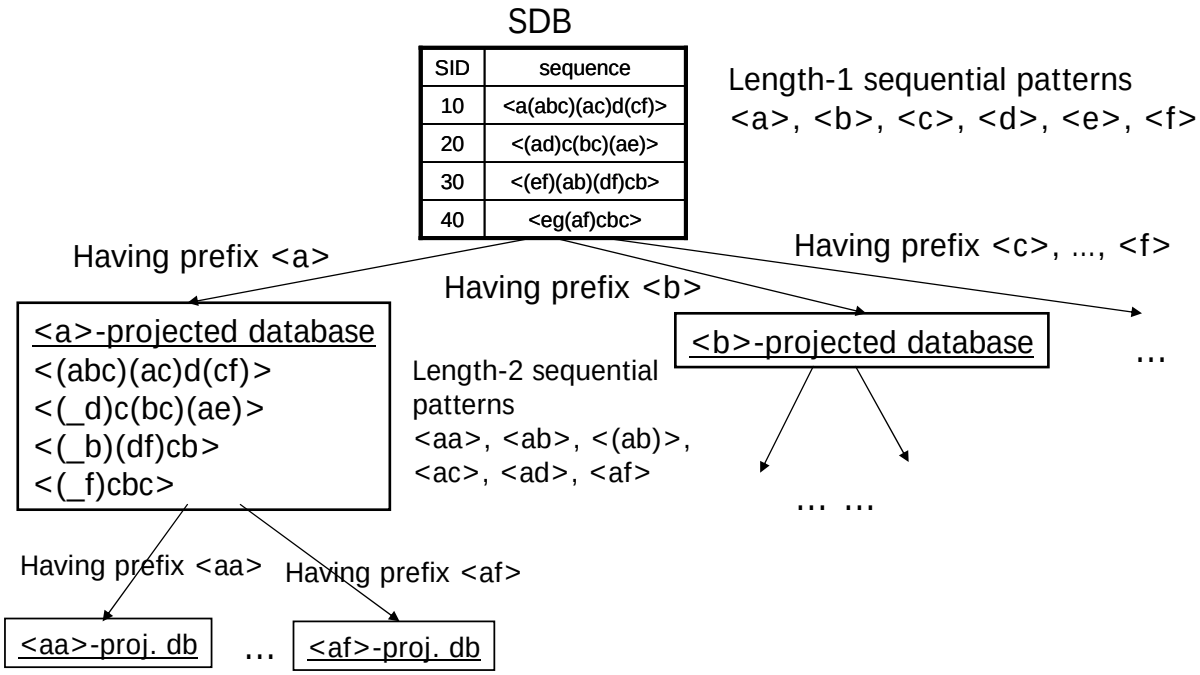


図３.１０　PrefixSpan の流れ

表３.１９ ＜aa＞について射影した結果

＜aa＞-projected database
＜(_bc)(ac)d(cf)＞
＜(_e)＞

表３.２０　射影されたデータベースとシーケンシャルパターン

Prefix	Projected(postfix) database	Frequent Patterns
＜a＞	＜(abc)(ac)d(cf)＞、＜(_d)c(bc)(ae)＞、＜(_b)(df)cb＞、＜(_f)cbc＞	＜a＞、＜aa＞、＜ab＞、＜a(bc)＞、＜a(bc)a＞、＜aba＞、＜abc＞、＜(ab)＞、＜(ab)c＞、＜(ab)d＞、＜(ab)dc＞、＜(ab)f＞、＜ac＞、＜aca＞、＜acb＞、＜acc＞、＜ad＞、＜adc＞、＜af＞
＜b＞	＜(_bc)(ac)d(cf)＞、＜(_c)(ae)＞、＜(df)cb＞、＜c＞	＜b＞、＜ba＞、＜bc＞、＜(bc)＞、＜(bc)a＞、＜bd＞、＜bdc＞、＜bf＞
＜c＞	＜(ac)d(cf)＞、＜(bc)(ae)＞、＜b＞、＜bc＞	＜c＞、＜ca＞、＜cb＞、＜cc＞
＜d＞	＜(cf)＞、＜c(bc)(ae)＞、＜(_f)(cb)＞	＜d＞、＜db＞、＜dc＞、＜dcb＞
＜e＞	＜(_f)(ab)(df)cb＞、＜(af)cbc＞	＜e＞、＜ea＞、＜eab＞、＜eac＞、＜eacb＞、＜eb＞、＜ebc＞、＜ec＞、＜ecb＞、＜ef＞、＜efb＞、＜efc＞、＜efcb＞
＜f＞	＜(ab)(df)cb＞、＜cbc＞	＜f＞、＜fb＞、＜fbc＞、＜fc＞、＜fcb＞

３.５.３　PrefixSpan の最適化

PrefixSpan における主要なコストは、射影されたデータベースを構築することにある。つまり、射影されたデータベースのサイズを減らすことができれば、PrefixSpan の速度を向上させることができる[4]。[4]では、PrefixSpan の最適化として bi-level projection と Psuedo-Projection の 2 つの手法が提案されている。

bi-level projection

bi-level projection は長さが奇数の頻出シーケンスを抽出するときは通常の PrefixSpan と同じように射影を行い、射影データベースを構築するが、長さが偶数の頻出シーケンスを抽出する際にはデータベースをスキャンし、三角行列形式のデータベースを構築する。このデータベースは通常の PrefixSpan の射影により構築されるデータベースよりもサイズを削減することができる。

再び表３.の例を用いて説明する。また、min_sup=2 とする。まずは、通常の PrefixSpan と同様に、データベースをスキャンし、長さ 1 の頻出アイテムを抽出する。抽出されるのは＜a＞、＜b＞、＜c＞、＜d＞、＜e＞、＜f＞の６つである。

次のステップでは､通常の PrefixSpan では射影を行い､新たなデータベースを構築するが、bi-level projection では、表３.１４のような 6×6 の三角行列を構築する。ここでは構築された行列を M とする。

表３.１４　長さ２における行列 M

a	2					
b	(4,2,2)	1				
c	(4,2,1)	(3,3,2)	3			
d	(2,1,1)	(2,2,0)	(1,3,0)	0		
e	(1,2,1)	(1,2,0)	(1,2,0)	(1,1,0)	0	
f	(2,1,1)	(2,2,0)	(1,2,1)	(1,1,1)	(2,0,1)	1
	a	**b**	**c**	**d**	**e**	**f**

それぞれの行列の要素は長さ 1 の頻出アイテムを組み合わせて生成した、長さ 2 のサポート値を表している。また、行列には値を３つ含んでいるもの(M[a,c](a と c の要素)=(4,2,1))と、1 つだけ含んでいるものの 2 種類が存在する。値が 1 つだけのものは対角線上に存在し、例えば、M[a,a]=2 は＜aa＞のサポート値が２であることを表している。また、値を 3 つ含んでいるものは対角線以外の場所に存在する。例えば、M[a,c]=(4,2,1)における＜ac＞のサポート値が４、＜ca＞のサポート値が２、＜(ac)＞のサポート値が１であることを表している。M[c,a]の表す値は M[a,c]の値と対称になっているので、表現する必要が無い。よって、データベースを削減することが可能になる。

長さ２の頻出シーケンスについては通常の PrefixSpan と同様に射影を行い、データベースを構築する。例えば表３.１４における頻出シーケンスの一つである＜ab＞について射影を行い、構築されたデータベースが表３.１５である。

