相関分析のためのラティス構造に基づくデータセット生成器

松石 浩輔† 沼尾 雅之‡

†‡電気通信大学 〒182-8585 東京都調布市調布ヶ丘 1-5-1

E-mail: †kosuke.matsuishi@uec.ac.jp, ‡numao@cs.uec.ac.jp

あらまし 相関分析の分野では，マイニング手法の評価のため，しばしば人工データセット生成器が利用される．しかし，事実上の標準として多用されている IBM の生成器 [1]は，その出力が“実データらしくない”との指摘がある [2] [3]．また既存の生成器は，出力に現れる頻出アイテム集合や相関ルールを指定できない．そこで本研究では，頻出パターンを入力し，それに従うデータセットを出力する生成器を提案する．提案手法は，トランザクションを生成しながら，生成済みのデータセットの頻出パターンを随時更新し，さらに必要に応じて生成済みのトランザクションをマージするものである．評価実験の結果，パラメータによっては提案手法により実データらしい出力が得られるが，提案手法の一部には課題があることも明らかとなった．

キーワード 相関分析，バスケット分析，データ生成，実データらしさ，頻出パターン

1. はじめに

相関分析 (バスケット分析) は，多数のバスケットデータから頻出アイテム集合や相関ルールを抽出するデータマイニング手法であり，主に小売業の POS データ分析に利用されている．

Apriori [1]に代表される相関分析のマイニング手法は，これまでによく研究されてきた．このマイニング手法を評価する際，大量の実データ収集は困難であるため，しばしば人工データセットが利用される [3]．ところが，この人工データセットの生成器については，マイニング手法に比べさほど研究がなされていない．とくに現在提案されている生成器では，出力データセットに表れる頻出アイテム集合や相関ルールを指定できない．そこで本研究では，頻出アイテム集合を入力し，そのような特徴をもつデータセットを出力する手法を提案する．すなわち提案手法は，マイニング手法の入力と出力を入れ替えたものである (図 1)．また本研究では，実データらしいデータを生成できるよう，その処理を工夫する．

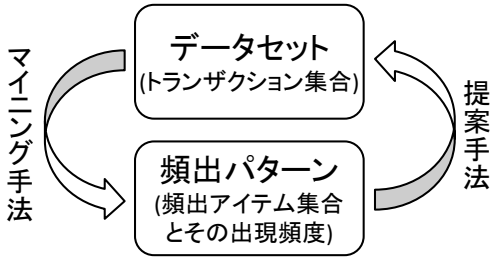

図 1 提案手法の位置づけ

2. 関連研究

ここでは関連研究として，既存のデータセット生成器と，データセットの“実データらしさ”の指標について述べる．

2.1. 既存のデータセット生成器

(1) IBM の生成器

IBM の R. Agrawal らは， POS データをモデル化した生成器を提案している [1]．この生成器は，しばしば同時に購入されるアイテムの集合を「潜在的ラージアイテム集合」として事前に生成し，そこから各トランザクションにアイテムを割り振る．

IBM の生成器は，相関分析の分野においてしばしば利用されており，特にこの生成器による 2 つの人工データセット `T10I4D100K` および `T40I10D100K` がそのままマイニング手法の評価に利用されることが多い．ところがこれらの人工データセットは，2.2 節で述べる観点から実データらしくないとの指摘がある [2] [3]．

(2) CoZi Generator

C. Cooper らは，論文の引用ネットワークをモデル化した生成器を提案している [2]．この生成器は，ネットワークの成長モデルである優先的選択モデル [4]を利用することで，冪乗則に従う実データらしいデータセットを出力するとされる．

2.2. 実データらしさの指標

2.2.1. トランザクションの大きさの分布

Z. Zheng らによると，IBM の生成器によるトランザクションは実データのトランザクションに比べ，平均的により多くのアイテムを含む [3]．

2.2.2. アイテムの出現数の分布

実世界のさまざまな現象は，冪乗則に従うことが知られている [5]．C. Cooper らによると，相関分析の分野においても，POS データや Web サイトのアクセス数といった実データにおけるアイテムの出現数の分布は冪乗則に従うが，IBM 生成器によるデータは冪乗則に従わない [3]．

3. 提案手法

既存のデータセット生成器は，出力に現れる具体的な頻出アイテム集合や相関ルールを指定できない．この指定が可能となれば，データセットから頻出アイテム集合を抽出するマイニング手法の逆操作が可能となる (図 1)．この逆操作は，マイニング手法の検証に利用できるほか，実データからパラメータを取り出し，それに類似の人工データを大量に生成するなど，相関分析の研究に応用できるものと考えられる．

そこで本研究では，具体的な頻出アイテム集合とその出現数のマップを入力し，それに合わせたデータセットを出力する生成器を提案する．提案手法は，アイテム集合がラティス構造をなすことに着目して生成を行う．また，実データらしい出力を得るため，アイテムの出現数の分布を入力により制御する．

3.1. 提案手法の入出力

提案手法は，表 1 に示す各項目を入力し，それを満たすデータセットを出力するものである．

表 1 提案手法の入力

パラメータ名	説明
pattern	入力頻出パターン (頻出アイテム集合とその出現数とのマップ)
minSupp	上記 pattern を抽出する際に用いた最低出現数 (サポート)
nTrans	トランザクション数
nItems	アイテムの種類数
aveTransSize	トランザクションの平均サイズ

3.2. 提案手法の処理

提案手法の擬似コードを表 2 に示す．提案手法は，頻出アイテムセットを生成する処理 freqGen()と，頻出しないアイテムをノイズとして加える処理 scatter()との 2 部から構成される．

freqGen()では，入力された頻出アイテム集合とその出現数を，アイテム集合の要素数の大きい順に 1 組ずつ取り出して処理する．また，これまでに生成したデータセットの頻出パターンを check に保持する．ある頻出アイテム集合 itemset とその出現数 count を処理するには，はじめに itemset のみからなるトランザクションを count-check[itemset]個生成する (22 行目)．さらに，itemset の全ての部分集合 s について，check[s]の値に count-check[itemset]を加える (23〜25 行目)．以上の処理により，入力に共通部分のあるアイテム集合同士が含まれていても，この共通部分のアイテム集合を必要以上に出力することを防ぐ．なお 18〜19 行目のマージ処理については，3.3 節で説明する．

表 2 提案手法の擬似コード

```
1   latticeBasedGen(pattern, minSupp,
2       nTrans, nItems, aveTransSize) {
3     datasetを初期化
4     pattern[NULL] <- nTrans
5     freqGen(dataset, pattern, minSupp)
6     scatter(dataset, pattern, minSupp,
7         nItems, aveTransSize)
8     return dataset
9   }
10
11  freqGen(dataset, pattern, minSupp) {
12    check, nMergeを初期化
13    foreach (itemset, count in pattern) {
14      # itemsetとcountは，length(itemset)降順
15      # に取り出す
16      if (count < check[itemset]) {
17        merge(child, pattern, dataset,
18            check, minSupp, nMerge)
19      } else if (check[itemset] < count &&
20          itemset != NULL) {
21        i <- count - check[itemset]
22        datasetにitemsetをi個追加
23        foreach (s in itemsetの部分集合) {
24          check[s] <- check[s] + i
25        }
26      }
27    }
28  }
29
30  scatter(dataset, pattern, minSupp, nItems,
31        aveTransSize) {
32    infreqItems <- patternに含まれないアイテム
33    infreqDist <- zipfEstimate(dataset,
34        infreqItems, minSupp, aveTransSize)
35    infreqPairs <- infreqItemsとinfreqDist
36        とをランダムにマッチさせたマップ
37    foreach (item, count in infreqPairs) {
38      dataset中のcount個の任意のトランザクション
39        にitemを追加
40    }
41  }
```

scatter()は，頻出しないアイテム (すなわち出現数が minSupp 未満となるアイテム) を，各アイテムが独立であると仮定してランダムなトランザクションに加える処理である．提案手法では，頻出しないアイテムの出現数の分布を，離散冪乗則である Zipf の分布 [5] により推定する (33〜34 行目)．この推定を行う zipfEstimate()関数は，単純な反復法により Zipf の分布のパラメータを推定するものであり，これ以上の説明は本稿では省略する．