表３.１５　＜ab＞について射影した結果

projected database
＜(_c)(ac)(cf)＞
＜(_c)a＞
＜c＞

ここで、表３.１５のデータベースに対しスキャンを行い、長さ１の頻出シーケンスを抽出する。ここで抽出される長さ１の頻出シーケンスは＜a＞、＜c＞、＜(_c)＞の３つである。これをもとに表３.１６のような＜ab＞について射影した行列を生成する。表３.１６の中のΦという記号は組み合わせを生成できないという意味である。

表３.１６　＜ab＞について射影した行列

a	0		
c	(1,0,1)	1	
(_c)	(Φ,2,Φ)	(Φ,1,Φ)	Φ
	a	**c**	**(_c)**

ここで、最小サポート値を満たす組み合わせは＜(_c)a＞だけなので、これ以上射影を行う必要は無い。

以降も長さが奇数の頻出シーケンスを抽出するときは通常の PrefixSpan と同じように射影を行い、長さが偶数の頻出シーケンスを抽出するときは三角行列形式のデータベースを構築する。そして、通常の PrefixSpan と同様に深さ優先探索でマイニングを行っていく。この結果、表３.のデータベースのマイニングにおいて、通常の PrefixSpan では合計で 53 回射影されたデータベースを構築しなければならなかったが、bi-projection では、射影されたデータベースの構築は 22 回ですむ。

Pseudo Projection

Prefix Projection とは、シーケンス s_1=＜a(abc)(ac)d(cf)＞が与えられた場合、＜a＞について射影した結果、＜(abc)(ac)d(cf)＞、＜ab＞について射影した結果、＜(_bc)(ac)d(cf)＞が得ることができる射影である。Psuedo Projection では射影されたデータベース（シーケンス）を冗長なものとみなし、別の表現を用いることによって高速化を図っている。具体的には、それぞれの射影を、射影元のシーケンスへのポインタと射影されたシーケンスのオフセットという２つの情報で表現する。ポインタとは射影する対象、オフセットは射影されたシーケンスが射影元のシーケンスの中で何番目以降が射影されているかを表している。s_1 の例を用いて説明すると、s_1 における＜a＞についての射影はポインタが s_1、オフセットが２と表される。＜a＞について射影したシーケンスは＜(abc)(ac)d(cf)＞なので、s_1 の中で 2 番目以降のアイテムで表されている。よってオフセットは２となる。同様に、s_1 の＜ab＞についての射影はポインタが s_1、オフセットが４と表すことができる。

３.６　各手法の比較

本章で紹介した４つの手法について、候補シーケンス削減、データベース分割、シーケンスの縮小という 3 つの戦略から比較してみる。

(1)　候補シーケンス削減

候補シーケンス削減とは、頻出シーケンスに為り得ない候補シーケンスをできるだけ早い段階で削除する戦略で、プロセスのコストとサポート値カウントの際のオーバーヘッドを減らすことができる。候補シーケンス削減の戦略は GSP[3]、SPADE[5]、SPAM[6]、PrefixSpan[4]のすべてのアルゴリズムで用いられている。

(2)　データベース分割

データベース分割は、データベースを複数のグループに分割する戦略で、それぞれのグループをメインメモリにうまく割り当てることができるようになる。SPADE[5]と SPAM[6]では、候補シーケンスをもとにしてデータベースを分割している。PrefixSpan[4]では、射影によって得られた頻出アイテムをもとにデータベースを分割している。GSP[3]ではデータベー

ス分割の戦略は用いられていない。

(3) シーケンスの縮小

シーケンスを減らす戦略とは、できるだけ多くのシーケンスを減らし、プロセスのコストを現象させる戦略である。この戦略は PrefixSpan[4]のみで用いられており、射影によってシーケンスの縮小を実現している。例えば、シーケンス＜(f) (ag) (bfh) (bf)＞に対し、＜a＞について射影を行うと＜(_g) (bfh) (bf)＞となり、シーケンスを縮小することができる。

以上の比較をまとめたものが表３.１７である。

表３.１７　アルゴリズムと戦略

	候補シーケンス削減	データベース分割	シーケンスの縮小	アルゴリズムの特徴
GSP[3](1996)	✔			Apriori をベースに候補シーケンスを生成
SPADE[4](2000)	✔	✔		Lattice と ID-List を用い高速化を実現
SPAM[5](2001)	✔	✔		ID-List をビットマップで表現
PrefixSpan[6](2001)	✔	✔	✔	Prefix Projection によってシーケンスデータベースを縮小

３.７　Sequential Pattern Mining の拡張[11]

近年、Sequential Pattern Mining の拡張手法として、Stream Mining という研究が盛んになっている([12][13])。Stream Mining とは、無限に入ってくるデータ(data stream)をマイニングの対象とした手法で、Web ページ閲覧の流れの解析、エネルギー消費測定、ネットワーク上のデータの解析、株取引といったデータが継続して入ってくる分野に応用が可能である。Stream Mining の概要を以下で説明する。

上述したように、Stream Mining が対象とするデータは、無限に流入してくるデータである。しかし、これに対して計算機の資源は有限であり、Stream Mining では有限な計算機資源を用いてどのようにマイニングを行なうかということが課題となっている。

この課題に対する解決方法として以下の２つの方法が挙げられる。

①　近似的な解を求める

図３.１１のようにデータ要約という小さなデータ構造を主記憶上に保持し、近似的な解を求める方法である。

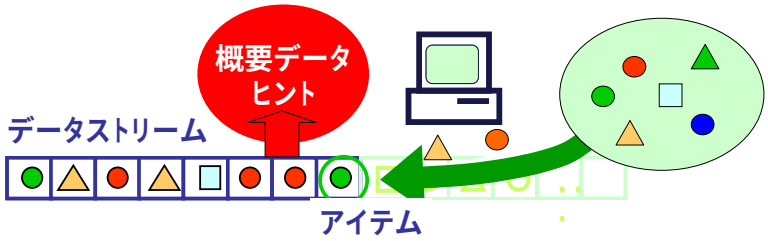


図３.１１　Data Stream から近似解を得る

②　ある時点にスポットを当ててマイニングを行なう

図３.１２のように対象をすべての過去のデータではなくある一定の時間内のデータに絞ってマイニングを行う方法である。

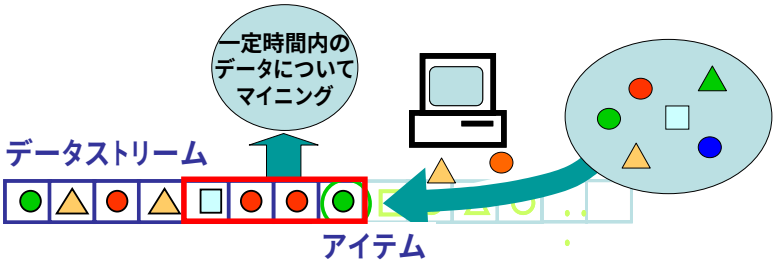


図３.１２　一定時間内のデータに絞ってマイニング

第４章　提案手法

本章では提案手法であるアイテム間の距離（時間）を考慮した Sequential Pattern Mining について述べる。まず、Sequential Pattern Mining における順序と時間について述べた後、提案手法についての具体的な説明をする。

４.１　Sequential Pattern Mining における順序と距離

Sequential Pattern Mining の最大の特徴はアイテム間の順序を考慮するところにある。現実の世界にはアイテム間の順序が重要となってくる事例が数多く存在する。例えば、最初にパソコンを購入し、その後 CD-ROM を購入した人と、最初に CD-ROM を購入し、その後パソコンを購入した人がいるとする。Frequent Pattern Mining では、両者は同じ組合せとして扱われている。しかし、最初に CD-ROM を購入し、その後パソコンを購入するという行動は考え難い。そこで、アイテム間の順序を考慮する点が特徴である Sequential Pattern Mining では、両者を別の組合せとして、区別してマイニングを行なう。