3.3. 頻出アイテム集合のマージ処理

3.2 節で述べた処理のみで予備実験を行ったところ，パラメータ minSupp が 1 より大きいときに入力 pattern を満たさない出力が生成されうることがわかった．本節ではこの問題の原因と，その解決策であるマージ処理について述べる．

3.3.1. マージ処理が必要となる例

いま，アイテム集合 {A,B,C} の出現数が 1，{A,B} の出現数が 3，{B,C} の出現数が 2，{B} の出現数が 4 となるようなデータセットを考えると，このデータセットは図 2 に示すラティス構造をなす．ここから頻出パターンを抽出し，マージ処理のない提案手法に入力したときの動作を以下で考える．

はじめに minSupp=1 として前述のデータセットから頻出パターンを抽出すると，{A,B,C}:1, {A,B}:3, {B,C}:2, {B}:4 が取り出される．この頻出パターンを入力として freqGen()の処理を行うと，表 3 (a)のようなトランザクションが生成される．

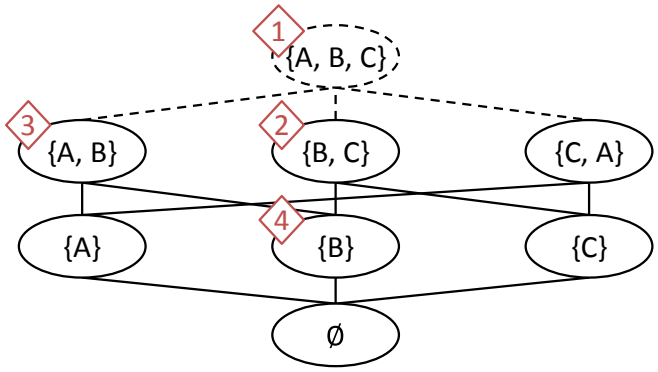

図 2 マージ処理が必要となる入力例

表 3 minSupp の変化による freqGen()の出力の変化 (マージ処理を行わない場合)

(a) minSupp=1	(b) minSupp=2
{A,B,C} {A,B} {A,B} {B,C}	{A,B} {A,B} {A,B} {B,C} {B,C}

同様に minSupp=2 の場合を考えると，前述のデータセットからは頻出パターン {A,B}:3, {B,C}:2, {B}:4 が取り出される ({A,B,C}:1 は出現数が minSupp=1 未満であるため含まれない)．この頻出パターンを入力として freqGen()に与えると，freqGen()ははじめに入力から {A,B}:3, {B,C}:2 を取り出し，{A,B} および {B,C} というトランザクションをそれぞれ 3 個，2 個ずつ出力する．このとき {B} の check の値は 5 となる．続いて {B}:4 を処理しようとするが，ここで {B} の check の値 5 が入力 4 よりも大きい．すなわち {B} を含むアイテムセットが，入力で指定された数よりも多く生成されたことになる．このときの出力 (表 3 (b)) において，{B} を含むトランザクションは 4 件であるべきところ 5 件出力されており，またトランザクションの総数も (a)と比べ 1 件多くなっている．

3.3.2. マージ処理

以上の現象が起こるのは，3.2 節で述べた提案手法が入力 pattern のうち，アイテム集合のラティス構造における頂点同士 (前節の例では {A,B} と {B,C}) を排他的に処理し，その和集合を出力しないためである．これを解決するには，check の値が入力より大きかった場合 (表 2 の 17 行目) に，現在処理中の itemset の上位集合同士をマージして，その出現数を削減すればよい．この処理を表 4 に示す．

merge()は，はじめに現在処理中の，すなわち出現数を減らしたいアイテム集合 tooMany の真上位集合 (tooMany 自身を含まないことに注意) から，マージできる全てのペアを探す (6〜10 行目)．マージできるペアの判定には isMergeable()を利用する．この関数は，アイテム集合 a および b と，これらをマージしたアイテム集合 merged，このマージ処理によって出現数を減らしたいアイテム集合 tooMany が以下の全ての性質を満たすときに真を返す．

- a と b の共通部分が tooMany と等しい
- merged のみからなるトランザクションを 1 件生成したとき，その出現数が pattern の指定以下となるか，または minSupp 未満となる
- merged の累積マージ回数 nMerge[a]+nMerge[b]+1 が，現在のマージ回数上限 curMergeLim より少ない

マージできる全てのペアを列挙したら，各ペア<a,b>の pattern における出現数の和 pattern[a]+pattern[b]を重みとしてルーレット選択を行う．そして選ばれたペアを mergePair()によりマージし，さらにマージ後の各アイテム集合の出現数に合わせて check を更新する．

また merge()は，各アイテム集合の累積マージ回数を nMerge に記憶しており，マージ処理を行うごとにこの値を更新する (43 行目)．そして，マージによって極端に大きなアイテム集合が生成されることを防ぐため，マージするペアの探索には，マージ累積回数の上限値 curMergeLim を用いる．この値を 1 ずつ増やしながらペアを探し，その上限値 nMergeLim まで増やしてもペアが見つからなかった場合は，マージ処理を断念する．

以上のマージ処理を 3.3.1 節の例に適用すると，アイテム集合 {B} を tooMany として 1 度だけマージ処理が行われる．この結果，{A,B} と {B,C} がマージされて {A,B,C} が生成され，表 3 (a)のデータセットが出力される (これは入力頻出パターン (図 2) を満たす)．

なお，表 4 の処理をそのまま実装すると，マージを 1 度行うごとにマージ可能なペアを列挙し直す (6〜10

表 4 提案手法の擬似コード (マージ処理)

```
1   merge(tooMany, pattern, dataset, check,
2       minSupp, nMerge) {
3     nMergeLim <- 10 # マージ回数の上限値
4     curMergeLim <- 1 # 現在のマージ回数上限
5     while (count < check[tooMany]) {
6       supersets <- tooManyの真上位集合かつ
7           datasetに存在するトランザクションの集合
8       mergeablePairs <-
9           supersetsとsupersetsとの直積<a,b>の
10          うちisMergeable(a, b, …)を満たすもの
11      if (mergeablePairsが存在しない) {
12        if (curMergeLim < nMergeLim) {
13          curMergeLim++
14          continue # マージ上限を増やしてリトライ
15        } else break # マージ不可能
16      }
17      pair <- mergeablePairsの要素<a,b>から
18          pattern[a]+pattern[b]を重みとして
19          ルーレット選択
20      mergePair(pair.a, pair.b, tooMany,
21          dataset, count, nMerge)
22    }
23  }
24
25  isMergeable(a, b, tooMany, pattern,
26      check, minSupp, nMerge, curMergeLim) {
27    merged <- aとbの和集合
28    p <- pattern[merged]
29    c <- check[merged]
30    if (aとbの共通部分 == tooMany &&
31        (c < p || c + 1 < minSupp) &&
32        nMerge[a] + nMerge[b] + 1 < curMergeLim) {
33      return true
34    }
35    return false
36  }
37
38  mergePair(a, b, tooMany, dataset,
39      check, nMerge) {
40    merged <- aとbの和集合
41    datasetにmergedを1件追加
42    datasetからaとbを各1件削除
43    nMerge[merged] <- nMerge[a]+nMerge[b]+1
44    foreach (s in mergedの部分集合) {
45      if (sがtooManyの部分集合) {
46        check[tooMany]--
47      } else if (sがaまたはbの部分集合でない) {
48        check[s]++
49      }
50    }
51  }
```