しかし、従来の Sequential Pattern Mining では、アイテム間の距離が考慮されないという問題がある。距離（時間）を考慮しないでマイニングを行なうと、短期的な行動と長期的な行動を区別することができない。シーケンス＜a b＞では a と b の間の距離は分からないのである。上の例を用いると、最初にパソコンを購入し、次に CD-ROM を購入するという行動の中でも１週間後に CD-ROM を購入する人と、１年後に CD-ROM を購入する人とではそれぞれ行動の持つ意味は異なっているのだが、Sequential Pattern Mining では両者の行動を同じものとして扱っている。

４.２　アイテム間の距離（時間）を考慮した Sequential Pattern Mining

４.１で説明したように、Sequential Pattern Mining では、アイテム間の距離（時間）によってトランザクションが区別されない。そのため、短期的な行動と長期的な行動という持つ意味の異なる 2 つの行動を区別することができない。図４.１にレンタルビデオ店の顧客の行動を例に示す。上の人物はシリーズ物のビデオを一度に 8 本まとめて借りている。それに対し、下の人物は、2 本だけ借りて、見終わったらまた 2 本借りている。このように、同じシリーズ物をレンタルする場合でも、人によって借り方が異なってくる。従来の Sequential Pattern Mining では、図４.１の２人の行動は全く別のものとして扱われてしまう。

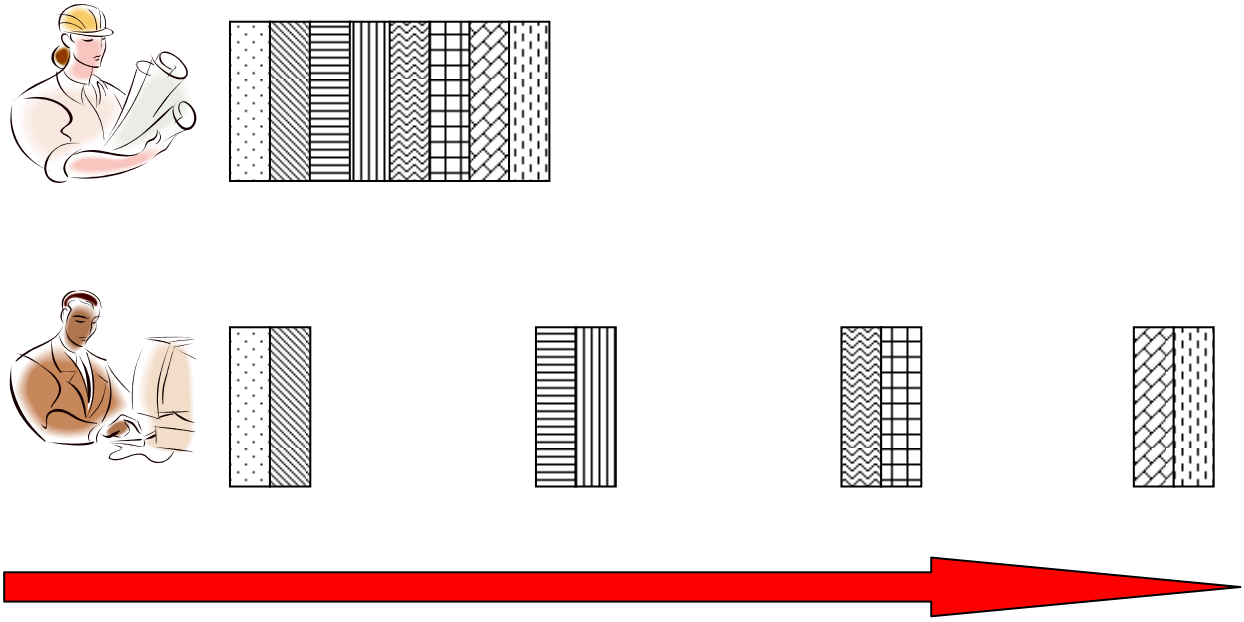


図４.１　シリーズ物の借り方の違い

以上のような問題を解決するために、本論文では、アイテム間の距離（時間）を考慮した Sequential Pattern Mining を提案する。

提案手法は、ユーザが定義した制限距離（時間）以下のトランザクションは、同時に起こったものとみなしてマイニングを行う。つまり、従来の Sequential Pattern Mining では別々のものとして扱われていたトランザクションの中で、ユーザにとっては同時に行われているとみなしてもよい距離のトランザクションを結合し、マイニングを行う。以下で、制限距離について定義を行なう。

シーケンス＜AB＞が与えられたとする。また、AB 間の距離を a（a≧0）とする。ここで、制限距離 x が与えられた場合について考える。a≦x の時は、図４.２のように AB が同時に起こったとみなして＜(AB)＞とする。

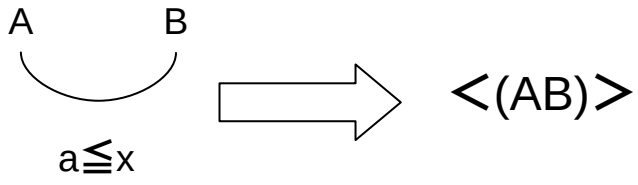


図４.２　アイテム間の距離が制限距離以下の場合

しかし、a＞x の時は、図４.３のように＜AB＞のままとする。

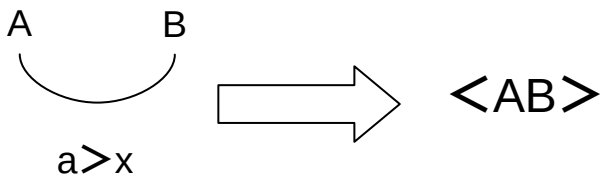


図４.３　アイテム間の距離が制限距離より大きい場合

また、制限距離以下のトランザクションが連続した場合は、連続した制限距離以下のトランザクションをすべて一つの” ( ) “でまとめる。ここでシーケンス＜ABCD＞が与えられている例について考える。AB 間の距離を a、BC 間の距離を b、CD 間の距離を c とする。制限距離が x が与えられたときに、a≦x、b≦x、c＞x ならば、シーケンスは図４.４のように＜(ABC)D＞となる。

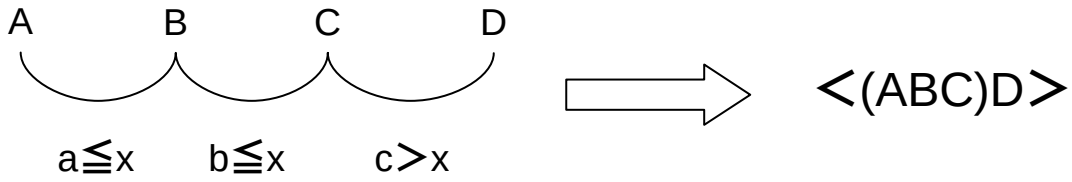


図４.４　制限距離以下のトランザクションが連続した例

４.３　提案手法のアルゴリズムの流れ

提案手法は、データ変換とマイニングの２つのフェーズに分けることができる。

データ変換のフェーズでは、制限距離（時間）n（n≧0）を与え、マイニングの対象となるデータベースから、距離が n 以下のトランザクションは同時に起こったものとしてシーケンスデータベースを作成する。