行目) ため，処理時間がきわめて長くなる．これを解決するには，各 tooMany に対するマージ可能なペアの列挙を 1 度だけ行い，その後はマージを行うごとに，そのマージにより影響を受けるアイテム集合ペアのみについて，マージ可能性の再判定 (isMergeable()の処理) を行えばよい．4 章の実験では，この改善を行った実装を用いた結果，処理速度が約 40 倍に向上した．

また提案手法は，最終的に生成されるトランザクション数を，check における空集合 NULL の値によって制御している．このため，本稿の説明における「部分集合」には空集合も含まれる点に注意が必要である．たとえば mergePair()の foreach ループは，空集合に関しても内部の処理を行う．

4. 評価実験

提案手法の評価のため，実データから抽出したパラメータを，提案手法および 2.1 節で挙げた既存手法に入力し，人工データセットを生成した．これと元の実データとを比較し，各手法の出力の実データらしさを評価した．

4.1. パラメータ

はじめに，家電量販店における販売履歴の実データセット BMS-POS [3]より，ランダムに 1000 件のトランザクションを取り出して実験用の元データとした (以下，元データと記す)．この元データから，IBM の生成器，CoZi Generator および提案手法のパラメータを抽出した．このうち提案手法のパラメータを表 5 に示す．ただしパラメータ pattern に関しては，minSupp の値に応じて異なる頻出パターンを利用した (表 6)．

表 5 実験に用いた提案手法のパラメータ

項目	値
pattern	元データより Eclat [6]を用いて抽出
minSupp	10, 20, 50
nTrans	1000 (元データのトランザクション数)
nItems	580 (元データのアイテム種類数)

表 6 minSupp の値による pattern の変化

minSupp	pattern の件数	pattern の平均長
10	1668	2.820
20	385	2.294
50	58	1.638

4.2. 実験結果

元データおよび各人工データの概要を表 7 に示す．また，2.2 節で示した実データらしさの指標により，各データセットを比較したグラフを図 3 および図 4 に示す．

表 7 各データセットの概要

データセット	処理時間 [sec]	トランザクション数	アイテム数	トランザクションの平均サイズ
元データ	n/a	1000	580	6.44
minSupp=10	28.0	1000	580	7.90
minSupp=20	7.14	1000	580	6.44
minSupp=50	2.84	1000	580	6.44
IBM	0.35	1000	309	7.19
CoZi	0.36	1000	105	6.74

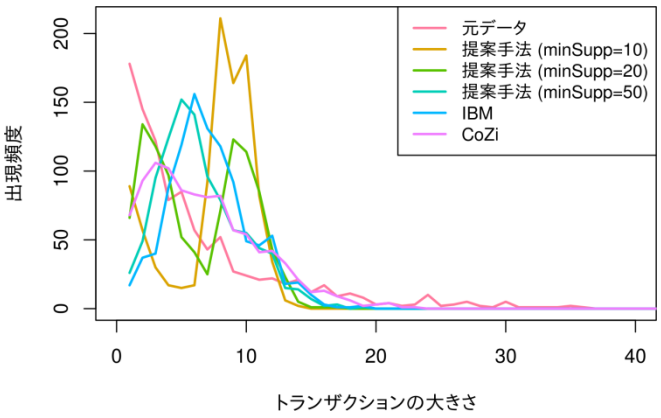

図 3 トランザクションの大きさの分布

4.3. 考察

4.3.1. 動作の検証

はじめに，提案手法による人工データセットから，各 minSupp 値をパラメータとして Eclat により頻出アイテム集合を抽出したところ，それぞれの入力 pattern とまったく同一となった．このため，提案手法とその実装が，正しく動作したといえる．

4.3.2. トランザクションの大きさの分布

図 3 より，元データにおいては大きさ 1 のトランザクションが最も頻出する一方，大きさ 20 以上のトランザクションもいくつか存在することがわかる．これに対し各人工データでは，大きさ 3 から 10 程度のトランザクションが頻出しており，この点においていずれも元データとは異なる傾向を示すことがわかった．

とくに提案手法において minSupp=10 としたときの分布は，元データおよび他の人工データともかけ離れた不自然な分布となった．これは，提案手法において minSupp を小さくすると，マージ回数が増えるだけでなく，マージの結果より大きなトランザクションが生成されるようになり，トランザクションの大きさの分布が偏ってしまうためである (表 8)．

この問題の解決は今後の課題であるが，その方法として，マージするトランザクションを選択する基準を変更することが考えられる．提案手法は，マージするトランザクションの出現頻度の和を重みとして確率的にマージを行っている．この重みを変更し，マージ処理により大きなトランザクションが生成される確率を下げることで，この問題を緩和できる可能性がある．

表 8 tooMany の大きさごとのマージ処理発生回数

大きさ / minSupp	0	1	2	3	4	計
10	250	986	1258	1067	505	4066
20	124	742	1303	381	0	2550
50	397	587	70	0	0	1054

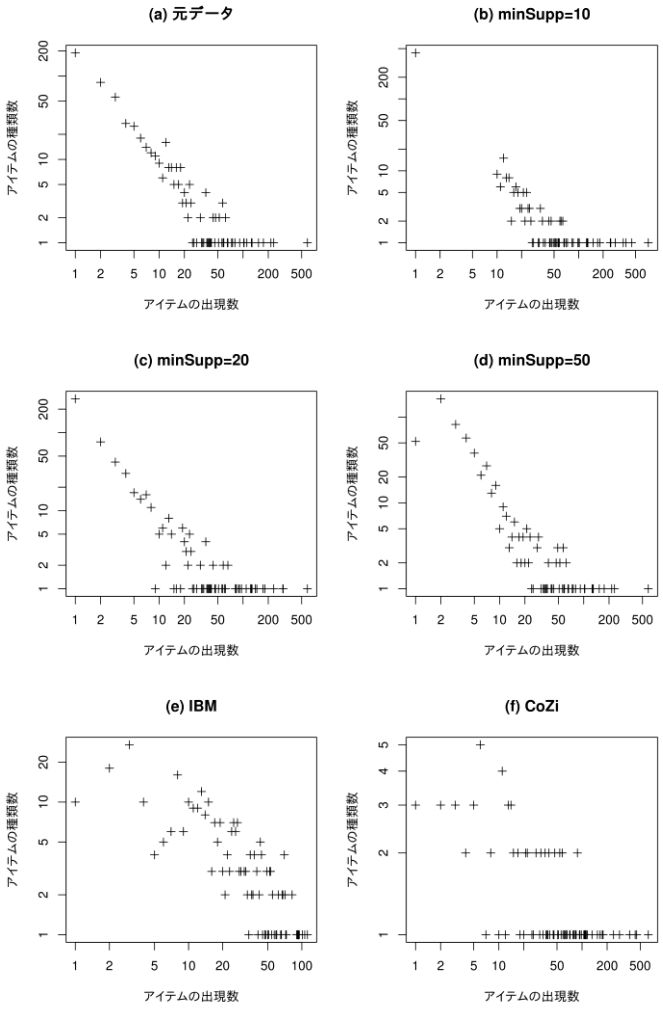

図 4 アイテムの出現数の分布

4.3.3. アイテム出現数の分布

図 4 より，アイテム出現数の分布について，既存手法による人工データは元データと異なり，冪乗則とはやや異なる分布となった．これに対し，提案手法による人工データは，元データとおおむね同様の分布となった．つまり，分布の概形のみを見ると，提案手法は既存手法と比べ，より実データらしい出力を得られるように見える．ただし，各 minSupp 値よりも横軸の大きな範囲については，入力頻出パターンにより分布を与えられる範囲であり，人工データにおいて元データを再現できることは当然であると言える．また，各 minSupp 値よりも横軸の小さな範囲を見ると，提案手法においても改善すべき点があることがわかる．これについて以下で述べる．