マイニングのフェーズでは、データ変換のフェーズで得られたシーケンスデータベースを元に実際にマイニングを行なう。提案手法では Psuedo-Projection 方式の PrefixSpan[9]を用いてマイニングを行なう。PrefixSpan の詳細な説明は３.５章において行なっている。

以下に提案手法の流れを例を用いて示す。

データ変換

マイニング対象のデータベースは表４.１のようなものである。

表４.１　マイニングに使用するデータベースの例

ユーザ ID	トランザクションタイム	トランザクション
１０	４月３０日	A
１０	４月３０日	B
１０	７月２４日	D
２０	５月１３日	B
２０	９月１７日	C
３０	４月２日	A
３０	６月２７日	B
３０	６月２８日	C
３０	７月１日	D

表４.１のデータベースをもとにシーケンスを作成する。提案手法を用いずに従来どおりにシーケンスを作成したものが表４.２である。

表４.２　通常通り作成したシーケンスデータベース

SequenceID	Sequence
１０	＜(A B) D＞
２０	＜B C＞
３０	＜A B C D＞

提案手法では、表４.１のような基のデータベースにおいて、ユーザが定義した距離内のトランザクションを同時に起こったものとして変換を行なう。表４.１のデータベースを制限距離を１０日にしてシーケンスを作成したシーケンスデータベースが表４.３である。

表４.３　制限距離をもとに作成したシーケンスデータベース

SequenceID	Sequence
１０	＜(A B) D＞
２０	＜B C＞
３０	＜A (B C) D＞

表４.２と表４.３のシーケンスデータベースを比較してみると、シーケンス ID３０のシーケンスにおいて、従来手法では＜A B C D＞になっているが、提案手法では、＜A (B C) D＞となっている。表４.１のデータベースを見てみると、トランザクション B は 6 月 27 日に行われており、トランザクション C は 6 月 28 日に行われている。制限距離（時間）は１０（日）なので、提案手法では、トランザクション B と C は同時に行われたと見なし、一つのトランザクション(B C)として扱う。

マイニング

提案手法によって、変換されたシーケンスデータベースについて、実際にマイニングを行う。上で述べたように、提案手法では Pseudo Projection 方式の PrefixSpan[9]を用いてマイニングを行う。

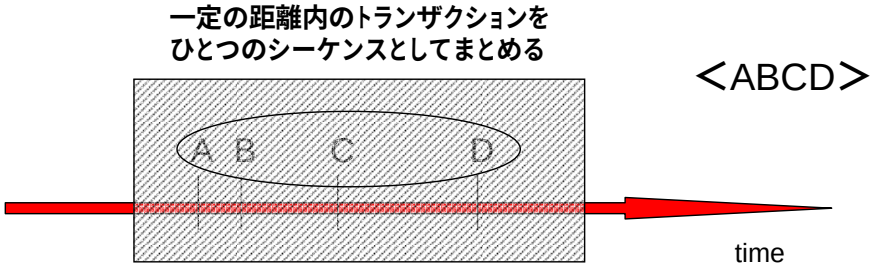
４.４　Stream Mining の観点から見た提案手法

関連研究でも紹介した Sequential Pattern Mining の拡張手法である Stream Mining の観点から見た提案手法はどのような位置づけになるのかについて説明する。

提案手法は制限距離をユーザが定義し、制限距離以下のトランザクション同士を一つのトランザクションにまとめる手法であるが、制限距離を設けて時間で区切るという点については Stream Mining の方法の一つ（関連研究における②の方法）と共通している。では、提案手法と Stream Mining の違いはどのような点にあるのだろうか。

Stream Mining ではある一定の距離（時間）内に起こったトランザクションを一つのシーケンスにまとめてマイニングを行なっている。それに対し、提案手法では、制限距離内にあるトランザクションを一つのトランザクションにまとめている。図４.５の例では A、B、C、D の４つのトランザクションが起こっているが、Stream Mining では制限時間内に入っているすべてのトランザクションをひとつのシーケンスとしてまとめるので、シーケンス＜ABCD＞になる。しかし、提案手法ではトランザクション間の距離に着目してその制限距離以下のトランザクションを一つのトランザクションにまとめるので、シーケンスは＜(AB)CD＞となる。

Stream Mining



提案手法

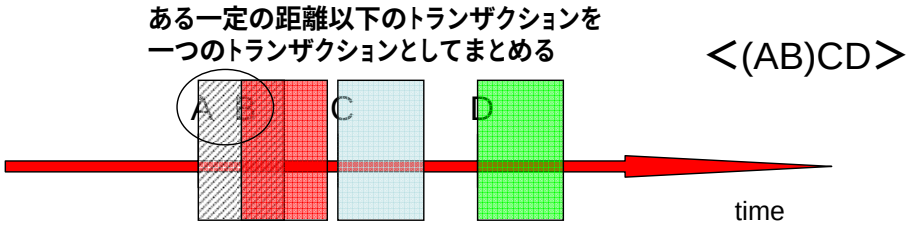


図４.５　制限距離の区切りによるトランザクションのまとめ方の違い

つまり、Stream Mining の観点から提案手法をみた場合、制限距離に対するトランザクションのまとめ方という点において Stream Mining とは異なっている。

第5章 性能評価

本章では、第 4 章で述べた提案手法を実装し、抽出されたシーケンスについて評価を行う。

５.１　実験環境

実験を行った計算機の環境を以下に示す。

表５.１　計算機の環境

CPU	Celeron M 1.4GHz
メモリ	752MB
OS	Windows XP Home Edition
コンパイラ	Borland C++ 5.5.1 for Win32

５.２　データセット

今回、実験に用いたデータセットとして、MUSASHI(Mining Utilities and System Architecture for Scalable processing of HIstorical data)[10]の顧客購買履歴データ生成スクリプトによって生成された人工データを使用した。MUSASHI では、店、日付、時間、レシート、顧客、商品、大分類、中分類、小分類、細分類、メーカー、ブランド、仕入単価、単価、数量、金額、仕入金額、粗利金額を含むデータを生成することができるが、今回はこの中から、日付、顧客、商品の３つのデータを切り取って使用した。ここで、使用した３つのデータについて生成する際に調整可能なパラメータについて表５.２に示す。

表５.２　調整可能なパラメータ

項目	計算ルール	使用したパラメータ
店別顧客人数	手入力にて店コードと来店人数を指定する	店舗数:1 顧客数:2000 人
顧客の一来店あたりの購入数量	正規分布に基づく乱数にて決定	nrand(nrand(5 個,2 個),3) 但し、０以下は省く
顧客の来店間隔	正規分布に基づく乱数にて決定	nrand(30 日,10 日)
顧客の来店開始日	ある日付を中心に正規分布するように乱数で決める	07/01＋nrand(0,200 日)
顧客の来店終了日	ある日付を中心に正規分布するように乱数で決める	07/01＋nrand(0,200 日)
購入商品	商品によって売れる頻度を乱数で変える	nrand(商品番号の中点, 商品数/3)

生成されたデータセットには表５.３のような顧客 ID、購入日、購入アイテムが含まれている。

表５.３　生成されたデータセットの例

顧客 ID	購入日(月/日)	購入アイテム
1	1/26	234
1	1/26	73
2	2/9	547
3	2/1	170
3	2/9	365

上述の MUSASHI によって生成されたデータセットをもとにして、顧客 ID ごとにシーケンスを作成する。作成するシーケンスの形式をシーケンス＜(2 5 7) (1 2) (3 9) 4＞を例に説明する。