はじめに minSupp=10 の場合について，図 4 (b)より，出現数が 2 から 9 のアイテムが存在しないことがわかる．この原因は，freqGen()の処理を終えた時点 (頻出アイテムのみを出力した時点) でのトランザクションの平均の大きさが，すでにパラメータ aveTransSize

を超えていたためである．このため，頻出しないアイテムの出現数の分布推定 (zipfEstimate()関数の処理) に失敗し，頻出しない 437 種類のアイテムは全て 1 件ずつ出力され，図 4 (b)における左上の点 (1, 437) となった．この現象への改善策としては，マージ処理をさらに追加で行い，freqGen()の処理が終わった時点での出力トランザクションに，ある程度の個数の空集合が存在するようにすればよいと考えられる．

次に minSupp=50 について，図 4 (d)より，出現数 1 のアイテムが出現数 2 のアイテムに比べ半分未満しか存在しないことがわかる．これは zipfEstimate()関数において，アイテム出現数の分布の推定結果が，どのアイテムも 2 件以上出現するような分布であったためである．ところが冪乗則では，出現数の少ないアイテムほど多数 (多種類) 存在すべきであり，この観点からは zipfEstimate()関数の推定結果は不自然であるといえる．実験に用いた zipfEstimate()関数は，パラメータ aveTransSize を可能な限り厳密に守るものとなっているが，この制限を緩和し，出現数 1 のアイテムが多数出力されるような分布を推定することで，この不自然さを解決することができる．

最後に minSupp=20 (図 4 (c)) については，元データ (図 4 (a)) とほぼ同じ分布が得られており，実データらしい良好な人工データを生成できたといえる．

5. おわりに

5.1. まとめ

本研究では，頻出アイテム集合とその出現数を入力し，それに従うデータセットを出力するデータセット生成器を提案した．これは，既存の相関分析手法の逆操作に位置づけられる．評価実験として，実データからパラメータを抽出し，これを既存手法および提案手法に入力して，出力の実データらしさを比較した．この結果，提案手法では，パラメータの指定によっては実データらしい出力が得られるが，マージ処理および頻出しないアイテムの出現数の分布推定処理に課題があることが明らかとなった．

5.2. 今後の課題

今後の課題としては，出力におけるトランザクションの大きさの分布を，より元データに近い自然なものとする方法の検討が挙げられる．この方法としては，前述の通り，マージするアイテム集合ペアの選択方法およびアイテム出現数の分布推定処理を改善することが考えられる．

参 考 文 献

[1] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," *VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487-499, 1994.

[2] C. Cooper and M. Zito, "Realistic Synthetic Data for Testing Association Rule Mining Algorithms for Market Basket Databases," *PKDD 2007 Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 398-405, 2007.

[3] Z. Zheng, R. Kohavi and L. Mason, "Real World Performance of Association Rule Algorithms," *KDD '01 Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 401-406, 2001.

[4] A.-L. Barabási and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509-512, 1999.

[5] E. J. M. Newman, "Power Laws, Pareto Distributions and Zipf's Law," *Contemporary Physics*, vol. 46, pp. 323-351, 2005.

[6] M. J. Zaki, "Scalable Algorithms for Association Mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 3, pp. 372-390, 2000.

[7] M. Hahsler, K. Hornik and T. Reutterer, "Implications of Probabilistic Data Modeling for Mining Association Rules," *From Data and Information Analysis to Knowledge Engineering*, pp. 598-605, 2006.