2　5　7　-1　19　1　2　-1　6　3　9　-1　30　4　-1　-2

－１はトランザクションの終了、－２はシーケンスの終了を表す。また、－１の直後の数字（上の数列では１９と６と３０）はトランザクション間の距離を表している。例えば、上の数列において、トランザクション(2 5 7)と(1 2)の距離は１９である。また、各トランザクション内のアイテムは昇順にソートされている。

このような形式の数列からトランザクション間の距離を除いた形に変換する。上の例の場

合、以下のような形になる。

2　5　7　-1　1　2　-1　3　9　-1　4　-1　-2

ここで、提案手法を用いた場合、ユーザが定義した制限距離以下のトランザクションは結合される。上の例において、制限距離を１０とした場合、以下のような形になる。

2　5　7　-1　1　2　3　9　-1　4　-1　-2

距離が 10 以下のトランザクション間は(1 2)と(3 9)なので、この 2 つが結合され、＜(2 5 7) (1 2 3 9) 4＞となる。

なお、実際にマイニングを行う際には、作成したシーケンスデータベースはバイナリ形式になっている。

今回の実験では、シーケンス数 2000、トランザクションが行なわれる期間が 1 年、アイテム数 577 となっている。

５.３　実験

５.３.１　実験方法

MUSASHI によって生成したデータセットに対して提案手法を用い、シーケンスを変換してマイニングを行なった場合と、従来手法を用いてマイニングを行なった場合について比較する。ここでいう従来手法とは、与えられたデータに対し、前処理を何もせずにマイニングを行なう手法を指す。また、それぞれの手法に対し、最小サポート値を変えて、マイニングを行なった結果についても比較を行なう。

５.３.２　実験結果

提案手法を用い、シーケンスの変換を行なった場合と、従来手法（提案手法を用いない場合）について、それぞれマイニングを行ない、比較を行なった。最小サポート値は 100%～82%まで変化させ、それぞれの抽出時間、抽出シーケンス数、1 シーケンスあたりの抽出時間について比較したものをまとめたのが表５.４である。また、抽出時間、頻出シーケンス数、1 シーケンスあたりの抽出時間のグラフをそれぞれ図５.１、図５.２、図５.３に示す。

表５.４　従来手法と提案手法の比較

	従来手法			提案手法		
min_sup	**時間 (sec)**	**頻出シーケンス数**	**１シーケンスあたりの抽出時間（Sec）**	**時間 (sec)**	**頻出シーケンス数**	**１シーケンスあたりの抽出時間 (Sec)**
100%	0.01	3	0.003333	0.01	2	0.005
99%	0.02	9	0.002222	0.04	20	0.002
98%	0.02	10	0.002	0.04	21	0.001905
97%	0.02	10	0.002	0.04	21	0.001905
96%	0.03	13	0.002308	0.05	24	0.002083
95%	0.03	13	0.002308	0.05	31	0.001613
94%	0.13	71	0.001831	0.07	46	0.001522
93%	0.13	71	0.001831	0.07	47	0.001489
92%	0.17	96	0.001771	0.08	47	0.001702
91%	0.18	96	0.001875	0.08	47	0.001702
90%	0.18	96	0.001875	0.12	83	0.001446
89%	0.18	97	0.001856	0.12	83	0.001446
88%	1.832	1195	0.001533	0.821	658	0.001248
87%	3.575	2225	0.001607	1.001	786	0.001274
86%	8.292	5243	0.001582	1.502	1093	0.001374
85%	28.811	25589	0.001126	2.463	2035	0.00121
84%	38.715	28118	0.001377	4.266	3350	0.001273
83%	207.528	164854	0.001259	7.270	6787	0.001071
82%	901.607	1113153	0.00081	22.472	23913	0.00094

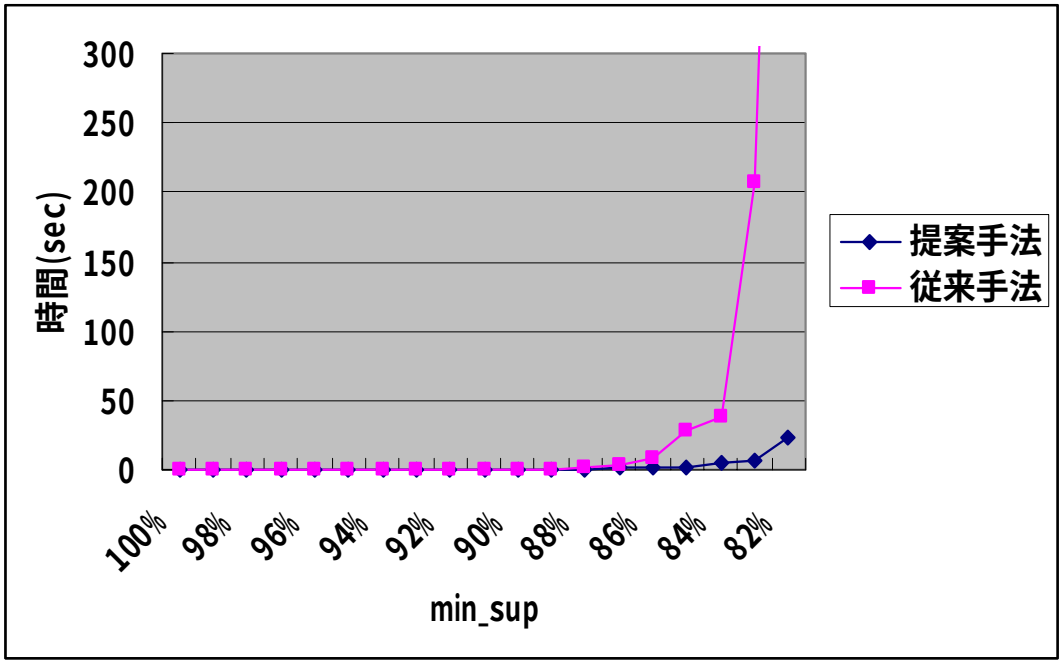


図５.１　抽出時間の比較のグラフ

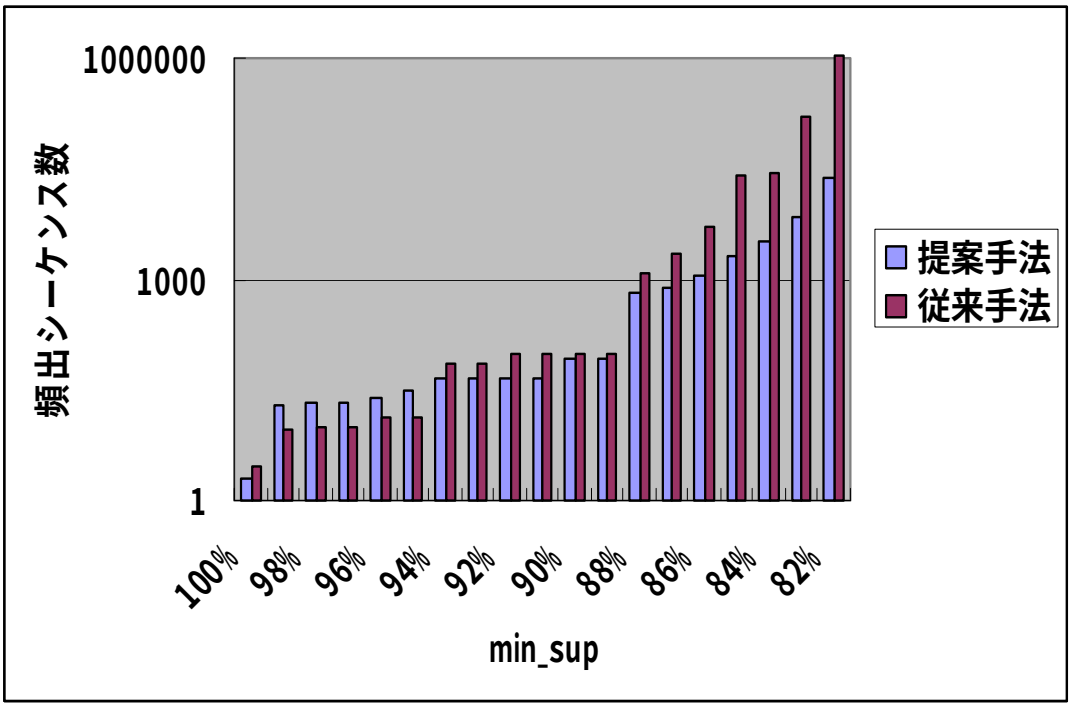


図５.２　抽出される頻出シーケンスの数の比較

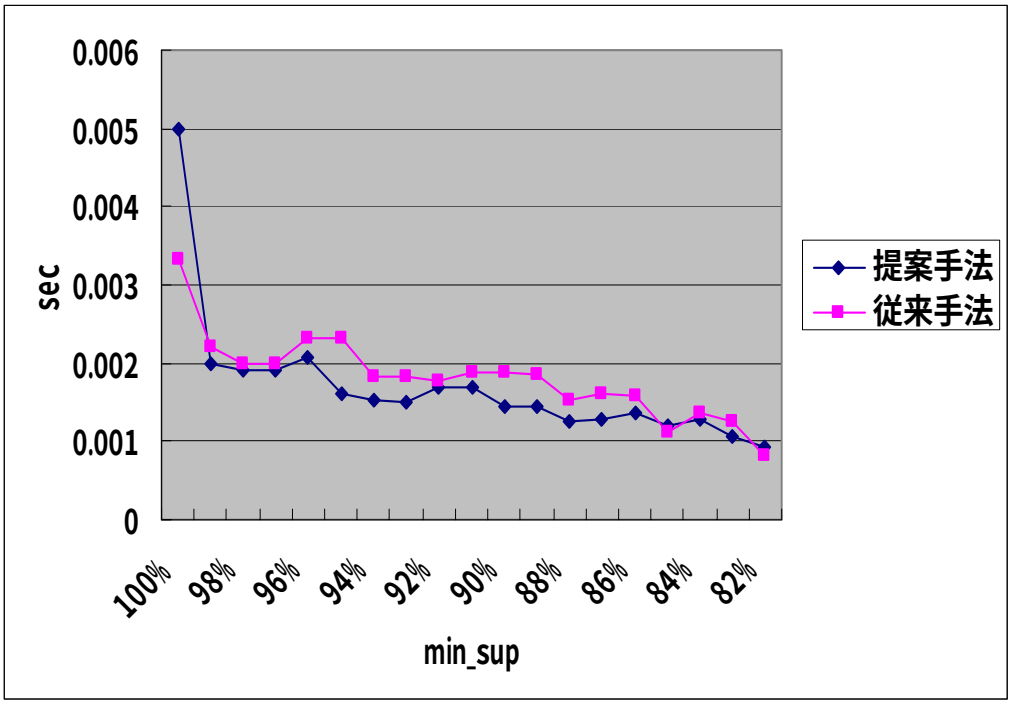


図５.３　１シーケンスあたりの抽出時間

次に、提案手法と従来手法では、抽出されるシーケンスの長さに変化があるのかを比較してみる。まず、提案手法を用いた場合に各最小サポート値ごとに抽出されるシーケンスを長さ別の個数を表５.５に示す。なお、表の縦軸は最小サポート値、横軸はシーケンスの長さである。

表５.５　提案手法を用いた場合に抽出されるシーケンスの長さごとの個数

	1	2	3	4	5	6	7	8	9	10	11
100%	2	1									
99%	8	7	4	1							
98%	9	7	4	1							
97%	9	7	4	1							
96%	9	9	5	1							
95%	9	12	8	2							
94%	14	15	11	5	1						
93%	14	16	11	5	1						
92%	14	16	11	5	1						
91%	14	16	11	5	1						
90%	14	22	24	16	6	1					
89%	14	22	24	16	6	1					
88%	25	132	193	166	95	37	9	1			
87%	28	175	240	192	103	38	9	1			
86%	32	254	343	268	138	47	10	1			
85%	38	367	617	562	317	110	22	2			
84%	39	421	883	966	660	288	79	13	1		
83%	44	538	1402	1879	1583	893	344	89	14	1	
82%	50	750	2932	5379	6045	4556	2381	861	207	30	2

同様に従来手法の場合における、抽出される頻出シーケンスの長さ別の個数を表５.６に示す。

表５.６　従来手法を用いたときに抽出されるシーケンスの長さごとの個数

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
100%	2	1															
99%	8	1															
98%	9	1															
97%	9	1															
96%	9	3	1														
95%	9	3	1														
94%	14	38	17	2													
93%	14	38	17	2													
92%	14	39	29	12	2												
91%	14	39	29	12	2												
90%	14	39	29	12	2												
89%	14	40	29	12	2												
88%	25	170	332	344	218	85	19	2									
87%	28	227	584	653	448	208	64	1									
86%	32	314	1033	1359	1209	790	367	115	22	2							
85%	38	465	2137	4330	5915	5747	4010	2018	723	177	27	2					
84%	39	498	2388	5000	6656	6257	4247	2090	736	178	27	2					
83%	44	682	4513	14378	27109	35398	34215	25316	14444	6257	1992	441	61	4			
82%	50	960	8259	36177	92303	161485	211702	216652	176240	114739	59724	24646	7922	1922	333	37	2

次に、各最小サポート値における抽出される頻出シーケンスの平均長を比較したグラフを図５.４に示す。

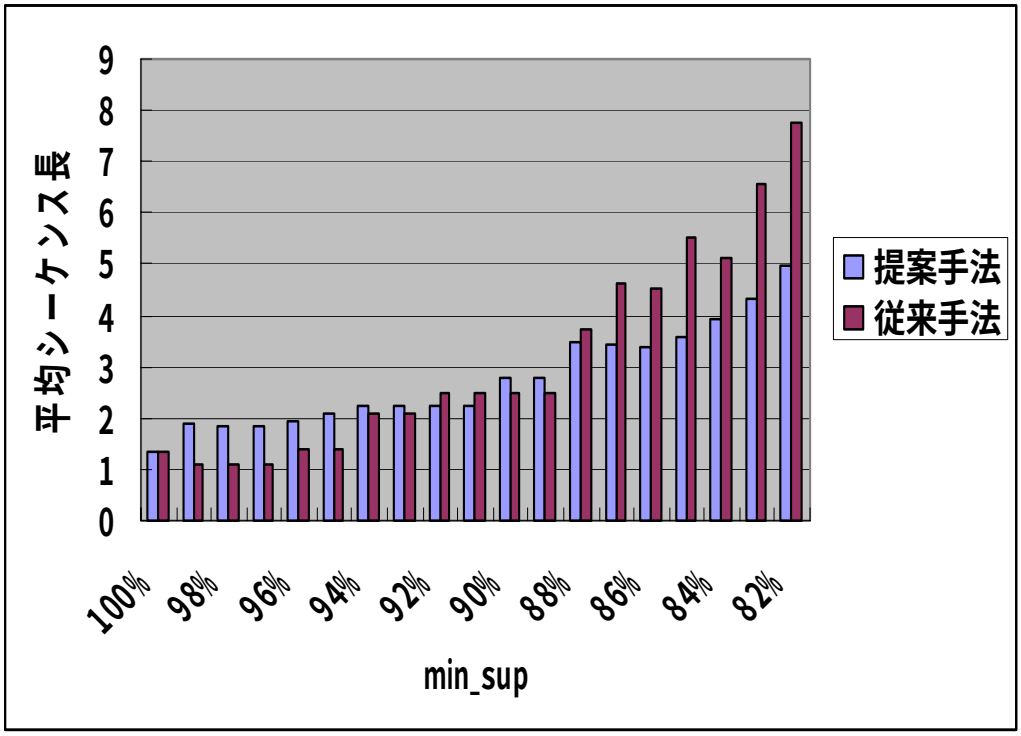


図５.４　各最小サポート値における頻出シーケンスの平均長

５.４　考察

提案手法のシーケンスの結合を行なうことによって起こる変化は以下の 2 つである。

① 従来手法では非頻出であったシーケンスが結合によって頻出になる

２つのシーケンス＜3 4＞と＜(3 4)＞があったとする。従来手法では当然２つのシーケンスは別々だが、前者のシーケンスにおいて、3 と 4 が結合されると、２つとも＜(3 4)＞となる。

② 従来手法では頻出であったシーケンスが結合によって非頻出になる

従来手法ではシーケンス＜1 2＞は頻出であったとする。しかし、提案手法を用いると、一部のシーケンス ID では＜(1 2)＞として扱われてしまうため、＜1 2＞が頻出でなくなる。

表５.４と図５.２から分かるように、最小サポート値が高いときには、提案手法を用いた場合のほうが、従来手法に比べて抽出される頻出シーケンスが多くなっている。しかし、最小サポート値が下がるにつれて、提案手法を用いた場合のほうが、従来手法に比べ、抽出される頻出シーケンス数が少なくなってきている。これは、①の現象は頻出でないものが結合によって頻出になる現象であり、①の現象によって頻出になるシーケンスは最小サポート値の高い時に起こりやすいといえる。また、②の現象は頻出であったものが非頻出になる現象であるために、高いサポート値では非頻出になるために高いサポート値では提案手法のほうが抽出される頻出シーケンスが多くなっている。しかし、最小サポート値が下がると、②で非頻出であったものも頻出になるため、従来手法で抽出される頻出シーケンス数が増大すると考えられる。

また、抽出される頻出シーケンス数が従来手法より少なくなったということは、従来手法では、同じ行動として扱われていたものを、提案手法を用いることにより、区別することができたということである。区別されるということはそれぞれのサポート値が下がるということであり、区別された結果の頻出シーケンスというものは従来手法で抽出されるシーケンスに比べ質の高いものであるということができる。

さらに、表５.４と図５.１が示すように、抽出される頻出シーケンスが少なくなった分、抽出時間も削減することができた。しかし、提案手法を用いた場合と用いない場合では、データの構造自体には変わりはない。そのため、表５.４と図５.３が示すように、抽出されるシーケンス１つあたりの抽出時間においては大きな差は見られなかった。

次に抽出された頻出シーケンスについて着目してみる。抽出される頻出シーケンスの長さは、表５.５、表５.６、図５.４からサポート値が高い時は、提案手法の方が従来手法に比べて長くなっていることが分かる。これは、先に述べたように、提案手法では最小サポート値が高いときは、結合によって頻出になるシーケンスが多い。結合されるということは、少なくとも長さ 2 以上であるので、頻出シーケンスの長さは長くなる。

また、頻出シーケンスのトランザクション数についてみてみると、提案手法はトランザク

ション同士を結合する手法であるので、抽出される頻出シーケンスは従来手法に比べて、”( )“の数が少なくなることが予想される。56 ページから 59 ページに最小サポート値が９０％の時に、提案手法によってマイニングし、抽出された頻出シーケンスと、従来手法によってマイニングを行い、抽出されたシーケンスを載せてある。これらを比較したときに、提案手法を用いた場合の方が、従来手法に比べて、頻出シーケンスの”（ ）”の数が少なくなっていることが確認できた。例えば、提案手法を用いた場合、(353 424 454 549)といったように、”（ ）”の少ない頻出シーケンスが多く見られる。サポート値 90％の場合、頻出シーケンスの平均の”( )“の数は 1.00 であった。しかし、従来手法の場合、(243 424) (383 491)のように”（ ）”の数が多い頻出シーケンスが多く見られるということである。最小サポート値 90％の時の頻出シーケンスの平均の”( )“の数は 1.21 であった。最小サポート値を変えても、同様に提案手法を用いた場合の方が、”（ ）”の少ない頻出シーケンスを多いことが確認できた。これは、当初の予想通り、提案手法は結合を行なっている手法であり、上述の 2 つの現象①、②においても、頻出になる場合は”（ ）”がついている。そのため、提案手法を用いると、”（ ）”の少ない頻出シーケンスが多く抽出されるということがいえる。

提案手法を用いてマイニング(min_sup=90%)を行い抽出された頻出シーケンスを以下に示す。(頻出シーケンス:サポート値)

(218) : 0.944839
(243) : 1.000000
(253) : 0.992900
(315) : 1.000000
(335) : 0.944839
(353) : 0.992900
(383) : 0.944839
(419) : 0.944839
(424) : 0.992900
(454) : 0.990169
(491) : 0.981431
(494) : 0.992900
(526) : 0.944839
(549) : 0.992900
(218 335) : 0.944839
(243 315) : 1.000000
(243 424) : 0.966685
(243 494) : 0.906062
(243 315 424) : 0.966685
(243 315 494) : 0.906062
(243 315 424 494) : 0.906062
(243 424 494) : 0.906062
(253 353) : 0.992900
(253 424) : 0.908247
(253 454) : 0.990169
(253 494) : 0.951939
(253 549) : 0.992900
(253 353 424) : 0.908247
(253 353 454) : 0.990169
(253 353 494) : 0.951939
(253 353 549) : 0.992900
(253 353 424 454) : 0.905516
(253 353 424 494) : 0.908247
(253 353 424 549) : 0.908247
(253 353 424 454 494) : 0.905516
(253 353 424 454 549) : 0.905516
(253 353 424 454 494 549) : 0.905516
(253 353 424 494 549) : 0.908247
(253 353 454 494) : 0.949208
(253 353 454 549) : 0.990169
(253 353 454 494 549) : 0.949208
(253 353 494 549) : 0.951939
(253 424 454) : 0.905516
(253 424 494) : 0.908247
(253 424 549) : 0.908247
(253 424 454 494) : 0.905516
(253 424 454 549) : 0.905516
(253 424 454 494 549) : 0.905516
(253 424 494 549) : 0.908247
(253 454 494) : 0.949208
(253 454 549) : 0.990169
(253 454 494 549) : 0.949208
(253 494 549) : 0.951939
(315 424) : 0.966685
(315 494) : 0.906062
(315 424 494) : 0.906062
(353 424) : 0.908247
(353 454) : 0.990169
(353 494) : 0.951939
(353 549) : 0.992900

(353 424 454) : 0.905516

(353 424 494) : 0.908247

(353 424 549) : 0.908247

(353 424 454 494) : 0.905516

(353 424 454 549) : 0.905516

(353 424 454 494 549) : 0.905516

(353 424 494 549) : 0.908247

(353 454 494) : 0.949208

(353 454 549) : 0.990169

(353 454 494 549) : 0.949208

(353 494 549) : 0.951939

(383 491) : 0.944839

(424 454) : 0.905516

(424 494) : 0.932277

(424 549) : 0.908247

(424 454 494) : 0.905516

(424 454 549) : 0.905516

(424 454 494 549) : 0.905516

(424 494 549) : 0.908247

(454 494) : 0.949208

(454 549) : 0.990169

(454 494 549) : 0.949208

(494 549) : 0.951939

提案手法を用いずに（従来手法で）マイニング(min_sup=90%)を行なった際に抽出された頻出シーケンスを以下に示す。（頻出シーケンス:サポート値）

(218) : 0.944839
(243) : 1.000000
(253) : 0.992900
(315) : 1.000000
(335) : 0.944839
(353) : 0.992900
(383) : 0.944839
(419) : 0.944839
(424) : 0.992900
(454) : 0.990169
(491) : 0.981431
(494) : 0.992900
(526) : 0.944839
(549) : 0.992900
(218 335) : 0.944839
(243) (218) : 0.944809
(243 315) : 1.000000
(243) (335) : 0.944809
(243) (383) : 0.944809
(243) (419) : 0.944809
(243 424) : 0.966667
(243) (491) : 0.944809
(243) (526) : 0.944262
(243) (218 335) : 0.944809
(243 315) (218) : 0.944809
(243 315) (335) : 0.944809
(243 315) (383) : 0.944809
(243 315) (419) : 0.944809
(243 315 424) : 0.966667
(243 315) (491) : 0.944809
(243 315) (526) : 0.944262
(243 315) (218 335) : 0.944809
(243 315) (383 491) : 0.944809
(243 315 424) (218) : 0.921311
(243 315 424) (335) : 0.921311
(243 315 424) (383) : 0.921311
(243 315 424) (419) : 0.921311
(243 315 424) (491) : 0.921311
(243 315 424) (526) : 0.920765
(243 315 424) (218 335) : 0.921311
(243 315 424) (383 491) : 0.921311
(243) (383 491) : 0.944809
(243 424) (218) : 0.921311
(243 424) (335) : 0.921311
(243 424) (383) : 0.921311
(243 424) (419) : 0.921311
(243 424) (491) : 0.921311
(243 424) (526) : 0.920765
(243 424) (218 335) : 0.921311
(243 424) (383 491) : 0.921311
(253) (218) : 0.944809
(253) (335) : 0.944809
(253) (419) : 0.944809
(253) (526) : 0.944262
(253) (218 335) : 0.944809
(315) (218) : 0.944809

(315) (335) : 0.944809

(315) (383) : 0.944809

(315) (419) : 0.944809

(315 424) : 0.966667

(315) (491) : 0.944809

(315) (526) : 0.944262

(315) (218 335) : 0.944809

(315) (383 491) : 0.944809

(315 424) (218) : 0.921311

(315 424) (335) : 0.921311

(315 424) (383) : 0.921311

(315 424) (419) : 0.921311

(315 424) (491) : 0.921311

(315 424) (526) : 0.920765

(315 424) (218 335) : 0.921311

(315 424) (383 491) : 0.921311

(353) (526) : 0.944262

(383 491) : 0.944809

(424) (218) : 0.944809

(424) (335) : 0.944809

(424) (383) : 0.944809

(424) (419) : 0.944809

(424) (491) : 0.944809

(424) (526) : 0.944262

(424) (218 335) : 0.944809

(424) (383 491) : 0.944809

(454) (526) : 0.920765

(494) (218) : 0.944809

(494) (335) : 0.944809

(494) (383) : 0.944809

(494) (419) : 0.944809

(494) (491) : 0.944809

(494) (526) : 0.944262

(494) (218 335) : 0.944809

(494) (383 491) : 0.944809

(549) (218) : 0.944809

(549) (335) : 0.944809

(549) (419) : 0.944809

(549) (526) : 0.944262

(549) (218 335) : 0.944809

第6章 おわりに

本論文では、ユーザが定義した制限距離（時間）以下のトランザクションは同時に起こったとみなして、トランザクションの結合を行ない、得られたデータベースに対してマイニングを行なう手法を提案した。その結果、従来手法と比較した時、頻出シーケンスの長さについては、最小サポート値 90%で頻出シーケンスの平均長が、提案手法では 2.771、従来手法では 2.469 となった。また、トランザクション数については、最小サポート値 90%で、提案手法では平均トランザクション数が 1.00、従来手法では 1.21 となり、質の異なる頻出シーケンスを抽出することができたといえる。つまり、短期的な行動と長期的な行動を区別してマイニングを行なうことができたといえる。また、抽出時間については、1 シーケンスあたりの抽出時間には変化はないが、全体の抽出時間については最小サポート値 90%で提案手法では 0.12 秒、従来手法では 0.18 秒となり、削減することができた。

今後、提案手法の拡張を考えた場合、長期的行動に着目して新たな処理を行なう方法が考えられる。具体的には、トランザクション間の距離（時間）が、ユーザが定義した制限距離（制限時間）より離れている場合は、相関関係がないと判断し、サポート値にカウントしないようにする。これにより、短期的な行動と長期的な行動を、さらに厳密に区別することが可能になる。そして、ユーザにより詳細な行動の情報を与えることができると考えられる。

さらに、提案手法ならびに拡張手法を実際に実世界の例に応用するということも考えられる。

謝辞

本研究を行なうにあたり、数々のご指導を頂いた山名早人助教授に深く感謝致します。また、様々な面でお世話になった斉田直幸先輩、平手勇宇先輩、並びにユビキタス班の方々に厚く御礼申し上げます。

参考文献

[1] J.Pei, “Frequent Pattern Mining”, http://www.cs.buffalo.edu/faculty/jianpei/

[2] R.Agrawal and R.Srikant. “Mining Sequential Patterns”. In *Proc. of the 11th international conference on Data Engineering*, Mar 1995.

[3] R.Srikant and R.Agrawal ,” Mining Sequential Patterns:Generalizations and Performance Improvements”In *Proc. of 5th Int. Conf. Extending Database Technology,* 1996

[4] J.Pei, J.Han, B.Mortazavi-Asl, H.Pinto, Q.Chen, U.Dayal and M.Hsu, “PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth” In. *Proc. of 2001 Int. Conf. Data Engineering (ICDE'01),* pp. 215-224 , April 2001

[5] M.J.Zaki, “SPADE: An efficient algorithm for mining sequences”,In *Proc. of Machine Learning,42(1/2):31-60,* 2001

[6] Ayres J, Gehrke J, Yu T and Flannick J, “Sequential PAttern Mining using Bitmap Represetation”, In *Proc. of ICDE ,*2001

[7] 福田剛志　森本康彦　徳山豪　“データマイニング”共立出版,2001

[8] D.Chiu, Y.Wu and A.Chen,”An Efficient Algorithm for Mining Frequent Sequences by a New Strategy without Support Counting”In *Proc. of ICDE,* 2004

[9] J. Han, http://www-sal.cs.uiuc.edu/~hanj/

[10] “Musashi 人工データ”, http://musashi.sourceforge.jp/artificialData/index.html

[11] 有村博紀，“ストリームデータマイニングに関する研究動向”, http://www.i.kyushu-u.ac.jp/~arim/ jtalks/arimura-dews04-mini-survey.ppt

[12] T.Asai, K.Abe, S.Kawasoe, H.Arimura, H.Sakamoto and S.Arikawa. “Online Algorithms for Mining Semistructured Data Stream” In *Proc. of DOI Technical Report*, June 2002

[13] Q.Zheng, K.Xu and S.Ma “When to Update the Sequential Patterns of Stream Data” In *Proc. of 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining,*2003